Proven COMPUTER Performance

SYSTEMS

A Subsidiary of GOULD INC.

REFERENCE MANUAL

SYSTEMS 32/70 SERIES

Computer

Supersedes 301-320070-000

• Publication Number 301-320070-001

32/70 SERIES INSTRUCTIONS BY CATEGORY

OP CODE	MNEMONIC	DESCRIPTION	PAGE
		LOAD INSTRUCTIONS	
AC08	LB	LOAD BYTE	6-10
AC00	LH	LOAD HALFWORD	6-11
AC00	LW	LOAD WORD	6-12
AC00	LD	LOAD DOUBLEWORD	6-13
B008	LMB	LOAD MASKED BYTE	6-14
B000	LMH	LOAD MASKED HALFWORD	6-15
B000	LMW	LOAD MASKED WORD	6-16
B000	LMD	LOAD MASKED DOUBLEWORD	6-17
B408	LNB	LOAD NEGATIVE BYTE	6-18
B400	LNH	LOAD NEGATIVE HALFWORD	6-19
B400	LNW	LOAD NEGATIVE WORD	6-20
B400	LND	LOAD NEGATIVE DOUBLEWORD	6-21
C800	LI	LOAD IMMEDIATE	6-22
D000	LEA	LOAD EFFECTIVE ADDRESS	6-23
8000	LEAR	LOAD EFFECTIVE ADDRESS REAL*	6-24
3400	LA	LOAD ADDRESS*	6-25
CC00	LF	LOAD FILE	6-28
		STORE INSTRUCTIONS	
D408	STB	STORE BYTE	6-29
D400	STH	STORE HALFWORD	6-30
D400	STW	STORE WORD	6-31
D400	STD	STORE DOUBLEWORD	6-32
D808	STMB	STORE MASKED BYTE	6-33
D800	STMH	STORE MASKED HALFWORD	6-34
D800	STMW	STORE MASKED WORD	6-35
D800	STMD	STORE MASKED DOUBLEWORD	6-36
DC00	STF	STORE FILE	6-37
		ZERO INSTRUCTIONS	
F808	ZMB	ZERO MEMORY BYTE	6-39
F800	ZMH	ZERO MEMORY HALFWORD	6-40
F800	ZMW	ZERO MEMORY WORD	6-41
F800	ZMD	ZERO MEMORY DOUBLEWORD	6-42
0C00	ZR	ZERO REGISTER	6-43
		TRANSFER INSTRUCTIONS	
2C0F	TSCR	TRANSFER SCRATCHPAD TO REGISTER	6-45
2C0E	TRSC	TRANSFER REGISTER TO SCRATCHPAD	6-46
2C00	TRR	TRANSFER REGISTER TO REGISTER	6-47
2C08	TRRM	TRANSFER REGISTER TO REGISTER MASKED	6-48
FB00	TRP	TRANSFER REGISTER TO PROTECT REGISTER	6-49
FB00	TPR	TRANSFER PROTECT REGISTER TO REGISTER	6-50
2C04	TRN	TRANSFER REGISTER NEGATIVE	6-51
2C0C	TRNM	TRANSFER REGISTER NEGATIVE MASKED	6-52
2C03	TRC	TRANSFER REGISTER COMPLEMENT	6-53
2C0B	TRCM	TRANSFER REGISTER COMPLEMENT MASKED	6-54
2C05	XCR	EXCHANGE REGISTERS	6-55
2C0D	XCRM	EXCHANGE REGISTERS MASKED	6-56
2800	TRSW	TRANSFER REGISTER TO PSWR	6-57
		MEMORY MANAGEMENT INSTRUCTIONS	
000D	SEA	SET EXTENDED ADDRESSING	6-59
000F	CEA	CLEAR EXTENDED ADDRESSING	6-60
2A07	LMAP	LOAD MAP*	6-61
2C0A	TMAPR	TRANSFER MAP TO REGISTER*	6-62
		WRITABLE CONTROL STORAGE INSTRUCTIONS	
000C	WWCS	WRITE WRITABLE CONTROL STORAGE	6-65
0008	RWCS	READ WRITABLE CONTROL STORAGE	6-66
FA00	JWCS	JUMP TO WRITABLE CONTROL STORAGE	6-67
		BRANCH INSTRUCTIONS	
EC00	BU	BRANCH UNCONDITIONALLY	6-72
F000	BCF	BRANCH CONDITION FALSE	6-73
EC00	BCT	BRANCH CONDITION TRUE	6-74
F000	BFT	BRANCH FUNCTION TRUE	6-75
F880	BL	BRANCH AND LINK	6-76
F400	BIB	BRANCH AFTER INCREMENTING BYTE	6-77
F420	BIH	BRANCH AFTER INCREMENTING HALFWORD	6-78
F440	BIW	BRANCH AFTER INCREMENTING WORD	6-79
F460	BID	BRANCH AFTER INCREMENTING DOUBLEWORD	6-80
		COMPARE INSTRUCTIONS	
9008	CAMB	COMPARE ARITHMETIC WITH MEMORY BYTE	6-83
9000	CAMH	COMPARE ARITHMETIC WITH MEMORY HALFWORD	6-84
9000	CAMW	COMPARE ARITHMETIC WITH MEMORY WORD	6-85
9000	CAMD	COMPARE ARITHMETIC WITH MEMORY DOUBLEWORD	6-86
1000	CAR	COMPARE ARITHMETIC WITH REGISTER	6-87
C805	CI	COMPARE IMMEDIATE	6-88
9408	CMMB	COMPARE MASKED WITH MEMORY BYTE	6-89
9400	CMMH	COMPARE MASKED WITH MEMORY HALFWORD	6-90
9400	CMMW	COMPARE MASKED WITH MEMORY WORD	6-91
9400	CMMD	COMPARE MASKED WITH MEMORY DOUBLEWORD	6-92
1400	CMR	COMPARE MASKED WITH REGISTER	6-93
		LOGICAL AND INSTRUCTIONS	
8408	ANMB	AND MEMORY BYTE	6-95
8400	ANMH	AND MEMORY HALFWORD	6-96
8400	ANMW	AND MEMORY WORD	6-97
8400	ANMD	AND MEMORY DOUBLEWORD	6-98
0400	ANR	AND REGISTER AND REGISTER	6-99
		LOGICAL OR INSTRUCTIONS	
8808	ORMB	OR MEMORY BYTE	6-100
8800	ORMH	OR MEMORY HALFWORD	6-101
8800	ORMW	OR MEMORY WORD	6-102
8800	ORMD	OR MEMORY DOUBLEWORD	6-103
0800	ORR	OR REGISTER AND REGISTER	6-104
0808	ORRM	OR REGISTER AND REGISTER MASKED	6-105
		LOGICAL EXCLUSIVE OR INSTRUCTIONS	
8C08	EOMB	EXCLUSIVE OR MEMORY BYTE	6-106
8C00	EOMH	EXCLUSIVE OR MEMORY HALFWORD	6-107
8C00	EOMW	EXCLUSIVE OR MEMORY WORD	6-108
8C00	EOMD	EXCLUSIVE OR MEMORY DOUBLEWORD	6-109
0C00	EOR	EXCLUSIVE OR REGISTER AND REGISTER	6-110
0C08	EORM	EXCLUSIVE OR REGISTER AND REGISTER MASKED	6-111

OP CODE	MNEMONIC	DESCRIPTION	PAGE
		SHIFT INSTRUCTIONS	
6000	NOR	NORMALIZE	6-113
6400	NORD	NORMALIZE DOUBLE	6-114
6800	SCZ	SHIFT AND COUNT ZEROS	6-115
6C40	SLA	SHIFT LEFT ARITHMETIC	6-116
7040	SLL	SHIFT LEFT LOGICAL	6-117
[illegible]	SLC	SHIFT LEFT CIRCULAR	6-118
7840	SLAD	SHIFT LEFT ARITHMETIC DOUBLE	6-119
7C40	SLLD	SHIFT LEFT LOGICAL DOUBLE	6-120
6C00	SRA	SHIFT RIGHT ARITHMETIC	6-121
7000	SRL	SHIFT RIGHT LOGICAL	6-122
7400	SRC	SHIFT RIGHT CIRCULAR	6-123
7800	SRAD	SHIFT RIGHT ARITHMETIC DOUBLE	6-124
7C00	SRLD	SHIFT RIGHT LOGICAL DOUBLE	6-125
		BIT MANIPULATION INSTRUCTIONS	
9808	SBM	SET BIT IN MEMORY	6-128
1800	SBR	SET BIT IN REGISTER	6-129
9C08	ZBM	ZERO BIT IN MEMORY	6-130
1C00	ZBR	ZERO BIT IN REGISTER	6-131
A008	ABM	ADD BIT IN MEMORY	6-132
2000	ABR	ADD BIT IN REGISTER	6-133
A408	TBM	TEST BIT IN MEMORY	6-134
2400	TBR	TEST BIT IN REGISTER	6-135
		FIXED-POINT ARITHMETIC INSTRUCTIONS	
8808	ADMB	ADD MEMORY BYTE	6-140
8800	ADMH	ADD MEMORY HALFWORD	6-141
8800	ADMW	ADD MEMORY WORD	6-142
8800	ADMD	ADD MEMORY DOUBLEWORD	6-143
3800	ADR	ADD REGISTER TO REGISTER	6-144
3808	ADRM	ADD REGISTER TO REGISTER MASKED	6-145
E808	ARMB	ADD REGISTER TO MEMORY BYTE	6-146
E800	ARMH	ADD REGISTER TO MEMORY HALFWORD	6-147
E800	ARMW	ADD REGISTER TO MEMORY WORD	6-148
E800	ARMD	ADD REGISTER TO MEMORY DOUBLEWORD	6-149
C301	ADI	ADD IMMEDIATE	6-150
8C08	SUMB	SUBTRACT MEMORY BYTE	6-151
8C00	SUMH	SUBTRACT MEMORY HALFWORD	6-152
8C00	SUMW	SUBTRACT MEMORY WORD	6-153
8C00	SUMD	SUBTRACT MEMORY DOUBLEWORD	6-154
3C00	SUR	SUBTRACT REGISTER FROM REGISTER	6-155
3C08	SURM	SUBTRACT REGISTER FROM REGISTER MASKED	6-156
C802	SUI	SUBTRACT IMMEDIATE	6-157
C008	MPMB	MULTIPLY BY MEMORY BYTE	6-158
C000	MPMH	MULTIPLY BY MEMORY HALFWORD	6-159
C000	MPMW	MULTIPLY BY MEMORY WORD	6-160
4000	MPR	MULTIPLY REGISTER BY REGISTER	6-161
C803	MPI	MULTIPLY IMMEDIATE	6-162
C408	DVMB	DIVIDE BY MEMORY BYTE	6-163
C400	DVMH	DIVIDE BY MEMORY HALFWORD	6-164
C400	DVMW	DIVIDE BY MEMORY WORD	6-165
4400	DVR	DIVIDE REGISTER BY REGISTER	6-166
C804	DVI	DIVIDE IMMEDIATE	6-167
0004	ES	EXTEND SIGN	6-168
0005	RND	ROUND REGISTER	6-169
		FLOATING-POINT ARITHMETIC INSTRUCTIONS	
E008	ADFW	ADD FLOATING-POINT WORD	6-172
E008	ADFD	ADD FLOATING-POINT DOUBLEWORD	6-173
E000	SUFW	SUBTRACT FLOATING-POINT WORD	6-174
E000	SUFD	SUBTRACT FLOATING-POINT DOUBLEWORD	6-175
E408	MPFW	MULTIPLY FLOATING-POINT WORD	6-176
E408	MPFD	MULTIPLY FLOATING-POINT DOUBLEWORD	6-177
E400	DVFW	DIVIDE FLOATING-POINT WORD	6-178
E400	DVFD	DIVIDE FLOATING-POINT DOUBLEWORD	6-179
		CONTROL INSTRUCTIONS	
F900	BRI	BRANCH AND RESET INTERRUPT	6-181
F980	LPSD	LOAD PROGRAM STATUS DOUBLEWORD*	6-182
FA80	LPSDCM	LOAD PROGRAM STATUS DOUBLEWORD AND CHANGE MAP*	6-183
0003	LCS	LOAD CONTROL SWITCHES	6-184
C807	EXR	EXECUTE REGISTER	6-185
C807	EXRR	EXECUTE REGISTER RIGHT	6-186
A800	EXM	EXECUTE MEMORY	6-187
0000	HALT	HALT	6-188
0001	WAIT	WAIT	6-189
0002	NOP	NO OPERATION	6-190
000A	SIPU	Signal IPU	6-191
3000	CALM	CALL MONITOR	6-192
C806	SVC	SUPERVISOR CALL*	6-193
2C09	SETCPU	SET CPU MODE	6-194
0009	RDSTS	READ CPU STATUS WORD*	6-195
0008	EAE	ENABLE ARITHMETIC EXCEPTION TRAP*	6-197
000E	DAE	DISABLE ARITHMETIC EXCEPTION TRAP*	6-198
		INTERRUPT INSTRUCTIONS	
FC00	EI	ENABLE INTERRUPT	6-202
FC02	RI	REQUEST INTERRUPT	6-203
FC03	AI	ACTIVATE INTERRUPT	6-204
FC01	DI	DISABLE INTERRUPT	6-205
FC04	DAI	DEACTIVATE INTERRUPT	6-206
FC77	ACI	ACTIVATE CHANNEL INTERRUPT*	6-207
FC67	ECI	ENABLE CHANNEL INTERRUPT*	6-208
FC6F	DCI	DISABLE CHANNEL INTERRUPT*	6-209
FC7F	DACI	DEACTIVATE CHANNEL INTERRUPT*	6-210
0006	BEI	BLOCK EXTERNAL INTERRUPTS*	6-211
000E	UEI	UNBLOCK EXTERNAL INTERRUPTS*	6-212
		INPUT/OUTPUT INSTRUCTIONS	
FC06	CD	COMMAND DEVICE	6-216
FC05	TD	TEST DEVICE	6-217
FC17	SIO	START I/O*	6-218
FC1F	TIO	TEST I/O*	6-219
FC27	STPIO	STOP I/O*	6-220
FC2F	RSCHNL	RESET CHANNEL*	6-221
FC37	HIO	HALT I/O*	6-222
FC3F	GRIO	GRAB CONTROLLER*	6-223
FC47	RSCTL	RESET CONTROLLER*	6-224
FC4F	ECWCS	ENABLE CHANNEL WCS LOAD*	6-225
FC5F	WCWCS	WRITE CHANNEL WCS*	6-226

*PSD mode instructions only

REVISION INSTRUCTIONS AND MANUAL HISTORY

EQUIPMENT: 32/70 Series Computer **PUBLICATION NO.** 301-320070-001

PURPOSE: This reissue upgrades the manual reflecting the requirements of the Model 2005 Internal Processing Unit (IPU)

REVISION INSTRUCTIONS: Delete and add pages as shown on the following table.

DELETE	ADD
N/A	N/A

NOTE: Revised pages are marked with the Rev. No. in the upper unbound corner. Revised areas are marked with a vertical bar.

MANUAL HISTORY

REV. TYPE	REV. NO.	DATE ISSUED	CONTROL DOC. NO.	ECL
1st Ptg	-	1/79		
R	-	1/80		

REV. TYPE	REV. NO.	DATE ISSUED	CONTROL DOC. NO.	ECL

I = INTERIM REVISION
F = FORMAL REVISION
R = REISSUE
A = ADDENDUM

LIST OF EFFECTIVE PAGES

The total number of pages is in this manual is 398 consisting of the following:

Page	Issue
Title	Original
Instructions	Original
iii through xiv	Original
Frontispiece	Original
1-1 through 1-18	Original
2-1 through 2-32	Original
3-1 through 3-12	Original
4-1 through 4-22	Original
5-1 through 5-22	Original
6-1 through 6-228	Original
7-1 through 7-18	Original
8-1 through 8-6	Original
A-1 through A-6	Original
B-1 through B-8	Original
C-1 through C-2	Original
D-1 through D-2	Original
E-1 through E-2	Original
F-1 through F-2	Original
G-1 through G-2	Original
OP Codes	Original

TABLE OF CONTENTS

SECTION I GENERAL DESCRIPTION

SECTION II CENTRAL PROCESSOR

TABLE OF CONTENTS (Cont'd)

SECTION II CENTRAL PROCESSOR (Cont'd)

SECTION II CENTRAL PROCESSOR (Cont'd)

SECTION III TRAPS AND INTERRUPTS

SECTION IV MEMORY MANAGEMENT

SECTION V INPUT/OUTPUT SYSTEM

SECTION V INPUT/OUTPUT SYSTEM (Cont'd)

SECTION VI INSTRUCTION REPERTOIRE

SECTION VI INSTRUCTION REPERTOIRE (Cont'd)

SECTION VII CONTROL PANEL

SECTION VII CONTROL PANEL (Cont'd)

SECTION VII CONTROL PANEL (Cont'd)

SECTION VIII SYSTEM INITIALIZATION

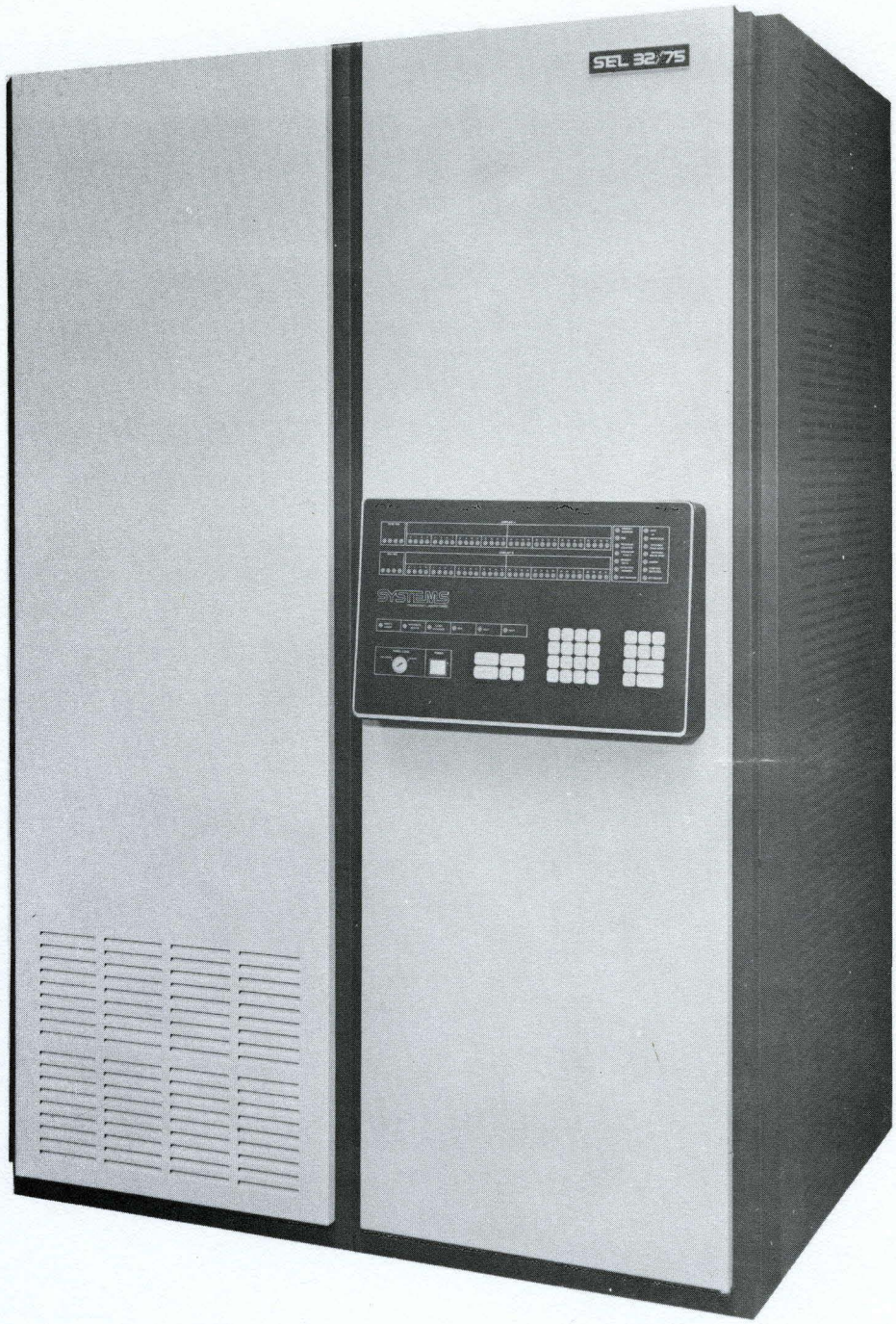
APPENDICES

LIST OF ILLUSTRATIONS

LIST OF TABLES

WARNING

This equipment generates, uses, and can radiate radio frequency energy, and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to subpart J of part 15 of FCC rules, which are designated to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.



32/75 Computer

N6786

SECTION I

GENERAL DESCRIPTION

INTRODUCTION

SYSTEM OVERVIEW

The 32/70 Series computer systems are high-speed, general purpose, digital systems that are designed for a variety of scientific, data acquisition, and real-time applications. A basic system includes a central processor, main memory subsystem, and microprogrammed input/output controllers. Each major system element operates semi-independently with respect to the other elements.

The basic system can be readily expanded to accommodate the user's requirements. Main memory (Core or MOS) has addressing space for 16 million bytes. In a multiprocessor environment, memory can be configured with up to 20 access routes. Input/output capability can be increased by adding more I/O Micro-programmable Processors (IOMs), Regional Processing Units (RPUs), multiplexers, device controllers, and I/O devices.

The CPU has a large instruction set that includes fixed- and floating-point arithmetic instructions. A special lookahead feature enables the CPU to overlap instruction execution with memory accessing, thereby reducing program execution time. A large main memory of up to 16 million bytes (4M words) is available. The memory can consist of up to 16 module increments on each of up to 16 memory buses. Memory can be shared by up to eight CPUs and their associated I/O processors.

Each memory module operates independently of all others and address interleaving can be provided between adjacent modules. This multiaccess memory subsystem with interleaving provides system performance far superior to other design concepts. A 32/70 Series system can support up to 16 independent I/O processors of four types - IOMs, RPUs, multiplexers, and high-speed data interfaces - with a maximum aggregate data transfer rate of up to 16.67 million bytes per second, concurrent with CPU instruction execution.

The existing 32/35 and 32/55 programs can be run on a 32/70 Series computer in the PSW mode. The upward compatibility of the software (assemblers, compilers, mathematical and utility routines, and application packages) virtually eliminates reprogramming.

GENERAL CHARACTERISTICS

All 32/70 Series computer systems contain features and functional characteristics that promote efficient operation in general purpose, multiprocessing, real-time, and multiusage environments.

- Byte-oriented memory (8-bit byte plus one parity bit) which can be addressed and altered as bit, byte (8-bit), halfword (2-byte), word (4-byte), and doubleword (8-byte) quantities.
- 600- or 900-nanosecond core memory.
- 900-nanosecond MOS memory with error checking and correction.

- Both core and MOS memory expandable to 16,777,216 (16M) bytes in some models.
- Indexed addressing capability (PSW or PSD mode with extended addressing) of entire memory.
- Multilevel indirect addressing with indexing at each level.
- Immediate operand instructions for greater storage efficiency and increased speed.
- Eight general purpose registers that may be used for arithmetic, logical, and shift operations, as well as masking, linking, and indexing.
- Hardware memory mapping to reduce memory fragmentation and to provide dynamic program relocation.
- Memory write protection to prevent inadvertent destruction of critical areas of memory.
- Real-time priority interrupt system of up to 112 levels with automatic identification and priority assignment; external interrupt levels which can be individually enabled, disabled and requested by program.
- Automatic traps (for error or fault conditions) that have masking capability and maximum recoverability under program control.
- Power fail-safe for automatic shutdown in the event of power failure and resumption of processing after power is restored.
- Multiple interval timers with a choice of resolutions for independent time bases.
- Privileged instruction logic for program integrity in multiusage environments.
- A complete instruction set that includes the following:
  - Bit, byte, halfword, word, and doubleword operations.
  - Register-to-register operations with halfword instructions to improve program execution time.
  - Fixed-point integer arithmetic operations on byte, halfword, word, and doubleword operands.
  - Floating-point arithmetic operations in single and double precision formats.
  - Full complement of logical operations (AND, OR, Exclusive OR) for bytes, halfwords, words, and doublewords.
  - Comparison operations for bit, byte, halfword, word, and doubleword operands.

- Call Monitor and Supervisory Call instructions that allow a program access to operating system functions.

- Shift operations (left and right) of word or doubleword, including logical, circular, and arithmetic shifts.

- Built-in reliability and maintainability features:

  - Full parity checking of all memory accesses.

  - Address stop feature that permits operator or maintenance personnel to:

    Stop on any instruction address.
    Stop on any memory read reference address.
    Stop on any memory write reference address.

  - CPU traps, which provide for detection of a variety of CPU and system fault conditions, designed to enable a high degree of system recoverability.

- Independently operating I/O system with up to 16 I/O processors per CPU.

- General Purpose Multiplexer Controller (GPMC) that provides for the concurrent operation of up to 16 devices on one I/O processor.

- High-Speed Data interface (HSD) for use with high-speed devices, that allows data transfer rates of up to 3.2 million bytes per second.

- Comprehensive software that is upward program compatible with the 32/35 and 32/55 computers.

  - Expands in capability and speed as system grows.

  - Real-Time Monitor (RTM and Mapped Programming Executive (MPX32)).

  - Language processors that include: Extended FORTRAN IV, ANS COBOL, BASIC, assembler, utilities, and applications software for real-time and scientific users.

- Standard and special purpose peripheral equipment:*

  - Cartridge Disc Units - 10 million byte capacity per unit, peak transfer rate of 312K bytes per second, average access time of 35 milliseconds.

  - Moving-Head Fixed Media Disc - 24 million byte capacity per unit, transfer rates of 1.2 million bytes per second, average access time of 40 milliseconds.

  - Moving-Head Disc - Units available with 40, 80, or 300 million byte per unit capacity, transfer rates of 1.2 million bytes per second, average access time of 30 milliseconds.

- Magnetic Tape Units 9-track, 800/1600 bpi, IBM compatible, high-speed units operating at 75 inches per second with transfer rates up to 120,000 bytes per second; other units operating at 45 inches per second with transfer rates up to 72,000 bytes per second.

- Card Equipment Reading speeds up to 1,000 cards per minute.

- Line Printers Fully buffered with speeds up to 900 lines per minute, 132 print positions with 64 characters.

- Keyboard/Printers 30 characters per second.

- Paper Tape Equipment Readers with speeds up to 300 characters per second, punches with speeds up to 120 characters per second.

- Data Communications Equipment Asynchronous, synchronous, and bisynchronous communications equipment to connect remote user terminals to the computer system via common carrier lines and local terminals directly.

  * Some packaged 32/70 Series systems are restricted in regard to peripherals due to environmental requirements.

STANDARD AND OPTIONAL FEATURES

A basic 32/70 Series System has the following standard features:

- A CPU that includes:

  - Floating-point arithmetic
  - Memory map with access protection
  - Memory write protection
  - Power fail-safe

- Real-Time Option Module that includes:

  - A real-time clock
  - A programmable interval timer
  - Sixteen interrupt levels

- Core or MOS memory (maximum amount and type varies depending on model).

- Teletype, Line Printer, and Card Reader (TLC) controller with three subchannels.

A 32/70 Series system can have the following optional features:

- High-Speed Floating-Point option with up to four times the performance of the standard unit for both single and double precision operands.

- Six additional Real-Time Option Modules
- Writable Control Storage (WCS): up to 4,096 64-bit words.
- An additional 96 external priority interrupts per CPU.
- Up to 13 High-Speed Data interfaces (HSD)
- Up to five General Purpose Multiplexer Controllers (GPMCs).
- Memory shared by up to eight CPUs.
- Up to 16 device controllers with each GPMC.
- Up to 13 user-microprogrammable General Purpose I/O modules (GPIOs) and Regional Processing Units (RPUs).
- Up to 13 high-speed controllers, such as magnetic tape and disc.

GENERAL PURPOSE FEATURES

All 32/70 Series Computer systems include the following general purpose features:

Floating-point instructions are available in both single (32-bit) and double (64-bit) precision formats.

Indirect addressing facilitates table linkages and permits keeping data sections of a program separate from procedure sections for ease of maintenance

The large instruction set (up to 189 instructions in some models) permits short, highly optimized programs to be written that minimize both program space and execution time.

Monitor and Supervisory Call instructions permit access to specified operating system services.

A four-bit condition code simplifies the checking of results by automatically providing information on instruction execution. It includes indicators for arithmetic exception, zero, minus, and plus, as appropriate.

Regional Processing Units (RPU) implement intelligent I/O controllers. Once initialized, an RPU operates independently of the CPU, leaving it free to provide fast response to system needs. The RPU requires minimal interaction with the CPU. Thus, many I/O devices can operate simultaneously without overloading the CPU.

The High-Speed Data Interface (HSD) is a single channel parallel controller that interfaces directly to the SelBUS. Once initiated, I/O operations proceed independently of the CPU. The HSD sustains a data transfer rate of up to three million bytes per second.

Hardware Memory Management of 32/70 Series core or MOS memory - which is available in sizes up to 16 million bytes and provides the needed capacity while assuring the potential for expansion - makes efficient use of available memory. The memory map hardware permits storing a user's program in segments of 8,192 words, wherever space is available. All segments appear as a single, contiguous block of storage at execution time. The memory map also automatically handles dynamic program relocation so the program appears to be stored in a standard way at execution time. Actually, it can be stored in a different set of locations each time it is brought into memory.

REAL-TIME FEATURES

Real-time applications require: (1) hardware to respond quickly to an external environment, (2) speed to keep up with the real-time process and (3) input/output flexibility to handle a wide variety of data types at varying speeds. A 32/70 Series system provides the following real-time computing features:

Multilevel, Priority Interrupt Structure of the real-time oriented 32/70 Series systems provides a quick response to interrupts with a maximum of 112 interrupt levels. The source of each interrupt is automatically identified and responded to according to its priority. For further flexibility, each level can be individually disabled to discontinue input acceptance and to defer responses.

The way interrupt levels are programmed is not affected by the priority assignment.

Programs that deal with interrupts from special purpose devices often require checkout before the equipment is actually available. To simulate special equipment, any external interrupt level can be requested by the CPU by executing a single Request Interrupt (RI) instruction. This capability is also useful in establishing a modified hierarchy of responses. For example, when servicing a high-priority interrupt and the urgent processing is finished, it is often desirable to assign a lower priority to the rest of the service routine so that the interrupt system can respond to other critical stimuli. A service routine can do this by requesting a lower-priority interrupt level, and thereby process the remaining data after other interrupts have been serviced.

Real-Time Clocks are needed to handle the real-time functions that must be timed to occur at specific instants. Other timing information is also needed, such as elapsed time since a given event or the current time of day. Clocks also allow easy handling of separate time bases and relative time priorities. A 32/70 can support up to seven real-time clocks synchronized to a line frequency of 50 Hz or 60 Hz. The clocks can also run at twice the line frequency, 100 Hz or 120 Hz, or on an external source.

Programmable Interval Timers can be set to request an interrupt after any specified time period with a 300-nanosecond resolution. In addition to the real-time clocks, the system can support seven programmable interval timers.

Context Switching must be done quickly with a minimum of time overhead. When responding to a new set of interrupt-initiated circumstances, a computer system must preserve the current operating environment, so the program can continue later, while setting up the new environment. In a 32/70 Series system, all relevant information about the current environment (instruction address, privilege state, condition codes, address modes, etc.) is kept in a 32-bit Program Status Word (PSW) or 64-bit Doubleword (PSD).

When an interrupt occurs, the CPU stores the current PSW or PSD in the memory location(s) selected by the interrupt level and loads a new PSW or PSD to establish a new environment.

Every 32/70 Series system also includes a Load File and Store File instruction so that the entire set of general purpose registers can be loaded or stored with one instruction. These instructions help make context switching fast and easy.

Quick Response is a 32/70 Series feature which involves the following combination: rapid context switching, store file and load file instructions, and a priority interrupt system. These features benefit all users because more of the system's resources are available for usesful work at any given time.

Memory Protection features that protect each user from every unprivileged user also guarantee the integrity of programs essential to critical real-time applications.

Input/Output requirements are available for a wide range of capacities and speeds. The 32/70 Series I/O system satisfies the needs of many different application areas economically and efficiently in terms of equipment and programming.

MULTIUSAGE FEATURES

A 32/70 Series system can run programs from two or more computer application areas concurrently. The most difficult general computing problem is the real-time application because it has several requirements. The most difficult multiusage problem is a terminal-oriented application that includes one or more real-time processes. Because the 32/70 Series systems have been designed on a real-time base, they are uniquely qualified for a mixture of applications in a multiusage environment. Many hardware features that prove valuable for one application area are useful in others, although in different ways. This multiple capability makes a 32/70 Series system particularly effective in multiusage applications.

The Instruction Set is large enough to provide the computational and data-handling capabilities required for widely differing application areas. This allows user programs to be short and fast.

Memory Protection makes it possible to run both real-time and batch programs concurrently in a 32/70 Series system. Real-time programs are protected against destruction by unchecked batch programs. Under Real-Time Monitor Control, the memory write-protection feature prevents destruction of information in protected memory.

Variable Precision Arithmetic is important in real-time systems where the data encountered is often 16 bits or less. To process this data efficiently, as well as the data in a batch environment, the 32/70 Series computers provide bit, byte, halfword, word, and doubleword arithmetic.

Priority Interrupts are especially useful because they make it possible for many elements to operate simultaneously and asynchronously. An interrupt system allows the computer to respond quickly and in proper sequence to the many demands made upon it.

MULTIPROCESSING FEATURES

Every 32/70 Series computer is designed to function as a shared-memory, multiprocessor system. It can support up to 20 Central Processor Units that share memory, and may have up to 16 Input/Output Microprogrammable Processors per CPU. All processors in a 32/70 Series system can address shared memory using identical addresses.

The 32/70 Series computers have the following major features that allow expansion of a single processor to a multiprocessor system:

Multiprocessor Interlock. In a multiprocessor system, a Central Processor Unit (CPU) often needs exclusive control of a system resource. This resource can be a region of memory, a particular peripheral device, or in some cases, a specific software routine. The 32/70 Series computers have a special set of instructions to provide this required multiprocessor interlock. The special instructions are Set Bit in Memory, Reset Bit in Memory, Test Bit in Memory, and Add Bit in Memory. The Set Bit in Memory instruction sets a bit in the selected position of the referenced memory location before other CPUs are allowed to access that memory location. If this bit had been previously set by another CPU, the interlock is set and the testing program proceeds to another task. On the other hand, if the bit of the tested location is a zero, the resource is allocated to the testing CPU. Simultaneously, the interlock can be set to lock out any other CPU.

Private Memory. Each CPU in a multiprocessor system must retain some private memory for its trap and interrupt locations, I/O communication locations, and other dedicated locations. This private memory consists of at least 8,192 words for each CPU. This private memory must begin with real address zero. The implicitly assigned trap locations and interrupt locations occupy the first 1,096 words of private memory. The remaining words in private memory can be used as private, independent storage by the CPU.

FUNCTIONAL DESCRIPTION

MAJOR SYSTEM ELEMENTS

The major elements of a typical 32/70 Series computer system include: the SelBUS, a Central Processor Unit, a Real-Time Option Module, main memory, an input/output subsystem, and a System Control Panel (see Figures 1-1 and 1-2 for system block diagram examples). The overall computer system can be viewed as a group of program-controlled subsystems communicating with a common memory. Each subsystem operates semi-independently with automatic overlap of subsystem operation occurring when conditions permit. This overlap greatly enhances the speed of operation. The major elements are listed below along with a brief functional description.

1. SelBUS - provides for high-speed communication between the major system elements.

2. Central Processor Unit - performs overall control and data reduction tasks.

3. Real-Time Option Module - implements internal and external interrupts and traps.

4. Main Memory - provides for private and shared storage.

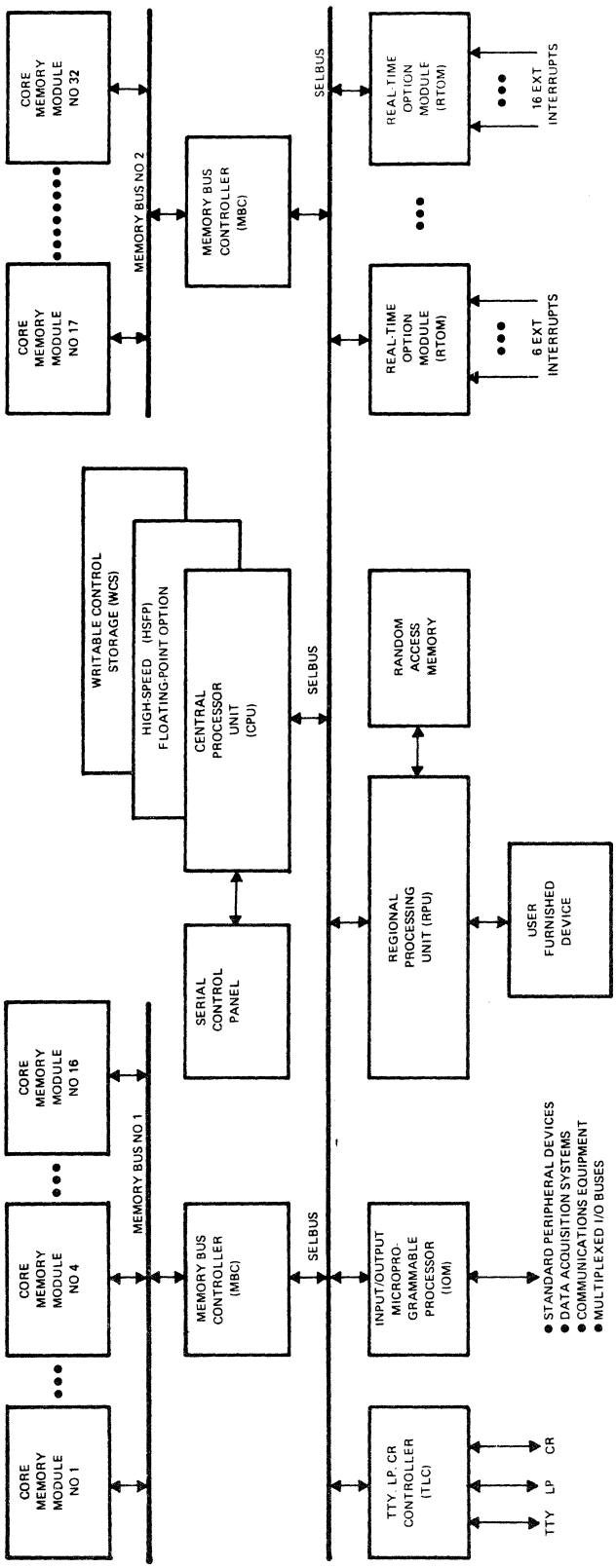


Figure 1-1. System Block Diagram Example:
Typical 32/70 Series System with Core Memory

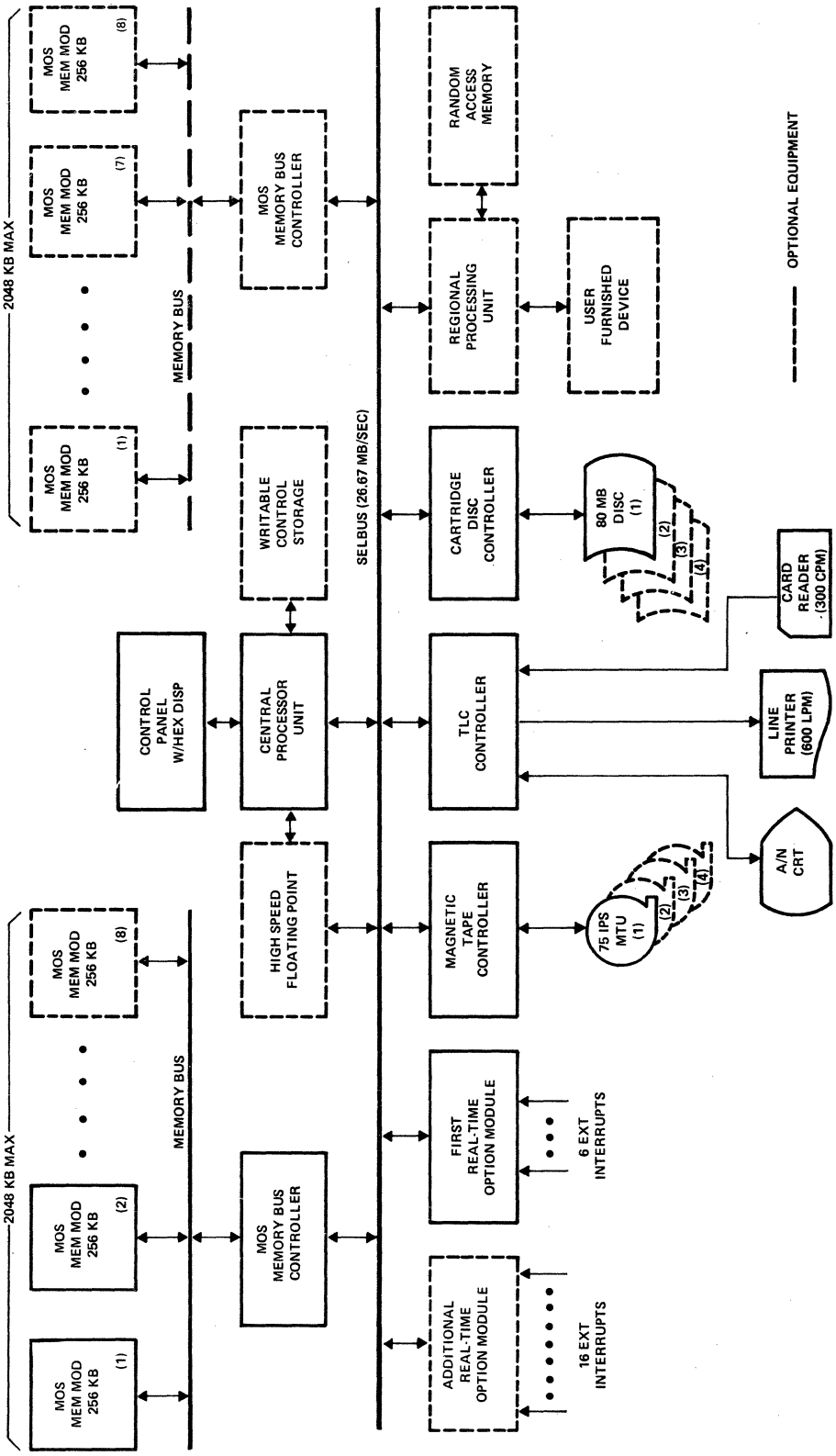


Figure 1-2. System Block Diagram Example:
Typical 32/70 Series System with MOS Memory

5. Input/Output Subsystem - enables information exchange between memory and selected peripheral devices.

6. System Control Panel - provides for user interaction with the system.

SelBUS

The SelBUS is a 184-line bidirectional bus that sends and receives data between the CPU, the memory subsystem, the Regional Processing Unit (RPU), the Input/Output Microprogrammable Processors (IOMs) on 32 data lines at a continuous data rate of 26.67 million bytes per second. Twenty-four address lines are used to address the selected IOM or memory interface for a read or write operation. Both data and address lines operate concurrently, and the transfers occur every 150 nanoseconds.

In a multiprocessor or special system configurations, remote memory subsystems, dual-processor shared-memory options, and memory ports may be connected to the SelBUS to support remote, shared, or private memory.

CENTRAL PROCESSOR UNIT

The 32/70 Series Central Processor Unit (CPU) is contained on three plug-in circuit boards. Two of the boards are the Micro Arithmetic/Logic Unit. The third board is the Micro Control Unit, which is sometimes referred to as the personality board.

Instructions on a 32/70 Series computer are continuously and automatically fetched for processing. This occurs concurrently with execution and decoding of previous instructions. Decoding is by proprietary parsing logic which employs parallel Read-Only Memories (ROMs) for high-speed decoding.

GENERAL PURPOSE REGISTERS

Eight integrated-circuit, 32-bit general purpose registers (GPRs) are used by the CPU. These eight registers of fast memory are referred to as the general purpose file.

Each general purpose register is identified by a 3-bit code in the range 000 through 111 (0 through 7 in decimal). Any general purpose register can be used as a fixed-point accumulator, floating-point accumulator, or temporary data storage location. A register can also contain control information such as a data address, count, or pointer. General purpose registers 1 through 3 can be used as index registers. Register 4 can be used as a mask register. Register 0 is a link register and an interval timer count.

FLOATING-POINT ARITHMETIC PROCESSOR

A firmware floating-point arithmetic processor is standard with the Central Processor Units. The firmware floating-point arithmetic processor executes all floating-point instructions significantly faster than normal software floating-point routines.

CPU MODES

A 32/70 Series computer can operate in eight different modes: four control modes (PSW-Privileged, PSW-Unprivileged, PSD-Privileged, PSD-Unprivileged) and four addressing modes (512 KB, 512 KB Extended, 512 KB Mapped, Mapped Extended).

The Extended mode can mean either 1 megabyte or 16 megabytes depending on the mapping mode. Table 1-1 shows the interrelationships among the control and address modes.

Table 1-1. Relationship of CPU Modes

Control Modes / Addressing Modes	PSW		PSD	
	Privileged	Unprivileged	Privileged	Unprivileged
Unmapped				
512 KB	X	X	X	X
512 KB Extended	X	X	X	X
Mapped				
512 KB	NA	NA	X	X
Extended	NA	NA	X	X

Control Modes

The basic control mode is designated either Program Status Word (PSW) or Program Status Doubleword (PSD) mode. The PSW mode allows a 32/70 Series computer to emulate the environment required to run the Real-Time Monitor (RTM); whereas the PSD mode makes it possible to create the environment required to run the Mapped Programming Executive (MPX).

The CPU, when in the PSW mode or PSD mode, can run in either the Privileged or Unprivileged mode.

Privileged operation allows the CPU to perform all of its control functions and to modify any part of the system. It is assumed that the resident operating system (operating in the Privileged mode) controls and supports the execution of other programs (which can operate in the Privileged or Unprivileged mode).

Unprivileged operation is the problem-solving mode of the CPU. In this mode, memory protection is in effect, and all privileged operations are prohibited. Privileged operations are those relating to input/output and to changes in the basic control state of the computer. All privileged operations are performed by a group of privileged instructions. Any attempt by a program to execute a privileged instruction while the computer is in the Unprivileged mode results in a trap.

The Privileged/Unprivileged mode control bit can be changed when the computer is in the Privileged mode. An Unprivileged mode program can gain direct access to certain executive program operations by means of Supervisory Call or Call Monitor instructions. The operations available through these instructions are established by the resident operating system.

Addressing Modes

The basic addressing modes are designated either Unmapped or Mapped. Addressing submodes are 512 KB or extended addressing (refer to Table 1-1).

Unmapped addressing establishes a one-to-one relationship between the effective virtual address of each operand or instruction and the physical address in memory.

Mapped addressing uses the memory management hardware to convert effective virtual operand and instruction addresses into physical (real) memory addresses located anywhere in up to 16 megabytes of physical memory. The memory management hardware contains a MAP which allows the privileged user to define how virtual addresses are converted to real addresses.

The MAP contains thirty-two 16-bit registers; the first 16 registers contain the Primary MAP to define a 512 KB primary logical address space, and the second 16 registers contain the Extended Operand Map to define an additional 512 KB extended operand address space for additional data storage.

Addressing Submodes

The addressing submodes are 512 KB and extended addressing. 512 KB addressing allows direct addressing of 512K bytes (128K words) of memory. In the 512 KB mode, this address space consists of the first 512K bytes in memory. In the 512 KB Mapped mode, this address space is the 512K bytes of primary logical address space for each user.

Extended Addressing allows a program through indexing to extend the address space beyond 512K bytes. In the Unmapped Extended mode, the extension is to 16 megabytes. In Mapped-Extended mode, provision is made for up to 1 megabyte of logical address space for each user. The mapping hardware can locate this 512 KB space in 8,192-word segments anywhere in up to 16 megabytes of physical memory.

HARDWARE MEMORY MANAGEMENT

The Hardware Memory Management feature of 32/70 Series computers use dynamic Memory Allocation and Protection (MAP) This allows programs to be loaded in one area of physical memory, rolled out to disc, rolled back into another area of memory, and to continue execution without requiring time-consuming software relocation biasing. In addition, user programs may be write protected and distributed throughout physical memory in 32K-byte blocks. Thus, the full utilization of available memory is a practical possibility.

Memory Map

A memory map deals with virtual and real addresses. A virtual address pertains to the logical space used by a machine-level program and is normally derived from programmer-supplied labels through an assembly (or compilation) process followed by a loading process. Virtual addresses may be used to designate an element of data, the location of an instruction, and either an indirect or immediate (explicit) address. A real (physical) address is the address a processor sends to the memory address register to access a specific physical memory location for storage or retrieval of information. Real addresses are determined by the hardware, whereas virtual addresses include all addresses.

The memory map provides dynamic program relocation into discontiguous segments of memory. When the CPU is operating in Mapped mode, a program can be segmented into an integral number of 8,192-word blocks and distributed throughout memory in whatever space is available. The memory map transforms virtual addresses, as seen by the individual program, into real addresses, as seen by the memory system.

When the CPU is not in the Mapped mode, as determined by a control bit in the Program Status Doubleword (PSD), all virtual addresses are used by the CPU as real addresses. When the CPU is operating in the Mapped mode, all virtual addresses are transformed into real addresses by replacing the high-order four or five bits (dependent upon extended addressing) of the virtual address with a 9-bit value obtained from the memory map register.

WRITE PROTECTION

The memory protection system provides write protection for individual memory pages. When the CPU is in the Mapped mode (either 512 KB or Extended), each 32 KB memory block of logical program address space may be write protected. Write protection for a 32 KB memory block is selected by setting the protect/unprotect bit that is stored, along with the block address, in the MAP register of the CPU.

When the CPU is in either the Unmapped or Mapped mode (either 512 KB or Extended), 512-word memory pages may be write protected. Up to 256 pages (128K words) can be protected at a time. Sixteen 16-bit Page Protect registers are provided in the CPU for write protection in the Unmapped or Mapped mode.

Write protection may be overridden by a CPU operating in the Privileged mode.

OPTIONAL WRITABLE CONTROL STORAGE

The optional Writable Control Storage (WCS) may be used to expand the 32/70 Series computer instruction repertoire and to enhance the performance of user programs. By microprogramming a 32/70 Series computer with firmware subroutines, the optional Writable Control Storage (WCS) can tailor the computer to perform specific applications such as Fourier transforms, coordinate transformation, polynomial evaluation, and number system conversion.

Further improvement in overall performance is achieved by using microprograms for frequently executed subroutines in the FORTRAN Run-Time Package, the FORTRAN Compiler, the BASIC Interpreter, and the 32/70 operating system. All high-speed firmware subroutines can be invoked from main memory for execution as needed.

Up to 4,096 64-bit words of Writable Control Storage (WCS) can be added to a 32/70 Series computer in increments of 2,048 64-bit words. Each increment plugs into the SelBUS for power and clock. However, communication with the CPU is independent of SelBUS operation.

OPTIONAL HIGH-SPEED FLOATING-POINT UNIT

The optional High-Speed Floating-Point Unit functions as an extension of the 32/70 Series central processor to perform high-speed execution of floating-point arithmetic instructions. Addition, subtraction, multiplication and division of single-precision (32-bit) or double-precision (64-bit) operands are possible with execution times that are significantly greater than with the standard floating-point feature of the CPU.

REAL-TIME OPTION MODULE

The first RTOM in the system provides the 10 basic interrupts and traps which comprise the system integrity features. These basic interrupts and traps include: Power Fail-Safe, System Override, Memory Parity, Nonpresent Memory, Undefined Instruction, Privilege Violation, Attention, Call Monitor, Real-Time Clock, and Arithmetic Exception.

The first RTOM also provides the six highest external interrupt levels, one of which may be used for the standard interval timer.

INTERVAL TIMER

The programmable interval timer provides a 32-bit counter that can be loaded examined, started, or stopped by way of a Command Device (CD) instruction. The Command Device (CD) enables the counter at one of four program-selectable rates. When the counter is decremented to zero, the interval timer requests a priority interrupt.

MAIN MEMORY

An introduction to the basic organization and operation of the main memory subsystem is provided in the paragraphs that follow.

A 32/70 Series system may have either core or MOS memory. Packaged systems are sold with one or the other but not both for the same system. The user may elect to mix the two types of memory, but only if it is done in accordance with the configuration rules specified in Section III of this manual.

MEMORY UNIT

The main memory for a 32/70 Series system is physically organized as a group of units. A memory unit is the smallest logically complete part of the system, and the smallest part that can be logically isolated from the rest of the memory system. A memory unit consists of 1 or 2 memory chassis, a power supply, 1 to 4 Memory Bus Controllers (MBCs), and 1 to 16 memory modules. Memory units with MOS memory also include a Refresh board.

MEMORY MODULE

A memory module is the basic functionally independent element of the memory system. Each module can operate concurrently with all others in a memory unit. A memory module consists of storage elements, drive and sense electronics, control timing, and data registers. Core and MOS memory modules are described separately, as follows:

1. Core memory modules have either 8,192-word (32K-byte) locations with a 600-nanosecond cycle time or 16,384-word (64K-byte) locations with a 900-nanosecond cycle time. Each word contains a total of 36 bits: 32 data bits and 4 parity bits (1 parity bit per byte). Byte, halfword, word, or doubleword addresses may be used to access memory.

2. MOS memory modules have either 65,536-word (256K-byte) or 131,072-word (512K-byte) locations; both have a cycle time of 900 nanoseconds. MOS memory is organized into 39-bit words: 32 data bits plus 7 error checking and correction (ECC) bits. The seven error correction bits report and correct single-bit errors. The ECC bits also detect and report (but do not correct) double-bit errors.

MEMORY INTERLEAVING

When a system consists of two memory modules (or a multiple thereof), memory can be two-way interleaved. If a system has four modules (or a multiple thereof), memory can be four-way interleaved. Memory interleaving is a built-in hardware feature that distributes sequential addresses into independently operating memory modules. Interleaving increases the probability that a processor can gain access to a given memory location without encountering interference from other processors. Thus, interleaving significantly reduces cycle time and increases the throughput rate.

With two-way interleaving, even addresses are assigned to even-numbered memory modules and odd addresses to odd-numbered memory modules. Four-way interleaving assigns every fourth address to its respective memory module and can occur when a multiple of four memory modules are included in a unit.

MEMORY UNIT ADDRESS IDENTITY

Each memory unit in a 32/70 Series system is provided with an individual identity by means of address range switches. These switches define the range of addresses to which the unit responds when servicing memory requests. All addresses, including the starting address, for a given unit should be the same for all Memory Bus Controllers (MBCs) in that unit; that is, the address of a given byte remains the same regardless of the MBC used to access the byte. The starting address of a unit must be on a boundary equal to a multiple of the size of the memory modules in the unit. If the unit is interleaved, the unit must contain a multiple of the memory modules'size times the number of interleaves.

MEMORY BUS CONTROLLERS

The Memory Bus Controllers (MBCs) in a memory unit act as an interface between the processing units (CPUs, IOMs, and RPUs) on the SelBUS and the memory modules. Each memory unit can have from one to four MBCs. Each MBC is capable of managing up to 16 memory modules with overlapped operation. All memory modules assigned to one MBC must be of the same type (either MOS or core but not both) and have the same cycle and access time.

MBCs examine incoming addresses to determine if the request is for a memory module within the memory unit. In addition, an MBC determines the priority of memory requests that are received simultaneously. Computer memory requests can be initiated every 150 nanoseconds due to the overlapped memory design.

The 32/70 Series systems can include from one to eight MBCs per SelBUS. All processors, either CPUs or I/O processors, must interface to memory by way of an MBC. MBCs are located, along with the memory modules, in a separate chassis from the CPU and I/O processors. Depending on the particular system and the needs of the user, an MBC may be configured in a variety of ways. For example, an MBC can connect directly to the SelBUS; or, a Memory Interface Adapter (MIA) and/or Memory Bus Adapter (MBA) may be employed to provide indirect connection between the SelBUS and an MBC.

MEMORY LOCK AND UNLOCK

MBCs can be locked and unlocked by a CPU. A Memory Lock signal can be sent to the MBC in conjunction with a read transfer, and a Memory Unlock signal can be sent during a write transfer. The Read and Lock transfer is used to access a word instruction in memory and to lock out all other processors from the MBC. A Write and Unlock transfer causes information to be written into memory and enables access to the MBC by other SelBUS devices. Only CPUs can use the Lock and Unlock feature.

When a Read and Lock transfer is received, the MBC involved is temporarily inhibited from accepting any additional transfer requests. However, all transfer requests already accepted by the MBC, but not yet completed, will be processed normally.

PRIVATE MEMORY

In a 32/70 Series multiprocessing system, all processors address memory in the same manner. The CPUs do not share the same interrupt or trap systems. Thus, it is necessary to provide private storage for each CPU to contain its trap and interrupt locations, I/O communication locations, and scratchpad locations. This private memory must begin at 0 and extend at least to 2,048 memory locations (bytes).

INPUT/OUTPUT SYSTEM

The Input/Output Microprogrammable Processor is the basic hardware structure of the I/O processor and consists of a SelBUS interface, a microprocessor, and interface logic for an external device.

The SelBUS interface provides for communication between the IOM and the CPU, or between the IOM and memory. The microprocessor has a Control Read-Only Memory (CROM) that contains the microprogram (firmware) for controlling the SelBUS interface, microprocessor, and device interface logic. The device interface logic may consist of some control logic for operating the I/O interface and the receivers/drivers necessary to communicate with the I/O device or external interface.

There are three classes of I/O processors in a 32/70 Series system: the IOM, the RPU, and the General Purpose Multiplexer I/O processor. The I/O processor can also be used to provide a General Purpose Input/Output interface (GPIO). The customer must design the device interface logic and supporting firmware to make the I/O processoor and device dependent interface operate as an I/O processor for some specific type of I/O device(s).

IOM

The IOM is the basic I/O processor which contains the microprogrammable processor, the SelBUS interface, and the device interface on a single logic card.

REGIONAL PROCESSING UNIT

The Regional Processing Unit (RPU) serves as a General Purpose Input/Output interface (GPIO) for the peripheral device(s). The RPU connects directly to the SelBUS, the major artery for transmitting information. The RPU consists of three individual elements which are self-contained on separate modules: the regional processor, the device interface, and optional high-speed Random Access Memory (RAM). The major characteristic of the RPU is that it supports Random Access Memory or Writable Control Storage that can be programmed to suit the user's requirements.

GENERAL PURPOSE MUTIPLEXER CONTROLLER

A third type of I/O processor is the General Purpose Multiplexer Controller (GPMC) which controls a number of individual controllers that are located at various distances from the processor. The GPMC can schedule requests for main memory between several controllers. The GPMC also connects each dependent controller to the CPU for initiation or termination of an I/O operation.

SECTION II

CENTRAL PROCESSOR

INTRODUCTION

This section of the manual describes the 32/70 Series Central Processor Unit (CPU). Included are an introduction to the instruction repertoire and descriptions of the modes of operation, their format, and the major functional elements of the CPU.

INSTRUCTION REPERTOIRE

The functional classifications and corresponding number of instructions for the 32/70 Series computer are as follows:

Classifications	Number
Fixed-Point Arithmetic	30
Floating-Point Arithmetic	8
Boolean	17
Load/Store	29
Bit Manipulation	8
Zero	5
Shift	13
Interrupt	13
Compare	11
Branch	9
Register Transfer	13
Input/Output	10
Control	16
Hardware Memory Management	4
Writable Control Storage	3
Total	189

Of particular significance are the bit manipulation and floating-point instructions. The eight bit manipulation instructions provide the capability to selectively set, zero, add, or test any bit in memory or register.

The eight floating-point instructions are unique because they can either be executed by the firmware in the CPU, or by the optional High-Speed Floating-Point Arithmetic Unit. Except for the execution speed, the presence or absence of the optional Floating-Point Arithmetic Unit is transparent to the user.

All of the instructions in the repertoire are classified as either being halfword instructions (16 bits) or word instructions (32 bits). The word instructions primarily reference memory locations; the halfword instructions primarily deal with register operands. Because approximately one-third of the instructions are halfword instructions, program core space can be conserved by packing two consecutive instructions into one memory location.

The 32/70's use instruction lookahead for fast instruction execution. Instruction fetches are made concurrently with instruction execution and with decoding a previously fetched instruction.

GENERAL PURPOSE REGISTERS

The 32/70 Series CPU has a set of eight high-speed, general purpose registers for use by the programmer for arithmetic, logical, and shift operations. Three general purpose registers - R1, R2, and R3 - can also be used for indexing operations. Register R0 can also be used as a link register. Register R4 can be used as a mask register.

CPU CONTROL MODES

The CPU operates in either of two basic control modes: the PSW mode or the PSD mode. The PSW mode provides an environment to run the Real-Time Monitor (RTM) Operating System. The PSD mode provides an environment to run the optional Mapped Programming Executive (MPX-32) Operating System. The functional difference between the PSW and PSD modes are outlined in Table 2-1.

PROGRAM STATUS WORD

A Program Status Word (PSW) is used to record all machine conditions that must be preserved prior to context switching when in the PSW mode of operation. The PSW supports only the Class 0,1,2,3, and E I/O devices using the Command Device (CD) and Test Device (TD) instructions. The format of the PSW is shown in Figure 2-1.

PROGRAM STATUS DOUBLEWORD

A Program Status Doubleword (PSD) is used to record all machine conditions that must be preserved prior to context switching when in the PSD mode of operation. The format of the PSD is shown in Figure 2-2. Execution of any Branch-and-Link instruction replaces the contents of bits 13-30 of the PSD with the effective address specified by the instruction. In addition, if the Branch instruction specifies an Indirect Branch operation, the contents of bits 1-4 of the PSD are replaced by the contents of the corresponding bit positions in the indirect address location.

CONDITION CODES

A 4-bit Condition Code is stored in the PSW or PSD upon completion of the execution of most instructions. These conditions may be tested to determine the status of the results obtained.

CC1 is set if an Arithmetic Exception occurs
CC2 is set if the result is greater than Zero
CC3 is set if the result is less than Zero
CC4 is set if the result is equal to Zero

The Branch Condition True (BCT), Branch Condition False (BCF), and the Branch Function True (BFT) instructions allow testing and branching on the condition codes.

PRIVILEGED AND UNPRIVILEGED OPERATION

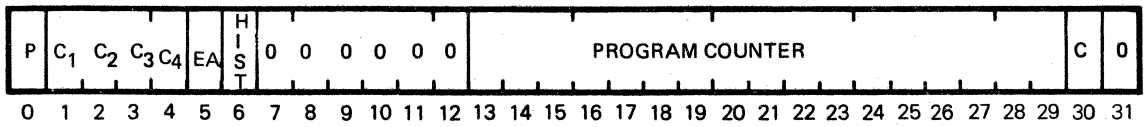
The CPU is capable of either privileged or unprivileged operation in both the PSW and PSD modes. Privileged operation allows the CPU to perform all of its control functions and to modify any part of the system. Privileged operation relates to input/output and to changes in the basic control state of the computer. Unprivileged operation is the problem-solving mode of the CPU. In this mode, memory protection is in effect and all privileged operations are prohibited.

One bit in the Program Status Doubleword (PSD) or Program Status Word (PSW) is designated as the Privileged State bit. If the Privileged State bit is set, privileged instructions can be executed. If the Privileged State bit is reset, any attempt to execute a privileged instruction will cause a Privileged Violation trap.

Table 2-1. PSW and PSD Modes: Functional Differences

Characteristics	PSW Mode*	PSD Mode**
Program Status	Word	Doubleword
Number of Instructions	160	189
Integrity Features	Interrupts on first RTOM	Traps
Memory Addressing		
Nonmapped		
Nonextended	512 KB	512 KB†
Extended	16 MB	16 MB†
Mapped		
Nonextended	None	512 KB per user
Extended	None	1 MB per user
CD I/O	Yes	Yes
Addressing	512 KB	512 KB
Extended I/O	No	Yes
Addressing	None	16 MB

* RTM supported
** MPX supported
† No software support



BIT 0 DESIGNATES THE PRIVILEGED STATE BIT

BIT 1-4 DESIGNATE THE CURRENT CONDITION CODE

BIT 5 DEFINES THE EXTENDED ADDRESSING MODE (ABOVE 128K)

BIT 5=0 NONEXTENDED ADDRESSING
BIT 5=1 EXTENDED ADDRESSING

BITS 6 DEFINES THE POSITION OF THE LAST INSTRUCTION EXECUTED

BIT 6 = 0 LEFT HALFWORD OR FULLWORD
BIT 6 = 1 RIGHT HALFWORD

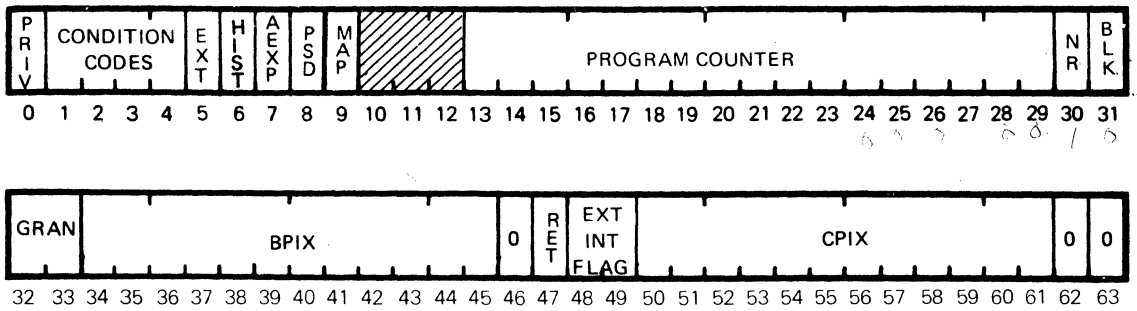
BITS 7-12 UNASSIGNED, MUST BE ZERO

BITS 13-29 CONTAIN THE WORD ADDRESS (PC) COUNT OF THE NEXT INSTRUCTION TO BE EXECUTED

BIT 30 DEFINES THE POSITION OF THE NEXT INSTRUCTION (LEFT OR RIGHT INSTRUCTION)

BIT 30=0 LEFT HALFWORD
BIT 30=1 RIGHT HALFWORD

Figure 2-1. Program Status Word (PSW) Format



BIT 0 = 0 UNPRIVILEGED MODE
= 1 PRIVILEGED MODE

BITS 1-4 ARE CONDITION CODES
BIT 1 = CC1
2 = CC2
3 = CC3
4 = CC4

BIT 5 = 0 EXTENDED MODE (OFF) CEA
= 1 EXTENDED MODE (ON) SEA

BIT 6 = 0 LAST INSTRUCTION EXECUTED WAS NOT A RIGHT HALFWORD
= 1 LAST INSTRUCTION EXECUTED WAS A RIGHT HALFWORD

BIT 7 = 0 ARITHMETIC EXCEPTION TRAP MASK (OFF)
= 1 ARITHMETIC EXCEPTION TRAP MASK (ON)

*BIT 8 = 0 COMPUTER IS IN PSW MODE (DISPLAYED PSD ONLY) *
= 1 COMPUTER IS IN PSD MODE (DISPLAYED PSD ONLY) *

*BIT 9 = 0 UNMAPPED (DISPLAYED PSD ONLY) *
= 1 MAPPED (DISPLAYED PSD ONLY) *

BITS 10-12 ARE NOT USED
BITS 13-29 ARE LOGICAL WORD ADDRESS
BIT 30 NEXT INSTRUCTION IS A RIGHT HALFWORD
*BIT 31 BLOCKED (DISPLAYED PSD ONLY) *
BITS 32-33 INDICATE MAP GRANULARITY, 00=UNMAPPED AND ALL OTHERS =8K MAP GRANULARITY
BITS 34-45 PROVIDE A WORD INDEX INTO THE MASTER PROCESS LIST (MPL) FOR THE BASE PROCESS
BIT 46 NOT USED
BIT 47 RETAIN CURRENT MAP CONTENTS
BITS 48-49 INTERRUPT CONTROL FLAGS

BITS		
48	49	
0	0	OPERATE WITH UNBLOCKED INTERRUPTS
0	1	OPERATE WITH BLOCKED INTERRUPTS
1	0	RETAIN CURRENT BLOCKING MODE
1	1	RETAIN CURRENT BLOCKING MODE

BITS 50-61 PROVIDE WORD INDEX INTO MASTER PROCESS LIST (MPL) FOR CURRENT PROCESS
BITS 62-63 NOT USED

* THESE BITS ARE USED FOR DISPLAY ONLY AND ARE NOT PRESENT IN THE PSD STORED IN MEMORY.

Figure 2-2. Program Status Doubleword (PSD) Format

The following instructions are privileged:

1. All interrupt related instructions such as Enable Interrupt or Request Interrupt.

2. All instructions that can modify the memory mapping registers.

3. All Input/Output instructions.

4. All instructions that can place the machine in a state that requires operator intervention to continue processing, such as Halt.

5. All instructions that modify Writable Control Storage.

User programs operating in the unprivileged state should use the Call Monitor (CALM) or Supervisor Call (SVC) instruction with the appropriate program flags to use the system features guarded by the privileged/unprivileged system.

Certain events can change the processor from the unprivileged to the privileged state by loading a new Program Status Word or Doubleword. These are:

1. An interrupt from an external event or the I/O system.

2. A hardware trap caused by addressing nonpresent memory, executing undefined instruction, executing privileged instruction by nonprivileged program, or writing to protected memory.

3. A hardware trap caused by a nonrecoverable condition such as an uncorrectable error on a memory read, or an arithmetic exception.

4. The execution of the Call Monitor or Supervisor Call instruction by a user requesting monitor services.

In all cases, traps or interrupts are vectored to monitor routines for proper handling. Both the interrupt/trap vectors and the monitor service routines are in protected memory. This insures that an unprivileged user has no way to become privileged or to alter protected state.

The execution of the Branch and Reset Interrupt (BRI) or the Load Program Status Doubleword (LPSD) instruction can cause the system to change from the privileged to the unprivileged state.

The operator can push the SYSTEM RESET button to initialize a 32/70 Series computer. SYSTEM RESET clears the eight general purpose registers, resets all memory protection, and sets the Privileged State bit.

CPU ADDRESSING MODES

The 32/70 Series CPU has four modes for accessing memory:

1. 512 KB mode

2. 512 KB Extended Mode

3. 512 KB Mapped mode

4. Mapped, Extended mode

512 KB MODE

The 512 KB addressing mode allows the 32/70 Series CPU to access instructions or operands (bit, byte, halfword, word, or doubleword) in the first 512K bytes of memory directly without mapping, indexing, or address modification. A 19-bit Address field is provided in memory referencing instructions for that purpose.

Bit addressing is accomplished by using the Register (R) field in the instruction word to select a bit in the byte specified by the 19-bit address. Therefore, any bit in the first 512K bytes of memory can be directly addressed by the Bit Manipulation instructions.

512 KB EXTENDED MODE

The 512 KB Extended mode provides the same capabilities as the 512 KB mode described above, and, in addition, it permits operand addressing beyond the first 512K bytes of memory. The effective address can reference any bit, byte, halfword, word, or doubleword residing anywhere within 16 megabytes of physical memory.

512 KB MAPPED MODE

The 512 KB Mapped mode allows a 32/70 Series CPU to access any instruction or operand (bit, byte, halfword, word, or doubleword) within a logical primary address space. This space consists of 512K bytes of logical memory, distributed within 16 megabytes of physical memory.

The 32/70 Series CPU allows multiple primary address spaces. A user can access instructions and operands within the logical primary address space in which his program resides. Physical blocks of memory can be common to many logical primary address spaces; thus, users in different spaces can share common blocks of memory.

The 512 KB Mapped addressing mode can be used only when the CPU is in the PSD control mode.

MAPPED EXTENDED MODE

The Mapped Extended mode provides all the capabilities of the 512 KB Mapped mode, plus access to a logical extended operand address space. This space consists of 512K bytes of memory beyond the logical primary address space and allows users additional memory space to store data (operands). Each logical extended operand address space can be 512K bytes long, dispersed anywhere within 16 megabytes of physical memory. The combination of logical primary address space and the logical extended operand address space supports programs up to one megabyte long. The executable code must lie within the logical primary address space, but operands can be in either the logical primary or extended operand address space.

The Mapped Extended addressing mode can be used only when the CPU is in the PSD control mode.

CPU MAJOR ELEMENTS

A brief description of some major elements of the CPU are provided in the paragraphs that follow. They include: the data structure, a microprogrammable processor, the implementation logic, and the SelBUS interface. A simplified block diagram of the CPU is shown in Figure 2-3. For a more comprehensive discussion of the CPU, refer to the 32/70 Series Computer Technical Manual.

CPU DATA STRUCTURE

The data structure contains the eight general purpose file registers and 10 hardware registers organized around an Arithmetic Logic Unit (ALU). Key circuits in the data structure include the following:

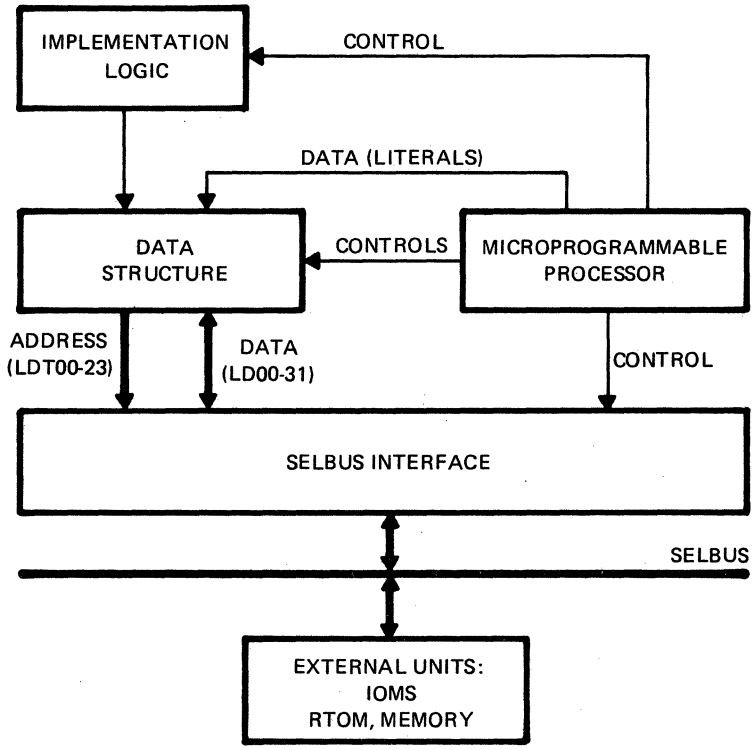


Figure 2-3. CPU Simplified Block Diagram

1. Arithmetic Logic Unit (ALU)
2. A-Multiplexer
3. B-Multiplexer
4. Literal Multiplexer
5. General File Register
6. Memory Address Register
7. Program Counter Register
8. N-Counter Register
9. Shift Register
10. Temporary Register/Data Output Register
11. Data Input Register
12. Instruction Register 0
13. Instruction Register 1

CPU MICRO-PROGRAMMABLE PROCESSOR

The Microprogrammable Processor of the CPU is on board C of the three CPU circuit boards. The logic circuit board which contains the Microprogrammable Processor is commonly referred to as the personality board.

The Microprogrammable Processor utilizes Read-Only Memory (ROM) integrated circuits which house the CPU's Elementary Operations (EO). The EOs, with the associated circuitry, control the CPU operations by testing, controlling, and directing the various functions to be performed. The format for the EOs (also referred to as microinstruction) is shown in Figure 2-4.

IMPLEMENTATION LOGIC

The Implementation Logic includes the ALU Decode PROM, a Scale circuit, the Floating-Point Assist PROMs, and a Multiply Assist PROM, all of which serve to implement CPU functions.

SelBUS INTERFACE

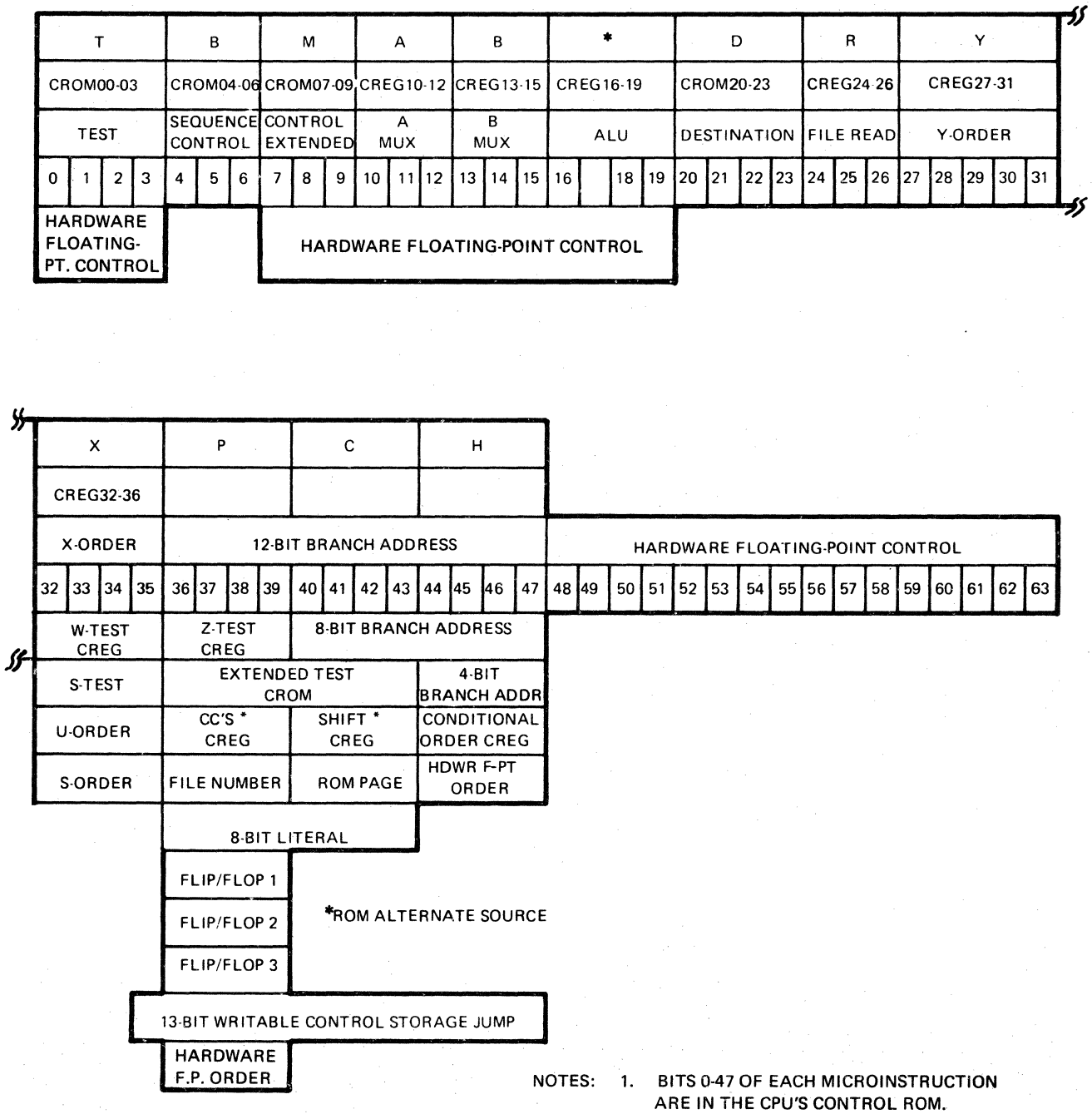
The SelBUS interface logic is implemented on all three of the CPU circuit boards and provides control and temporary storage for information being output to and input from the SelBUS. Since the SelBUS is the high-speed communication link between system modules external to the CPU, the SelBUS interface logic plays a vital role in CPU operation.

OPTIONAL WRITABLE CONTROL STORAGE

Writable Control Storage is an option which may be used with the 32/70 Series CPU to expand the instruction set, to enhance the performance of user programs, or to tailor the computer to specific user needs.

Up to 4,096 64-bit words of Writable Control Storage (WCS) can be added to a 32/70 Series computer in increments of 2,048 64-bit words. Each increment plugs into the SelBUS for power and clock. However, communication with the CPU is independent of SelBUS operation.

The block diagram in Figure 2-5 shows two optional WCS units as they could be implemented in conjunction with a 32/70 Series CPU and the optional High-Speed Floating-Point Unit.



NOTES:

1. BITS 0-47 OF EACH MICROINSTRUCTION ARE IN THE CPU'S CONTROL ROM.
2. PORTIONS OF THE FORMAT DESIGNATED FOR HARDWARE FLOATING-POINT APPLY TO THE OPTIONAL HIGH-SPEED FLOATING-POINT UNIT (FPU).
3. BITS 48-63 ARE PHYSICALLY PART OF A CONTROL ROM IN THE OPTIONAL HIGH-SPEED FPU.

Figure 2-4. Microinstruction Format

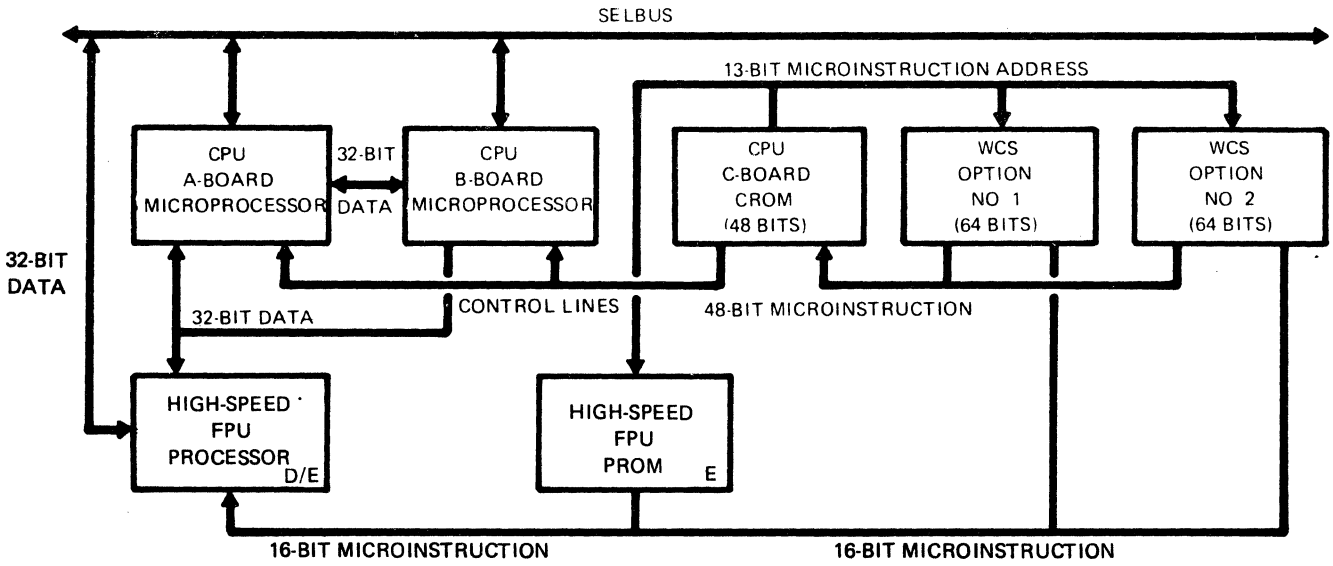


Figure 2-5. Functional Interrelationship of the CPU, WCS, and High-Speed Floating-Point Unit

OPTIONAL HIGH-SPEED FLOATING-POINT UNIT

The High-Speed Floating-Point Unit (FPU) is an option that may be used with a 32/70 Series CPU to increase the speed of floating-point arithmetic operations. The unit consists of two circuit boards which may be plugged in adjacent to the CPU. No alternations in the software are required.

If the High-Speed Floating-Point Unit (FPU) is installed, addition, subtraction, multiplication, and division of single-precision (32-bit) or double-precision (64-bit) operands can be executed much faster than with the CPU's standard floating-point feature.

An operand in floating-point format has three parts: a sign bit, a fraction, and an exponent. The sign bit indicates whether the fraction is a positive or negative value. The fraction is a binary number with an assumed radix point immediately to the left of its most significant bit. The exponent is a 7-bit binary power to which the base 16 is raised. The quantity that the floating-point number represents can be determined by multiplying the fraction by the number 16 raised to the power represented by the exponent.

Two operands of the same format and length are received by the FPU for each arithmetic operation. One operand is input from a CPU general purpose register (GPR), whereas the other operand is input from memory. The precise GPR and memory location are specified in the floating-point instruction. Upon completion of an operation, the result is returned to the CPU general purpose register.

Figure 2-6 illustrates the major functional elements of the FPU, the general routing of operands, and the relationships between the FPU, the CPU, and the SelBUS.

INTERNAL PROCESSING UNIT

INTRODUCTION

GENERAL

The Model 2005 Processing Unit is a high-performance processor which has been added to the SYSTEMS 32/70 Series Computer line. The Model 2005 processor's role as a Central Processing Unit (CPU) or Internal Processing Unit (IPU) is selected by a jumper on the C board of the processor. Both CPU and IPU on the same SelBUS must be Model 2005 processors. The IPU is designed for a computer configuration in which a large amount of arithmetic calculation is anticipated and is ideal for compute-bound number processing tasks and subroutines. The IPU, a three-board plug-in module, operates on the same SelBUS with a CPU and shares all of memory (including the resident operating system area) with the CPU.

The IPU and CPU operate in parallel, with the IPU executing task level, SYSTEMS 32 code at the same time the CPU is executing. The capability of paralleled processing of instructions allows for faster completion of code which would normally be processed in a serial manner by the CPU. The CPU is responsible for all task scheduling I/O and system services as well as for the execution of its own scheduled tasks.

Options available with the IPU are the Model 2341 High-Speed Floating Point and the Model 2343 and 2347 Scientific Accelerator.

The IPU is similar, in many instances, to the CPU. Because of this similarity, the IPU information presented in this section will emphasize only the differences and the unique aspects compared to the CPU as presented throughout this manual.

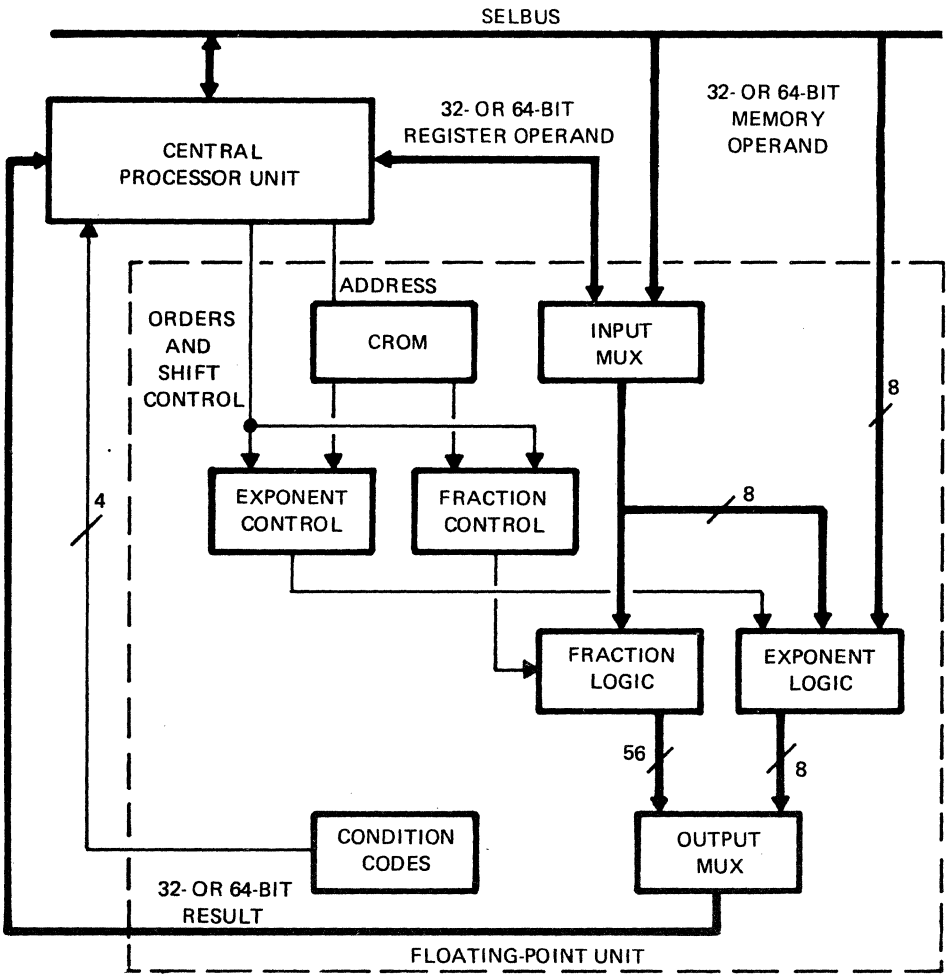


Figure 2-6. Optional High-Speed Floating-Point Unit

GENERAL CHARACTER-ISTICS

- INSTRUCTIONS
  - New Instruction - SIGNAL IPU (SIPU)
  - Instructions not used by IPU

    Control Instructions

    Branch and Reset Interrupt (BRI)

    Interrupt Instructions

    All Interrupt Instructions except UEI

    Input/output instructions

    All instructions

- TRAPS - Six new traps for IPU/CPU Operation
  - End IPU Processing
  - Start IPU Processing
  - IPU Supervisor CALL
  - IPU Errors
  - IPU Call Monitor
  - Stop IPU Processing
- Software
  - Under MPX-32 the IPU can be transparent to the user, or the user can designate which programs run on the IPU and which run on the CPU.
  - Two programs can run simultaneously because of the parallel operation of the IPU and CPU on the SelBUS.

INSTRUCTION REPERTOIRE

The functional classifications and corresponding number of instructions of the Internal Processing Unit are as follows:

Classification	Number of Instructions	
Fixed-Point Arithmetic	30	
Floating-Point Arithmetic	8	
Boolean	17	
Load/Store	26	
Bit Manipulation	8	
Zero	5	
Shift	13	
Interrupt	1	UEI
Compare	11	
Branch	9	
Register Transfer	13	
Input/Output	0	Unimplemented in IPU
Control	15	BRI unimplemented in IPU
Hardware Memory Management	4	
Writable Control Storage	3	
Total	163	

Of particular significance are the bit manipulation and floating-point instructions. The eight bit manipulation instructions provide the capability to selectively set, zero, add, or test any bit in memory or register.

The eight floating-point instructions are unique because they can either be executed by the firmware in the IPU, or by the optional High-Speed Floating-Point Arithmetic Unit. Except for the execution speed, the presence or absence of the optional Floating-Point Arithmetic Unit is transparent to the user.

All of the instructions in the repertoire are classified as either being halfword instructions (16 bits) or word instructions (32 bits). The word instructions primarily reference memory locations; the halfword instructions primarily deal with register operands. Because approximately one-third of the instructions are halfword instructions, program core space can be conserved by packing two consecutive instructions into one memory location.

The IPU uses instruction lookahead for fast instruction execution. Instruction fetches are made concurrently with instruction execution and with decoding a previously fetched instruction.

GENERAL PURPOSE REGISTERS

The IPU includes a set of eight high-speed, general purpose registers for programmer use for arithmetic, logical, and shift operations. Three general purpose registers (R1, R2, and R3) can also be used for indexing operations. Register R0 can also be used as a link register. Register R4 can be used as a mask register. These registers are distinctly separate from the registers used in the controlling CPU.

IPU CONTROL MODE

The IPU operates in the PSD mode. The PSD mode provides an environment to run the Mapped Programming Executive (MPX-32) Operating System. The PSD mode is outlined in Table 2-2.

PROGRAM STATUS DOUBLEWORD

A Program Status Doubleword (PSD) is used to record all machine conditions that must be preserved prior to context switching when in the PSD mode of operation. The format of the PSD is shown in Figure 2-7. Execution of any Branch-and-Link instruction replaces the contents of bits 13 through 30 of the PSD with the effective address specified by the instruction. In addition, if the Branch instruction specifies an Indirect Branch operation, the contents of bits 1 through 4 of the PSD are replaced by the contents of the corresponding bit positions in the indirect address location.

CONDITION CODES

A four-bit Condition Code is stored in the PSD upon completion of the execution of most instructions. These conditions may be tested to determine the status of results obtained.

CC1 is set if an Arithmetic Exception occurs

CC2 is set if the result is greater than zero

CC3 is set if the result is less than zero

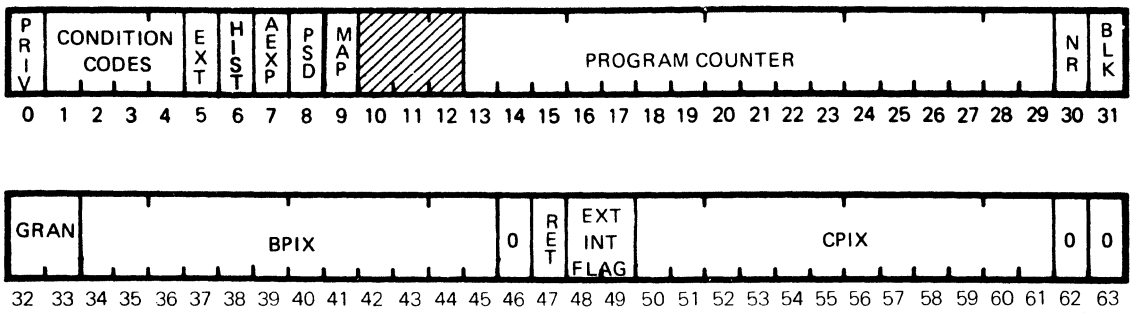
CC4 is set if the result is equal to zero

The Branch Condition True (BCT), Branch Condition False (BCF), and the Branch Function True (BFT) instructions allow testing and branching of the condition codes.

PRIVILEGED AND UNPRIVILEGED OPERATION

The IPU is capable of either privileged or unprivileged operation in the PSD mode. Privileged operation allows the IPU to perform all of its control functions and to modify any part of the system; it relates to changes in the basic control state of the computer. Unprivileged operation is the problem-solving mode of the IPU. In this mode, memory protection is in effect and all privileged operations are prohibited.

One bit in the Program Status Doubleword (PSD) is designated as the Privileged State bit. If the Privileged State bit is set, privileged instructions can be executed. If the Privileged State bit is reset, any attempt to execute a privileged instruction will cause a Privilege Violation trap.



BIT 0 = 0 UNPRIVILEGED MODE
= 1 PRIVILEGED MODE

BITS 1-4 ARE CONDITION CODES
BIT 1 = CC1
2 = CC2
3 = CC3
4 = CC4

BIT 5 = 0 EXTENDED MODE (OFF) CEA
= 1 EXTENDED MODE (ON) SEA

BIT 6 = 0 LAST INSTRUCTION EXECUTED WAS NOT A RIGHT HALFWORD
= 1 LAST INSTRUCTION EXECUTED WAS A RIGHT HALFWORD

BIT 7 = 0 ARITHMETIC EXCEPTION TRAP MASK (OFF)
= 1 ARITHMETIC EXCEPTION TRAP MASK (ON)

*BIT 8 = 0 COMPUTER IS IN PSW MODE (DISPLAYED PSD ONLY) * (PSW MODE NOT USED BY IPU)
= 1 COMPUTER IS IN PSD MODE (DISPLAYED PSD ONLY) *

*BIT 9 = 0 UNMAPPED (DISPLAYED PSD ONLY) *
= 1 MAPPED (DISPLAYED PSD ONLY) *

BITS 10-12 ARE NOT USED
BITS 13-29 ARE LOGICAL WORD ADDRESS
BIT 30 NEXT INSTRUCTION IS A RIGHT HALFWORD
*BIT 31 BLOCKED (DISPLAYED PSD ONLY) *
BITS 32-33 INDICATE MAP GRANULARITY, 00=UNMAPPED AND ALL OTHERS =8K MAP GRANULARITY
BITS 34-45 PROVIDE A WORD INDEX INTO THE MASTER PROCESS LIST (MPL) FOR THE BASE PROCESS
BIT 46 NOT USED
BIT 47 RETAIN CURRENT MAP CONTENTS
BITS 48-49 INTERRUPT CONTROL FLAGS

BITS		
48	49	
0	0	OPERATE WITH UNBLOCKED INTERRUPTS
0	1	OPERATE WITH BLOCKED INTERRUPTS
1	0	RETAIN CURRENT BLOCKING MODE
1	1	RETAIN CURRENT BLOCKING MODE

BITS 50-61 PROVIDE WORD INDEX INTO MASTER PROCESS LIST (MPL) FOR CURRENT PROCESS
BITS 62-63 NOT USED

* THESE BITS ARE USED FOR DISPLAY ONLY AND ARE NOT PRESENT IN THE PSD STORED IN MEMORY.

Figure 2-7. Program Status Doubleword (PSD) Format

Table 2-2. PSD Mode (IPU)

Characteristic	PSD Mode
Program Status	Doubleword
Number of instructions	163
Integrity Features	Traps
Memory Addressing	
Nonmapped	
Nonextended	512 KB+
Extended	16 MB+
Mapped	
Nonextended	512 KB per user
Extended	16 MB per user
+ No software support	

The following IPU instructions are privileged:

1. All instructions that can modify the memory mapping registers.
2. All instructions that can place the machine in a state that requires CPU intervention to continue processing, such as Halt.
3. All instructions that modify Writable Control Storage.

Certain events can change the processor from the unprivileged to the privileged state by loading a new Program Doubleword. These are:

- A hardware trap caused by addressing nonpresent memory, executing undefined instruction, executing a privileged instruction by a nonprivileged program, or writing to protected memory.
- A hardware trap caused by a nonrecoverable condition such as an uncorrectable error on a memory read, or an arithmetic exception.
- The execution of the Call Monitor or Supervisor Call instruction by a user requesting monitor services.

As long as traps are set they are vectored to monitor routines for proper handling. The trap vectors and the monitor service routines are in protected memory. This insures that an unprivileged user has no way to become privileged or to alter protected state.

The execution of the Load Program Status Doubleword (LPSD) instruction can cause the system to change from the privileged to the unprivileged state.

The operator can depress the SYSTEM RESET pushbutton to initialize a 32/70 SERIES computer and IPU. SYSTEM RESET clears the eight general purpose registers, resets all memory protection, and sets the Privileged State bit.

IPU ADDRESSING MODES

1. 512-KB mode
2. 512-KB Extended mode
3. 512-KB Mapped mode
4. Mapped, Extended mode

512-KB MODE

The 512-KB addressing mode allows the IPU to access instructions or operands (bit, byte, halfword, word, or doubleword) in the first 512K bytes of memory directly without mapping, indexing, or address modification. A 19-bit address field is provided in memory referencing instructions for that purpose.

Bit addressing is accomplished by using the register (R) field in the instruction word to select a bit in the byte specified by the 19-bit address. Therefore, any bit in the first 512K bytes of memory can be directly manipulated by the Bit Manipulation instructions.

512-KB EXTENDED MODE

The 512-KB Extended mode provides the same capabilities as the 512-KB mode described above, and, in addition, it permits operand addressing only beyond the first 512K bytes of memory. The effective address can reference any bit, byte, halfword, word, or doubleword residing anywhere within 16 megabytes of physical memory.

512-KB MAPPED MODE

The 512-KB Mapped mode allows the IPU to access any instruction or operand (bit, byte, halfword, word, or doubleword) within a logical primary address space. This space consists of 512K bytes of logical memory map, distributed within 16 megabytes of physical memory.

The IPU allows multiple primary address spaces. A user can access instructions and operands within the logical primary address space in which his program resides. Physical blocks of memory can be common to many logical primary address spaces; thus, users in different spaces can share common blocks of memory.

MAPPED EXTENDED MODE

The Mapped Extended mode provides all the capabilities of the 512 KB Mapped mode, plus access to a logical extended operand address space. This space consists of 512K bytes of mapped memory beyond the logical primary address space and allows users additional memory space to store data (operands). Each logical extended operand address space can be 512K bytes long, dispersed anywhere within 16 megabytes of physical memory. The combination of logical primary address space and the logical extended operand address space supports programs up to one megabyte long. The executable code must lie within the logical primary address space, but operands can be in either the logical primary or extended operand address space.

FUNCTIONAL DESCRIPTION

MAJOR SYSTEM ELEMENTS

The major elements of a typical SYSTEMS 32/70 SERIES Computer System with an IPU include: 32/70 CPU, the SelBUS, Real-Time Option Module, main memory, input/output system (not supported by the IPU), Serial Control Panel, optional High-Speed Floating Point, and Scientific Accelerator modules. (See Figure 2-8 for a system block diagram.) The performance gains of a CPU and IPU system over a CPU alone system is application dependent. The IPU allows the user to run two tasks simultaneously in the computer system.

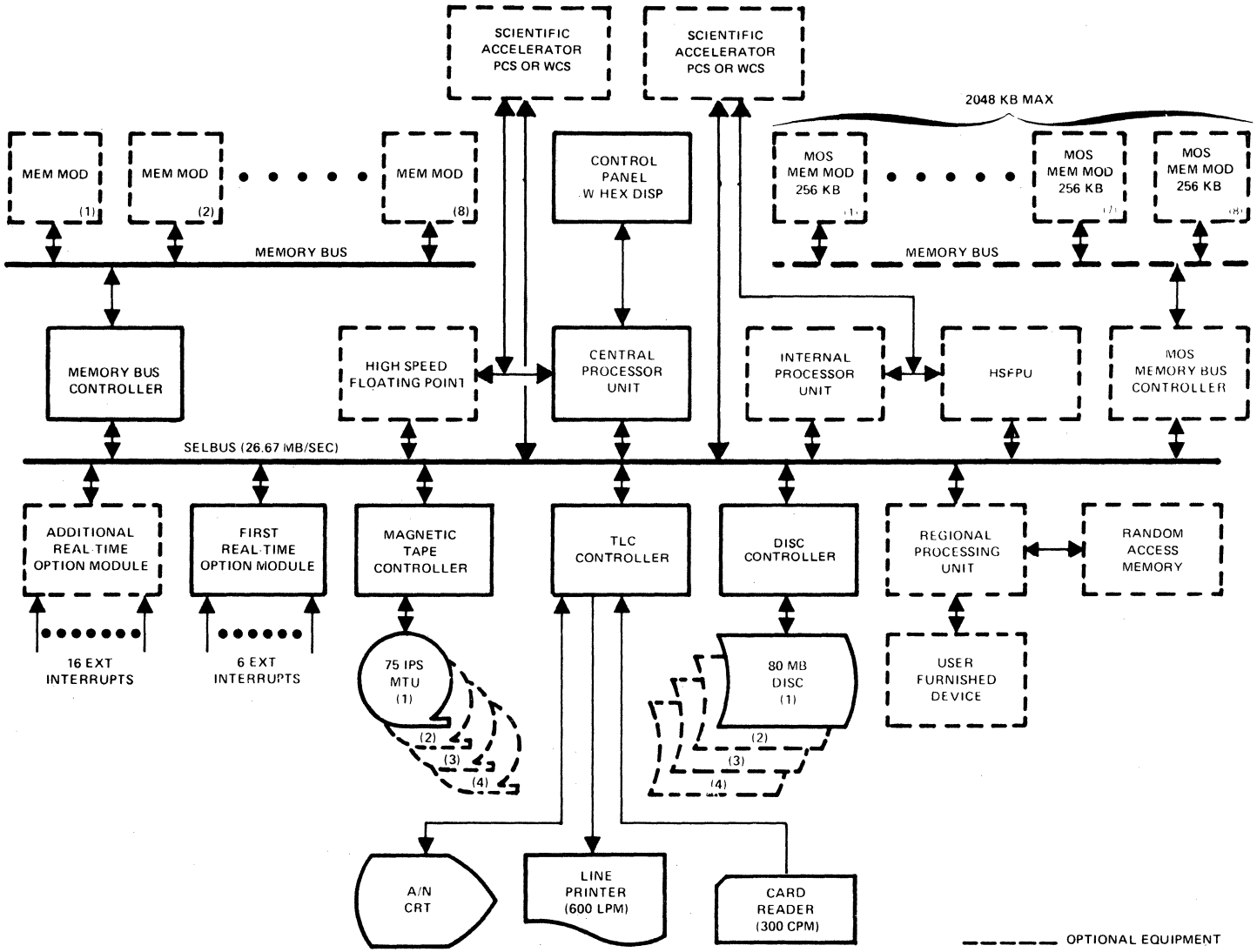


Figure 2-8. System Block Diagram

Central Processing Unit

The CPU in the system plays the dominant role in its relationship with the IPU. The CPU is responsible for all task scheduling I/O and system services as well as for the execution of its own scheduled tasks.

IPU Major Elements

A brief description of some major elements of the IPU are provided in the paragraphs that follow. They include: the data structure, a microprogrammable processor, the implementation logic, and the SelBUS interface. A simplified block diagram of the IPU is shown in Figure 2-9.

IPU Data Structure

The data structure contains the eight general purpose file registers and ten hardware registers organized around an Arithmetic Logic Unit (ALU). Key circuits in the data structure include the following:

1. Arithmetic Logic Unit (ALU)
2. A Multiplexer
3. B Multiplexer
4. Literal Multiplexer
5. General File Register
6. Memory Address Register
7. Program Counter Register
8. N Counter Register
9. Shift Register
10. Temporary Register/Data Output Register
11. Data Input Register
12. Instruction Register 0
13. Instruction Register 1

IPU Microprogrammable Processor

The Microprogrammable Processor of the IPU is on board C of the three IPU circuit boards. The logic circuit board, which contains the Microprogrammable Processor, is commonly referred to as the personality board.

The Microprogrammable Processor utilizes Read-Only Memory (ROM) integrated circuits which house the IPU's Elementary Operations (EO). The EOs, with the associated circuitry, control the IPU operations by testing, controlling, and directing the various functions to be performed. The format for the EOs (also referred to as microinstruction) is shown in Figure 2-10.

Implementation Logic

The Implementation Logic includes the ALU Decode PROM, a Scale circuit, the Floating-Point Assist PROMs, and a Multiply Assist PROM, all of which serve to implement IPU functions.

SelBUS Interface

The SelBUS interface logic is implemented on all three of the IPU circuit boards and provides control and temporary storage for information being output to and input from the SelBUS. Since the SelBUS is the high-speed communication link between system modules external to the IPU, the SelBUS interface logic plays a vital role in IPU operation.

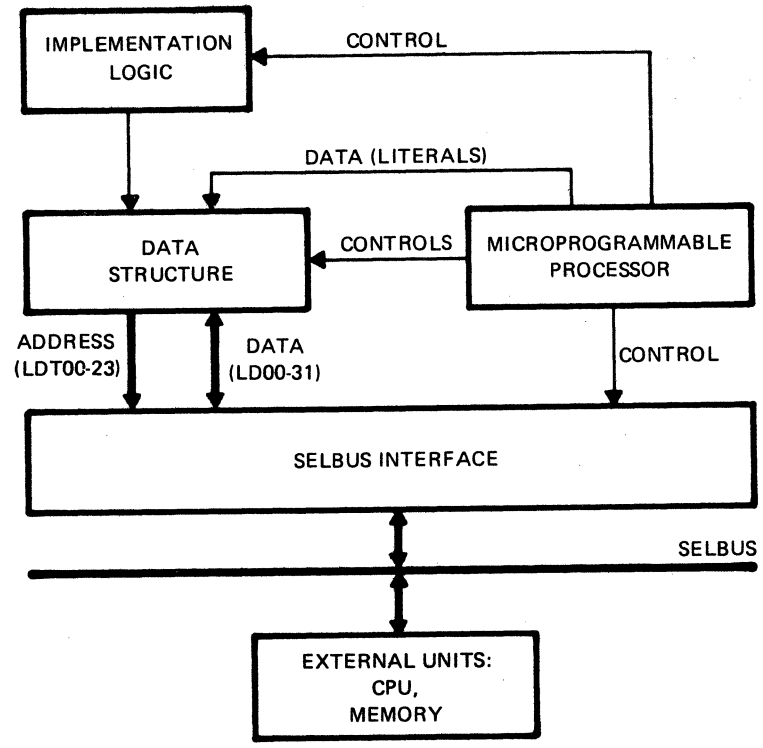


Figure 2-9. IPU Simplified Block Diagram

OPTIONAL HIGH-SPEED FLOATING-POINT UNIT

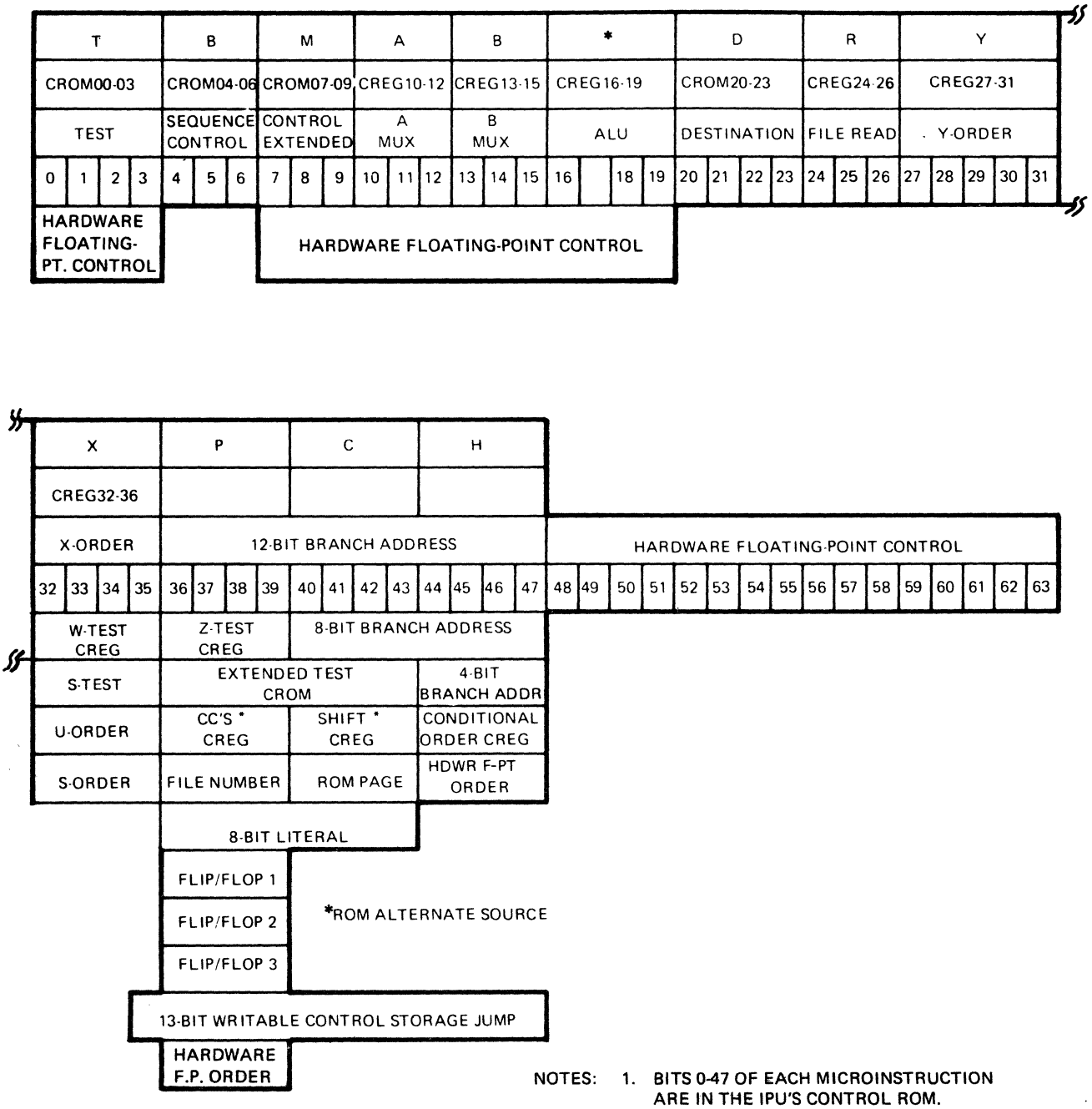
The High-Speed Floating-Point Unit (FPU) is an option that may be used with an IPU and CPU to increase the speed of floating-point arithmetic operations. The unit consists of two circuit boards which may be plugged in adjacent to the IPU. No alterations in the software are required.

If the High-Speed Floating-Point Unit (FPU) is installed, addition, subtraction, multiplication, and division of single-precision (32-bit) or double-precision (64-bit) operands can be executed much faster than the IPU's standard floating-point feature.

An operand in floating-point format has three parts: a sign bit, a fraction, and an exponent. The sign bit indicates whether the fraction is a positive or negative value. The fraction is a binary number with an assumed radix point immediately to the left of its most-significant bit. The exponent is a 7-bit binary power to which the base 16 is raised. The quantity that the floating-point number represents can be determined by multiplying the fraction by the number 16 raised to the power represented by the exponent.

Two operands of the same format and length are received by the FPU for each arithmetic operation. One operand is input from an IPU general purpose register (GPR), whereas the other operand is input from memory. The precise GPR and memory location are specified in the floating-point instruction. Upon completion of an operation, the result is returned to the CPU general purpose register.

It is recommended that any option which is added to the system be for both the IPU and CPU. This will allow the same runtime library to be utilized for the respective software programs.



NOTES:

1. BITS 0-47 OF EACH MICROINSTRUCTION ARE IN THE IPU'S CONTROL ROM.
2. PORTIONS OF THE FORMAT DESIGNATED FOR HARDWARE FLOATING-POINT APPLY TO THE OPTIONAL HIGH-SPEED FLOATING-POINT UNIT (FPU).
3. BITS 48-63 ARE PHYSICALLY PART OF A CONTROL ROM IN THE OPTIONAL HIGH-SPEED FPU.

Figure 2-10. Microinstruction Format

Figure 2-11 illustrates the major functional elements of the FPU, the general routing of operands, and the relationship between the FPU, the IPU, and the SelBUS.

OPTIONAL SCIENTIFIC ACCELERATOR

The optional Model 2343 Scientific Accelerator (with PROM control store PCS), or Model 2347 with Writable Control (WCS), may be used in the IPU system. The Scientific Accelerator provides fully-integrated hardware, software, and firmware to improve user program execution speeds.

It is recommended that any option which is added to the system be for both the IPU and CPU. This will allow the same runtime library to be utilized for the respective software programs.

OPTIONAL WRITABLE CONTROL

Writable Control Storage is an option which may be used with the IPU to expand the instruction set, to enhance the performance of user programs, or to tailor the computer to specific user needs.

Up to 4,096 64-bit words of Writable Control Storage (WCS) can be added to an IPU in increments of 2,048 64-bit words. Each increment plugs into the SelBUS for power and clock. However, communication with the IPU is independent of SelBUS operation.

The block diagram in Figure 2-12 shows two optional WCS units as they could be implemented in conjunction with an IPU and the optional High-Speed Floating-Point Unit.

TRAPS

NEW TRAPS

For synchronization and communication between the IPU and CPU, six new traps are dedicated in low memory. These traps are listed in Table 2-3. The trap vector location 2E0 is used by the CPU when the IPU has executed the SIPU (X'000A') instruction. The CPU handles this trap in the same manner as any other CPU trap. The trap vectors found at locations 2E4, 2E8, 2EC, 2F0, and 2F4 are traps used by the IPU during IPU processing. A brief description of the communications between the IPU and CPU follows later in this section and involves primarily the use of new traps.

OPERATING MODE

The IPU uses the Program Status Doubleword Mode of operation to run the Mapped Programming Executive, MPX-32. This mode identifies the firmware routing and method of handling traps.

In the IPU mode, the firmware will not execute control instruction BRI, any I/O instructions, nor interrupt control instructions except UEI.

TRAP CONTEXT SWITCHING

Trap Context Switching in the IPU is accomplished through the use of the Program Status Doubleword Mode using the Trap Context Block (TCB) format. Trap context switching by the IPU is functionally identical to the CPU, except that the trap entry by the IPU is not associated with a service interrupt.

Trap Format

The Trap Context Block (TCB) format type (see Figure 2-13) is used for the PSD mode traps. Words one through four of the TCB contain the IPU Ending and Starting PSDs. Word five of the TCB contains the IPU Hardware Status Word. This status word is assembled by firmware at the time the trap occurs, and is stored in the TCB. The IPU Hardware Status Word is defined later in this section. Word 6 of the TCB is not used.

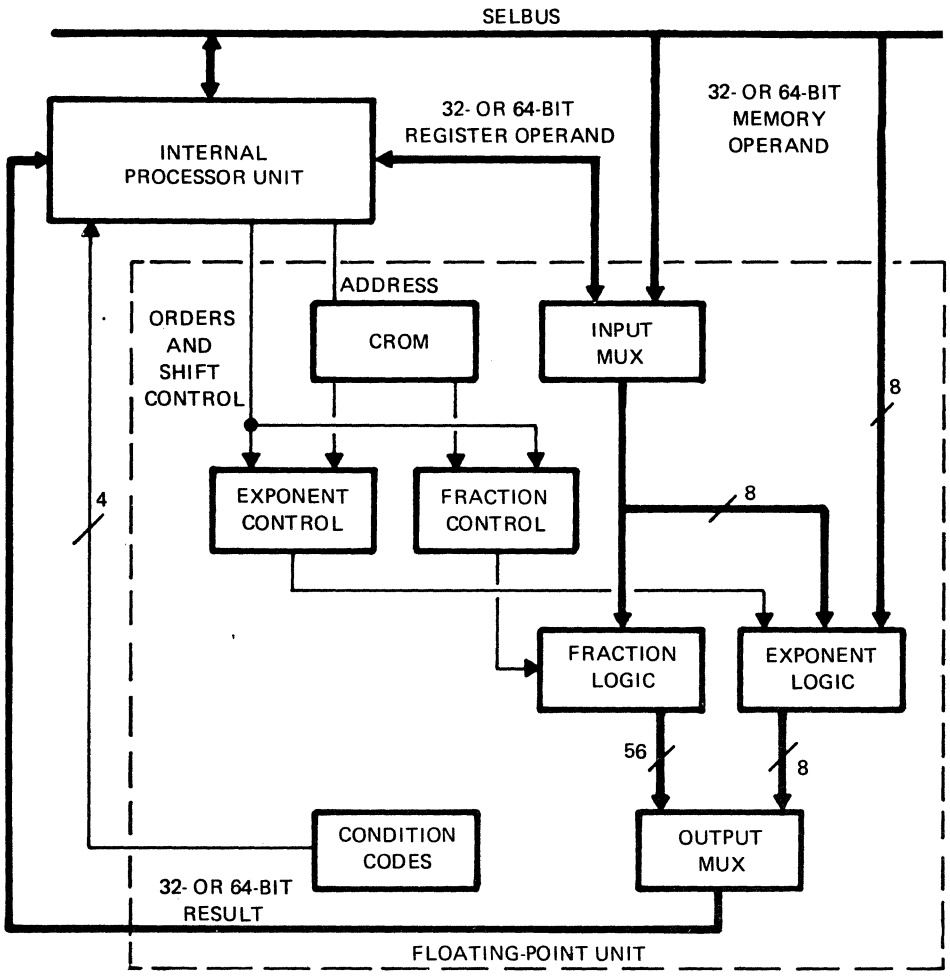


Figure 2-11. Optional High-Speed Floating-Point Unit

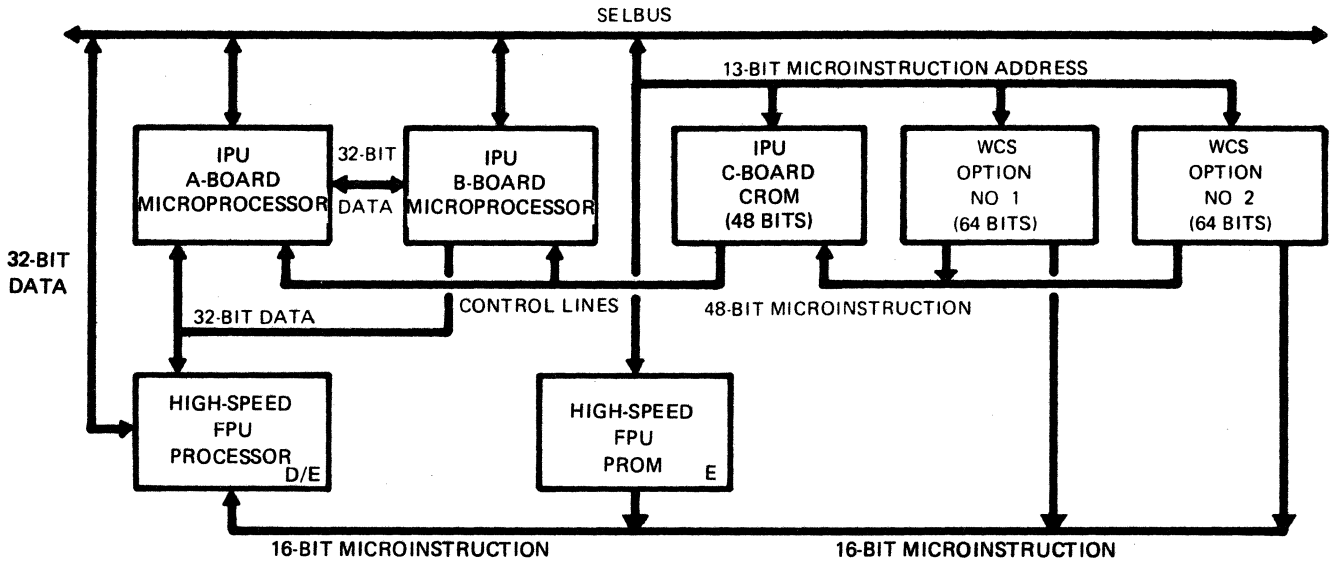


Figure 2-12. Functional Interrelationship of the IPU, WCS, and High-Speed Floating-Point Unit

Table 2-3. CPU/IPU Communication Traps

Trap Relative Priority	Trap Vector Location TVL	Description	User
78	2E0	Ending of IPU Processing	CPU
79	2E4	Start IPU Processing	IPU
7A	2E8	Supervisor Call	IPU
7B	2EC	Error Trap	IPU
7C	2F0	Call Monitor	IPU
7D	2F4	Stop IPU Processing	IPU

IPU STATUS WORD

The IPU status word is a 32-bit word that is used by IPU firmware to track trap error processing and internal operating modes. The status word is available to software in either of two methods as follows:

1. The Read Status (RDSTS) instruction (when executed by the IPU) causes the status word to be loaded into the general purpose register specified by the instruction.

2. Automatically, upon occurrence of an error trap which causes the status word to be stored in the fifth word of the trap context block.

The status word can be divided into a 24-bit field and an 8-bit field. The 24-bit field is used for error flag reporting and is cleared to zeros after the status word has been reported to software. The 8-bit field of the status word is used for IPU mode control and will always reflect the current operating mode of the IPU. Table 2-4 lists the bits of the status word and their definitions.

CPU/IPU INTERFACE OPERATION

The following discussion provides information pertaining to the CPU and IPU interface operation. This discussion is centered primarily around the use of the six new traps, which are used to control the synchronization and communication between the CPU and IPU. The basic interface operational flow between the CPU and IPU is shown in Figure 2-14.

START IPU TRAP (VECTOR ADDRESS 2E4)

To start IPU processing, the CPU stores the new Program Status Doubleword (PSD) into words 3 and 4 of the Start IPU trap context block which was pointed to from the address contained in the Start IPU trap vector location 2E4. The CPU then executes the SIPU X'000A' instruction which sends a start signal to the IPU and informs the IPU that a new PSD is available for execution. The IPU then fetches the Start IPU trap Context Block pointer at the 2E4 trap location, stores the old PSD into words 1 and 2 of the Start IPU Trap Context block and the IPU Status into word 5. The IPU then reads the new PSD words 3 and 4 from the context block and begins to execute the instructions in memory as directed by the new PSD.

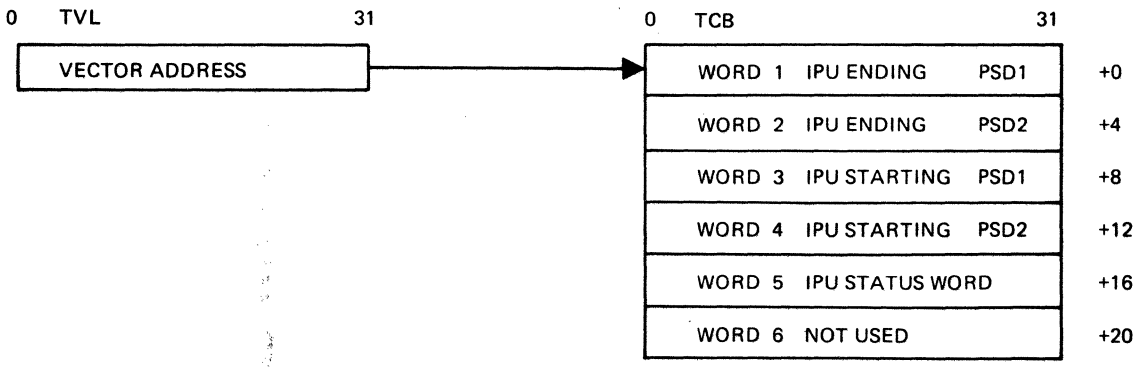


Figure 2-13. Trap Context Block Format (Internal Processing Unit)

RESTART IPU

If the Signal IPU instruction (SIPU) is issued by the CPU to an active IPU, the second SIPU will cause the following events in the IPU to occur:

1. The IPU will terminate present active execution, and vector to the start IPU Trap Vector Address (TVA) 2E4. The old PSD is stored into Words 1 and 2 of the Context Block as was pointed by the contents of the TVA.

2. The old IPU status word is stored into the context block and the new PSD is used to begin execution of another group of 32/70 macro-assembler instructions.

The End of IPU trap is not generated until the IPU has completed execution as directed by the interrupting SIPU instruction.

IPU ERROR CONDITION TRAP (VECTOR ADDRESS 2EC)

The vector address found at memory location 2EC points to the TCB which is used upon the occurrence of an error condition within the IPU. The error conditions include non-present memory, undefined instruction, parity error, arithmetic exception, and privilege violation. The undefined instruction error is caused by the execution of any I/O instruction (e.g., CD, TD), any interrupt instruction (BRI, AI), or any instruction not defined in the PSD mode 32/75 instruction set.

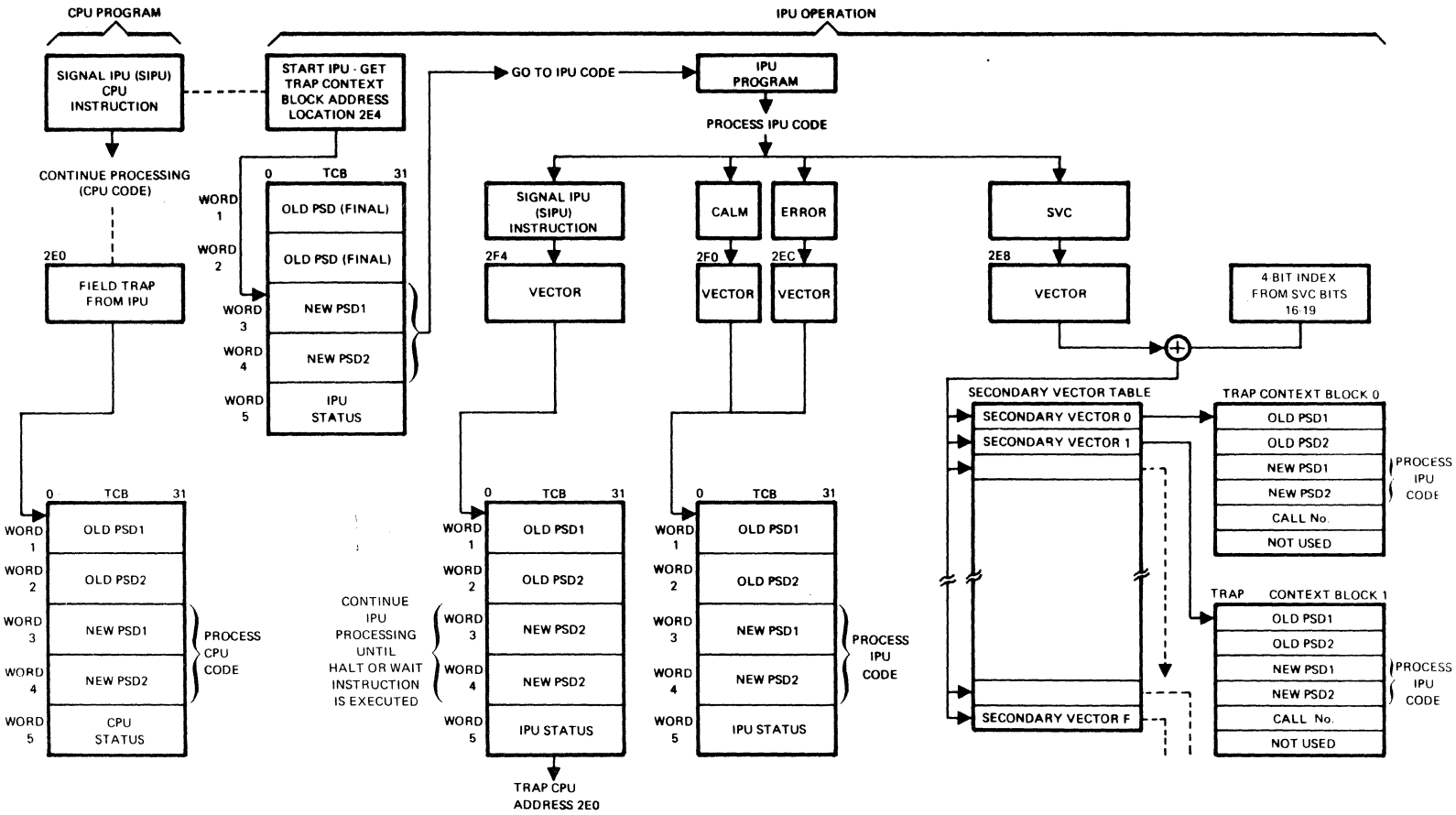


Figure 2-14. CPU/IPU Interface Operational Flow

Table 2-4. IPU Status Word Bit Definitions

Bit	Definition
0	=0, Class 0, 1, 2, or E Error* =1, Class F (Extended I/O) Error*
1	=0, I/O Processing Error* =1, Interrupt Processing Error*
2	Final Bus Transfer Error
3	Bus No Transfer Error
4	I/O Channel Busy or Busy Status Bit Error*
5	Ready Timeout Error*
6	I/O DRT Timeout Error*
7	Retry Count Exhausted Error*
8	Operand Fetch Parity Error
9	Instruction Fetch Parity Error
10	Operand Nonpresent Error
11	Instruction Nonpresent Error
12	Undefined PSD Mode Instruction Error
13	Memory Fetch DRT Timeout Error
14	Reset Channel Error*
15	Channel WCS not Enabled Error
16	Map Register Address Overflow (Map Context Switch)
17	Unexplained Memory Error
18	BRI I/O Error*
19	Undefined Instruction PSW Mode Only*
20	Map Invalid Access or Map Mode Restrict Register
21	IPU Privileged Violation
22	Not Used
23	IPU Arithmetic Exception
24	Enable Arithmetic Exception Trap
25	Disable PSD Mode Traps
26	Block Mode is Active
27	IPU Status
28	Not Used
29	CPU ELSA Mode*
30	Not Used
31	=1, IPU Mode PSD

*Not Applicable to IPU.

The privilege violation error is generated by the IPU attempting to execute an instruction which is defined as privileged but does not have the privileged bit set in the PSD. HALT and WAIT in the IPU must be privileged.

The error status is reflected in the IPU status word as stored into the fifth word of the IPU error TCB (vector location 2EC). The PSD at the time the error occurs is stored into words 1 and 2 of that TCB. The next executed instruction is dictated by the new PSD found in words 3 and 4 of the error TCB.

IPU CALL MONITOR TRAP (VECTOR ADDRESS 2F0)

When the IPU executes a Call Monitor (CALM) instruction, control is transferred to the IPU call monitor trap located at memory address 2F0. The execution which follows the call monitor instruction, as well as any other trap within the IPU, is directed by the contents of the context block related to that specific trap. Execution is directed to the code as defined by the new PSD within the IPU CALM Trap Context Block.

IPU SUPERVISOR CALL TRAP (VECTOR ADDRESS 2E8)

When the IPU executes a Supervisor Call (SVC) instruction, control is transferred to the SVC trap. The address of the context block for the IPU service of a SVC instruction is located at trap address 2E8.

This address is the beginning of the 16-entry secondary vector address table. Bits 16 through 19 of the SVC instruction direct the IPU to one of the 16 secondary vector addresses. The secondary vector address selected points the IPU to a TCB for that SVC.

Once the IPU has the address of the TCB, trap processing is handled as a normal trap. The IPU stores the present PSD into words 1 and 2 of the TCB and the status into word 5. Then the IPU uses the new PSD from words 3 and 4 to continue execution.

STOP IPU TRAP VECTOR ADDRESS 2F4

To stop the IPU processing, the CPU stores a new PSD in words 3 and 4 of the STOP IPU Trap Context Block (TCB) which is pointed to by the address contained in the stop IPU trap vector location 2F4. The STOP IPU TCB is used when the IPU executes an SIPU (X'000A') instruction which is imbedded in the IPU software code. The IPU stores the old PSD into words 1 and 2 of the context block and the IPU Status into word 5 of the context block. The IPU then traps the CPU at location 2E0 which indicates that the IPU execution of the SIPU instruction has taken place. The IPU then fetches the new PSD from words 3 and 4 of the context block which can point to a privileged HALT or WAIT instruction to stop the IPU.

The new PSD in the STOP IPU context block may direct the IPU to execute code other than a HALT or WAIT instruction. This utilization of the STOP Trap allows the IPU to signal the CPU at milestones without stopping IPU execution. In either use of this stop IPU trap, the present PSD is stored into words 1 and 2 of TCB and the present IPU status into word 5. The IPU done signal is sent to the CPU after storage of the present PSD and IPU status word and before vectoring to the new PSD address.

CPU (END IPU PROCESSING) TRAP (VECTOR ADDRESS 2E0)

The End IPU Processing trap address 2E0 is used by the CPU when the IPU generates the IPU done signal. The CPU handles this trap in the same manner as any other CPU Trap, except that this trap can be blocked at the CPU by setting the block mode.

MEMORY MANAGEMENT

All information as presented in Section IV for the Memory Management is valid for the IPU.

INPUT/OUTPUT SYSTEM

The Internal Processing Unit does not perform I/O operations. All I/O operations are performed by the CPU in the system.

SCRATCHPAD MEMORY

Except for the Scratchpad locations related to I/O and interrupts, the IPU utilizes the internal scratchpad in the same manner as the CPU uses its internal scratchpad.

The scratchpad locations loaded by the IPL are not used by the IPU. Thus, no loading procedure is necessary. The IPU can execute the TRSC and TSCR instructions if the user deems it necessary to load or read scratchpad locations.

INITIALIZATION

INTRODUCTION

The Internal Processing Unit is initialized by a system reset and remains quiescent until a Signal IPU (SIPU) Instruction occurs.

INITIAL PROGRAM LOAD

The IPU does not perform an IPL. This procedure is controlled by the CPU and the I/O device. Refer to Section VIII for CPU-IPL operation.

POWER FAIL-SAFE FEATURE

The Power Fail Safe feature as implemented in the CPU is not applicable to the IPU operation. The saving of the IPU scratchpad information is not necessary by the IPU since the CPU must re-initialize any IPU operation when the CPU is restarted.

SECTION III

TRAPS AND INTERRUPTS

INTRODUCTION

Traps and interrupts report asynchronous or synchronous events to the software. Traps are error conditions that are generated internally and interrupts are requests that are generated externally. The events that caused the trap or interrupt can be generated asynchronously by hardware or synchronously scheduled by software when an interrupt control instruction is executed. The trap or interrupt causes a transfer of control to unique vector locations in main memory (see Table 3-1).

TRAPS

The traps for the PSW mode (in order of priority) are:

1. Power Fail
2. Memory Parity
3. Nonpresent Memory
4. Undefined Instruction
5. Privileged Violation
6. System Override

Six additional traps are present in the PSD mode. They are:

1. Supervisor Call Trap (software generated)
2. Machine Check Trap
3. System Check Trap
4. MAP Fault Trap
5. Block Mode Timeout (Watchdog) Trap
6. Arithmetic Exception Trap
7. End of IPU processing

INTERRUPTS

Interrupts consist of the following:

1. Any external event scheduled through the Real-Time Option Module (RTOM)
2. Input/Output (I/O) termination interrupts
3. Software request interrupt control instruction

OPERATING MODES

The 32/70 Series CPU is capable of operating in two modes: the PSW mode and the PSD mode. The two modes identify the firmware routing required to operate with a PSW, thereby allowing existing 32/55 software to operate on a 32/70 Series CPU without modifications. The PSD mode is the default at system reset and remains in effect until a Set CPU Mode macro instruction is executed or an Initial Program Load (IPL) sequence is set up to force the CPU into PSW mode of operation.

Table 3-1. PSW/PSD Mode Relative Trap/Interrupt Priorities

INTERRUPT AND TRAP RELATIVE PRIORITY	INTERRUPT LOGICAL PRIORITY	INTERRUPT VECTOR LOCATION (IVL)	TCW ADDRESS **	IOCD ADDRESS **	DESCRIPTION
00		0F4			Power Fail Safe Trap
01		0FC			System Override Trap (Not used)
02		0E8*			Memory Parity Trap
03		190			Nonpresent Memory Trap
04		194			Undefined Instruction Trap
05		198			Privilege Violation Trap
06		180			Supervisor Call Trap
07		184			Machine Check Trap
08		188			System Check Trap
09		18C			MAP Fault Trap
0A					Not Used
0B					Not Used
0C					Not Used
0D					Not Used
0E		0E4			Block Mode Timeout (Watchdog) Trap
0F		1A4*			Arithmetic Exception Trap
10	00	0F0			Power Fail Safe Interrupt
11	01	0F8			System Override Interrupt
12	12	0E8*			***Memory Parity Trap
13	13	0EC			Attention Interrupt
14	14	140	100	700	I/O Channel 0 Interrupt
15	15	144	104	708	I/O Channel 1 Interrupt
16	16	148	108	710	I/O Channel 2 Interrupt
17	17	14C	10C	718	I/O Channel 3 Interrupt
18	18	150	110	720	I/O Channel 4 Interrupt
19	19	154	114	728	I/O Channel 5 Interrupt
1A	1A	158	118	730	I/O Channel 6 Interrupt
1B	1B	15C	11C	738	I/O Channel 7 Interrupt
1C	1C	160	120	740	I/O Channel 8 Interrupt
1D	1D	164	124	748	I/O Channel 9 Interrupt
1E	1E	168	128	750	I/O Channel A Interrupt
1F	1F	16C	12C	758	I/O Channel B Interrupt
20	20	170	130	760	I/O Channel C Interrupt
21	21	174	134	768	I/O Channel D Interrupt
22	22	178	138	770	I/O Channel E Interrupt
23	23	17C	13C	778	I/O Channel F Interrupt
24	24	190*			***Nonpresent Memory Trap
25	25	194*			***Undefined Instruction Trap
26	26	198*			***Privilege Violation Trap
27	27	19C			Call Monitor Interrupt
28	28	1A0			Real-Time Clock Interrupt
29	29	1A4*			***Arithmetic Exception Interrupt
2A	2A	1A8			External/Software Interrupts
2B	2B	1AC			External/Software Interrupts
2C	2C	1B0			External/Software Interrupts
2D	2D	1B4			External/Software Interrupts
2E	2E	1B8			External/Software Interrupts
2F	2F	1BC			External/Software Interrupts
30	30	1C0			External/Software Interrupts
31	31	1C4			External/Software Interrupts
THRU	THRU	THRU			THRU
77	77	2DC			External/Software Interrupts

* Vector Locations Shared With Traps
** For Nonextended I/O Devices
*** PSW Function-Now External/Software Interrupts-For PSD Mode
**** IPU Related Traps
All Interrupts Are Externally Generated

Table 3-1. PSW/PSD Mode Relative Trap/Interrupt Priorities (Cont'd)

INTERRUPT AND TRAP RELATIVE PRIORITY	INTERRUPT LOGICAL PRIORITY	INTERRUPT VECTOR LOCATION (IVL)	TCW ADDRESS **	IOCD ADDRESS **	DESCRIPTION
78		2E0****			Ending of IPU Processing Trap (Used by CPU)
79		2E4****			Start IPU Processing Trap (Used by IPU)
7A		2E8****			Supervisor Call Trap (Used by IPU)
7B		2EC****			Error Trap (Used by IPU)
7C		2F0****			Call Monitor Trap (Used by IPU)
7D	7D	2F4****			Stop IPU Processing Trap (Used by IPU)
7E	7E	2F8			External/Software Interrupts
7F	7F	2FC			External/Software Interrupts

** For Nonextended I/O Devices
**** IPU Related Traps (See Section II)

All Interrupts Are Externally Generated

PSW MODE

The PSW mode identifies traps and interrupts on a prioritized, scheduled basis No distinction is made between traps and interrupts, and both are scheduled by some mechanism external to the CPU (i.e., IOM or RTOM). The trap conditions that are created internally within the CPU are scheduled by the firmware on an RTOM board if the following requirements are met:

1. Trap level is enabled.

2. Trap level is not active.

3. Any other higher priority level is not active or requesting.

If any of the above requirements are not met, the firmware will reset the condition that caused the trap and continue to the next sequential instruction as if the trap never occurred.

Traps and interrupts in the PSW mode require the participation of three component levels in order to function properly. The three component levels are the IOM or RTOM, the CPU, and the software.

The IOM or RTOM schedules a hardware- or software-initiated interrupt service request. When the requesting level becomes the highest contending level, the CPU acknowledges the interrupt request. In order to enqueue the associated software processing, the IOM or RTOM advances from requesting to active, blocking interrupt requests from lower priority levels. When the software interrupt handler completes its processing, the software dequeues itself by executing a Deactivate Interrupt (DAI) or Branch and Reset Interrupt (BRI) instruction which allows the currently active level and all other lower priority levels to resume requesting for interrupts. This operating mode is also referred to as Block with Activate. In summary, the six steps shown below are required to enqueue or dequeue an interrupt process:

1. The IOM, RPU, or RTOM requests an interrupt.

2. The CPU acknowledges the interrupt.

3. The IOM or RTOM goes active, blocking lower priority interrupts.

4. The software handler is given control. (First instruction is noninterruptible)

5. The software executes a Deactivate Interrupt (DAI) or Branch and Reset Interrupt (BRI).

6. The IOM or RTOM deactivates, allowing lower priority levels to resume requesting.

PSD MODE

Two types of software trap and interrupt queueing methods exist in the PSD mode. The first method is identical to the queueing described as the PSW mode, where the requesting level advances to active state, blocking all lower priority levels to insure that software is not interruptible by its level or any lower priority levels during the interrupt processing. This method applies to all classes of I/O interrupts and external (RTOM) interrupts.

The second method applies to traps, I/O interrupts and external interrupts. The enqueueing of the software interrupt and trap handlers does not rely on the active state of the applicable channel or RTOM to prevent interrupts or traps for the specific or lower priority levels. The enqueueing function blocks externally generated interrupt requests (channel or RTOM) from being sensed by the CPU firmware. Software must now explicitly dequeue its process with an Unblock External Interrupts (UEI) or a Load PSD (LPSD) macro instruction. The general sequence is:

1. The IOM, RPU, or RTOM requests an I/O interrupt.

2. When the requesting level becomes the highest contending level, the CPU acknowledges the interrupt request and blocks all interrupts until the UNBLOCK command is received (if bits 48 and 49 of the PSD are 0 and 1, respectively).

3. The channel does not go active and is now free to continue I/O related processing.

4. The software is given control with all interrupts blocked.

5. When the software interrupt handler completes its enqueued processing, it will execute an Unblock External Interrupt (UEI) or a Load Program Status Doubleword (LPSD) macro instruction which will allow externally generated interrupts to be sensed by the CPU firmware. This operating mode is also referred to as Block without Activate.

IVL AND ICB

Each trap or interrupt that may occur in the PSD mode has an associated Interrupt Vector Location (IVL) and an Interrupt Context Block (ICB). The IVL contains a 24-bit real address that points to the starting memory address of the ICB. Table 3-1 includes a list of the memory locations dedicated for IVLs.

ICB FORMATS

Generally speaking, an ICB consists of six consecutive memory words. However, for some types of ICBs only four or five words are required. The four different ICB formats are listed as follows:

1. External and Non-Class F I/O Format
2. Trap Format
3. Class F I/O Format
4. Supervisor Call Format

Figures 3-1 through 3-4 illustrate the four ICB formats.

OLD AND NEW PSD

The first four words of all ICB formats are identical in that they contain the old PSD followed by the new PSD.

The old PSD is stored in the ICB whenever a trap or interrupt occurs and is acknowledged. The old PSD locations provide storage for hardware and software CPU context information current at the time a particular trap or interrupt occurs. Normally, when the software interrupt processing is completed, a BRI, LPSD or LPSDCM instruction will be used to restore the old PSD context information.

The new PSD information must be loaded in the ICB by software before a trap or interrupt occurs. The new PSD must contain the necessary information to set up the hardware and software in the appropriate context for servicing the interrupt.

EXTERNAL AND NON-CLASS F FORMAT

The External and Non-Class F ICB format type (see Figure 3-1) is used with all RTOM interrupts and all CD and TD I/O interrupts. RTOM interrupts include: Console Interrupt (Panel Attention), Call Monitor Interrupt, and Real-Time Clock-Interrupt.

Words 1 through 4 contain the old and new PSDs.

Words 5 and 6 of this ICB format type are optional and may be omitted.

TRAP FORMAT

The Trap ICB format type (see Figure 3-2) is used for PSD mode traps.

Words 1 through 4 of the Trap ICB contain the old and new PSDs.

Word 5 of the Trap ICB contains the CPU hardware status word. This is stored in the ICB at the time a trap occurs. The CPU status word may provide additional descriptor bits for defining the error condition. For a detailed description of the CPU status word, refer to the 32/70 Series Technical Manual.

Word 6 of the Trap ICB is optional.

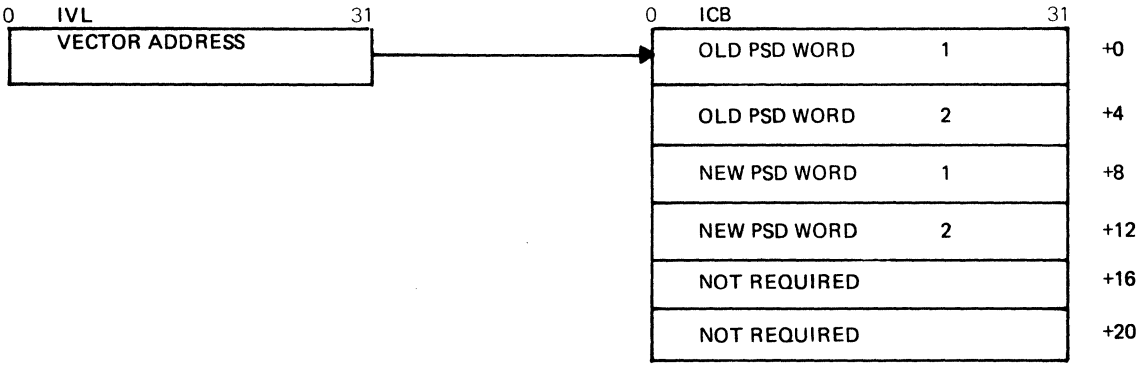


Figure 3-1. Interrupt Context Block Format - External Interrupts and Non-Class F I/O Interrupts

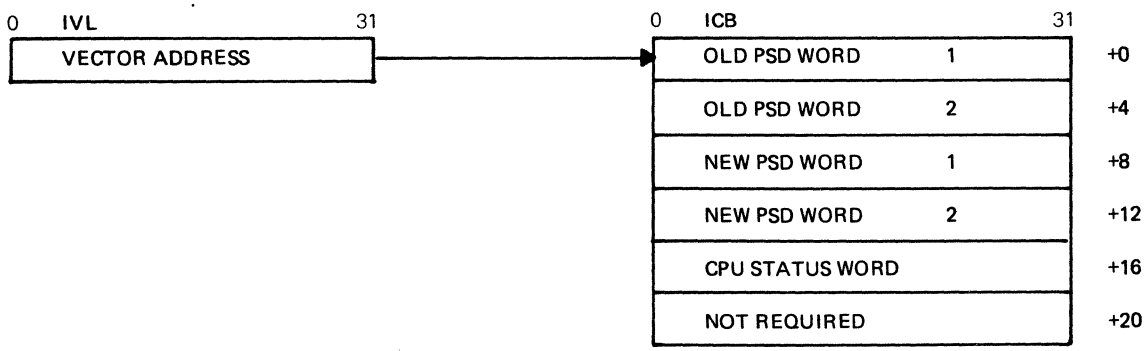


Figure 3-2. Trap Context Block Format

CLASS F I/O FORMAT

The Class F I/O format type (see Figure 3-3) requires the use of all six ICB words.

Words 1 through 4 contain the old and new PSDs.

Word 5 of the Class F I/O ICB provides the Input/Output Command List (IOCL) address for the associated Class F I/O channel. This word must be set up in the ICB by software prior to the execution of either a Start I/O or Write Channel WCS instruction. The ICL address is transmitted to the I/O channel by the CPU during the Start I/O or Write Channel WCS SelBUS sequences. The IOCL address must be in a 24-bit real address format.

Word 6 of the Class F I/O ICB contains the 24-bit real address of the channel status word. Whenever the channel reports status to the CPU (and software), the channel stores the channel status word in memory. The CPU then stores the memory address of the channel status word into word 6 of the ICB.

The channel may report status when any one of the following events occur:

1. An interrupt is acknowledged (a hardware event).

2. A Start I/O instruction is executed.

3. A Test I/O instruction is executed.

4. A Halt I/O instruction is executed.

When status is stored during a Start I/O, Test I/O, or Halt I/O instruction, the channel rejects the instruction, and the CPU Condition Codes are set to reflect the Status Stored condition. Under the Status Stored condition, the channel clears its status pending flags, as well as any interrupt pending flags that are relative to the status just reported.

SUPERVISOR CALL FORMAT

The Supervisor Call (SVC) instruction is provided with up to 16 different ICBs. These multiple ICBs are provided to reduce the amount of time required for a user program to request service from the operating system program. The address of a specific ICB is obtained by adding a 4-bit word index value from bits 16-19 of the SVC instruction to the 24-bit address that is in the SVC Interrupt Vector Location (IVL). The sum of these values provides a 24-bit real address of a Secondary Vector Location. The contents of the Secondary Vector Location is the 24-bit real address of the appropriate Supervisor Call ICB. Reference Figure 3-4.

Words 1 through 4 of the Supervisor Call ICB contain the Old and New PSD.

Word 5 of the ICB is available for use by the software SVC Trap processor as an index (call number) for the requested operating system service. Bits 20 through 31 of the SVC instruction are used by the CPU to format word 5 of the Supervisor Call ICB.

Word 6 of the Supervisor Call ICB is optional.

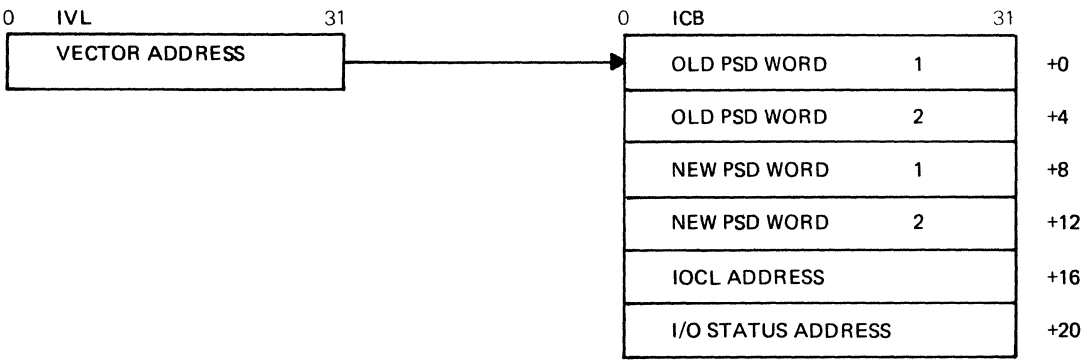


Figure 3-3. Interrupt Context Block Format - Class F I/O Interrupts

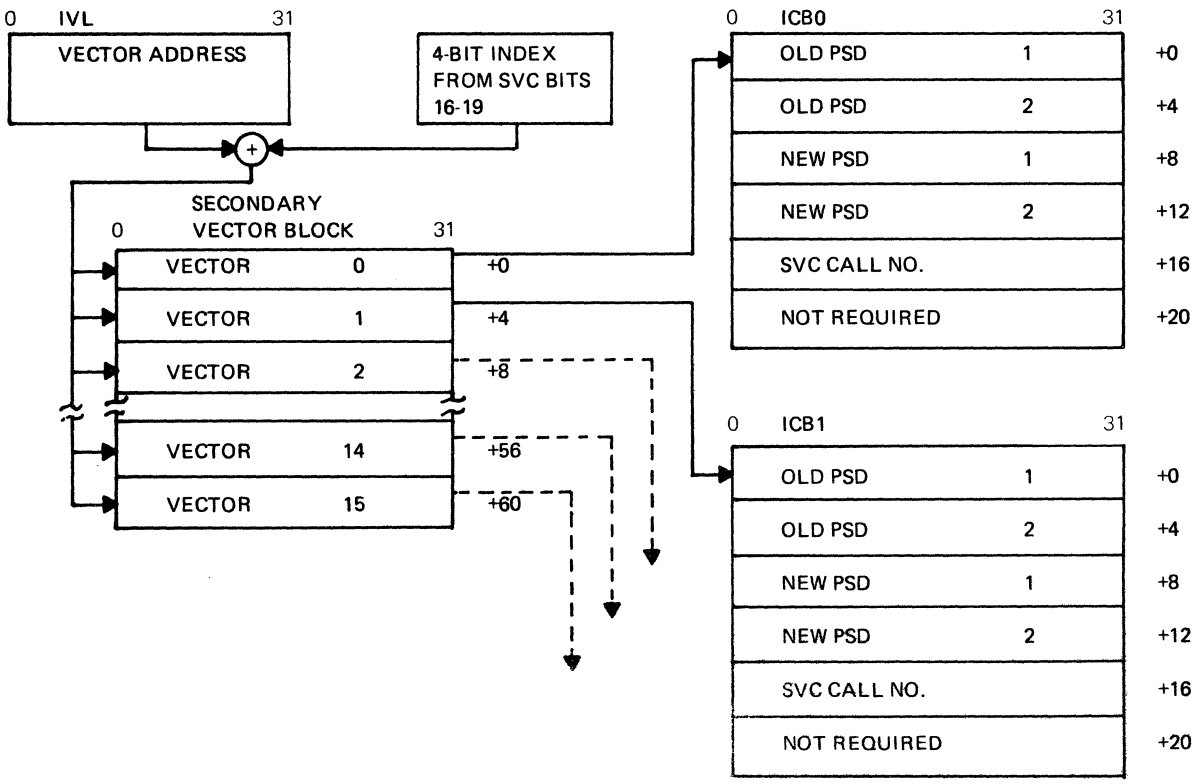


Figure 3-4. Supervisor Call (SVC) Trap Context Block Format

PSD MACRO INSTRUCTIONS

The eight PSD interrupt and trap related macro instructions are:

1. Block External Interrupts (BEI)
2. Unblock External Interrupts (UEI)
3. Load Program Status Doubleword (LPSD)
4. Load Program Status Doubleword Change Map (LPSDCM)
5. Set CPU Mode (SETCPU)
6. Supervisor Call (SVC)
7. Enable Arithmetic Exception Trap (EAE)
8. Disable Arithmetic Exception Trap (DAE)

All of the above macro instructions, except SVC, can be executed only in the privileged state and BEI, UEI, LPSD, EAE, DAE, and SVC will be valid instructions only if the CPU mode is set to other than the PSW mode. If the PSW mode is set, an undefined instruction trap will occur.

In the PSD mode, traps cannot be inhibited by the Blocked mode or by the activation of any high level interrupt.

A list of the traps, interrupts, and vector addresses is presented in Table 3-1.

AUTOMATIC TRAP HALTS

The 32/70 Series CPU provides for automatic trap halts in both the PSW and PSD modes of operation.

PSW TRAP HALTS

A PSW mode trap halt* can occur under any of the following conditions:

1. A Memory Parity Error or Nonpresent Memory Error, while handling the dedicated memory locations associated with an interrupt level. This error must occur during the firmware interrupt Store, Place, and Branch sequence or the Branch and Reset Interrupt (BRI) sequence.
2. An I/O communication protocol violation during the interrupt or BRI communication sequence.

*Implementation of the PSW trap halt is the same as described in the PSD trap halt discussion.

PSD TRAP HALTS

A PSD mode trap halt only occurs if the software has not enabled the PSD mode traps by the SETCPU Enable Trap instruction. The PSD mode traps that arm the Trap Halt logic are:

1. Memory Parity Error
2. Nonpresent Memory
3. Undefined Instruction
4. Privileged Violation Trap
5. Machine Check Trap
6. System Check Trap
7. MAP Fault Trap

The PSD mode traps that do not arm the Trap Halt logic are:

1. Supervisor Call Trap
2. Arithmetic Exception Trap
3. Call Monitor Interrupt Trap

MACHINE CHECK TRAP

A Machine Check trap is a hardware/firmware failure that has occurred during an interrupt or context switch. These failures include Memory Parity error, Nonpresent Memory error, or I/O and Interrupt SelBUS protocol violations. The specific type of error that causes the trap is described by the CPU Status Word that is stored in the interrupt (trap) context block.

SYSTEM CHECK TRAP

A System Check trap is primarily a software failure that attempted to force the CPU into an illogical sequence. The specific type of error that caused the trap is described by the CPU status word stored in the interrupt (trap) context block.

BLOCK MODE TIME-OUT TRAP

The Block Mode Time-Out (watchdog) trap occurs under the following conditions:

1. If a Wait instruction is executed with interrupts blocked.
2. If the Block Mode Time-Out trap has been enabled by a SETCPU instruction and more than 128 instructions have been executed with interrupts blocked.

PSD TRAP HALT IMPLEMENTATION

The detection of a PSD trap condition causes the following events to occur if traps are not enabled:

1. The CPU is halted.
2. The Interrupt Active light on the Serial Control Panel is turned on.
3. The PC portion of the PSW (PSD1) contains the dedicated memory address for the trap causing the halt.
4. The CPU halfword indicator (PSD1, bit 5) may or may not be on.
5. Starting at memory location 530_{16}, the following error information is stored:

Location	Contents
530	Error PSW (PSD1)
534	Error PSD2 (PSD mode only)
538	CPU Status Word
53C	R(RDEV) Device Table Entry
540	R(INTRTAB) Device Interrupt Entry

SECTION IV

MEMORY MANAGEMENT

INTRODUCTION

This section provides information that includes the rules for configuring MOS and core memory, as well as memory management programming methods and formats. For a functional description of the major elements in a 32/70 Series Memory Subsystem, the reader should refer to Section I of this manual.

OVERVIEW

All memory subsystems in the 32/70 Series are configured with a Memory Bus Controller (MBC) that communicates with the SelBUS and controls the memory bus to which the memory modules are attached. The MBC and CPU provide for byte, halfword, or word accesses of memory. The Memory Bus Controller is capable of managing up to 16 overlapped memory modules which operate asynchronously on their bus. Computer memory requests can be initiated every 150 nanoseconds due to the overlapped memory design. All modules under one Memory Bus Controller have the same cycle and access time; however, other MBCs may manage up to 16 fully overlapped modules.

MOS AND CORE MEMORY

Depending on the model, 32/70 Series systems can have either core or MOS memory. Core memory systems are organized into 36-bit words: 32 data bits plus 4 parity bits. MOS memory systems are organized with 39-bit words: 32 data bits plus 7 error checking correcting (ECC) bits. The MOS memory module corrects single-bit errors and has the capability of detecting and reporting double-bit errors.

Core memory packages include the following components:

1. Core memory modules
2. Memory chassis
3. Power supply
4. Memory Bus Controller

Core memory for 32/70 Series computers is available in the following forms:

1. The basic 32,768-byte core memory modules with a full memory cycle time of 600 nanoseconds
2. 65,536-byte core memory packages of 600-nanosecond memory
3. 131,072-byte core memory packages of 600-nanosecond memory
4. 65,536-byte core memory modules with a full memory cycle time of 900 nanoseconds
5. 131,072 core memory packages of 900-nanosecond memory

MOS memory packages include the following components:

1. 128 KB or 256 KB 900-nanosecond MOS memory modules(s)
2. Memory chassis
3. Power supply
4. Refresh board
5. Memory Bus Controller (MBC)

600/900 NANOSECOND CORE MEMORY MODULES

The 32/70 Series computers will support both 600- and 900-nanosecond core memory modules if they are not intermixed with one memory interface. Since the individual memory modules connected to the memory interface have a full cycle time of 600 or 900 nanoseconds, and the SelBUS operates synchronously with full 32-bit word transfers occurring every 150 nanoseconds, the memory chassis handles the following combinations of overlapped memory operations:

1. a. Four memory write operations (26.67M bytes/second) (for 600 ns memory)
   b. Six memory write operations (26.67M bytes/second) (for 900 ns memory)
2. a. One memory read and two memory write operations (19.99M bytes/second) (for 600 ns memory)
   b. One memory read and two memory write operations (22.22M bytes/second) (for 900 ns memory)
3. a. Two memory read operations (13.33M bytes/second) (for 600 ns memory)
   b. Three memory read operations (10.00M bytes/second) (for 900 ns memory)

MIXED MEMORY RULES

MOS and core memory may be mixed on 32/70 Series systems. However, it should be done only in accordance with the rules listed below:

1. Mixed memory can be accomplished on 32/70 Series systems only.
2. The higher speed memory must be the low order address space.
3. Separate MBCs, chassis, and power supplies must be used for the different memory types.
4. The core memory should occupy the low order address space.
5. The total amount of core memory in the low order address range must be equal to or a multiple of the MOS memory module size.

An amplification of the preceding rules is provided in the paragraphs that follow.

Mixing MOS and core memory should not be attempted on systems other than the 32/70 Series. For example, the 32/35 and 32/55 cannot support MOS memory. The 32/30 and 32/57 cannot have mixed memory because they use a split backplane.

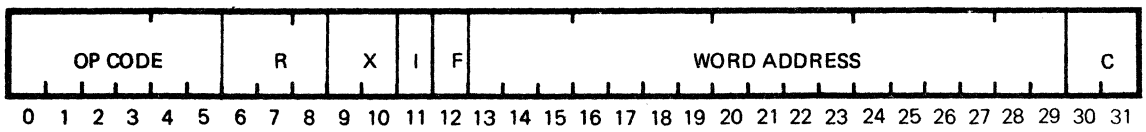
Separate MBCs, chassis, and power supplies are necessary because MOS and core memory units have different requirements in this regard. When adding core memory to a Model 32/77 processor, it is necessary to add Model 2332 Memory Carriage for 900 ns core memory. The Memory Carriage includes the chassis, power supply, and MBC required to support the core memory. This MBC will not support MOS memory. To add MOS memory to a Model 32/75 processor, a Model 2375 or 2380 Memory Package is required and provides the chassis, power supply, MBC, and memory.

Core memory should occupy the low order address space. This is to ensure that register save areas are in nonvolatile memory locations. If a customer is unconcerned about the state of the processor at the time of a power failure, then the core memory could be high address locations.

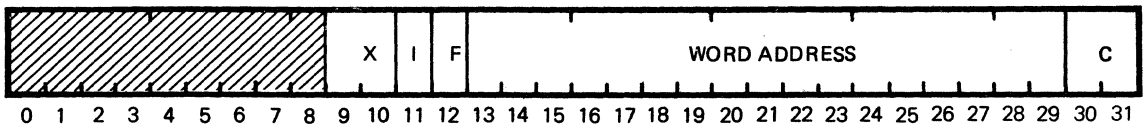
Assuming the core memory is in the low order address space, it is necessary to protect the memory from unwanted discontiguous memory locations (holes). The amount of memory on the first MBC will be dictated by the incremental granularity of the MOS memory modules on successive MBCs. Since the smallest granularity of the MOS memory boards is 32 KW, there would have to be at least 32 KW of core on the first MBC. If the MOS memory module used contained 64 KW, the amount of core on the first MBC would have to be 64 KW. After the first MOS memory board size is established, any additional boards must be of the same size. An example would be a Model 32/75 CPU with four 8 KW, 600 ns core memory modules (Model 2152). If a customer wished to add the 64 KW MOS Memory Package (Model 2380) to the CPU, a prerequisite would be to add four additional 8 KW, 600 ns core memory modules (2152) to the first MBC. This establishes the memory on the first MBC (64 KW) and is equal to the granularity of the MOS Memory Package of 64 KW. Additional 64 KW memory modules (Model 2381) can then be added to the MOS Memory Package.

MEMORY REFERENCE INSTRUCTIONS

Bits 9-31 have the same format in every memory reference instruction whether the effective address is used for storage or retrieval of an operand, as an indirect address operand, or to alter program flow. The Memory Reference instruction format is shown below:



Bits 9 and 10 specify the general purpose register (GPR) to be used as an index register, bit 11 is the indirect bit, and bits 12-31 define the word address and data type. The effective address of the instruction depends on the values of I, X, and bits 12-31. If I and X are both Zero bits 12-31 address the data type defined by bits 13-29.



F- AND C-BITS

The format of the F- and C-bits have been selected so that any selected data type (byte, 16-bit halfword, 32-bit word, or 64-bit doubleword) can be conviently indexed by that data type. The possible combinations of F- and C-bits are as follows:

F	C	Data Type
0	00	32-bit word
0	01	16-bit left halfword (bits 0-15)
0	10	64-bit doubleword
0	11	16-bit right halfword (bits 16-32)
1	00	Byte 0 (bits 0-7)
1	01	Byte 1 (bits 8-15)
1	10	Byte 2 (bits 16-23)
1	11	Byte 3 (bits 24-31)

DIRECT ADDRESSING

When an X is equal to Zero (no indexing), and I is equal to Zero (no indirect), the effective memory address is taken directly from bits 13-29 of the Memory Reference instruction.

The Store Word instruction is coded:

```
STW          0,0
```

and is assembled as hexadecimal D4000000. When executed, this instruction stores the contents of General Purpose Register 0 directly into memory byte location 0.

The Store Byte instruction is coded:

```
STB          0,1
```

and is assembled as hexadecimal D4080001. Note that the F- and C-fields of the instruction have been altered. When executed, this instruction stores the least significant byte of General Purpose Register 0 directly into memory byte location 1.

INDIRECT AND INDEXED ADDRESSING

Indirect addressing can be combined with indexing at any indirect level. An example of indirect addressing with indexing follows:

Location Counter	Machine Instruction	Byte Address	Label	Operation	Operand
				PROGRAM	
P00000				REL	
P00000	C9800004		STRT	LI	3,4
P00004	AC90000C	P0000C		LW	1,*LOC1
P00008	3055			CALM	X'55'
P0000A	0002				
P0000C	00100010	P00010	LOC1	ACW	*LOC2
P00010	00700014	P00014	LOC2	ACW	*LOC3,3
P00014	00000000		LOC3	DATAW	0
P00018	0000001C	P0001C		ACW	LOC4
P0001C	0000FFFF		LOC4	DATAW	X'0000FFFF'
P00020		P00000		END	STRT

The first executable instruction is a Load Immediate (LI) to load a value of 4 into GPR3 (index register). The next instruction to be executed is the Load Word (LW). This instruction directs the machine to load GPR1, indirectly using the contents of LOC1 as the operand address. The address in LOC1, however, has the indirect bit on; the machine uses this address to fetch the contents of LOC2. The contents of LOC2 has an indirect bit on, but it also points to GPR3 for indexing. The machine then takes the address contents of LOC2 and adds to it the contents of GPR3 (which increases the address by four bytes). The resulting address points to LOC4. The address stored in LOC4 has the indirect bit off. The machine then uses the address P0001C stored in LOC4 as the final operand logical address and loads GPR1 with the hexadecimal value 0000FFFF. The ACW statement is a Macro Assembler directive used to generate an address constant. The DATAW is also a Macro Assembler directive.

INDEXED ADDRESSING

Any data type may be indexed by adding a bit at the bit position corresponding to the displacement value for each data type. These are as follows:

Data Type	Bit Position
Byte	31
Halfword	30
Word	29
Doubleword	28

If X is nonzero (specifying indexing), bits 13-31 are used to produce a memory address by adding it to the contents of the general purpose register specified by X. Only General Purpose Registers 1, 2, and 3 function as index registers.

For selective or indexed addressing, the displacement is a two complement integer within one of the general purpose registers used for indexing. For word indexing, bit 29 of the index register is the least significant bit of the address. If bit 29 of GPR3 is set to One to provide a displacement of one word, the indexed Store Word instruction is coded:

```
STW          0,0,3
```

This now stores the contents of GPR0 in memory indexed by the contents of GPR3. The instruction would assemble as D4600000. The calculated logical effective word operand address (after indexing) would be 00004. Therefore, the contents of GPR0 will be stored in memory location 00004.

INDIRECT ADDRESSING

If I is equal to Zero, addressing is direct, and the address already determined from X and bits 12-31 is the effective address used in the execution of the instruction.

If I is equal to One, addressing is indirect, and the processor retrieves another address specified by the operand address. In this new address, bits 9 and 10 select the index register and bit 11 is the indirect bit; bits 12-31 specify the effective address as in the memory reference instructions. To use the indirect addressing capability the instruction would be coded:

```
STW          0,*0
```

which causes bit 11, the indirect bit, to be set to One. When executed, this instruction stores the contents of GPR0 in the memory location whose address is stored in memory location 0.

Multilevel indirect addressing can be performed when each new address taken from memory has the indirect bit (bit 11) set to One. The process of fetching indirect addresses continues until an address has bit 11 equal to Zero. This address is the logical effective operand address.

WORDS, HALFWORDS, AND BYTES

Each fullword instruction (32 bits) must be stored in memory on a word boundary (bits 30 and 31 equal to Zero). Memory information boundaries are illustrated in Figure 4-1.

Halfword instructions are stored two per word. When a halfword is followed by a word instruction, the Assembler positions the instruction in the left half of the word and stores a No Operation (NOP) instruction in the right half of the word. This maintains the word boundary discipline.

Memory Reference instructions which address a byte in memory do not alter the other three bytes in the memory word containing the specified byte. Memory instructions which address a halfword do not alter the other halfword of the memory location. The exeception to the preceding is that the Add Bit in Memory instruction may propagate a carry to the most significant bit of the word containing the specified bit.

WORD AND DOUBLEWORD OPERANDS

Word operands must be stored in memory on a word boundary. The most significant word of a doubleword operand must be stored in a memory location having an even word address with the least significant word stored in the next sequentially higher (i.e., odd word) location. Some examples of memory addressing follow:

Byte	Halfword	Word	Doubleword
00000	00000	00000	00000
00001			
00002	00002		
00003			
00004	00004	00004	
00005			
00006	00006		
00007			
00008	00008	00008	00008
00009			
0000A	0000A		
0000B			
0000C	0000C	0000C	
0000D			
0000E	0000E		
0000F			
00010	00010	00010	00010

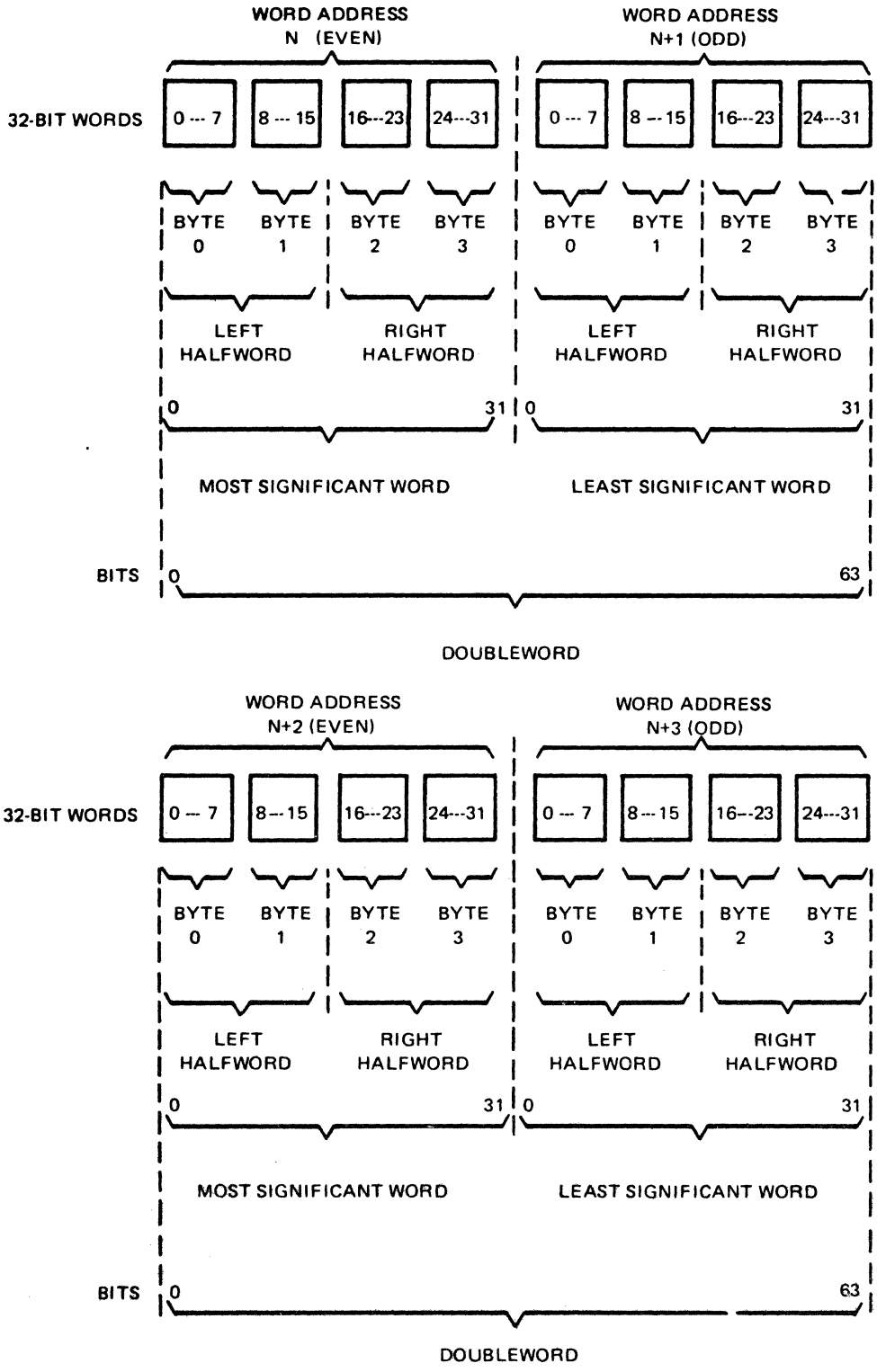


Figure 4-1. Information Boundaries in Memory

HARDWARE MEMORY MANAGEMENT

The 32/70 Series computer features Hardware Management that provides full utilization of all available memory. The memory management hardware includes: hardware Memory Allocation and Protection (MAP), extensions to the interrupt, I/O, and memory subsystems. This feature also allows programs to be loaded in one area of physical memory, rolled out to disc, rolled back into another area of memory, and to continue execution without requiring time-consuming software relocation biasing.

In addition, these programs may be distributed throughout physical memory in 32K-byte blocks to take complete advantage of available memory. Hardware Memory Management, including automatic context switching, is accomplished through the processing and control of the MAP. The MAP consists of up to thirty-two 16-bit halfwords. The first 16 halfwords (the Primary MAP) are used to define a 512K-byte logical primary address space into which may be loaded either data or executable programs. The second 16 halfwords (the Extended Operand MAP) are used to define a 512K-byte logical extended operating address space into which only data may be loaded.

By using the MAP, a 512K-byte logical primary address space may be distributed in 32K-byte blocks throughout the 16,777,216 bytes of physical memory and may contain data or instructions. The 32/70 Series computer can access and execute programs up to 512K bytes in size, located anywhere within physical memory (16M bytes). The user can also use an additional 512-K byte logical extended operand address space for data storage. The combination of the logical primary address space and the additional extended operand address space provides support throughout physical memory, provided that the executable code lies entirely within the logical primary address space.

ADDRESSING MODES

The 32/70 Series computer provides the capability of accessing memory in any of the following modes:

1. 512 KB mode
2. 512 KB Extended mode
3. 512 KB Mapped mode
4. Mapped, Extended mode

512 KB MODE

The 512 KB mode of memory address allows the 32/70 Series Central Processor Unit to directly access any byte, halfword, word, or doubleword in the first 512K bytes of memory without mapping, indexing, or address modification. A 19-bit address field is provided in all Memory Reference instructions for this purpose.

Bits are addressed by using the R (register) field of the instruction word to designate a bit in the byte specified by the 19-bit address. Therefore, any bit in 512K bytes of memory can be directly addressed by the Bit Manipulation instructions.

512 KB EXTENDED MODE

The 512 KB Extended mode of memory addressing provides the same capabilities as the 512 KB mode plus operand addressing beyond the first 512K bytes of memory to reference all bits, bytes, halfwords, words, and doublewords residing anywhere within 16 megabytes of physical memory, This mode of addressing combines the contents of an index register with the 19 bits of locical address in the Memory Reference instruction to produce a 24-bit physical memory address anywhere in the 16 megabytes of memory. All memory above the first 512K bytes is usable only for data storage and retrieval and not for executable instructions. This mode of memory addressing is applicable to both the PSW and the PSD modes of operation.

512 KB MAPPED MODE

The 512 KB Mapped mode of memory addressing allows a 32/70 Central Processor Unit to access any byte, halfword, word, or doubleword within 16 megabytes of memory through memory mapping. In this mode, the memory management hardware supports up to 16 logical address pages (a page is 32K bytes) distributed throughout 16 megabytes of physical memory by providing mapping and automatic context, MAP, and protection switching. All 16 pages of logical address pages may be used for executable code instructions or for data storage and retrieval. Physical blocks of memory may be common to multiple address spaces, providing a way for users in different address spaces to share common blocks of memory.

MAPPED/ EXTENDED MODE

The Mapped/Extended mode of memory addressing allows a 32/70 Series Central Processor Unit to access any byte, halfword, word, or doubleword within 16 megabytes of memory through memory mapping. In this mode, the memory management hardware supports up to 32 logical address pages (a page is 32K bytes) distributed throughout 16 megabytes of physical memory by providing mapping and automatic context, MAP, and protection switching. The first 16 pages of logical address pages may be used for executable code or data, and the last 16 pages of logical address pages must be used for data storage and retrieval only. Multiple-user programs may be loaded into any or all of the first 16 pages of logical address pages. A 32/70 Series Computer allows each of these users to directly address any bit, byte, halfword, word or doubleword within the address space in which it resides. Physical blocks of memory may be common to multiple address spaces, providing a way for users in different address spaces to share common blocks of memory.

MEMORY MAPPING

The 32/70 Series computer includes thirty-two 16-bit (halfword) locations, the Primary MAP, and the Extended Operand MAP. The Primary MAP and the Extended Operand MAP are used to map the 512K-byte logical primary address space and the 512K-byte logical extended operand address space, respectively, onto physical memory addresses. Each of the 16-bit MAP locations associates 32K bytes of the logical primary address space or logical extended operand address space with 32K bytes (8K words) of physical memory. Logical address spaces are defined by building MAP Image Descriptor Lists (MIDL) as shown in Figure 4-2.

Each MIDL contains up to 32 halfword page entries (a page is 32K bytes or 8K words), which contains a 12-bit Page Entry, a Page Valid or Nonvalid bit, and a Write Protect/Unprotect bit. Any or all of the 32 pages may be designated as Write Protected. The first 16 page entries (logical primary address space) may be used for executable instructions or for data storage and retrieval. The second 16 page entries (Extended Operand MAP Image) may only be used for data storage and retrieval purposes. For a complete description of the Memory Mapping, refer to the Memory Addressing section of the Instruction Repertoire.

A logical representation of the components involved in the memory management process of a 32/70 Series system are depicted in Figure 4-3.

BIT / WORD$_{16}$	0 VALID	1 PROTECT	2	EVEN HALFWORDS 3 4 5 6 7 8 9 10 11 12 13 14 15	16 VALID	17 PROTECT	18	ODD HALFWORDS 19 20 21 22 23 24 25 26 27 28 29 30 31
0				PRIMARY MAP PAGE 0				PRIMARY MAP PAGE 1
1				PRIMARY MAP PAGE 2				PRIMARY MAP PAGE 3
2				PRIMARY MAP PAGE 4				PRIMARY MAP PAGE 5
3				PRIMARY MAP PAGE 6				PRIMARY MAP PAGE 7
4				PRIMARY MAP PAGE 8				PRIMARY MAP PAGE 9
5				PRIMARY MAP PAGE 10				PRIMARY MAP PAGE 11
6				PRIMARY MAP PAGE 12				PRIMARY MAP PAGE 13
7				PRIMARY MAP PAGE 14				PRIMARY MAP PAGE 15
8				EXTENDED OPERAND MAP PAGE 0				EXTENDED OPERAND MAP PAGE 1
9				EXTENDED OPERAND MAP PAGE 2				EXTENDED OPERAND MAP PAGE 3
A				EXTENDED OPERAND MAP PAGE 4				EXTENDED OPERAND MAP PAGE 5
B				EXTENDED OPERAND MAP PAGE 6				EXTENDED OPERAND MAP PAGE 7
C				EXTENDED OPERAND MAP PAGE 8				EXTENDED OPERAND MAP PAGE 9
D				EXTENDED OPERAND MAP PAGE 10				EXTENDED OPERAND MAP PAGE 11
E				EXTENDED OPERAND MAP PAGE 12				EXTENDED OPERAND MAP PAGE 13
F				EXTENDED OPERAND MAP PAGE 14				EXTENDED OPERAND MAP PAGE 15

Figure 4-2. MAP Image Descriptor List

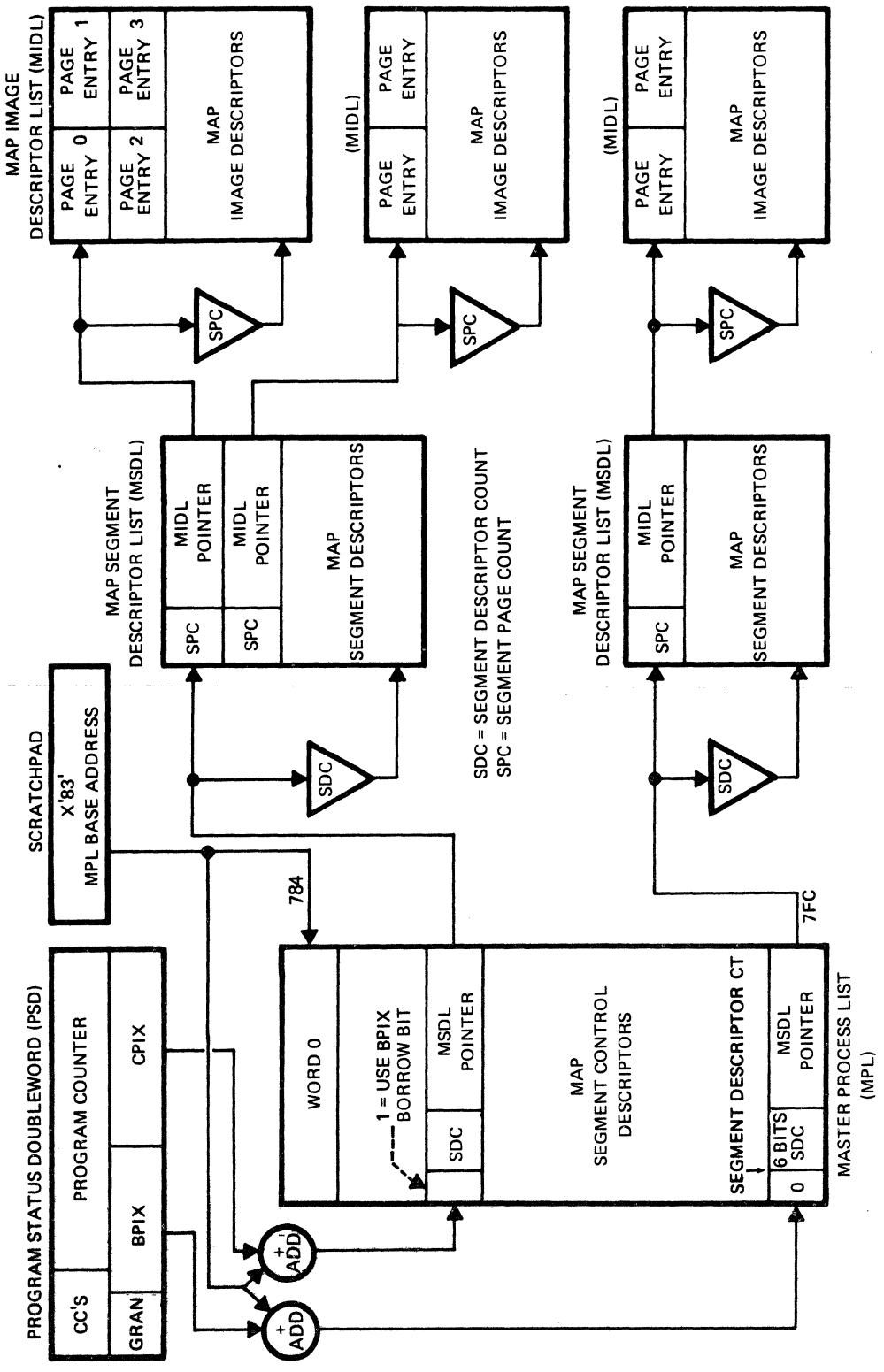


Figure 4-3. Memory Management Components

MEMORY PROTECTION

The memory protection system provides write protection for individual memory pages. When the CPU is in the Mapped mode (either 512 KB or Extended), each 32 KB memory block of logical program address space may be write protected. Write protection for a 32 KB memory block is selected by setting the protect/unprotect bit that is stored, along with the block address, in the MAP register of the CPU.

When the CPU is in the Unmapped mode (either 512 KB or Extended), 512-word memory pages may be write protected. Up to 256 pages (128K words) can be protected at a time. The sixteen 16-bit Page Protect registers are provided in the Unmapped mode.

Write Protection may be overridden by a CPU operating in the Privileged mode.

PROGRAM STATUS DOUBLEWORD

The Program Status Doubleword (PSD) provides information relating to the operation that was interrupted or trapped (Old PSD), and the mode and instruction address that is to be given control during context switching (New PSD). The format of the PSD is shown in Figure 4-4.

Execution of any Branch or Branch-and-Link instruction replaces the contents of bits 13-30 of the PSD with the effective address specified by the instruction. In addition, if the Branch instruction specifies an Indirect Branch operation, the contents of bits 1-4 of the PSD are replaced by the contents of the corresponding bit positions in the indirect addresss location.

PSD FIELDS

The PSD fields are coded as follows:

1. PRIV (bit 0) indicates the Privileged mode.

   0 = Nonprivileged
   1 = Privileged

2. CCs (bits 1-4) indicate the condition codes.

   Bit 1 = CC1

   Bit 2 = CC2

   Bit 3 = CC3

   Bit 4 = CC4

3. EXT (bit 5) indicates Indexing mode.

   0 = Off
   1 = On

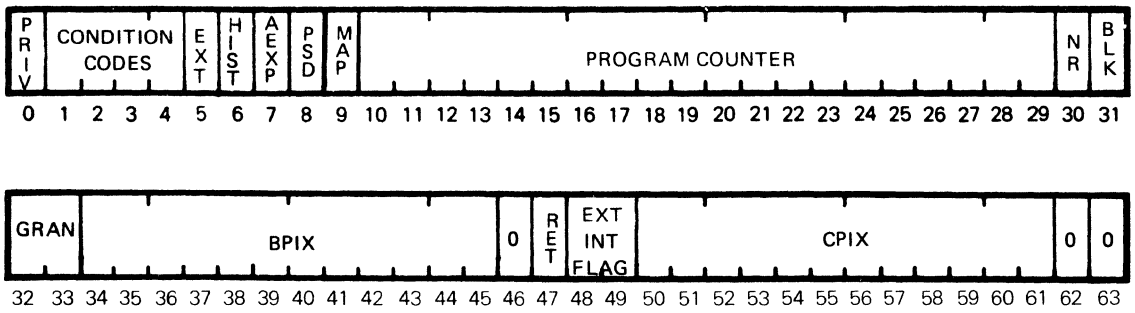
4. HIST (Bit 6) indicates last instruction was a right halfword (Old PSD only).

5. AEXP (Bit 7) indicates Arithmetic Exception Trap Mask.

   0 = OFF (Do not generate Arithmetic Exception Trap)
   1 = ON (Generates Arithmetic Exception Trap)

6. PSD (Bit 8) indicates PSD mode.

   0 = PSD mode off (Displayed PSD only)
   1 = PSD mode on (Displayed PSD only)



BIT 0 = 0 UNPRIVILEGED MODE
= 1 PRIVILEGED MODE

BITS 1-4 ARE CONDITION CODES
BIT 1 = CC1
2 = CC2
3 = CC3
4 = CC4

BIT 5 = 0 EXTENDED MODE (OFF)
= 1 EXTENDED MODE (ON)

BIT 6 = 0 LAST INSTRUCTION EXECUTED WAS NOT A RIGHT HALFWORD
= 1 LAST INSTRUCTION EXECUTED WAS A RIGHT HALFWORD

BIT 7 = 0 ARITHMETIC EXCEPTION TRAP MASK (OFF)
= 1 ARITHMETIC EXCEPTION TRAP MASK (ON)

BIT 8 = 0 COMPUTER IS IN PSW MODE (DISPLAYED PSD ONLY) *
= 1 COMPUTER IS IN PSD MODE (DISPLAYED PSD ONLY) *

BIT 9 = 0 UNMAPPED (DISPLAYED PSD ONLY) *
= 1 MAPPED (DISPLAYED PSD ONLY) *

BITS 10-12 ARE NOT USED
BITS 13-29 ARE LOGICAL WORD ADDRESS
BIT 30 NEXT INSTRUCTION IS A RIGHT HALFWORD
BIT 31 BLOCKED (DISPLAYED PSD ONLY) *
BITS 32-33 INDICATE MAP GRANULARITY, 00=UNMAPPED AND ALL OTHERS =8K MAP GRANULARITY
BITS 34-45 PROVIDE A WORD INDEX INTO THE MASTER PROCESS LIST (MPL) FOR THE BASE PROCESS
BIT 46 NOT USED
BIT 47 RETAIN CURRENT MAP CONTENTS
BITS 48-49 INTERRUPT CONTROL FLAGS

BITS		
48	49	
0	0	OPERATE WITH UNBLOCKED INTERRUPTS
0	1	OPERATE WITH BLOCKED INTERRUPTS
1	0	RETAIN CURRENT BLOCKING MODE
1	1	RETAIN CURRENT BLOCKING MODE

BITS 50-61 PROVIDE WORD INDEX INTO MASTER PROCESS LIST (MPL) FOR CURRENT PROCESS
BITS 62-63 NOT USED

* THESE BITS ARE USED FOR DISPLAY ONLY AND ARE NOT PRESENT IN THE PSD STORED IN MEMORY.

Figure 4-4. Formats for PSD1 and PSD2

7. MAP (Bit 9) indicates Mapped mode

   0 = Unmapped mode (Displayed PSD only)
   1 = Mapped mode (Display PSD only)

8. PROGRAM COUNTER (Bits 10-29) indicate the logical program counter (Word Address).

   Bits 10-12 are reserved for possible later use. (They must be zero)
   Bits 13-29 are the logical address.

9. NR (Bit 30) indicates next instruction is a right halfword.

10. Blocked (Bit 31) indicates Blocked mode (Displayed PSD only).

11. MAP MODE (Bits 32-33) indicate the Granularity as:

    00 = Unmapped
    01 = Mapped 8K Granularity
    10 = Mapped 8K Granularity
    11 = Mapped 8K Granularity

12. BPIX (Bits 34-46) provide a word index into the Master Process List (MPL) for the base process. (Bit 46 is ignored.)

13. Bit 47 = Retain current MAP contents. (New PSD only)

14. EXT INT FLAG (Bits 48 and 49) indicate external interrupt state.

Bits		
48	49	
0	0	= Operate with Unblocked interrupts (interrupt level active)
0	1	= Operate with Blocked interrupts (interrupt level not active)
1	0	= Retain Current Blocking Mode (New PSD only)
1	1	= Retain Current Blocking Mode (New PSD only)

15. CPIX (Bits 50-63) provide a word index into the Master Process List (MPL) for the current process. Bits 62 and 63 are ignored.

CONDITION CODES

A 4-bit Condition Code is stored in the PSD on completion of the execution of most instructions. These conditions may be tested to determine the status of the results obtained.

CC1 is set if an Arithmetic Exception occurs
CC2 is set if the result is greater than zero
CC3 is set if the result is less than zero
CC4 is set if the result is equal to zero

The Branch Condition True (BCT), Branch Condition False (BCF), and the Branch Function True (BFT) instructions allow testing and branching on the Condition Codes.

MAP DESCRIPTION

The second word of the PSD contains two 12-bit fields whose primary purpose is to provide the linkage from that PSD to the correct map entries for execution of the process associated with that PSD. The CPU MAP consists of a RAM with 32 locations, and the firmware will locate the appropriate entries for this RAM in main memory through a set of software-maintained tables which are interpreted by firmware on these two values from the PSD.

The 12-bit fields are named as follows:

1. BPIX - Base Process Index

2. CPIX - Current Process Index

The software maintains a Master Process List in memory. The base address is kept in a known (scratchpad) location. It contains one entry for every value which can appear in either the BPIX or CPIX fields, and it is quite reasonable for PSDs to exist where the CPIX and BPIX are identically equal. This Master Process List is maintained by the most privileged code of the system, and destruction of its contents will surely lead to immediate disaster.

MASTER PROCESS LIST (MPL)

The address of the MPL is set by the CPU firmware at System Reset time by the loading of a predetermined scratchpad cell with the 24-bit physical MPL address. The MPL entries contain the physical address of the MAP Segment Descriptor List (MSDL) and a 6-bit count of the number of Map Segments which concantenates to form the appropriate map contents.

When a PSD is being entered into the CPU, the firmware is faced with one of three possible actions relating to the map:

1. The PSD being loaded has its mode set to Unmapped, which basically means that it is going to operate with physical rather than logical memory addresses. Firmware action when loading this type of PSD is simply to leave the map contents as they are, and cause them to become inactive for the duration of this PSD execution.

   The Unmapped indication in the PSD overrides the Load Program Status Doubleword And Change Map (LPSDCM) instruction.

2. The PSD is being loaded as a result of the software instruction LPSD. In this event, firmware is being assured by the software that the map contains the appropriate contents and the only firmware action necessary is to reactivate the map circuitry. The basic function of this is to avoid the cost of reloading the map when returning from an excursion into an unmapped function, and software will insure that no other mapped process has intervened.

3. With the exception of the two preceding cases, the entry of a new PSD into the CPU always results in a total initialization of the map cirucit.

   The MAP RAM will be loaded from page 0 up with values obtained from main memory.

The PSD being loaded contains sufficient information for the firmware to make its way through the series of software-maintained tables in main memory to assemble the information necessary to initialize the map circuit. The objective of the table design is to provide for the assemblage of an addressability for that PSD from three distinct types of elements:

1. Private data which is unique to that process.

2. Statically shared data which is shared between several processes. This sharing is known at load (map creation) time. Since there exists in reality only a single copy of the data, it is important to software that a single physical copy of its logical/physical map exists, and that all PSDs using this shared data are funnelled through that copy for both software sanity and usage statistics.

3. Data that is shared by means of dynamic invocation. This data (like a Task Service Area (TSA)) is logically "owned" by a particular process, but needed by a variety of other processes which are invoked by the original process in the course of its execution. This data is generally of the type that it is a "per process global" set of data where any number of Operating System (OS) services need a random subset of the information which defies the organization as a reasonable parameter package, and is likely unalterable directly by the "owning" process. The OS services which need this data essentially have a partial map in memory covering their private code and data, which must be completed by adding this invocation page for them to correctly perform their functions.

It would be possible to accomplish this dynamic completion of the OS service map by moving into the service map image in memory, but the complexity of maintaining a stack of these invocations and returns (which are totally unsequenced due to the dispatching strategy) is large, and a dynamic link through the PSD relieves both complexity and overhead in this area.

The key elements of the PSD which provide firmware with the ability to satisfy these requirements are two 12-bit fields in the second word of the PSD, the CPIX (Current Process Index), and the BPIX (Base Process Index).

These two fields are both direct word indices into a software-maintained Master Process List (MPL) which is located in physical memory. It is both reasonable and frequent that the BPIX and CPIX fields of a PSD contain the identical number. The MPL is maintained by the most privileged OS code and any destruction will result in immediate disaster.

When the firmware must initialize the map circuit during the loading of a PSD, the following procedure is followed:

1. Using CPIX, locate the MAP Segment Control Descriptor (MSCD) in the MPL. This word is the controlling factor in map initialization. This word consists of three fields (see Figure 4-5):

    a. Borrowed Bit (Bit 0) - Tells the firmware (1) that the first set of map entries are to be obtained from the BPIX MSCD to satisfy the invocation sharing time of creation of this entry, and (2) the numeric value of the BPIX was unknown (and there exists a multiplicity of BPIXs).

    b. Segment Descriptor Count (SDC) - The count of the number of Segment Descriptors which are required to describe the addressability of the PSD.

    c. MAP Segment Descriptor List (MSDL) Pointer - The physical address in main memory of the first (or second if the borrowed bit was set) CPIX Segment Descriptor.

A MAP Segment Descriptor (MSD) is a single word entry which has two fields (see Figure 4-6):

1. Segment Page Count (SPC) - A count of the number of pages (map locations) which this Segment Descriptor covers.

2. Map Image Descriptor List (MIDL) Pointer - The starting physical address of the map cell block which contains the MAP Image Descriptors (MID). A MAP Image Descriptor is a single word with one or two halfword page entries (see Figure 4-7).

If the borrowed bit is set when the firmware locates the MSCD, the first segment descriptor is taken from the segment list which is described by the BPIX, and the second and subsequent segment descriptors are taken from the list described by this MSCD. When this indirection has been completed, the only noticeable impact on further processing is that the first map cell to be loaded from this list is "n" rather than "0" (if the borrow bit had not been set).

The variable length of pages described by each segment descriptor word are concantenated into the map until the segment count from the MPL is exhausted. The initialization is complete.

ADDRESS GENERATION

Address generation is accomplished by adding the contents of the instruction to the contents of the index register to form a logical address. In the Unmapped modes, the logical address is the same as the physical address. In Mapped modes, a portion of the logical address is used to address the MAP, while the remaining portion is used in the physical address. A graphical representation of the address generation process for each of the four modes is presented in Figures 4-8 to 4-11.

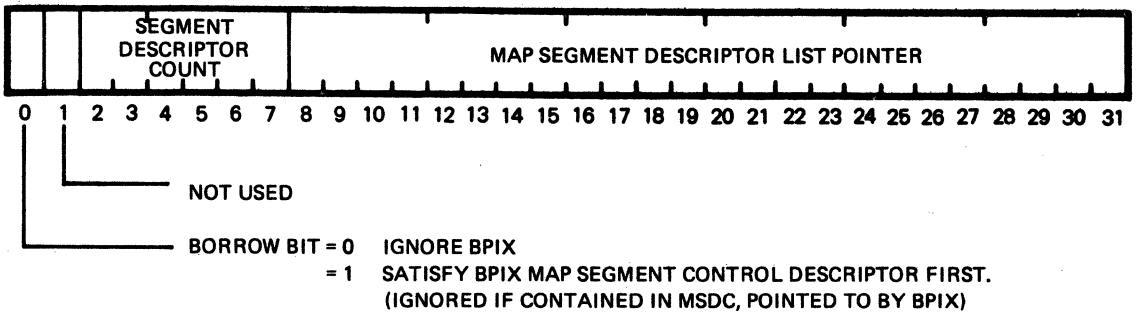


Figure 4-5. MAP Segment Control Descriptor (MSCD)

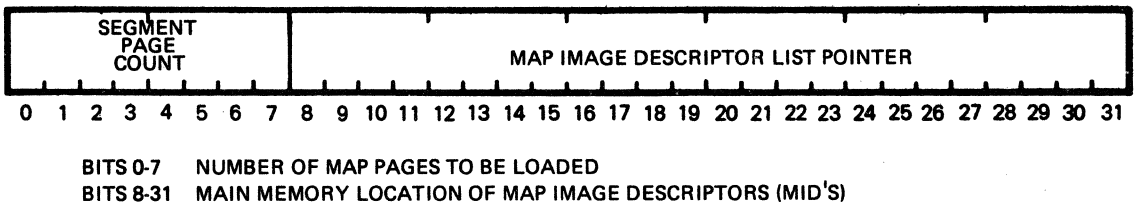


Figure 4-6. MAP Segment Descriptor (MSD)

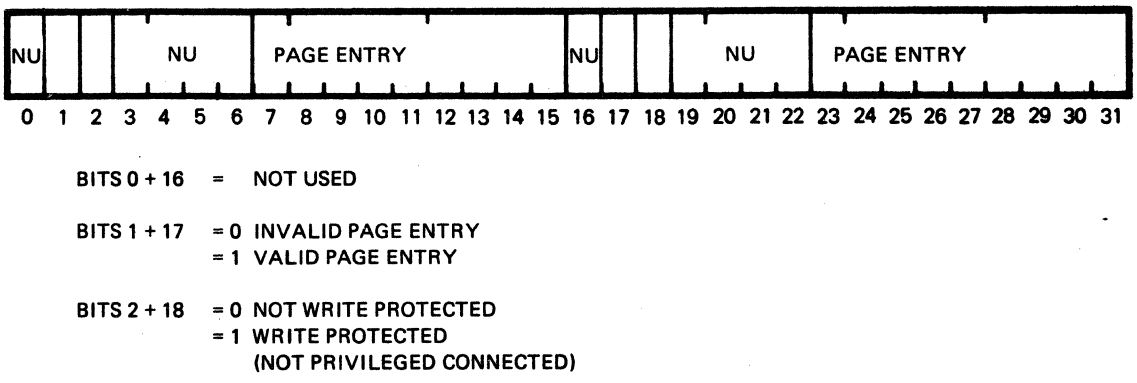
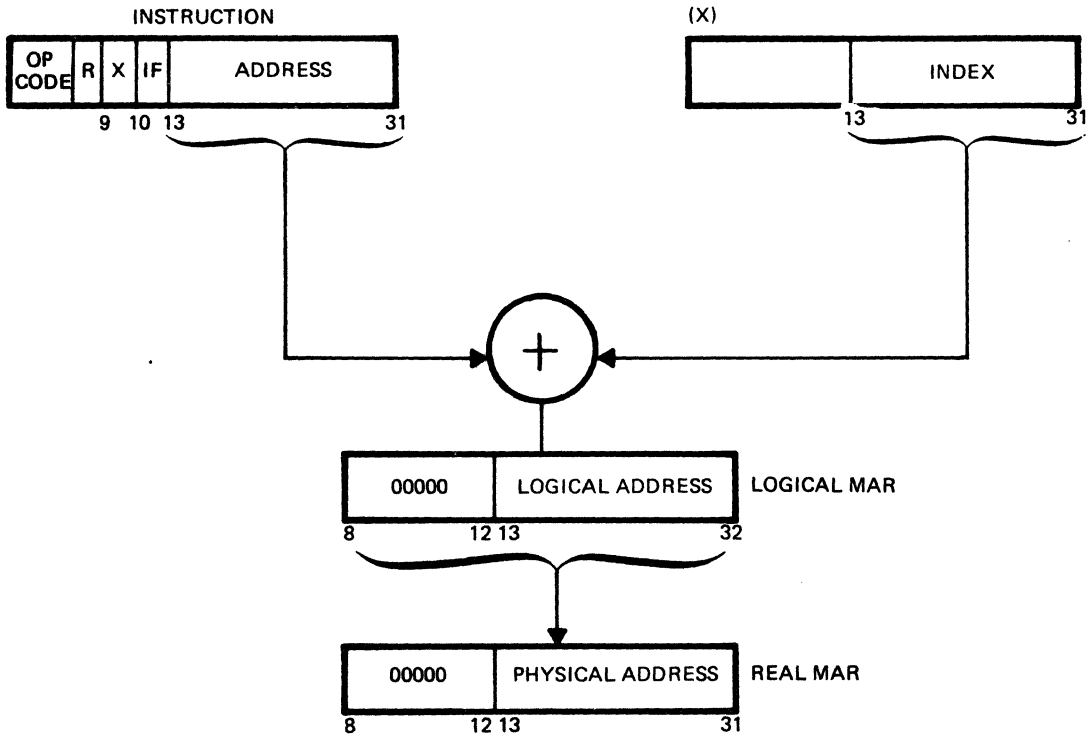
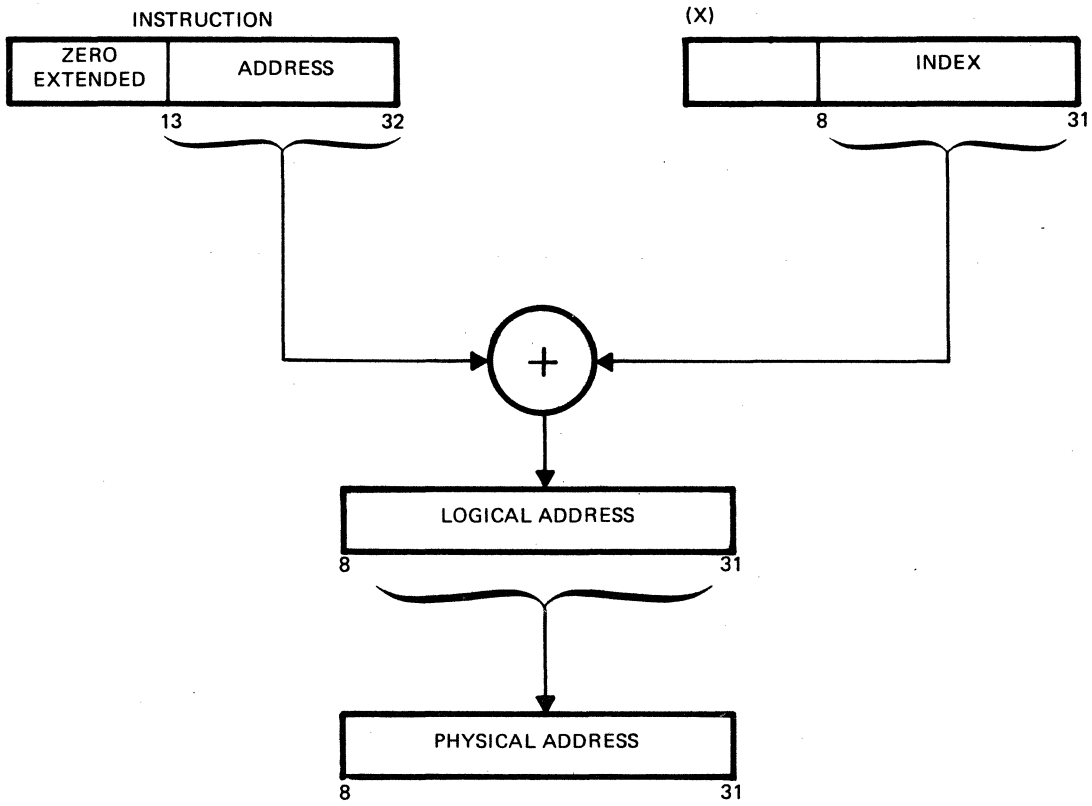


Figure 4-7. MAP Image Descriptor (MID)



NOTE: THIS METHOD MAY ADD OR SUBTRACT INDEXED ADDRESSES DEPENDING ON THE SIGN OF THE INSTRUCTION.

Figure 4-8. Address Generation (128 KW)



NOTE: THE INSTRUCTION BEING ZERO EXTENDED DOES NOT ALLOW SUBTRACTION OF INDEXED ADDRESSES.

Figure 4-9. Address Generation (512 KB Extended Mode)

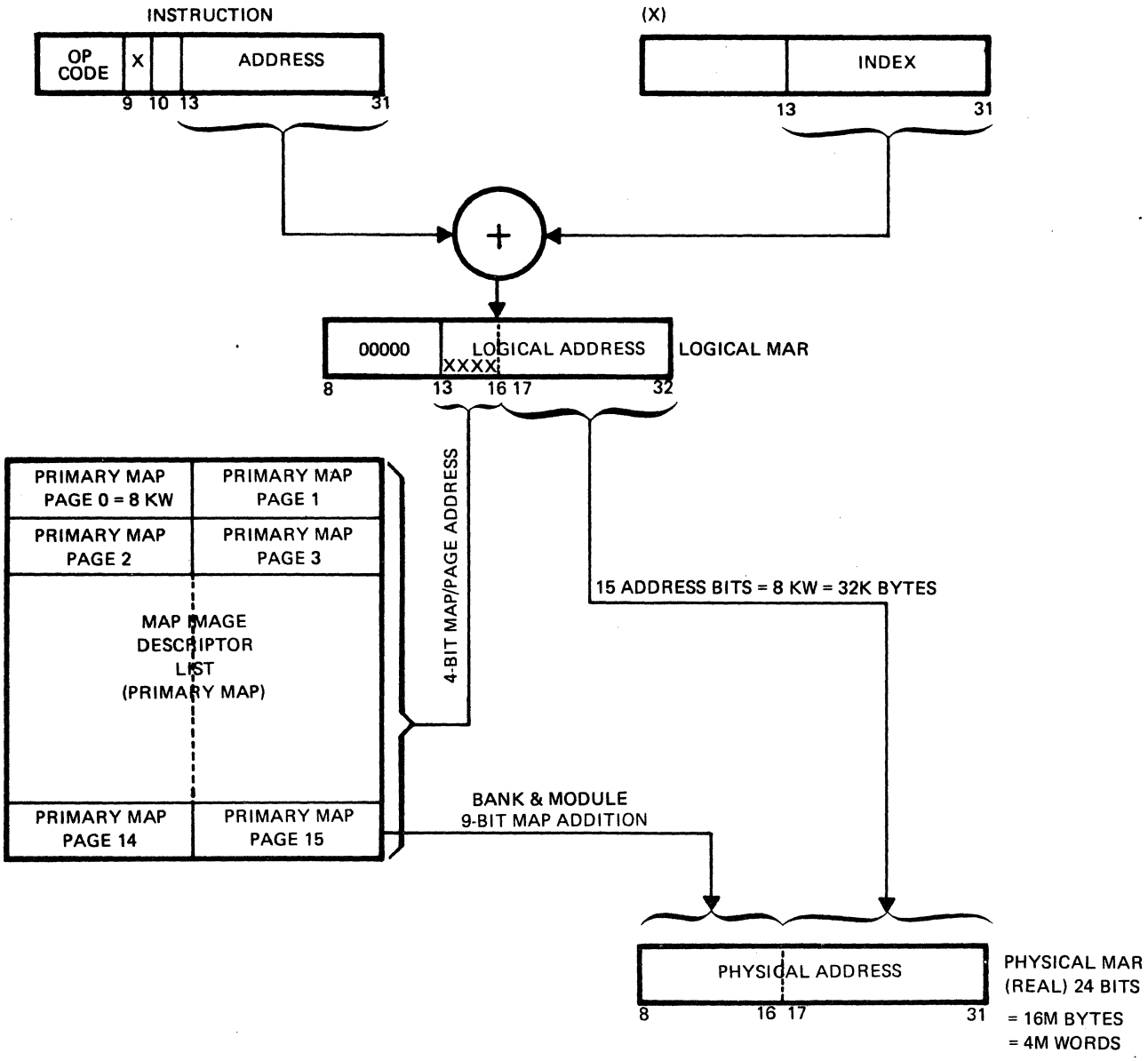


Figure 4-10. Address Generation (512 KB Mapped Mode) (Non-Extended)

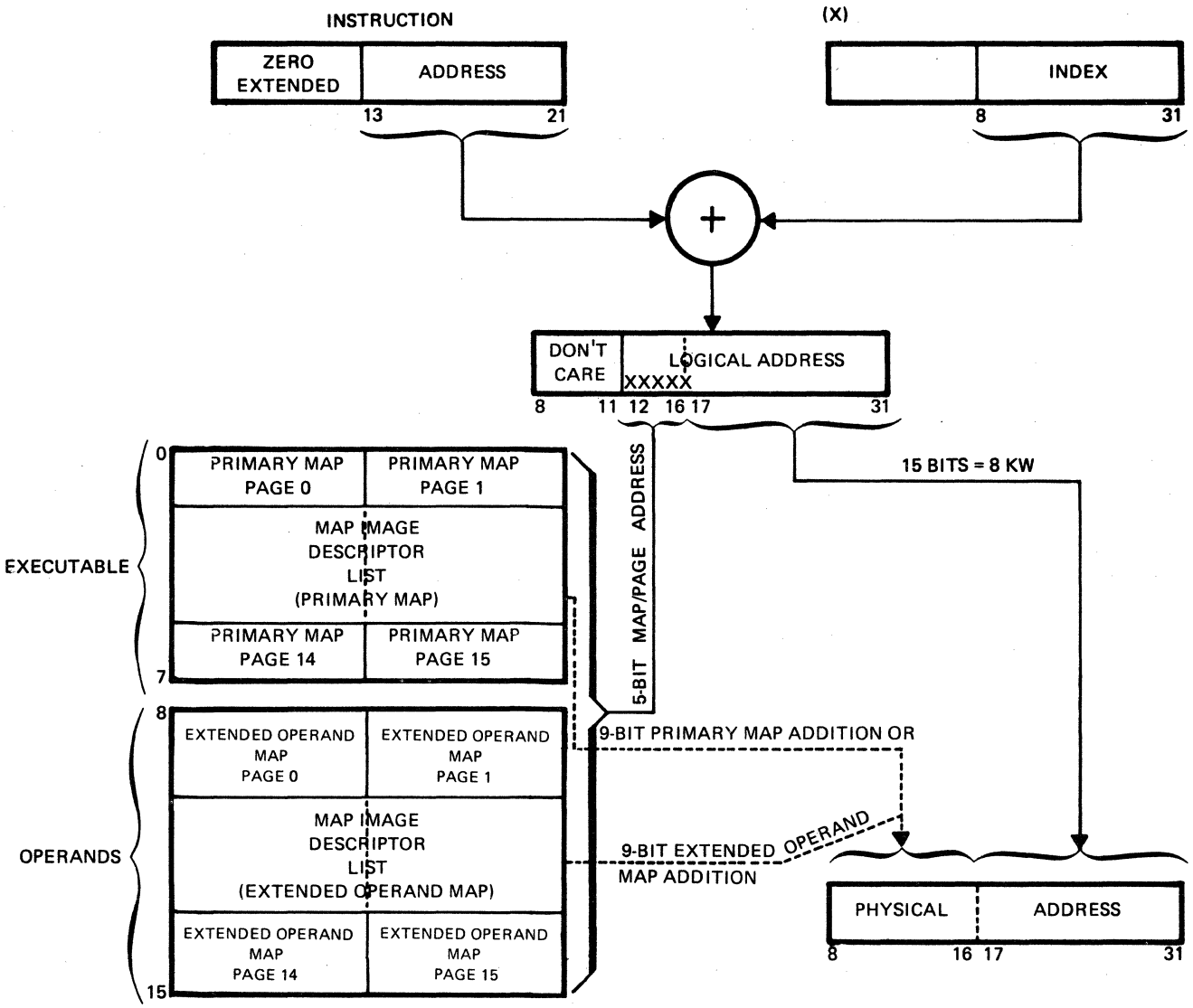


Figure 4-11. Address Generation (Mapped, Extended Mode)

SECTION V

INPUT/OUTPUT SYSTEM

INTRODUCTION

Input/Output (I/O) operations consist of transferring blocks of bytes, halfwords, or words between core memory and peripheral devices. Transfers are performed automatically, requiring minimal CPU involvement.

All system components which participate in the execution of an I/O operation are illustrated in Figure 5-1. The peripheral device(s) shown may be either data processing devices such as disc files, magnetic tape units, line printers, card readers, and card punches; or they may be real-time system devices such as data acquisition subsystems, communications control units, or system control units.

There are two modes of I/O operation possible, the first being the Program Status Word (PSW) mode which responds only to Class 0, 1, 2, 3, and E I/O processors. The second is the Program Status Doubleword (PSD) mode, which will respond to all of the preceding I/O processors as well as Class F I/O processors.

The I/O processors used in a 32/70 Series computer are available in three types. The first type is the standard Input/Output Microprogrammable Processor (IOM) containing a SelBUS interface, Microprogrammable Processor, and Device Dependent logic. The second type of I/O processor is the Integrated Channel Controller, also known as the Regional Processing Unit (RPU) (Figure 5-2) which combines the functions of a channel and a controller into one unit. The function of a channel is to schedule the requests for main memory between a number of controllers. The channel also interfaces the controller with the CPU to initiate or terminate an I/O operation. The third type of I/O processor is the General Purpose Multiplexer Controller (GPMC) and General Purpose Device Controller (GPDC) combination. The GPMC functions as the SelBUS interface, and as the decode and control logic for up to 16 device addresses. The GPMC also controls a number of independent device controllers that are located some distance from itself. The independent device controllers (GPDCs) function as device interface logic for one or more devices per GPDC.

DEFINITIONS

The following definitions are presented to aid in understanding the Input/output operations.

1. I/O Processor-The entire subsystem that interfaces the SelBUS and provides I/O ports to the devices.

2. External Media-A general term for punched cards, printed forms, magnetic tape, or discs.

3. Input/Output Devices-The peripheral devices interfaced to a 32/70 Series computer, e.g., card reader, card punch, paper tape reader, paper tape punch, line printer, and magnetic tape drives.

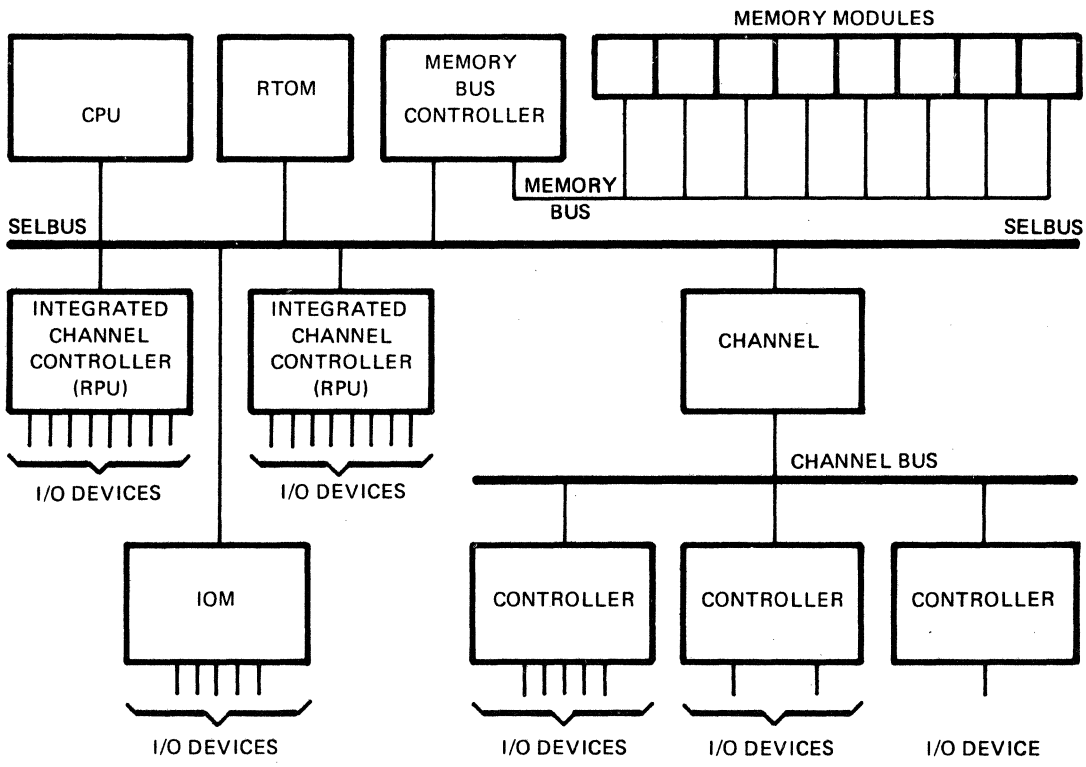


Figure 5-1. 32/70 Series Input/Output Organization

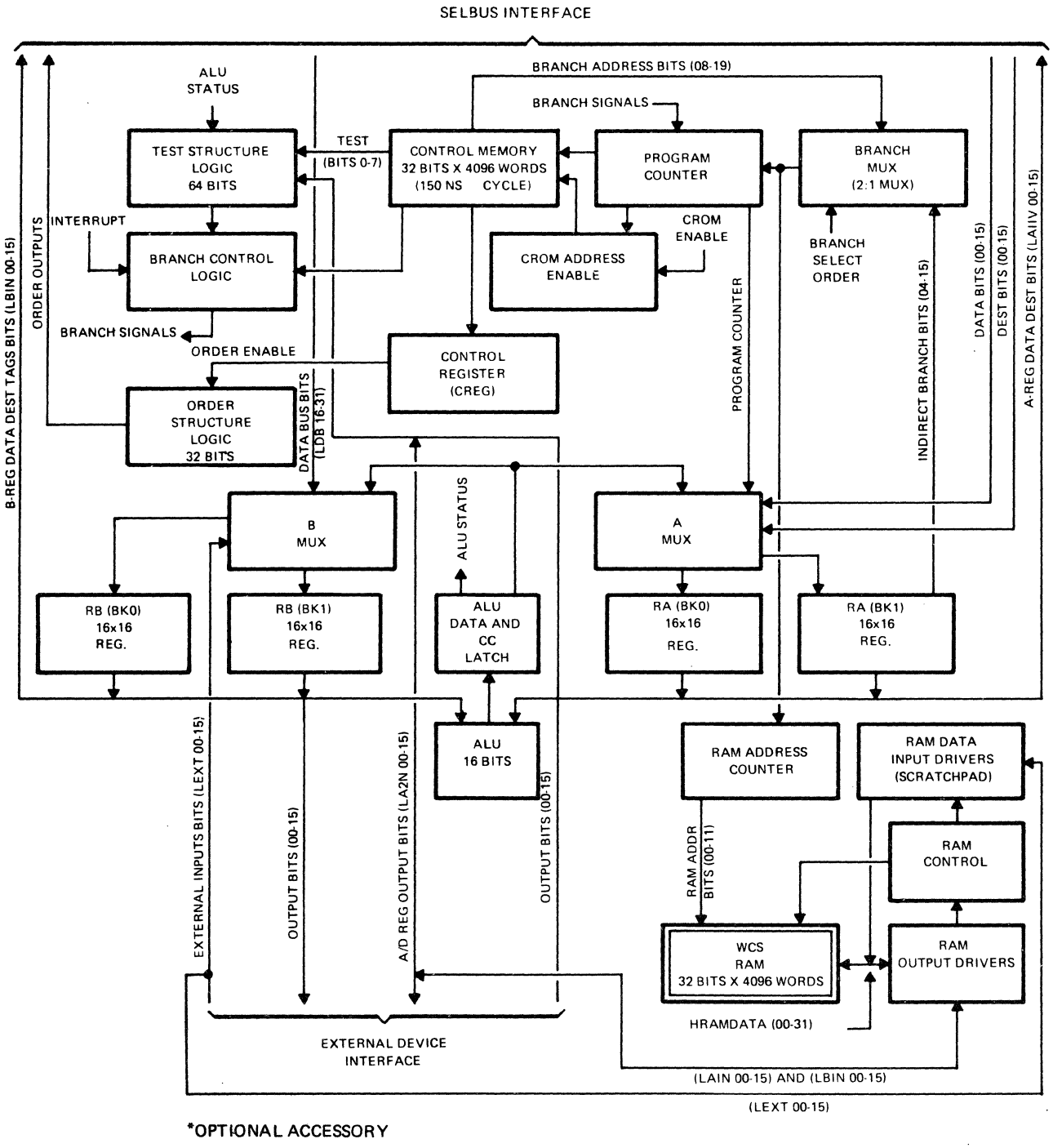


Figure 5-2. Block Diagram - Regional Processing Unit (RPU)

4. Direct Access Devices-A type of storage device wherein access to the next position from which information is to be obtained is in no way dependent on the position from which information was previously obtained. Magnetic disc drives and magnetic drums are examples of direct access devices.

5. Communications Devices-Real-time devices, such as teletypewriters and process control devices, that interface to a 32/70 Series computer.

6. Controllers-A general term used to describe the peripheral device interface logic. One controller may handle several devices.

7. Channel-That portion of an I/O processor containing the logic to interface the SelBUS and to control the device interface logic. One channel may handle one or more controllers.

8. Commands-Commands are directives that are decoded and executed by the channel, controller, and I/O device to initiate the I/O operation.

9. Instructions-Directives to the CPU that are decoded and executed by the CPU. Instructions are a part of the CPU program.

10. Command List-One or more commands arranged for sequential execution.

11. Data Chaining-Data Chaining is specified by a flag in the IOCD and causes a channel to fetch the next IOCD when the byte count in the current IOCD reaches zero.

12. Local Store-Another name for the CPU scratchpad memory.

13. Channel End-A termination condition that indicates all information associated with the operation has been received or provided, and that the channel and controller are no longer needed. This condition resets all conditions in the CPU scratchpad pertaining to the specific channel and controller.

14. Device End-An indication from the controller to the channel that an I/O device has terminated execution of its operation.

15. Controller End-Operations that keep the controller busy after reporting a Channel End cause Controller End reporting (at the end of its operation) indicating that the controller is available for initiation of another operation.

I/O PROCESSOR CLASSIFICATION

I/O processors are classified as types 0, 1, 2, 3, E, and F. The type 0, 1, and 2 I/O processors are associated with the teletype, line printer, and card reader respectively, and are contained on a single IOM. The type 3 I/O Processor is the RTOM Interval Timer. A type E I/O processor is one which is controlled by the use of the Command Device (CD) and Test Device (TD) instructions and has the capability of only addressing 512 KB of memory. The type F I/O processor responds to the 32/70 Series I/O instructions, has the capability of addressing memory throughout a 16 MB range, and in some cases supports an optional Writable Control Storage (WCS) unit.

OPERATION WITH CLASS 0, 1, 2, AND E I/O PROCESSORS

Input/Output (I/O) operations with the Class 0, 1, 2, and E I/O processors consist of transferring blocks of bytes, halfwords, or words between core memory and peripheral devices. Core memory locations addressed by these I/O processors are limited to the first 128K words (512K bytes) of contiguous memory. Transfers are possible at rates up to 1.2 million bytes per second. The system components which participate in the execution of an I/O operation are illustrated in Figure 5-3.

A 32/70 Series system will support a total of 16 I/O processors. Each I/O processor may in turn support as many as 16 device addresses, allowing as many as 128 separate addressed devices to be connected to a 32/70 Series computer at one time.

Two types of I/O instructions, Command Device (CD) and Test Device (TD), are executable by Class 0, 1, 2, and E I/O processors.

COMMAND DEVICE INSTRUCTION

Transfer of a block of information is initiated by execution of a Command Device instruction in the CPU. This instruction, illustrated in Figure 5-4, specifies the device, the direction of transfer, and other control parameters required to condition the device to generate or accept data. The control parameters are defined in Figure 5-5. The I/O processor, consisting of an IOM and Device Dependent logic, accepts the Command Device from the CPU, routes the device control parameters to the device specified in the instruction, and initializes the transfer of a block of data. A Transfer Control Word contains the starting memory address and the number of transfers to be made, and is contained in a memory location dedicated to each device address.

TRANSFER CONTROL WORD

The Transfer Control Word (TCW) contains a 20-bit address which defines the memory location for each transfer. It also contains a positive 12-bit binary Transfer Count (TC). The Transfer Count plus the Format Code (FC) permits transfers of blocks of information having any number of bytes, halfwords, or words up to 4,096. The format of the Transfer Control Word (TCW) is shown in Figure 5-6.

The presence of the Format Code in the TCW permits transfers of bytes, halfwords, or words. The Format Code is designed such that when F is equal to One in a given TCW, the address is incremented in bit position 31 each time a transfer occurs. Therefore, each transfer is stored in or read from a consecutive byte in memory in this order:

Word N	Word N+1
---Byte 0,Byte 1,Byte 2,Byte 3	Byte 0,Byte 1,Byte 2,Byte 3---

The proper binary value of Format Code for accessing consecutive halfwords in memory is F equal to 0, C equal to Y1, where Y equal to Zero designates left halfword and Y equal to One designates right halfword. With this value of Format Code, the address is incremented in bit position 30 each time a transfer is made. This results in the desired accessing of consecutive halfwords.

The proper value of Format Code for consecutive word accessing is TCW equal to 000. When this value is present in a given TCW, the I/O processor increments the TCW in bit position 29 each time a transfer occurs.

The Format Code values discussed above are summarized in Table 5-1.

Each time the address is incremented, the Transfer Count is decremented. Therefore, the block length is always defined by the number of memory accesses and not by the number of words transferred. For specific I/O processors (i.e., GPMC, HSD, ADI, and FMS), the TCW address field is used to supply an Input/ Output Command Doubleword (IOCD) address.

The dedicated memory addresses used with the 16 I/O Processors are included in the list of Relative Trap/Interrupt Priorities (reference Table 3-1).

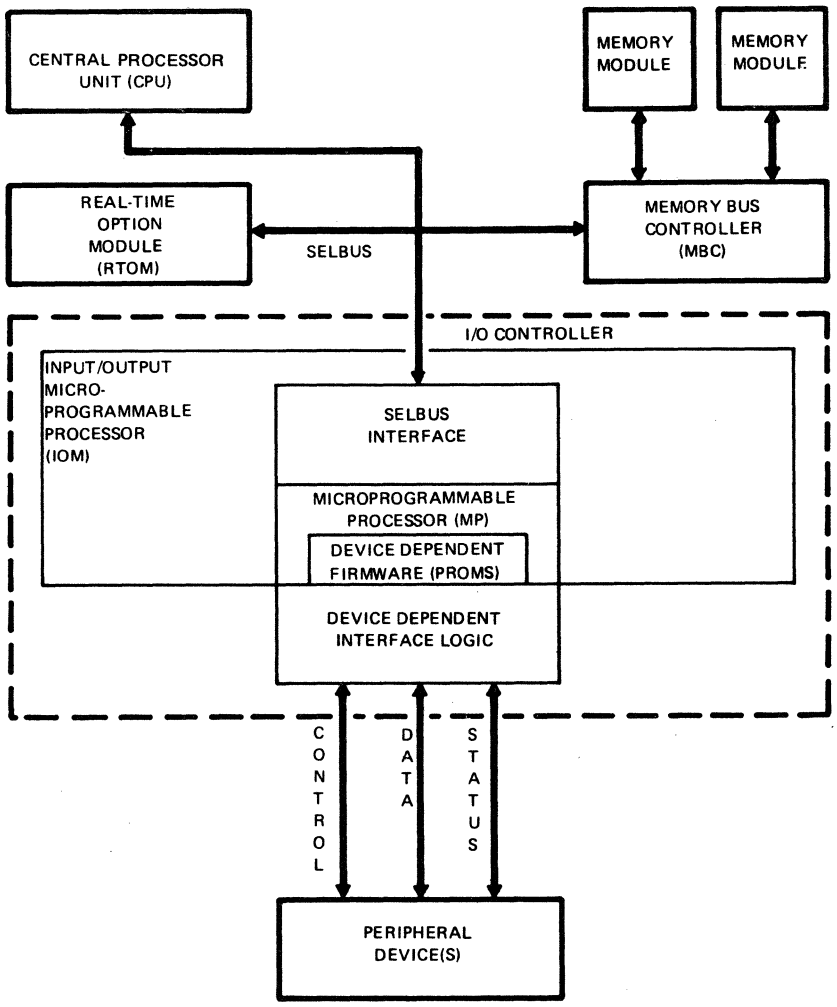
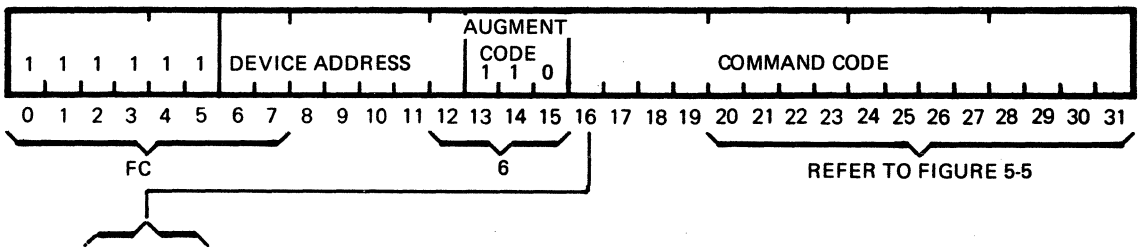


Figure 5-3. Class 0, 1, 2, and E I/O Organization



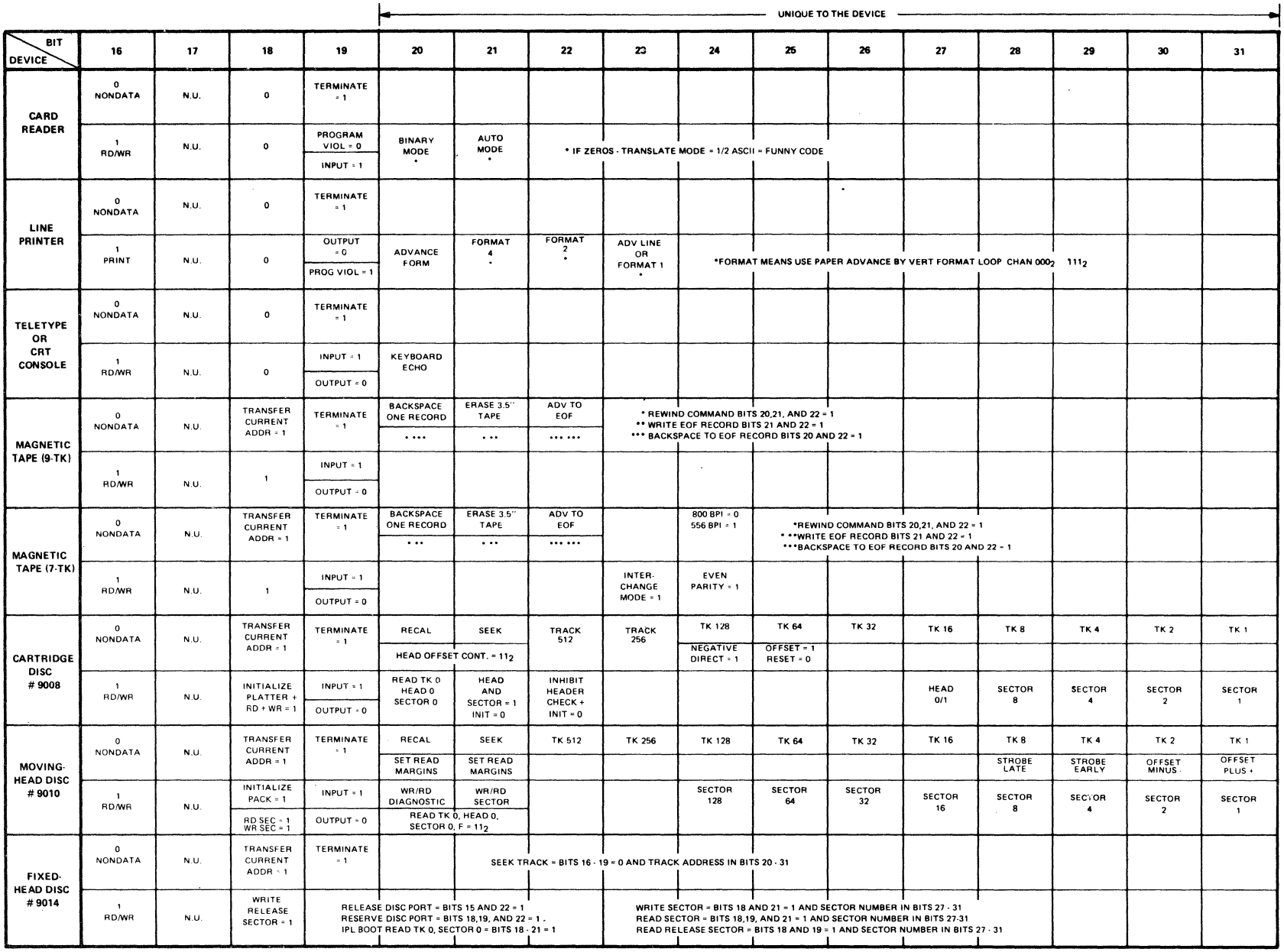
BIT 16 = 0 BIT POSITIONS 20 THROUGH 31 OF THE FUNCTION CODE ARE UNIQUE TO THE DEVICE
BIT POSITIONS 18 AND 19 PROVIDE THE FOLLOWING INFORMATION:

BIT 18 = 1 TRANSFER CURRENT WORD ADDRESS
BIT 19 = 1 TERMINATE (RESET I/O CONTROLLER)

BIT 16 = 1 A TRANSFER IS TO BE INITIALIZED AND BITS 18 AND 19 OF THE FUNCTION CODE WILL PROVIDE THE FOLLOWING INFORMATION:

BIT 19 = 0 OUTPUT TRANSFER (WRITE)
BIT 19 = 1 INPUT TRANSFER (READ)

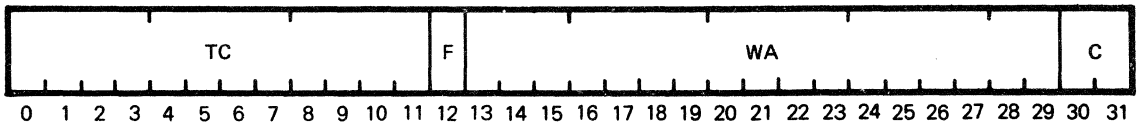
Figure 5-4. Command Device Instruction Format



UNIQUE TO THE DEVICE

BIT / DEVICE	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CARD READER	0 NONDATA	N.U.	0	TERMINATE = 1												
	1 RD/WR	N.U.	0	PROGRAM VIOL = 0 / INPUT = 1	BINARY MODE *	AUTO MODE *	* IF ZEROS - TRANSLATE MODE = 1/2 ASCII = FUNNY CODE									
LINE PRINTER	0 NONDATA	N.U.	0	TERMINATE = 1												
	1 PRINT	N.U.	0	OUTPUT = 0 / PROG VIOL = 1	ADVANCE FORM	FORMAT 4 *	FORMAT 2 *	ADV LINE OR FORMAT 1 *	*FORMAT MEANS USE PAPER ADVANCE BY VERT FORMAT LOOP CHAN 000_2 111_2							
TELETYPE OR CRT CONSOLE	0 NONDATA	N.U.	0	TERMINATE = 1												
	1 RD/WR	N.U.	0	INPUT = 1 / OUTPUT = 0	KEYBOARD ECHO											
MAGNETIC TAPE (9-TK)	0 NONDATA	N.U.	TRANSFER CURRENT ADDR = 1	TERMINATE = 1	BACKSPACE ONE RECORD * •••	ERASE 3.5" TAPE • ••	ADV TO EOF ••• •••	* REWIND COMMAND BITS 20,21, AND 22 = 1 ** WRITE EOF RECORD BITS 21 AND 22 = 1 *** BACKSPACE TO EOF RECORD BITS 20 AND 22 = 1								
	1 RD/WR	N.U.	1	INPUT = 1 / OUTPUT = 0												
MAGNETIC TAPE (7-TK)	0 NONDATA	N.U.	TRANSFER CURRENT ADDR = 1	TERMINATE = 1	BACKSPACE ONE RECORD • ••	ERASE 3.5" TAPE • ••	ADV TO EOF ••• •••		800 BPI = 0 556 BPI = 1	*REWIND COMMAND BITS 20,21, AND 22 = 1 **WRITE EOF RECORD BITS 21 AND 22 = 1 ***BACKSPACE TO EOF RECORD BITS 20 AND 22 = 1						
	1 RD/WR	N.U.	1	INPUT = 1 / OUTPUT = 0				INTER-CHANGE MODE = 1	EVEN PARITY = 1							
CARTRIDGE DISC # 9008	0 NONDATA	N.U.	TRANSFER CURRENT ADDR = 1	TERMINATE = 1	RECAL	SEEK	TRACK 512	TRACK 256	TK 128	TK 64	TK 32	TK 16	TK 8	TK 4	TK 2	TK 1
					HEAD OFFSET CONT. = 11_2				NEGATIVE DIRECT = 1	OFFSET = 1 RESET = 0						
	1 RD/WR	N.U.	INITIALIZE PLATTER + RD + WR = 1	INPUT = 1 / OUTPUT = 0	READ TK 0 HEAD 0 SECTOR 0	HEAD AND SECTOR = 1 INIT = 0	INHIBIT HEADER CHECK + INIT = 0					HEAD 0/1	SECTOR 8	SECTOR 4	SECTOR 2	SECTOR 1
MOVING-HEAD DISC # 9010	0 NONDATA	N.U.	TRANSFER CURRENT ADDR = 1	TERMINATE = 1	RECAL	SEEK	TK 512	TK 256	TK 128	TK 64	TK 32	TK 16	TK 8	TK 4	TK 2	TK 1
					SET READ MARGINS	SET READ MARGINS							STROBE LATE	STROBE EARLY	OFFSET MINUS -	OFFSET PLUS +
	1 RD/WR	N.U.	INITIALIZE PACK = 1 RD SEC = 1 WR SEC = 1	INPUT = 1 OUTPUT = 0	WR/RD DIAGNOSTIC	WR/RD SECTOR			SECTOR 128	SECTOR 64	SECTOR 32	SECTOR 16	SECTOR 8	SECTOR 4	SECTOR 2	SECTOR 1
					READ TK 0, HEAD 0, SECTOR 0, F = 11_2											
FIXED-HEAD DISC # 9014	0 NONDATA	N.U.	TRANSFER CURRENT ADDR = 1	TERMINATE = 1	SEEK TRACK = BITS 16 - 19 = 0 AND TRACK ADDRESS IN BITS 20 - 31											
	1 RD/WR	N.U.	WRITE RELEASE SECTOR = 1	RELEASE DISC PORT = BITS 15 AND 22 = 1 RESERVE DISC PORT = BITS 18,19, AND 22 = 1 IPL BOOT READ TK 0, SECTOR 0 = BITS 18 - 21 = 1				WRITE SECTOR = BITS 18 AND 21 = 1 AND SECTOR NUMBER IN BITS 27 - 31 READ SECTOR = BITS 18,19, AND 21 = 1 AND SECTOR NUMBER IN BITS 27-31 READ RELEASE SECTOR = BITS 18 AND 19 = 1 AND SECTOR NUMBER IN BITS 27 - 31						

Figure 5-5. Command Device Function Bit Format For Peripheral Devices



BITS 0-11 DESIGNATE THE NUMBER OF TRANSFERS TO BE MADE BETWEEN MEMORY AND THE DEVICE CONTROLLER CHANNEL.

BITS 12,30,31 SPECIFY THE FORMAT CODE FOR EACH TRANSFER (SEE TABLE 5-1).

BITS 13-29 DESIGNATE THE MEMORY LOCATION FOR EACH TRANSFER.

NOTE

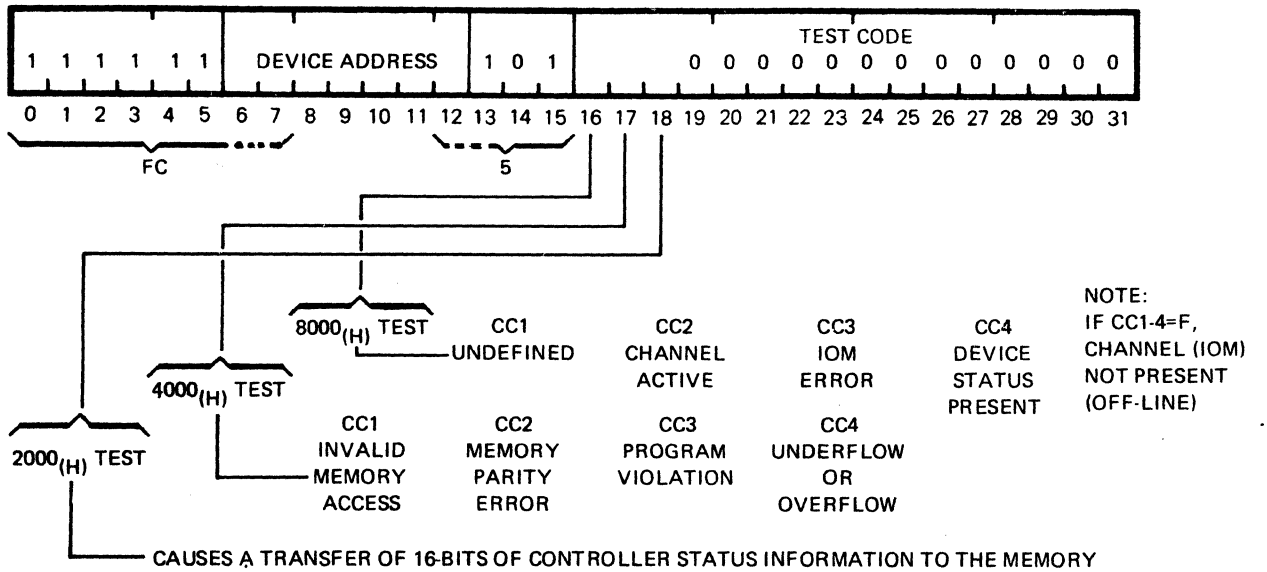
THE WA FIELD IS INTERPRETED AS A 24-BIT REAL ADDRESS BY THE I/O PROCESS. THEREFORE, THE ADDRESS RANGE IS LIMITED TO THE FIRST 512 KB OF MEMORY.

Figure 5-6. Transfer Control Word Format

Table 5-1. Transfer Control Word Format Code

Information Format	FC
Byte Halfword Word	1XX 0Y1 000

XX = Byte number
Y = 0 designates left halfword
Y = 1 designates right halfword



CAUSES A TRANSFER OF 16-BITS OF CONTROLLER STATUS INFORMATION TO THE MEMORY LOCATION SPECIFIED IN THE TCW DEDICATED LOCATION. THE MEANING OF EACH BIT IN THE 16-BIT STATUS HALFWORD DIFFERS ACCORDING TO DEVICE TYPE. SEE FIGURE 5-8.

CC2 = 0 STATUS TRANSFER WAS PERFORMED
CC2 = 1 STATUS TRANSFER WAS NOT PERFORMED
CC4 = 1 CONTROLLER IS ABSENT OR POWERED OFF

Figure 5-7. Test Device Instruction Format

UPPER HW	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LOWER HW	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LINE PRINTER	0	PROG VIOL	DEV INOP	0	0	0	0	0	0	BOF	0	0	0	DEV BUSY	0	0
MAG TAPE	0	PROG VIOL	DEV INOP	VRC ERROR	0	REW IN PROG	CRC LRC	0	0	EOT	BOT	EOF	0	DEV BUSY	FILE PROT VIO	ODD REC LGT
MOVING-HEAD DISC	0	PROG VIOL	DEV INOP	UNCORR DATA ERROR	0	FILE UN-SAFE	SEEK IN PROG	CORR DATA ERROR	0	0	ADDR ERROR	0	0	0	0	SEEK TRACK ERROR
FIXED-HEAD DISC	0	PROG VIOL	DEV INOP	CHK SUM	0	0	0	0	0	0	SECTOR ERROR	0	MUX BSY (DUAL CPU)	0	FILE PROT VIO	SEEK TRACK ERROR
CARD READER/ PUNCH	0	0	FILE MARK RD	READ CHECK	0	STACKER FULL	PUNCH CHECK	HOPPER EMPTY	0	PICK FAILURE	TRANSMIT ERROR	INCORRECT LENGTH	UNWS CHAN END	ILLEGAL END	INT PEND	CHAN END

THE STATUS HALFWORD IS STORED IN THE MEMORY HALFWORD SPECIFIED BY THE ASSOCIATED TRANSFER CONTROL WORD (TCW).

Figure 5-8. Test Device 2000 Status Information

TEST DEVICE INSTRUCTION

The Test Device (TD) instruction is used to acquire status information from the Input/Output processor and the associated device(s). Three levels of the TD instruction (8000, 4000 and 2000) may be used to acquire this information. The status information is in the form of four condition code bits for each level of test. The TD instruction does not initiate any action in the device. The TD 8000 instruction is used by the CPU to test the general status of the addressed device and associated I/O processor. The TD 4000 instruction is used by the CPU to allow further definition of the errors indicated in the TD 8000. The TD 2000 instruction is used by the CPU to obtain 16 bits of status information from the device/processor. This instruction causes the addressed I/O processor to transfer a 16-bit status word to the memory address specified by the TCW. The 16-bit status word may be placed in memory in either the right or left halfword position, depending on bits 30 and 31 of the TCW address. A TCW used with a TD 2000 should always specify halfword memory addressing. Figure 5-7 provides a breakdown of the Test Device instruction format. Figure 5-8 provides the status information returned from standard peripheral devices upon execution of TD 2000 instructions.

INPUT/OUTPUT PROCESSOR

Each Input/Output processor consists of an Input/Output Micropro grammable Processor (IOM) and Device Dependent Interface logic. The Microprogrammable Processor (MP) and the Device Dependent Interface logic are customized for each device. The firmware for a given Input/Output processor is contained in a set of PROMs that plug into the processor board. The information contained within the PROMs is device dependent.

This design technique provides extreme flexibility for custom designed interfaces since the basic MP and SelBUS interface are also available as a General Purpose I/O Processor (GPIO). All that is needed to convert the GPIO processor into a special purpose I/O processor is the Device Dependent Interface logic and the firmware microprogram.

The maximum throughput of an Input/Output processor is 1.2 million bytes per second.

There are two types of Input/Output processors:

1. Multiple Device Controller (MDC)

2. Multiple Controller Controller (MCC)

The MDC controls like devices, such as four magnetic tapes. The MCC emulates multiple controllers such as the TLC Input/Output processor that controls a teletype, a card reader, and a printer. MCC Input/Output processors are multiplexed processors handling more than one device simultaneously accessing memory. The Asynchronous Data Set Interface (ADS) is an example of a multiplexed processor. The ADS handles four half- or full-duplex lines directly to memory on a message basis. Four memory input buffers and four output buffers can be active at one time.

SelBUS INTERFACE

The Input/Output SelBUS interface contains the registers and SelBUS drivers for a full 32-bit data transfer. The main function of this logic is to receive and drive communications on the SelBUS. All the interface control logic, including processor address recognition, interrupt polling, and data transfer to and from the SelBUS, are included in the interface.

The bus priority logic is controlled by the interface control logic. It polls for the SelBUS, determines when it wins the poll, and then drives the transfer on the bus. Priorities are set through physical switches in the Input/Output processor.

TRANSFER RESPONSES

An Input/Output processor will respond to all bus transfers that it receives. It has three immediate responses:

1. Retry
2. Busy
3. Transfer Acknowledge

The sending bus device can determine the status of its transfer to the Input/ Output processor by monitoring these lines. A Retry answer means that the Input/Output processor of the MCC type is temporarily busy. A Busy means to set the busy condition code bit in the software instruction and proceed with the next instruction. An Input/Output processor of the MDC type would generate such a return. A Transfer Acknowledge indicates that the transfer was accepted and is being processed. If no answer is present in the bus cycle following the transfer, a non-present Input/Output processor was addressed.

IOM DATA STRUCTURE

The IOM data structure provides for the transfer of data, arithmetic and logical manipulation of data, storing of device and processor status, decoding of commands, and data buffering. Figure 5-9 provides a block diagram of the IOM.

Two 16- by 16-bit word register groups, RA and RB, are available as working read/write memory. The output for each register pair is the input to the Arithmetic/Logic Unit.

The destination address and the most significant 16 bits of the data bus are directed to the RA register group. The program counter and the ALU output are also directed to the RA register group. The least significant 16 bits of the data bus and 16 bits of data from the peripheral devices are directed to the RB register group. The ALU output and a 16-bit literal from the control register are also input to the RB register group.

ARITHMETIC LOGIC UNIT

The data structure includes a full 16-bit Arithmetic/Logic Unit which inputs from RA and RB. The ALU is equipped with a 3-bit status register which contains previous carry, all zeros condition, and the most significant bit.

DATA STRUCTURE CONTROL

A 32-bit by 1,024-word microprogrammed control memory and a 48-bit test structure (32 implemented) control the flow of data and commands between the SelBUS and peripheral devices.

TEST STRUCTURE

The IOM test structure is used with the Wait and Conditional Branch operations to control the sequencing and timing of instructions.

INTERRUPTS

The IOM has a single Master Interrupt line. For device controllers requiring more interrupts, the necessary mask register and Priority Decode logic is included in the Device Interface logic.

CLASS F I/O OPERATION

The following discussions refer to the organization and operation of Series Class F I/O processors.

Class F Input/Output operations consist of transferring blocks of bytes, halfwords, or words between core memory and the peripheral devices. Transfers are performed automatically requiring a minimum of CPU involvement.

A typical configuration for Class F I/O operation is illustrated in Figure 5-10. The I/O devices include card readers, line printers, discs, magnetic tapes, and telecommunications equipment. The controller provides the logical and buffering capabilities necessary to operate an I/O device. The controller is attached to a channel. The channel's function is to schedule the requests for main memory between a number of controllers. The channel also connects the controller to the CPU to initiate or terminate an I/O operation.

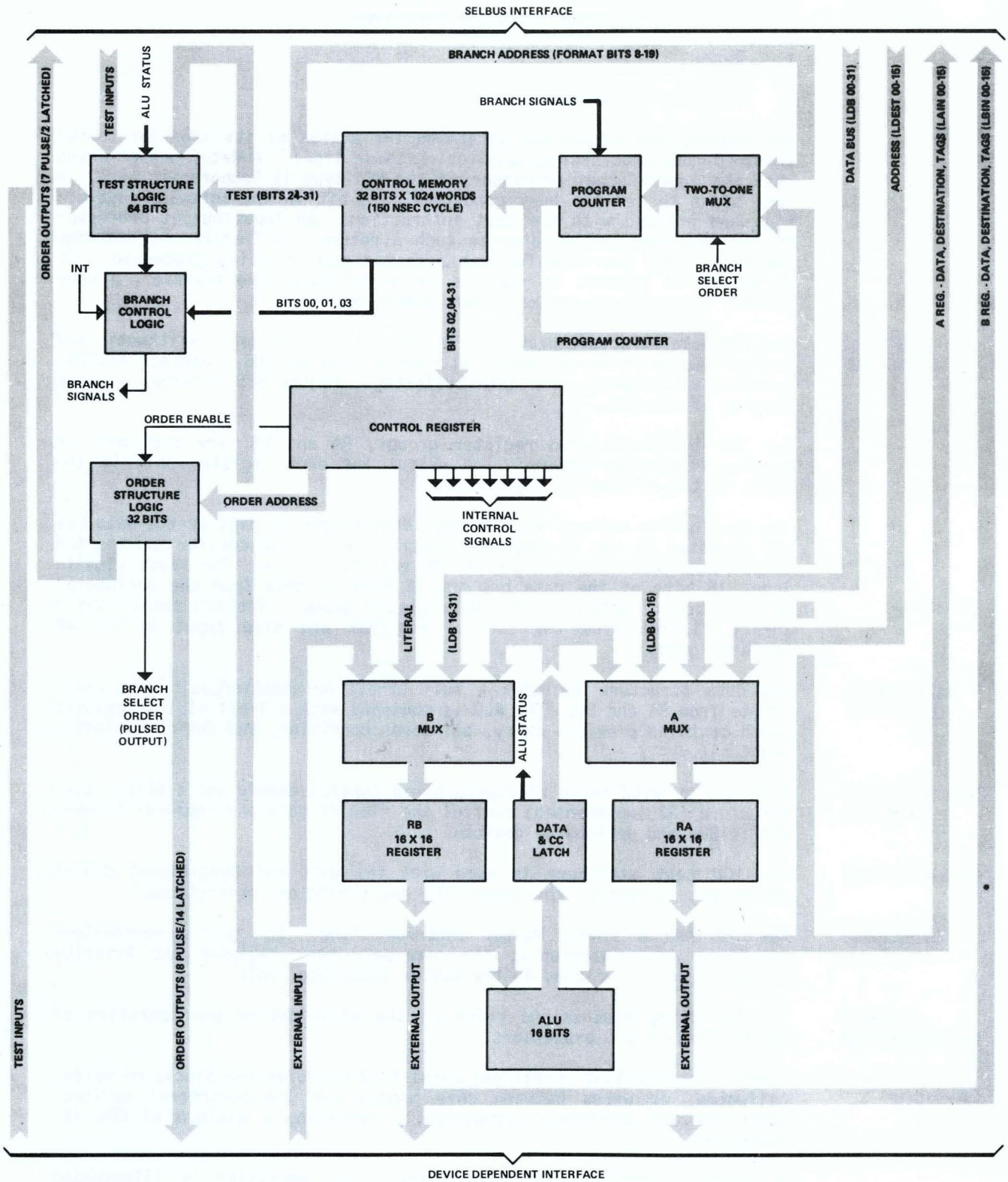


Figure 5-9. Block Diagram - I/O Microprogrammable Processor

The integrated channel controller, also known as the RPU, combines the functions of a channel and a controller into an indistinguishable unit.

CLASS F I/O PROCESSOR

An I/O processor consists of two or more distinct logic subassemblies which are:

1. The Channel-which interfaces with the SelBUS to send and receive information between the channel, the CPU, and/or memory. The other side of the channel interfaces with one or more controllers to provide control signal and data paths to/from the controllers.

2. The Controller-which interfaces between the channel and the device itself. The purpose of the controller is to provide the proper protocol for the device and to convert that protocol to a standard protocol for use by the channel.

3. Writable Control Storage-which interfaces the channel, provides a source of Read/Write memory for the channel. The use of the Writable Control Storage is to customize an I/O processor for specific uses. The Writable Control Storage is loaded by special software instructions and may contain any program the user requires.

The main subassemblies common to all Class F I/O processors are the controller and channel, with the Writable Control Storage being an option.

Dedicated memory locations are associated with each I/O processor and provide main memory locations to transmit or receive control information required to initiate or terminate an I/O operation. The control information consists of:

1. Service Interrupt Vector Address

2. Input/Output Command Doubleword (IOCD) Address

3. Status Address

4. New Program Status Doubleword (PSD)

5. Old Program Status Doubleword (PSD)

A graphic representation of the I/O control words is shown in Figure 5-11.

MEMORY ADDRESSING METHOD

Memory addresses are transferred to the channel when a Start I/O (SIO) or Write Channel Write Control Storage (WCWCS) instruction is executed by the CPU. Prior to the execution of the I/O instruction, the software stores the address of the first Input/Output Command Doubleword (IOCD) to be executed into the word indicated by adding 20 (decimal) to the contents of the Service Interrupt Vector (SIV). The word indicated is referred to as the Input/Output Command List Address (IOCLA).

The memory addressing method used for Class F I/O is real addressing. Real addressing is the capability to directly address any memory location within the 16 MB maximum capacity of the system without any address translation. This method of addressing differs from the method normally used by the software programmer, who relies on a hardware address conversion to transform the logical address to a real address in order to address memory locations greater than 512K bytes.

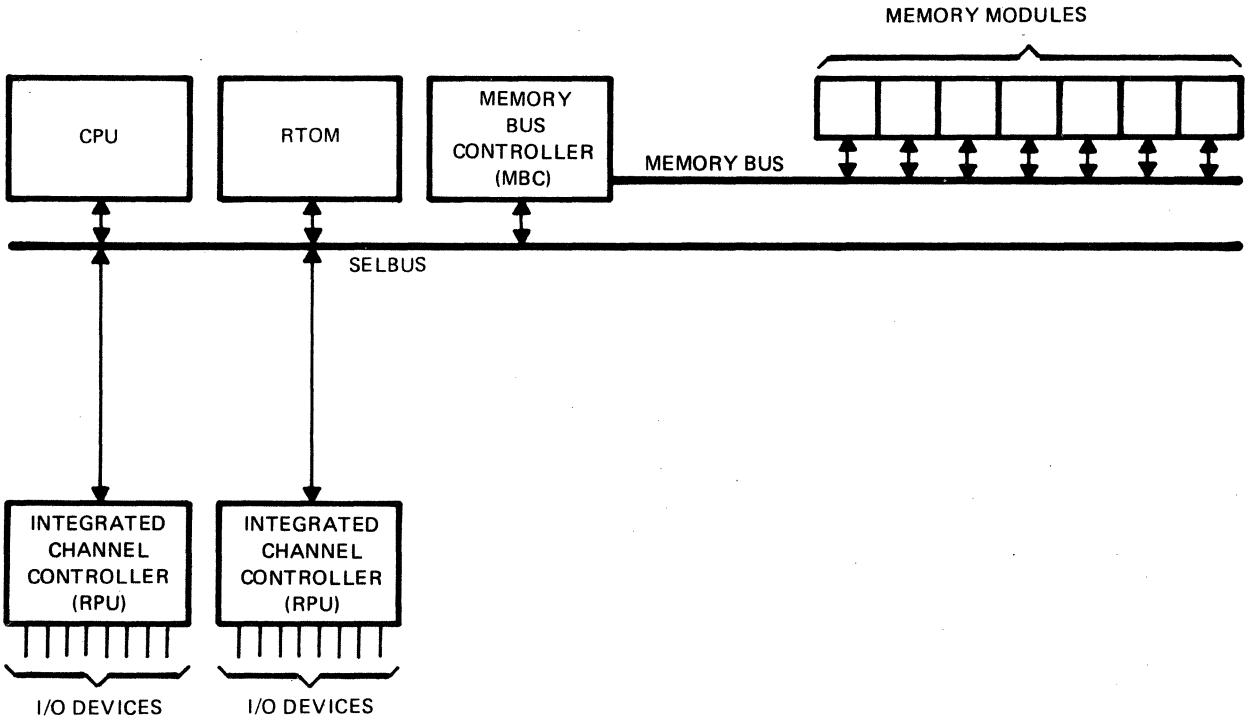


Figure 5-10. System Configuration with Class F I/O Processor

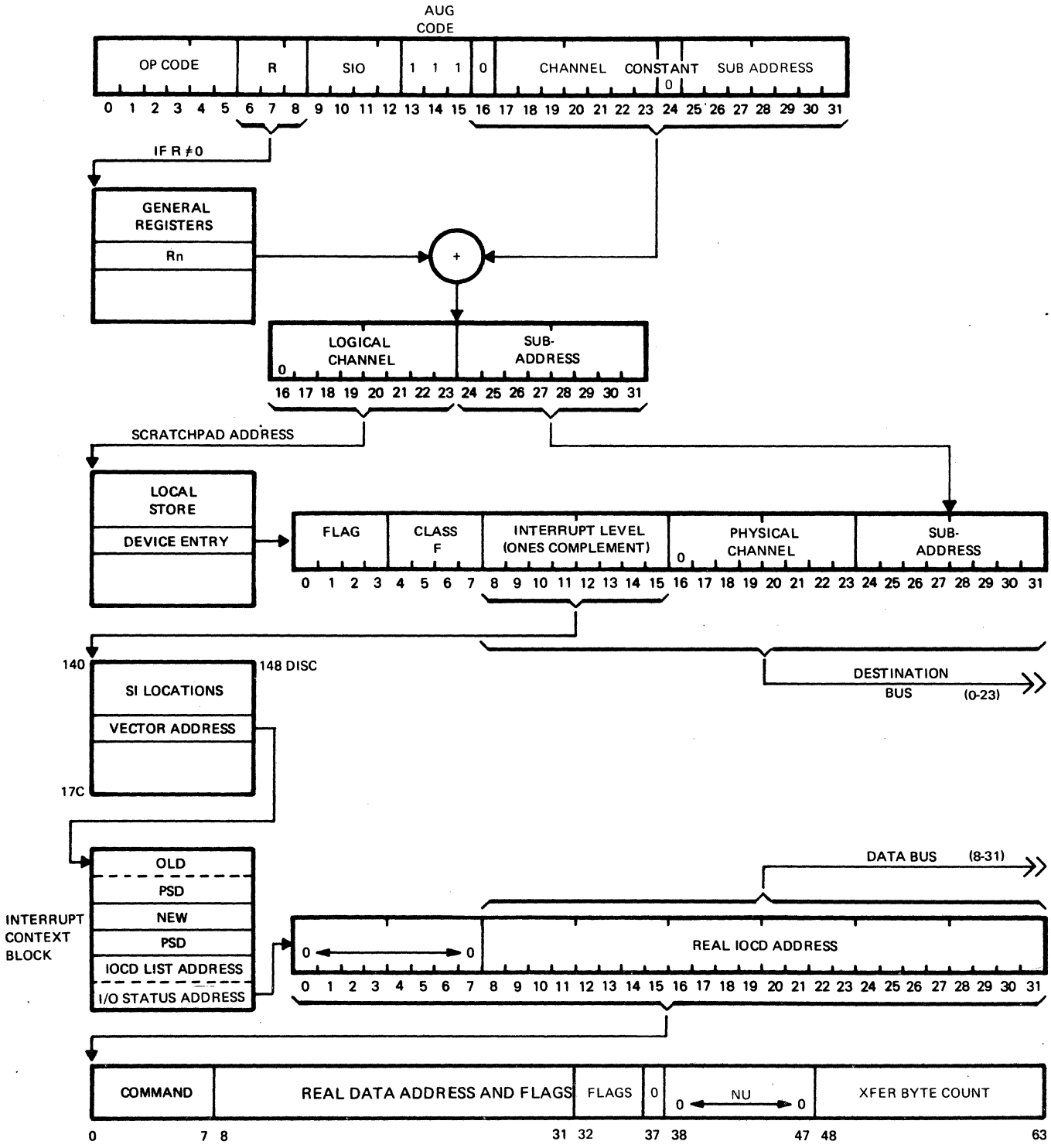


Figure 5-11. I/O Control Words (Class F)

PSD MODE I/O INSTRUCTIONS

When operating in the PSD mode, a set of special instructions augments or replaces those used for the PSW mode of operation. The PSD I/O instructions include the following:

1. Start I/O (SIO)
2. Test I/O (TIO)
3. Halt I/O (HIO)
4. Stop I/O (STPIO)
5. Grab Controller (GRIO)
6. Reset Controller (RSCTL)
7. Reset Channel (RSCHNL)
8. Enable Channel WCS Load (ECWCS)
9. Write Channel WCS (WWCS)
10. Enable Channel Interrupt (ECI)
11. Disable Channel Interrupt (DCI)
12. Activate Channel Interrupt (ACI)
13. Deactivate Channel Interrupt (DACI)

For all Class F I/O instructions, the logical channel and device addresses are specified by bits 16-31 of the instruction plus the contents of the General Purpose Register (GPR) specified by the instruction (if the GPR specified is nonzero). The channel will ignore the subaddress for operations that pertain only to the channel.

The Class F I/O instructions can be executed only when the CPU is in privileged mode and operating in the PSD mode.

START I/O (SIO)

The Start I/O initiates an I/O operation. If the necessary channel, subchannel or controller is available, the SIO is accepted and the CPU continues to the next sequential instruction. The channel/controller independently governs the I/O device specified by the instruction.

TEST I/O (TIO)

The Test I/O interrogates the current state of the channel, subchannel, controller and device and may be used to clear pending interrupt conditions.

HALT I/O (HIO)

The Halt I/O terminates a channel, controller, and/or device operation.

ENABLE CHANNEL WCS LOAD (ECWCS)

The Enable Channel WCS Load conditions the channel to have its WCS loaded.

WRITE CHANNEL WCS (WCWCS)

The Write Channel WCS is the second part of a two-instruction sequence and causes the specified channel's WCS to be loaded.

ENABLE CHANNEL INTERRUPT (ECI)

The Enable Channel Interrupt allows the channel to request interrupts from the CPU.

DISABLE CHANNEL INTERRUPT (DCI)

The Disable Channel Interrupt prohibits the channel from requesting an interrupt. Pending status conditions can only be cleared by the execution of a Start I/O, Test I/O, or Halt I/O if the channel is disabled.

ACTIVATE CHANNEL INTERRUPT (ACI)

The Activate Channel Interrupt causes the channel to actively contend for interrupt priority except that the channel never requests an interrupt. The instruction has no effect on pending status conditions except that it can be cleared by a Start I/O, Test I/O, or Halt I/O.

DEACTIVATE CHANNEL INTERRUPT (DACI)

The Deactivate Channel Interrupt causes the channel to suspend contention for interrupt priority. If an interrupt request is queued, the channel may then request interrupt. All DACI instruction abnormalities or I/O protocol violations are connected to the System Check Trap unless an initial channel nonpresent or inoperable condition is found.

RESET CHANNEL (RSCHNL)

The Reset Channel resets all activity in the channel. All requesting and pending conditions are cleared.

STOP I/O (STPIO)

The Stop I/O terminates the operation in the controller after the completion of the current IOCD. The termination is orderly. The channel will suppress command and data chaining.

RESET CONTROLLER (RSCTL)

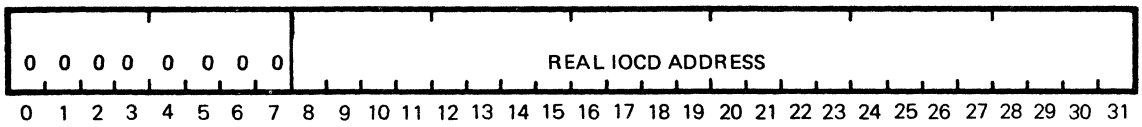
The Reset Controller resets a specific controller if the resetting channel maintains ownership. The reset is immediate.

GRAB CONTROLLER (GRIO)

The Grab Controller takes away control of a controller which is reserved to another channel. The grabbing channel is assigned as the reserving channel.

INPUT/OUTPUT COMMAND LIST ADDRESS

Successful execution of the SIO and WCWCS causes the CPU to transmit the Input/Output Command List Address (IOCLA) to the channel/controller. The IOCLA is located in main memory at locations specified by the service interrupt vector plus 16 (decimal). Each of the 16 channels has a corresponding service interrupt vector. The format for the IOCLA indicated by the contents of the service interrupt vector 11 is:



The real IOCLA is passed to the channel/controller on the data bus.

INPUT/OUTPUT COMMAND DOUBLEWORD (IOCD)

The address indicated in the IOCLA specifies the word address of the first IOCD to be executed. The IOCD format is shown in Figure 5-12.

The SIO is the only instruction that is able to cause the Channel/Controller to fetch an IOCD. One or more IOCDs create an Input/Output Command List (IOCL).

The command field specifies one of the following seven commands:

Write
Read
Read Backward
Control
Sense
Transfer in Channel
Channel Control

If more than one IOCD is specified, the IOCDs are fetched sequentially except when Transfer in Channel (TIC) is specified. Search (compare) commands can cause the skipping of the next sequential IOCD if the condition becomes true (i.e., Search Equal, Search Low, or Search High). The channel or controller will then increment by 16 rather than 8.

The real data address specifies the starting address of the data area. The data address will be a byte address and the channel will internally align the information transferred to or from main memory. Exclusions to the byte alignment may be required by the lower priced channel(s) operating in Burst mode in high performance controllers.

The byte count specifies the number of bytes that are to be transferred to or from main memory. The actual number of memory transfers performed by the channel will be dependent upon the channel implementation.

INPUT/OUTPUT COMMANDS

WRITE The Write command causes a Write (output) operation to the selected I/O device from the specified main memory address.

READ The Read command causes a Read (input) operation from the selected I/O device to the specified main memory address.

READ BACKWARD The Read Backward command causes a Read (input) operation from the selected I/O device to the specified main memory address in descending order.

CONTROL The Control command causes control information to be passed to the selected device. A Control command may provide a data address and byte count for additional control information that may be stored in main memory.

Control information is device dependent and may instruct a magnetic tape to rewind or a printer to space a certain number of lines.

SENSE The Sense command causes the storing of controller/device information in the specified location of main memory. One or more bytes of information will be transferred depending upon the device. The sense information provides additional device dependent information not provided in the status flags.

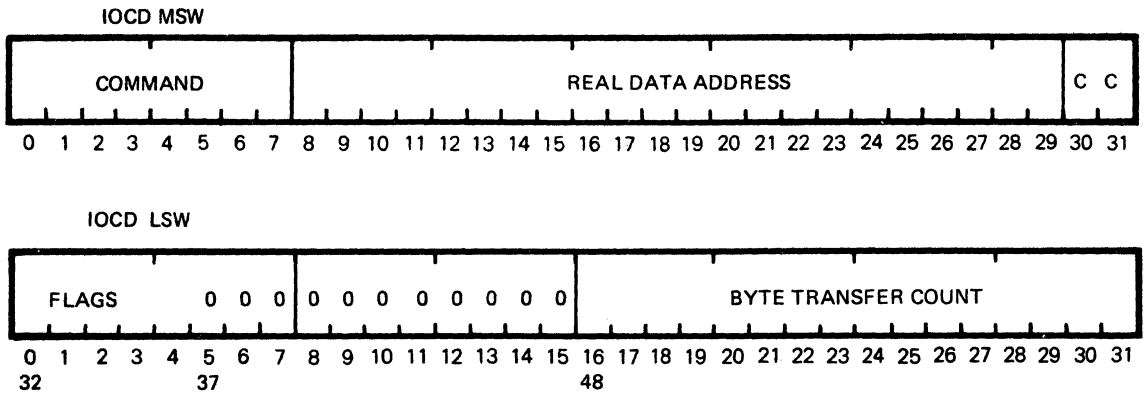
TRANSFER IN CHANNEL The Transfer in Channel (TIC) command specifies the address of the next IOCD to be executed. The TIC command allows the programmer to change the sequence of the IOCDs executed. The IOCLA cannot specify a TIC as the first IOCD in a command list nor can a TIC specify another TIC command.

CHANNEL CONTROL The Channel Control command causes the transfer of information to or from a specific location in main memory. One or more bytes of information will be transmitted or received from the channel. The channel control provides for the passing of information required to initialize all channels.

INPUT/OUTPUT TERMINATION

An I/O operation terminates when the channel, controller, and/or device indicates the end of an operation. All I/O operations accepted by the channel will always terminate with at least one termination status being presented to software.

An I/O operation can also fail to be accepted by the channel during I/O initiation. Conditions that prevent I/O initiation are: (1) channel or subchannel busy, (2) channel not operational or nonexistent, or (3) pending termination status from a previously initiated I/O operation.



BIT ASSIGNMENTS IN THE COMMAND ARE:

X X X X 0 0 0 0	CHANNEL CONTROL
M M M M 0 1 0 0	SENSE
X X X X 1 0 0 0	TRANSFER IN CHANNEL
M M M M 1 1 0 0	READ BACKWARD
M M M M M M 0 1	WRITE
M M M M M M 1 0	READ
M M M M M M 1 1	CONTROL

FLAG BIT ASSIGNMENTS ARE:

1 0 0 0 0 0	DATA CHAIN (HOLDS OFF TERMINATION WHEN XFER CT = 0)
0 1 0 0 0 0	CMD CHAIN
0 0 1 0 0 0	SUPPRESS INCORRECT LENGTH
0 0 0 1 0 0	SKIP
0 0 0 0 1 0	POST PROGRAM CONTROLLED INTERRUPT

C - BIT ASSIGNMENTS ARE:

BIT 30	BIT 31	
0	0	BYTE 0 OR FULLWORD
0	1	BYTE 1 OR FIRST HALFWORD
1	0	BYTE 2 OR DOUBLEWORD*
1	1	BYTE 3 OR SECOND HALFWORD

*IF DOUBLEWORD IS INDICATED TO A CHANNEL, AMBIGUOUS RESULTS MAY OCCUR.

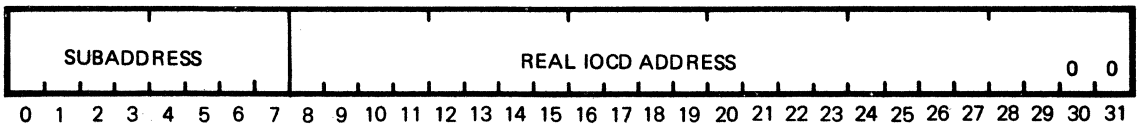
Figure 5-12. Input/Output Command Doubleword (IOCD)

I/O initiation failures are reported to software by the setting of condition codes and, where applicable, the storing of status.

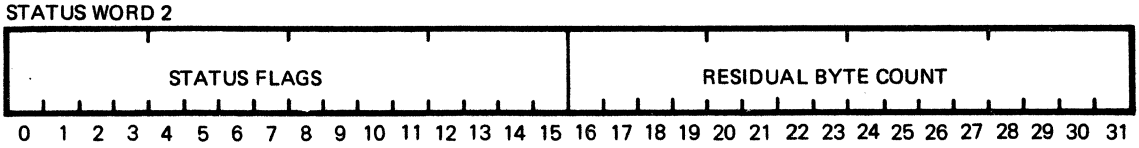
INPUT/OUTPUT STATUS WORDS

The status words are maintained and stored by the channel. The address of the status words is transmitted to the CPU when an interrupt is acknowledged or when another I/O instruction is executed. The status words contain information relating to the execution of the last IOCD or from an asynchronous condition requiring software notification (i.e., tape loaded, disc pack mounted). The status words are in the following format:

STATUS WORD 1



STATUS WORD 2



The status flags contain termination information pertaining to both the channel and controller. IOMs that function as integrated channel controllers will maintain both sections.

The address of the status is stored in main memory and can be located by adding $2010 = 14_{16}$ to the contents of the service interrupt vector.

INPUT/OUTPUT INTERRUPTS

Input/Output interrupts can be caused by a response to a probe instruction (i.e., TIO) by the termination of an I/O operation, by operator intervention at the I/O device, or when a post program controlled interrupt is requested by an IOCD. The associated I/O interrupt causes the status address, and the current PSD to be stored in the memory location specified by the service interrupt address. The new PSD (specified by the contents of the service interrupt vector +8) is then loaded.

An I/O interrupt can be caused by the device, controller, or channel. If a channel or controller has multiple I/O interrupt requests pending, it establishes a priority sequence for them before initiating an I/O interrupt request to the CPU. This priority sequence is maintained when the channel stores the status and reports the status address to the CPU.

The mode in which the channel operates during the software interrupt processing is determined by the mode setting of the channel and the implementation of the channel. The software may use bits 48 and 49 of the new PSD to select one of two options: Unblocked or Blocked operation.

Unblocked operation specifies that the CPU, upon receipt of an interrupt, causes the channel to go active and block all interrupts of a lower priority. The channel services the interrupt, and the software in turn issues a DACI or BRI command to restore the interrupt processing.

Blocking specifies that the CPU, upon receipt of an interrupt, causes the channel to deactivate. The CPU blocks all incoming interrupts and services the pending interrupt. The software in turn issues an UEI command or a BRI, LPSD, or LPSDCM to the CPU, thereby restoring interrupt processing. The target PSD of the BRI, LPSD, or LPSDCM instruction should specify Unblocked operation in bits 48 and 49.

SECTION VI

INSTRUCTION REPERTOIRE

INTRODUCTION This section contains the description of each computer instruction. The following paragraphs list the standard information given with each instruction.

MNEMONIC A two- to six-letter symbolic representation of the instruction name accepted by the assembler program.

INSTRUCTION NAME A title that indicates the function performed by the instruction.

OPERATION CODE The Operation Code for each instruction is given in left-justified hexadecimal format. This format is presented in a 16-bit skeleton form and takes into consideration the Augmenting Code and the format bit used with byte-oriented instructions.

FORMAT A 16- or 32-bit machine language representation of the instruction. The operation code and all other fixed bits are given in their binary value.

DEFINITION The function performed by the instruction is described following the instruction format. All registers or memory locations which are modified are defined. Special considerations are given in notes following the basic functional description.

SUMMARY EXPRESSION This expression supplements the verbal description of most instructions by symbolically showing the function performed by execution of the instruction. The symbols are defined in Table 6-1. The abbreviations are listed in Table 6-2.

Summary expression examples are given below:

$$(s_{24\text{-}31}) \rightarrow (d_{24\text{-}31})$$

The contents of bits 24-31 of GPR d are replaced with the contents of bits 24-31 of GPR s.

$$[\text{zeros}_{0\text{-}23}, \text{ byte operand}] \rightarrow (d)$$

The byte operand is appended with zeros in positions 0-23 and the resulting word replaces the contents of GPR d.

(m), (m+1) is a doubleword effective memory address.
(d), (d+1) is a doubleword even/odd GPR pair.

ASSEMBLY CODING CONVENTIONS A symbolic representation of the assembler coding format. Table 6-2 lists all abbreviations and symbols used in the operand coding format.

Table 6-1. Symbol Definitions

Symbol	Definition
—	Logical NOT function, for example ($\bar{s}$) is the ones complement of the GPR number s.
→	Replaces; the data to the left of the symbol replaces data to the right. For example, (s) → (d) means the contents of GPR number s replaces the contents of GPR number d.
+1	The register number or memory address is incremented by one register number or one memory word.
>	Greater Than.
<	Lesser Than.
+	Algebraic Addition.
-	Algebraic Subtraction.
x	(or no symbol) Algebraic Multiplication.
/	Algebraic Division.
&	Logical AND.
B_{m-n}	Bits m through n of a computer word.
B_n	Bit n of a computer word where B_0 always refers to the most significant bit of a computer word (the letter n is also used to indicate scaling; e.g., 1_{15} indicates a 1 scaled at bit position 15).
CC_n	Condition Code bit n.
:	Comparison Symbol.
.	Concatenation Sign (e.g., R, R+1 indicates a doubleword consisting of (R) and (R+1), where R must be an even numbered register).
EA	Effective Address of an operand or instruction stored in memory.
EBA	Effective Byte Address.
EBL	Eight-Bit Location in memory specified by the EBA.
EDA	Effective Doubleword Address.
EDL	Sixty-four bit location in memory consisting of an even numbered word location and the next higher word location, specified by the EDA.
EHA	Effective Halfword Address.
EHL	Sixteen-bit location in memory specified by the EHA.
EWA	Effective Word Address.

Table 6-1. Symbol Definitions (Cont'd)

Symbol	Definition
EWL	Thirty-two bit location in memory specified by the EWA.
I	Indirect Address bit.
ISI	Is Set If, used to indicate conditions which set referenced bit locations.
IW	Instruction Word.
( )	Contents of.
$\oplus$	Exclusive OR.
MIDL	Memory Image Descriptor List.
PSDR	Program Status Doubleword Registers.
PSWR	Program Status Word Register.
R	General Register 0-7 (R0-R7).
R_{m-n}	Bits m through n of General Register R.
R_n	Bit n of General Register R.
SBL	Specified Bit Location with a byte (used as a subscript to designate that the bit location is specified in the Instruction Word).
SCC	Sets Condition Code bits.
SE	Used as a subscript to denote a sign extended halfword.
v	Logical OR.
X	Index Register: X Value — GPR Used for Indexing 00 — None 01 — R1 10 — R2 11 — R3
-Y	Twos complement of Y.
Y	Ones completion of Y, logical NOT function.

CONDITION CODE RESULTS

An interpretation of the resulting 4-bit Condition Code in the Program Status Doubleword register. This code defines the result of the operation. The circumstances in which these Condition Codes are set (i.e., equal to One) are noted with each instruction.

EXAMPLES

Included in the examples with many of the instructions are memory and register contents before and after execution.

INSTRUCTION MNEMONICS

The 32/70 Series instruction mnemonics follow a very simple format. The basic types are:

L	load	or	LM	load masked
ST	store	or	STM	store masked
AD	add			
ADM	add memory to register			
ARM	add register to memory			
SU	subtract			
SUM	subtract memory from register			
MP	multiply			
DV	divide			
ADF, SUF, MPF, DVF	floating-point arithmetic			
B	branch			
AN	AND			
OR	logical OR			
EO	exclusive OR			
C	compare			

These basic mnemonics are then augmented to define the operand data type. (A special set of instructions are provided for bit manipulation.) The five basic data types are:

B	Byte	(8 bits)
H	Halfword	(16 bits)
W	Word	(32 bits)
D	Doubleword	(64 bits)
I	Immediate	(16 bits)

Therefore, the resulting instruction mnemonics have the form:

LB	Load Byte
LMH	Load Masked Halfword
STMW	Store Masked Word
ADI	Add Immediate to Register
SUMD	Subtract Memory Doubleword

A complete summary of the 32/70 Series instructions is presented in the Appendix of this manual.

ASSEMBLER CODING CONVENTIONS

The basic assembler coding format for memory reference instructions is:

```
XXXXXX    |s|    ,   *m,    x
          |d|
```

which translates to

XXXXXX	Instruction mnemonic
|s| |d|	Source or destination General Purpose Register
*	Indirectly (optional)
m	Memory operand
x	Indexed by register number x

Nonmemory reference instruction coding is similar to the memory reference format. Table 6-2 lists all codes used in defining the Assembler coding formats.

INSTRUCTION DEFINITION FORMAT

Each instruction definition includes the following information:

Instruction Name — The full name of the instruction.

Op Code — The four most significant hexadecimal digits of the instruction word are listed. Additional bits in the op code are set when the instruction is coded to address a General Purpose Register (GPR), for indirect addressing, or for byte addressing.

Assembler Coding Format — The coding format used by the 32 Macro Assembler. Table 6-2 includes all the abbreviations and symbols used in the operand coding format.

Instruction Definition — A definition of the operation performed by executing the instruction.

Summary Expression — A symbolic or graphic description of the operation performed by the instruction. Summary expressions use the same abbreviations used in the assembler coding format, Table 6-2. In addition, Table 6-1 lists the codes and symbols used in the summary expressions.

Condition Codes — The Condition Codes are set based on the results obtained by executing an instruction. The circumstances in which these condition codes are set (i.e., equal to one) are noted with each instruction.

Table 6-2. Assembler Coding Symbols

Code	Description
Capital Letters	Instruction Mnemonic
b	Bit number (0-31) in a General Purpose Register
c	Bit number (0-7) within a byte
d	Destination General Purpose Register number (0-7)
f	Function
m	Operand Memory Address
n	Device Address
s	Source General Purpose Register number (0-7)
v	Value for Immediate Operands, number of shifts, etc.
x	Index register number 1, 2, or 3. Optional
*	Indirect Addressing. Optional
,	Assembler Syntax
z	Special register field for instructions requiring three register fields

LOAD/STORE INSTRUCTIONS

GENERAL DESCRIPTION

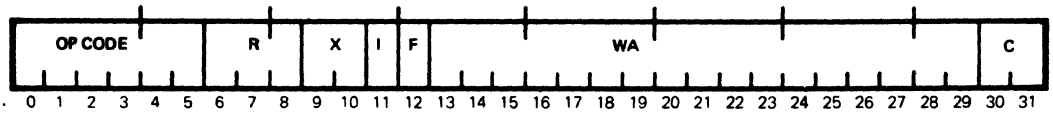
The Load/Store instruction group is used to manipulate data between memory and General Purpose Registers. In general, Load instructions transfer operands from specified memory locations to General Purpose Registers; Store instructions transfer data from General Purpose Registers to specified memory locations. Provisions have also been made to Mask or Clear the contents of General Purpose Registers, memory bytes, halfwords, words, or doublewords during instruction execution.

INSTRUCTION FORMATS

The Load/Store instructions use the following three formats:

MEMORY REFERENCE

The format for most memory reference instructions is defined below. These instructions contain two addresses: a register number R and a memory address with a 20-bit format.



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-10 designate one of three General Purpose Registers to be used as an index register.

- X = 00 designates that no indexing operation is to be performed.
- X = 01 designates the use of R1 for indexing.
- X = 10 designates the use of R2 for indexing.
- X = 11 designates the use of R3 for indexing.

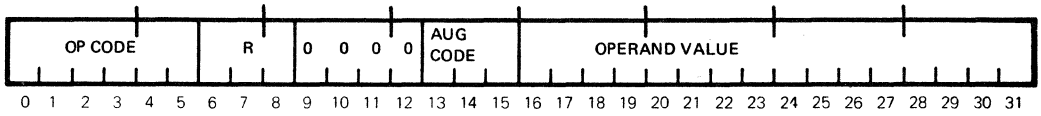
Bit 11 designates whether an indirect addressing operation is to be performed.

- I = 0 designates that no indirect addressing operation is to be performed.
- I = 1 designates that an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when the X and I fields are equal to zero.

IMMEDIATE

In immediate operand instructions, the right halfword of the instruction contains the 16-bit operand value. The format for these instructions is given below.



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-12 unassigned.

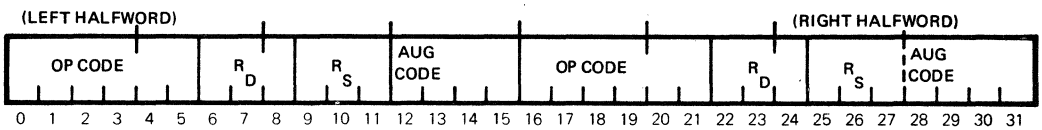
Bits 13-15 define Augmenting Operation Code.

Bits 16-31 contain the 16-bit operand value.

Arithmetic operands are assumed to be represented in two's complement with the sign in bit 16.

INTERREGISTER

Interregister instructions are halfword instructions and as such may be stored in either the left or right half of a memory word. The format for interregister instructions is given below.



Left Halfword	Right Halfword	
Bits 0-5	16-21	define the Operation Code.
Bits 6-8	22-24	designate the register to contain the result of the operation.
Bits 9-11	25-27	designate the register which contains the source operand.
Bits 12-15	28-31	define the Augmenting Operation Code.

CONDITION CODE UTILIZATION

A Condition Code is set during most Load instructions to indicate if the operand being transferred is greater than, less than, or equal to zero. Arithmetic exceptions are also reflected by the Condition Code results. All Store instructions leave the Condition Code unchanged.

MEMORY TO REGISTER TRANSFERS

Figure 6-1 depicts the positioning of information for transfer from memory to any General Purpose Register.

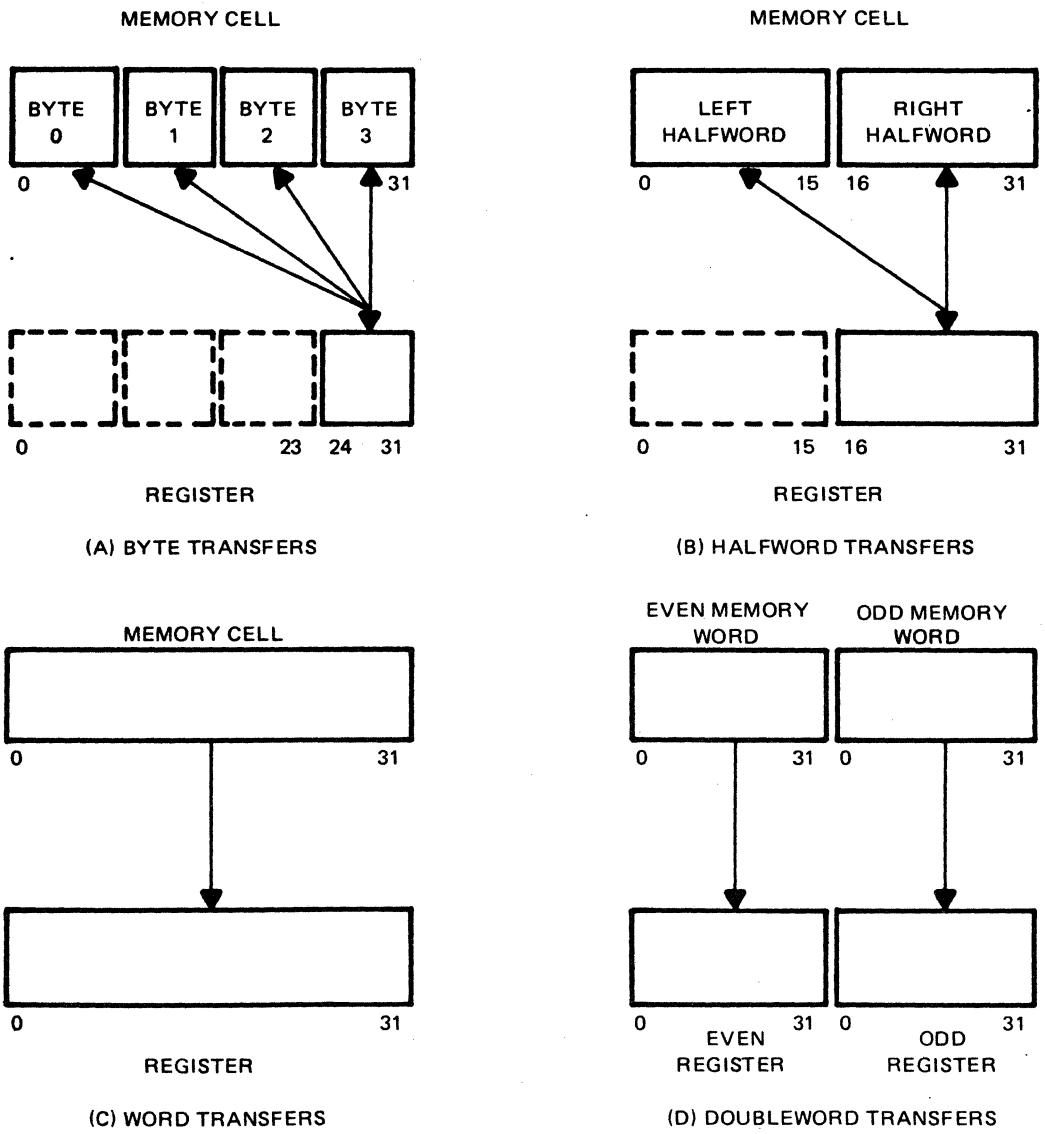
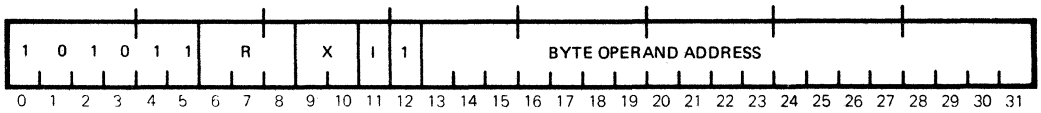


Figure 6-1. Positioning of Information Transferred Between Memory and Registers

LB
d,*m,x

LOAD BYTE

AC08



DEFINITION — The byte in memory specified by the Effective Byte Address (EBA) is accessed and transferred to bit positions 24-31 of the General Purpose Register (GPR) specified by R. Bit positions 0-23 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

$(EBL) \rightarrow R_{24-31}$

$0 \rightarrow R_{0-23}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: Always zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE 1

Memory Location: 01000
Hex Instruction: AC 88 11 01 (R=1, X=0, I=0)
Assembly Language Coding: LB 1,X'1101'

Before Execution	PSWR 00001000	GPR1 517CD092	Memory Byte 01101 B6
After Execution	PSWR 20001004	GPR1 000000B6	Memory Byte 01101 B6

Note — The contents of memory byte 01101 are transferred to bits 24-31 of GPR1, bits 0-23 of GPR1 are cleared. CC2 is set because the contents of GPR1 are greater than zero.

EXAMPLE 2

Memory Location: 01000
Hex Instruction: AD 28 14 00 (R=2, X=1, I=0)
Assembly Language Coding: LB 2,X'1400',1

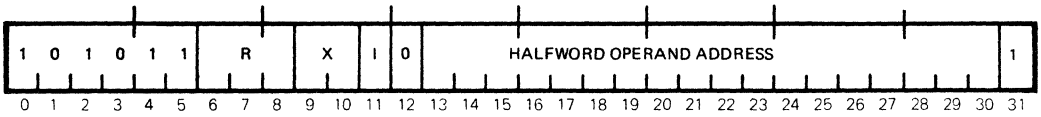
Before Execution	PSWR 10001000	GPR1 00000203	GPR2 12345678	Memory Byte 01603 A1
After Execution	PSWR 20001004	GPR1 00000203	GPR2 000000A1	Memory Byte 01603 A1

Note — The contents of memory byte 01603 are transferred to bits 24-31 of GPR2. Bits 0-23 are cleared, and CC2 is set.

LOAD HALFWORD

LH
d,*m,x

AC00



DEFINITION — The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign bit (bit 16) is extended left 16 bit positions to form a word. This word is transferred to the GPR specified by R.

SUMMARY EXPRESSION — (EHL)SE → R

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	00408
Hex Instruction:	AE 00 05 03 (R=4, X=0, I=0)
Assembly Language Coding:	LH 4,X'502'

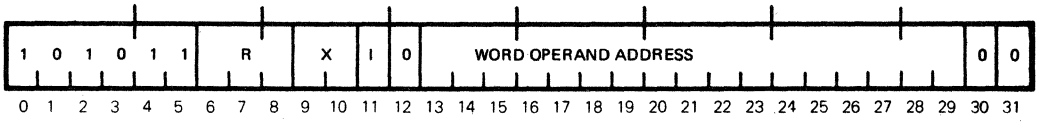
	PSWR	GPR4	Memory Halfword 00502
Before Execution	10000408	5C00D34A	930C
After Execution	1000040C	FFFF930C	930C

Note — The contents of memory halfword 00502 are transferred to bits 16-31 of GPR4. Bits 0-15 of GPR4 are set by the sign extension, and CC3 is set.

LW
d,*m,x

LOAD WORD

AC00



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed and transferred to the GPR specified by R.

SUMMARY EXPRESSION — (EWL) → R

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	02390
Hex Instruction:	AF 80 27 A4 (R=7, X=0, I=0)
Assembly Language Coding:	LW 7,X'27A4'

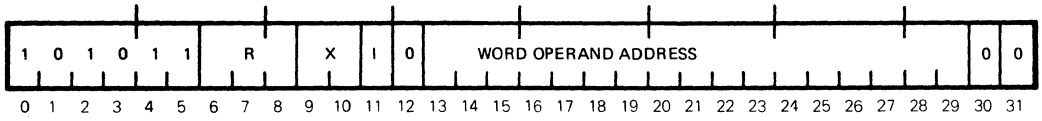
	PSWR	GPR7	Memory Word 027A4
Before Execution	00002390	0056879A	4D61A28C
After Execution	20002394	4D61A28C	4D61A28C

Note — The contents from memory word 027A4 are transferred to GPR7, and CC2 is set.

LOAD DOUBLEWORD

LD
d,*m,x

AC00



DEFINITION — The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and transferred to the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The least significant memory word is accessed first and transferred to the GPR specified by R+1. The most significant memory word is accessed last and transferred to the GPR specified by R.

NOTE — The GPR specified by R must have an even address.

SUMMARY EXPRESSION

(EWL+1) → R+1

(EWL) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location: 281C4
Hex Instruction: AF 02 8B 7A (R=6, X=0, I=0)
Assembly Language Coding: LD 6,X'28B78'

Before Execution	PSWR	GPR6	GPR7	Memory Word 28B78
	400281C4	03F609C3	39BB510E	F05B169A

Memory Word 28B7C
137F8CA2

After Execution	PSWR	GPR6	GPR7	Memory Word 28B78
	100281C8	F05B169A	137F8CA2	F05B169A

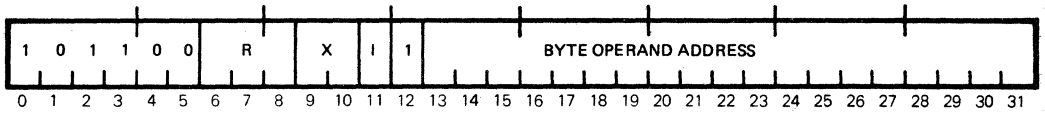
Memory Word 28B7C
137F8CA2

Note — The contents of memory word 28B78 are transferred to GPR6 and the contents of memory word 28B7C are transferred to GPR7. CC3 is set.

LMB
d,*m,x

LOAD MASKED BYTE

B008



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and masked (Logical AND function) with the least significant byte (bits 24-31) of the Mask register (R4). The result of the mask operation is transferred to bit positions 24-31 of the GPR specified by R. Bit positions 0-23 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

$(EBL)\&(R4_{24-31}) \rightarrow (R_{24-31})$

$0 \rightarrow R_{0-23}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: Always zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00900
Hex Instruction: B0 88 00 A3 (R=1, X=0, I=0)
Assembly Language Coding: LMB 1,X'A3'

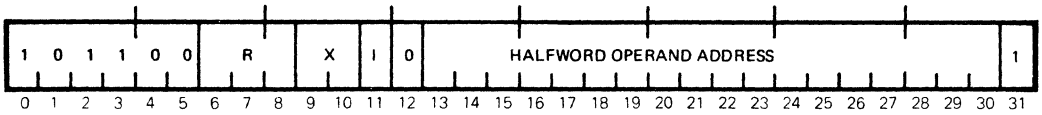
	PSWR	GPR1	GPR4	Memory Byte 000A3
Before Execution	00000900	AA3689B0	000000F0	29
After Execution	20000904	00000020	000000F0	29

Note

The contents of memory byte 000A3 are logically ANDed with the rightmost byte of GPR4, and the result is transferred to bits 24-31 of GPR1. Bits 0-23 of GPR1 are cleared, and CC2 is set.

LOAD MASKED HALFWORD

B000



DEFINITION The halfword in memory specified by the Effective Halfword Address (EHA) is accessed, and the sign bit (bit 16) is extended 16 bit positions to the left to form a word. This word is then masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION $(EHL)_{SE}\&(R4) \rightarrow R$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

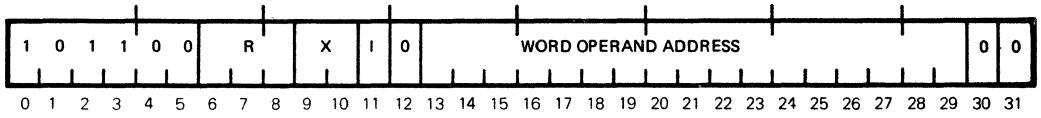
Memory Location:			00300	
Hex Instruction:			B2 80 03 A1	(R=5, X=0, I=0)
Assembly Language Coding:			LMH 5,X'3A0'	
Before Execution	PSWR	GPR4	GPR5	Memory Halfword 003A0
	08000300	0FF00FF0	C427B319	A58D
After Execution	PSWR	GPR4	GPR5	Memory Halfword 003A0
	20000304	0FF00FF0	0FF00580	A58D

Note The contents of memory halfword 003A0 are accessed, the sign is extended 16 bit positions, the result is logically ANDed with the contents of GPR4, and the final result is transferred to GPR5. CC2 is set.

LMW
d,*m,x

LOAD MASKED WORD

B000



DEFINITION The word in memory specified by the Effective Word Address (EWA) is accessed and masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION (EWL)&(R4) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00F00
Hex Instruction: B3 80 0F FC (R=7, X=0, I=0)
Assembly Language Coding: LMW 7,X'FFC'

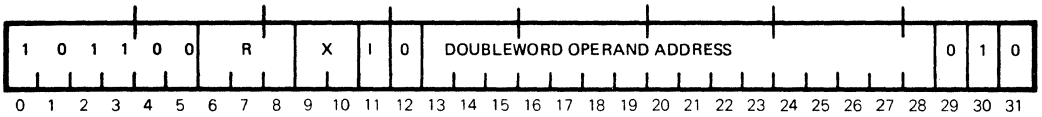
	PSWR	GPR4	GPR7	Memory Word 00FFC
Before Execution	00000F00	FF00007C	12345678	8923F8E8
After Execution	10000F04	FF00007C	89000068	8923F8E8

Note The contents of memory word 00FFC are ANDed with the contents of GPR4. The result is transferred to GPR7, and CC3 is set.

LOAD MASKED DOUBLEWORD

LMD
d,*m,x

B000



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed, and the contents of each word are masked (Logical AND Function) with the contents of the Mask register (R4). The least significant memory word is masked first. The resulting masked doubleword is transferred to the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R.

SUMMARY EXPRESSION

(EWL+1)&(R4) → R+1

(EWL)&(R4) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00200
Hex Instruction: B3 00 02 F2 (R=6, X=0, I=0)
Assembly Language Coding: LMD 6,X'2F0'

Before Execution

PSWR	GPR4	GPR6	GPR7
00000200	3F3F3F3F	12345678	9ABCDEF0

Memory Word 002F0	Memory Word 002F4
AE69D10C	63B208F0

After Execution

PSWR	GPR4	GPR6	GPR7
20000204	3F3F3F3F	2E29110C	23320830

Memory Word 002F0	Memory Word 002F4
AE69D10C	63B208F0

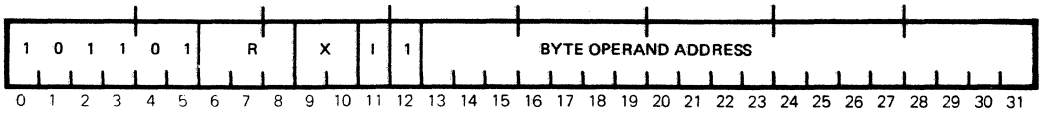
Note

The contents of memory word 002F4 are ANDed with the contents of GPR4, and the result is transferred to GPR6. CC2 is set.

LNB
d,*m,x

LOAD NEGATIVE BYTE

B408



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed, and 24 zeros are appended to the most significant end to form a word. The two's complement of this word is then taken and transferred to the GPR specified by R.

SUMMARY EXPRESSION

$-\left[00\text{-}23,\ (EBL)\right] \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

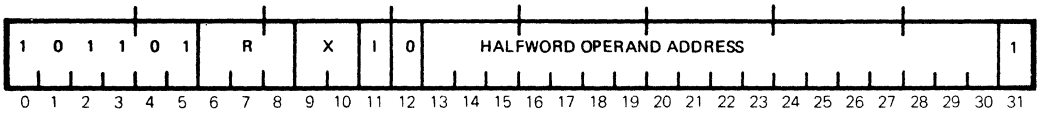
Memory Location:			0D000
Hex Instruction:			B4 88 D1 02 (R=1, X=1, I=0)
Assembly Language Coding:			LNB 1,X'D102'
Before Execution	PSWR	GPR1	Memory Byte 0D102
	0000D000	00000000	3A
After Execution	PSWR	GPR1	Memory Byte 0D102
	1000D004	FFFFFFC6	3A

Note

The contents of memory byte 0D102 are prefixed with 24 zeros to form a word; the result is negated and transferred to GPR1. CC3 is set.

LOAD NEGATIVE HALFWORD

B400



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed, and the sign bit (bit 16) is extended 16 bit positions to the left to form a word. The two's complement of this word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$$-\left[(EHL)_{SE}\right] \rightarrow R$$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:			08000
Hex Instruction:			B6 00 84 03 (R=4, X=0, I=0)
Assembly Language Coding:			LNH 4,X'8402'

Before Execution	PSWR	GPR4	Memory Halfword 08402
	40008000	12345678	960C
After Execution	PSWR	GPR4	Memory Halfword 08402
	20008004	000069F4	960C

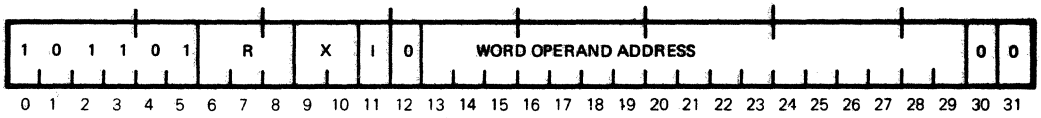
Note

The contents of memory halfword 08402 are sign extended and negated. The result is transferred to GPR4, and CC2 is set.

LNW
d,*m,x

LOAD NEGATIVE WORD

B400



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed, and its two's complement is transferred to the GPR specified by R.

SUMMARY EXPRESSION — $-(EWL) \rightarrow R$

CONDITION CODE RESULTS
CC1: ISI Arithmetic Exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	00500
Hex Instruction:	B6 80 06 C8 (R=5, X=0, I=0)
Assembly Language Coding:	LNW 5,X'6C8'

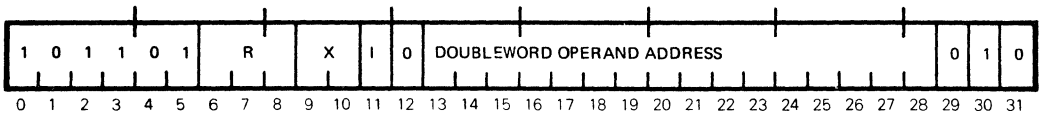
	PSWR	GPR5	Memory Word 006C8
Before Execution	08000500	00000000	185E0D76
After Execution	10000504	E7A1F28A	185E0D76

Note — The contents of memory word 006C8 are negated and transferred to GPR5, and CC3 is set.

LOAD NEGATIVE DOUBLEWORD

LND
d,*m,x

B400



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and its two's complement is formed. The least significant memory word is complemented first and the result is transferred to the GPR specified by R+1. R+1 is the GPR one greater than specified by R. The most significant memory word is complemented, and the result is transferred to the GPR specified by R1.

SUMMARY EXPRESSION

-(EDL)→R,R+1

CONDITION CODE RESULTS

CC1: ISI Arithmetic Exception
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location: 02344
Hex Instruction: B5 00 24 A2 (R=2, X=0, I=0)
Assembly Language Coding: LND 2,X'24A0'

Before Execution

PSWR	GPR2	GPR3
00002344	01234567	89ABCDEF

Memory Word 024A0	Memory Word 024A4
00000000	00000001

After Execution

PSWR	GPR2	GPR3
10002348	FFFFFFFF	FFFFFFFF

Memory Word 024A0	Memory Word 024A4
00000000	00000001

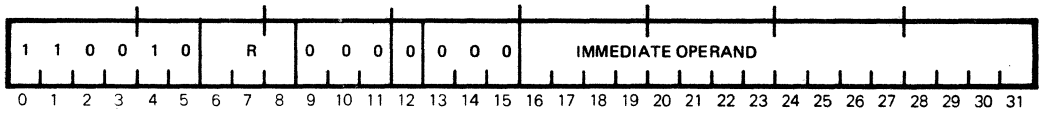
Note

The doubleword obtained from the contents of memory words 024A0 and 024A4 is negated, and the result is transferred to GPR2 and GPR3. CC3 is set.

LI
d,v

LOAD IMMEDIATE

C800



DEFINITION — The halfword immediate operand in the Instruction Word (IW) is sign-extended (bit 16 extended 16 positions to the left) to form a word. This word is transferred to the GPR specified by R.

SUMMARY EXPRESSION — $(IW_{16-31})_{SE} \rightarrow R$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI (R_{0-31}) is greater than zero
CC3: ISI (R_{0-31}) is less than zero
CC4: ISI (R_{0-31}) is equal to zero

EXAMPLE

Memory Location:	0630C
Hex Instruction:	C8 80 FF FB (R=1)
Assembly Language Coding:	LI 1,-5

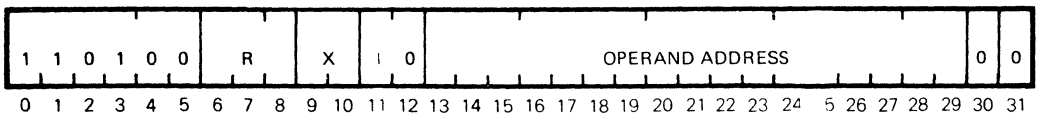
	PSWR	GPR1
Before Execution	0000630C	12345678
After Execution	PSWR	GPR1
	10006310	FFFFFFFB

Note — The halfword operand is sign-extended and the result is transferred to GPR1. CC3 is set.

LOAD EFFECTIVE ADDRESS

LEA
d,*m,x

D000



DEFINITION

The effective address (bit 12-31) of the LEA instruction is generated in the same manner as in all other memory reference instructions and then is transferred to bit positions 12-31 of the GPR specified by R.

In PSD mode or PSW mode extended, bits 2-7 are cleared and bits 8-31 indicate results of EA.

Notes

1. If I=X=0, the entire 32-bit Instruction Word is transferred to the GPR specified by R. (512 KB mode only)

2. If I=0 and X=0, bit positions 0-11 of the GPR specified by R will contain the sum of bit positions 0-11 of the Instruction Word and bit positions 0-11 of the index register specified by X. (512 KB mode only)

3. If I=1, bit positions 0-11 of the GPR specified by R will contain the sum of bit positions 0-11 of the last word of the indirect chain and bit positions 0-11 of the index register specified (if any) in the last word of the indirect chain. (512 KB mode only)

4. In cases 2 and 3 above, an additional bit may be added to bit position 11 of the GPR specified by R as a result of overflow in the sum of the address and the index values. (512 KB mode only)

SUMMARY EXPRESSION

$EA \rightarrow R_{12-31}$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

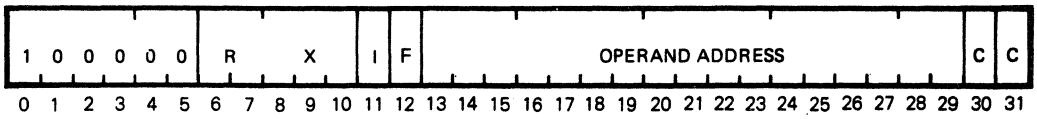
EXAMPLE

Memory Location:			1000
Hex Instruction:			D0 804000 (R=1, X=I=0)
Assembly Language Codings:			LEA 1,X'4000'
Before Execution	PSWR	GPR1	Memory Word 4000
	08001000	00000000	AC881203
After Execution (PSD Mode)	PSWR	GPR1	Memory Word 4000
	08001004	D0804000	AC881203
	08001004	C0004000	AC881203

LEAR
d,*m,x

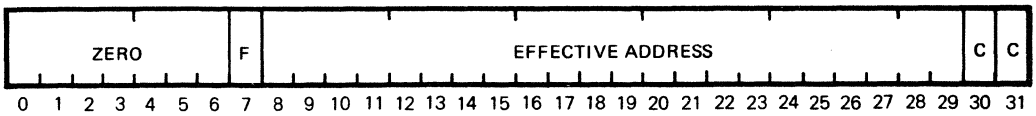
LOAD EFFECTIVE ADDRESS REAL

8000



DEFINITION — This instruction causes the Effective Real (nonmapped) Address of the referenced operand to be transferred to bit positions 7-31 of the GPR specified by R.

NOTE — The format of the 25-bit Effective Real Address transferred to the GPR is as follows:



SUMMARY EXPRESSION

$ERA \rightarrow R_{7-31}$

$0 \rightarrow R_{0-6}$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: LEAR d,*m,x

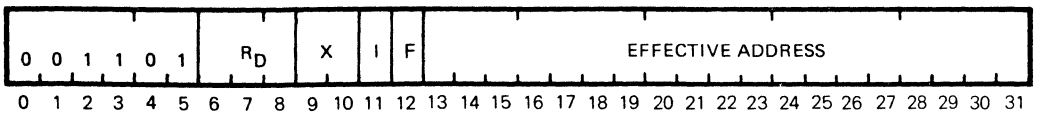
NOTES

1. Privileged Instruction
2. Attempt to execute in PSW mode will result in an undefined instruction trap.
3. This instruction may not be the target of an execute instruction.

LOAD ADDRESS

LA
d,*m,x

3400



DEFINITION

Loads the Effective Address (EA) into R_D. Bits 0-7 are cleared in R_D. Bits 8-11 receive the results of Extended Indexing (if active). Bit 12 is the F-bit if 512 KB mode and is an Effective Address (EA) bit if in 512 KB Extended mode.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: LA d,*m,x

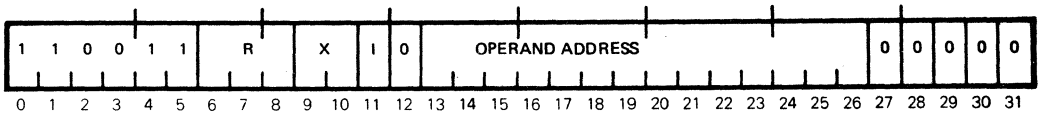
DELETED

DELETED

LF
d,*m,x

LOAD FILE

CC00



DEFINITION

This instruction is used to load from one to eight GPR's. The word in memory specified by the Effective Word Address (EWA) in the Instruction Word is accessed and transferred to the GPR specified by R. Next, the EWA and the GPR address are incremented. The next sequential memory word is then transferred to the next sequential GPR. Successive transfers continue until GPR7 is loaded from memory.

NOTE

The EWA must be specified such that, when incremented, no carry will be propagated from bit position 27. Therefore, if all eight registers are to be loaded, bit positions 27-29 must initially be equal to zero.

SUMMARY EXPRESSION

(EWL) → R

(EWL)+1 → R+1

. .

. .

. .

(EWL+N) → R7

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00300
Hex Instruction: CE 00 02 00 (R=4, X=0, I=0)
Assembly Language Coding: LF 4,X'200'

Before Execution

PSWR	GPR4	GPR5	GPR6	GPR7
08000300	00000000	00000000	00000000	00000000

Memory Word 00200 00000001

Memory Word 00204 00000002

Memory Word 00208 00000003

Memory Word 0020C 00000004

After Execution

PSWR	GPR4	GPR5	GPR6	GPR7
08000304	00000001	00000002	00000003	00000004

Memory Word 00200 00000001

Memory Word 00204 00000002

Memory Word 00208 00000003

Memory Word 0020C 00000004

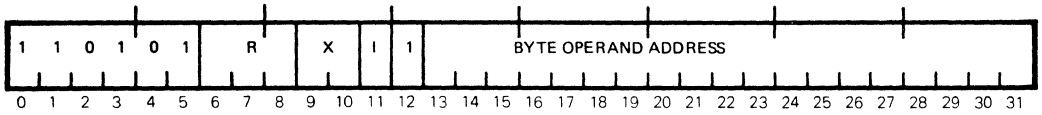
Note

The contents of memory word 00200 are transferred to GPR4, of memory word 00204 to GPR5, of memory word 00208 to GPR6, and of memory word 0020C to GPR7.

STORE BYTE

STB
s,*m,x

D408



DEFINITION

The least significant byte (bits 24-31) of the GPR specified by R is transferred to the memory byte location specified by the Effective Byte Address (EBA) in the Instruction Word. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31}) \rightarrow EBL$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:			03708
Hex Instruction:			D4 88 3A 13 (R=1, X=0, I=0)
Assembly Language Coding:			STB 1,X'3A13'
Before Execution	PSWR	GPR1	Memory Byte 03A13
	10003708	01020304	78
After Execution	PSWR	GPR1	Memory Byte 03A13
	1000370C	01020304	04

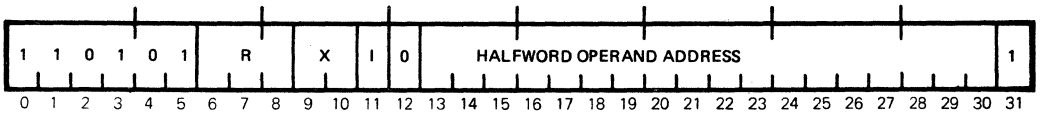
Note

The contents of bits 24-31 of GPR1 are transferred to memory byte 03A13.

STH
s,*m,x

STORE HALFWORD

D400



DEFINITION — The least significant halfword (bit 16-31) of the GPR specified by R is transferred to the memory halfword location specified by the Effective Halfword Address (EHA) in the Instruction Word. The other halfword of the memory word containing the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION — $(R_{16-31}) \rightarrow EHL$

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

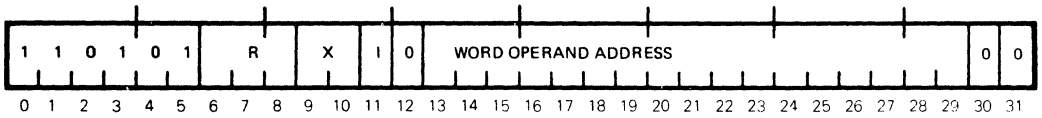
Memory Location:			082A400
Hex Instruction:			D6 00 83 13 (R=4, X=0, I=0)
Assembly Language Coding:			STH 4,X'8312'
Before Execution	PSWR	GPR4	Memory Halfword 08312
	000082A4	01020304	A49C
After Execution	PSWR	GPR4	Memory Halfword 08312
	000082A8	01020304	0304

Note — The contents of the right halfword of GPR4 are transferred to memory halfword 08312.

STORE WORD

STW
s,*m,x

D400



DEFINITION — The word in the GPR specified by R is transferred to the memory word location specified by the Effective Word Address in the Instruction Word.

SUMMARY EXPRESSION — (R) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 03904
Hex Instruction: D7 00 3B 3C (R=6, X=0, I=0)
Assembly Language Coding: STW 6,X'3B3C'

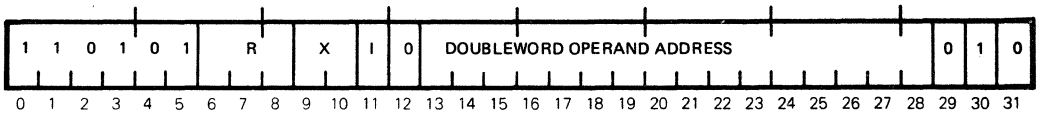
	PSWR	GPR6	Memory Word 03B3C
Before Execution	10003904	0485A276	00000000
After Execution	10003908	0485A276	0485A276

Note — The contents of GPR6 are transferred to memory word 03B3C.

STD
s,*m,x

STORE DOUBLEWORD

D400



DEFINITION

The doubleword in the GPR specified by R and R+1 (R+1 is the GPR one greater than specified by R) is transferred to the memory doubleword location specified by the Effective Doubleword Address (EDA). The word in the GPR specified by R+1 is transferred to the least significant word of the doubleword memory location first.

SUMMARY EXPRESSION

(R+1) → EWL+1

(R) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0596C
Hex Instruction: D7 00 5C 4A (R=6, X=0, I=0)
Assembly Language Coding: STD 6,X'5C48'

Before Execution

PSWR	GPR6	GPR7
2000596C	E24675C2	5923F8E8

Memory Word 05C48	Memory Word 05C4C
0A400729	8104A253

After Execution

PSWR	GPR6	GPR7
20005970	E24675C2	5923F8E8

Memory Word 05C48	Memory Word 05C4C
E24675C2	5923F8E8

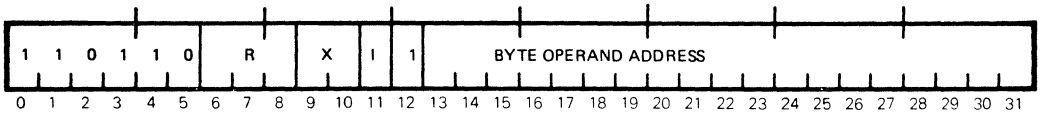
Note

The contents of GPR6 are transferred to memory word 05C48, and the contents from GPR7 are transferred to memory word 05C4C.

STORE MASKED BYTE

STMB
s,*m,x

D808



DEFINITION

The least significant byte (bits 24-31) of the GPR specified by R is masked (Logical AND Function) with the least significant byte of the Mask register (R4). The resulting byte is transferred to the memory byte location specified by the Effective Byte Address (EBA) in the Instruction Word. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31})\&(R4_{24-31}) \rightarrow EBL$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 01D80
Hex Instruction: D8 08 1E 91 (R=0, X=0, I=0)
Assembly Language Coding: STMB 0,X'1E91'

	PSWR	GPR0	GPR4	Memory Byte 01E91
Before Execution	10001D80	AC089417	0000FFFC	94
After Execution	10001D84	AC089417	0000FFFC	14

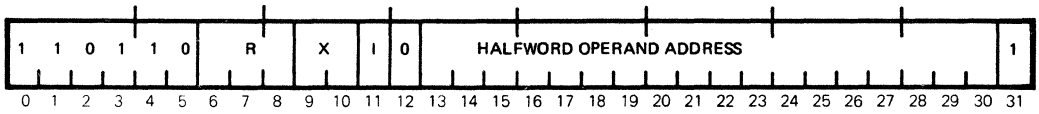
Note

The right-hand byte of GPR0 is ANDed with the right-hand byte of GPR4. The result is transferred to memory byte 01E91.

STMH
s,*m,x

STORE MASKED HALFWORD

D800



DEFINITION

The least significant halfword (bits 16-31) of the GPR specified by R is masked (Logical AND Function) with the least significant halfword of the Mask register (R4). The resulting halfword is transferred to the memory halfword location specified by the Effective Halfword Address (EHA) in the Instruction Word. The other halfword of the memory word containing the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31})\&(R4_{16-31}) \rightarrow EHL$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 01000
Hex Instruction: DA 80 11 AF (R=5, X=0, I=0)
Assembly Language Coding: STMH 5,X'11AE'

Before Execution	PSWR	GPR4	GPR5	Memory Halfword 011AD
	20001000	00003FFC	716A58AB	0000
After Execution	PSWR	GPR4	GPR5	Memory Halfword 011AD
	20001004	00003FFC	716A58AB	18A8

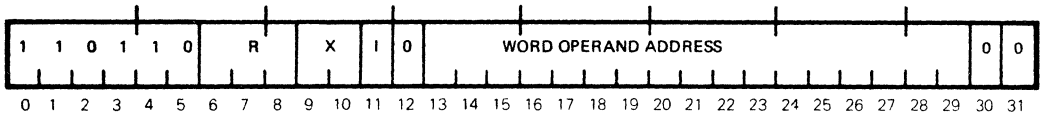
Note

The right-hand halfword of GPR5 is ANDed with the right-hand halfword of GPR4, and the result is transferred to memory halfword 011AD.

STORE MASKED WORD

STMW
s,*m,x

D800



DEFINITION — The word in the GPR specified by R is masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is transferred to the memory word location specified by the Effective Word Address.

SUMMARY EXPRESSION — (R)&(R4) → EWL

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

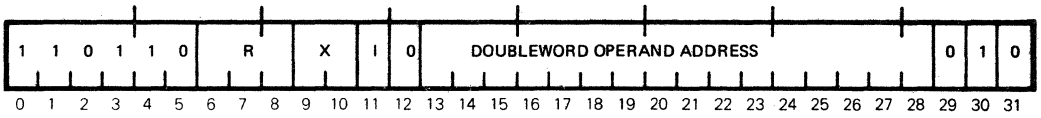
Memory Location:			04000	
Hex Instruction:			DB 00 43 7C (R=6, X=0, I=0)	
Assembly Language Coding:			STM W 6,X'4376'	
Before Execution	PSWR	GPR4	GPR6	Memory Word 0437C
	08004000	00FF00FF	718C3594	12345678
After Execution	PSWR	GPR4	GPR6	Memory Word 0437C
	08004004	00FF00FF	718C3594	008C0094

Note — The contents of GPR6 are ANDed with the contents of GPR4. The result is transferred to memory word 0437C.

STMD
s,*m,x

STORE MASKED DOUBLEWORD

D800



DEFINITION

Each word of the doubleword in the GPR specified by R and R+1 is masked (Logical AND Function) with the contents of the Mask register (R4). R+1 is GPR one greater than specified by R. The resulting doubleword is transferred to the memory doubleword location specified by the Effective Doubleword Address (EDA) in the Instruction Word.

SUMMARY EXPRESSION

(R+1)&(R4) → EWL+1

(R)&(R4) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0A498
Hex Instruction: DB 00 A6 52 (R=6, X=0, I=0)
Assembly Language Coding: STMD 6,X'A650'

Before Execution

PSWR	GPR4	GPR6	GPR7
1000A498	0007FFFC	AC88A819	988B1407

Memory Word 0A650	Memory Word 0A654
51CD092	AE69D10C

After Execution

PSWR	GPR4	GPR6	GPR7
1000A49C	0007FFFC	AC88A819	988B1407

Memory Word 0A650	Memory Word 0A654
0000A818	00031404

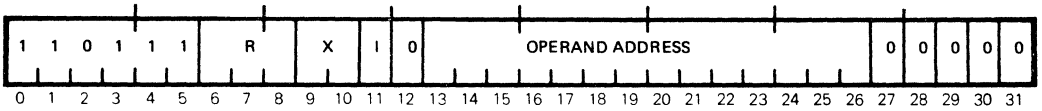
Note

The contents of GPR6 are ANDed with the contents of GPR4, and the result is transferred to memory word 0A650. The contents of GPR7 are ANDed with the contents of GPR4, and the result transferred to memory word 0A654.

STORE FILE

STF
s,*m,x

DC00



DEFINITION

This instruction is used to transfer the contents from one to eight GPR's to the specified memory locations. The contents of the GPR specified by R are transferred to the memory location specified by the Effective Word Address (EWA). The next sequential GPR is then transferred to the next sequential memory location. Successive transfers continue until GPR7 is loaded into memory.

NOTE

The EWA must be specified such that, when incremented, no carry will be propagated from bit position 27. Therefore, if all eight General Purpose Registers are transferred, bit positions 27-29 must initially be equal to zero.

SUMMARY EXPRESSION

(R) → EWL

(R+1) → EWL+1

.
.
.

(R7) → EWL+N

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 02000
Hex Instruction: DE 00 21 00 (R=4, X=0, I=0)
Assembly Language Coding: STF 4,X'2100'

Before Execution

PSWR	GPR4	GPR5	GPR6	GPR7
40002000	11111111	22222222	33333333	44444444

Memory Word 02100	Memory Word 02104
00210000	00210400
Memory Word 02108	**Memory Word 0210C**
00210800	00210C00

After Execution

PSWR	GPR4	GPR5	GPR6	GPR7
40002004	11111111	22222222	33333333	44444444

Memory Word 02100	Memory Word 02104
11111111	22222222
Memory Word 02108	**Memory Word 0210C**
33333333	44444444

Note

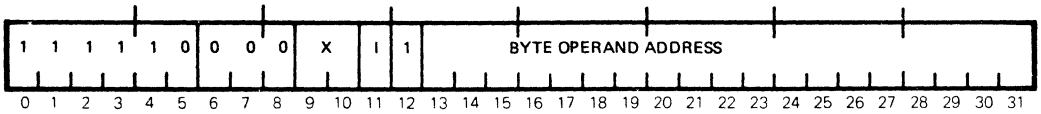
The contents of GPR4 are transferred to memory word 02100, of GPR5 to 02104, of GPR6 to 02108, and of GPR7 to 0210C.

DELETED

ZERO MEMORY BYTE

ZMB
*m,x

F808



DEFINITION — The byte in memory specified by the Effective Byte Address (EBA) is cleared to zero. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION — 0 → EBL

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:		00308
Hex Instruction:		F8 08 04 9F
Assembly Language Coding:		ZMB X'49F'

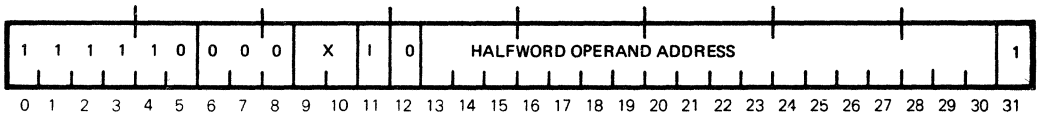
Before Execution	PSWR	Memory Byte 0049F
	10000308	6C
After Execution	PSWR	Memory Byte 0049F
	1000030C	00

Note — The contents of memory byte 0049F are cleared to zero.

ZMH
*m,x

ZERO MEMORY HALFWORD

F800



DEFINITION — The halfword in memory specified by the Effective Halfword Address (EHA) is cleared to zero. The remaining halfword containing the 16-bit location in memory specified by EHA remains unchanged.

SUMMARY EXPRESSION — 0 → EHL

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:		2895C
Hex Instruction:		F8 00 2A 42 7 (X=0, I=0)
Assembly Language Coding:		ZMH X'2A426'

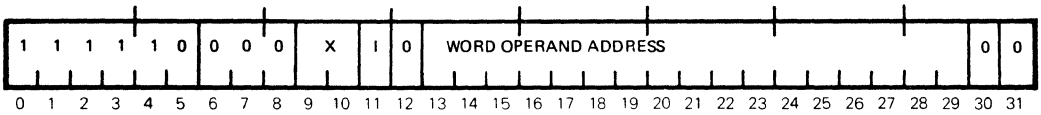
Before Execution	PSWR	Memory Halfword 2A426
	0802895C	9AE3
After Execution	PSWR	Memory Halfword 2A426
	08028960	0000

Note — The contents of memory halfword 2A426 are cleared to zero.

ZERO MEMORY WORD

ZMW
*m,x

F800



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is cleared to zero.

SUMMARY EXPRESSION — 0 → EWL

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE
Memory Location: 05A14
Hex Instruction: F8 00 5F 90 (X=0, I=0)
Assembly Language Coding: ZMW X'5F90'

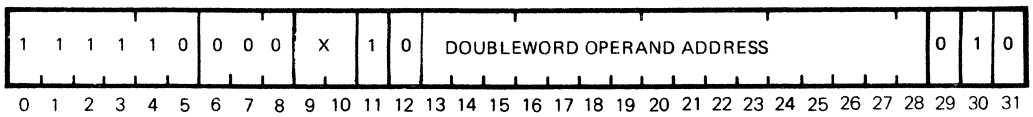
	PSWR	Memory Word 05F90
Before Execution	00005A14	12345678
After Execution	00005A18	00000000

Note — The contents of memory word 05F90 are cleared to zero.

ZMD
*m,x

ZERO MEMORY DOUBLEWORD

F800



DEFINITION — The doubleword in memory specified by the Effective Doubleword Address (EDA) is cleared to zero.

SUMMARY EXPRESSION

0 → EWL

0 → EWL+1

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 15B3C
Hex Instruction: F8 01 5D 6A (X=0, I=0)
Assembly Language Coding: ZMD X'15D68'

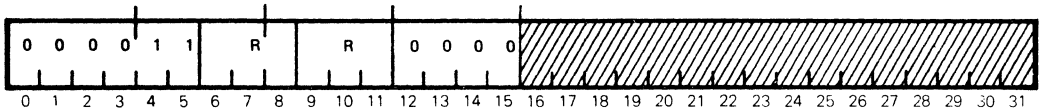
	PSWR	Memory Word 15D68	Memory Word 15D6C
Before Execution	10015B3C	617E853C	A2976283
After Execution	10015B40	00000000	00000000

Note — The contents of memory words 15D68 and 15D6C are cleared to zero.

ZERO REGISTER

ZR
d

0C00



DEFINITION

The word in the GPR specified by R (bits 6-8) is logically Exclusive ORed with the word in the GPR specified by R (bits 9-11) resulting in zero. This result is then transferred to the GPR specified by R. The contents of the two R fields must specify the same GPR.

SUMMARY EXPRESSION

(R)⊕(R) R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Always one

EXAMPLE

Memory Location:	309A6
Hex Instruction:	0C 90 (R=1)
Assembly Language Coding:	ZR 1

	PSWR	GPR1
Before Execution	100309A6	8495A6B7
After Execution	080309A8	00000000

Note

The contents of GPR1 are cleared to zero, and CC4 is set.

REGISTER TRANSFER INSTRUCTIONS

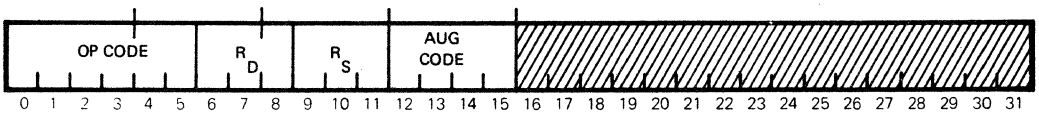
GENERAL DESCRIPTION

The Register Transfer instruction group provides the capability to perform a transfer or exchange of information between registers. Provisions have also been made in some instructions to allow two's complement, one's complement, and Mask operations to be performed during execution.

INSTRUCTION FORMATS

The following basic instruction format is used by the Register Transfer instruction group.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

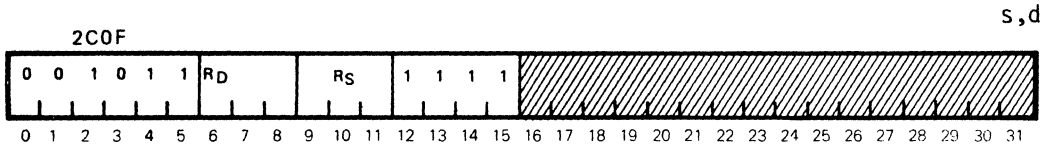
Bits 12-15 define the Augmenting Operation Code.

CONDITION CODE UTILIZATION

A Condition Code is set during execution of most Register Transfer instructions to indicate whether the contents of the Destination register (R_D) are greater than, less than, or equal to zero.

TRANSFER SCRATCHPAD TO REGISTER

TSCR
s,d



DEFINITION

The word in the Scratchpad specified by R_S, bits 8-15, is transferred to the GPR specified by R_D. The contents of R_S is not modified and only bits 8-15 are used by the instruction.

SUMMARY EXPRESSION

Scratchpad addressed by $R_S \rightarrow R_D$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: TSCR R_S,R_D

NOTES

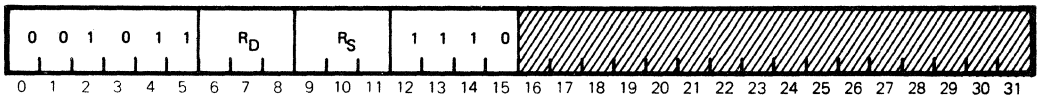
1. TSCR is a halfword privileged instruction.

2. The valid address range for R_S to address the 256 Scratchpad locations is $XX00XXXX_H$ to $XXFFXXXX_H$.

TRSC
s,d

TRANSFER REGISTER TO SCRATCHPAD

2C0E



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is transferred to the Scratchpad location specified by R_D bits 8-15. The contents of R_D is not modified by the instruction and only bits 8-15 are used by the instruction.

SUMMARY EXPRESSION

$(R_S) \rightarrow$ Scratchpad addressed by $R_{D\ 8\text{-}15}$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change
Assembly Language Coding: TRS R_S,R_D

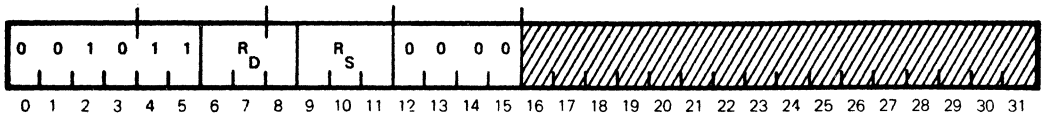
NOTES

1. TRSC is a halfword privileged instruction.
2. The valid address range for R_D to address the 256 Scratchpad locations is $XX00XXXX_H$ to $XXFFXXXX_H$.

TRANSFER REGISTER TO REGISTER

TRR
s,d

2C00



DEFINITION — The word in the GPR specified by R_S is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_S) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

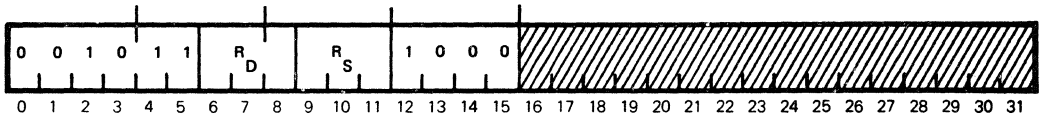
Memory Location			00206
Hex Instruction			2C A0 (R_D=1, R_S=2)
Assembly Language Coding:			TRR 2,1
Before Execution	PSWR	GPR1	GPR2
	00000206	00000000	000803AB
After Execution	PSWR	GPR1	GPR2
	20000208	000803AB	000803AB

Note — The contents of GPR2 are transferred to GPR1 and CC2 is set.

TRRM
s,d

TRANSFER REGISTER TO REGISTER MASKED

2C08



DEFINITION — The word in the GPR specified by R_S is masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	00206
Hex Instruction:	2C A8 (R_D=1, R_S=2)
Assembly Language Coding:	TRRM 2,1

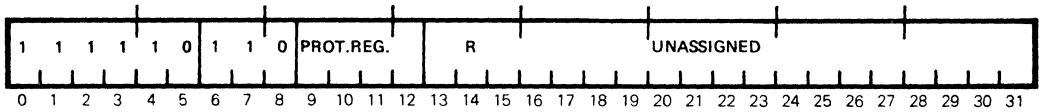
	PSWR	GPR1	GPR2	GPR4
Before Execution	00000206	00000000	000803AB	0007FFFD
After Execution	20000208	000003A9	000803AB	0007FFFD

Note — The contents of GPR2 are ANDed with the contents of GPR4, and the result is transferred to GPR1. CC2 is set.

TRANSFER REGISTER TO PROTECT REGISTER

TRP
s,p

FB00



DEFINITION

The word in the GPR specified by R is transferred to the Protect register specified by the Protect register field (bits 9-12) in the Instruction Word. The Protect register address is the same as the four high order memory address bits used to specify all memory locations within a given module.

SUMMARY EXPRESSION

(R) → PR

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:			0050C
Hex Instruction:			FB0F (R=7, Protect Register=1)
Assembly Language Coding:			TRP 7,1
Before Execution	PSWR	GPR7	Protect Register 1
	800005C0	0000FFFE	0000
After Execution	PSWR	GPR7	Protect Register 1
	80000510	0000FFFE	FFFE

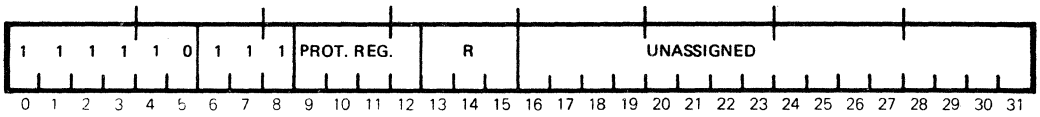
Note

The contents of bits 16-31 of GPR7 are transferred to Protect Register 1. The protection status of Memory Module 1 is established such that a program operating in the unprivileged state can store information only in locations 8000 through 87FF without generating a Privilege Violation trap.

TPR
d,p

TRANSFER PROTECT REGISTER TO REGISTER

FB80



DEFINITION

The word in the Protect register specified by the Protect register field (bits 9-12) is transferred to the GPR specified by R. The Protect register address is the same as the four high order memory address bits used to specify all memory locations within a given module.

SUMMARY EXPRESSION

(PR) → R

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:			0050C
Hex Instruction:			FB8F (R=7, Protect Register=1)
Assembly Language Coding:			TPR 1,7
Before Execution	PSWR	GPR7	Protect Register 1
	0000050C	00000000	FFFE
After Execution	PSWR	GPR7	Protect Register 1
	00000510	0000FFFE	FFFE

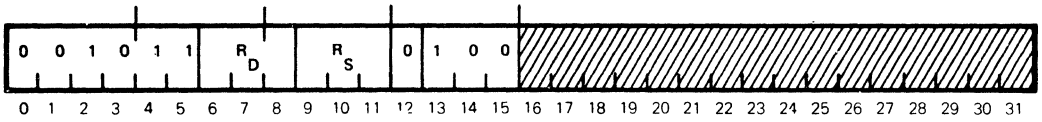
Note

The contents of Protect Register 1 are transferred to bits 16-31 of GPR7. This value defines the protection status of Memory Module 1.

TRANSFER REGISTER NEGATIVE

TRN
s,d

2C04



DEFINITION — The word in the GPR specified by R_S is two's complemented and transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $-(R_S) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: ISI Arithmetic exception
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	00AAE
Hex Instruction:	2F E4 (R_D=7, R_S=6)
Assembly Language Coding:	TRN 6,7

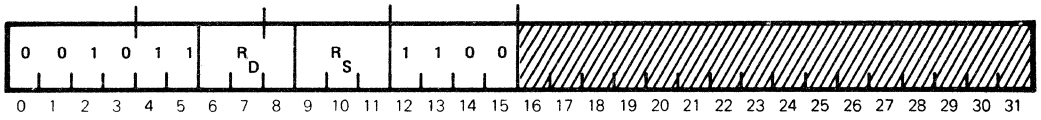
	PSWR	GPR6	GPR7
Before Execution	00000AAE	00000FFF	12345678
After Execution	10000AB0	00000FFF	FFFFF001

Note — The contents of GPR6 are negated and transferred to GPR7. CC3 is set.

TRNM
s,d

TRANSFER REGISTER NEGATIVE MASKED

2C0C



DEFINITION — The word in the GPR specified by R_S is two's complemented and masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $-(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: ISI Arithmetic exception
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	00AAE
Hex Instruction:	2F EC (R_D=7, R_S=6)
Assembly Language Coding:	TRNM 6,7

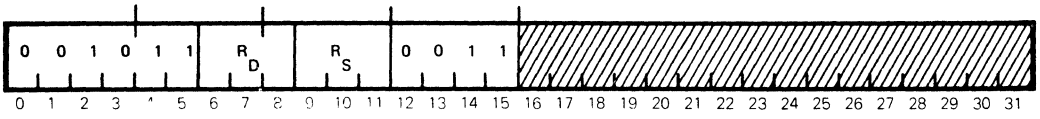
	PSWR	GPR4	GPR6	GPR7
Before Execution	00000AAE	7FFFFFFF	00000FFF	12345678
After Execution	20000AB0	7FFFFFFF	00000FFF	7FFFF001

Note — The contents of GPR6 are negated; the result is ANDed with the content of GPR4 and transferred to GPR7. CC2 is set.

TRANSFER REGISTER COMPLEMENT

TRC
s,d

2C03



DEFINITION

The word in the GPR specified by R_S is one's complemented and transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(\overline{R_S}) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	01001
Hex Instruction:	2F E3 (R_D=7, R_S=6)
Assembly Language Coding:	TRC 6,7

	PSWR	GPR6	GPR7
Before Execution	0800100A	55555555	00000000
After Execution	1000100C	55555555	AAAAAAAA

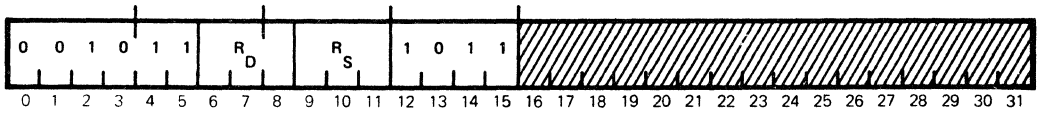
Note

The contents of GPR6 are complemented and transferred to GPR7. CC3 is set.

TRCM
s,d

TRANSFER REGISTER COMPLEMENT MASKED

2C0B



DEFINITION

The word in the GPR specified by R_S is one's complemented and masked (Logical AND Function) with the contents of the Mask register (R4). The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(\overline{R_S})\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location: 0100A
Hex Instruction: 2F EB (R_D=7, R_S=6)
Assembly Language Coding: TRCM 6,7

	PSWR	GPR4	GPR6	GPR7
Before Execution	0800100A	00FFFF00	55555555	00000000
After Execution	2000100C	00FFFF00	55555555	00AAAA00

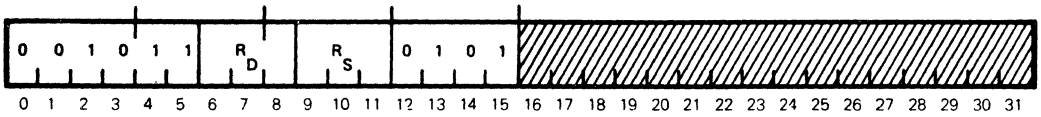
Note

The content of GPR6 are complemented and ANDed with the contents of GPR4. The result is transferred to GPR4. The result is transferred to GPR7. CC2 is set.

EXCHANGE REGISTERS

XCR
s,d

2C05



DEFINITION The word in the GPR specified by R_S is exchanged with the word in the GPR specified by R_D.

SUMMARY EXPRESSION $(R_S) \rightarrow R_D$

$(R_D) \rightarrow R_S$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI Original (R_D) is greater than zero
CC3: ISI Original (R_D) is less than zero
CC4: ISI Original (R_D) is equal to zero

EXAMPLE

Memory Location:	02002
Hex Instruction:	2C A5 (R_D=1, R_S=2)
Assembly Language Coding:	XCR 2,1

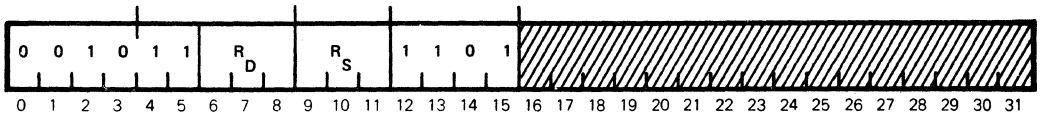
	PSWR	GPR1	GPR2
Before Execution	40002002	00000000	AC8823C1
After Execution	08002004	AC8823C1	00000000

Note The contents of GPR1 and GPR2 are exchanged. CC4 is set.

XCRM
s,d

EXCHANGE REGISTERS MASKED

2C0D



DEFINITION

The contents of the GPR specified by R_S and R_D are each masked (Logical AND Function) with the contents of the Mask register (R4). The results of both masked operations are exchanged.

SUMMARY EXPRESSION

$(R_S)\&(R4) \rightarrow R_D$

$(R_D)\&(R4) \rightarrow R_S$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI original (R_D) and (R4) is greater than zero
CC3: ISI original (R_D) and (R4) is less than zero
CC4: ISI original (R_D) and (R4) is equal to zero

EXAMPLE

Memory Location: 02002
Hex Instruction: 2C AD (R_D=1, R_S=2)
Assembly Language Coding: XCRM 2,1

	PSWR	GPR1	GPR2	GPR4
Before Execution	40002002	6B000000	AC8823C1	000FFFFF
After Execution	08002004	000823C1	00000000	000FFFFF

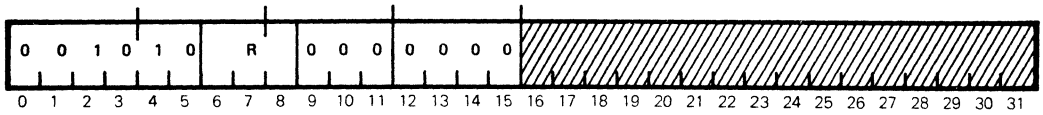
Note

The contents of GPR1 and GPR2 are each ANDed with the contents of GPR4. The results of the masking operation are exchanged and transferred to GPR2 and GPR1, respectively. CC4 is set.

TRANSFER REGISTER TO PSWR

TRSW
S

2800



DEFINITION — Bit positions 1-4 and 13-30 of the General Purpose Register (GPR) specified by R are transferred to the corresponding bit positions of the Program Status Word Register (PSWR).

SUMMARY EXPRESSION — $R_{1\text{-}4,\ 13\text{-}30} \rightarrow PSWR_{1\text{-}4,\ 13\text{-}30}$

CONDITION CODE RESULTS

CC1: ISI (R_1) is equal to one

CC2: ISI (R_2) is equal to one

CC3: ISI (R_3) is equal to one

CC4: ISI (R_4) is equal to one

EXAMPLE

Memory Location:	0069E
Hex Instruction:	28 00 (R=0)
Assembly Language Coding:	TRSW 0

	PSWR	GPR0
Before Execution	6000069E	A0000B4C
After Execution	20000B4C	A0000B4C

Note

1. The contents of GPR0, bits 1-4 and 13-30 are transferred to the PSWR. PSWR bits 0, 5-12, and 31 are unchanged.
2. This instruction can be used in PSD mode to modify CC and PC portions of PSW1.

MEMORY MANAGEMENT INSTRUCTIONS

GENERAL DESCRIPTION

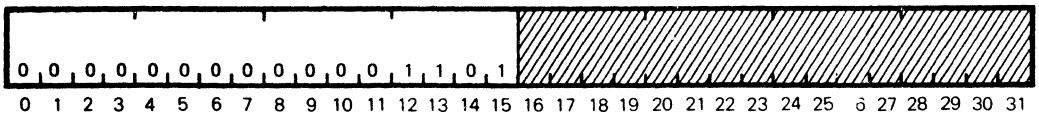
The 32/70 Series Computer provides the capability of accessing memory in any of the following four modes:

1. 512 KB Mode
2. 512 KB Extended Mode
3. 512 KB Mapped Mode
4. Mapped, Extended Mode

The format for the Memory Management instructions vary to the extent that no single format can represent them. The instructions are presented on the following pages.

SET EXTENDED ADDRESSING

000D



DEFINITION The CPU enters the Extended Addressing mode.

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

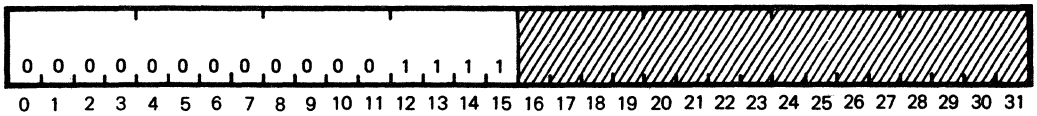
Assembly Language Coding: SEA

NOTES
1. This is a nonprivileged instruction.
2. Sets bit 5 in PSD, word 1.

CEA

CLEAR EXTENDED ADDRESSING

000F



DEFINITION

The CPU enters the Normal (Nonextended) Addressing mode.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

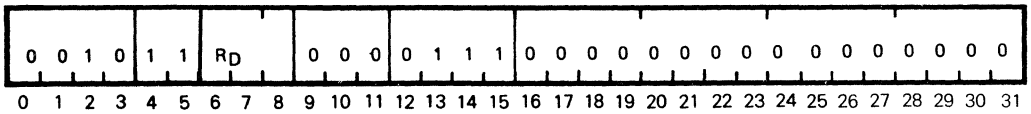
Assembly Language Coding: CEA

NOTES

1. This is a nonprivileged instruction.
2. Clears bit 5 in PSD, word 1.

LOAD MAP

2C07



DEFINITION

Loads the MAP Image Descriptor List (MIDL) from main memory into the CPU MAP Registers. R_D contains the Real Address of a PSD to be used in the MAP loading process.

SUMMARY EXPRESSION

(MIDL) → MAP Registers

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: LMAP R_D

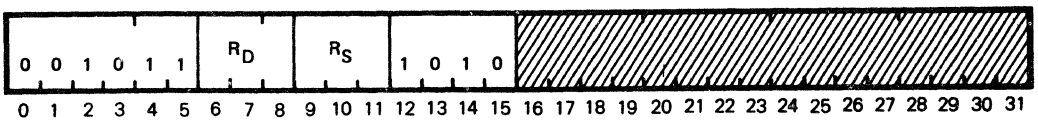
NOTES

1. This instruction primarily used for diagnostic purposes.
2. The CPU must be unmapped.
3. Only MAP Load functions are performed, with no context switching.
4. Attempts to execute this instruction in PSW mode will result in an undefined instruction trap.
5. This is a privileged instruction.
6. This is a fullword instruction.

TMAPR
s,d

TRANSFER MAP TO REGISTER

2C0A



DEFINITION

This instruction causes the even and odd map entries, specified by R_S bits 27-31 to be transferred to the GPR specified by R_D. The least significant map address bit (R_S bit 31) is ignored by the instruction.

SUMMARY EXPRESSION

MAP addressed by R_S 27-31 → R_D

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

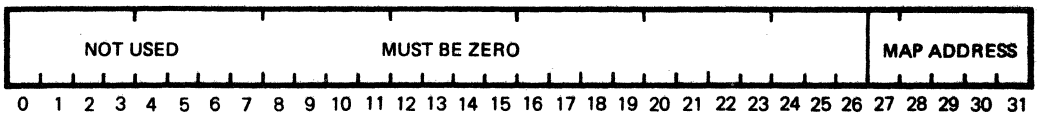
Assembly Language Coding: TMAPR R_S,R_D

NOTES

1. If this instruction is executed in the PSW mode, an undefined instruction trap will occur.

2. This is a halfword privileged instruction.

3. The format for R_S is as follows:



4. The CPU must be Unmapped.

WRITABLE CONTROL STORAGE (WCS) INSTRUCTIONS

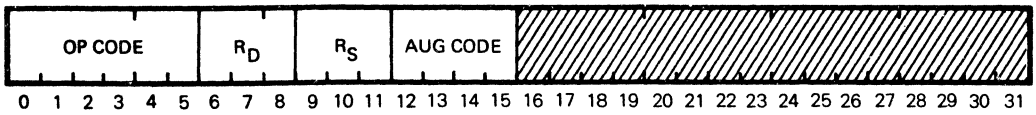
GENERAL DESCRIPTION

Writable Control Storage (WCS) is an option available for use with the CPU or Class F I/O controller. The WCS consists of one or two Random Access Memory (RAM) logic boards, each containing 2K- x 64-bits of RAM memory. The WCS is used to supplement the firmware in the CPU or the Class F I/O controller.

INSTRUCTION FORMAT

There are two instruction formats used for WCS instructions, one for the CPU associated WCS, and one for the Class F I/O Controller associated WCS. The formats are as follows:

CPU ASSOCIATED WCS FORMAT



Bits 0-5 Define the Operation Code.

Bits 6-8 Varies in usage as follows:

Instruction	Usage
WWCS	Specifies the register containing the WCS address.
RWCS	Specifies the register containing the Logical Address in main memory that is to receive the WCS contents.

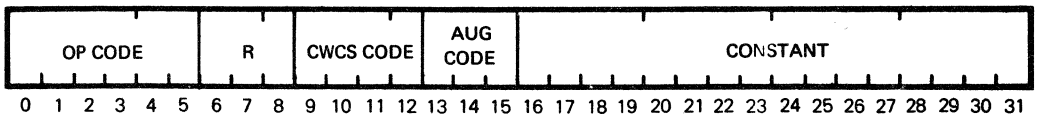
Bits 9-11 Varies in usage as follows:

Instruction	Usage
WWCS	Specifies the register containing the Logical Address in main memory containing the information to be loaded into WCS.
RWCS	Specifies the register containing the WCS address.

Bits 12-15 Define the Augmenting Operating Code.

Bits 16-31 Not used. This is a halfword instruction.

CLASS F I/O CONTROLLER ASSOCIATED WCS FORMAT



Bits 6-8 Specify the GPR, when nonzero, whose contents will be added to the constant to form the logical channel and subaddress.

Bits 9-12 Specifies the Channel WCS Operation Code.

Bits 13-15 Define the Augmenting Operation Code.

Bits 16-31 Specifies a constant that will be added to the contents of R to form the logical Channel and subaddress. If R is zero, only the constant will be used to specify the logical Channel and subaddress.

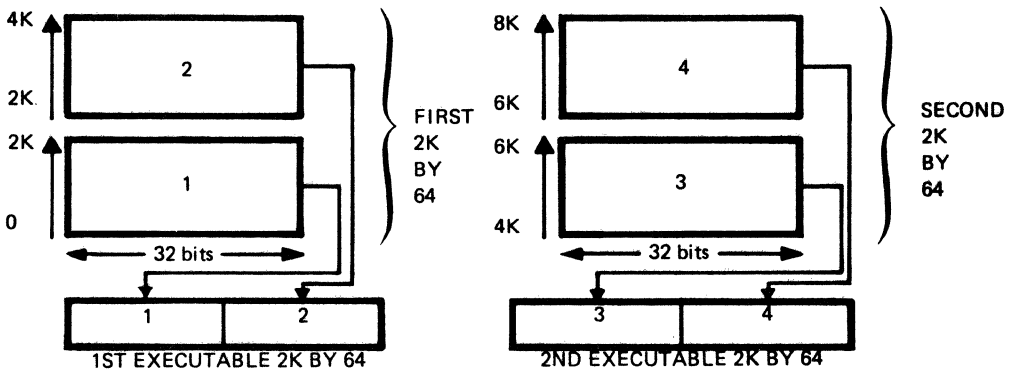
CONDITION CODE UTILIZATION

The Condition Codes remain unchanged when using the CPU associated WCS. When using the class F I/O controller associated WCS, the Condition Codes are changed in accordance with the WCS instructions. Refer to the individual Class F I/O controller WCS instructions for details.

WCS PROGRAMMING

Programming the CPU associated WCS is accomplished by the use of the Write WCS (WWCS) instruction. The contents of the WCS are in the form of microinstructions, which are used to augment the firmware in the CPU. It is beyond the scope of this publication to provide the microinstruction techniques used in the implementation of WCS.

The WCS is organized in 64 bits by 2K modules, allowing up to two modules to be used (4K x 64 bits). Reading or writing WCS is accomplished by alternately placing the first 32-bit word in the first 32 bits and then the second 32-bit word in the second 32 bits. A graphic representation of the Read/Write sequence is shown as follows:



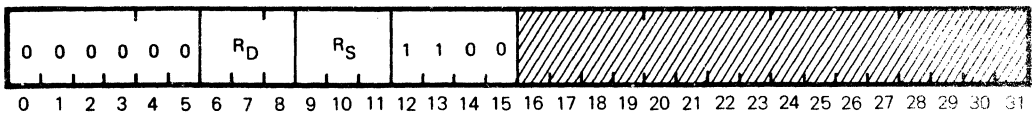
Accessing the CPU associated WCS is accomplished through the use of the Jump to WCS (JWCS) instruction. More complete information of the programming of the WCS is contained in the Writable Control Storage Technical Manual.

Programming of the Class F I/O controller associated WCS is presented in the individual I/O Processor publications.

WRITE WRITABLE CONTROL STORAGE

WWCS
s,d

000C



DEFINITION

This privileged instruction causes the WCS to be written with a single 64-bit word at the location specified by the contents of R_D, with two words in main memory specified by the logical addresses contained in R_S.

The contents of R_S must contain a logical word address that specifies the first word of a two-word pair. F- and C-bits, if specified, are ignored and the address will be interpreted as a word address.

The contents of R_D must contain a right-justified, zero-filled address of the WCS location that is to be written.

If the WCS option is not present or if the WCS address is greater than 4095: CC1 will be set, an Undefined Instruction Trap will occur, and no writing into WCS will take place.

CONDITION CODE RESULTS

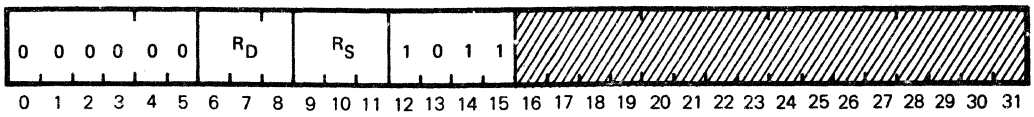
CC1: WCS option not present or address out of range
CC2: Zero
CC3: Zero
CC4: Zero

Assembly Language Coding: WWCS R_S,R_D

RWCS
s,d

READ WRITABLE CONTROL STORAGE

000B



DEFINITION

This privileged instruction causes the contents of a single 64-bit location of WCS specified by the contents of R_S to be written into main memory at the location specified by the logical address contained in R_D.

The contents of R_D must contain a logical word address that specifies the first word pair. F- and C-bits, if specified, are ignored and the address will be interpreted as a word address.

The contents of R_S must contain a right-justified, zero-filled address of the WCS location that is to be read.

If the WCS option is not present or if the WCS address is greater than 4095: CC1 will be set, an Undefined Instruction Trap will occur, and no information will be stored into main memory.

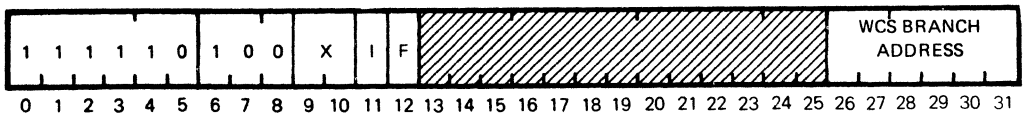
CONDITION CODE RESULTS

CC1: WCS option not present or address out of range
CC2: Zero
CC3: Zero
CC4: Zero

Assembly Language Coding: RWCS R_S, R_D

JUMP TO WRITABLE CONTROL STORAGE

FA00



DEFINITION

This instruction causes an Unconditional Branch to the location specified by the resolved Effective Address. The rules for the Effective Address are as follows:

- Nonindirect - the least significant 6 bits of the Effective Address (index and address) will be used as the WCS entry point address
- Indirect - the least significant 6 bits of the final resolved Effective Address after the resolution of all indirect addresses will be used as the WCS entry point address.

Only the least significant 6 bits of the Effective Address are used and all other bits will be ignored.

When execution in WCS is complete, control will be returned to the next sequential instruction after this instruction.

NOTES

1. Since no registers can be specified by this instruction, the authors of the WCS instructions and the software instructions must mutually agree upon the parameter registers. In general cases, registers 0 and 1 can be used. If the WCS facility is not supported, an Undefined Instruction Trap will occur.
2. If indirect accesses are used, the F-bit must be present in each indirect word.

CONDITION CODE RESULTS

CC1:
CC2:
CC3:
CC4:

All condition code settings will be determined by the WCS routines.

Assembly Language Coding: JWCS X'WCS Branch Addr'

BRANCH INSTRUCTIONS

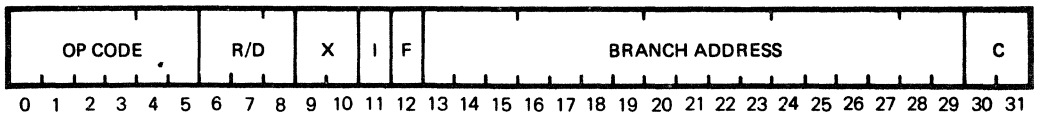
GENERAL DESCRIPTION

Branch instructions provide the capability of testing for certain conditions and branching to another address if the conditions specified by the instruction are satisfied. Branch instructions permit referencing subroutines, repeating segments of programs, or returning to the next instruction within a sequence.

INSTRUCTION FORMAT

The Branch instruction group uses the following instruction format:

MEMORY REFERENCE



Bits 0-5 define the Operation Code.
Bits 6-8 vary in usage as follows:

Instruction	Contents/Usage
BU, BFT	000
BCT, BCF	D field
BIB, BIH, BIW, BID	Register Number
BL	001
BRI	010

Bits 9-10 designate one of three index registers.
Bit 11 indicates whether an indirect addressing operation is to be performed.
Bit 12 is zero.
Bits 13-30 specifies the branch address when X and I fields are zero.
Bit 31 is zero.

CONDITION CODE UTILIZATION

Condition Code results during branching operations are unique because they reflect the state of the indirect bit of the instruction and the state of bits 1, 2, 3, and 4 of the indirect address obtained from the specified memory location.

BRANCH PROGRAMMING

The usual procedure for calling a subroutine is to execute a Branch and Link (BL) whose effective address is the starting location of the routine. Since PC+1 is saved in GPR0, a subsequent return can be made to the location following the BL by executing a TRSW 0. The PSW including the PC+1 word is saved in GPR0. Hence, the subroutine can be reentrant (pure); i.e., memory is not modified by calling it. If we wish to use GPR0 in the subroutine, we can store the return address in a convenient location in memory, location B, with an STW 0, B, and then return with a BU *B.

Consider a move subroutine to move 50 words beginning at TAB. The routine begins at MOVE, whose address is stored in C.MOVE. The main program would contain:

```
BL        *C.MOVE

...                          ; Return here
```

GPR1 is used as an Index register for counting through the table and GPR5 is used to output the data. The starting address of the table is in TAB 1. The subroutine is as follows:

```
                        COUNT EQU 50
MOVE      LI    1,      -COUNT          Negative of table length
LOOP      LW    5,      TAB+COUNT,1     Get next word
          STW   5,      TAB1+COUNT,1    Store in new buffer
          BIW   1,      LOOP            Increment and test for end
          TRSW  0                       Return
```

Argument Passing

Given an arithmetic subroutine that operates on arguments in GPR5 and GPR6, leaving the result in GPR6, the subroutine call is as follows:

```
BL        SQRT      Call with arguments in GPR5 and GPR6

. . .
```

The subroutine is as follows:

```
SQRT                          Arithmetic operations
  .
  .
TRSW 0                        Return to Call + 1 word
```

In the preceding example, the calling program must load the General Purpose Registers before calling the subroutine. It is often convenient for the program to supply the arguments (or the addresses of the locations that contain them) with the call, and for the subroutine to handle the data transfers. With this method, the program gives the arguments in the two memory locations following the BL.

```
BL        SQRT

...                           Argument 1
...                           Argument 2
...                           Return here with result in GPR6
```

The return is made to the location following the second argument with the result in GPR6.

```
SQRT    TRR     0,1
        LD      6,0,1       Pick up Arguments 1 and 2
        . .
        . .
        ADI     0,8         Increment return address by 2 words
        TRSW    0           Return to Call + 3 words
```

An alternate method which allows up to six arguments to be passed per instruction utilizes the Load File instruction as follows:

```
SQRT    TRR     0,1
        LF      2,0,1       Pick up Arguments 1-6
        . .
        . .
        ADI     0,24        Increment return address by 6 words
        TRSW    0           Return to Call + 7 words
```

The next method passes an address list instead of arguments following the BL; otherwise, it is identical to the method described above.

```
        BL      SQRT
        . .                 Address of Argument 1
        . .                 Address of Argument 2
        .
        .
        .

SQRT    TRR     0,1
        LW      6,*0,1      Pick up Argument 1
        ADI     1,4
        LW      7,*0,1      Pick up Argument 2
        .
        .
        ADI     0,8         Increment return address by 2 words
        TRSW    0           Return to Call + 3 words
```

The next method is the same as the previous example except that argument 1 is a table, and the result replaces the second argument in memory:

```
        BL      SQRT
        ...                 Address of Argument 1
        ...                 Address of Argument 2 and result
        .
        .
        .

SQRT    TRR     0,3         Pick up base address of table, Argument 1
        TRR     0,1
        ABR     29,1        Increment return address by 4 words
        LW      6,*0,1      Pick up Argument 2
        .
        .
        .
```

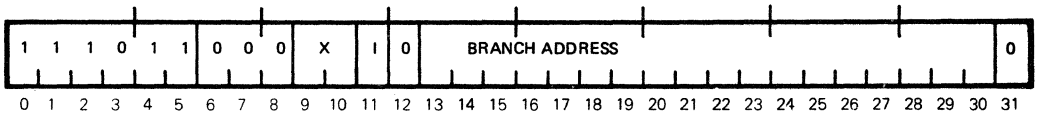
The final method is similar to the previous versions except that GPR1-GPR7 are not disturbed:

```
SQRT    STF     0, SAVE   Save General Purpose Registers
        TRR     0,1
        LW      6, *0,1   Pick up Arguments
        ADI     1,4
        LW      7,*0,1
        .
        .
        .
        ST      6,*0,1    Store result
        LF      0, SAVE   Restore General Purpose Registers
        ADI     0,8       Increment return address by 2 words
        TRSW    0         Return to Call + 3 words
SAVE    REZ     1F        Eight zero-filled words on a file boundary
```

BU
*m,x

BRANCH UNCONDITIONALLY

EC00



DEFINITION

The Effective Address (bits 13-30) in the instruction is transferred to the corresponding bit positions in the Program Status Word Register (PSWR). This causes program control to be transferred to any word or halfword location in memory. Bit positions 1-12 of the PSWR remain unchanged if the indirect bit is equal to zero. If the indirect bit of the Instruction Word is equal to one, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR. Bit 0 (privileged state bit) of the PSWR remains unchanged. The Extended mode bit remains unchanged. Bits 0 and 5 are changed only by a BRI indirect.

SUMMARY EXPRESSION

$EA_{13-30} \rightarrow PSWR_{13-30}$, IF I=0

$(EWL_{1-4}$ and $_{13-30}) \rightarrow PSWR_{1-4}$ and $_{13-31}$, IF I=1

CONDITION CODE RESULTS

If the indirect bit is equal to zero, the Condition Code remains unchanged.

CC1: ISI (I) is equal to one and (EWL_1) is equal to one
CC2: ISI (I) is equal to one and (EWL_2) is equal to one
CC3: ISI (I) is equal to one and (EWL_3) is equal to one
CC4: ISI (I) is equal to one and (EWL_4) is equal to one

EXAMPLE 1

Memory Location: 01000
Hex Instruction: EC 00 14 14 (X=0, I=0)
Assembly Language Coding: BU X'1414'

Before Execution

PSWR
20001000

After Execution

PSWR
20001414

Note

The contents of bits 13-30 of the instruction replace the corresponding portion of the PSWR. The Condition Code remains unchanged.

EXAMPLE 2

Memory Location: 01000
Hex Instruction: EC 10 14 14 (X=0, I=1)
Assembly Language Coding: BU *X'1414'

Before Execution

PSWR	Memory Word 01414
80001000	700015AC

After Execution

PSWR	Memory Word 01414
F00015AC	700015AC

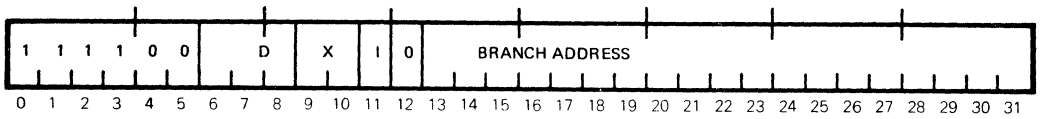
Note

The contents of bits 1-30 of memory word 01414 replace the previous contents of bits 1-4 and 13-31 of the PSWR.

BRANCH CONDITION FALSE

BCF
v,*m,x

F000



DEFINITION

The Effective Address (bits 13-30) in the instruction is transferred to the corresponding bit positions in the Program Status Word Register (PSWR), if the condition specified by the D field (bits 6-8 of the instruction) is present. The seven specifiable conditions are tabulated below. If the condition is not as specified, the next instruction in sequence is executed. If the indirect bit of the Instruction Word is equal to one, and the branch occurs, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR. Bits 0, and 5-15 are unchanged.

D Field (Hex)	Branch Condition (Branch if):
1	CC1=zero
2	CC2=zero
3	CC3=zero
4	CC4=zero
5	CC2 and CC4 both = zero
6	CC3 and CC4 both = zero
7	CC1, CC2, CC3, and CC4 all = zero

CONDITION CODE RESULTS

The resulting Condition Code remains unchanged if the indirect bit (bit 11) is equal to zero.

CC1: ISI (I) is equal to one and (EWL_1) is equal to one
CC2: ISI (I) is equal to one and (EWL_2) is equal to one
CC3: ISI (I) is equal to one and (EWL_3) is equal to one
CC4: ISI (I) is equal to one and (EWL_4) is equal to one

EXAMPLE

Memory Location: 02094
Hex Instruction: F1 00 21 4C ($C_1C_2C_3$=2,X=0,I=0)
Assembly Language Coding: BCF 2,X'214C'

Before Execution

PSWR
10002094

After Execution

PSWR
1000214C

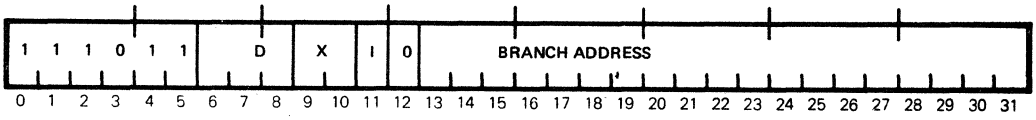
Note

Condition Code bit 2 is not set. The Effective Address (in this case bit 13-30 of the instruction) is transferred to the PSWR.

BCT
v,*m,x

BRANCH CONDITION TRUE

EC00



DEFINITION

The Effective Address (bits 13-30) in the instruction is transferred to the corresponding bit positions in the Program Status Word Register (PSWR), if the condition specified by the D field (bits 6-8) is present. The seven specifiable conditions are tabulated below. If the indirect bit of the Instruction Word is equal to one, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR. Bits 0 and 5-12 are unchanged.

D Field (Hex)	Branch Condition (Branch if):
1	CC1=one
2	CC2=one
3	CC3=one
4	CC4=one
5	CC2 v CC4=one
6	CC3 v CC4=one
7	CC1 v CC2 v CC4=one

CONDITION CODE RESULTS

The resulting Condition Code remains unchanged if the indirect bit (bit 11) is equal to zero.

CC1: ISI (I) is equal to one and (EWL_1) is equal to one
CC2: ISI (I) is equal to one and (EWL_2) is equal to one
CC3: ISI (I) is equal to one and (EWL_3) is equal to one
CC4: ISI (I) is equal to one and (EWL_4) is equal to one

EXAMPLE

Memory Location: 01000
Hex Instruction: EC 80 14 14 (Condition=1, X=0, I=0)
Assembly Language Coding: BCT, 1,X'1414'

Before Execution

PSWR
50001000

After Execution

PSWR
50001414

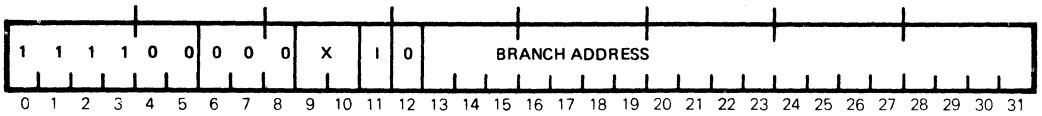
Note

The contetns of bits 13-30 of the instruction are transferred to bits 13-30 of the PSWR.

BRANCH FUNCTION TRUE

BFT
*m,X

F000



DEFINITION

The Effective Address (bits 13-30) in the instruction is transferred to the corresponding bit positions in the Program Status Word Register (PSWR) if the function bit in the mask register (R4) for the Condition Code, 1 of the 16 possible combinations of the 4 Condition Code bits which corresponds to the current condition code, is equal to one. The function F is defined by the 16 least significant bits of the mask register. All 16 Condition Codes of the 4 variables A=CC1, B=CC2, C=CC3, D=CC4 are defined below.

$$F = \bar{A}\bar{B}\bar{C}\bar{D}\ R4_{16} \vee \bar{A}\bar{B}\bar{C}D\ R4_{17} \vee \bar{A}\bar{B}C\bar{D}\ R4_{18} \vee \bar{A}\bar{B}CD\ R4_{19}$$
$$\bar{A}B\bar{C}\bar{D}\ R4_{20} \vee \bar{A}B\bar{C}D\ R4_{21} \vee \bar{A}BC\bar{D}\ R4_{22} \vee \bar{A}BCD\ R4_{23}$$
$$A\bar{B}\bar{C}\bar{D}\ R4_{24} \vee A\bar{B}\bar{C}D\ R4_{25} \vee A\bar{B}C\bar{D}\ R4_{26} \vee A\bar{B}CD\ R4_{27}$$
$$AB\bar{C}\bar{D}\ R4_{28} \vee AB\bar{C}D\ R4_{29} \vee ABC\bar{D}\ R4_{30} \vee ABCD\ R4_{31}$$

Therefore, any logical function of the four variables stored in the Condition Code register can be evaluated by storing the proper 16-bit function code in the mask register. The next instruction in sequence is executed if the function is equal to zero. If the Indirect bit of the instruction word is equal to one, bit positions 1-12 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR. Bits 0 and 5 are unchanged.

SUMMARY EXPRESSION

If F=1 & I=0, $EA_{13-30} \rightarrow PSWR_{13-30}$

If F=1 & I=1, $EA_{1-30} \rightarrow PSWR_{1-30}$

If F=0 $PSWR_{13-30} + 1_{29} \rightarrow PSWR_{13-30}$

CONDITION CODE RESULTS

The resulting condition code remains unchanged if the indirect bit (bit 11) is equal to zero.

CC1: ISI (I) is equal to one and EA_1 is equal to one
CC2: ISI (I) is equal to one and EA_2 is equal to one
CC3: ISI (I) is equal to one and EA_3 is equal to one
CC4: ISI (I) is equal to one and EA_4 is equal to one

EXAMPLE

Memory Location:	01000
Hex Instruction:	F0 00 20 00 (X=0, I=0)
Assembly Language Coding:	BFT X'2000'

	PSWR	GPR4
Before Execution	70001000	00000002
After Execution	PSWR	GPR4
	70002000	00000002

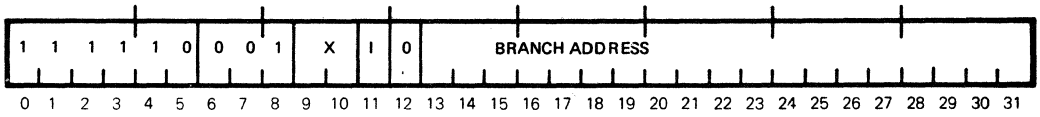
Note

Bit 30 of GPR4 defines a function for which CC1=CC2=CC3=1,CC4=0. This function is true, so a branch is effected.

BL
*m,x

BRANCH AND LINK

F880



DEFINITION

The contents of the Program Status Word Register (PSWR) are incremented by one word and transferred to General Purpose Register 0. If the indirect bit of the Instruction Word is equal to zero, the Effective Address (bit 13-30) is transferred to the corresponding bit positions of the PSWR. Bit positions 1-12 of the PSWR remain unchanged. If the indirect bit of the Instruction Word is equal to one, bit positions 1-4 of the last memory word in the indirect chain are also transferred to the corresponding bit positions of the PSWR. Bit 0 (privileged state bit), and bits 5-12 of the PSWR remain unchanged.

SUMMARY EXPRESSION

(PSWR) → R0

EA → $PSWR_{13-30}$, if I=zero

EWL_{1-12}, EA → $PSWR_{1-4 \text{ and } 13-30}$, if I=one

CONDITION CODE RESULTS

If the indirect bit is equal to zero, the Condition Code remains unchanged.

CC1: (ISI) (I) is equal to one and (EWL_1) is equal to one
CC2: (ISI) (I) is equal to one and (EWL_2) is equal to one
CC3: (ISI) (I) is equal to one and (EWL_3) is equal to one
CC4: (ISI) (I) is equal to one and (EWL_4) is equal to one

EXAMPLE

Memory Location:	0894C
Hex Instruction:	F8 80 A3 78 (X=0, I=0)
Assembly Language Coding:	BL X'A378'

	PSWR	GPR0
Before Execution	1000894C	12345678
After Execution	1000A378	10008950

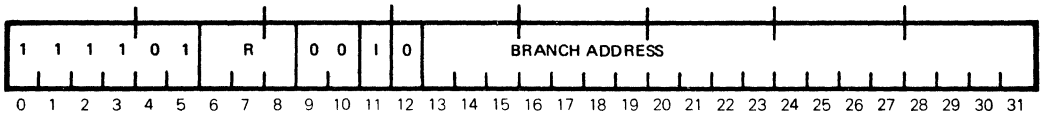
Note

The contents of the PSWR are transferred to GPR0. The contents of bits 13-30 of the instruction are transferred to bits 13-30 of the PSWR.

BRANCH AFTER INCREMENTING BYTE

BIB
d,*m

F400



DEFINITION

The contents of the GPR specified by R are incremented in bit position 31. If the result is nonzero the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13-30 and bit positions 1-4 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed. Bits 0 and 5 are unchanged.

SUMMARY EXPRESSION

$(R) + 1_{31} \rightarrow R$

$EA \rightarrow PSWR_{13\text{-}30}$, if result $\neq 0$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	1B204
Hex Instruction:	F4 01 B1 A8 (R=0, I=0)
Assembly Language Coding:	BIB 0,X'1B1A8'

Before Execution	PSWR	GPR0
	2001B204	FFFFFFFF
After Execution	PSWR	GPR0
	2001B208	00000000

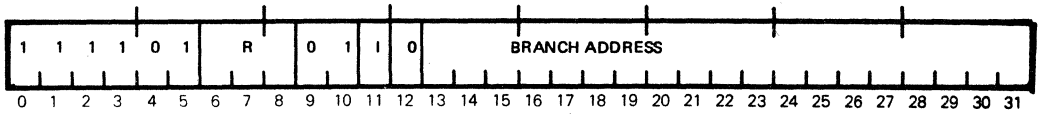
Notes

1. The contents of the GPR0 are incremented by one at bit position 31. Since the result is zero, no branch occurs.
2. Indexing is not allowed.
3. If the indirect bit of the Instruction Word is equal to one, and the branch occurs, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR. Bits 0 and 5-12 are unchanged.
4. The instruction following may not be the target of the System Control Panel or Serial Control Panel Halt.

BIH
d,*m

BRANCH AFTER INCREMENTING HALFWORD

F420



DEFINITION

The contents of the GPR specified by R are incremented in bit position 30. If the result is nonzero the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13-30 and bit positions 1-4 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{30} \rightarrow R$

$EA \rightarrow PSWR_{13-30}$, if result $\neq 0$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	039A0
Hex Instruction:	F5 20 39 48 (R=2, I=0)
Assembly Language Coding:	BIH 2,X'3948'

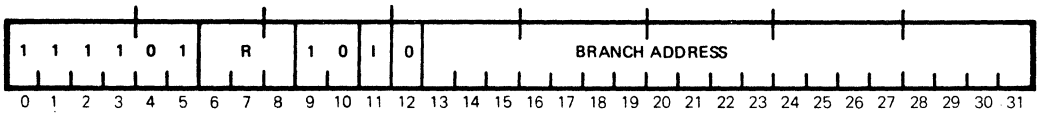
Before Execution	PSWR	GPR2
	100039A0	FFFFD72A
After Execution	PSWR	GPR2
	10003948	FFFFD72C

Notes

1. The contents of GPR2 are incremented by one in bit position 30. The result is replaced in GPR2 and a branch occurs to address 03948.
2. Indexing is not allowed.
3. If the indirect bit of the Instruction Word is equal to one, and the branch occurs, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR. Bits 0 and 5-12 are unchanged.
4. The instruction following may not be the target of the System Control Panel or Serial Control Panel Halt.

BRANCH AFTER INCREMENTING WORD

F440



DEFINITION

The contents of the GPR specified by R are incremented in bit position 29. If the result is nonzero, the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13-30 and bit positions 1-4 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{29} \rightarrow R$

$EA \rightarrow PSWR_{13-30}$, if result $\neq 0$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	04A38
Hex Instruction:	07 40 4B 2C (R=6, I=0)
Assembly Language Coding:	BIW 6,X'4B2C'

	PSWR	GPR6
Before Execution	60004A38	FFFFDC18
After Execution	PSWR	GPR6
	60004B2C	FFFFDC1C

Notes

1. The content of GPR6 is incremented by one at bit position 29, and the result is transferred to GPR6. The Effective Address of the BIW instruction, (04B2C), replaces the previous contents of the PSWR, bits 12-30.

2. Indexing is not allowed.

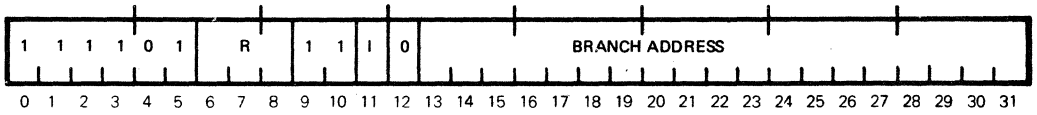
3. If the indirect bit of the Instruction Word is equal to one, and the branch occurs, bit positions 1-4 of the last memory word in the direct chain are transferred to the corresponding bit positions of the PSWR. Bits 0 and 5-12 are unchanged.

4. The instruction following may not be the target of the System Control Panel or Serial Control Panel Halt.

BID
d,*m

BRANCH AFTER INCREMENTING DOUBLEWORD

F460



DEFINITION

The contents of the GPR specified by R are incremented in bit position 28. If the result is nonzero the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13-30 and bit positions 1-4 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{28} \rightarrow R$

EA $\rightarrow$ $PSWR_{13-30}$, if result $\neq 0$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	0930C
Hex Instruction:	F5 E0 91 A6 (R=3, I=0)
Assembly Language Coding:	BID 3,X'91A6'

	PSWR	GPR3
Before Execution	0800930C	FFFFFFF8
After Execution	08009310	00000000

Notes

1. The content of GPR3 is incremented by one at bit position 28 and replaced. Since the result is zero, no branch occurs.
2. Indexing is not allowed.
3. If the indirect bit of the Instruction Word is equal to one, and the branch occurs, bit positions 1-4 of the last memory word in the direct chain are transferred to the corresponding bit positions of the PSWR. Bits 0 and 5-12 are unchanged.
4. The instruction following may not be the target of the System Control Panel or Serial Control Panel Halt.

COMPARE INSTRUCTIONS

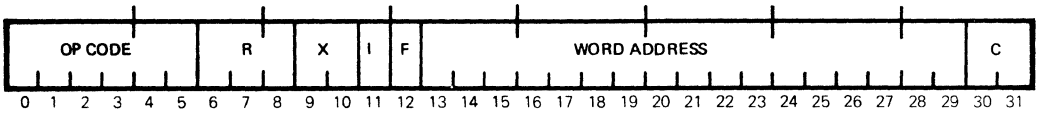
GENERAL DESCRIPTION

Compare instructions provide the capability of comparing data in memory and General Purpose Registers. These operations can be performed on bytes, halfwords, words, or doublewords. Provisions have also been made to allow the result of compare operations to be masked with the contents of the Mask register before final testing.

INSTRUCTION FORMAT

The Compare instruction group uses three instruction formats.

MEMORY REFERENCE



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0-7).

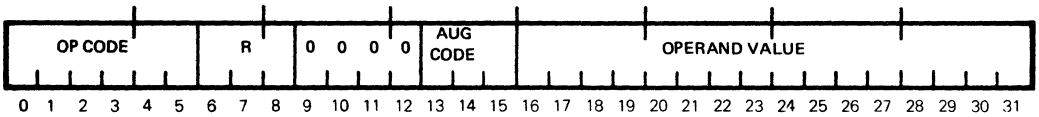
Bits 9-10 designate one of three index registers.

Bit 11 indicates whether an indirect addressing operation is to be performed.

Bit 12-31 specify the address of the operand when the X and I fields equal to zero.

Note Additional information on the Memory Reference instruction format is included with the Load/Store instruction formats.

IMMEDIATE



Bits 0-5 define the Operation Code.

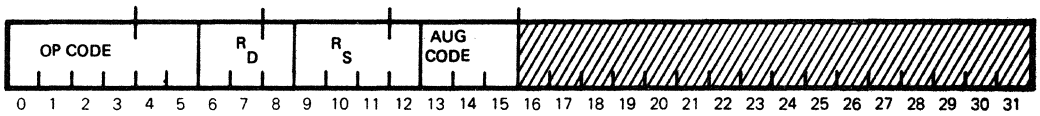
Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-12 unassigned.

Bits 13-15 define Augmenting Operation Code.

Bits 16-31 contain the 16-bit operand value.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

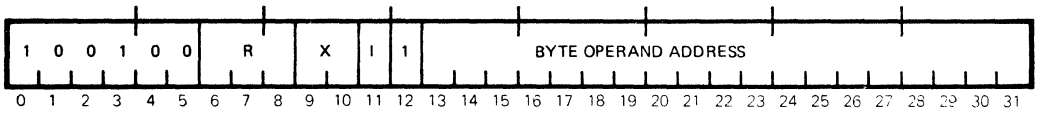
CONDITION CODE UTILIZATION

A Condition Code is set during most Compare instructions to indicate whether the operation produced a result greater than, less than, or equal to zero.

COMPARE ARITHMETIC WITH MEMORY BYTE

CAMB
d,*m,x

9008



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed, right justified, and subtracted algebraically from the word in the GPR specified by R. The result of the subtraction causes one of the Condition Code bits (2-4) to be set. The contents of the GPR specified by R and the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

(R) - (EBL) → SCC_{2-4}

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R) is greater than (EBL)
CC3: ISI (R) is less than (EBL)
CC4: ISI (R) is equal to (EBL)

EXAMPLE

Memory Location: 01000
Hex Instruction: 90 88 10 B5 (R=1, X=0, I=0)
Assembly Language Coding: CAMB 1,X'10B5'

Before Execution	PSWR	GPR1	Memory Byte 010B5
	08001000	000000B6	C7
After Execution	PSWR	GPR1	Memory Byte 010B5
	10010004	000000B6	C7

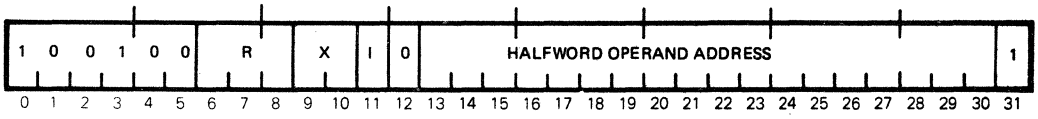
Note

CC3 is set, indicating that the contents of GPR1 are less than the contents of memory byte 010B5.

CAMH
d,*m,x

COMPARE ARITHMETIC WITH MEMORY HALFWORD

9000



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed, and the sign bit is extended 16 bits to the left to form a word. The resulting word is subtracted algebraically from the word in the GPR specified by R. The result of the subtraction causes one of the Condition Code bits (2-4) to be set. The word in the GPR specified by R and the halfword specified by the EHA remain unchanged.

SUMMARY EXPRESSION

$(R) - (EHL)_{SE} \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R) is greater than $(EHL)_{SE}$
CC3: ISI (R) is less than $(EHL)_{SE}$
CC4: ISI (R) is equal to $(EHL)_{SE}$

EXAMPLE

Memory Location: 0379C
Hex Instruction: 92 00 39 77 (R=4, X=0, I=0)
Assembly Language Coding: CAMH 4,X'3976'

	PSWR	GPR4	Memory Halfword 03976
Before Execution	0800379C	00008540	8640
After Execution	200037A0	00008540	8640

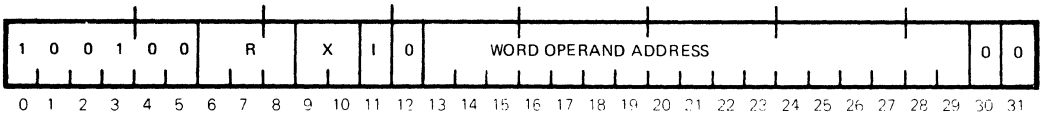
Note

CC2 is set indicating that the contents of GPR4 are greater than the contents of memory halfword 03976 (a negative value).

COMPARE ARITHMETIC WITH MEMORY WORD

CAMW
d,*m,x

9000



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and subtracted algebraically from the word in the GPR specified by R. The result of the subtraction causes one of the Condition Code bits (2-4) to be set. The word in the GPR specified by R and the word specified by the EWA remain unchanged.

SUMMARY EXPRESSION

(R) - (EWL) → SCC_{2-4}

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R) is greater than (EWL)
CC3: ISI (R) is less than (EWL)
CC4: ISI (R) is equal to (EWL)

EXAMPLE

Memory Location: 05B20
Hex Instruction: 93 00 5C 78 (R=6, X=0, I=0)
Assembly Language Coding: CAMW 6,X'5C78'

Before Execution	PSWR	GPR6	Memory Word 05C78
	40005B20	9E03B651	A184F207
After Execution	PSWR	GPR6.	Memory Word 05C78
	10005B24	9E03B651	A184F207

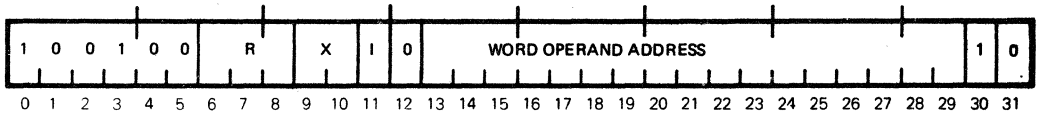
Note

CC3 is set indicating that the contents of the GPR6 are less than the contents of memory word 05C78.

CAMD
d,*m,x

COMPARE ARITHMETIC WITH MEMORY DOUBLEWORD

9000



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and subtracted algebraically from the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The result of the subtraction causes one of the Condition Code bits (2-4) to be set. The doubleword in the GPR specified by R and R+1, and the doubleword specified by the EDA remain unchanged.

SUMMARY EXPRESSION

(R, R+1) - (EDL) → SCC_{2-4}

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R, R+1) is greater than (EDL)
CC3: ISI (R, R+1) is less than (EDL)
CC4: ISI (R, R+1) is equal to (EDL)

EXAMPLE

Memory Location: 27C14
Hex Instruction: 92 02 7F 52 (R=4, X=0, I=0)
Assembly Language Coding: CAMD 4,X'27F50'

Before Execution

PSWR	GPR4	GPR5
20027C14	7AE0156D	47B39208

Memory Word 27F50	Memory Word 27F54
7AE0156D	47B39208

After Execution

PSWR	GPR4	GPR5
08027C18	7AE0156D	47B39208

Memory Word 27F50	Memory Word 27F54
7AE0156D	47B39208

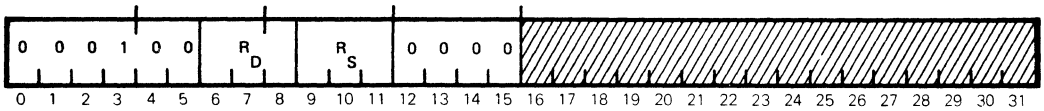
Note

CC4 is set indicating that the doubleword obtained from GPR4 and GPR5 is equal to that obtained from the memory words 27F50 and 27F54.

COMPARE ARITHMETIC WITH REGISTER

CAR
s,d

1000



DEFINITION

The word in the GPR specified by R_S is subtracted algebraically from the word in the GPR specified by R_D. The result of the subtraction causes one of the Condition Code bits (2-4) to be set. The words specified by R_S and R_D remain unchanged.

SUMMARY EXPRESSION

$(R_D) - (R_S) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_D) is greater than (R_S)
CC3: ISI (R_D) is less than (R_S)
CC4: ISI (R_D) is equal to (R_S)

EXAMPLE

Memory Location:	0B3C2
Hex Instruction:	10 10 (R_D=0, R_S=1)
Assembly Language Coding:	CAR 1,0

	PSWR	GPR0	GPR1
Before Execution	0800B3C2	58DF620A	6A92B730
After Execution	1000B3C4	58DF620A	6A92B730

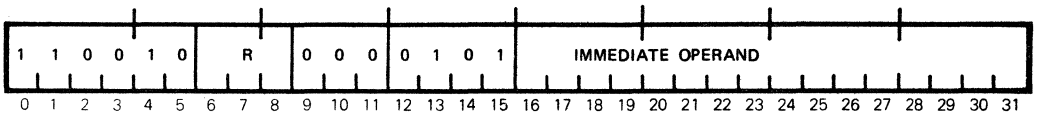
Note

CC3 is set indicating that the contents of GPR0 are less than the contents of GPR1.

CI
d,v

COMPARE IMMEDIATE

C805



DEFINITION

The sign bit (bit 16) of the immediate operand is extended 16 bit positions to the left to form a word. This word is subtracted from the word in the GPR specified by R. The result of the subtraction causes one of the Condition Code bits (2-4), to be set. The word in the GPR specified by R and the immediate operand (bit 16-31) remain unchanged.

SUMMARY EXPRESSION

$(R) - (IW_{16-31})_{SE} \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R) is greater than $(IW_{16-31})_{SE}$
CC3: ISI (R) is less than $(IW_{16-31})_{SE}$
CC4: ISI (R) is equal to $(IW_{16-31})_{SE}$

EXAMPLE

Memory Location: 0A794
Hex Instruction: C8 85 71 A2 (R=1)
Assembly Language Coding: CI 1,X'71A2'

	PSWR	GPR1
Before Execution	4000A794	00005719
After Execution	1000A798	00005719

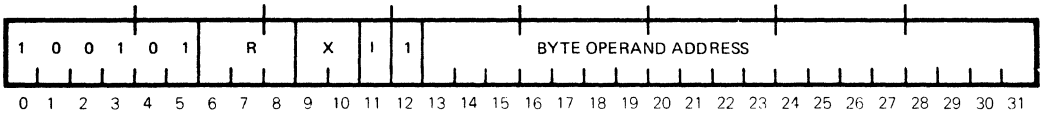
Note

CC3 is set, indicating that the contents of GPR1 are less than the immediate operand.

COMPARE MASKED WITH MEMORY BYTE

CMMB
d,*m,x

9408



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed, and 24 zeros are appended to the most significant end to form a word. This word is logically compared (Exclusive OR Function) with the word in the GPR specified by R. The resulting word is then masked (Logical AND Function) with the contents of the Mask register (R4). The masked result is tested and Condition Code bit 4 is set if all 32 bits equal zero. The word in the GPR specified by R and the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus 0_{0-23},\ (EBL)]\ \&\ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: ISI Result is equal to zero

EXAMPLE

Memory Location: 00800
Hex Instruction: 94 08 09 17 (R=0, X=0, I=0)
Assembly Language Coding: CMMB 0,X'917'

	PSWR	GPR0	GPR4	Memory Byte 00917
Before Execution	10000800	000000A1	000000F0	A9
After Execution	08000804	000000A1	000000F0	A9

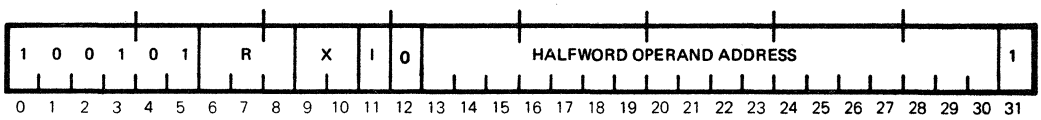
Note

The contents of GPR0 and memory byte 00917 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

CMMH
d,*m,x

COMPARE MASKED WITH MEMORY HALFWORD

9400



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed, and the sign (bit 16) is extended 16 bits to the left to form a word. The resulting word is logically compared (Exclusive OR Function) with the word in the GPR specified by R. The resulting word is then masked (Logical AND Function) with the contents of the Mask register (R4). The masked result is tested and Condition Code bit 4 is set if all 32 bits equal zero. The word in the GPR specified by R and the halfword specified by the EHA remain unchanged.

SUMMARY EXPRESSION

$[\ (R) \oplus (EHL)_{SE}]\ \&\ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: ISI result is equal to zero

EXAMPLE

Memory Location: 061B8
Hex Instruction: 95 00 62 93 (R=2, X=0, I=0)
Assembly Language Coding: CMMH 2,X'6293'

Before Execution	PSWR	GPR2	GPR4	Memory Halfword 06292
	100061B8	09A043B6	00004284	46FC
After Execution	PSWR	GPR2	GPR4	Memory Halfword 06292
	080061BC	09A043B6	00004284	46FC

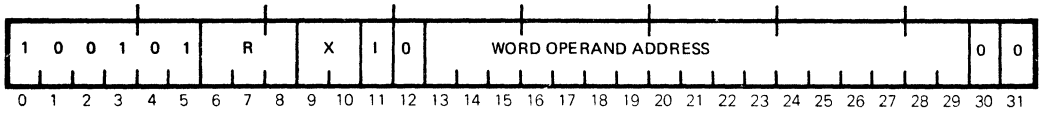
Note

The contents of GPR2 and memory halfword 06292 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

COMPARE MASKED WITH MEMORY WORD

CMMW
d,*m,x

9400



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and logically compared (Exclusive OR Function) with the word in the GPR specified by R. The result of the comparison is then masked (Logical AND Function) with the contents of the Mask register (R4). The masked result is tested and Condition Code bit 4 is set if all 32 bits equal zero. The word in the GPR specified by R and the word specified by the EWA remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus (EWL)] \ \& \ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: ISI result is equal to zero

EXAMPLE

Memory Location: 13A74
Hex Instruction: 97 01 3C 94 (R=6, X=0, I=0)
Assembly Language Coding: CMMW 6,X'3C94'

	PSWR	GPR4	GPR6	Memory Word 13C94
Before Execution	08013A74	00FFFF00	132A1C04	472A3D04
After Execution	00013A78	00FFFF00	132A1C04	472A3D04

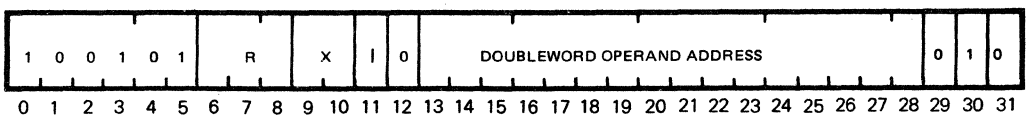
Note

The contents of GPR6 and memory word 13C94 are not equal within the bit positions specified by the contents of GPR4.

CMMD
d,*m,x

COMPARE MASKED WITH MEMORY DOUBLEWORD

9400



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and compared (Exclusive OR Function) with the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. Each result from the comparison is then masked (Logical AND Function) with the contents of the Mask register (R4). The doubleword masked result is tested and Condition Code bit 4 is set if all 64 bits equal zero. The doubleword in the GPR specified by R and R+1 and the doubleword specified by the EDA remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus (EWL)]$ & (R4), $[(R+1) \oplus (EWL+1)]$ & (R4) $\rightarrow$ SCC_4

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: ISI result is equal to zero

EXAMPLE

Memory Location:		03000
Hex Instruction:		97 00 31 BA (R=6, X=0, I=0)
Assembly Language Coding:		CMMD 6,X'31B8'

Before Execution

PSWR	GPR4	GPR6	GPR7
10003000	000FFFFF	FFF3791B	890A45D6

Memory Word 031B8	Memory Word 031BC
0003791B	890A45C2

After Execution

PSWR	GPR4	GPR6	GPR7
00003004	000FFFFF	FFF3791B	890A45D6

Memory Word 031B8	Memory Word 031BC
0003791B	890A45C2

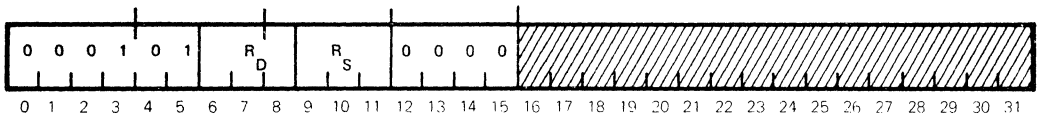
Note

The contents of GPR7 and memory word 031BC differ within the bit positions specified by the contents of GPR4.

COMPARE MASKED WITH REGISTER

CMR
s,d

1400



DEFINITION

The word in the GPR specified by R_D is logically compared (Exclusive OR Function) with the word in the GPR specified by R_S. The result of the comparison is then masked (Logical AND function) with the contents of the Mask register (R4). The result is tested and Condition Code bit 4 is set if all 32 bits equal zero. The words specified by R_S and R_D remain unchanged.

SUMMARY EXPRESSION

$[(R_D) \oplus (R_S)]$ & (R4) $\rightarrow$ SCC_4

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: ISI result is equal to zero

EXAMPLE

Memory Location: 050D2
Hex Instruction: XXXX14 A0 (R_D=1, R_S=2)
Assembly Language Coding: CMR 2,1

	PSWR	GPR1	GPR2	GPR4
Before Execution	100050D2	583C94A2	0C68C5F6	AAAAAAAA
After Execution	080050D4	583C94A2	0C68C5F6	AAAAAAAA

Note

The contents of GPR1 and GPR2 are identical within the bit positions specified by the contents of GPR4. CC4 is set.

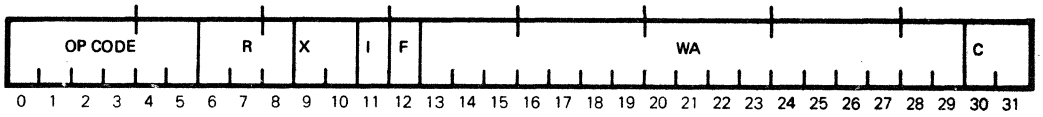
GENERAL DESCRIPTION

The Logical instruction group provides the capability of performing AND, OR, and Exclusive OR operations on bytes, halfwords, and doublewords in memory and General Purpose Registers. Provisions have also been made to allow the result of Register-to-Register OR and Exclusive OR operations to be masked with the contents of Mask register (R4) before final storage.

INSTRUCTION FORMATS

The Logical instruction group uses the following two instruction formats:

MEMORY REFERENCE



Bits 0-5 define the Operation Code.

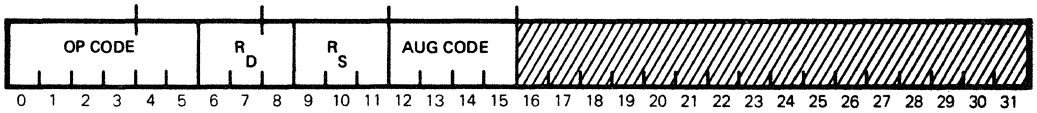
Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-10 designate one of three index registers.

Bit 11 indicates whether an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when the X and I fields are equal to zero.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

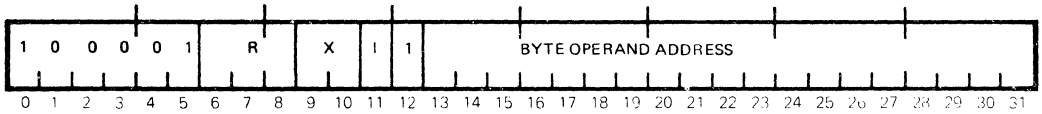
CONDITION CODE UTILIZATION

A Condition Code is set during execution of most Logical instructions to indicate whether the result of that operation was greater than, less than, or equal to zero.

AND MEMORY BYTE

ANMB
d,*m,x

8408



DEFINITION The byte in memory specified by the Effective Byte Address (EBA) is accessed and logically ANDed with the least significant byte (bits 24-31) of the GPR specified by R. The result is transferred to bit positions 24-31 of the GPR specified by R. Bit positions 0-23 of the GPR specified by R remain unchanged.

SUMMARY $(EBL)\&(R_{24-31}) \rightarrow R_{24-31}$

R_{0-23} Unchanged

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{24-31} is greater than zero
CC3: Always zero
CC4: ISI R_{24-31} is equal to zero

EXAMPLE

Memory Location:	00200
Hex Instruction:	84 88 03 73 (R=1, X=0, I=0)
Assembly Language Coding:	ANMB 1,X'373'

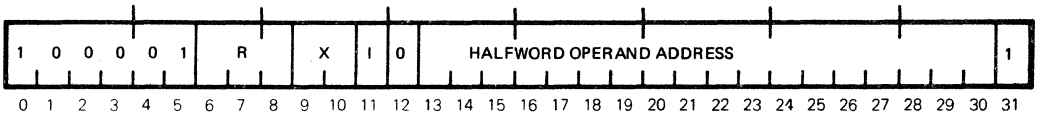
	PSWR	GPR1	Memory Byte 00373
Before Execution	00000200	36AC718F	C7
After Execution	20000204	36AC7187	C7

Note The contents of memory byte 00373 are ANDed with the right-hand byte of GPR1, and the result replaces the byte in GPR1. CC2 is set.

ANMH
d,*m,x

AND MEMORY HALFWORD

8400



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and logically ANDed with the least significant halfword (bits 16-31) of the GPR specified by R. The result is transferred to bit positions 16-31 of the GPR specified by R. Bit positions 0-15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL)\&(R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{16-31} is greater than zero
CC3: Always zero
CC4: ISI R_{16-31} is equal to zero

EXAMPLE

Memory Location: 01000
Hex Instruction: 87 00 12 A3 (R=6, X=0, I=0)
Assembly Language Coding: ANMH 6,X'12A2'

	PSWR	GPR6	Memory Halfword 012A2
Before Execution	40001000	4F638301	70F6
After Execution	08001004	4F630000	70F6

Note

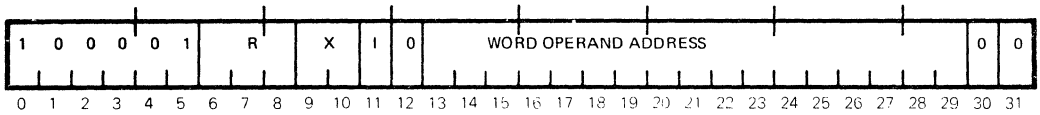
The contents of memory halfword 012A2 are ANDed with the right halfword of GPR6, and the result replaces the halfword in GPR6. CC4 is set.

AND MEMORY WORD

ANMW

d,*m,x

8400



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed and logically ANDed with the word located in the GPR specified by R.

SUMMARY EXPRESSION — (EWL)&(R) → R

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:			00F1C
Hex Instruction:			87 80 0F D0 (R=7, X=0, I=0)
Assembly Language Coding:			ANMW 7,X'FD0'

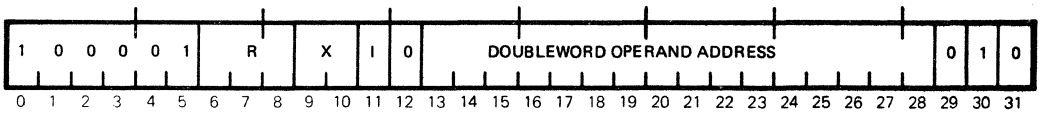
Before Execution	PSWR	GPR7	Memory Word 00FD0
	08000F1C	F0F0F0F0	9ED13854
After Execution	PSWR	GPR7	Memory Word 00FD0
	10000F20	90D03050	9ED13854

Note — The contents of memory word 00FD0 are ANDed with the contents of GPR7, and the result replaces the contents of that register. CC3 is set.

ANMD
d,*m,x

AND MEMORY DOUBLEWORD

8400



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and logically ANDed with the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting doubleword is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL+1)&(R+1) → R+1

(EWL)&(R) → R[4]

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00674
Hex Instruction: 86 00 08 1A (R=4, X=0, I=0)
Assembly Language Coding: ANMD 4,X'818'

Before Execution	PSWR	GPR4	GPR5
	00000674	9045C64A	32B08F00
	Memory Word 00818		Memory Word 0081C
	684A711C		8104A2BC
After Execution	PSWR	GPR4	GPR5
	20000678	00404008	00008200
	Memory Word 00818		Memory Word 0081C
	684A711C		8104A2BC

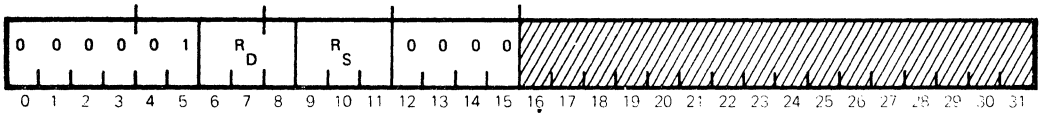
Note

The contents of memory word 00818 are ANDed with the contents of GPR4, and the result replaces the contents of GPR4. The contents of memory word 0081C are ANDed with the contents of GPR5, and the result replaces the contents of GPR5. CC2 is set.

AND REGISTER AND REGISTER

ANR
s,d

0400



DEFINITION The word in the GPR specified by R_D is logically ANDed with the word in the GPR specified by R_S. The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION $(R_S)\&(R_D) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:		03812
Hex Instruction:		04 F0 (R_D=1, R_S=7)
Assembly Language Coding:		ANR 7,1

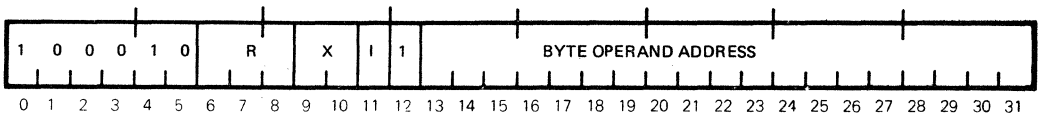
Before Execution	PSWR	GPR1	GPR7
	40003812	AC881101	000FFFFF
After Execution	PSWR	GPR1	GPR7
	20003814	00081101	000FFFFF

Note The contents of GPR1 and GPR7 are ANDed, and the result is transferred to GPR1. CC2 is set.

ORMB
d,*m,x

OR MEMORY BYTE

8808



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and logically ORed with the least significant byte (bits 24-31) of the GPR specified by R. The resulting byte is transferred to bit positions 24-31 of the GPR specified by R. Bit positions 0-23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EBL) v (R_{24-31}) \rightarrow R_{24-31}$

R_{0-23} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00600
Hex Instruction: 88 88 08 A3 (R=1, X=0, I=0)
Assembly Language Coding: ORMB 1,X'8A3'

	PSWR	GPR1	Memory Byte 8A3
Before Execution	00000600	40404040	3C
After Execution	20000604	4040407C	3C

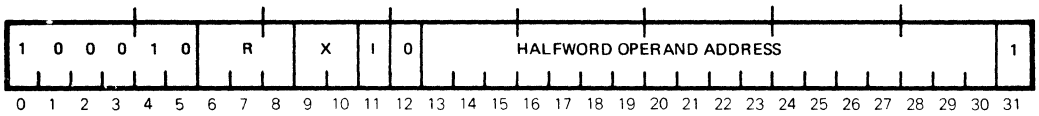
Note

The contents of memory byte 8A3 are logically ORed with the right-hand byte of GPR1, and the result replaces that byte in GPR2. CC2 is set.

OR MEMORY HALFWORD

ORMH
d,*m,x

8800



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and logically ORed with the least significant halfword (bits 16-31) of the GPR specified by R. The resulting halfword is transferred to bit positions 16-31 of the GPR specified by R. Bit positions 0-15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL) v (R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 018AC
Hex Instruction: 8B 00 19 45 (R=6, X=0, I=0)
Assembly Language Coding: ORMH 6,X'1944'

Before Execution	PSWR	GPR6	Memory Halfword 01944
	000018AC	BD71A4C6	45F3
After Execution	PSWR	GPR6	Memory Halfword 01944
	100018B0	BD71E5F7	45F3

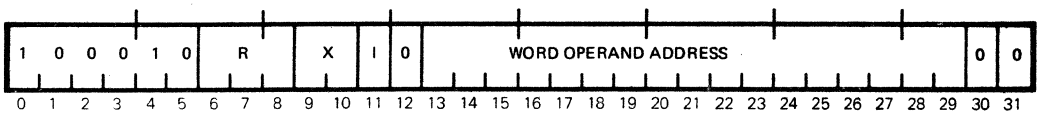
Note

The contents of memory halfword 01944 are ORed with the right halfword from GPR6, and the result replaces that halfword in GPR6. CC3 is set.

ORMW
d,*m,x

OR MEMORY WORD

8800



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed and logically ORed with the word in the GPR specified by R. The result is transferred to the GPR specified by R.

SUMMARY EXPRESSION — (EWL)v(R)→ R

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	05000
Hex Instruction:	89 80 52 0C (R=3, X=0, I=0)
Assembly Language Coding:	ORMW 3,X'520C'

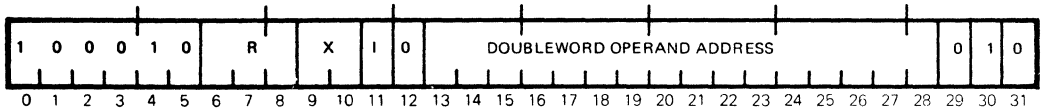
	PSWR	GPR	Memory Word 0520C
Before Execution	40005000	88888888	0EDC4657
After Execution	PSWR	GPR3	Memory Word 0520C
	10005004	8EDCCEDF	0EDC4657

Note — The contents of memory word 0520C are ORed with the contents of GPR3, and the result is transferred to GPR3. CC3 is set.

OR MEMORY DOUBLEWORD

ORMD
d,*m,x

8800



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and logically ORed with the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL+1)v(R+1) → R+1

(EWL)v(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00B68
Hex Instruction: 8B 00 0C 32 (R=6, X=0, I=0)
Assembly Language Coding: ORMD 6,X'C30'

Before Execution

PSWR	GPR6	GPR7
10000B68	002A0031	001D0039

Memory Word 00C30	Memory Word 00C34
18004C00	09002400

After Execution

PSWR	GPR6	GPR7
20000B6C	182A4C31	091D2439

Memory Word 00C30	Memory Word 00C34
18004C00	09002400

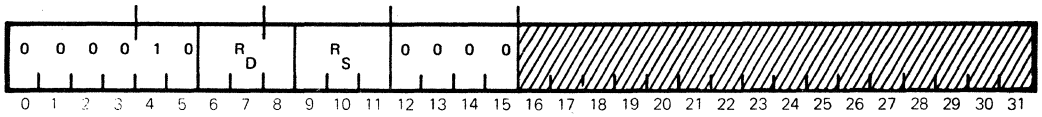
Note

The contents of memory word 00C30 are ORed with the contents of GPR6, and the result is transferred to GPR6. The contents of memory word 00C34 are ORed with the contents of GPR7, and the result is transferred to GPR7. CC2 is set.

ORR
s,d

OR REGISTER AND REGISTER

0800



DEFINITION — The word in the GPR specified by R_D is logically ORed with the word in the GPR specified by R_S. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_S) v (R_D) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	00F8A
Hex Instruction:	08 A0 (R_D=1, R_S=2)
Assembly Language Coding:	ORR 2,1

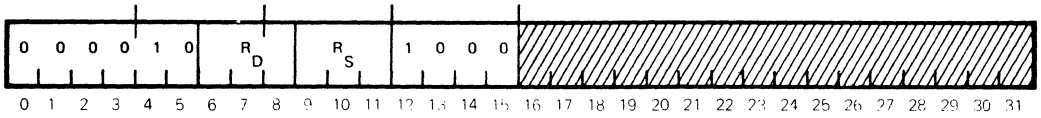
	PSWR	GPR1	GPR2
Before Execution	40000F8A	0001D63F	88880000
After Execution	10000F8C	8889D635	88880000

Note — The contents of GPR1 and GPR2 are ORed, and the result is transferred to GPR1. CC3 is set.

OR REGISTER AND REGISTER MASKED

ORRM
s,d

0808



DEFINITION

The word in the GPR specified by R_D is logically ORed with the word in the GPR specified by R_S. The resulting word is then masked (Logical AND Function) with the contents of the Mask register (R4). The result is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) v (R_D) \& (R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location: 03956
Hex Instruction: 0B 58 (R_D=6, R_S=5)
Assembly Language Coding: ORRM 5,6

	PSWR	GPR4	GPR5	GPR6
Before Execution	08003956	EEEEEEEE	37735814	2561CA95
After Execution	10003958	EEEEEEEE	37735814	2662CA84

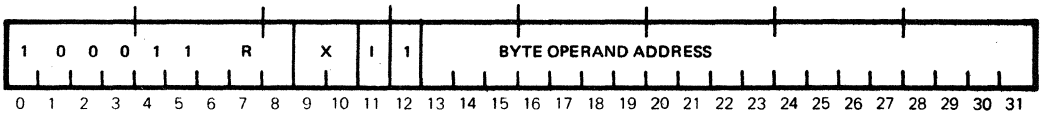
Note

The contents of GPR5 and GPR6 are ORed; the result is ANDed with the contents of GPR4 and transferred to GPR6. CC3 is set.

EOMB
d,*m,x

EXCLUSIVE OR MEMORY BYTE

8C08



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and logically Exclusive ORed with the least significant byte (bits 24-31) of the GPR specified by R. The result is transferred to bit positions 24-31 of the GPR specified by R. Bits 0-23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EBL) \oplus (R_{24-31}) \rightarrow R_{24-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 012F8
Hex Instruction: 8C 08 13 A1 (R=0, X=0, I=0)
Assembly Language Coding: EOMB 0,X'13A1'

Before Execution	PSWR	GPR0	Memory Byte 013A1
	000012F8	D396F458	A9
After Execution	PSWR	GPR0	Memory Byte 013A1
	100012FC	D396F4F1	A9

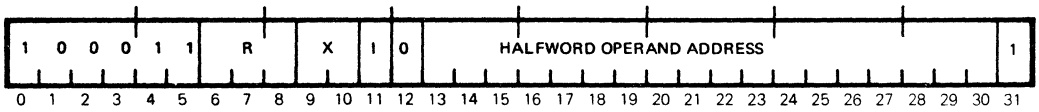
Note

The contents of memory byte 013A1 are Exclusive ORed with the right-hand byte of GPR0; the result replaces that byte in GPR0. CC3 is set.

EXCLUSIVE OR MEMORY HALFWORD

EOMH
d,*m,x

8C00



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and logically Exclusive ORed with the least significant halfword (bits 16-31) of the GPR specified by R. The result is transferred to bit positions 16-31 of the GPR specified by R. Bit positions 0-15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL) \oplus (R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:			00958
Hex Instruction:			8E 80 0A 41 (R=5, X=0, I=0)
Assembly Language Coding:			EOMH 5,X'A40'
Before Execution	PSWR	GPR5	Memory Halfword 00A40
	40000958	96969696	5CAB
After Execution	PSWR	GPR5	Memory Halfword 00A40
	1000095C	9696CA3D	5CAB

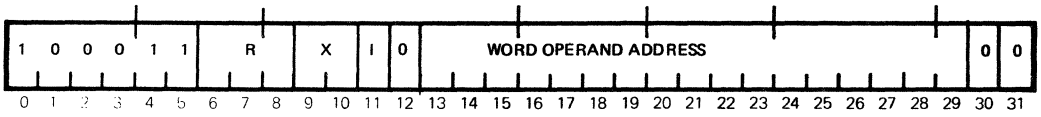
Note

The contents of memory halfword 00A40 are Exclusive ORed with the right halfword of GPR5, and the result replaces that halfword in GPR5. CC3 is set.

EOMW
d,*m,x

EXCLUSIVE OR MEMORY WORD

8C00



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed and logically Exclusive ORed with the word in the GPR specified by R. The result is transferred to the GPR specified by R.

SUMMARY EXPRESSION — (EWL) $\oplus$ (R) → R

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	185BC
Hex Instruction:	8F 81 86 94 (R=7, X=0, I=0)
Assembly Language Coding:	EDMW 7,X'18694'

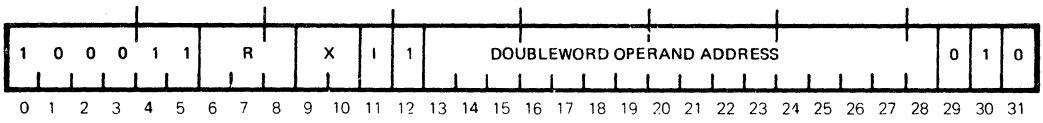
	PSWR	GPR7	Memory Word 18694
Before Execution	010185BC	13579BDF	22222222
After Execution	200185C0	3175B9FD	22222222

Note — The contents of memory word 18694 are Exclusive ORed with the contents of GPR7. The result replaces the contents of GPR7. CC2 is set.

EXCLUSIVE OR MEMORY DOUBLEWORD

EOMD
d,*m,x

8C00



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and logically Exclusive ORed with the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL+1) ⊕ (R+1) → R+1

(EWL) ⊕ (R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location:			00448
Hex Instruction:			8F 00 05 3A (R=6, X=0, I=0)
Assembly Language Coding:			EOMD 6,X'538'
Before Execution	PSWR	GPR6	GPR7
	00000448	00FFFF00	00FFF000
	Memory Word 00538		Memory Word 0053C
	482144C0		2881433A
After Execution	PSWR	GPR6	GPR7
	2000044C	48DEBBC0	287EB33A
	Memory Word 00538		Memory Word 0053C
	482144C0		2881433A

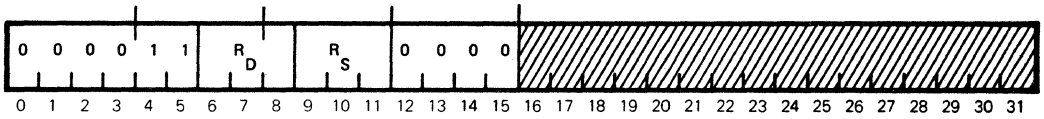
Note

The contents of memory word 00538 and GPR6 are Exclusive ORed and the result is transferred to GPR6. The contents of memory word 0053C and GPR7 are Exclusive ORed and the result is transferred to GPR7. CC2 is set.

EOR
s,d

EXCLUSIVE OR REGISTER AND REGISTER

0C00



DEFINITION — The word in the GPR specified by R_D is logically Exclusive ORed with the word in the GPR specified by R_S. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_S) \oplus (R_D) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	0139E
Hex Instruction:	OF E0 (R_D=7, R_S=6)
Assembly Language Coding:	EOR 6,7

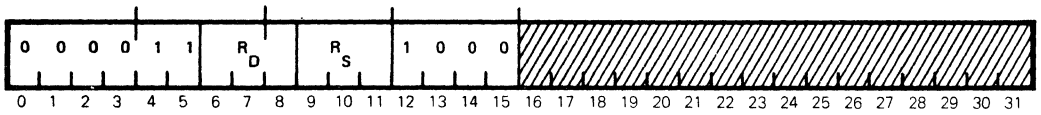
	PSWR	GPR6	GPR7
Before Execution	0100139E	33333333	55555555
After Execution	200013A0	33333333	66666666

Note — The contents of GPR6 and GPR7 are Exclusive ORed, and the result is transferred to GPR7. CC2 is set.

EXCLUSIVE OR REGISTER AND REGISTER MASKED

EORM
s,d

0C08



DEFINITION

The word in the GPR specified by R_D is logically Exclusive ORed with the word in the GPR specified by R_S. The resulting word is then masked (Logical AND Function) with the contents of the Mask register (R4). The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \oplus (R_D)$ & $(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location: 25A32
Hex Instruction: OF E8 (R_D=7, R_S=6)
Assembly Language Coding: EORM 6,7

	PSWR	GPR4	GPR6	GPR7
Before Execution	00025A32	00FEDF00	9725A2C8	6C248237
After Execution	08025A34	00FEDF00	9725A2C8	00000000

Note

The contents of GPR6 and GPR7 are Exclusive ORed. The result is ANDed with the contents of GPR4 and transferred to GPR7. CC4 is set.

SHIFT OPERATION INSTRUCTIONS

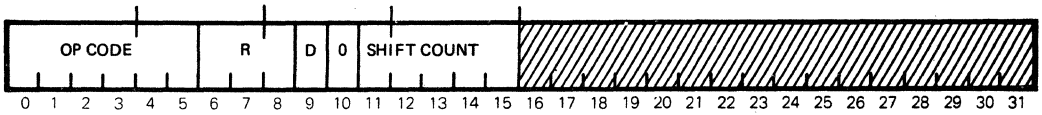
GENERAL DESCRIPTION

This group of instructions provides the capability to perform Arithmetic, Logical, and Circular Left or Right shift operations on the contents of words or doublewords in General Purpose Registers. Provisions have also been made to allow Normalize operations to be performed on the contents of words or doublewords in General Purpose Registers.

INSTRUCTION FORMATS

The following two instruction formats are used by the Shift instruction group:

SHIFT INFORMATION



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0-7).

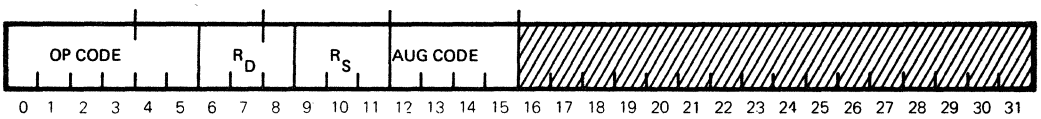
Bit 9 designates direction.

D=1 designates shift left
D=0 designates shift right

Bit 10 unassigned.

Bits 11-15 define the number of shifts to be made.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

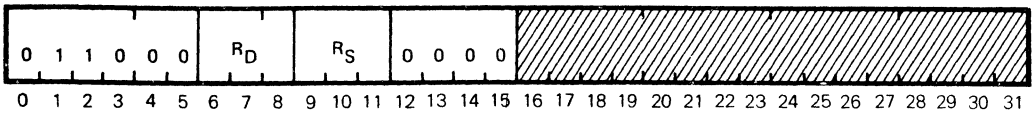
CONDITION CODE UTILIZATION

Most Shift instructions leave the Condition Code unchanged.

NORMALIZE

NOR
d,s

6000



DEFINITION

The word in the GPR specified by R_S is shifted left, 4 bit positions at a time, until the contents are normalized for the base 16 exponent. The contents of R_S are less than one or equal to or greater than 1/16 ($1 > (R_S) \geq 1/16$.) The exponent is set to 40_{16} and is decremented once for each group of 4 shifts performed. When normalization is complete, the exponent is stored in bit positions 25-31 of the GPR specified by R_D. Bit positions 0-24 of the GPR specified by R_S are cleared to zeros. If the contents of the GPR specified by R_S are equal to zero, the exponent stored in bit positions 25-31 of the GPR specified by R_D will equal zero and no shifting will be performed.

Note

The normalized result must be converted to the format defined on page 6-171 prior to use by the floating-point arithmetic unit or standard FORTRAN floating-point subroutines. In addition, a test must be made for minus full scale (1XXX XXXX 0000 0000 --- 0000) and a conversion made to (1YYY YYYY 1111 0000 --- 0000), where YYY YYYY is one less than XXX XXXX.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00D32
Hex Instruction: 63 10 (R_S=6, R_D=1)
Assembly Language Coding: NOR 6,1

	PSWR	GPR1	GPR6
Before Execution	20000D32	12345678	0002E915
After Execution	20000D34	0000003D	2E915000

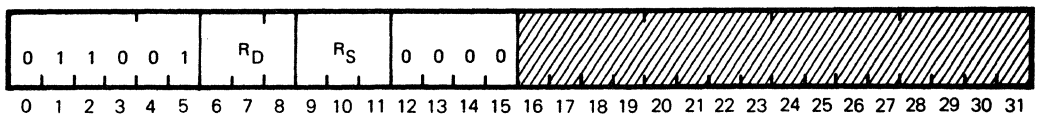
Note

The content of GPR6 is normalized by three left shifts of four bits each. The exponent is determined by decrementing 40_H once for each shift and transferred to GPR1.

NORD
s,d

NORMALIZE DOUBLE

6400



DEFINITION

The doubleword in the GPR specified by R_S and R_S+1 is shifted left, 4 bit positions at a time, until the contents are normalized for the base 16 exponent ($1 > (R_S,R_S+1) \geq 1/16$). The contents of R_S and R_S+1 are less than one or equal to or greater than 1/16. R_S+1 is the GPR one greater than specified by R_S. The exponent of the doubleword is set to 40_{16} and is decremented once for each group of four shifts performed. When normalization is complete, the exponent is stored in bit positions 25-31 of the GPR specified by R_D. Bit positions 0-24 of the GPR specified by R_D are cleared to zeros. If the contents of the doubleword specified by R_S and R_S+1 are equal to zero, the exponent stored in bit positions 25-31 of the GPR specified by R_D will equal zero, and no shifting will be performed.

Note

The normalized result must be converted to the format defined on page 6-171 prior to use by the floating-point arithmetic unit or standard FORTRAN floating-point subroutines. In addition, a test must be made for minus full scale (1XXX XXXX 0000 0000 --- 0000) and a conversion made to (1YYY YYYY 1111 0000 --- 0000), where YYY YYYY is one less than XXX XXXX.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	0046E
Hex Instruction:	67 10 (R_S=6, R_D=1)
Assembly Language Coding:	NORD 6,1

	PSWR	GPR1	GPR6	GPR7
Before Execution	1000046E	9ABCDEF0	FFFFFFFF	FF3AD915
After Execution	10000470	00000037	F3AD9150	00000000

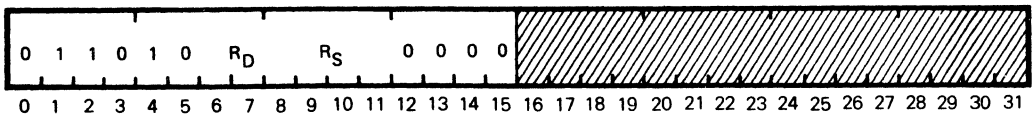
Note

The doubleword obtained from the contents of GPR6 and GPR7 is normalized by nine left shifts of four bit positions each. The result is returned to GPR6 and GPR7, and the exponent (40_H-9) is transferred to GPR1.

SHIFT AND COUNT ZEROS

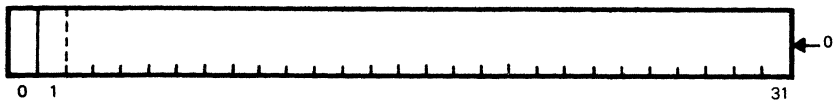
SCZ
d,s

6800



DEFINITION

The word in the GPR specified by R_S is shifted left, one bit position at a time, until the sign (bit 0) changes from zero to one. The contents are then shifted left one more bit position, and the total number of shifts minus one is placed in bit positions 27-31 of the GPR specified by R_D. Bit positions 0-26 of the GPR specified by R_D are set to zeros. The shift count specifies the most significant bit position (0-31) of R_S that was equal to one.



NOTES

1. If the contents of the GPR specified by R_S are equal to zero, the shift count placed in bit positions 27-31 of the GPR specified by R_D is zero, and Condition Code bit 4 is set to one.

2. If the sign (bit 0) of the GPR specified by R_S is equal to one, the shift count placed in bit positions 27-31 of the GPR specified by R_D is zero, and Condition Code bit 4 is set to zero.

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: ISI R_S 0-31 is equal to zero

EXAMPLE

Memory Location: 0399E
Hex Instruction: 6A 20 (R_S=4, R_D=2)
Assembly Language Coding: SCZ 2,N

	PSWR	GPR2	GPR4
Before Execution	2000399E	12345678	00300611
After Execution	000039A0	0000000A	80308800

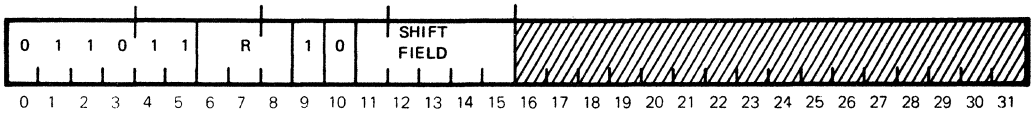
Note

The content of GPR4 are left shifted 10 bits when bit 0 is equal to one. The contents are then shifted one more bit position, and the zero count of 10 (A_H) is transferred to GPR2.

SLA
d,v

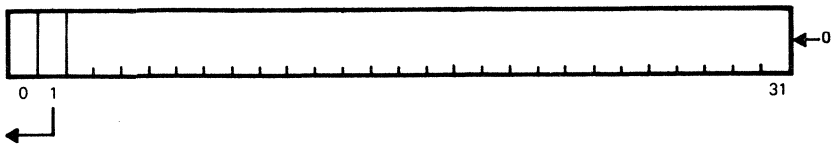
SHIFT LEFT ARITHMETIC

6C40



DEFINITION

Bit positions 1-31 of the GPR specified by R are shifted left the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. Bit position 0 (sign bit) of the GPR specified by R remains unchanged. Condition Code bit 1 is set to one if any bit shifted out of position 1 differs from the sign bit.



CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: Always zero
CC3: Always zero
CC4: Always zero

EXAMPLE

Memory Location: 00106
Hex Instruction: 6F 4C (R=6, Shift Count=12_{10})
Assembly Language Coding: SLA 6,12

	PSWR	GPR6
Before Execution	10000106	000013AD
After Execution	00000108	013AD000

Note

The contents of GPR6 are left shifted 12 bit positions and then zero-filled from the right. The result is transferred to GPR6.

EXAMPLE 2

Memory Location: 00106
Hex Instruction: 6F 4C (R=6, Shift Count=12_{10})
Assembly Language Coding: SLA 6,12

	PSWR	GPR6
Before Execution	10000106	001FAD58
After Execution	40000108	7AD58000

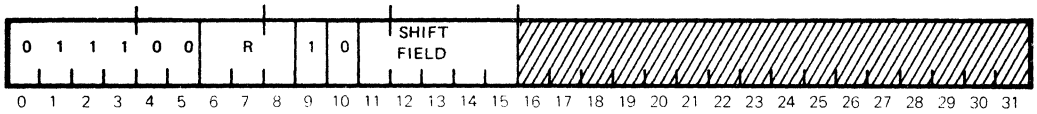
Note

Overflow occurs and is indicated by CC1.

SHIFT LEFT LOGICAL

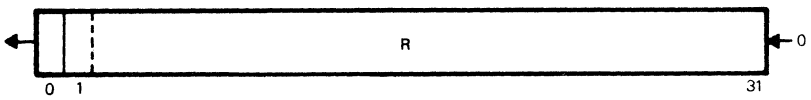
SLL
d,v

7040



DEFINITION

The word in the GPR specified by R is shifted left the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	00812
Hex Instruction:	73 D4 (R=7, Shift Count=20_{10})
Assembly Language Coding:	SLL 7,20

	PSWR	GPR7
Before Execution	A0000812	12345678
After Execution	A0000814	67800000

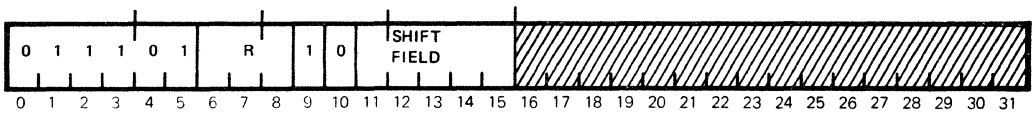
Note

The contents of GPR7 are left-shifted 20 bits and replaced.

SLC
d,v

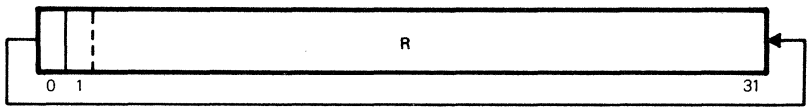
SHIFT LEFT CIRCULAR

7440



DEFINITION

The word in the GPR specified by R is shifted left the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. Bits shifted out of bit position 0 are shifted into bit position 31.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 001FA
Hex Instruction: 77 CF (R=7, Shift Field=16_{10})
Assembly Language Coding: SLC 7,16

	PSWR	GPR7
Before Execution	000001FA	12345678
After Execution	000001FC	56781234

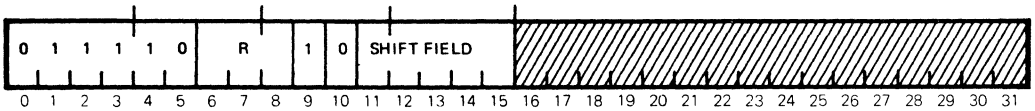
Note

The contents of GPR7 are shifted left circular for 16 bit positions.

SHIFT LEFT ARITHMETIC DOUBLE

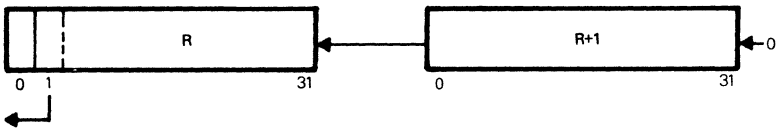
SLAD
d,v

7840



DEFINITION

The doubleword in the GPR specified by R and R+1 is shifted left the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. R+1 is the GPR one greater than specified by R. The sign (bit 0) of the GPR specified by R remains unchanged. Condition Code bit 1 is set to One if any bit shifted out of position 1 differs from the sign bit, position 0.



CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: Always zero
CC3: Always zero
CC4: Always zero

EXAMPLE

Memory Location:	02DF6
Hex Instruction:	7A 58 (R=4, Shift Field=24_{10})
Assembly Language Coding:	SLAD 4,24

	PSWR	GPR4	GPR5
Before Execution	80002DF6	FFFFFFA3	9A178802
After Execution	80002DF8	A39A1788	02000000

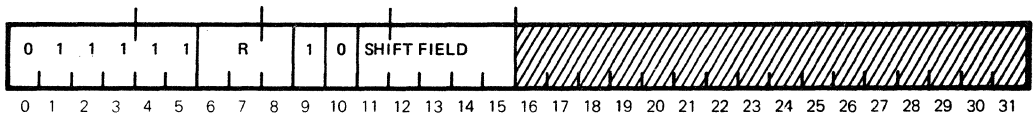
Note

The doubleword obtained from the contents of GPR4 and GPR5 is left-shifted 24 bit positions, then zero-filled from the right. The result is returned to GPR4 and GPR5.

SLLD
d,v

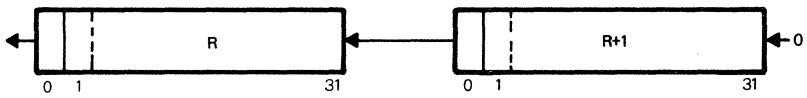
SHIFT LEFT LOGICAL DOUBLE

7C40



DEFINITION

The doubleword in the GPR specified by R and R+1 is shifted left the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. R+1 is the GPR one greater than specified by R.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:		001FE
Hex Instruction:		7F 58 (R=6, Shift Field=24)
Assembly Language Coding:		SLLD 6,24

Before Execution	PSWR	GPR6	GPR7
	100001FE	01234567	89ABCDEF
After Execution	PSWR	GPR6	GPR7
	10000200	6789ABCD	EF000000

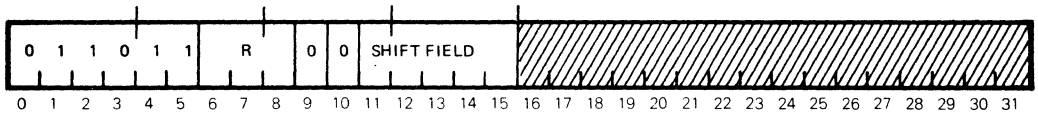
Note

The doubleword obtained from GPR6 and GPR7 is left-shifted 24 bit positions, then zero-filled from the right. The result is returned to GPR6 and GPR7.

SHIFT RIGHT ARITHMETIC

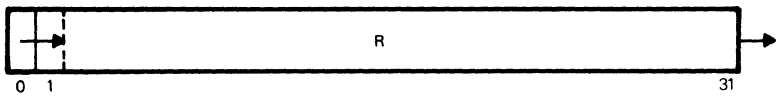
SRA
d,v

6C00



DEFINITION

The word in the GPR specified by R is shifted right the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. Bit position 0 (sign bit) is shifted into bit position 1 on each shift. The sign bit remains unchanged.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	00372
Hex Instruction:	6D 0A (R=4, Shift Field=10_{10})
Assembly Language Coding:	SRA 4,10

	PSWR	GPR4
Before Execution	10000372	B69825F1
After Execution	PSWR	GPR4
	10000374	FFEDA609

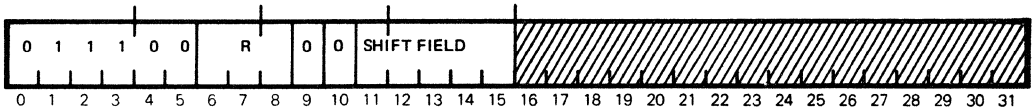
Note

The contents of GPR4 are shifted right 10 bit positions. Since that value is negative, a one is entered into bit position 1 with each shift.

SRL
d,v

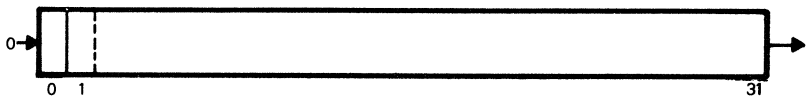
SHIFT RIGHT LOGICAL

7000



DEFINITION

The word in the GPR specified by R is shifted right the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	00372
Hex Instruction:	72 0A (R=4, Shift Field=10_{10})
Assembly Language Coding:	SRL 4,10

	PSWR	GPR4
Before Execution	10000372	B69825F1
After Execution	PSWR	GPR4
	10000374	002DA609

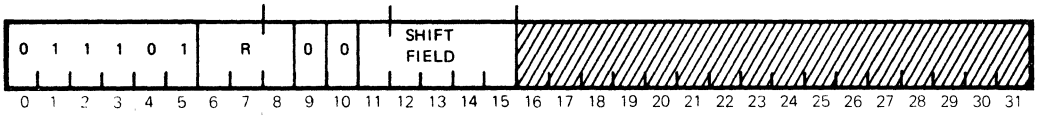
Note

The content of GPR4 is shifted right 10 bit positions, then zero-filled from the left.

SHIFT RIGHT CIRCULAR

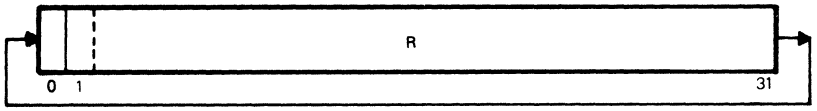
SRC
d,v

7400



DEFINITION

The word in the GPR specified by R is shifted right the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. Bits shifted out of bit position 31 are shifted into bit position 0.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00372
Hex Instruction: 76 0C (R=4, Shift Field=12_{10})
Assembly Language Coding: SRC 4,12

	PSWR	GPR4
Before Execution	20000372	01234567
After Execution	PSWR	GPR4
	20000374	56701234

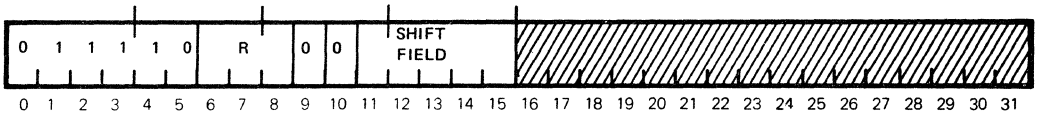
Note

The contents of GPR4 are shifted right circular 12 bit positions and replaced in GPR4.

SRAD
d,v

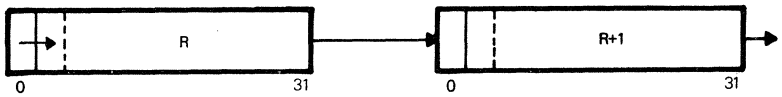
SHIFT RIGHT ARITHMETIC DOUBLE

7800



DEFINITION

The doubleword in the GPR specified by R and R+1 is shifted right the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. R+1 is the GPR one greater than specified by R. The sign (bit 0) of the GPR specified by R remains unchanged. Bit position 0 (sign bit) is shifted into bit position 1 with each shift.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:			02B46
Hex Instruction:			7B 18 (R=6, Shift Field=24_{10})
Assembly Language Coding:			SRAD 6,24

Before Execution	PSWR	GPR6	GPR7
	20002B46	8E2A379B	58C1964D
After Execution	PSWR	GPR6	GPR7
	20002B48	FFFFFF8E	2A379B58

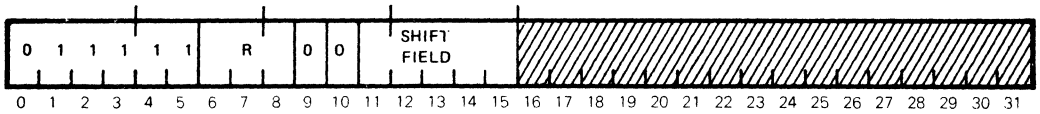
Note

The doubleword obtained from the contents of GPR6 and GPR7 is shifted right 24 bit positions, with the sign extended 24 bits from the left. The result is transferred to GPR6 and GPR7.

SHIFT RIGHT LOGICAL DOUBLE

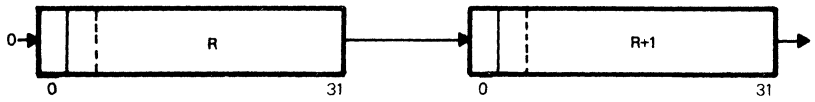
SRLD
d,v

7C00



DEFINITION

The doubleword in the GPR specified by R and R+1 is shifted right the number of bit positions specified by the shift field (bits 11-15) in the Instruction Word. R+1 is the GPR one greater than specified by R.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	02B46
Hex Instruction:	7F 18 (R=6, Shift Field=24_{10})
Assembly Language Coding:	SRLD 6,24

	PSWR	GPR6	GPR7
Before Execution	20002B46	8E2A379B	58C1964D
After Execution	20002B48	0000008E	2A379B58

Note

The doubleword obtained from the contents of GPR6 and GPR7 is shifted right 24 bit positions, then zero-filled from the left. The result is transferred to GPR6 and GPR7.

BIT MANIPULATION INSTRUCTIONS

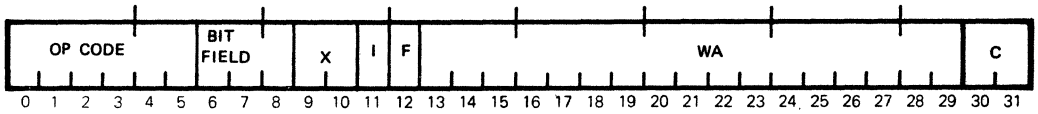
GENERAL DESCRIPTION

The Bit Manipulation instruction group provides the capability to set, read, or add a bit to a specified bit location within a specified byte of a memory location or General Purpose Register. Provisions have also been made to test a bit in memory or a General Purpose Register by transferring the contents of that bit position to the Condition Code register.

INSTRUCTION FORMATS

The Bit Manipulation instruction group uses the following two instruction formats:

MEMORY REFERENCE



Bits 0-5 define the Operation Code.

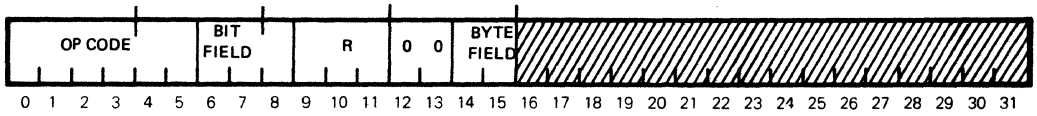
Bits 6-8 specify a bit (0-7).

Bits 9-10 designate one of three index registers.

Bit 11 indicates whether an indirect addresssing operation is to be performed.

Bits 12-31 specify the address of the operand when the X and I fields are equal to zero.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 specify a bit (0-7).

Bits 9-11 designate a General Purpose Register address (0-7).

Bits 12-13 unassigned.

Bits 14-15 specify a byte (0-3).

CONDITION CODE UTILIZATION

A Condition Code is set during execution of Set Bit, Zero Bit, and Test Bit operations, if the bit on which the operation is being performed is equal to one. During Add Bit operations, a Condition Code is set to indicate whether the execution of the instruction caused a result greater than zero, less than zero, equal to zero, or an arithmetic exception.

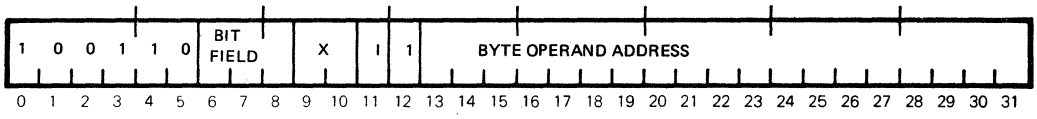
INTERPROCESSOR SEMAPHORES

When two processors share memory and other resources, a simple positive method must be provided for dynamically reserving/releasing shared memory pages and the other shared resources. The Set Bit in Memory, Zero Bit in Memory, or Add Bit in Memory instructions (SBM, ZBM) are used for this purpose. If both processors attempt to set (or zero) the same semaphore bit at the same time, one processor will actually access the memory location before the other processor by virtue of the shared memory bus design. The first processor to access the bit will copy the previous contents of the bit into its Condition Code register before setting (or clearing) the bit. On the very next memory cycle, the other processor will copy the state of the bit as set by the first processor into its Condition Code register and then set (or clear) the bit again. Both processors then execute Branch on Condition Code instructions to test the status of the bit prior to changing it. The first processor will find the bit previously not set (or set), indicating that it was able to reserve the resource which the user has associated with the bit. The second processor will find the bit already set (or not set), indicating that the resource is currently reserved by the other processor and that subsequent attempts should be made.

SBM
c,*m,x

SET BIT IN MEMORY

9808



DEFINITION — The byte in memory specified by the Effective Byte Address (EBA) is accessed, and the specified bit (bit field) within the byte set to one. All other bits within the byte remain unchanged. The resulting byte is replaced in the location specified by the EBA. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the original status of the specified bit of the byte specified by the EBA is transferred to CC1.

NOTE — Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any 4 bits in memory or the GPR's can be stored in the Condition Code register for a combined Conditional Branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(EBL_{SBL}) → CC1
1 → EBL_{SBL}

CONDITION CODE RESULTS

CC1: ISI EBL_{SBL} is equal to one
CC2: ISI CC1 was one
CC3: ISI CC2 was one
CC4: ISI CC3 was one

EXAMPLE

Memory Location: 01000
Hex Instruction: 98 88 14 03 (bit field = 1)
Assembly Language Coding: SBM 1,X'1403'

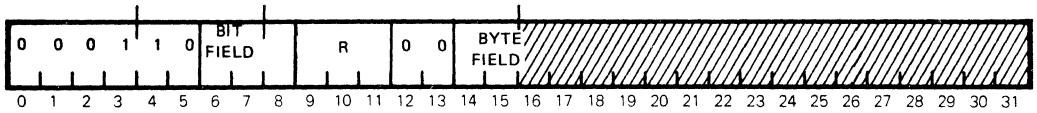
	PSWR	Memory Byte 01403
Before Execution	20001000	1A
After Execution	PSWR	Memory Byte 01403
	10001004	5A

Note — Bit 1 of memory byte 01403 is set to one.

SET BIT IN REGISTER

SBR
d,b

1800



DEFINITION — The specified bit (bit field) of the specified byte (byte field) in the GPR specified by R is set to one. All other bits within the GPR specified by R remain unchanged. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the original status of the specified bit in register R is transferred to CC1.

NOTE — Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPR's can be stored in the Condition Code register for a combined Conditional Branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(R_{SBL}) \rightarrow CC1$
$1 \rightarrow EBL_{SBL}$

CONDITION CODE — CC1: ISI R_{SBL} is equal to one

RESULTS
CC2: ISI CC1 was one
CC3: ISI CC2 was one
CC4: ISI CC3 was one

EXAMPLE

Memory Location	01002
Hex Instruction:	XXXX1B 82 (bit field=7, R=0, byte field=2)
Assembly Language Coding:	SBR 0,2

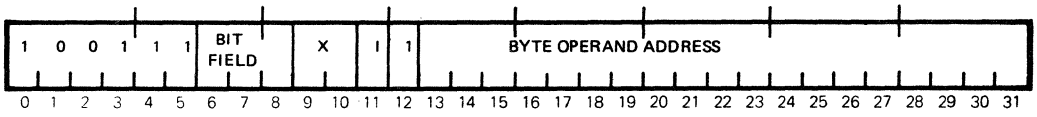
	PSWR	GPR0
Before Execution	10001002	0374B891
After Execution	08001004	0374B991

Note — Bit 23 of GPR0 is set to one.

ZBM
c,*m,x

ZERO BIT IN MEMORY

9C08



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and the specified bit (bit field) within the byte is set to zero. All other bits within the byte remain unchanged. The resulting byte is replaced in the location specified by the EBA. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2 and the original status of the specified bit of the byte specified by the EBA is transferred to CC1.

NOTE

Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPR's can be stored in the Condition Code register for a combined Conditional Branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(EBL_{SBL}) → CC1
0 → EBL_{SBL}

CONDITION CODE RESULTS

CC1: ISI EBL_{SBL} is equal to one
CC2: ISI CC1 was one
CC3: ISI CC2 was one
CC4: ISI CC3 was one

EXAMPLE

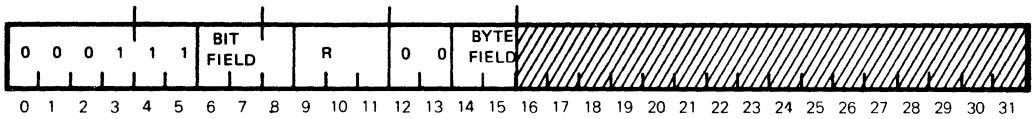
Memory Location: 1F684
Hex Instruction: 9E 8A 01 22 (bit field=5)
Assembly Language Coding: ZMB 5,X'20122'

	PSWR	Memory Byte 20122
Before Execution	1001F684	34
After Execution	4801F688	30

ZERO BIT IN REGISTER

ZBR
d,b

1C00



DEFINITION — The specified bit (bit field) of the specified byte (byte field) in the GPR specified by R is set to zero. All other bits within the GPR specified by R remain unchanged. Condition Code bit 3(CC3) is transferred to CC2, and the original status of the specified bit of the specified byte in register R is transferred to CC1.

NOTE — Since the contents of the Condition Code register are shifted to the next highest position before the bit is loaded into CC1, any four bits in memory or the GPR's can be stored in the Condition Code register for a combined Conditional Branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(R_{SBL}) → CC1
0 → EBL_{EBL}

CONDITION CODE

CC1: ISI R_{SBL} is equal to one
CC2: ISI CC1 was one
CC3: ISI CC2 was one
CC4: ISI CC3 was one

EXAMPLE

Memory Location:	00C56
Hex Instruction:	1C51 (bit field=0, R=5, byte field=1)
Assembly Language Coding:	ZBR 5,8

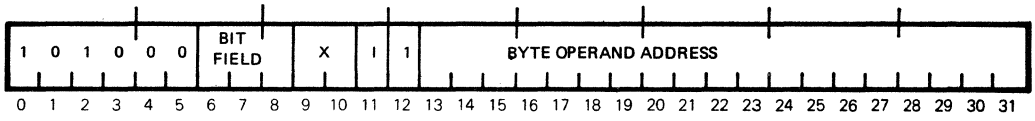
Before Execution	PSWR	GPR5
	10000C56	76A43B19
After Execution	PSWR	GPR5
	48000C58	76243B19

Note — Bit 8 of GPR5 is cleared to zero. CC4 is set.

ABM
c,*m,x

ADD BIT IN MEMORY

A008



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and one is added to the bit position specified by the bit field. The addition is performed on the entire memory word containing the byte specified by the EBA. Therefore, a carry may be propagated left to the sign bit. The resulting word is transferred to the memory word location containing the byte specified by the EBA.

SUMMARY EXPRESSION

(EBL)+1SBL → EBL

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI (EWL) is greater than zero
CC3: ISI (EWL) is less than zero
CC4: ISI (EWL) is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: A2 08 31 92 (bit field=4, X=0, I=0)
Assembly Language Coding: ABM 4,X'3192'

	PSWR	Memory Word 03190
Before Execution	00003000	51A3F926
After Execution	20003004	51A40126

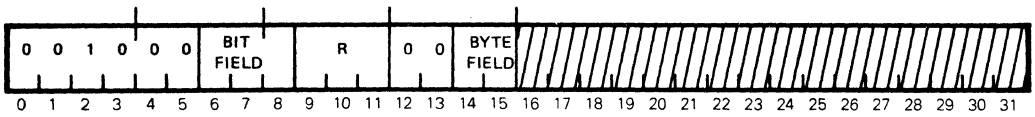
Note

A one is added to bit position 20_{10} of memory word 03190 (byte 2, bit 4) which propagates a carry left to bit position 13_{10}. The result is returned to memory word 03190. CC2 is set.

ADD BIT IN REGISTER

ABR
d,b

2000



DEFINITION

A one is added to the specified bit (bit field) of the specified byte (byte field) in the GPR specified by R. The addition is performed on the entire word of the GPR specified by R. Therefore, a carry may be propagated left to the sign bit. The result is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)+1_{SBL} \rightarrow R$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 0184E
Hex Instruction: 21 61 (bit field=2, R=6, byte field=1)
Assembly Language Coding: ABR 6,10

	PSWR	GPR6
Before Execution	0800184E	3BE9AC48
After Execution	PSWR	GPR6
	20001850	3C09AC48

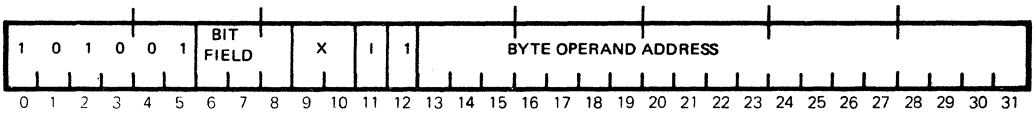
Note

A One is added to bit position 10_{10} of GPR6, and the result is replaced in GPR6. CC2 is set.

TBM
c,*m,x

TEST BIT IN MEMORY

A408



DEFINITION — The specified bit in memory is transferred to the Condition Code register. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the specified bit (bit field) of the byte specified by the Effective Byte Address (EBA) is transferred to CC1.

NOTE — Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPR's can be stored in the Condition Code register for a combined Conditional Branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(EBL_{SBL}) → CC1

CONDITION CODE RESULTS

CC1: ISI R_{SBL} is equal to one
CC2: ISI CC1 was equal to one
CC3: ISI CC2 was equal to one
CC4: ISI CC3 was equal to one

EXAMPLE

Memory Location: 05A38
Hex Instruction: A6 08 5B 21 (bit field=4, X=0, I=0)
Assembly Language Coding: TBM 4,X'5B21'

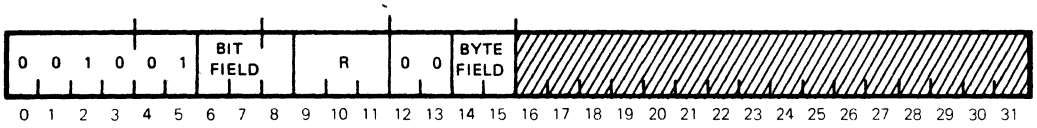
Before Execution	PSWR	Memory Byte 05B21
	10005A38	29
After Execution	PSWR	Memory Byte 05B21
	48005A3C	29

Note — Bit 4 of memory byte 05B21 is transferred to CC1. CC3 is transferred to CC4.

TEST BIT IN REGISTER

TBR
d,b

2400



DEFINITION — The specified bit in the GPR specified by R is transferred to the Condition Code register. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the specified bit (bit field) of the specified byte (byte field) in the GPR specified by R is transferred to CC1.

NOTE — Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPR's can be stored in the Condition Code register for a combined Conditional Branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(R_{SBL}) → CC1

CONDITION CODE RESULTS

CC1: ISI R_{SBL} was equal to one
CC2: ISI CC1 was equal to one
CC3: ISI CC2 was equal to one
CC4: ISI CC3 was equal to one

EXAMPLE

Memory Location	01982
Hex Instruction:	25 D3 (bit field=3, R=5, byte field=3)
Assembly Language Coding:	TBR 5,27

	PSWR	GPR5
Before Execution	18001982	81A2C64D
After Execution	PSWR	GPR5
	08001984	81A2C64D

Note — CC2 through CC4 are right-shifted one bit position. CC1 is cleared to zero since bit 27_{10} of GPR5 is zero.

FIXED-POINT ARITHMETIC INSTRUCTIONS

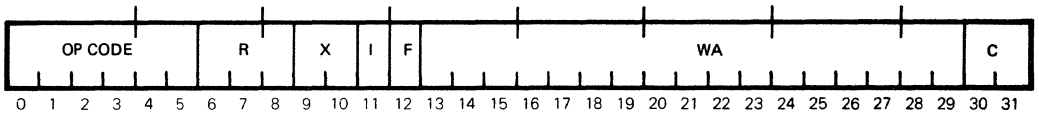
GENERAL DESCRIPTION

The Fixed-Point Arithmetic group is used to perform addition, subtraction, multiplication, division, and sign control functions on bytes, halfwords, words, and doublewords in memory and General Purpose Registers. Provisions have also been made to allow the result of a register-to-register addition or subtraction to be masked before final storage.

INSTRUCTION FORMATS

The Fixed-Point Arithmetic instructions use the following three instruction formats:

MEMORY REFERENCE



Bits 0-5 define the Operation Code.

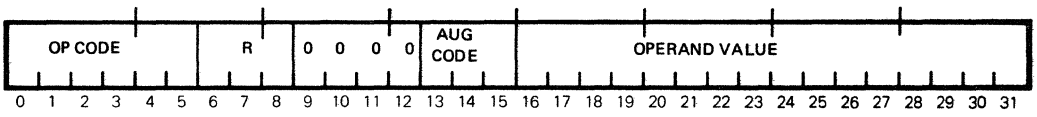
Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-10 designate one of three index registers.

Bit 11 designates whether an Indirect Addressing operation is to be performed.

Bits 12-31 specify the address of the operand when the X and I fields are equal to zero.

IMMEDIATE



Bits 0-5 define the Operation Code.

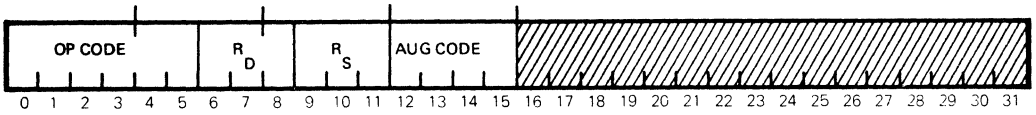
Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-12 unassigned.

Bits 13-15 define Augmenting Operation Code.

Bits 16-31 contain the 16-bit operand value.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

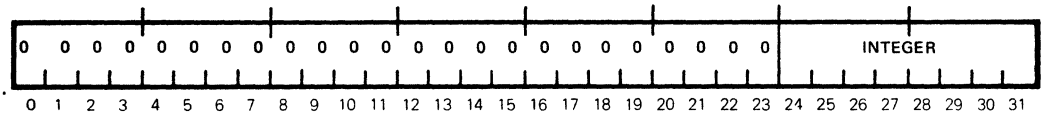
Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

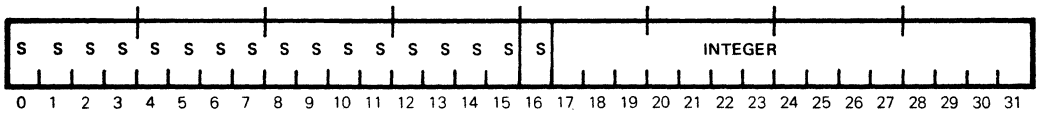
Data Formats

The Fixed-Point Arithmetic instructions use the following data formats:

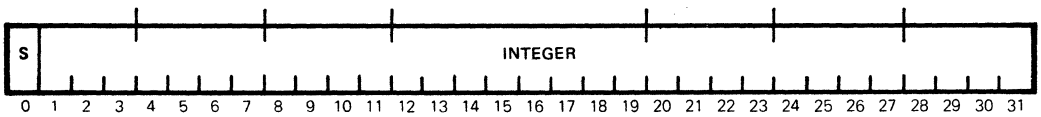
Byte



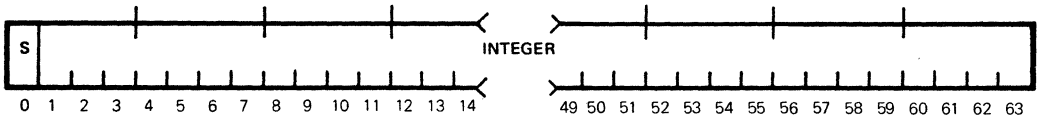
Halfword (Sign Extended)



Word



Doubleword

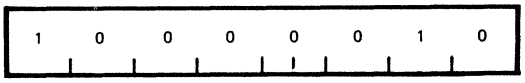


CONDITION CODE UTILIZATION

Execution of most Fixed-Point Arithmetic instructions causes a Condition Code to be set to indicate whether the result of the operation was greater than, less than, or equal to zero. Arithmetic exceptions produced by an arithmetic operation are also reflected by the Condition Code results.

TREATMENT OF SIGNED NUMBERS

To perform logical operations, the hardware interprets operands as logical words. For fixed-point arithmetic operations, operands are treated as unsigned numbers. Logical and arithmetic operations can be performed on any of the data types available in the SEL 32 Series Computer bytes, 16-bit halfwords, 32-bit words, and 64-bit doublewords. A program executing on the SEL 32 Series Computer however, can interpret any of the available data types as a two's complement notation number. It is a property of two's complement arithmetic that operations on signed numbers using two's complement conversions are identical to operations on unsigned numbers; in other words, the hardware treats the sign as the most significant magnitude bit. Consider a General Purpose Register that contains:



As an unsigned number, this would be equivalent to:

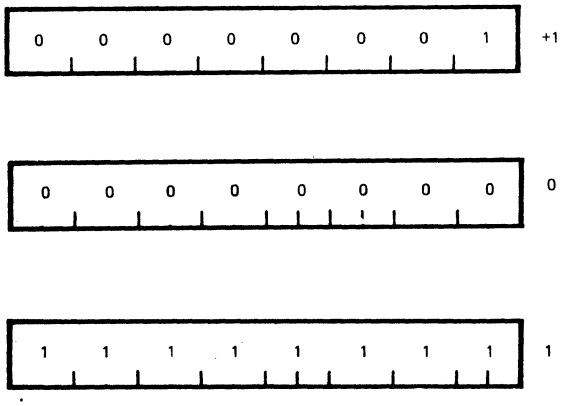
$$82_{16} = 130_{10}$$

Interpreted as a signed number using two's complement notation, it would be:

$$7E_{16} = 126_{10}$$

It makes no difference as to how the programmer interprets data as far as processor operation is concerned. However, the programmer is aided in the use of two's complement notation by the Condition Code (CC) bits of the Program Status Word (PSW), which are generally set based on two's complement notation.

Numbers in two's complement notation are symmetrical in magnitude around a zero representation, so all even numbers, both positive and negative, will end in zero, and all odd numbers will end in one (binary word containing all one's represents minus one).



If one's complement notation was used for negative numbers, a negative number could be read by attaching significance to the zeros instead of the one's.

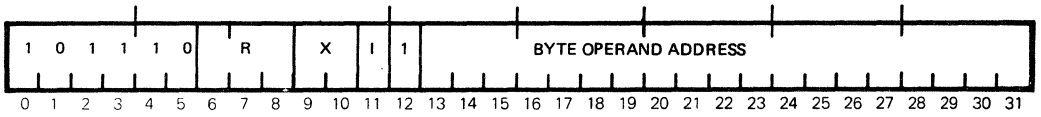
In two's complement notation, each number is one greater than the complement of the positive number of the same magnitude, so a negative number can be read by attaching significance to the right-hand one and to the zeros to the left of it. (The negative number of the largest magnitude has a one only in the sign position.) Assuming a binary integer, one's may be discarded at the left in a negative integer in the same way that leading zeros may be dropped from a positive integer.

Associated with the Arithmetic/Logic Unit is a 4-bit Condition Code register which forms the CC portion of the PSW. These CC bits are altered during all Arithmetic/Logical operations and data transfers. The CC bits indicate such conditions as arithmetic exception, overflow, zero, and positive or negative magnitude.

ADMB
d,*m,x

ADD MEMORY BYTE

B808



DEFINITION — The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros are appended to the most significant end to form a word. This word is algebraically added to the contents of the GPR specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION — 0_{0-23},(EBL)+(R) → R

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	00800
Hex Instruction:	BA 08 09 15 (R=4, X=0, I=0)
Assembly Language Coding:	ADMB 4,X'915'

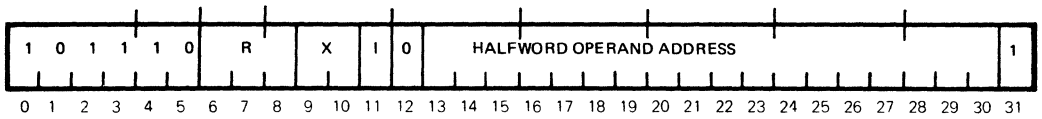
	PSWR	GPR4	Memory Byte 00915
Before Execution	10000800	00000099	8A
After Execution	20000804	00000123	8A

Note — The contents of memory byte 00915, with zeros prefixed, are added to the contents of GPR4, and the result is transferred to GPR4. CC2 is set.

ADD MEMORY HALFWORD

ADMH
d,*m,x

B800



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign bit (bit 16) is extended 16 bits to the left to form a word. This word is algebraically added to the contents of the GPR specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(EHL)_{SE}+(R) \rightarrow R$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 40D68
Hex Instruction: BB 84 10 97 (R=7, X=0, I=0)
Assembly Language Coding: ADMH 7,X'41096'

Before Execution	PSWR	GPR7	Memory Halfword 41096
	20040D68	000006C4	8C42
After Execution	PSWR	GPR7	Memory Halfword 41096
	10040D6C	FFFF9306	8C42

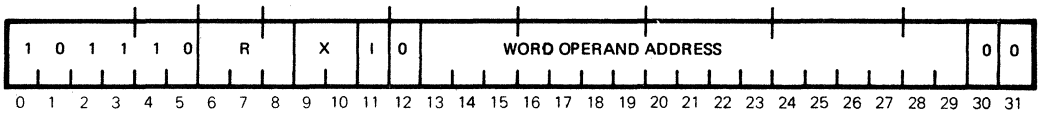
Note

The contents of memory halfword 41096 with sign extension are added to the contents of GPR7, and the result replaces the contents of GPR7. CC3 is set.

ADMW
d,*m,x

ADD MEMORY WORD

B800



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed and algebraically added to the contents of the GPR specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION — (EWL)+(R) → R

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	00D50
Hex Instruction:	BB 00 11 AC (R=6, X=0, I=0)
Assembly Language Coding:	ADMW 6,X'11AC'

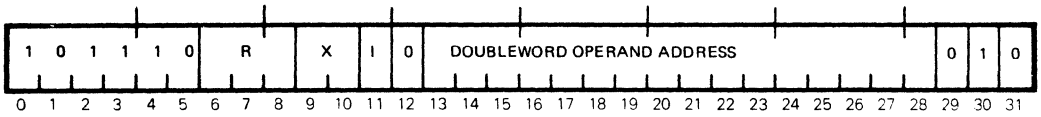
	PSWR	GPR6	Memory Word 011AC
Before Execution	400000D50	0037C1F3	004FC276
After Execution	200000D54	00878469	004FC276

Note — The contents of memory word 011AC are added to the contents of GPR6. The result is transferred to GPR6. CC2 is set.

ADD MEMORY DOUBLEWORD

ADMD
d,*m,x

B800



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and algebraically added to the contents of the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The contents of the GPR specified by R+1 are added to the contents of the least significant word of the doubleword first. The contents of the GPR specified by R are added to the contents of the most significant word of the doubleword last. The resulting doubleword is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL + 1) + (R+1)→R+1 + Carry

(EWL) + (R) + Carry → R

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI (R, R+1) is greater than zero
CC3: ISI (R, R+1) is less than zero
CC4: ISI (R, R+1) is equal to zero

EXAMPLE

Memory Location: 08E3C
Hex Instruction: BA 00 92 52 (R=4, X=0, I=0)
Assembly Language Coding: ADMD 4,X'9250'

Before Execution

PSWR	GPR4	GPR5
08008E3C	000298A1	815BC63E

Memory Word 09250	Memory Word 09254
3B69A07E	7F3549A4

After Execution

PSWR	GPR4	GPR5
20008E40	3B6C3920	00913FE2

Memory Word 09250	Memory Word 09254
3B69A07E	7F3579A4

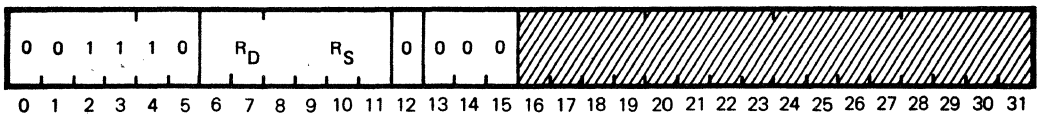
Note

The doubleword obtained from the contents of memory words 09250 and 09254 is added to the doubleword obtained from the contents of GPR4 and GPR5. The result is transferred to GPR4 and GPR5. CC2 is set.

ADR
s,d

ADD REGISTER TO REGISTER

3800



DEFINITION — The word in the GPR specified by R_D is algebraically added to the word in the GPR specified by R_S. The resulting word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_S+R_D) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	03FA2
Hex Instruction:	3B 70 (R_D=6, R_S=7)
Assembly Language Coding:	ADR 7,6

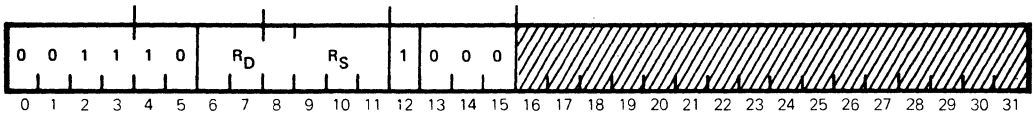
	PSWR	GPR6	GPR7
Before Execution	08003FA2	FF03C67D	045C6E3F
After Execution	20003FA4	036034BC	045C6E3F

Note — The contents of GPR6 and GPR7 are added and the result is transferred to GPR6. CC2 is set.

ADD REGISTER TO REGISTER MASKED

ADRM
s,d

3808



DEFINITION

The word in the GPR specified by R_D is algebraically added to the word in the GPR specified by R_S. The sum of this addition is masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)+(R_D)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location: 16A9A
Hex Instruction: 3B 78 (R_D=6, R_S=7)
Assembly Language Coding: ADRM 7,6

	PSWR	GPR4	GPR6	GPR7
Before Execution	40016A9A	007FFFFC	004FC276	0037C1F3
After Execution	20016A9C	0007FFFC	00078468	0037C1F3

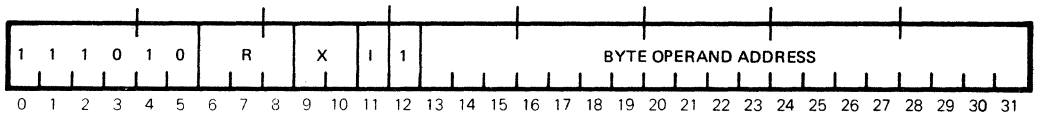
Note

The contents of GPR6 and GPR7 are added; the result is ANDed with the contents of GPR4 and transferred to GPR6. CC2 is set.

ARMB
s,*m,x

ADD REGISTER TO MEMORY BYTE

E808



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and algebraically added to the contents of the GPR specified by R. Bits 24-31 of the result are then transferred to the memory byte location specified by the EBA. The GPR and the other three bytes in the word which contains the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

(R)+(EBL) → EBL

CONDITION CODE RESULTS

CC1: Undefined
CC2: Undefined
CC3: Undefined
CC4: ISI the 32-bit sum is equal to zero

EXAMPLE

Memory Location: 01A64
Hex Instruction: EB 08 1A 97 (R=6, X=0, I=0)
Assembly Language Coding: ARMB 6,X'1A97'

Before Execution	PSWR	GPR6	Memory Byte 01A97
	00001A64	0000004A	39
After Execution	PSWR	GPR6	Memory Byte 01A97
	00001A68	0000004A	83

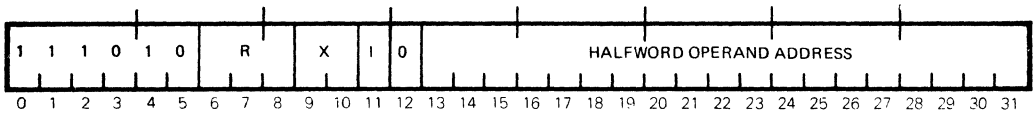
Note

The contents of GPR6 and memory byte 01A97 are added and the result is transferred to memory byte 01A97.

ADD REGISTER TO MEMORY HALFWORD

ARMH
S,*M,X

E800



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and algebraically added to the least significant halfword (bits 16-31) of the GPR specified by R. The result is then transferred to the memory halfword location specified by the EHA. The other halfword of the word which contains the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31})+(EHA) \rightarrow EHL$

CONDITION CODE RESULTS

CC1: Undefined
CC2: Undefined
CC3: Undefined
CC4: ISI (EHL) is equal to zero

EXAMPLE

Memory Location: 200B4
Hex Instruction: EA 82 09 19 (R=5, X=0, I=0)
Assembly Language Coding: ARMH 5,X'20918'

Before Execution	PSWR	GPR5	Memory Halfword 20918
	000200B4	FFFF8C42	06C4
After Execution	PSWR	GPR5	Memory Halfword 20918
	000200B8	FFFF8C42	9306

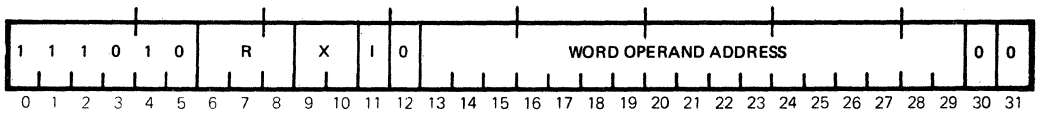
Note

The contents of bits 16-31 of GPR5 and memory halfword 20918 are added and the result is transferred to memory halfword 20918.

ARMW
s,*m,x

ADD REGISTER TO MEMORY WORD

E800



DEFINITION — The word in memory specified by the Effective Word Address (EWA) is accessed and algebraically added to the word in the GPR specified by R. The resulting word is then transferred to the memory word location specified by the EWA.

SUMMARY EXPRESSION — (E)+(EWL) → EWL

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI (EWL) is greater than zero
CC3: ISI (EWL) is less than zero
CC4: ISI (EWL) is equal to zero

EXAMPLE

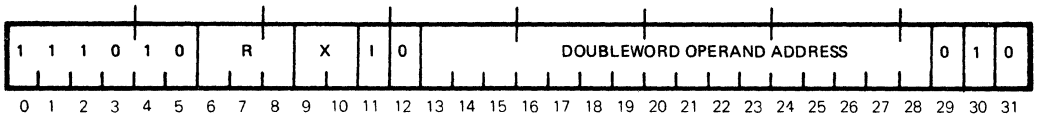
Memory Location:			03000
Hex Instruction:			EB 80 31 00 (R=7, X=0, I=0)
Assembly Language Coding:			ARMW 7,X'3100'
Before Execution	PSWR	GPR7	Memory Word 03100
	08003000	245C6E3F	FF03C67D
After Execution	PSWR	GPR7	Memory Word 03100
	20003004	245C6E3F	236034BC

Note — The contents of GPR7 and memory word 03100 are added and the result is transferred to memory word 03100. CC2 is set.

ADD REGISTER TO MEMORY DOUBLEWORD

ARMD
s,*m,x

E800



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and algebraically added to the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The contents of the GPR specified by R+1 are added to the contents of the least significant word of the doubleword first. The resulting doubleword is transferred to the memory doubleword location specified by the EDA.

SUMMARY EXPRESSION

(R+1)+(EQL+1) → EWL+1+Carry

(R)+(EWL)+Carry → EWL

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI (EDL) is greater than zero
CC3: ISI (EDL) is less than zero
CC4: ISI (EDL) is equal to zero

EXAMPLE

Memory Location: 0819C
Hex Instruction: EB 00 83 AA (R=6, X=0, I=0)
Assembly Language Coding: ARMD 6,X'83A8'

Before Execution

PSWR	GPR6	GPR7
4000819C	01A298A1	F15BC63E

Memory Word 083A8	Memory Word 083AC
3B69A07E	7F3579A4

After Execution

PSWR	GPR6	GPR7
200081A0	01A298A1	F15BC63E

Memory Word 083A8	Memory Word 083AC
3D0C3920	70913FE2

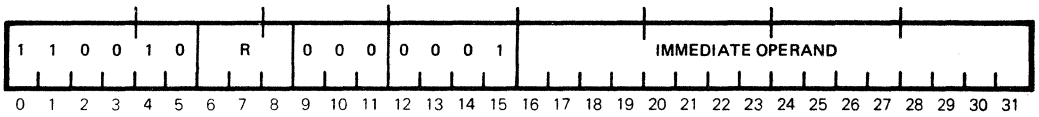
Note

The doubleword obtained from GPR6 and GPR7 is added to the doubleword from memory words 083A8 and 083AC. The result is transferred to memory words 083A8 and 083AC. CC2 is set.

ADI
d,v

ADD IMMEDIATE

C801



DEFINITION — The sign of the least significant bit (bit 16) of the Instruction Word is extended 16 bits to the left to form a word. This word is algebraically added to the word in the GPR specified by R. The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION — $(IW_{16-31})_{SE}{}^{+}(R) \rightarrow R$

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	00D88
Hex Instruction:	C8 01 86 B2 (R=0)
Assembly Language Coding:	ADI 0,X'86B2'

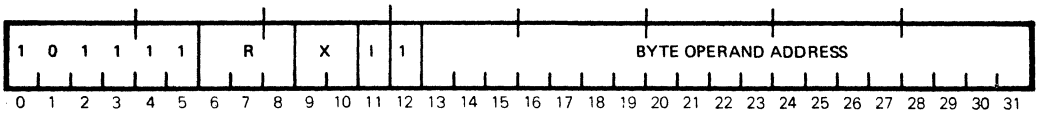
	PSWR	GPR0
Before Execution	20000D88	0000794E
After Execution	08000D8C	00000000

Note — The immediate operand, sign extended, is added to the contents of the GPR0 and the result replaces the previous contents of GPR0. CC4 is set.

SUBTRACT MEMORY BYTE

SUMB
d,*m,x

BC08



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros are appended to the most significant end to form a word. This word is algebraically subtracted from the word in the GPR specified by R. The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)-\left[0_{0-23},(EBL)\right] \rightarrow R$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	01000
Hex Instruction:	BC 88 12 01 (R=1, X=0, I=0)
Assembly Language Coding:	SUMB 1,X'1201'

	PSWR	GPR1	Memory Byte 01201
Before Execution	40001000	0194A7F2	9A
After Execution	20001004	0194A758	9A

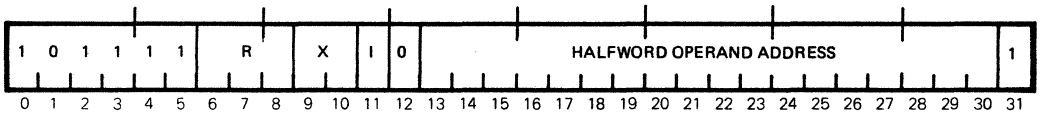
Note

The contents of memory byte 01201, with 24 zeros prefixed, are subtracted from the contents of GPR1. The result is transferred to GPR1. CC2 is set.

SUMH
d,*m,x

SUBTRACT MEMORY HALFWORD

BC00



DEFINITION

The halfword in memory specified by the Effective Halfword Address is accessed and the sign bit (bit 16) is extended 16 bits to the left to form a word. This word is algebraically subtracted from the word in the GPR specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)-(EHL)_{SE} \rightarrow R$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 01604
Hex Instruction: BF 00 18 77 (R=6, X=0, I=0)
Assembly Language Coding: SUMH 6,X'1876'

Before Execution	PSWR	GPR6	Memory Halfword 01876
	10001604	00024CB3	34C6
After Execution	PSWR	GPR6	Memory Halfword 01876
	20001608	000217ED	34C6

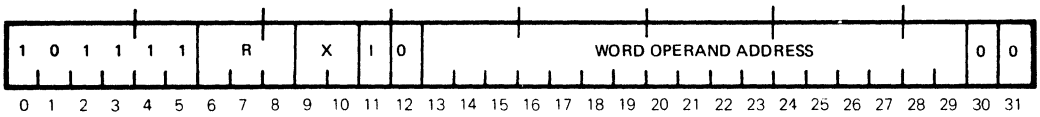
Note

The contents of memory halfword 01876, sign extended, are subtracted from the contents of GPR6. The result is transferred to GPR6. CC2 is set.

SUBTRACT MEMORY WORD

SUMW
d,*m,x

BC00



DEFINITION — The word in memory specified by the Effective Word Address is accessed and algebraically subtracted from the word in the GPR specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION — (R)-(EWL) → R

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	6C208
Hex Instruction:	BC 86 F9 14 (R=1, X=0, I=0)
Assembly Language Coding:	SUMW 1,X'6F914'

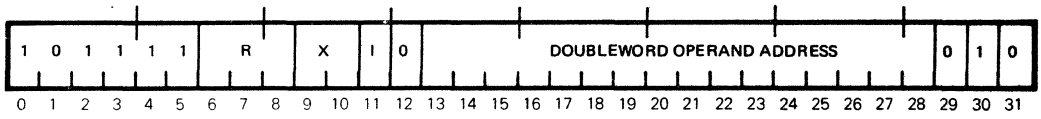
	PSWR	GPR1	Memory Word 6F914
Before Execution	0406C208	00A6264D	00074BC3
After Execution	2006C20C	009EDA8A	00074BC3

Note — The contents of memory word 6F914 are subtracted from the contents of GPR1 and the result is transferred to GPR1. CC2 is set.

SUMD
d,*m,x

SUBTRACT MEMORY DOUBLEWORD

BC00



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and algebraically subtracted from the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The word located in the GPR specified by R+1 is subtracted from the least significant word of the doubleword first. The resulting doubleword is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(R+1)-(EWL+1) → R+1-Borrow

(R)-(EWL)-Borrow → R

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI (R, R+1) is greater than zero
CC3: ISI (R, R+1) is less than zero
CC4: ISI (R, R+1) is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: BF 00 31 02 (R=6, X=0, I=0)
Assembly Language Coding: SUMD 6,X'3100

Before Execution

PSWR	GPR6	GPR7
10003000	5AD983B7	C833D509

Memory Word 03100	Memory Word 03104
153B0492	5BE87A16

After Execution

PSWR	GPR6	GPR7
20003004	459E7F25	6C4B5AF3

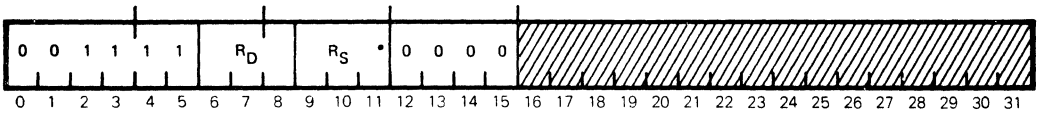
Memory Word 03100	Memory Word 03104
153B0492	5BE87A16

Note

The doubleword obtained from memory words 03100 and 03104 is subtracted from the doubleword in GPR6 and GPR7. The result is transferred to GPR6 and GPR7. CC2 is set.

SUBTRACT REGISTER FROM REGISTER

3C00



DEFINITION

The word in the GPR specified by R_S is algebraically subtracted from the word in the GPR specified by R_D. The resulting word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_D)-(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	106AE
Hex Instruction:	3C A0 (R_D=1, R_S=2)
Assembly Language Coding:	SUR 2,1

	PSWR	GPR1	GPR2
Before Execution	100106AE	12345678	12345678
After Execution	080106B0	00000000	12345678

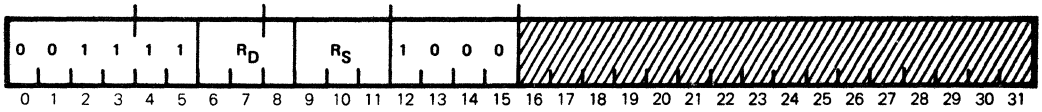
Note

The contents of GPR2 are subtracted from the contents of GPR1. The result is replaced in GPR1. CC4 is set.

SURM
s,d

SUBTRACT REGISTER FROM REGISTER MASKED

3C08



DEFINITION — The word in the GPR specified by R_S is algebraically subtracted from the word in the GPR specified by R_D. The difference of this subtraction is then masked (Logical AND Function) with the contents of the Mask register (R4). The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_D)-(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI (R_D) is greater than zero
CC3: ISI (R_D) is less than zero
CC4: ISI (R_D) is equal to zero

EXAMPLE

Memory Location:	00496
Hex Instruction:	3F 58 (R_D=6, R_S=5)
Assembly Language Coding:	SURM 5,6

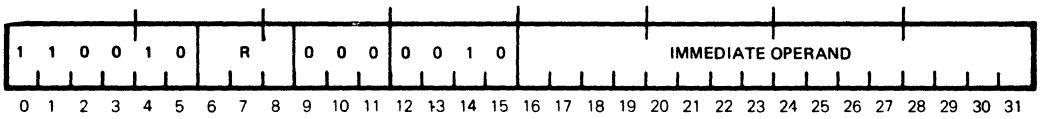
	PSWR	GPR4	GPR5	GPR6
Before Execution	10000496	00FFFF00	00074BC3	00A6264D
After Execution	20000498	00FFFF00	00074BC3	009EDA00

Note — The contents of GPR5 are subtracted from the contents of GPR6. The result is ANDed with the contents of GPR4 and transferred to GPR6. CC2 is set.

SUBTRACT IMMEDIATE

SUI
d,v

C802



DEFINITION

The sign of the least significant halfword (bits 16-31) of the Instruction Word is extended 16 bits to the left to form a word. This word is algebraically subtracted from the word in the GPR specified by R. The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)-(W_{16-31})_{SE} \rightarrow R$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location: 019B8
Hex Instruction: CB 82 83 9A (R=7)
Assembly Language Coding: SUI 7,X'839A'

	PSWR	GPR7
Before Execution	100019B8	FFFF839A
After Execution	080019BC	00000000

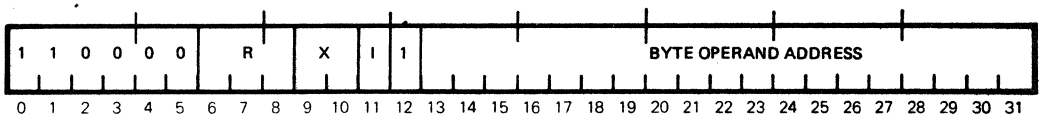
Note

The immediate operand with sign extension is subtracted from the contents of GPR7. The result is transferred to GPR7. CC4 is set.

MPMB
d,*m,x

MULTIPLY BY MEMORY BYTE

C008



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros are appended to the most significant end to form a word. This word is algebraically multiplied by the word in the GPR specified by R+1. R+1 is the GPR one greater than specified by R. The double-precision result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION

0_{0-23},(EBA)x(R+1) → R,R+1

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R, R+1) is greater than zero
CC3: ISI (R, R+1) is less than zero
CC4: ISI (R, R+1) is equal to zero

EXAMPLE

Memory Location:	2BA28
Hex Instruction:	C0 0A C3 D9
Assembly Language Coding:	MPMB 0,X'2C3D9'

Before Execution

PSWR	GPR0	GPR1
0002BA28	12345678	6F90C859

Memory Byte 2C3D9
40

After Execution

PSWR	GPR0	GPR1
2002BA2C	0000001B	E4321640

Memory Byte 2C3D9
40

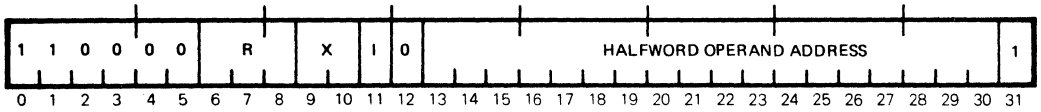
Note

The contents of memory byte 2C3D9, with zeros prefixed, are multipled by the contents of GPR1. The result is transferred to GPR0 and GPR1. CC2 is set.

MULTIPLY BY MEMORY HALFWORD

MPMH
d,*m,x

C0000



DEFINITION The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign bit (bit 16) is extended 16 bits to the left to form a word. This word is algebraically multiplied by the word in the GPR specified by R+1. R+1 is the GPR one greater than specified by R. The double-precision result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION $(EHL)_{SE} \times (R+1) \rightarrow R,R+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI(R, R+1) is greater than zero
CC3: ISI (R, R+1) is less than zero
CC4: ISI (R, R+1) is equal to zero

EXAMPLE

Memory Location:	096A4
Hex Instruction:	C1 00 9B 57 (R=2, X=0, I=0)
Assembly Language Coding:	MPMH 2,X'9B56'

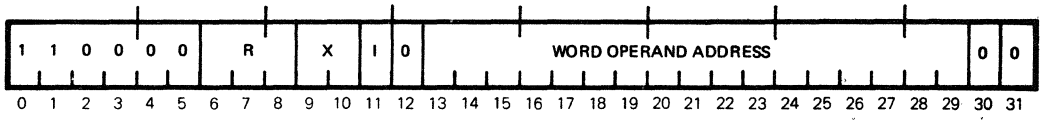
	PSWR	GPR2	GPR3	Memory Halfword 09B56
Before Execution	080096A4	12345678	00000003	FFFD
After Execution	100096A8	FFFFFFFF	FFFFFFF7	FFFD

Note The contents of GPR3 are multiplied by the contents of memory halfword 09B56. The doubleword result is transferred to GPR2 and GPR3. CC3 is set.

MPMW
d,*m,x

MULTIPLY BY MEMORY WORD

C000



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and algebraically multiplied by the word GPR specified by R+1. R+1 is the GPR one greater than specified by R. The double-precision result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION

(EWL)x(R+1) → (R,R+1)

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R, R+1) is greater than zero
CC3: ISI (R, R+1) is less than zero
CC4: ISI (R, R+1) is equal to zero

EXAMPLE

Memory Location: 04AC8
Hex Instruction: C3 00 4B 1C (R=6, X=0, I=0)
Assembly Language Coding: MPMW 6,X'4B1C'

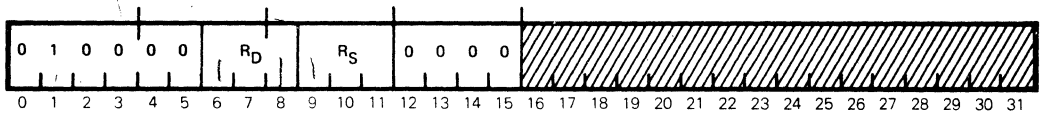
	PSWR	GPR6	GPR7	Memory Word 04B1C
Before Execution	10004AC8	00000000	80000000	80000000
After Execution	20004ACC	40000000	00000000	80000000

Note

The contents of GPR7 and memory word 04B1C are multiplied, and the result is transferred to GPR6 and GPR7. CC2 is set.

MULTIPLY REGISTER BY REGISTER

4000



DEFINITION

The word GPR specified by R_S is algebraically multiplied by the word in the GPR specified by R_D+1. R_D+1 is the GPR one greater than specified by R_D. The double-precision result is transferred to the GPR specified by R_D and R_D+1.

NOTES

1. The multiplicand register R_S can be any register, including register R_D+1; however, R_D must be an even-numbered register.

2. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

SUMMARY EXPRESSION

$(R_S) \times (R_D+1) \rightarrow R_D, R_D+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_D,R_D+1) is greater than zero
CC3: ISI (R_D,R_D+1) is less than zero
CC4: ISI (R_D,R_D+1) is equal to zero

EXAMPLE

Memory Location: 0098E
Hex Instruction: 40 10 (R_D=0,R_S=1)
Assembly Language Coding: MPR 1,0

	PSWR	GPR0	GPR1
Before Execution	1000098E	00000000	0000000F
After Execution	20000990	00000000	000000E1

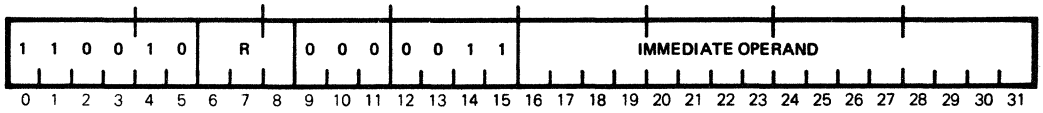
Note

The content of GPR1 is multiplied by itself, and the doubleword product is transferred to GPR0 and GPR1. CC2 is set.

MPI
d,v

MULTIPLY IMMEDIATE

C803



DEFINITION

The sign of the least significant halfword (bits 16-31) of the Instruction Word is extended 16 bits to the left to form a word. This word is algebraically multiplied by the word in the GPR specified by R+1. R+1 is the GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(IW_{16-31})_{SE} \times (R+1) \rightarrow R,R+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R,R+1) is greater than zero
CC3: ISI (R,R+1) is less than zero
CC4: ISI (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00634
Hex Instruction: CB 03 01 00 (R=6)
Assembly Language Coding: MPI 6,X'0100'

	PSWR	GPR6	GPR7
Before Execution	20000634	12345678	F37A9B15
After Execution	10000638	FFFFFFF3	7A9B1500

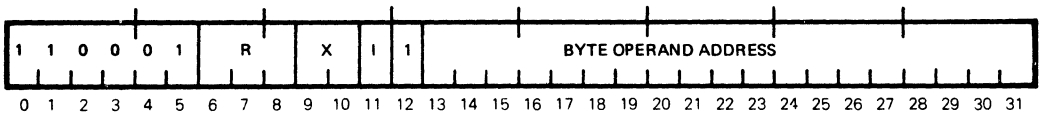
Note

The immediate operand, sign extended, is multiplied by the contents of GPR7. The result is transferred to GPR6 and GPR7. CC3 is set.

DIVIDE BY MEMORY BYTE

DVMB
d,*m,x

C408



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros are appended to the most significant end to form a word. This word is algebraically divided into the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1, and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is set to the original sign of the dividends. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the original signs of the dividend and the divisor except when the absolute value of the dividend is less than the absolute value of the divisor. In that case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R1) / \left[0_{0-23},(EBL)\right] \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI $(R+1_{0-31})$ is greater than zero
CC3: ISI $(R+1_{0-31})$ is less than zero
CC4: ISI $(R+1_{0-31})$ is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: C4 08 30 BF (R=0, X=0, I=0)
Assembly Language Coding: DVMB 0,X'30BF'

	PSWR	GPR0	GPR1	Memory Byte 030BF
Before Execution	10003000	00000000	00000139	04
After Execution	20003004	00000001	0000004E	04

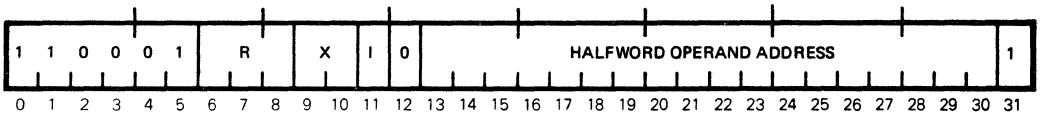
Note

The doubleword contents of GPR0 and GPR1 are divided by the content of memory byte 030BF with 24 zeros prefixed. The quotient is transferred to GPR1 and the remainder is transferred to GPR0. CC2 is set.

DVMH
d,*m,x

DIVIDE BY MEMORY HALFWORD

C400



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed, and the sign is extended 16 bits to the left to form a word. This word is algebraically divided into the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1 and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is set to the original sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In that case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R+1)/(EHL)_{SE} \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI $R+1_{0-31}$ is greater than zero
CC3: ISI $R+1_{0-31}$ is less than zero
CC4: ISI $R+1_{0-31}$ is equal to zero

EXAMPLE

Memory Location: 05A94
Hex Instruction: C7 00 5D 6B (R=6, X=0, I=0)
Assembly Language Coding: DVMH 6,X'5D6A'

Before Execution	PSWR	GPR6	GPR7	Memory Halfword 05D6A
	08005A94	00000000	0000003B	FFF8
After Execution	PSWR	GPR6	GPR7	Memory Halfword 05D6A
	10005A98	00000005	FFFFFFF9	FFF8

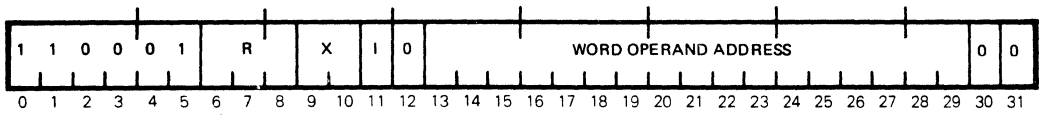
Note

The doubleword contents of GPR6 and GPR7 are divided by the contents of memory halfword 05D6A with sign extension. The quotient is transferred to GPR7 and the remainder is transferred to GPR6. CC3 is set.

DIVIDE BY MEMORY WORD

DVMW
d,*m,x

C400



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and algebraically divided into the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1, and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is set to the original sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In that case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

(R,R+1)/(EWL) → R+1

Remainder → R

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI $R+1_{0-31}$ is greater than zero
CC3: ISI $R+1_{0-31}$ is less than zero
CC4: ISI $R+1_{0-31}$ is equal to zero

EXAMPLE

Memory Location: 078C0
Hex Instruction: C6 00 7B 5C (R=4, X=0, I=0)
Assembly Language Coding: DVMW 4,X'7B5C'

	PSWR	GPR4	GPR5	Memory Word 07B5C
Before Execution	400078C0	00000000	039A20CF	FC000000
After Execution	080078C4	039A20CF	00000000	FC000000

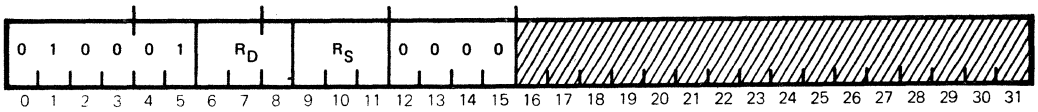
Note

The doubleword obtained from GPR4 and GPR5 is divided by the contents of memory word 07B5C. The quotient is transferred to GPR5, and the remainder is transferred to GPR4. CC4 is set.

DVR
s,d

DIVIDE REGISTER BY REGISTER

4400



DEFINITION

The word in the GPR specified by R_S is algebraically divided into the doubleword in the GPR specified by R_D and R_D+1. R_D+1 is the GPR one greater than specified by R_D. The resulting quotient is then transferred to the GPR specified by R_D+1, and the remainder is transferred to the GPR specified by R_D. The sign of the GPR specified by R_D (remainder) is set to the original sign of the dividend. The sign of the GPR specified by R_D+1 (quotient) will be the algebraic product of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In that case, the resulting quotient (GPR specified by R_D+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R_D must have an even address.

3. R_S must not equal R_D or R_D+1.

SUMMARY EXPRESSION

$(R_D,R_D+1)/R_S \rightarrow R_D+1$

Remainder $\rightarrow R_D$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI R_D+1_{0-31} is greater than zero
CC3: ISI R_D+1_{0-31} is less than zero
CC4: ISI R_D+1_{0-31} is equal to zero

EXAMPLE

Memory Location: 04136
Hex Instruction: 47 20 (R_D=6,R_S=2)
Assembly Language Coding: DVR 2,6

	PSWR	GPR2	GPR6	GPR7
Before Execution	10004136	0000000A	00000000	000000FF
After Execution	20004138	0000000A	00000005	00000019

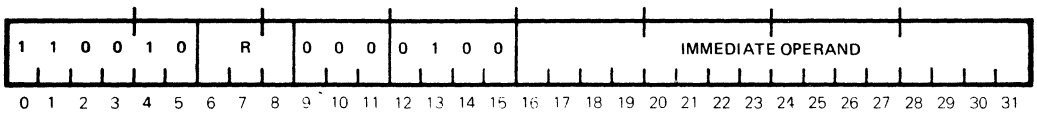
Note

The doubleword obtained from GPR6 and GPR7 is divided by the contents of GPR2. The quotient is transferred to GPR7, and the remainder is transferred to GPR6. CC2 is set.

DIVIDE IMMEDIATE

DVI
d,v

C804



DEFINITION

The sign of the least significant halfword (bits 16-31) of the Instruction Word is extended 16 bits to the left to form a word. This word is algebraically divided into the doubleword in the GPR specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1, and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is set to the original sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In that case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R+1)/(IW_{16-31})_{SE} \rightarrow R+1$

Remainder $\rightarrow$ R

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI $R+1_{0-31}$ is greater than zero
CC3: ISI $R+1_{0-31}$ is less than zero
CC4: ISI $R+1_{0-31}$ is equal to zero

EXAMPLE

Memory Location:	08000
Hex Instruction:	C9 04 FF FD (R=2)
Assembly Language Coding:	DVI 2,-3

	PSWR	GPR2	GPR3
Before Execution	04008000	00000000	000001B7
After Execution	10008004	00000001	FFFFFF6F

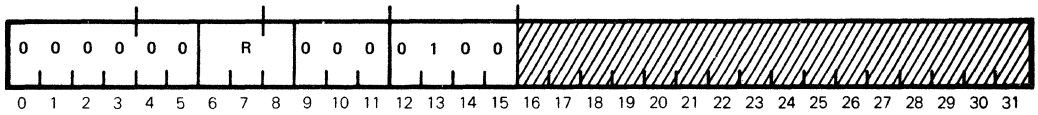
Note

The doubleword obtained from GPR2 and GPR3 is divided by the immediate operand, sign extended. The quotient is transferred to GPR3, and the remainder is transferred to GPR2. CC3 is set.

ES
d

EXTEND SIGN

0004



DEFINITION — The sign (bit 0) of the contents of the GPR specified by R+1 is extended through all 32 bit positions of the GPR specified by R.

SUMMARY EXPRESSION — $(R+1_0) \rightarrow R_{0-31}$

CONDITION CODE RESULTS
CC1: Always zero
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	0083A
Hex Instruction:	00 84 (R=1)
Assembly Language Coding:	ES 1

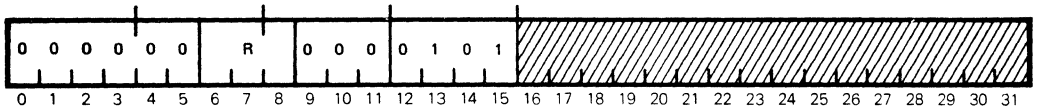
	PSWR	GPR1	GPR2
Before Execution	0800083A	0000B074	8000C361
After Execution	1000083C	FFFFFFFF	8000C361

Note — Bits 0-31 of GPR1 are set to one's. CC3 is set.

ROUND REGISTER

RND
d

0005



DEFINITION The contents of the GPR specified by R are incremented by one if bit position 0 of the GPR specified by R+1 is equal to one. R+1 is the GPR one greater than specified by R.

SUMMARY EXPRESSION $(R)+1, if(R+1_0)=1$

CONDITION CODE RESULTS
CC1: ISI arithmetic exception
CC2: ISI R_{0-31} is greater than zero
CC3: ISI R_{0-31} is less than zero
CC4: ISI R_{0-31} is equal to zero

EXAMPLE

Memory Location:	00FFE
Hex Instruction:	03 75 (R=6)
Assembly Language Coding:	RND 6

	PSWR	GPR6	GPR7
Before Execution	40000FFE	783A05B2	92CD061F
After Execution	20001000	783A05B3	92CD061F

Note The contents of GPR6 are incremented by one, and the result is returned to GPR6. CC2 is set.

FLOATING-POINT ARITHMETIC INSTRUCTION

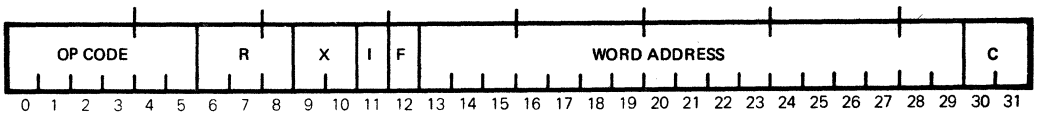
GENERAL DESCRIPTION

The Floating-Point Arithmetic instructions provide the capability to add, subtract, multiply, or divide operands of large magnitude with precise results. A floating-point number is made up of three parts: a sign, a fraction, and an exponent. The sign applies to the fraction and denotes a positive or negative value. The fraction is a binary number with an assumed radix point between the sign bit and the most significant bit. The exponent is a 7-bit binary power to which the base 16 is raised. The quantity that the floating-point number represents is obtained by multiplying the fraction by the number 16 raised to the power represented by the exponent.

INSTRUCTION FORMAT

The following instruction format is used for all floating-point operations:

MEMORY REFERENCE



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-10 designate one of three index registers.

Bit 11 indicates whether an indirect addressing operation is to be performed.

Bits 12-31 directly specifies the address of the operand when the X and I fields are equal to zero. If X is not equal to zero, indirect addressing is specified. Bit 12 (F) is used as an augment bit by the Floating-Point instructions.

CONDITION CODE UTILIZATION

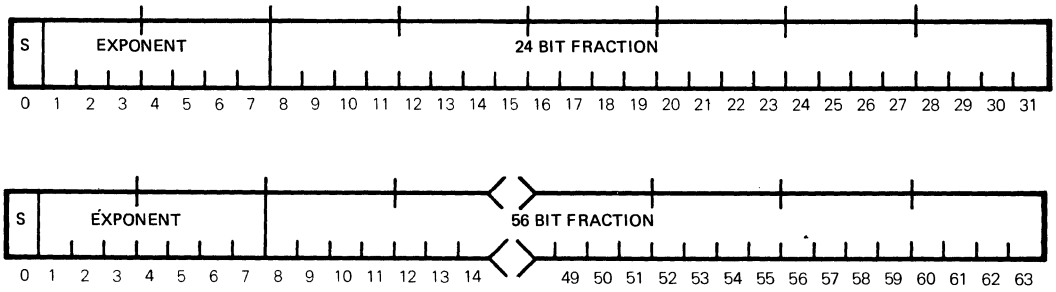
Execution of all Floating-Point Arithmetic instructions causes a Condition Code to be set to indicate whether the result of the operation was greater than, less than, or equal to zero. Arithmetic exceptions produced by a Floating-Point operation are also reflected by the Condition Code results.

The meaning of the Condition Codes differ for the execution of the Floating-Point instructions. CC1 is set by an Arithmetic Exception condition (underflow or overflow). To differentiate between these exceptions, CC4 is also set when the overflow condition occurs. In both instances, either CC2 or CC3 is used to indicate the state of what would have been the sign of the resultant fraction had the arithmetic exception not occurred. The following table reflects the possible Condition Code settings:

Condition Code				Definition
CC1	CC2	CC3	CC4	
1	0	0	0	Arithmetic exception
0	1	0	0	Positive fraction
0	0	1	0	Negative
0	0	0	1	Zero fraction
1	1	0	0	Exponent Underflow, positive fraction
1	0	1	0	Exponent Underflow, negative fraction
1	1	0	1	Exponent Overflow, positive fraction
1	0	1	1	Exponent Overflow, negative fraction

FLOATING-POINT ARITHMETIC OPERANDS

A floating-point number can be represented in two different formats: word and doubleword. These two formats are the same except that the doubleword contains eight additional hexadecimal digits of significance in the fraction These two formats are shown below.



The floating-point number, in either format, is made up of three parts: a sign, a fraction, and an exponent. The sign bit (bit 0) applies to the fraction and denotes a positive or negative value. The fraction is a hexadecimal normalized number with a radix point to the left of the highest order fraction bit (bit 8). The exponent (bits 1-7) is a 7-bit binary number to which the base 16 is raised.

Negative exponents are carried in the two's complement format. To remove the sign and therefore enable exponents to be compared directly, both positive and negative exponents are biased up by 40_{16} (excess 64_{10} notation).

The quantity that a floating-point number represents is obtained by multiplying the fraction by the number 16_{10} raised to the power of the exponent minus 40_{16}.

A positive floating-point number is converted to a negative floating-point number by taking the two's complement of the positive fraction and the one's complement of the biased exponent. If the minus one case is ruled illegitimate, all floating-point numbers can be converted from positive to negative and from negative to positive by taking the two's complement of the number in floating-point format. Signed numbers in the floating-point format can then be compared directly, one with another, by using the Compare Arithmetic class of instructions.

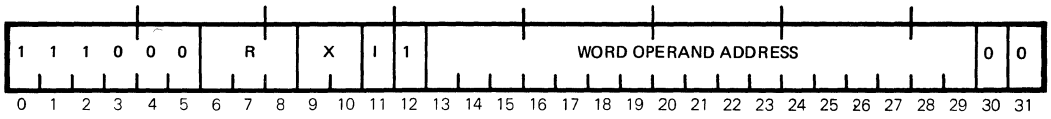
All floating-point operands must be normalized before being operated on by a floating-point instruction. A positive floating-point number is normalized when the value of the fraction is less than one and greater than or equal to one-sixteenth ($1 > F \geq 1/16$). A negative floating-point number is normalized when the value of the fraction is greater than minus one and less than or equal to minus one-sixteenth ($-1 < F \leq -1/16$). All floating-point answers are normalized by the CPU. If a floating-point operation results in a minus one of the form 1 XXX XXXX 0000...0000, the CPU will convert that result to a legitimate normalized floating-point number of the form 1 YYY YYYY 1111 0000...0000, where YYY YYYY is one less than XXX XXXX.

A hexadecimal guard digit is appended to the least significant hexadecimal digit of the floating-point word operands by the CPU. This guard digit is carried throughout all floating-point word computations. The most significant bit of the guard digit is used as the basis for rounding by the CPU at the end of every floating-point word computation.

ADFW
d,*m,x

ADD FLOATING-POINT WORD

E008



DEFINITION

The floating-point operand in memory is accessed. If either of the floating-point numbers is negative, the one's complement of the base 16 exponent (bits 1-7) is taken of the negative number. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the signed fractions of the two numbers are algebraically added. If the exponents differ, and the difference is greater than or equal to one, or less than or equal to six (1 exponent difference 6), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit. After exponent equalization, the fractions are added algebraically. The normalized and rounded sum of the two fractions is placed in bit positions 0 and 8-31 of GPR d. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit positions 1-7 of GPR d.

NOTES

1. If the resulting fraction equals zero, the exponent and fraction are set to zero in GPR d.

2. Operands are expected to be normalized.

3. If the exponent difference is greater than six, the operand having the larger exponent is normalized and placed in the GPR specified by R.

SUMMARY EXPRESSION

(R)+(EWL) → (R)

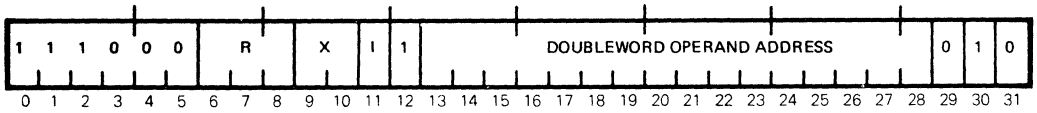
CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

ADD FLOATING POINT DOUBLEWORD

ADFD
d,*m,x

E008



DEFINITION

The floating-point operand in memory is accessed. If either of the floating-point numbers is negative, the one's complement of the base 16 exponent (bits 1-7) is taken of the negative number. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the signed fractions of the two numbers are algebraically added. If the exponents differ, and the difference is greater than or equal to one, or less than or equal to six ($1 \leq$ exponent difference ≤ 6), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit. After exponent equalization, the fractions are added algebraically. The normalized and rounded sum of the two fractions is placed in bit positions 0 and 8-63 of GPR d+1. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit positions 1-7 of GPR d.

NOTES

1. If the resulting fraction equals zero, the exponent and fraction are set to zero in GPR d+1.

2. Operands are expected to be normalized.

3. If the exponent difference is greater than 13, the operand having the larger exponent is normalized and placed in the GPR specified by R, R+1.

SUMMARY EXPRESSION

(R),(R+1)+(EWL),(EWL+1)→ (R),(R+1)

CONDITION CODE RESULTS

CC1: ISI arithmetic exception

CC2: ISI $R_{0,8-31}$ is greater than zero

CC3: ISI $R_{0,8-31}$ is less than zero

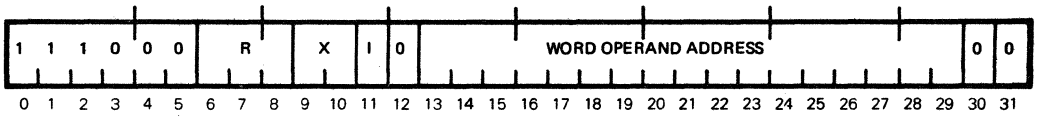
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: ADFD R,X'(DW Op Addr)'

SUFW
d,*m,x

SUBTRACT FLOATING-POINT WORD

E000



DEFINITION

The floating-point operand in memory is accessed. If either the floating-point number in the GPR or memory is negative, the one's complement of the base 16 exponent (bits 1-7) is taken. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the 24-bit signed fractions are algebraically subtracted (i.e., the memory operand is subtracted from the GPR or GPR s). If the exponents differ, and the difference is greater than one, or less than six ($1 \leq$ exponent difference ≤ 6), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit at a time until the exponents are equalized. The exponent of this operand is effectively incremented by one each time the fraction is shifted right one hexadecimal. After exponent equalization, the fractions are subtracted algebraically. The normalized and rounded difference between the two fractions is placed in bit positions 0 and 8-31 of GPR d. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit positions 1-7 of GPR d.

NOTES

1. If the resulting fraction is equal to zero, the exponent and fraction are set to zero in the GPR or GPR s.

2. Operands are expected to be normalized.

3. If the exponent difference is greater than six, the operand having the larger exponent is normalized and placed in the GPR specified by R.

SUMMARY EXPRESSION

(R)-(EWL) → (R)

CONDITION CODE RESULTS

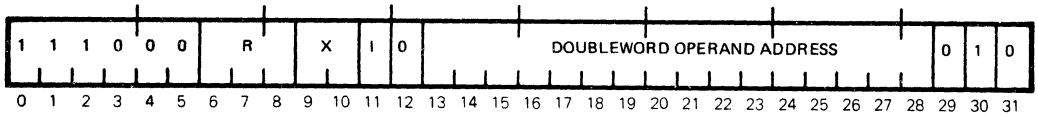
CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: SUFW R, X'(W Op Addr)'

SUBTRACT FLOATING-POINT DOUBLEWORD

SUFD
d,*m,x

E000



DEFINITION

The floating-point operand in memory is accessed. If either the floating-point number in the GPR or memory is negative, the one's complement of the base 16 exponent (bits 1-7) is taken. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the 24-bit signed fractions are algebraically subtracted (i.e., the memory operand is subtracted from the GPR or GPR s). If the exponents differ, and the difference is greater than or equal to one, or less than or equal to six ($1 \leq$ exponent difference ≤ 6), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit at a time until the exponents are equalized. The exponent of this operand is effectively incremented by one each time the fraction is shifted right one hexadecimal digit. After exponent equalization, the fractions are subtracted algebraically. The normalized and rounded difference between the two fractions is placed in bit positions 0 and 8-63 of GPR d+1. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit positions 1-7 of GPR d.

NOTES

1. If the resulting fraction is equal to zero, the exponent and fraction are set to zero in the GPR or GPR s.

2. Operands are expected to be normalized.

3. If the exponent difference is greater than 13, the operand having the larger exponent is normalized and placed in the GPR specified by R, R+1.

SUMMARY EXPRESSION

(R),(R+1)-(EWL),(EWL+1) → (R),(R+1)

CONDITION CODE RESULTS

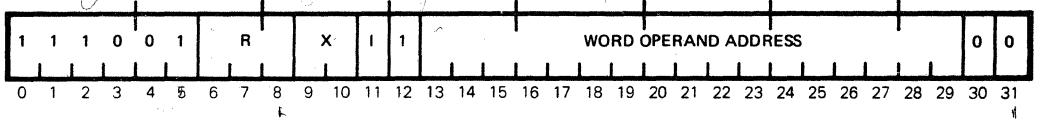
CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: SUFD R,X'(DW Op Addr)'

MPFW
d,*m,x

MULTIPLY FLOATING-POINT WORD

E408



DEFINITION

The floating-point operand fraction is multiplied by the fraction of GPR d. If either one or both of the floating-point numbers are negative, the exponent of the negative number is changed to its one's complement. Both exponents are then stripped of their 40_{16} bias and algebraically added. The normalized and rounded product of the multiplication is placed in bits 0 and 8-31 of GPR d. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the resulting exponent is placed in bits 1-7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(EWL_{0,8-31})x(R_{0,8-31}) \rightarrow R_{0,8-31}$

$(EWL_{1-7})+(R_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

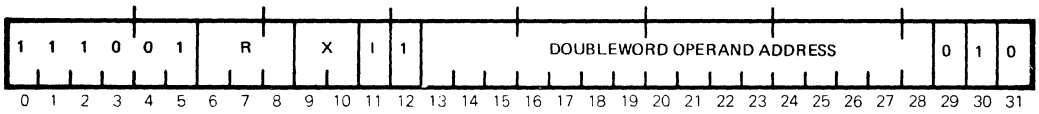
CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: MPFW R,X'(W Op Addr)'

MULTIPLY FLOATING-POINT DOUBLEWORD

MPFD
d,*m,x

E408



DEFINITION

The floating-point operand fraction is multiplied by the fraction of GPR d+1. If either one or both of the floating-point numbers are negative, the exponent of the negative number is changed to its one's complement. Both exponents are then stripped of their 40_{16} bias and algebraically added. The normalized and rounded product of the multiplication is placed in bits 0 and 8-63 of GPR d+1. The resulting exponent is biased up by 40_{16}, and if the resulting fraction is negative, the one's complement of the resulting exponent is placed in bits 1-7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(EWL_{0,8-31}, EWL+1_{0-31}) \times (R_{0,8-31}, R+1_{0-31})$

$\rightarrow R_{0,8-31}, R+1_{0-31}$

$(EWL_{1-7}) + (R_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

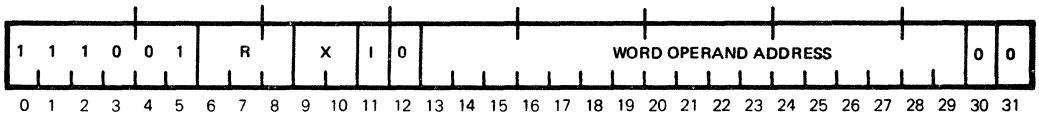
CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: MPFD R,X'(DW Op Addr)'

DVFW
d,*m,x

DIVIDE FLOATING-POINT WORD

E400



DEFINITION

The floating-point operand in memory (divisor) is accessed, and the fraction is divided into the fraction of GPR d. If either one or both of the floating-point numbers are negative, the one's complement of the exponent is taken. Both exponents are then stripped of their 40_{16} bias, and the exponent of the divisor is subtracted algebraically from the exponent of the dividend. The normalized and rounded quotient is placed in bit 0 and bit positions 8-31 of the GPR d. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the resulting fraction is placed in bits 1-7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_{0,8-31})/(EWL_{0,8-31}) \rightarrow R_{0,8-31}$

$(R_{1-7})-(EWL_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

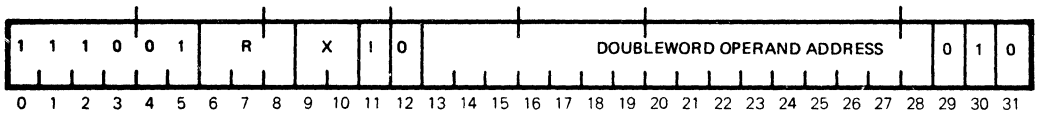
CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: DVFW R,X'(W Op Addr)'

DIVIDE FLOATING-POINT DOUBLEWORD

DVFD
d,*m,x

E400



DEFINITION

The floating-point operand in memory (divisor)is accessed and the fraction is divided into the fraction of GPR d+1. If either one or both of the floating-point numbers are negative, the one's complement of the exponent is taken. Both exponents are then stripped of their 40_{16} bias, and the exponent of the divisor is subtracted algebraically from the exponent of the dividend. The normalized and rounded quotient is placed in bit 0 and bit positions 8-63 of the GPR d+1. The resulting exponent is biased up by 40_{16}, and, if the resulting fraction is negative, the one's complement of the resulting fraction is placed in bits 1-7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_{0,8-31},{}^{R+1}{}_{0-31})/(EWL_{0,8-31},{}^{EWL+1}{}_{0-31})$

$\rightarrow R_{0,8-31},{}^{R+1}{}_{0-31}$

$(R_{1-7})-(EWL_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

CC1: ISI arithmetic exception
CC2: ISI $R_{0,8-31}$ is greater than zero
CC3: ISI $R_{0,8-31}$ is less than zero
CC4: ISI $R_{0,8-31}$ is equal to zero

Assembly Language Coding: DVFD R,X'(DW Op Addr)'

CONTROL INSTRUCTIONS

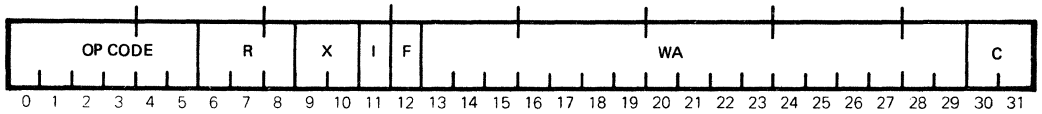
GENERAL DESCRIPTION

This group of instructions allows the mainframe to perform Execute, No Op , Halt, and Wait operations.

INSTRUCTION FORMATS

Control instructions use the Memory Reference and Interregister instruction formats. Several of the Control instructions vary the basic Interregister format in that certain portions are not used and are left blank.

MEMORY REFERENCE



Bits 0-5 define the Operation Code.

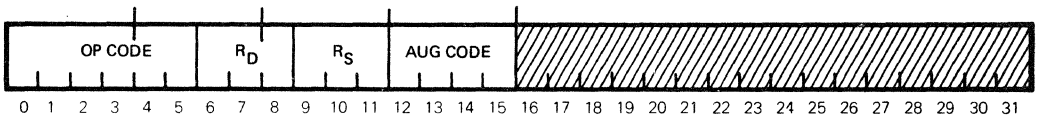
Bits 6-8 designate a General Purpose Register address (0-7).

Bits 9-10 designate one of three index registers.

Bit 11 indicates whether an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when the X and I fields are equal to zero.

INTERREGISTER



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

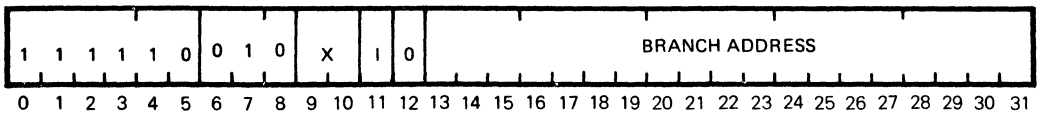
CONDITION CODE UTILIZATION

Condition Code results for Execute operations will be dependent on the instruction that was performed. All other control operations leave the current Condition Code unchanged.

BRANCH AND RESET INTERRUPT

BRI
*m,x

F900



DEFINITION

This instruction resets the highest active interrupt level and branches to the address indicated.

When coded indirect, this instruction causes the target PSW or PSD to be loaded into the CPU, resets the highest active interrupt level, and branches to the address in the PSW or PSD.

CONDITION CODE RESULTS

CC1: ISI if (I) is equal to one and (EWL_1) is equal to one.
CC2: ISI if (I) is equal to one and (EWL_2) is equal to one.
CC3: ISI if (I) is equal to one and (EWL_3) is equal to one.
CC4: ISI if (I) is equal to one and (EWL_4) is equal to one.

Assembly Language Coding: BRI X'(Branch Addr)'

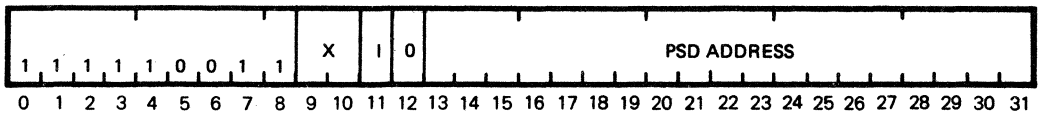
NOTES

1. Used only with interrupts operating in Active mode.
2. Privileged instruction.
3. If granularity of PSD is MAP, the contents of the MAP are changed in accord with the instructions in PSD word 2.
4. This instruction cannot be used with Post-indexing.

LPSD
*m,x

LOAD PROGRAM STATUS DOUBLEWORD

F980



DEFINITION — Causes the PSD addressed by the instruction to be loaded into the Program Status Doubleword Registers.

SUMMARY EXPRESSION — (EDL) → (PSDR)

CONDITION CODE RESULTS
CC1: Changed by the PSD being loaded
CC2: Changed by the PSD being loaded
CC3: Changed by the PSD being loaded
CC4: Changed by the PSD being loaded

Assembly Language Coding: LPSD X'(PSD Addr)'

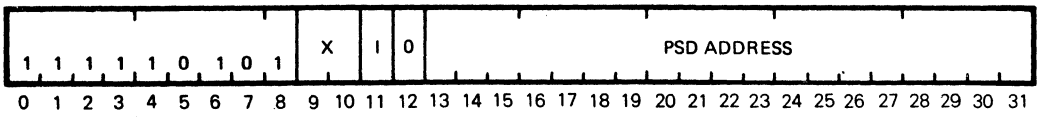
NOTES

1. Privileged instruction.
2. Causes system to go Mapped or Unmapped in accordance with codes in PSD that is being loaded.
3. This instruction does not modify contents of the MAP.
4. Attempt to execute this instruction in PSW mode will result in an undefined instruction trap.
5. The Block External Interrupts will be changed in accord with bits 48 and 49 of the PSD.

LOAD PROGRAM STATUS DOUBLEWORD AND CHANGE MAP

LPSDCM
d,*m,x

FA80



DEFINITION

Causes the PSD addressed by the instruction to be loaded into the Program Status Doubleword Registers, and the MAP to be loaded in accord with the BPIX and CPIX contents of the PSD. If the PSD defines the mapped condition, this instruction will cause the CPU to go mapped.

SUMMARY EXPRESSION

(EDL) → (PSDR)
(MIDL) → Map Registers

CONDITION CODE RESULTS

CC1: Changed by the PSD being loaded
CC2: Changed by the PSD being loaded
C3: Changed by the PSD being loaded
CC4: Changed by the PSD being loaded

Assembly Language Coding: LPSDCM X'(PSD Addr)'

NOTES

1. Privileged instruction.

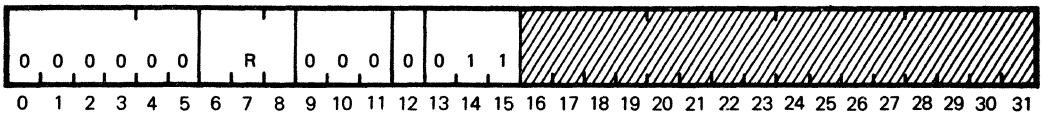
2. The Block External Interrupts will be changed in accord with bits 48 and 49 of the PSD.

3. Attempt to execute this instruction in PSW mode will result in an undefined instruction trap.

LCS
d

LOAD CONTROL SWITCHES

0003



DEFINITION — The contents of Control Switches (CSW) 0-15 are transferred to bit positions 0-15 of the GPR specified by R. Bit positions 16-31 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

$(CS_{0-15}) \rightarrow R_{0-15}$

$0 \rightarrow R_{15-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: ISI (R_{0-31}) is greater than zero
CC3: ISI (R_{0-31}) is less than zero
CC4: ISI (R_{0-31}) is equal to zero

EXAMPLE

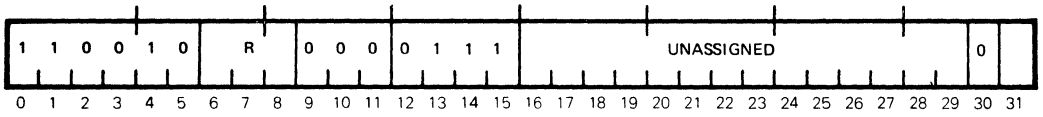
Memory Location:	06002
Hex Instruction:	03 83 (R=7)
Assembly Language Coding:	LCS 7

	PSWR	GPR7	
Before Execution	00006002	FFFFFFFF	Control Switches 0, 6 set
After Execution	10006004	82000000	

Note — Bit positions 0 and 6 of GPR7 are set and all other bits are cleared. CC3 is set.

EXECUTE REGISTER

C807



DEFINITION The word in the GPR specified by R is transferred to the Instruction register to be executed as the next instruction. If this instruction is not a Branch, the next instruction executed (following execution of the instruction in register R) is in the sequential memory location following the EXR instruction. If the GPR specified by R does contain a Branch instruction, the Program Status Word Register (PSWR) is changed accordingly.

NOTES

1. If two halfword instructions are in the GPR specified by R, only the left halfword instruction is executed.
2. An Unimplemented Instruction trap is generated if an EXR instruction attempts to execute an Unimplemented instruction or another Execute instruction.
3. The "PSD mode only" instructions cannot be targets of EXR, EXRR, or EXM.

SUMMARY EXPRESSION (R) → I

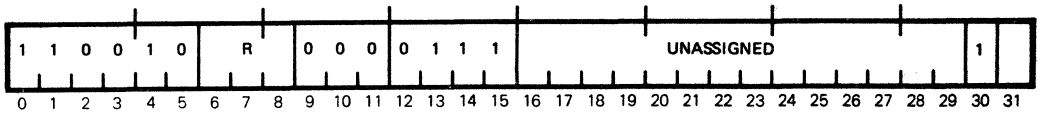
CONDITION CODE RESULTS Defined by the executed instruction.

Assembly Language Coding: EXR R

EXRR
S

EXECUTE REGISTER RIGHT

C807



DEFINITION

The contents of the least significant halfword (bits 16-31) of the GPR specified by R are transferred to the most significant halfword position (bits 0-15) of the Instruction register to be executed as the next instruction. If this halfword instruction is not a Branch, the next instruction executed (following execution of the halfword instruction transferred to the Instruction register) is in the sequential memory location following the EXRR instruction. If the instruction transferred to the Instruction register is a Branch instruction, the Program Status Word Register (PSWR) is changed accordingly.

NOTE

1. An unimplemented Instruction trap is generated if an EXRR instruction attempts to execute an Unimplemented instruction or another Execute instruction.

2. The "PSD mode only" instructions cannot be targets of EXR, EXRR, or EXM.

SUMMARY EXPRESSION

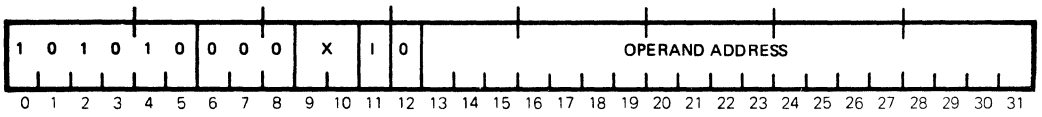
$(R_{16-31}) \rightarrow I_{0-15}$

CONDITION CODE RESULTS

Defined by the executed instruction.

Assembly Language Coding: EXRR R

A800



DEFINITION

The word in memory specified by the Effective Address (EA) is accessed and executed as the next instruction. If this instruction is not a Branch, the next instruction executed (following execution of the instruction specified by the EA) is in the next sequential memory location following the EXM instruction. If the instruction in memory specified by the EA is a Branch instruction, the Program Status Word Register (PSWR) is changed accordingly.

NOTES

1. If two halfword instructions are in the memory location specified by the EA, bit 30 of the EA determines which halfword instruction is executed. When bit 30 equals zero, the left halfword is executed. When bit 30 equals one, right halfword is executed.

2. An Unimplemented Instruction trap is generated if an EXM instruction attempts to execute an Unimplemented instruction or another Execute instruction.

3. The "PSD mode only" instructions cannot be targets of EXR, EXRR, or EXM.

SUMMARY EXPRESSION

$(EWL_{0-31}) \rightarrow I$, if $EA_{30}=0$

$(EWL_{16-31}) \rightarrow I$, if $EA_{30}=1$

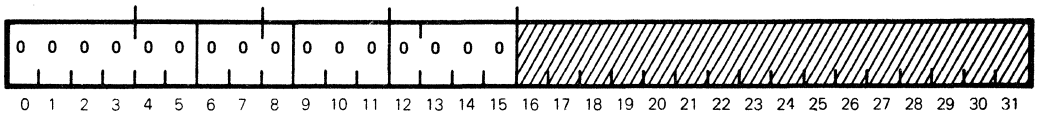
CONDITION CODE RESULTS

Defined by the executed instruction.

Assembly Language Coding: EXM X'(Op Addr)'

HALT

0000



DEFINITION

The execution of this instruction causes computer operation to be stopped. This includes input/output transfers and the servicing of priority interrupts. I/O in progress will be completed, but no interrupts will be serviced. Leaving a HALT condition requires depressing the RUN/HALT switch on the Systems Control Panel.

CONDITION CODE RESULTS

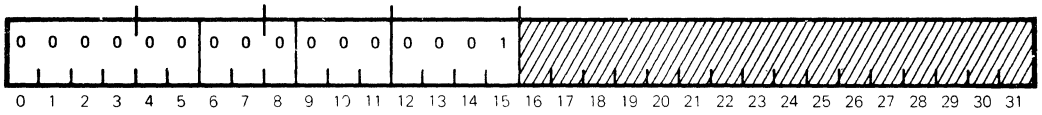
CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: HALT

NOTE

This is a privileged instruction.

0001



DEFINITION

The execution of this instruction causes the CPU to enter the Idle mode and lights the Wait indicator on the System Control Panel. Input/output transfers and priority interrupt servicing continue. If an interrupt occurs during a Wait condition, a return to the Wait occurs after the interrupt is serviced.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: WAIT

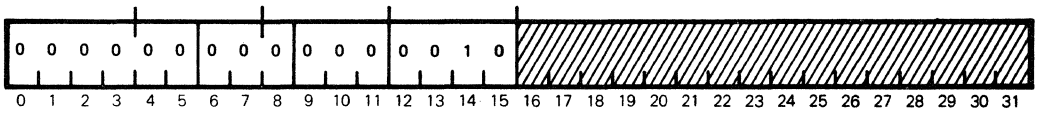
NOTE

If there is an attempt to execute a WAIT with interrupts blocked, a Block Mode Timeout Trap will be generated.

NOP

NO OPERATION

0002



DEFINITION

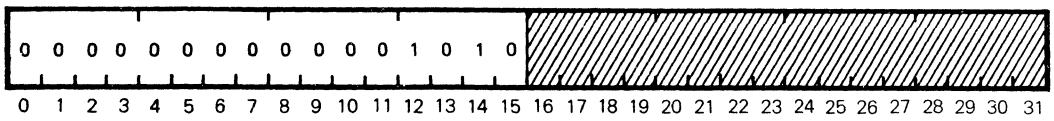
The Assembler uses the No Operation instruction to pad a halfword instruction which forces the next instruction to start on a word boundary, if the next instruction is a word instruction. It is also used whenever there is a need for an executable instruction that does not alter the machine status.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: NOP

000A



DEFINITION INTRODUCTION

This instruction is a control class unprivileged instruction used to start and stop the operation of the Internal Processing Unit. When the SIPU instruction is executed, this instruction functions as a START IPU instruction in the CPU and a STOP IPU instruction in the IPU.

START IPU TRAP 2E4

To start IPU processing, the CPU stores the new Program Status Doubleword (PSD) into words 3 and 4 of the Start IPU Trap Context Block which is pointed to by the address contained in the Start IPU trap vector location 2E4. The CPU then executes the SIPU X'000A' instruction which sends a start signal to the IPU and informs the IPU that a new PSD is available for execution. The IPU stores the old PSD into words 1 and 2 of the Start IPU Trap Context Block and IPU Status into word 5. The IPU then fetches the new PSD words 3 and 4 from the context block and begins to execute the instructions in memory as directed by the new PSD.

STOP IPU TRAP 2F4

To stop the IPU processing, the CPU stores a new PSD in words 3 and 4 of the Stop IPU Trap Context Block (TCB) which is pointed to by the address contained in the Stop IPU Trap vector location 2F4. The Stop IPU Trap Context Block (TCB) is used when the IPU then executes an SIPU (X'000A') instruction which is imbedded in the IPU software code. The IPU stores the old PSD into words 1 and 2 of the context block and the IPU Status into word 5 of the context block. The IPU then traps the CPU at location 2E0 which indicates that the IPU execution of the SIPU instruction has taken place. The IPU then fetches the new PSD from words 3 and 4 of the context block which can point to a privileged HALT or WAIT instruction to stop the IPU.

The new PSD in the STOP IPU context block may direct the IPU to execute code other than a HALT or WAIT instruction. This utilization of the Stop Trap allows the IPU to signal the CPU at milestones without stopping IPU execution. In either use of this stop IPU trap, the old PSD is stored into words 1 and 2 of TCB and the ending IPU status into word 5. The IPU DONE signal is sent to the CPU after storage of the old PSD and IPU status word and before vectoring to the new PSD address.

CONDITION CODE RESULTS

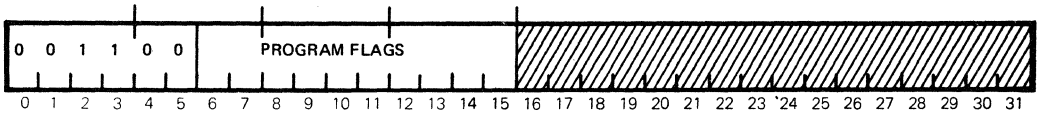
No change.

Assembly Language Coding: SIPU

CALM
V

CALL MONITOR

3000



DEFINITION

The execution of this instruction causes an interrupt request signal to be applied to interrupt priority 27_{16}. Bit positions 6-15 of the Instruction Word may be used to contain program flags which can be examined by the interrupt service routine.

CONDITION CODE

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: CALM PROGRAM. FLAGS

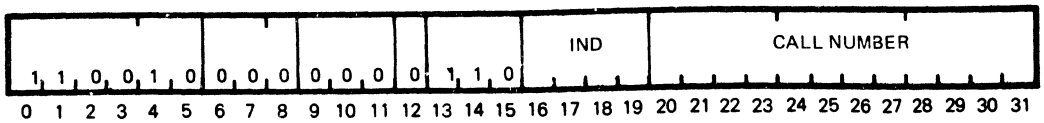
NOTES

1. Interrupt level 27 must be enabled prior to execution of this instruction.

2. This instruction must not be executed with a higher priority level active.

SUPERVISOR CALL

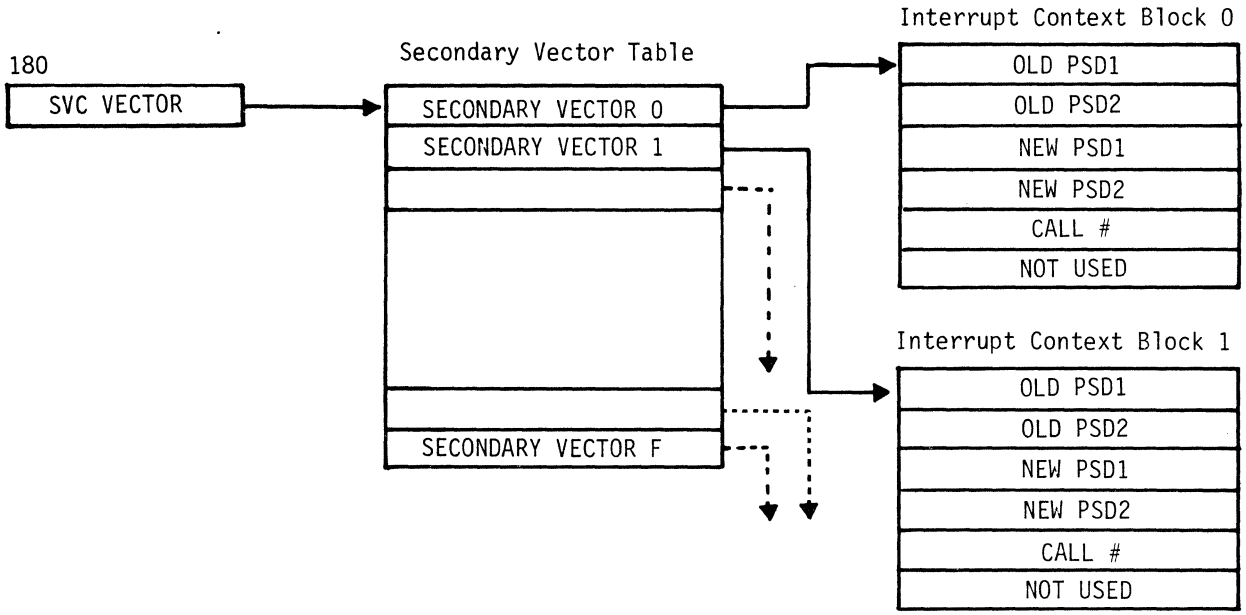
C806



DEFINITION

The execution of this instruction causes a pseudo-trap to the trap/interrupt vector for relative priority level 6. Bits 16-19 may be used to index the interrupt vector (location 180) with up to 16 locations. This index vector address will point to a SVC vector table whose content will point to the trap subroutine.

Bits 20-31 are used for the call number. This call number serves as an identifier parameter for the software use.



CONDITION CODE RESULTS

CC1: zero
CC2: zero
CC3: zero
CC4: zero

Assembly Language Coding: SVC IND, CALL#

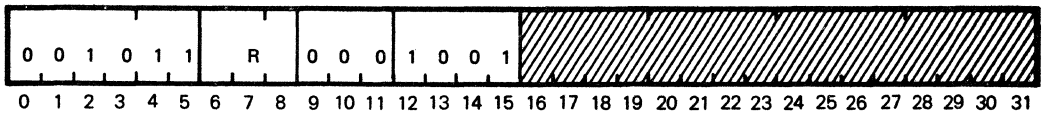
NOTE

The CPU must have previously been set to PSD mode. Otherwise, an Undefined Instruction Trap will occur.

SETCPU
S

SET CPU MODE

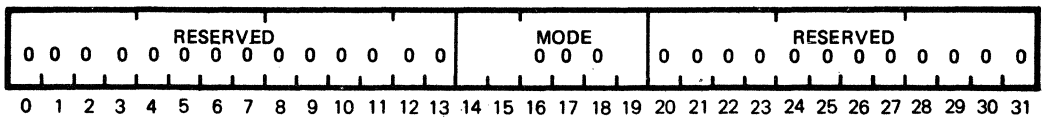
2C09



DEFINITION

The execution of this instruction causes the operating characteristic of the CPU to change to the mode specified by the contents of R.

The contents of R will be:



Bits 0-13 Must be zeros and reserved for future use.

Bit 14 Enable Block Mode Timeout Trap.

Bit 15 Enable PSD Traps (m/c halts if not enabled)

Bit 16-18 Reserved (must be zero).

Bit 19 0=PSW mode 55
1=PSD mode 75

CONDITION CODE

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: SETCPU S

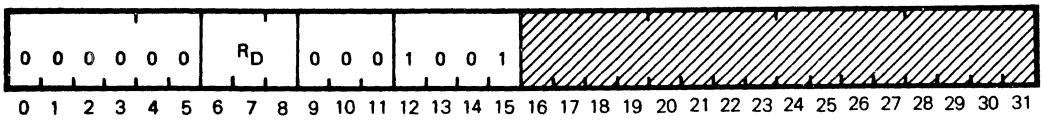
NOTE

The PSD mode of operation must be enabled (allowed) by way of a hardware jumper on the C Board, or an Undefined Instruction Trap will occur.

READ CPU STATUS WORD

RDSTS
d

0009



DEFINITION

This instruction places the CPU Status Word into Register R_D. The source of the CPU Status Word is location 91_H in the CPU Scratchpad. After reporting status, bits 00-23 of the Status Word (in the Scratchpad) are set to zero. Bits 24-31 of the Scratchpad Status Word remain unchanged. The CPU Status Word in Register R_D is defined as follows:

Bit

0 =0, CLASS 0,1,2, OR E ERROR
=1, CLASS F (EXTENDED I/O) ERROR
1 =0, I/O PROCESSING ERROR
=1, INTERRUPT PROCESSING ERROR
2 FINAL BUS TRANSFER ERROR
3 BUS NO RESPONSE ERROR
4 I/O CHANNEL BUSY OR BUSY STATUS BIT ERROR
5 READY TIMEOUT ERRO
6 I/O DRT TIMEOUT ERROR
7 RETRY COUNT EXHAUSTED ERROR
8 OPERAND FETCH PARITY ERROR
9 INSTRUCTION FETCH PARITY ERRO
10 OPERAND NONPRESENT ERROR
11 INSTRUCTION NONPRESENT ERROR
12 UNDEFINED PSD MODE INSTRUCTION ERROR
13 MEMORY FETCH DRT TIMEOUT ERROR
14 RESET CHANNEL ERROR
15 CHANNEL WCS NOT ENABLED ERROR
16 MAP NOT FOUND
MAP REGISTER ADDRESS OVERFLOW (MAP CONTEXT SWITCH)
17 UNEXPLAINED MEMORY ERROR
18 BRI I/O ERROR
19 UNDEFINED INSTRUCTION PSW MODE ONLY
20 MAP INVALID ACCESS OR MAP MODE RESTRICTION ERROR
21 IPL I/O OR MEMORY ERROR FLAG
22 CPU WCS NOT PRESENT ERROR
23 NOT USED
24 ENABLE ARITHMETIC EXCEPTION TRAP
25 DISABLE PSD MODE TRAPS
26 BLOCK MODE IS ACTIVE
27 CPU POWER FAIL UP MEMORY ERROR
28 NOT USED
29 NOT USED
30 NOT USED
31 =0, CPU MODE PSW 55
=1, CPU MODE PSD 75

CONDITION CODE RESULTS

CC1: Not used
CC2: ISI PSD mode
CC3: ISI interrupts are blocked
CC4: ISI R_D bits 0-23 equal zero

Assembly Language Coding: RDSTS R_D

NOTES

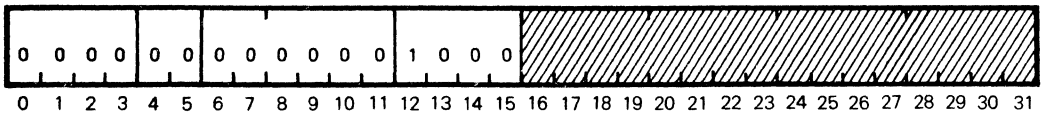
1. This instruction is a Privileged Halfword instruction.

2. This instruction may not be the target of an Execute instruction.

3. The PSD mode of operation must be enabled (allowed) by way of a hardware jumper on the C-board, or an undefined instruction trap will occur.

ENABLE ARITHMETIC EXCEPTION TRAP

0008



DEFINITION

Sets bit 7 of PSD to enable Arithmetic Exception Trap.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: EAE

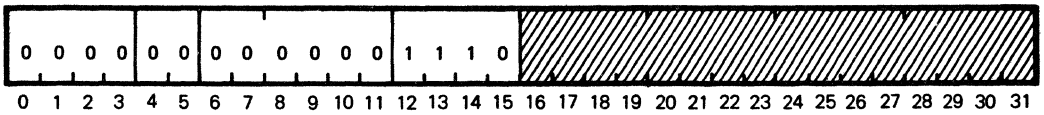
NOTES

1. Halfword Instruction.

2. Attempt to execute this instruction in PSW mode will result in an Undefined Instruction Trap.

DAE DISABLE ARITHMETIC EXCEPTION TRAP

000E



DEFINITION Resets bit 7 of PSD to disable Arithmetic Exception Trap.

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

Assembly Language Coding: DAE

NOTES
1. Halfword Instruction.
2. Attempt to execute this instruction in 55 mode will result in an Undefined Instruction Trap.

INTERRUPT INSTRUCTIONS

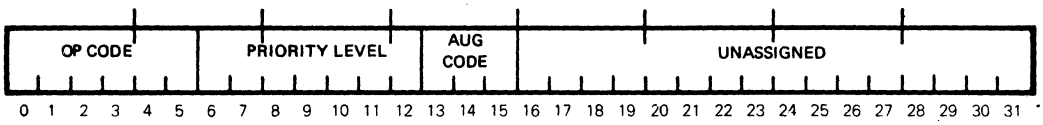
GENERAL DESCRIPTION

The Interrupt Control instruction group provides the availability to permit selective Enable, Disable, Request, Activate, and Deactivate operations to be performed on any addressed interrupt level. These instructions can only be executed when bit 0 of the PSWR equals one (Privileged State).

INSTRUCTION FORMATS

The following instruction format is used for all Interrupt Control operations: (Trap/Interrupt priorities are shown in Table 6-3.)

INTERRUPT CONTROL



Bits 0-5 define the Operation Code.

Bits 6-12 define the binary priority level number of the interrupt being commanded.

Bits 13-15 define the Augmenting Operation Code.

Bits 16-31 unassigned.

CONDITION CODE UTILIZATION

All Interrupt Control instructions leave the current Condition Code unchanged.

Table 6-3. 32/70 Series Relative Trap/Interrupt Priorities

INTERRUPT AND TRAP RELATIVE PRIORITY	INTERRUPT LOGICAL PRIORITY	INTERRUPT VECTOR LOCATION (IVL)	TCW ADDRESS **	IOCD ADDRESS **	DESCRIPTION
00		0F4			Power Fail Safe Trap
01		0FC			System Override Trap (Not Used)
02		0E8*			Memory Parity Trap
03		190			Nonpresent Memory Trap
04		194			Undefined Instruction Trap
05		198			Privilege Violation Trap
06		180			Supervisor Call Trap
07		184			Machine Check Trap
08		188			System Check Trap
09		18C			Map Fault Trap
0A					Not Used
0B					Not Used
0C					Not Used
0D					Not Used
0E		0E4			Block Mode Timeout Trap
0F		1A4*			Arithmetic Exception Trap
10	00	0F0			Power Fail Safe Interrupt
11	01	0F8			System Override Interrupt
12	12	0E8*			***Memory Parity Trap
13	13	0EC			Attention Interrupt
14	14	140	100	700	I/O Channel 0 Interrupt
15	15	144	104	708	I/O Channel 1 Interrupt
16	16	148	108	710	I/O Channel 2 Interrupt
17	17	14C	10C	718	I/O Channel 3 Interrupt
18	18	150	110	720	I/O Channel 4 Interrupt
19	19	154	114	728	I/O Channel 5 Interrupt
1A	1A	158	118	730	I/O Channel 6 Interrupt
1B	1B	15C	11C	738	I/O Channel 7 Interrupt
1C	1C	160	120	740	I/O Channel 8 Interrupt
1D	1D	164	124	748	I/O Channel 9 Interrupt
1E	1E	168	128	750	I/O Channel A Interrupt
1F	1F	16C	12C	758	I/O Channel B Interrupt
20	20	170	130	760	I/O Channel C Interrupt
21	21	174	134	768	I/O Channel D Interrupt
22	22	178	138	770	I/O Channel E Interrupt
23	23	17C	13C	778	I/O Channel F Interrupt
24	24	190*			***Nonpresent Memory Trap
25	25	194*			***Undefined Instruction Trap
26	26	198*			***Privilege Violation Trap
27	27	19C			Call Monitor Interrupt
28	28	1A0			Real-Time Clock Interrupt
29	29	1A4*			***Arithmetic Exception Interrupt
2A	2A	1A8			External/Software Interrupts
2B	2B	1AC			External/Software Interrupts
2C	2C	1B0			External/Software Interrupts

Table 6-3. 32/70 Series Relative Trap/Interrupt Priorities (Cont'd)

INTERRUPT AND TRAP RELATIVE PRIORITY	INTERRUPT LOGICAL PRIORITY	INTERRUPT VECTOR LOCATION (IVL)	TCW ADDRESS **	IOCD ADDRESS **	DESCRIPTION
2D	2D	1B4			External/Software Interrupts
2E	2E	1B8			External/Software Interrupts
2F	2F	1BC			External/Software Interrupts
30	30	1C0			External/Software Interrupts
31	31	1C4			External/Software Interrupts
THROUGH	THROUGH	THROUGH			THROUGH
77	77	2DC			External/Software Interrupts
78		2E0****			Ending of IPU Processing Trap (Used by CPU)
79		2E4****			Start IPU Processing Trap (Used by IPU)
7A		2E8****			Supervisor Call Trap (Used by IPU)
7B		2EC****			Error Trap (Used by IPU)
7C		2F0****			Call Monitor Trap (Used by IPU)
7D	7D	2F4****			Stop IPU Processing Trap (Used by IPU)
7E	7E	2F8			External/Software Interrupts
7F	7F	2FC			External/Software Interrupts

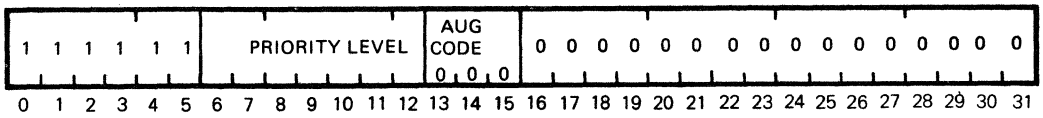
* Vector Locations Shared With Traps
** For Nonextended I/O Devices
*** PSW Function - Now External/Software Interrupts - For PSD Mode.
**** IPU Related Traps (See Section II)

All Interrupts Are Externally Generated

EI
V

ENABLE INTERRUPT

FC00



DEFINITION

If bit position 0 of the PSWR is equal to one (Privileged State), the priority interrupt level specified by the priority level field (bits 6-12) in the Instruction Word (IW) is conditioned to respond to an interrupt signal. If bit position 0 of the PSWR is equal to zero (Unprivileged State), execution of this instruction will generate the Privileged Violation trap.

NOTES

1. This instruction does not operate with priority levels 2_{16} - 11_{16}.
2. Any stored requests for the specified level are eligible to become active.
3. In the PSD mode, traps are always enabled.
4. This instruction has no affect on levels assigned to Class F I/O and is treated as NOP.
5. For levels 0 and 1, the RTOM jumpers provide either constant enable or software enable/disable.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex)
0010010	12
0010011	13
0010100	14
-	-
-	-
-	-
-	-
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

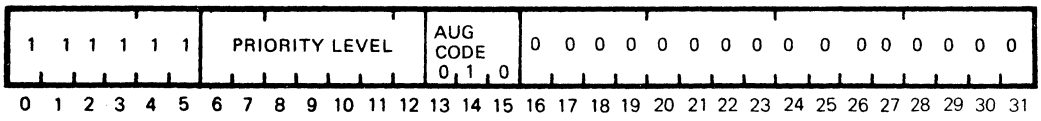
EI ,LEVEL

NOTE

Any stored requests for the specified level are eligible to become active.

REQUEST INTERRUPT

FC02



DEFINITION

If bit position 0 of the PSWR is equal to one (Privileged State), an interrupt request signal is applied to the interrupt level specified by the priority level field (bits 6-12) in the Instruction Word (IW). This signal simulates the signal generated by the internal or external condition connected to the specified level. If bit position 0 of the PSWR is equal to zero (Unprivileged State), execution of this instruction will generate the Privileged Violation Trap. The interrupt request signal is stored in the specified level whether or not it is enabled and/or active.

NOTES

1. This instruction does not operate with priority levels 2_{16} - 11_{16}.
2. For RI's on levels 0 or 1, the RTOM jumpers select either that levels 0 and 1 are enabled, or that software enables are required.
3. This instruction has no affect on levels assigned to Class F I/O and is treated as NOP.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex)
0000000	00
0000001	01
0010010	12
-	-
-	-
-	-
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

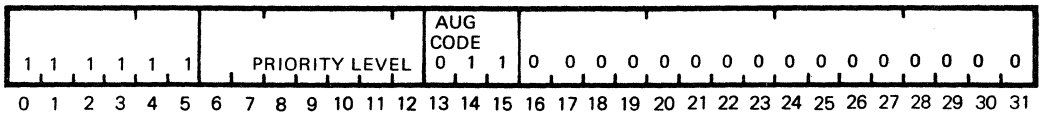
ASSEMBLY LANGUAGE CODING

RI LEVEL

AI
V

ACTIVATE INTERRUPT

FC03



DEFINITION

If bit position 0 of the PSWR is equal to one (Privileged State), a signal is applied to set the active condition in the priority interrupt level specified by the priority level field (bits 6-12) in the Instruction Word (IW). The active level is set in the specified level whether or not that level is enabled. This condition prohibits this level and any lower levels not already in service from being serviced until this level is deactivated. However, request signals occurring at this or lower levels are stored for subsequent servicing. If bit position 0 of the PSWR is equal to zero (Unprivileged State), execution of this instruction will generate the Privileged Violation Trap.

NOTES

1. This instruction does not operate with priority levels 2_{16} - 11_{16}.
2. This instruction has no affect on levels assigned to Class F I/O and is treated as NOP.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex)
0000000	00
0000001	01
0010010	12
-	-
-	-
-	-
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

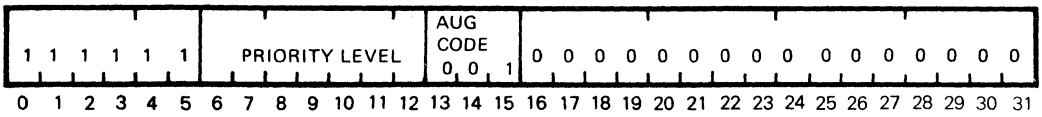
ASSEMBLY LANGUAGE CODING

AI LEVEL

DISABLE INTERRUPT

DI
V

FC01



DEFINITION

If bit position 0 of the PSWR is equal to one (Privileged State), the priority interrupt level specified by the priority level field (bits 6-12) in the Instruction Word (IW) is disabled and will not respond to an interrupt signal. If bit position 0 of the PSWR is equal to zero (Unprivileged State), execution of this instruction will generate the Privileged Violation Trap. The active state of the interrupt is not affected.

NOTES

1. Any unserviced request signal at this level is cleared by execution of this instruction.
2. This instruction does not operate with priority levels 2_{16} - 11_{16}.
3. In the PSD mode, traps are always enabled.
4. This instruction has no affect on levels assigned to Class F I/O and is treated as NOP.
5. For levels 0 and 1, the RTOM jumpers provide either constant enable or software enable/disable.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex)
0010010	12
0010011	13
0010100	14
-	-
-	-
-	-
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

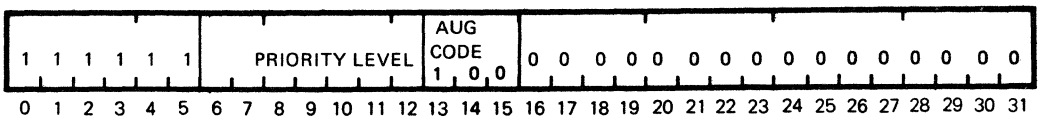
ASSEMBLY LANGUAGE CODING

DI LEVEL

DAI
V

DEACTIVATE INTERRUPT

FC04



DEFINITION

If bit position 0 of the PSWR is equal to one (Privileged State), a signal is applied to reset the active condition for the priority interrupt level specified by the priority level field (bits 6-12) in the Instruction Word. The specified level is set inactive whether the level is enabled or disabled. Execution of the Deactivate Interrupt instruction does not clear any request signals on the specified level or any other level. If bit position 0 of the PSWR is equal to zero (Unprivileged State), execution of this instruction will generate the Privileged Violation Trap.

NOTE

1. This instruction does not operate with priority levels 2_{16} - 11_{16}.
2. This instruction has no affect on levels assigned to Class F I/O and is treated as a NOP.
3. In PSD mode, DAI and the following instruction are executed as an uninterruptible pair.
4. Using a Deactivate Interrupt and then LPSD (Load Program Status Doubleword) or a Deactivate Interrupt and then LPSDCM, is preferable to using a BRI (faster)

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex)
0000000	00
0000001	01
0010010	12
-	-
-	-
-	-
1111110	7E
1111111	7F

CONDITION CODE RESULTS

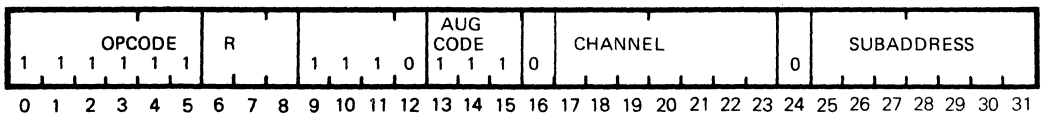
CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

DAI LEVEL

ACTIVATE CHANNEL INTERRUPT

FC77



DEFINITION

The Activate Channel Interrupt will cause the addressed channel to begin actively contending with other interrupt levels, causing a blocking of its level, and all lower priority levels, from requesting an interrupt. If a request is currently pending in the channel, the request interrupt is removed but the interrupt level remains in contention.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress field to form the logical channel and subaddress.

Bits 9-12 specify the operation as an ACI, hex E.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero only, constant will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
ACI   R,'(Constant)'
```

NOTES

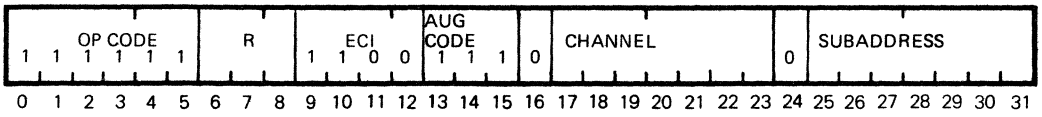
1. Condition Codes, after execution of the ACI,, will be set and can be tested by a subsequent BCT or BCF to determine if the ACI was accepted by the channel.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

ECI
S,V

ENABLE CHANNEL INTERRUPT

FC67



DEFINITION

The Enable Channel Interrupt causes the addressed channel to be enabled to request interrupts from the CPU.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress field to form the logical channel and subaddress.

Bits 9-12 specify the operation as ECI, ^{hex} C.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero only constant will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations, refer to the Class F Condition Codes on Page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
ECI   R,'(Constant)'
```

NOTES

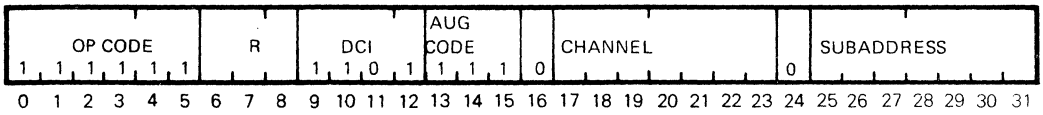
1. Condition Codes after execution of the ECI will be set and can be tested by a subsequent BCT or BCF to determine if the ECI was accepted by the channel.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

DISABLE CHANNEL INTERRUPTS

DCI
S,V

FC6F



DESCRIPTION

The Disable Channel Interrupt causes the addressed channel to be disabled from requesting interrupts from the CPU.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress field to form the logical channel and subaddress.

Bits 9-12 specify the operation as DCI, hex D.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only constant will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
DCI   R,'(Constant)'
```

NOTES

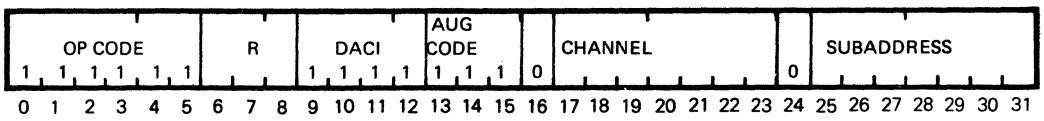
1. Condition Codes after execution of the DCI will be set and can be tested by a subsequent BCT or BCF to determine if the DCI was accepted by the channel.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

DACI
S,V

DEACTIVATE CHANNEL INTERRUPT

FC7F



DEFINITION

The Deactivate Channel Interrupt will cause the addressed channel to remove its interrupt level from contention. If a request interrupt is currently queued, the deactivate will cause the queued request to actively request if the channel is enabled.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as DACI, hex F.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only constant will be used to specify the logical channel and subaddress.

CONDITION CODE

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations, refer to the Class F Condition Codes on page 6-214 of this manual.

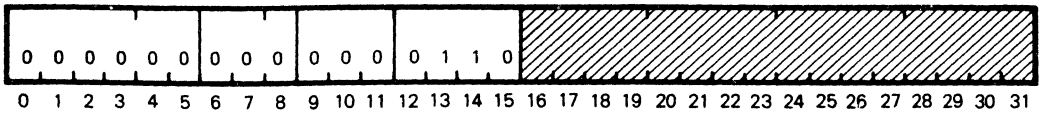
ASSEMBLY LANGUAGE CODING

```
DACI   R,'(Constant)'
```

NOTES

1. Condition Codes after execution of the DACI will be set and can be tested by a subsequent BCT or BDF to determine if the DACI was successfully executed.
2. On PSD mode, the DACI and following instructions are executed as an uninterruptible pair.
3. Using Deactivate Channel Interrupt and LPSD or Deactivate Channel Interrupt and LPSDCM is preferable to using a BRI.
4. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.
5. All DACI instruction abnormalities or I/O protocol violations are connected to the System Check Trap unless an initial channel nonpresent or inoperable condition is found.

0006



DEFINITION

The execution of this instruction prevents the CPU from sensing all interrupt requests generated by the I/O channel and RTOM.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

BEI

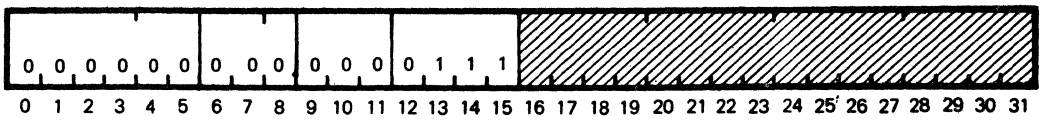
NOTE

The CPU must have previously been set to PSD mode.

UEI

UNBLOCK EXTERNAL INTERRUPTS

0007



DEFINITION

The execution of this instruction causes the CPU to sense all interrupt requests generated by the I/O channel and RTOM.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

UEI

NOTE

The CPU must have previously been set to PSD mode.

INPUT/OUTPUT INSTRUCTIONS

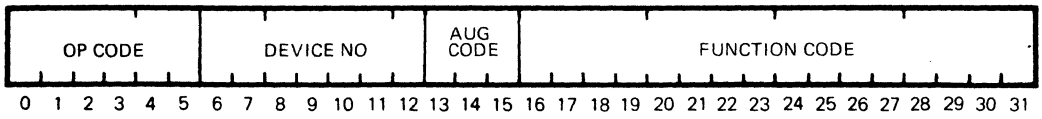
GENERAL DESCRIPTION

The Input/Output instructions provide the capability to perform Command or Test operations to attached peripheral devices. Both the Command Device and the Test Device instructions cause a 16-bit function code to be sent to the device specified by the instruction.

INSTRUCTION FORMATS

The following instruction format is used by both Input/Output instructions.

INPUT/OUTPUT



Bits 0-5 define the Operation Code.

Bits 6-12 designate the device number.

Bits 13-15 define the Augmenting Operation Code.

Bits 16-31 contain the 16-bit function code.

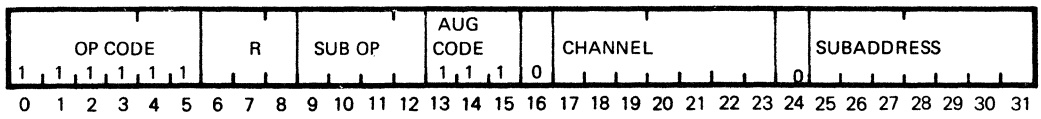
CONDITION CODE UTILIZATION

The Condition Code is set during execution of a Test Device instruction to indicate the result of the test being performed. The Command Device instruction leaves the current Condition Code unchanged.

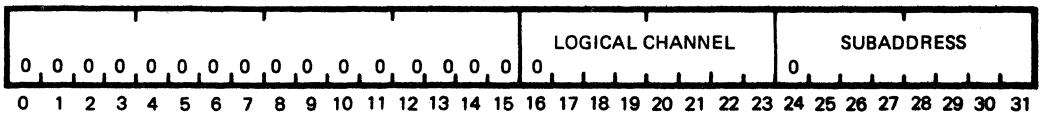
CLASS F I/O INSTRUCTIONS

INSTRUCTION FORMAT

All Class F I/O instructions will be in the following format:



Op Code bits 0-5 and Aug Code bits 13-15 must contain ones. The R field (bits 6-8), if nonzero, specifies the general register whose contents will be added to the channel and subaddress field bits 16-31 to form the logical channel and subaddress. If R is specified as zero, only the channel and subaddress fields will be used. The format of the computed logical channel and subaddress is:



The subaddress will be ignored by the channel if the operation does not apply to a controller or device.

The sub op field bits 09-12 specify the type of operation that is to be performed as described below:

BITS 09-12	SUB OP
0 0 0 0 - X'0'	Unassigned
0 0 0 1 - X'1'	Unassigned
0 0 1 0 - X'2'	START I/O (SIO)
0 0 1 1 - X'3'	TEST I/O (TIO)
0 1 0 0 - X'4'	STOP I/O (STPIO)
0 1 0 1 - X'5'	RESET CHANNEL (RSCHNL)
0 1 1 0 - X'6'	HALT I/O (HIO)
0 1 1 1 - X'7'	GRAB CONTROLLER (GRIO)
1 0 0 0 - X'8'	RESET CONTROLLER (RSCTL)
1 0 0 1 - X'9'	ENABLE WRITE CHANNEL WCS (ECWCS)
1 0 1 0 - X'A'	Unassigned
1 0 1 1 - X'B'	WRITE CHANNEL WCS (WCWCS)
1 1 0 0 - X'C'	ENABLE CHANNEL INTERRUPT (ECI)
1 1 0 1 - X'D'	DISABLE CHANNEL INTERRUPT (DCI)
1 1 1 0 - X'E'	ACTIVATE CHANNEL INTERRUPT (ACI)
1 1 1 1 - X'F'	DEACTIVATE CHANNEL INTERRUPT (DACI)

NOTES

1. Channel must be ICL'd as Class F.
2. EXR, EXRR, and EXM may not be used.
3. Must be in PSD mode.
4. CCs must be tested after each instruction.
5. CD, TD, EI, DI, AI, DAI, and RI cannot be executed to Class F channel.

CLASS F CONDITION CODES

The condition codes will be set for the execution of all Class F I/O instructions and indicate the successful or unsuccessful initiation of an I/O instruction. The condition codes can be set by the CPU, for channel busy and inoperable or undefined channel, or by the information passed directly from the channel. The assignments for the condition codes are:

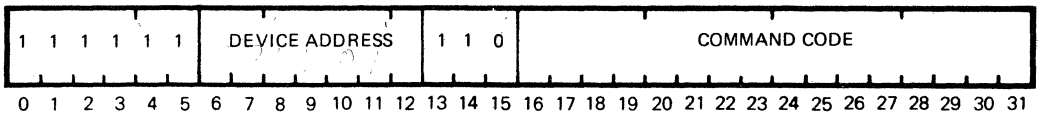
CC1	CC2	CC3	CC4	
0	0	0	0	REQUEST ACTIVATED, WILL ECHO STATUS
0	0	0	1	CHANNEL BUSY
0	0	1	0	CHANNEL INOPERABLE OR UNDEFINED
0	0	1	1	SUBCHANNEL BUSY
0	1	0	0	STATUS STORED
0	1	0	1	UNSUPPORTED TRANSACTION
0	1	1	0	UNASSIGNED
0	1	1	1	UNASSIGNED
1	0	0	0	REQUEST ACCEPTED AND QUEUED, NO ECHO STATUS
1	0	0	1	UNASSIGNED
1	0	1	0	UNASSIGNED
1	0	1	1	UNASSIGNED
1	1	0	0	UNASSIGNED
1	1	0	0	UNASSIGNED
1	1	1	0	UNASSIGNED
1	1	1	1	UNASSIGNED

Although 16 encoded condtions are possible, only the assigned patterns will occur.

CD
n,f

COMMAND DEVICE

FC06



DEFINITION

The contents of the Command Code field (bits 16-31) are transferred to the Device Controller Channel specified by the device address contained in bit positions 6-12 of the Instruction Word.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY EXAMPLE

	Dev Add	Comm Code	Command
CD	X'7A'	X'8000'	Output data to device 7A
CD	X'78'	X'9000'	Input data from device 78

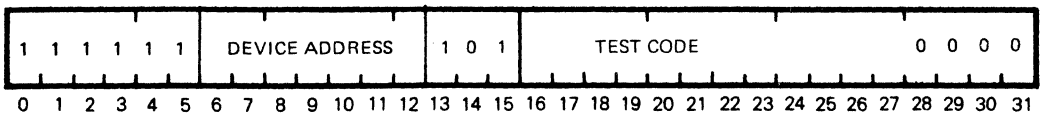
NOTES

1. Class 0,1,2,3, and E I/O Processor instruction only.
2. If the CPU is in the PSW mode and a CD instruction to a Class F channel is attempted, a No Operation (NOP) will be executed instead.
3. If the CPU is in the PSD mode and a CD instruction to a Class F channel is attempted, a System Check Trap will occur.

TEST DEVICE

TD
n,f

FC05



DEFINITION

The contents of the Test Code field (bits 16-27) are transferred to the Device Controller Channel (DCC) specified by the device address contained in bit positions 6-12 of the Instruction Word. The device test defined by the Test Code is performed in the DCC, and the test results are stored in Condition Code bits 1-4 (CC_{1-4}).

NOTE

A TD having a unique Test Code is available with most peripheral devices. Execution of a TD with this code causes a snapshot of all device and DCC status to be stored in memory. The individual peripheral device reference manuals define the operation of this instruction with each device.

CONDITION CODE RESULTS

Test results defined for specific peripheral device.

ASSEMBLY EXAMPLE

	Dev Add	Comm Code	Command
TD	X'10'	X'8000'	Request the Controller Status for unit 10
TD	X'10'	X'2000'	Request the Device status for unit 10

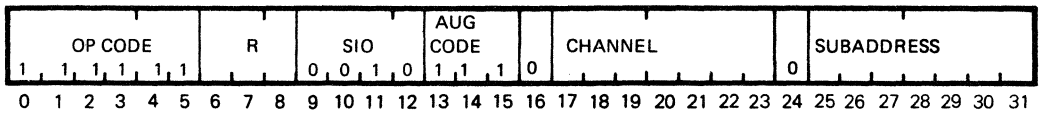
NOTES

1. Class 0,1,2,3, and E I/O Processor instruction only.
2. If the CPU is in the PSW mode and a TD instruction to a Class F channel is attempted, the following Condition Codes will be set:
   a. TD 8000 - CC3 (Channel Error)
   b. TD 4000 - CC3 (Program Violation
   c. TD 2000 - CC2 (Status Transfer Not Performed)
3. If the CPU is in the PSD mode and a TD instruction to a Class F channel is attempted, a System Check Trap will occur.

SIO
s,v

START I/O

FC17



DEFINITION

Start I/O will be used to begin I/O execution or to return appropriate Condition Codes and status if I/O execution could not be accomplished.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as an SIO, hex 2.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = (0000_2) or (1000_2)

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
SIO   R,'(Constant)'
```

NOTES

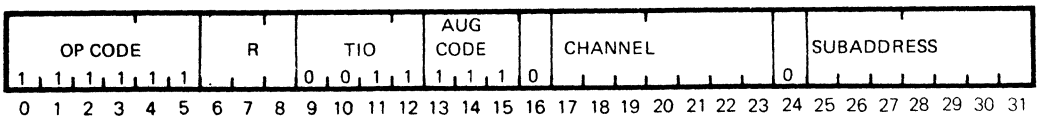
1. Condition Codes, after execution of an SIO, will be set and can be tested by a subsequent BCT or BCF to ascertain if the I/O was accepted.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

TEST I/O

TIO
S,V

FC1F



DEFINITION

Test I/O will be used to test controller state and to return appropriate Condition Codes and status reflecting the state of the addressed controller and/or device. Channel implementation will dictate the depth that the channel must test to determine current state.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as a TIO, hex 3.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 Specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on Page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
TIO   R,'(Constant)'
```

NOTES

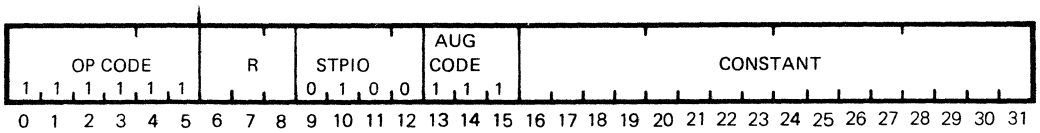
1. Condition Codes, after execution of the TIO, will be set and can be tested by a subsequent BCT or BCF to ascertain channel/controller/device state.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

STPIO
s,v

STOP I/O

FC27



DEFINITION

The STOP I/O (STPIO) is used to terminate the current I/O operation after the completion of the current IOCD. The STOP I/O applies only to the addressed subchannel, and the only function is to suppress command and data chain flags in the current IOCD.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as a STPIO, hex 4.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
STPIO   R,'(Constant)'
```

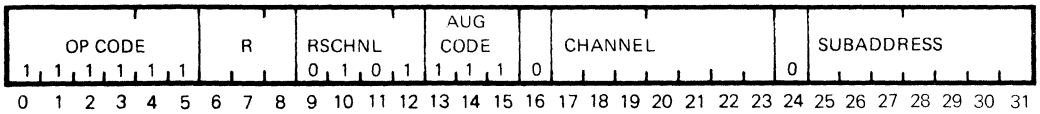
NOTES

1. Condition Codes, after execution of an STPIO, will be set and can be tested by a subsequent BCT or BCF to ascertain the channel/controller/device state.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

RESET CHANNEL

FC2F



DEFINITION

The Reset Channel (RSCHNL) causes the addressed channel to cease and reset all activity and to return to the idle state. The channle will also reset all subchannels. No controller or device will be affected. Any requesting or active interrupt level will be reset.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as a RSCHNL, hex 5.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condtion Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
RSCHNL   R,'(Constant)'
```

NOTES

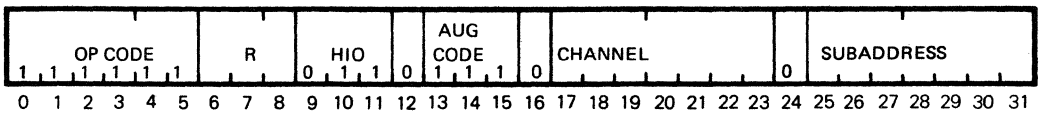
1. Condition Codes, after execution of a RSCHNL, will be set and can be tested by a subsequent BCT or BCF to ascertain the channel/controller/device state.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

HIO
s,v

HALT I/O

FC37



DEFINITION

Halt I/O (HIO) is used to cause an immediate but orderly termination in the controller. The Device End condition will notify the software of the actual termination in the controller; thus, indicating its availability for new requests. If the Halt I/O caused the generation of status relating to the terminated I/O operation, then the Device End condition for the termination of the I/O operation will be the only Device End condition generated.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as a HIO, hex 6.

Bits 13-15 specify the augment code, octal 7.

Bists 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
HIO   R,'(Constant)'
```

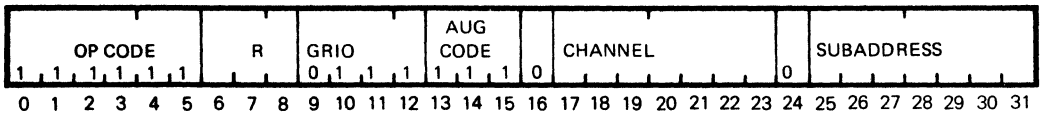
1. Condition Codes after execution of the HIO, will be set and be tested by a subsequent BCT or BCF to ascertain if the HIO was successfully executed.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

GRAB CONTROLLER

GRIO
S,V

FC3F



DEFINITION

The Grab Controller (GRIO) will cause the addressed controller to release itself from the currently assigned channel and to reserve itself for the grabbing channel.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as GRIO, hex 7.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
GRIO   R,'(Constant)'
```

NOTES

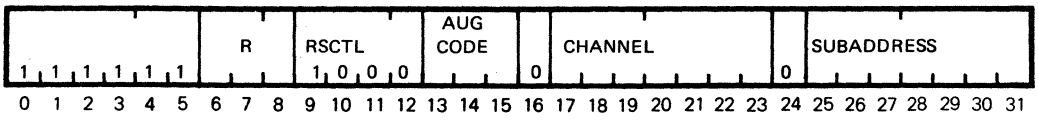
1. Condition Codes, after execution of the GRIO, will be set and can be tested by a subsequent BCT or BCF to determine if the GRIO was successfully executed.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

RSCTL
S,V

RESET CONTROLLER

FC47



DEFINITION

This instruction causes the addressed controller to be completely reset. In addition, the subchannel and all pending and generated status conditions are cleared.

Bits 0-5 specify the operation code, octal 77.

Bits 6-8 specify the General Purpose Register (R), when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 specify the operation as RSCTL, hex 8.

Bits 13-15 specify the augment code, octal 7.

Bits 16-31 specifies a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
RSCTL   R,'(Constant)'
```

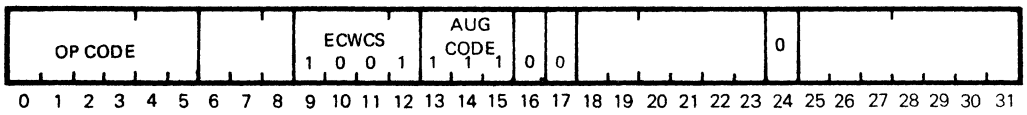
NOTE

If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

ENABLE CHANNEL WCS LOAD

ECWCS
S,V

FC4F



DEFINITION

The Enable Channel WCS Load (ECWCS) sets an interlock within the CPU to enable the loading of WCS. The ECWCS must be the first of a 2-instruction sequence.

Bits 0-5 Specify the operation code, octal 77.

Bits 6-8 Specify the general register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 Specify the operation as an ECWCS, hex 9.

Bits 13-15 Specify the augment code, octal 7.

Bits 16-31 Specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations, refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
ECWCS   R,'(Constant)'
```

NOTES

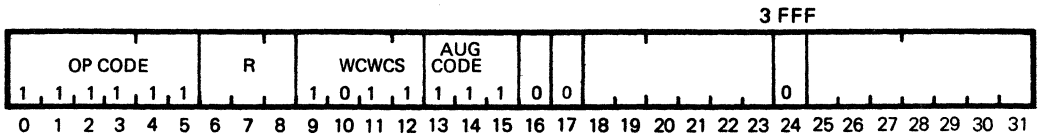
1. Condition Codes after the execution of the ECWCS instruction will be set and can be tested by a subsequent BCT or BCF to ascertain whether the ECWCS instruction was successfully executed.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

WCWCS
S,V

WRITE CHANNEL WCS

FC5F



DEFINITION

The Write Channel WCS (WCWCS) causes the loading of the channel WCS. The WCWCS must be the second instruction executed to the Class F I/O controller, the first being ECWCS, without any intervening I/O instructions to the Class F I/O controller to be loaded.

Bits 0-5 — Specify the operation code, octal 77.

Bits 6-8 — Specify the general register, when nonzero, whose contents will be added to the channel and subaddress fields to form the logical channel and subaddress.

Bits 9-12 — Specify the operation as a WCWCS, hex B.

Bits 13-15 — Specify the augment code, octal 7.

Bits 16-31 — Specify a constant that will be added to the contents of R to form the logical channel and subaddress. If R is zero, only bits 16-31 will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1, 2, 3, and 4 = $(0000)_2$ or $(1000)_2$

This indicates that the instruction was accepted. For other Condition Code combinations refer to the Class F Condition Codes on page 6-214 of this manual.

ASSEMBLY LANGUAGE CODING

```
WCWCS   R,'(Constant)'
```

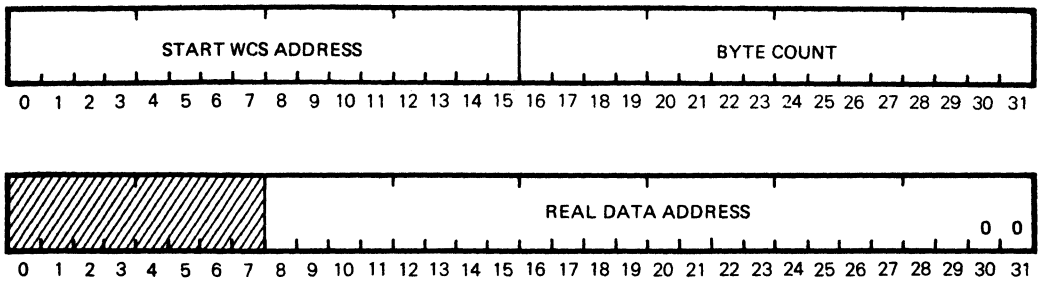
NOTES

1. The information that is required by the WCS load will be passed to the Class F I/O controller by a parameter list. The IOCD address location specified for this controller will be initialized by software prior to the execution of this instruction. The subaddress field will be ignored.

2. If this instruction is executed for a Non-Class F channel, an Undefined Instruction Trap will occur.

3. If the WCWCS instruction is not preceded by an ECWCS instruction, a System Check Trap will occur.

IOCD FORMAT FOR
CLASS F I/O WCS



Real Data Address: Bits 8-31 (MSW) will contain the address of the memory location for the first word to be loaded.

Start WCS Address: Bits 0-15 (LSW) will contain the address of WCS where the first word is to be loaded.

Byte Count: Bits 16-31 (LSW) will contain the number of bytes to be loaded.

SECTION VII

CONTROL PANEL

INTRODUCTION

This section describes the function and operation of the Serial System Control Panel of the 32/70 Series Computer. Figure 7-1 shows the controls, keyboards, and displays of the Serial System Control Panel.

PANEL LOCK

The PANEL LOCK switch is a two-position rotary key switch having an unlocked and locked position. The turnkey can be removed in either position. When the switch is in the unlocked position, all panel keys on the Serial System Control Panel are operational. In the locked position, all panel keys are disabled except for the ATTENTION key and those panel keys for write/read of control switches on the Hexadecimal Keyboard and the Function Keyboard which remain operational at all times.

POWER

The POWER switch is a two-position latching pushbutton which provides the capability to power the system on or off. The state of the power is determined by the RUN and HALT indicators. When the power is on, either the RUN or HALT indicator is on. When the power is off, all indicators on the panel will be off.

RUN/HALT

Depressing the RUN/HALT key while the CPU is in the Halt mode causes the CPU to enter the Run mode and begin executing instructions from the location specified in the Program Status Word.

Depressing the RUN/HALT key while the CPU is in the Run mode causes the CPU to enter the Halt mode. In the Halt mode, the CPU no longer executes instructions from memory; instead, it is placed in a micro-routine which monitors selected panel support functions.

SYSTEM RESET

Depressing the SYSTEM RESET key when the system is in the Halt mode initializes all appropriate logic in all SelBUS devices.

ATTENTION

Depressing the ATTENTION key causes an interrupt to occur at the Attention Interrupt level, priority level 13_{16}.

INITIAL PROGRAM LOAD

Depressing the INITIAL PROGRAM LOAD key when the CPU is in the Halt mode puts the CPU in the Initial Program Load mode. This initiates the microprogram loading sequence which consists of reading a dedicated device address and then reading from the specified device. The device number is entered through the Serial System Control Panel.

CLOCK OVERRIDE

Depressing the CLK OVRD key activates the override condition; no further interrupts from the Real-Time Clock or the Interval Timer will be permitted. A second depression of this key deactivates the clock override condition.

OPERATION/MODE INDICATORS

The Operation/Mode indicators consist of single-bit, light-emitting diodes. These indicators display either the operational mode of the CPU or a conditioned interruption in computer operation.

PARITY ERROR

The PARITY ERROR display, when lit, indicates that a memory parity error has occurred during a CPU memory access.

INTERRUPT ACTIVE

The INTERRUPT ACTIVE display is on if any interrupt (I/O or external) is in the active state.

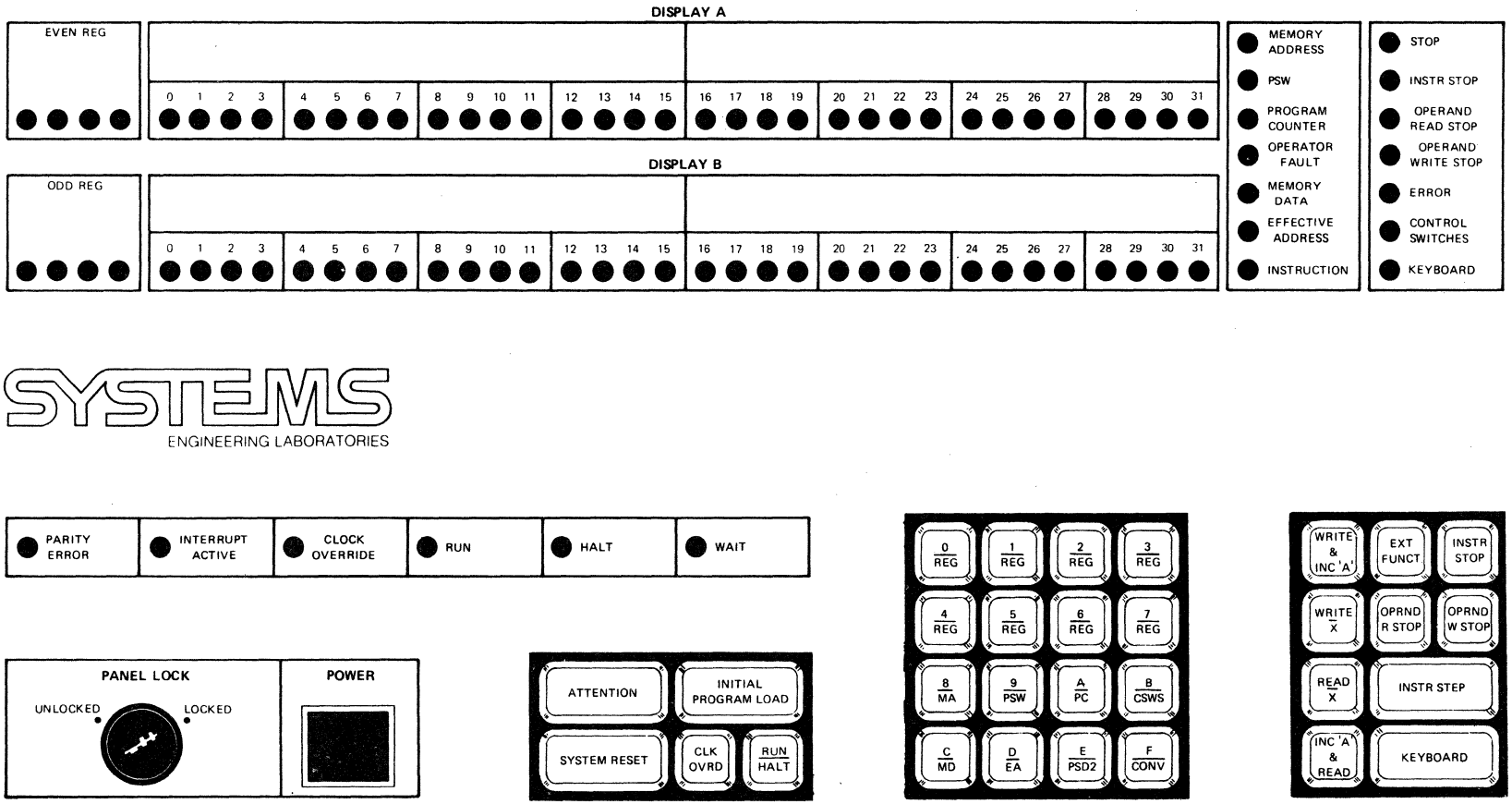


Figure 7-1. The 32/70 Series Control Panel

CLOCK OVERRIDE — The CLOCK OVERRIDE display is on when the clock override condition is active (The CLK OVRD key is depressed.)

RUN — The RUN display is on when the CPU is in the Run mode. While in the Run mode, the CPU is executing instructions.

HALT — The HALT display is on when the CPU is in the Halt mode. In this mode, no instructions are executed.

WAIT — The WAIT display is on when the CPU is in the Wait state: that is, no instructions are being executed. However, I/O operations continue to completion.

KEYBOARDS

The Hexadecimal keyboard and the Function keyboard operate in conjunction with the panel displays as a unified Input/Output device to the CPU. Operation of the keyboards provides the capability to selectively store and/or read data in memory or in registers.

HEXADECIMAL KEYBOARD

The Hexadecimal keyboard, referred to as the "Hex keyboard," is used to either enter data into the B-Display or to enter the source/destination of the panel function to be performed. The dual function of each Hex keyboard key is indicated by the upper and lower case values printed on each key.

The upper case values are used when data is entered into the B-Display. The upper case values are enabled by first depressing the Function keyboard KEYBOARD key. The Function keyboard KEYBOARD key causes the B-Display to be cleared and the KEYBOARD indicator to illuminate. When the KEYBOARD indicator is illuminated, all entries from the Hex keyboard are interpreted as data and are entered into the B-Display by a 4-bit left shift of the contents of the B-Display and insertion of the hex value of the depressed key into the four least significant bit positions (hex digit) of the B-Display. If the 32-bit capacity of the B-Display is exceeded, the most significant four bits of the B-Display are shifted out of the display and lost, and the new digit is loaded into the least significant bit positions.

The lower case values of the Hex keyboard are used to specify the source/destination of a function to be performed by the Serial System Control Panel. The lower case values are enabled by first depressing the Function keyboard $\frac{\text{WRITE}}{X}$ key or the $\frac{\text{READ}}{X}$ keys, causing the subsequent entry from the Hex keyboard to be interpreted as the source/destination of the Write or Read function. When a source/destination is entered in the Hex keyboard, it causes a corresponding indicator to illuminate on the Serial System Control Panel. The Hex keyboard keys that cause an indicator to illuminate are listed as follows:

1. The $\frac{0}{\text{REG}}$, $\frac{2}{\text{REG}}$, $\frac{4}{\text{REG}}$, and $\frac{6}{\text{REG}}$ keys cause the EVEN register Hex indicator to indicate the hexadecimal value of the even register addressed.
2. The $\frac{1}{\text{REG}}$, $\frac{3}{\text{REG}}$, $\frac{5}{\text{REG}}$, and $\frac{7}{\text{REG}}$ keys cause the ODD REGISTER Hex indicator to indicate the hexadecimal value of the odd register addressed.
3. The $\frac{8}{\text{MA}}$ key causes the MEMORY ADDRESS indicator to illuminate.

4. The 9/PSW key causes the PSW (Program Status Word) indicator to illuminate.

5. The A/PC key causes the PROGRAM COUNTER indicator to illuminate.

6. The B/CSWS key causes the CONTROL SWITCHES indicator to illuminate.

7. The C/MD key causes the MEMORY DATA indicator to illuminate.

8. The D/EA key causes the EFFECTIVE ADDRESS indicator to illuminate.

9. The E/PSD2 key causes the second word of the PSD to be displayed in the B-Display.

10. The F/CONV key causes a logical address in the A-Display to be converted to a 24-bit physical address and be displayed in the B-Display.

FUNCTION KEYBOARD

The Function keyboard sets the function to be performed by the Control Panel according to the key that is depressed. The functions that can be selected by the Function keyboard keys are as follows:

WRITE X KEY

Depressing the WRITE/X key causes the operand in the B-Display to be stored in the destination specified by a subsequent depression of a Hex keyboard key. The lower case value of the Hex keyboard key describes the destination of the operand and the function indicator that will illuminate. The use of the Hex keyboard D/EA key is prohibited for the destination of a Write function. If the Hex keyboard C/MD is depressed, the contents of the A-Display (which must contain a valid memory address, PSW, or Program Counter Value) are used to address memory. The operand in the B-Display is stored at that memory address.

READ X KEY

Depressing the READ/X key causes the operand specified by a subsequent depression of a Hex keyboard key to be loaded into either the A- or B-Display. The lower case value of the Hex keyboard key describes the source of the operand and the function indicator that will illuminate. The use of the Hex keyboard 8/MA key is prohibited as a source of a Read function.

If the Hex keyboard C/MD key is depressed, the contents of the A-Display (which must contain a valid memory address, PSW, or Program Counter Value) are used to address memory. The contents of the addressed memory location are loaded into the B-Display.

WRITE & INC 'A' KEY

Depressing the WRITE & INC 'A' key causes the operand in the B-Display to be stored in the memory location addressed by the A-Display. The A-Display is then incremented by four (one memory word). The A-Display must contain a valid memory address, and the B-Display must contain the operand to be stored in memory. The WRITE & INC 'A' key is used for Write functions to sequential memory locations.

INC 'A' & READ KEY

The INC 'A' & READ key causes the address in the A-Display to be incremented by four (one memory word), and the updated address is used to address memory. The contents of the addressed memory location are then loaded into the B-Display. The A-Display must contain a valid memory address. The INC 'A' & READ Key is used for Read functions of sequential memory locations.

EXT FUNCT KEY

The EXT FUNCT key is used for extended functions, such as a lamp test routine.

INSTR STOP KEY

Depressing the INSTR STOP key causes the Instruction Stop function to become active or inactive. If the Instruction Stop function was active, and the INSTR STOP indicator was illuminated, depressing the Function keyboard INSTR STOP key would deactivate the Instruction Stop function and turn off the indicator. If the Instruction Stop function was inactive, and the INSTR STOP indicator was off, depressing the Function keyboard INSTR STOP key would activate the Instruction Stop function, illuminate the INSTR Stop indicator and load the memory address from the B-Display into the Address Compare register. When the CPU fetches an instruction from the memory location specified by the Address Compare register, the STOP indicator illuminates, and the CPU halts. The B-Display must be loaded with the instruction address by way of the Hex keyboard before depressing the Function keyboard INSTR STOP key.

OPRND R STOP KEY

Depressing the OPRND R STOP key causes the Operand Read Stop function to become active or inactive. If the Operand Read Stop function was active, and the OPERAND READ STOP indicator was illuminated, depressing the Function keyboard OPRND R STOP key would deactivate the Operand Read Stop function and turn off the indicator.If the Operand Read Stop was inactive, depressing the Function keyboard OPRND R STOP key would activate the Operand Read Stop function and load the memory address from the B-Display into the Address Compare register. When the CPU reads an operand from the specified memory location, the STOP indicator illuminates, and the CPU halts. The B-Display must be loaded with the operand memory address by way of the Hex keyboard before depressing the OPRND R STOP key. The address in the B-Display for Compare Halt must be entered in a 24-bit physical address format.

OPRND W STOP KEY

Depressing the OPRND W STOP key causes the Operand Write Stop function to become active or inactive. If the Operand Write Stop function was active, and the OPERAND WRITE STOP indicator was illuminated, depressing the function keyboard OPRND W STOP key would deactivate the Operand Write Stop function and turn off the indicator. If the Operand Write Stop was inactive, depressing the Function keyboard OPRND W STOP key would activate the Operand Write Stop function, illuminate the OPERAND WRITE STOP indicator, and load the memory address from the B-Display into the Address Compare register. When the CPU stores an operand in the specified memory location, the STOP indicator illuminates, and the CPU halts. The B-Display must be loaded with the operand memory address by way of Hex keyboard before depressing the OPRND W STOP key. The address in the B-Display for Compare Halt must be entered in a 24-bit physical address format.

INSTR STEP KEY

Depressing the INSTR STEP key causes both the A- and B-Displays and all function indicators, except the Instruction and Operand STOP indicators, to be cleared. It then causes the CPU to execute one software instruction that is addressed by the CPU Program Status Word Register. After one instruction has been executed, the CPU halts, the A-Display will indicate the next Program Status Word, and the B-Display will indicate the new Instruction word.

KEYBOARD KEY

Depressing the KEYBOARD key causes the B-Display to be cleared, the KEYBOARD indicator to illuminate, and any subsequent Hex keyboard entries to be interpreted at their upper case values and inserted into the four rightmost bit positions of the B-Display. The KEYBOARD key is normally used to clear the B-Display before entering an operand into the B-Display from the Hex keyboard.

PANEL DISPLAYS

A-DISPLAY

The A-Display consists of 32 binary indicators that are divided into eight 4-bit fields for easy hexadecimal read-out. When the Hex Display option is included in the Serial Control Panel, a hex display indicator above each 4-bit field provides a direct hexadecimal read-out of the contents of the field.

The contents of the A-Display are described by the function indicators directly to the right of the A-Display or by the EVEN REGISTER hex display indicator to the left of the A-Display. The contents of the A-Display can be any of the following:

1. A memory address in bit positions 8-31.
2. The contents of the CPU Program Status Word Register.
3. The Program Counter bits from the CPU Program Status Word Register in bit positions 8-31.
4. The most significant word of the Program Status Doubleword.
5. The contents of any of four even-numbered CPU general purpose registers.

The A-Display can be loaded in either a Write or a Read function, as specified by the corresponding keys of the Function keyboard. In a Write function, the A-Display is loaded as follows:

1. The B-Display is loaded with an operand or address by way of the Hex keyboard.
2. The Function keyboard $\frac{\text{WRITE}}{\text{X}}$ key is depressed to specify the Write function.
3. The Hex keyboard lower case value (operand destination) is specified by depressing one of the even-numbered register keys on the MA, PSW, or PC keys.

In a Read function, the A-Display is loaded as follows:

1. The Function keyboard $\frac{\text{READ}}{\text{X}}$ key is depressed to specify the Read function.
2. The Hex keyboard lower case value (operand source) is specified by depressing one of the even-numbered register keys, the PSW or the PC key.

When the Read function is complete, the operand specified by the Hex keyboard will be loaded into the A-Display, and the corresponding function indicator will illuminate to define the contents of the A-Display. The exception being the E key which will load PSD word 2 into the B-Display.

When the A-Display contains a memory address, Program Status Word, or Program Counter, the contents of the A-Display can be used to address memory during memory Read or Write functions. In these types of functions, the WRITE & INC 'A' and the INC 'A' & READ keys of the Function keyboard can be used to access memory and increment the contents of the A-Display to the next sequential memory word address.

B-DISPLAY

The B-Display consists of 32 binary indicators that are divided into eight 4-bit fields for easy hexadecimal read-out. When the Hex Display option is included in the Serial System Control Panel, a hex display indicator above each 4-bit field provides a direct hexadecimal read-out of the contents of the field.

The contents of the B-Display are described by the function indicators to the right of the B-Display or by the ODD REGISTER hex display indicator to the left of the B-Display. The contents of the B-Display can be any of the following:

1. Keyboard data being entered from the Hex keyboard.
2. A memory data word.
3. An Effective Address of the instruction addressed by the PSW or PC in the A-Display.
4. An instruction addressed by the PSW or PC in the A-Display.
5. The contents of the CPU Control Switches in bit positions 0-11.
6. The contents of any of four odd-numbered CPU General Purpose Registers.
7. The least significant word of the Program Status Doubleword (PSD).
8. The physical address in an address conversion operation.

The B-Display can be loaded in either a Write or Read function, as specified by the corresponding keys of the Function keyboard. In a Write function, the B-Display is loaded as follows:

1. An operand is loaded from the Hex keyboard.
2. The Function keyboard $\frac{\text{WRITE}}{\text{X}}$ key is depressed to specify the Write function.
3. The contents of the B-Display can be transferred to the A-Display by depressing any even-numbered register key, the MA key, the PSW key, or the PC key to specify the operand destination.
4. The contents of the B-Display can be transferred directly to an odd-numbered register, the CPU Control Switch register, or to the memory location addressed by the A-Display by depressing one of the odd-numbered register keys, the CSWS key, or the MD key, respectively, to specify the operand destination.

In a Read function, the B-Display is loaded as follows:

1. The Function keyboard $\frac{\text{READ}}{\text{X}}$ key is depressed to specify a Read function.
2. The Hex keyboard lower case value (operand source) is specified by depressing an odd-numbered register key, the CSWS key, the MD key, the EA key, or the PSD2 key.

When the Read function is complete, the corresponding indicator will illuminate to define the contents of the B-Display.

ODD/EVEN INDICATORS

EVEN REGISTER INDICATOR

The EVEN REGISTER indicator consists of a hexadecimal display (optional) indicator that provides a direct read-out of the even-numbered register being addressed by the Serial System Control Panel. The contents of this register are displayed to the left of the A-Display. The EVEN REGISTER indicator will be illuminated only when the A-Display contains the contents of an even-numbered register.

The four binary indicators directly below the EVEN REGISTER indicator correspond to the even register address.

ODD REGISTER INDICATOR

The ODD REGISTER indicator consists of a hexadecimal display (optional) indicator that provides a direct read-out of the odd-numbered register being addressed by the Serial System Control Panel. The contents of this register are displayed in the B-Display. The ODD REGISTER indicator will be illuminated only when the B-Display contains the contents of an odd-numbered register.

The four binary displays directly below the ODD REGISTER indicator correspond to the odd register address.

MISCELLANEOUS INDICATORS

MEMORY ADDRESS INDICATOR

The MEMORY ADDRESS indicator is a 1-bit display that defines the contents of the A-Display as a memory address. The memory address can only be loaded into the into the A-Display with a Write function. The memory address is primarily used for memory addressing in subsequent memory read or write operations.

PSW INDICATOR

The PSW indicator is a 1-bit display that defines the contents of the A-Display as the CPU Program Status Word Register. The PSW can be used for changing the contents of the CPU PSW and for memory addressing in subsequent memory read or write operations. In PSD mode, the A-Display represents the most significant word of the PSD.

PROGRAM COUNTER INDICATOR

The PROGRAM COUNTER indicator is a 1-bit display that defines the contents of the A-Display as the current value of the CPU Program Counter portion of the Program Status Word Register. The Program Counter can be loaded into the A-Display with either a Write or a Read function. The Program Counter can be used for changing the Program Counter portion of the Program Status Word Register and for memory addressing in subsequent memory read or write operations.

OPERATOR FAULT INDICATOR

The OPERATOR FAULT indicator is a 1-bit display that indicates that an operator fault has occurred on the Serial System Control Panel. Two types of Operator Faults can normally occur:

1. The function selected by the Function keyboard was illogical with respect to the operand source/destination selected by the Hex keyboard.

2. The function selected by the Function keyboard combined with the operation and source/destination specified by the Hex keyboard cannot be performed because the CPU is in a Run mode and the specified function is not is not allowed.

The specific type of Operator Fault that has occurred must be determined by the Serial System Control Panel operator.

MEMORY DATA INDICATOR

The MEMORY DATA indicator is a 1-bit display that defines the contents of the B-Display as memory data from the memory location addressed by the A-Display. For the MEMORY DATA indicator to be illuminated, the A-Display must contain a memory address and the MEMORY ADDRESS indicator must be illuminated. Memory data can be manually loaded into the B-Display and the addressed memory location in a Write function or read into the B-Display from the addressed memory location in Read function.

EFFECTIVE ADDRESS INDICATOR

The EFFECTIVE ADDRESS indicator is a 1-bit display that defines the contents of the B-Display as an effective address of a software memory reference instruction that is addressed by the contents of the A-Display. The A-Display must contain either a PSW or Program Counter Value, which is used by the CPU to access the software memory reference instruction. The CPU then computes the instruction's effective address based on any indexed or indirect addressing specified by the instruction. When the addressing is complete, the effective address can only be loaded into the B-Display by a Read function.

ERROR INDICATOR

The ERROR indicator is a 1-bit display that defines the contents of the B-Display as an internal error code. The internal errors exclude operator errors and include Serial System Control Panel errors, CPU acknowledge errors, SelBUS transmission errors, and memory errors.

CONTROL SWITCHES INDICATOR

The CONTROL SWITCHES indicator is a 1-bit display that defines the contents of the B-Display as the CPU Control Switches. The Control Switches can be loaded into the B-Display in either a Write or a Read function. In a Write function, the B-Display is loaded from the Hex keyboard. The contents of the B-Display (Control Switches) are then loaded into a dedicated memory location. In a Read function, the Serial System Control Panel reads the dedicated memory location and transfers its contents (Control Switches) to the B-Display.

The specific dedicated memory address used for storage of the Control Switches is a function of the computer system configuration and CPU firmware.

KEYBOARD INDICATOR

The KEYBOARD indicator is a 1-bit display that indicates when the upper case values (hex digits 0 through F) can be loaded into the B-Display from the Hex keyboard. The KEYBOARD indicator illuminates in response to the KEYBOARD switch on the Function keyboard.

INSTRUCTION INDICATOR

The INSTRUCTION indicator is a 1-bit display that defines the contents of the B-Display as an instruction addressed by a PSW or Program Counter Value in the A-Display. An instruction can be manually loaded into the B-Display and addressed memory location in a Write function or read into the B-Display from the addressed memory location in a Read function. The Serial System Control Panel defines the contents of any memory location as an instruction if the A-Display contains a PSW or Program Counter Value. If the A-Display contains a memory address (the MEMORY ADDRESS indicator is illuminated), the contents of the addressed memory location is defined as memory data, which illuminates the MEMORY DATA indicator.

STOP INDICATOR

The STOP indicator is a 1-bit display that indicates when the CPU has been halted by the Instruction Stop, Operand Read Stop, or Operand Write Stop logic. In addition to the STOP indicator, one or more of the INSTR STOP, OPERAND READ STOP, or OPERAND WRITE STOP indicators should also be illuminated indicating the type of stop logic that is active. When the STOP indicator illuminates and CPU halts, the A-Display will contain the current contents of the CPU PSW, and the B-Display will contain the instruction addressed by the Program Counter portion of the PSW (A-Display).

INSTR STOP INDICATOR

The INSTR STOP indicator is a 1-bit display that defines the active condition of the Instruction Stop logic. When the Instruction Stop is active, a memory address is in the Address Compare register. When the CPU fetches an instruction from that memory location, the CPU will halt and the STOP indicator will illuminate.

OPERAND READ STOP INDICATOR

The OPERAND READ STOP indicator is a 1-bit display that defines the active condition of the Operand Read Stop logic. When Operand Read Stop is active, a memory address is in the Address Compare register. When the CPU performs a memory read from that memory location, the CPU will halt and the STOP indicator will illuminate.

OPERAND WRITE STOP INDICATOR

The OPERAND WRITE STOP indicator is a 1-bit display that defines the active condition of the Operand Write Stop logic. When the Operand Write Stop is active, a memory address is in the Address Compare register. When the CPU performs a memory write to that location, the CPU will halt and the STOP indicator will illuminate.

OPERATOR FAULT INDICATOR

The Serial System Control Panel is equipped with an OPERATOR FAULT indicator that illuminates when the panel detects an operator fault condition. When the OPERATOR FAULT indicator lights, the rightmost digit of the B-Display will indicate the source of the fault as follows:

Fault Number	Description
1.	Does not Apply to the Serial Panel
2.	Operation Not Allowed - Run on Lock Restrictions
3.	Invalid Operand Source or Destination
4.	A-Display Not Valid for Operation to be Performed
5.	Invalid Extended Function
6.	Special Extended Function Not Enabled
7.	Does not Apply to the Serial Panel

ERROR INDICATOR

The Serial System Control Panel is equipped with an ERROR indicator that illuminates when a panel error is detected. When the ERROR indicator lights, the rightmost digit of the B-Display will indicate the source of the fault as follows:

Fault Number	Description
1.	CPU Uart Error
2.	Transmission Error other than CPU Uart
3.	No Response from Memory
4.	Nonpresent Memory
5.	Parity Error in Memory
6.	Write/Read Compare Error in Memory
7.	Bus Interchange or Memory is Broken

MISCELLANEOUS INDICATIONS

Several indicators are available to the operator when the computer, while in the PSD mode, enters the Halt mode or when the PSW is read by the panel switches. They are as follows:

1. Bit 6 indicates last instruction executed was a right halfword.
2. Bit 7 indicates Arithmetic Exception.
3. Bit 8 indicates PSD mode if set or PSW mode if zero.
4. Bit 9 indicates Mapped if set or Unmapped if zero.
5. Bit 32 indicates Interrupts Blocked if set.

OPERATING INSTRUCTIONS

The following discussions provide step-by-step instructions for using the controls and indicators of the Serial System Control Panel. Each heading designates a specific function to be performed and the sequential steps necessary to complete the function. Each discussion includes two significant conditions necessary for each function: Panel Lock position and CPU mode.

Description of the Load B-Display from Hex keyboard and description of the Load A-Display provide the primary functions of the Serial System Control Panel that are necessary for all other functions. After these descriptions are initially presented, they are referred to by title only in subsequent descriptions.

LOAD B-DISPLAY FROM HEX KEYBOARD

1. The Panel Lock must be in the Unlocked mode.
2. The CPU can be in the Run or Halt mode.
3. Depress the KEYBOARD key on the Function keyboard.
4. Observe that the B-Display clears and the KEYBOARD indicator illuminates.
5. Enter the operand into the B-Display by depressing the correct hex digit key on the Hex keyboard, one digit at a time.
6. Observe that the last digit entered from the Hex keyboard is loaded into the four least significant bit positions of the B-Display and that any previous contents of the B-Display is left-shifted by four bit positions.
7. When the B-Display is full, or the complete operand has been entered into the B-Display, the operation is complete.
8. If the 32-bit capacity of the B-Display is exceeded, the four most significant bit positions of the B-Display will be lost as each new digit is entered into the B-Display.
9. If a mistake is made while entering the operand, depress the KEYBOARD key on the Function keyboard and return to step 4.

LOAD A-DISPLAY

The Load A-Display function can be divided into seven subfunctions that described separately in the following descriptions. The seven subfunctions are:

1. Write Memory Address
2. Write PSW (Program Status Word)
3. Read PSW (Program Status Word)

4. Write PSD2

5. Read PSD2

6. Write Program Counter

7. Read Program Counter

WRITE MEMORY ADDRESS

1. The Panel Lock must be in the Unlocked mode.

2. The CPU can be in the Run or Halt mode.

3. Enter the memory address into the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard.)

4. Depress the $\frac{\text{WRITE}}{\text{X}}$ key on the Function keyboard.

5. Depress the $\frac{8}{\text{MA}}$ key on the Hex keyboard.

6. Observe that the memory address is transferred from the B-Display to the A-Display and that the MEMORY ADDRESS indicator illuminates.

7. The operation is complete. If a mistake was made during the sequence, return to Step 3.

WRITE PSW

1. The Panel Lock must be in the Unlocked mode

2. The CPU must be in the Halt mode.

3. Enter the PSW operand into the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard.)

4. Depress the $\frac{\text{WRITE}}{\text{X}}$ key on the Function keyboard.

5. Depress the $\frac{9}{\text{PSW}}$ key on the Hex keyboard.

6. Observe that the PSW operand is transferred from the B-Display to the A-Display and that PSW indicator illuminates. At this time, the PSW operand has also been loaded into the CPU Program Status Word Register.

7. The operation is complete. If a mistake was made during the sequence, return to Step 3.

READ PSW

1. The Panel Lock must be in the Unlocked mode.

2. The CPU must be in the Halt mode.

3. Depress the $\frac{\text{READ}}{\text{X}}$ key on the Function keyboard.

4. Depress the $\frac{9}{\text{PSW}}$ key on the Hex keyboard.

5. Observe that the Program Status Word is transferred from the CPU Program Status Word Register to the A-Display and that the PSW indicator illuminates.

6. The operation is complete. If a mistake was made during the sequence, return to Step 3.

WRITE PSD2

1. The Panel Lock must be in the Unlocked mode.
2. The CPU must be in the Halt mode.
3. Enter the PSD2 (least significant word of the PSD) operand into the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard).
4. Depress the $\frac{\text{WRITE}}{\text{X}}$ key on the Function keyboard.
5. Depress the $\frac{\text{E}}{\text{PSD2}}$ key on the Hex keyboard.
6. The operation is complete. If a mistake was made during the sequence, return to Step 3.

READ PSD2

1. The Panel Lock must be in the Unlocked mode.
2. The CPU must be in the Halt mode.
3. Depress the $\frac{\text{READ}}{\text{X}}$ key on the Function keyboard.
4. Depress the $\frac{\text{E}}{\text{PSD2}}$ key on the Hex keyboard.
5. The operation is complete. If a mistake was made during the sequence, return to Step 3.

WRITE PROGRAM COUNTER

1. The Panel Lock must be in the Unlocked mode.
2. The CPU must be in the Halt mode.
3. Enter the Program Counter Value into bits 8-31 of the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard.)
4. Depress the $\frac{\text{WRITE}}{\text{X}}$ key on the Function keyboard.
5. Depress the $\frac{\text{A}}{\text{PC}}$ key on the Hex keyboard.
6. Observe that bits 13-31 of the B-Display are transferred to the A-Display and that the PROGRAM COUNTER indicator illuminates. At this time, the Program Counter Value has been loaded into the Program Counter portion of the CPU Program Status Word Register.
7. The operation is complete. If a mistake was made during the sequence, return to Step 3.

READ PROGRAM COUNTER

1. The Panel Lock must be in the Unlocked mode.
2. The CPU must be in the Halt mode.
3. Depress the $\frac{\text{READ}}{\text{X}}$ key on the Function keyboard.

4. Depress the $\frac{A}{PC}$ key on the Hex keyboard.

5. Observe that the Program Counter Value is transferred from the CPU Program Status Word Register and transferred to bits 13-31 of the A-Display and that the PROGRAM COUNTER indicator illuminates.

6. The operation is complete. If a mistake was made during the sequence, return to Step 3.

WRITE MEMORY (SINGLE ADDRESS)

The Write Memory sequence is dependent on a valid address (Memory Address, PSW, or Program Counter Value) in the A-Display. This value can be set in the A-Display by using any of the subfunctions described in the Load A-Display discussion.

1. The Panel Lock must be in the Unlocked mode.

2. Enter a Memory Address, PSW, or Program Counter Value into the A-Display as described in the Load A-Display discussion.

3. Enter the operand to be stored in memory into the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard.)

4. Depress the $\frac{\text{WRITE}}{X}$ key on the Function keyboard.

5. Depress the $\frac{C}{MD}$ key on the Hex keyboard.

6. Observe that the operand in the B-Display remains unchanged and that either the MEMORY DATA or INSTRUCTION indicator illuminates as follows:

   a. If the A-Display contains a memory address, the MEMORY DATA indicator should illuminate.

   b. If the A-Display contains either a PSW or Program Counter Value, the INSTRUCTION indicator should illuminate.

7. The operation is complete. If a mistake was made during the sequence, return to Step 3.

READ MEMORY (SINGLE ADDRESS)

The Read Memory sequence is dependent on a valid address (Memory Address, PSW, or Program Counter Value) in the A-Display. This value can can be set in the A-Display by using any of the subfunctions described in the Load A-Display discussion.

1. The Panel Lock must be in the Unlocked mode.

2. Enter a Memory Address, PSW, or Program Counter Value into the A-Display as described in the Load A-Display discussion.

3. Depress the INC 'A' & READ key on the Function keyboard.

4. Observe that the A-Display is incremented by four to the next sequential memory address.

5. Observe that the MEMORY DATA or INSTRUCTION indicator illuminates as follows:

   a. If the A-Display contains a memory address, the MEMORY DATA indicator should illuminate.

b. If the A-Display contains a PSW or Program Counter Value, the INSTRUCTION indicator should illuminate.

6. The operand in the B-Display should be the contents of the memory location addressed by the A-Display.

7. If no mistakes occurred in the above sequence, return to Step 4 to read the next memory location.

8. If a mistake was made, the same memory address can be reread by performing the Read Memory (Single Address) sequence beginning with Step 4.

When using the Read Memory (Sequential Addresses) sequence, the first address entered into the A-Display will not be read. To read the first address, perform the Read Memory (Single Address) sequence, then enter the Read Memory (Sequential Addresses) sequence beginning with Step 4.

INSTRUCTION STEP

The Instruction Step function causes the CPU to enter the Run mode and execute one software instruction. After the instruction has been executed, the CPU returns to the Halt mode.

The sequence for the Instruction Step function is as follows:

1. The Panel Lock must be in the Unlocked mode.

2. The CPU must be in the Halt mode.

3. If the CPU Program Status Word Register does not point to the instruction to be executed, load a Program Counter or PSW Value into the A-Display and CPU register as described in the Load A-Display description.

4. Depress the INSTR STEP key on the Function keyboard.

5. Observe that the PANEL HALT indicator is illuminated.

6. The system halts with the updated PSW Value in the A-Display and instruction addressed by the A-Display (PSW) in the B-Display.

7. To execute the next instruction, return to Step 4.

READ EFFECTIVE ADDRESS

The Read Effective Address sequence causes the CPU to fetch the instruction addressed by the Program Counter of PSW Value in the A-Display. The instruction fetched should be a memory reference instruction to generate a valid effective address. After the instruction has been fetched, the CPU calculates the instruction's effective memory address by performing the indexing and indirect addressing specified by the instruction. When the address computations are complete, the CPU transfers the effective address to the Serial System Control Panel's B-Display.

The Read Effective Address sequence is as follows:

1. The Panel Lock must be in the Unlocked mode.

2. The CPU must be in the Halt mode.

3. Enter a PSW or Program Counter Value into the A-Display as described in the Load A-Display discussion.

4. Depress the $\frac{\text{READ}}{\text{X}}$ key on the Function keyboard.

5. Depress the $\frac{D}{EA}$ key on the Hex keyboard.

6. Observe that the EFFECTIVE ADDRESS indicator illuminates and the effective address is loaded into the B-Display.

7. The operation is complete. If a mistake occurred, return to Step 3.

CONVERT ADDRESS

The Convert Address sequence causes conversion of a logical address in the A-Display to a 24-bit physical address in the B-Display.

The Convert Address sequence is as follows:

1. The Panel Lock must be in the Unlocked mode.

2. The CPU must be in the Halt mode.

3. The CPU must be in the PSD mode.

4. Enter a PSW, Program Counter Value, or memory address in the A-Display as described in the Load A-Display discussion.

5. Depress the $\frac{READ}{X}$ key on the Function keyboard.

6. Depress the $\frac{F}{CONV}$ key on the Hex keyboard.

7. The operation is complete. If a mistake occurred, return to Step 4.

STOP SEQUENCE

The Stop sequence includes the Instruction Stop, Operand Read Stop, and Operand Write Stop functions. Each function has its own key on the Function Keyboard and its own indicator to indicate when that function is active.

The sequence for the Stop functions is as follows:

1. The Panel Lock must be in the Unlocked mode.

2. The CPU must be in the Halt mode.

3. Enter the memory stop address into the B-Display from the Hex keyboard.

4. Depress the INSTR STOP, OPRND R STOP, or OPRND W STOP key on the Function keyboard.

5. Observe that the indicator for the Stop function selected by the Function keyboard illuminates.

6. If the CPU is in the Run mode and the specified memory location is accessed in the correct operating mode (Instruction Fetch, Operand Read, or Operand Write), the following events should occur.

   a. The PANEL HALT indicator should illuminate.

   b. The STOP indicator should illuminate.

   c. The current contents of the CPU PSWR should appear in the A-Display, and the PSW indicator should illuminate.

d. The instruction addressed by the Program Counter portion of the PSW should appear in the B-Display, and the INSTRUCTION indicator should illuminate.

7. To clear any active Stop function, perform the following steps:

   a. Depress the Function keyboard key that corresponds to the function to be cleared.

   b. Observe that the corresponding Stop function indicator turns.

When using the Stop function, multiple Stop functions can be set by entering the Stop functions sequentially; however, if a different Stop address is entered with each Stop function, the most recently entered Stop address will be used for all active Stop functions.

CONTROL SWITCHES SEQUENCE

The Control Switches sequence is used to set or monitor the CPU Control Switches that are stored in a dedicated memory location. The Control Switches sequence is divided into the Write Control Switches function that sets the Control Switches in the dedicated memory location and the Read Control Switches function that reads the contents of the dedicated memory location.

WRITE CONTROL SWITCHES

1. The Panel Lock must be in the Unlocked mode.
2. Enter the Control Switch configuration into bit positions 0-12 of the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard).
3. Depress the $\frac{\text{WRITE}}{\text{X}}$ key on the Function keyboard.
4. Depress the $\frac{\text{B}}{\text{CSWS}}$ key on the Hex keyboard.
5. Observe that the CONTROL SWITCHES indicator illuminates. At this time, the contents of the B-Display have been transferred to the control switches dedicated memory location.
6. The operation is complete. If a mistake was made, return to Step 3.

READ CONTROL SWITCHES

1. The Panel Lock must be in the Unlocked mode.
2. The CPU can be in the Run or Halt mode.
3. Depress the $\frac{\text{READ}}{\text{X}}$ key on the Function keyboard.
4. Depress the $\frac{\text{B}}{\text{CSWS}}$ key on the Hex keyboard.
5. Observe that the CONTROL SWITCHES indicator illuminates, and the contents of the control switches dedicated memory location are transferred to the B-Display.
6. The operation is complete. If a mistake was made, return to Step 3.

INITIAL PROGRAM LOAD SEQUENCE

The Initial Program Load (IPL) sequence is a function of the Serial System Control Panel and CPU firmware. The IPL sequence is as follows:

1. The Panel Lock must be in the Unlocked mode.
2. The CPU must be in the Halt mode.
3. Depress the SYSTEM RESET key.
4. Enter the peripheral device address of the IPL device into the B-Display from the Hex keyboard. (See Load B-Display from Hex keyboard.) Note: If an all-zeros device address is entered into the B-Display, the CPU firmware will default to a firmware-specified IPL device address.
5. Depress the INITIAL PROGRAM LOAD key.
6. When the IPL sequence is complete, the CPU will be in the Halt mode. Any changes in the software program can be made at this time.
7. The operation is complete. Refer to the software description of the Bootstrap program for operating instructions of the Bootstrap program.

SECTION VIII

SYSTEM INITIALIZATION

INITIAL PROGRAM LOAD (IPL)

Initialization and configuration of a 32/70 Series System is accomplished through the use of the Initial Program Load (IPL) sequence. This sequence initializes the system, sets up the I/O configuration, and boots in the operating system. The usual method of initializing the system is through the use of the card reader to read in a deck of cards containing the I/O device configuration and assigned interrupt organization. The IPL sequence is initiated by placing the Initial Configuration Load (ICL) deck of cards in the card reader, setting up of the address of the card reader on the system front panel, and depressing the IPL button on the system front panel.

It should be noted that if the mode jumper on the CPU is set up for the PSD mode, the CPU will come up in the PSD mode. If, when placing the address of the IPL device in the B-Display of the front panel, additional information is added, then the CPU can be made to come up in the PSW mode of operation. The procedure for establishing the PSW mode of operation is as follows:

1. If using either the parallel or serial front panel for data entry, add 8000 to the device address (sets bit 16 to One). For example, if the address of the card reader is 7800, then by the setting of bit 16 to One (or adding 8000), the resultant address becomes F800.

2. If using the serial front panel, entering a 55 plus the card reader address results in the CPU coming up in the PSW mode. The resultant address in the B-Display is then 00557800.

After the cards are read into the system, the SYSTEM RESET button is depressed, the address of the device (disc) containing the operating system is entered on the front panel, and the IPL button is again depressed, thereby booting in the operating system.

The Initial Configuration Load (ICL) deck of cards contains three basic record formats. The following sections provide descriptions for each format.

FORMATS OF THE INITIAL CONFIGURATION LOAD (ICL)

Initial Configuration Load (ICL) records are read from a default or selected peripheral device. The ICL records are converted into information that is used to initialize the 256- x 32-bit Configuration RAM (CR) contained in the 32/70 Series Central Processor Unit (CPU). Information contained in the CR is used by the CPU to address and maintain the status of the 128 possible devices and the 112 possible interrupts.

Initial Configuration Load records must be in the following ASCII or Hollerith formats:

FORMAT #1 *DEVXX=FCILCASA (,NN)

where:

*DEV defines that the record contains a controller definition entry.

XX is the hexadecimal address that will be used by macro level input/output instructions to address the controller.

= is a necessary delimiter. Each letter to the right of this delimiter represents one hexadecimal digit (four binary bits).

F flags used by the CPU for input/output emulation. Presently, this field is always zero.

C defines the class of controller being emulated. Presently, this field can contain one of the following values:

0	=	LINE PRINTER
1	=	CARD READER
2	=	TELETYPE
3	=	INTERVAL TIMER
4	=	PANEL
5 to D	=	Unassigned
E	=	ALL OTHERS
F	=	EXTENDED I/O

IL is the hexadecimal interrupt priority level of the Service Interrupt (i.e., priority levels 14_{16} through 23_{16}) for the defined controller.

CA is the hexadecimal controller address as defined by the hardware switches on the IOM.

SA is the lowest hexadecimal device subaddress used by the controller. This field is normally zero when more than one device is configured.

( ) denotes optional parameter.

' is a delimiter that must be used when more than one device is configured.

NN is a 2-digit hexadecimal number that specifies the number of devices configured on the controller.

NOTE 1: The subaddress (SA) field must reflect the following for the Teletype, Line Printer, Card Reader (TLC) controller:

1. Card Reader is subaddress 0_{16}.
2. Teletype is subaddress 1_{16}.
3. Line Printer is subaddress 2_{16}.

FORMAT #2 *INTXX RS

where:

*INT defines that the record contains an interrupt definition entry.

XX is the hexadecimal interrupt priority level that is to be emulated.

= is a necessary delimiter. Each letter to the right of this delimiter represents one hexadecimal digit (four binary bits).

R is the hexadecimal RTOM board number to which the interrupt XX is assigned.

S is the hexadecimal subaddress on the RTOM board to which the interrupt XX is assigned.

NOTE 1: RTOM physical controller address 79_{16} is RTOM board number 1, address $7A_{16}$ is RTOM board number 2, etc.

NOTE 2: Real-Time Clock hardware is connected to subaddress 6_{16} on the RTOM board.

NOTE 3: Interval Timer hardware is connected to subaddress 4_{16} on the RTOM board.

NOTE 4: RTOM physical controller addresses must be 79_{16} or above. This convention allows a maximum of seven RTOM boards to be defined on a single 32/70 Series system. Seven RTOM boards will support 112_{10} interrupt levels.

FORMAT #3 *END

where:

*END is the last record of an Initial Configuration Load (ICL) deck. This record signifies the end of the load process.

EXAMPLES OF INITIAL CONFIGURATION LOAD (ICL) RECORDS

A device entry:

*DEV04=0E140100,04

The device entry above specifies the following information:

1. The 32/70 series input/output commands will address the controller as 04_{16}.

2. The ",04" is an optional parameter that specifies that there are 4_{16} devices on the controller. There will be four entries defined in the Configuration RAM (CR). The input/output commands (i.e., CD and TD) will address the devices as 4_{16}, 5_{16}, 6_{16}, and 7_{16}.

3. The controller is an "E" class controller.

4. The priority of the Service Interrupt (SI) is 14_{16}.

Assigning a priority to a controller has the following implications:

a. The Transfer Interrupt location for priority 14_{16} is 100_{16}.

b. The Service Interrupt vector location for priority 14_{16} is 140_{16}.

c. The emulation IOCD will be stored at location 700_{16}.

d. The interrupt control instructions (i.e., DI, EI, RI, AI, DAI) will control the interrupt on the controller by addressing priority 14_{16}.

5. The physical address of the controller is 01_{16}.

An interrupt entry (RTOM):

```
*INT28=16
```

The interrupt entry above specifies the following information:

1. The 32/70 Series interrupt control instructions (i.e., DI, EI, RI, AI, DAI) will control the interrupt on the RTOM by addressing priority 28_{16}.

2. The number of the RTOM board is 1.

3. The subaddress on the RTOM board is 6_{16} (jumpered logic subaddress is 9).

A sample Initial Configuration Load (ICL) Deck is given in Figure 8-1.

EXAMPLE	COMMENTS
(SEE NOTE)	READ ASCII CARD READER IOCD
*DEV04=0E150400,02	CARTRIDGE DISC WITH TWO PLATTERS
*DEV08=0E160800,04	MOVING-HEAD DISC
*DEV10=0E181000,04	9-TRACK MAG TAPE
*DEV20=0E1A2000,10	GPMC
*DEV60=0E1E6000,08	ADS
*DEV78=01207800	PRIMARY CARD READER
*DEV7A=00217802	PRIMARY LINE PRINTER
*DEV7E=02237801	PRIMARY TELETYPE
*INT00=1F	POWER FAIL/AUTO RESTART
*INT01=1E	SYSTEM OVERRIDE
*INT12=1D	MEMORY PARITY TRAP
*INT13=1C	CONSOLE INTERRUPT
*INT24=1B	NONPRESENT MEMORY
*INT25=1A	UNDEFINED INSTRUCTION TRAP
*INT26=19	PRIVILEGE VIOLATION
*INT27=18	CALL MONITOR
*INT28=16	REAL-TIME CLOCK
*INT29=17	ARITHMETIC EXCEPTION
*INT2A=15	EXTERNAL INTERRUPT
*INT28=14	EXTERNAL INTERRUPT
*INT2C=13	EXTERNAL INTERRUPT
*INT2D=12	EXTERNAL INTERRUPT
*END	LAST CARD

NOTE: THE FIRST RECORD IS DEVICE DEPENDENT AND REPRESENTS TWO 32-BIT WORDS, THE FIRST BEING ALL ZEROS AND THE SECOND A VALID IOCD TO READ THE FOLLOWING RECORDS.

Figure 8-1. System Initial Configuration Load (ICL) Deck

APPENDIX A

INSTRUCTION SET

(FUNCTIONALLY GROUPED)

The 32/70 Series instructions are listed alphabetically by mnemonic code within one of the following functional groupings:

- Load/Store Instructions
- Branch Instructions
- Compare Instructions
- Logical Instructions
- Register Transfer Instructions
- Shift Operation Instructions
- Bit Manipulation Instructions
- Fixed-Point Arithmetic Instructions
- Floating-Point Arithmetic Instructions
- Control Instructions
- Interrupt Instructions
- Input/Output Instructions
- Memory Management
- Writable Control Storage

Each entry includes the following information:

- Instruction Mnemonic
- Operand Format
- Operation Code
- Instruction Function

The following symbols are used to denote required entries for operand formats:

b	-	Bit Number In General Register (0-31)
c	-	Bit Number In Memory Byte
d	-	Destination General Register (0-7)
f	-	Function
m	-	Memory Address
n	-	Channel Or Device Number
p	-	Protect Register Number
s	-	Source General Register (0-7)
v	-	Value of Operand For Immediate, Shift, and Condition Code Instructions
x	-	Index Register (1-3)
*	-	Indirect Addressing
z	-	Register Address Field for Special Instructions

Halfword instructions are denoted by # preceding the instruction mnemonic. The halfword instructions are all interregister (except TRP and TPR) instructions: CALM, WAIT, HALT, and NOP.

LOAD/STORE INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
LB	d,*m,x	AC08	6-10	Load Byte
LD	d,*m,x	AC00	6-13	Load Doubleword
LH	d,*m,x	AC00	6-11	Load Halfword
LW	d,*m,x	AC00	6-12	Load Word
LF	d,*m,x	CC00	6-28	Load File
LEA	d,*m,x	D000	6-23	Load Effective Address
LEAR	d,*m,x	8000	6-24	Load Effective Address Real
LA	d,*m,x	3400	6-25	Load Address
LI	d,v	C800	6-22	Load Immediate
LMB	d,*m,x	B008	6-14	Load Masked Byte
LMD	d,*m,x	B000	6-17	Load Masked Doubleword
LMH	d,*m,x	B000	6-15	Load Masked Halfword
LMW	d,*m,x	B000	6-16	Load Masked Word
LNB	d,*m,x	B408	6-18	Load Negative Byte
LND	d,*m,x	B400	6-21	Load Negative Doubleword
LNH	d,*m,x	B400	6-19	Load Negative Halfword
LNW	d,*m,x	B400	6-20	Load Negative Word
STB	s,*m,x	D408	6-29	Store Byte
STD	s,*m,x	D400	6-32	Store Doubleword
STH	s,*m,x	D400	6-30	Store Halfword
STW	s,*m,x	D400	6-31	Store Word
STF	s,*m,x	DC00	6-37	Store File
STMB	s,*m,x	D808	6-33	Store Masked Byte
STMD	s,*m,x	D800	6-36	Store Masked Doubleword
STMH	s,*m,x	D800	6-34	Store Masked Halfword
STMW	s,*m,x	D800	6-35	Store Masked Word
ZMB	*m,x	F808	6-39	Zero Memory Byte
ZMD	*m,x	F800	6-42	Zero Memory Doubleword
ZMH	*m,x	F800	6-40	Zero Memory Halfword
ZMW	*m,x	F800	6-41	Zero Memory Word
#ZR	d	0C00	6-43	Zero Register

MEMORY MANAGEMENT INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
#SEA		000D	6-59	Set Extended Addressing
#CEA		000F	6-60	Clear Extended Addressing
LMAP	d	2C07	6-61	Load MAP
#TMAPR	s,d	2C0A	6-62	Transfer MAP to Register

# Indicates Halfword Instruction
* Indicates Indirect Addressing

BRANCH INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
BCF	v,*m,x	F000	6-73	Branch Condition False
BCT	v,*m,x	EC00	6-74	Branch Condition True
BFT	*m,x	F000	6-75	Branch Function True
BIB	d,m	F400	6-77	Branch After Incrementing Byte
BID	d,m	F460	6-80	Branch After Incrementing Doubleword
BIH	d,m	F420	6-78	Branch After Incrementing Halfword
BIW	d,m	F440	6-79	Branch After Incrementing Word
BL	*m,x	F880	6-76	Branch and Link
BU	*m,x	EC00	6-72	Branch Unconditionally

COMPARE INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
CAMB	d,*m,x	9008	6-83	Compare Arithmetic with Memory Byte
CAMD	d,*m,x	9000	6-86	Compare Arithmetic with Memory Doubleword
CAMH	d,*m,x	9000	6-84	Compare Arithmetic with Memory Halfword
CAMW	d,*m,x	9000	6-85	Compare Arithmetic with Memory Word
#CAR	s,d	1000	6-87	Compare Arithmetic with Register
CI	d,v	C805	6-88	Compare Immediate
CMMB	d,*m,x	9408	6-89	Compare Masked with Memory Byte
CMMD	d,*m,x	9400	6-92	Compare Masked with Memory Doubleword
CMMH	d,*m,x	9400	6-90	Compare Masked with Memory Halfword
CMMW	d,*m,x	9400	6-91	Compare Masked with Memory Word
#CMR	s,d	1400	6-93	Compare Masked with Register

LOGICAL INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ANMB	d,*m,x	8408	6-95	AND Memory Byte
ANMD	d,*m,x	8400	6-98	AND Memory Doublword
ANMH	d,*m,x	8400	6-96	AND Memory Halfword
ANMW	d,*m,x	8400	6-97	AND Memory Word
#ANR	s,d	0400	6-99	AND Register and Register
EOMB	d,*m,x	8C08	6-106	Exclusive OR Memory Byte
EOMD	d,*m,x	8C00	6-109	Exclusive OR Memory Doubleword
EOMH	d,*m,x	8C00	6-107	Exclusive OR Memory Halfword
EOMW	d,*m,x	8C00	6-108	Exclusive OR Memory Word
#EOR	s,d	0C00	6-110	Exclusive OR Register and Register
#EORM	s,d	0C08	6-111	Exclusive OR Register and Register Masked
ORMB	d,*m,x	8808	6-100	OR Memory Byte
ORMD	d,*m,x	8800	6-103	OR Memory Doubleword
ORMH	d,*m,x	8800	6-101	OR Memory Halfword
ORMW	d,*m,x	8800	6-102	OR Memory Word
#ORR	s,d	0800	6-104	OR Register and Register
#ORRM	s,d	0808	6-105	OR Register and Register Masked

# Indicates Halfword Instruction
* Indicates Indirect Addressing

REGISTER TRANSFER INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
#XCR	s,d	2C05	6-55	Exchange Registers
#XCRM	s,d	2C0D	6-56	Exchange Registers Masked
TPR	r,p	FB80	6-50	Transfer Protect Register to Register
#TRC	s,d	2C03	6-53	Transfer Register Complement
#TRCM	s,d	2C0B	6-54	Transfer Register Complement Masked
#TRN	s,d	2C04	6-51	Transfer Register Negative
#TRNM	s,d	2C0C	6-52	Transfer Register Negative Masked
TRP	s,p	FB00	6-49	Transfer Register to Protect Register
#TRR	s,d	2C00	6-47	Transfer Register to Register
#TRRM	s,d	2C08	6-48	Transfer Register to Register Masked
#TRSW	s	2800	6-57	Transfer Register to PSWR
#TRSC	s,d	2C0E	6-46	Transfer Register to Scratchpad
#TSCR	s,d	2C0F	6-45	Transfer Scratchpad to Register

SHIFT OPERATION INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
#NOR	d,s	6000	6-113	Normalize
#NORD	d,s	6400	6-114	Normalize Double
#SCZ	d,s	6800	6-115	Shift and Count Zeros
#SLA	d,v	6C40	6-116	Shift Left Arithmetic
#SLAD	d,v	7840	6-119	Shift Left Arithmetic Double
#SLC	d,v	7440	6-118	Shift Left Circular
#SLL	d,v	7040	6-117	Shift Left Logical
#SLLD	d,v	7C40	6-120	Shift Left Logical Double
#SRA	d,v	6C00	6-121	Shift Right Arithmetic
#SRAD	d,v	7800	6-124	Shift Right Arithmetic Double
#SRC	d,v	7400	6-123	Shift Right Circular
#SRL	d,v	7000	6-122	Shift Right Logical
#SRLD	d,v	7C00	6-125	Shift Right Logical Double

BIT MANIPULATION INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ABM	c,*m,x	A008	6-132	Add Bit in Memory
#ABR	d,b	2000	6-133	Add Bit in Register
SBM	c,*m,x	9808	6-128	Set Bit in Memory
#SBR	d,b	1800	6-129	Set Bit in Register
TBM	c,*m,x	A408	6-134	Test Bit in Memory
#TBR	d,b	2400	6-135	Test Bit in Register
ZBM	c,*m,x	9C08	6-130	Zero Bit in Memory
#ZBR	d,b	1C00	6-131	Zero Bit in Register

# Indicates Halfword Instruction
* Indicates Indirect Addressing

FIXED-POINT ARITHMETIC INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ADI	d,v	C801	6-150	Add Immediate
ADMB	d,*m,x	B808	6-140	Add Memory Byte
ADMD	d,*m,x	B800	6-143	Add Memory Doubleword
ADMH	d,*m,x	B800	6-141	Add Memory Halfword
ADMW	d,*m,x	B800	6-142	Add Memory Word
#ADR	s,d	3800	6-144	Add Register to Register
#ADRM	s,d	3808	6-145	Add Register to Register Masked
ARMB	s,*m,x	E808	6-146	Add Register to Memory Byte
ARMD	s,*m,x	E800	6-149	Add Register to Memory Doubleword
ARMH	s,*m,x	E800	6-147	Add Register to Memory Halfword
ARMW	s,*m,x	E800	6-148	Add Register to Memory Word
SUI	s,v	C802	6-157	Subtract Immediate
SUMB	d,*m,x	BC08	6-151	Subtract Memory Byte
SUMD	d,*m,x	BC00	6-154	Subtract Memory Doubleword
SUMH	d,*m,x	BC00	6-152	Subtract Memory Halfword
SUMW	d,*m,x	BC00	6-153	Subtract Memory Word
#SUR	s,d	3C00	6-155	Subtract Register from Register
#SURM	s,d	3C08	6-156	Subtract Register from Register Masked
MPMH	d,*m,x	C000	6-159	Multiply by Memory Halfword
MPMW	d,*m,x	C000	6-160	Multiply by Memory Word
#MPR	s,d	4000	6-161	Multiply Register by Register
MPI	d,v	C803	6-162	Multiply Immediate
MPMB	d,*m,x	C008	6-158	Multiply by Memory Byte
DVI	d,v	C804	6-167	Divide Immediate
DVMB	d,*m,x	C408	6-163	Divide by Memory Byte
DVMH	d,*m,x	C400	6-164	Divide by Memory Halfword
DVMW	d,*m,x	C400	6-165	Divide by Memory Word
#DVR	s,d	4400	6-166	Divide Register by Register
#ES	d	0004	6-168	Extend Sign
#RND	d	0005	6-169	Round Register

FLOATING-POINT ARITHMETIC INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Format
ADFD	d,*m,x	E008	6-173	Add Floating-Point Doubleword
ADFW	d,*m,x	E008	6-172	Add Floating-Point Word
SUFD	d,*m,x	E000	6-175	Subtract Floating-Point Doubleword
SUFW	d,*m,x	E000	6-174	Subtract Floating-Point Word
MPFD	d,*m,x	E408	6-177	Multiply Floating-Point Doubleword
MPFW	d,*m,x	E408	6-176	Multiply Floating-Point Word
DVFD	d,*m,x	E400	6-179	Divide Floating-Point Doubleword
DVFW	d,*m,x	E400	6-178	Divide Floating-Point Word

# Indicates Halfword Instruction
* Indicates Indirect Addressing

CONTROL INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
BRI	*m,x	F900	6-181	Branch and Reset Interrupt
LPSD	d,*m,x	F980	6-182	Load Program Status Doubleword
LPSDCM	d,*m,x	FA80	6-183	Load Program Status Doubleword and Change Map
#CALM	v	3000	6-192	Call Monitor
DAE		000E	6-198	Disable Arithmetic Exception Trap
EAE		0008	6-197	Enable Arithmetic Exception Trap
EXM	*m,x	A800	6-187	Execute Memory
EXR	s	C807	6-185	Execute Register
EXRR	s	C807	6-186	Execute Register Right
#HALT		0000	6-188	Halt
#LCS		0003	6-184	Load Control Switches
#NOP		0002	6-190	No Operation
RDSTS	d	0009	6-195	Read CPU Status Word
SVC	IND,CALL#	C806	6-193	Supervisor Call
#SIPU		000A	6-191	Signal IPU
#SETCPU	s	2C09	6-194	Set CPU Mode
#WAIT		0001	6-189	Wait

INTERRUPT INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ACI	s,v	FC77	6-207	Activate Channel Interrupt
AI	v	FC03	6-204	Activate Interrupt
#BEI		0006	6-211	Block External Interrupts
DACI	s,v	FC7F	6-210	Deactivate Channel Interrupt
DAI	v	FC04	6-206	Deactivate Interrupt
DCI	s,v	FC6F	6-209	Disable Channel Interrupt
DI	v	FC01	6-205	Disable Interrupt
ECI	s,v	FC67	6-208	Enable Channel Interrupt
EI	v	FC00	6-202	Enable Interrupt
RI	v	FC02	6-203	Request Interrupt
#UEI		0007	6-212	Unblock External Interrupts

INPUT/OUTPUT INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
CD	n,f	FC06	6-216	Command Device
TD	n,f	FC05	6-217	Test Device
SIO	s,v	FC17	6-218	Start I/O
TIO	s,v	FC1F	6-219	Test I/O
STPIO	s,v	FC27	6-220	Stop I/O
RSCHNL	s,v	FC2F	6-221	Reset Channel
HIO	s,v	FC37	6-222	Halt I/O
GRIO	s,v	FC3F	6-223	Grab Controller
RSCTL	s,v	FC47	6-224	Reset Controller
ECWCS	s,v	FC4F	6-225	Enable Channel WCS Load
WCWCS	s,v	FC5F	6-226	Write Channel WCS

WRITABLE CONTROL STORAGE INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
#WWCS	s,d	000C	6-65	Write WCS
#RWCS	s,d	000B	6-66	Read WCS
#JWCS	*m,x	FA00	6-67	Jump WCS

# Indicates Halfword Instruction
* Indicates Indirect Addressing

APPENDIX B

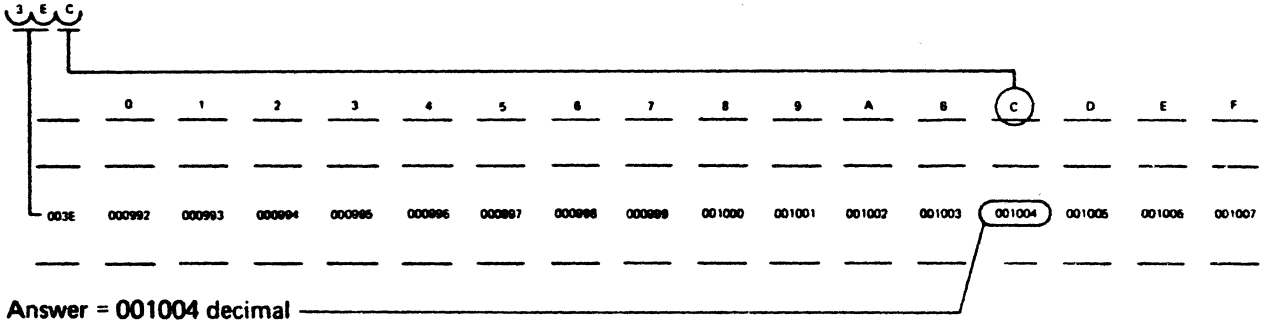
HEXADECIMAL-DECIMAL CONVERSION TABLE

The following table contains the necessary information for direct conversion of decimal and hexadecimal numbers in these ranges:

Hexadecimal	Decimal
00000 to 01FFF	000000 to 008191

To convert a hexadecimal number to a decimal value, locate all but the last digit of the hexadecimal value in the left-most column of the table, then follow that line of figures to the right to the column under the last digit of the hexadecimal value. At this intersection is the decimal value of the hexadecimal number.

Example: Convert hexadecimal 3EC to decimal.



Answer = 001004 decimal

For decimal to hexadecimal conversion as in the example, first find the decimal value (1004) in the table, then construct the hexadecimal value from the hexadecimal characters above the column and in the left-most column.

For numbers outside the range of the table, add the following values to the table figures:

Hexadecimal	Decimal
3000	12288
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	52248
E000	57344
F000	61440

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	000000	000001	000002	000003	000004	000005	000006	000007	000008	000009	000010	000011	000012	000013	000014	000015
0001	000016	000017	000018	000019	000020	000021	000022	000023	000024	000025	000026	000027	000028	000029	000030	000031
0002	000032	000033	000034	000035	000036	000037	000038	000039	000040	000041	000042	000043	000044	000045	000046	000047
0003	000048	000049	000050	000051	000052	000053	000054	000055	000056	000057	000058	000059	000060	000061	000062	000063
0004	000064	000065	000066	000067	000068	000069	000070	000071	000072	000073	000074	000075	000076	000077	000078	000079
0005	000080	000081	000082	000083	000084	000085	000086	000087	000088	000089	000090	000091	000092	000093	000094	000095
0006	000096	000097	000098	000099	000100	000101	000102	000103	000104	000105	000106	000107	000108	000109	000110	000111
0007	000112	000113	000114	000115	000116	000117	000118	000119	000120	000121	000122	000123	000124	000125	000126	000127
0008	000128	000129	000130	000131	000132	000133	000134	000135	000136	000137	000138	000139	000140	000141	000142	000143
0009	000144	000145	000146	000147	000148	000149	000150	000151	000152	000153	000154	000155	000156	000157	000158	000159
000A	000160	000161	000162	000163	000164	000265	000166	000167	000168	000169	000170	000171	000172	000173	000174	000175
000B	000176	000177	000178	000179	000180	000181	000182	000183	000184	000185	000186	000187	000188	000189	000190	000191
000C	000192	000193	000194	000195	000196	000197	000198	000199	000200	000201	000202	000203	000204	000205	000206	000207
000D	000208	000209	000210	000211	000212	000213	000214	000215	000216	000217	000218	000219	000220	000221	000222	000223
000E	000224	000225	000226	000227	000228	000229	000230	000231	000232	000233	000234	000235	000236	000237	000238	000239
000F	000240	000241	000242	000243	000244	000245	000246	000247	000248	000249	000250	000251	000252	000253	000254	000255

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0010	000256	000257	000258	000259	000260	000261	000262	000263	000264	000265	000266	000267	000268	000269	000270	000271
0011	000272	000273	000274	000275	000276	000277	000278	000279	000280	000281	000282	000283	000284	000285	000286	000287
0012	000288	000289	000290	000291	000292	000293	000294	000295	000296	000297	000298	000299	000300	000301	000302	000303
0013	000304	000305	000306	000307	000308	000309	000310	000311	000312	000313	000314	000315	000316	000317	000318	000319
0014	000320	000321	000322	000323	000324	000325	000326	000327	000328	000329	000330	000331	000332	000333	000334	000335
0015	000336	000337	000338	000339	000340	000341	000342	000343	000344	000345	000346	000347	000348	000349	000350	000351
0016	000352	000353	000354	000355	000356	000357	000358	000359	000360	000361	000382	000363	000364	000365	000366	000367
0017	000368	000369	000370	000371	000372	000373	000374	000375	000376	000377	000378	000379	000380	000381	000382	000383
0018	000384	000385	000386	000387	000388	000389	000390	000391	000392	000393	000394	000395	000396	000397	000398	000399
0019	000400	000401	000402	000403	000404	000405	000406	000407	000408	000409	000410	000411	000412	000413	000414	000415
001A	000416	000417	000418	000419	000420	000421	000422	000423	000424	000425	000426	000427	000428	000429	000430	000431
001B	000432	000433	000434	000435	000436	000437	000438	000439	000440	000441	000442	000443	000444	000445	000446	000447
001C	000448	000449	000450	000451	000452	000453	000454	000455	000456	000457	000458	000459	000460	000461	000462	000463
001D	000464	000465	000466	000467	000468	000469	000470	000471	000472	000473	000474	000475	000476	000477	000478	000479
001E	000480	000481	000482	000483	000484	000485	000486	000487	000488	000489	000490	000491	000492	000493	000494	000495
001F	000496	000497	000498	000499	000500	000501	000502	000503	000504	000505	000506	000507	000508	000509	000510	000511

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0020	000512	000513	000514	000515	000516	000517	000518	000519	000520	000521	000522	000523	000524	000525	000526	000527
0021	000528	000529	000530	000531	000532	000533	000534	000535	000536	000537	000538	000539	000540	000541	000542	000543
0022	000544	000545	000546	000547	000548	000549	000550	000551	000552	000553	000554	000555	000556	000557	000558	000559
0023	000560	000561	000562	000563	000564	000565	000566	000567	000568	000569	000570	000571	000572	000573	000574	000575
0024	000576	000577	000578	000579	000580	000581	000582	000583	000584	000585	000586	000587	000588	000589	000590	000591
0025	000592	000593	000594	000595	000596	000597	000598	000599	000600	000601	000602	000603	000604	000605	000606	000607
0026	000608	000609	000610	000611	000612	000613	000614	000615	000616	000617	000618	000619	000620	000621	000622	000623
0027	000624	000625	000626	000627	000628	000629	000630	000631	000632	000633	000634	000635	000636	000637	000638	000639
0028	000640	000641	000642	000643	000644	000645	000646	000647	000648	000649	000650	000651	000652	000653	000654	000655
0029	000656	000657	000658	000659	000660	000661	000662	000663	000664	000665	000666	000667	000668	000669	000670	000671
002A	000672	000673	000674	000675	000676	000677	000678	000679	000680	000581	000682	000683	000684	000685	000686	000687
002B	000688	000689	000690	000691	000692	000693	000694	000695	000696	000697	000698	000699	000700	000701	000702	000703
002C	000704	000705	000706	000707	000708	000709	000710	000711	000712	000713	000714	000715	000716	000717	000718	000719
002D	000720	000721	000722	000723	000724	000725	000726	000727	000728	000729	000730	000731	000732	000733	000734	000735
002E	000736	000737	000738	000739	000740	000741	000742	000743	000744	000745	000746	000747	000748	000749	000750	000751
002F	000752	000753	000754	000755	000756	000757	000758	000759	000760	000761	000762	000763	000764	000765	000766	000767

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0030	000768	000769	000770	000771	000772	000773	000774	000775	000776	000777	000778	000779	000780	000781	000782	000783
0031	000784	000785	000786	000787	000788	000789	000790	000791	000792	000793	000794	000795	000796	000797	000798	000799
0032	000800	000801	000802	000803	000804	000805	000806	000807	000808	000809	000810	000811	000812	000813	000814	000815
0033	000816	000817	000818	000819	000820	000821	000822	000823	000824	000825	000826	000827	000828	000829	000830	000831
0034	000832	000833	000834	000835	000836	000837	000830	000839	000840	000841	000842	000843	000844	000845	000846	000847
0035	000848	000849	000850	000851	000852	000853	000854	000855	000856	000857	000858	000859	000860	000861	000862	000863
0036	000864	000865	000866	000867	000868	000869	000870	000871	000872	000873	000874	000875	000876	000877	000878	000879
0037	000880	000881	000882	000883	000884	000885	000886	000887	000888	000889	000890	000891	000892	000893	000894	000895
0038	000896	000897	000898	000899	000900	000901	000902	000903	000904	000905	000906	000907	000908	000909	000910	000911
0039	000912	000913	000914	000915	000916	000917	000918	000919	000920	000921	000922	000923	000924	000925	000926	000927
003A	000928	000929	000930	000931	000932	000933	000934	000935	000936	000937	000938	000939	000940	000941	000942	000943
003B	000944	000945	000946	000947	000948	000949	000950	000951	000952	000953	000954	000955	000956	000957	000958	000959
003C	000960	000961	000962	000963	000964	000965	000966	000967	000968	000969	000970	000971	000972	000973	000974	000975
003D	000976	000977	000978	000979	000980	000981	000982	000983	000984	000985	000986	000987	000988	000989	000990	000991
003E	000992	000993	000994	000995	000996	000997	000998	000999	0010000	001001	001002	001003	001004	001005	001006	001007
003F	001008	001009	001010	001011	001012	001013	001014	001015	001016	001017	001018	001019	001020	001021	001022	001023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0040	001024	001025	001026	001027	001028	001029	001030	001031	001032	001033	001034	001035	001036	001037	001038	001039
0041	001040	001041	001042	001043	001044	001045	001046	001047	001048	001049	001050	001051	001052	001063	001064	001055
0042	001056	001057	001058	001059	001060	001061	001062	001063	001064	001055	001066	001067	001068	001069	001070	001071
0043	001072	001073	001074	001075	001076	001077	001078	001079	001080	001081	001082	001083	001084	001085	001086	001087
0044	001088	001089	001090	001091	001092	001093	001094	001095	001096	001097	001098	001099	001100	001101	001102	001103
0045	001104	001105	001106	001107	001108	001109	001110	001111	001112	001113	001114	001115	001116	001117	001118	001119
0046	001120	001121	001122	001123	001124	001125	001126	001127	001128	001129	001130	001131	001132	001133	001134	001135
0047	001136	001137	001138	001139	001140	001141	001142	001143	001144	001145	001146	001147	001148	001149	001150	001151
0048	001152	001153	001154	001155	001156	001157	001158	001159	001160	001161	001162	001163	001164	001165	001166	001167
0049	001168	001169	001170	001171	001172	001173	001174	001175	001176	001177	001178	001179	001180	001181	001182	001183
004A	001184	001185	001186	001187	001188	001189	001190	001191	001192	001193	001194	001195	001196	001197	001198	001199
004B	001200	001201	001202	001203	001204	001205	001206	001207	001208	001209	001210	001211	001212	001213	001214	001215
004C	001216	001217	001218	001219	001220	001221	001222	001223	001224	001225	001226	001227	001228	001229	001230	001231
004D	001232	001233	001234	001235	001236	001237	001238	001239	001240	001241	001242	001243	001244	001245	001246	001247
004E	001248	001249	001250	001251	001252	001253	001254	001255	001256	001257	001258	001259	001260	001261	001262	001263
004F	001264	001265	001266	001267	001268	001269	001270	001271	001272	001273	001274	001275	001276	001277	001278	001279

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0050	001280	001281	001282	001283	001284	001285	001286	001287	001288	001289	001290	001291	001292	001293	001294	001295
0051	001296	001297	001298	001299	001300	001301	001302	001303	001304	001305	001306	001307	001308	001309	001310	001311
0052	001312	001313	001314	001315	001316	001317	001318	001319	001320	001321	001322	001323	001324	001325	001326	001327
0053	001328	001329	001330	001331	001332	001333	001334	001335	001336	001337	001338	001339	001340	001341	001342	001343
0054	001344	001345	001346	001347	001348	001349	001350	001351	001352	001353	001354	001355	001356	001357	001358	001359
0055	001360	001361	001362	001363	001364	001365	001366	001367	001368	001369	001370	001371	001372	001373	001374	001378
0056	001376	001377	001378	001379	001380	001381	001382	001383	001384	001385	001386	001387	001388	001389	001390	001391
0057	001392	001393	001394	001395	001396	001397	001398	001399	001400	001401	001402	001403	001404	001405	001406	001407
0058	001408	001409	001410	001411	001412	001413	001414	001415	001416	001417	001418	001419	001420	001421	001422	001423
0059	001424	001425	001426	001427	001428	001429	001430	001431	001432	001433	001434	001435	001436	001437	001438	001439
005A	001440	001441	001442	001443	001444	001445	001446	001447	001448	001449	001450	001451	001452	001453	001454	001455
005B	001456	001457	001458	001459	001460	001461	001462	001463	001464	001465	001466	001467	001468	001469	001470	001471
005C	001472	001473	001474	001475	001476	001477	001478	001479	001480	001481	001482	001483	001484	001485	001486	001487
005D	001488	001489	001490	001491	001492	001493	001494	001495	001496	001497	001498	001499	001500	001501	001502	001503
005E	001504	001505	001506	001507	001508	001509	001510	001511	001512	001513	001514	001515	001516	001517	001518	001519
005F	001520	001521	001522	001523	001524	001525	001526	001527	001528	001529	001530	001531	001532	001533	001534	001535

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0060	001536	001537	001538	001539	001540	001541	001542	001543	001544	001545	001546	001547	001548	001549	001550	001551
0061	001552	001553	001554	001555	001556	001557	001558	001559	001560	001561	001562	001563	001564	001565	001566	001567
0062	001568	001569	001570	001571	001572	001573	001574	001575	001576	001577	001578	001579	001580	001581	001587	001583
0063	001584	001585	001586	001587	001588	001589	001590	001591	001592	001593	001594	001595	001596	001597	001598	001599
0064	001600	001601	001602	001603	001604	001605	001606	001607	001608	001609	001610	001611	001612	001613	001614	001615
0065	001616	001617	001618	001619	001620	001621	001622	001623	001624	001625	001626	001627	001628	001629	001630	001631
0066	001632	001633	001634	001635	001636	001637	001638	001639	001640	001641	001642	001643	001644	001645	001646	001647
0067	001648	001649	001650	001651	001652	001653	001654	001655	001656	001657	001658	001659	001660	001661	001662	001663
0068	001664	001665	001656	001667	001668	001669	001670	001671	001672	001673	001674	001675	001676	001677	001678	001679
0069	001680	001681	001682	001683	001684	001685	001686	001687	001688	001689	001690	001691	001692	001693	001694	001695
006A	001696	001697	001698	001699	001700	001701	001702	001703	001704	001705	001706	001707	001708	001709	001710	001711
006B	001712	001713	001714	001715	001716	001717	001718	001719	001720	001721	001722	001723	001724	001725	001726	001727
006C	001728	001729	001730	001731	001732	001733	001734	001735	001736	001737	001738	001739	001740	001741	001742	001743
006D	001744	001745	001746	001747	001748	001749	001750	001751	001752	001753	001754	001755	001756	001757	001758	001759
006E	001760	001761	001762	001763	001764	001765	001766	001767	001768	001769	001770	001771	001772	001773	001774	001775
006F	001776	001777	001778	001779	001780	001781	001782	001783	001784	001785	001786	001787	001788	001789	001790	001791

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0070	001792	001793	001794	001795	001796	001797	001798	001799	001800	001801	001802	001803	001804	001805	001806	001807
0071	001808	001809	001810	001811	001812	001813	001814	001815	001816	001817	001818	001819	001820	001821	001822	001823
0072	001824	001825	001826	001827	001828	001829	001830	001831	001832	001833	001834	001835	001836	001837	001838	001839
0073	001840	001841	001842	001843	001844	001845	001846	001847	001848	001849	001850	001851	001852	001853	001854	001855
0074	001856	001857	001858	001859	001860	001861	001862	001863	001864	001865	001866	001867	001868	001869	001870	001871
0075	001872	001873	001874	001875	001876	001877	001878	001879	001880	001881	001882	001883	001884	001885	001886	001887
0076	001888	001889	001890	001891	001892	001893	001894	001895	001896	001897	001898	001899	001900	001901	001902	001903
0077	001904	001905	001906	001907	001908	001909	001910	001911	001912	001913	001914	001915	001916	001917	001918	001919
0078	001920	001921	001922	001923	001924	001925	001926	001927	001928	001929	001930	001931	001932	001933	001934	001935
0079	001936	001937	001938	001939	001940	001941	001942	001943	001944	001945	001946	001947	001948	001949	001950	001951
007A	001952	001953	001954	001955	001956	001957	001958	001959	001960	001961	001962	001963	001964	001965	001966	001967
007B	001968	001969	001970	001971	001972	001973	001974	001975	001976	001977	001978	001979	001980	001981	001982	001983
007C	001984	001985	001986	001987	001988	001989	001990	001991	001992	001993	001994	001995	001996	001997	001998	001999
007D	002000	002001	002002	002003	002004	002005	002006	002007	002008	002009	002010	002011	002012	002013	002014	002015
007E	002016	002017	002018	002019	002020	002021	002022	002023	002024	002025	002026	002027	002028	002029	002030	002031
007F	002032	002033	002034	002035	002036	002037	002038	002039	002040	002041	002042	002043	002044	002045	002046	002047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0080	002048	002049	002050	002051	002052	002053	002054	002055	002056	002057	002058	002059	002060	002061	002062	002063
0081	002064	002065	002066	002067	002068	002069	002070	002071	002072	002073	002074	002075	002076	002077	002078	002079
0082	002080	002081	002082	002083	002084	002085	002086	002087	002088	002089	002090	002091	002092	002093	002094	002095
0083	002096	002097	002098	002099	002100	002101	002102	002103	002104	002105	002106	002107	002108	002109	002110	002111
0084	002112	002113	002114	002115	002116	002117	002118	002119	002120	002121	002122	002123	002124	002125	002126	002127
0085	002128	002129	002130	002131	002132	002133	002134	002135	002136	002137	002138	002139	002140	002141	002142	002143
0086	002144	002145	002146	002147	002148	002149	002150	002151	002152	002153	002154	002155	002156	002157	002158	002159
0087	002160	002161	002162	002163	002164	002165	002166	002167	002168	002169	002170	002171	002172	002173	002174	002175
0088	002176	002177	002178	002179	002180	002181	002182	002183	002184	002185	002186	002187	002188	002189	002190	002191
0089	002192	002193	002194	002195	002196	002197	002198	002199	002200	002201	002202	002203	002204	002205	002206	002207
008A	002208	002209	002210	002211	002212	002213	002214	002215	002216	002217	002218	002219	002220	002221	002222	002223
008B	002224	002225	002226	002227	002228	002229	002230	002231	002232	002233	002234	002235	002236	002237	002238	002239
008C	002240	002241	002242	002243	002244	002245	002246	002247	002248	002249	002250	002251	002252	002253	002254	002255
008D	002256	002257	002258	002259	002260	002261	002262	002263	002264	002265	002266	002267	002268	002269	002270	002271
008E	002272	002273	002274	002275	002276	002277	002278	002279	002280	002281	002282	002283	002284	002285	002286	002287
008F	002288	002289	002290	002291	002292	002293	002294	002295	002296	002297	002298	002299	002300	002301	002302	002303

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0090	002304	002305	002306	002307	002308	002309	002310	002311	002312	002313	002314	002315	002316	002317	002318	002319
0091	002320	002321	002322	002323	002324	002325	002326	002327	002328	002329	002330	002331	002332	002333	002334	002335
0092	002336	002337	002338	002339	002340	002341	002342	002343	002344	002345	002346	002347	002348	002349	002350	002351
0093	002352	002353	002354	002355	002356	002357	002358	002359	002360	002361	002362	002363	002364	002365	002366	002367
0094	002368	002369	002370	002371	002372	002373	002374	002375	002376	002377	002378	002379	002380	002381	002382	002383
0095	002384	002385	002386	002387	002388	002389	002390	002391	002392	002393	002394	002395	002396	002397	002398	002399
0096	002400	002401	002402	002403	002404	002405	002406	002407	002408	002409	002410	002411	002412	002413	002414	002415
0097	002416	002417	002418	002419	002420	002421	002422	002423	002424	002425	002426	002427	002428	002429	002430	002431
0098	002432	002433	002434	002435	002436	002437	002438	002439	002440	002441	002442	002443	002444	002445	002446	002447
0099	002448	002449	002450	002451	002452	002453	002454	002455	002456	002457	002458	002459	002460	002461	002462	002463
009A	002454	002465	002466	002467	002468	002469	002470	002471	002472	002473	002474	002475	002476	002477	002478	002479
009B	002480	002481	002482	002483	002484	002485	002486	002487	002488	002489	002490	002491	002492	002493	002494	002495
009C	002496	002497	002498	002499	002500	002501	002502	002503	002504	002505	002506	002507	002508	002509	002510	002511
009D	002512	002513	002514	002515	002516	002517	002518	002519	002520	002521	002522	002523	002524	002525	002526	002527
009E	002528	002529	002530	002531	002532	002533	002534	002535	002536	002537	002538	002539	002540	002541	002542	002543
009F	002544	002545	002546	002547	002548	002549	002550	002551	002552	002553	002554	002555	002556	002557	002558	002559

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00A0	002560	002561	002562	002563	002564	002565	002566	002567	002568	002569	002570	002571	002572	002573	002574	002575
00A1	002576	002577	002578	002579	002580	002581	002582	002583	002584	002585	002586	002587	002588	002589	002590	002591
00A2	002592	002593	002594	002595	002596	002597	002598	002599	002600	002601	002602	002603	002604	002605	002606	002607
00A3	002608	002609	002610	002611	002612	002613	002614	002615	002616	002617	002618	002619	002620	002621	002622	002623
00A4	002624	002625	002626	002627	002628	002629	002630	002631	002632	002633	002634	002635	002636	002637	002638	002639
00A5	002640	002641	002642	002643	002644	002645	002646	002647	002648	002649	002650	002651	002652	002653	002654	002655
00A6	002656	002657	002658	002659	002660	002661	002662	002663	002664	002665	002666	002667	002668	002669	002670	002671
00A7	002672	002673	002674	002675	002676	002677	002678	002679	002680	002681	002682	002683	002684	002685	002686	002687
00A8	002688	002689	002690	002691	002692	002693	002694	002695	002696	002697	002698	002699	002700	002701	002702	002703
00A9	002704	002705	002706	002707	002708	002709	002710	002711	002712	002713	002714	002715	002716	002717	002718	002719
00AA	002720	002721	002722	002723	002724	002725	002726	002727	002728	002729	002730	002731	002732	002733	002734	002735
00AB	002736	002737	002738	002739	002740	002741	002742	002743	002744	002745	002746	002747	002748	002749	002750	002751
00AC	002752	002753	002754	002755	002756	002757	002758	002759	002760	002761	002762	002763	002764	002765	002766	002767
00AD	002768	002769	002770	002771	002772	002773	002774	002775	002776	002777	002778	002779	002780	002781	002782	002783
00AE	002784	002785	002786	002787	002788	002789	002790	002791	002792	002793	002794	002795	002796	002797	002798	002799
00AF	002800	002801	002802	002803	002804	002805	002806	002807	002808	002809	002810	002811	002812	002813	002814	002815

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00B0	002816	002817	002818	002819	002820	002821	002822	002823	002824	002825	002826	002827	002828	002829	002830	002831
00B1	002832	002833	002834	002835	002836	002837	002838	002839	002840	002841	002842	002843	002844	002845	002846	002847
00B2	002848	002849	002850	002851	002852	002853	002854	002855	002856	002857	002858	002859	002860	002861	002862	002863
00B3	002864	002865	002866	002867	002868	002869	002870	002871	002872	002873	002874	002875	002876	002877	002878	002879
00B4	002880	002881	002882	002883	002884	002885	002886	002886	002888	002889	002890	002891	002892	002893	002894	002895
00B5	002896	002897	002898	002899	002900	002901	002902	002903	002904	002905	002906	002907	002908	002909	002910	002911
00B6	002912	002913	002914	002915	002916	002917	002918	002919	002920	002921	002922	002923	002924	002925	002926	002927
00B7	002928	002929	002930	002931	002932	002933	002934	002935	002936	002937	002938	002939	002940	002941	002942	002943
00B8	002944	002945	002946	002947	002948	002949	002950	002951	002952	002953	002954	002955	002956	002957	002958	002959
00B9	002960	002961	002962	002963	002964	002965	002966	002967	002968	002969	002970	002971	002972	002973	002974	002975
00BA	002976	002977	002978	002979	002980	002981	002982	002983	002984	002985	002986	002987	002988	002989	002990	002991
00BB	002992	002993	002994	002995	002996	002997	002998	002999	003000	003001	003002	003003	003004	003005	003006	003007
00BC	003008	003009	003010	003011	003012	003013	003014	003015	003016	003017	003018	003019	003020	003021	003022	003023
00BD	003024	003025	003026	003027	003028	003029	003030	003031	003032	003033	003034	003035	003036	003037	003038	003039
00BE	003040	003041	003042	003043	003044	003045	003046	003047	003048	003049	003050	003051	003052	003053	003054	003055
00BF	003056	003057	003058	003059	003060	003061	003062	003063	003064	003065	003066	003067	003068	003069	003070	003071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00C0	003072	003073	003074	003075	003076	003077	003078	003079	003080	003081	003082	003083	003084	003085	003086	003087
00C1	003088	003089	003090	003091	003092	003093	003094	003095	003096	003097	003098	003099	003100	003101	003102	003103
00C2	003104	003105	003106	003107	003108	003109	003110	003111	003112	003113	003114	003115	003116	003117	003118	003119
00C3	003120	003121	003122	003123	003124	003125	003126	003127	003128	003129	003130	003131	003132	003133	003134	003135
00C4	003136	003137	003138	003139	003140	003141	003142	003143	003144	003145	003146	003147	003148	003149	003150	003151
00C5	003152	003153	003154	003155	003156	003157	003158	003159	003160	003161	003162	003163	003164	003165	003166	003167
00C6	003268	003169	003170	003171	003172	003173	003174	003175	003176	003177	003178	003179	003180	003181	003182	003183
00C7	003184	003185	003186	003187	003188	003189	003190	003191	003192	003193	003194	003195	003196	003197	003198	003199
00C8	003200	003201	003202	003203	003204	003205	003206	003207	003208	003209	003210	003211	003212	003213	003214	003215
00C9	003216	003217	003218	003219	003220	003221	003222	003223	003224	003225	003226	003227	003228	003229	003230	003231
00CA	003232	003233	003234	003235	003236	003237	003238	003239	003240	003241	003242	003243	003244	003245	003246	003247
00CB	003248	003249	003250	003251	003252	003253	003254	003255	003256	003257	003258	003259	003260	003261	003262	003263
00CC	003264	003265	003266	003267	003268	003269	003270	003271	003272	003273	003274	003275	003276	003277	003278	003279
00CD	003280	003281	003282	003283	003284	003285	003286	003287	003288	003289	003290	003291	003292	003293	003294	003295
00CE	003296	003297	003298	003299	003300	003301	003302	003303	003304	003305	003306	003307	003308	003309	003310	003311
00CF	003312	003313	003314	003315	003316	003317	003318	003319	003320	003321	003322	003323	003324	003325	003326	003327

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00D0	003328	003329	003330	003331	003332	003333	003334	003335	003336	003337	003338	003339	003340	003341	003342	003343
00D1	003344	003345	003346	003347	003348	003349	003350	003351	003352	003353	003354	003355	003356	003357	003358	003359
00D2	003360	003361	003362	003363	003364	003365	003366	003367	003368	003369	003370	003371	003372	003373	003374	003375
00D3	003376	003377	003378	003379	003380	003381	003382	003383	003384	003385	003386	003387	003388	003389	003390	003391
00D4	003392	003393	003394	003395	003396	003397	003398	003399	003400	003401	003402	003403	003404	003405	003406	003407
00D5	003408	003409	003410	003411	003412	003413	003414	003415	003416	003417	003418	003419	003420	003421	003422	003423
00D6	003424	003425	003426	003427	003428	003429	003430	003431	003432	003433	003434	003435	003436	003437	003438	003439
00D7	003440	003441	003442	003443	003444	003446	003446	003447	003448	003449	003450	003451	003457	003453	003454	003456
00D8	003456	003457	003458	003459	003460	003461	003462	003463	003464	003465	003466	003467	003468	003469	003470	003471
00D9	003472	003473	003474	003475	003476	003477	003478	003479	003480	003481	003482	003483	003484	003485	003486	003487
00DA	003488	003489	003490	003491	003492	003493	003494	003495	003496	003497	003498	003499	003500	003501	003502	003503
00DB	003504	003505	003506	003507	003508	003509	003510	003511	003512	003513	003514	003515	003516	003517	003518	003519
00DC	003520	003521	003522	003523	003524	003525	003526	003527	003528	003529	003530	003531	003532	003533	003534	003535
00DD	003536	003537	003538	003539	003540	003541	003542	003543	003544	003545	003546	003547	003548	003549	003550	003551
00DE	003552	003553	003554	003555	003556	003557	003558	003559	003560	003561	003562	003563	003564	003565	003566	003567
00DF	003568	003569	003570	003571	003572	003573	003574	003575	003576	003577	003578	003579	003580	003581	003582	003583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00E0	003584	003585	003586	003587	003588	003589	003590	003591	003592	003593	003594	003595	003596	003597	003598	003599
00E1	003600	003601	003602	003603	003604	003605	003606	003607	003608	003609	003610	003611	003612	003613	003614	003615
00E2	003616	003617	003618	003619	003620	003621	003622	003623	003624	003625	003626	003627	003628	003629	003630	003631
00E3	003632	003633	003634	003635	003636	003637	003638	003639	003640	003641	003642	003643	003644	003645	003646	003647
00E4	003648	003649	003650	003651	003652	003653	003654	003655	003656	003657	003658	003659	003660	003661	003662	003663
00E5	003664	003666	003666	003667	003668	003669	003670	003671	003672	003673	003674	003675	003676	003677	003678	003679
00E6	003680	003681	003682	003683	003684	003685	003686	003687	003688	003689	003690	003691	003692	003693	003694	003695
00E7	003696	003697	003698	003699	003700	003701	003702	003703	003704	003705	003706	003707	003708	003709	003710	003711
00E8	003712	003713	003714	003715	003716	003717	003718	003719	003720	003721	003722	003723	003724	003725	003726	003727
00E9	003728	003729	003730	003731	003732	003733	003734	003735	003736	003737	003738	003739	003740	003741	003742	003743
00EA	003744	003745	003746	003747	003748	003749	003750	003751	003752	003753	003754	003755	003756	003757	003758	003759
00EB	003760	003761	003762	003763	003764	003765	003766	003767	003768	003769	003770	003771	003772	003773	003774	003775
00EC	003776	003777	003778	003779	003780	003781	003782	003783	003784	003785	003786	003787	003788	003789	003790	003791
00ED	003792	003793	003794	003795	003796	003797	003798	003799	003800	003801	003802	003803	003804	003805	003806	003807
00EE	003808	003809	003810	003811	003812	003813	003814	003815	003816	003817	003818	003819	003820	003821	003822	003823
00EF	003824	003825	003826	003827	003828	003829	003830	003831	003832	003833	003834	003835	003836	003837	003838	003839

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00F0	003840	003841	003842	003843	003844	003845	003846	003847	003848	003849	003850	003851	003852	003853	003854	003856
00F1	003856	003857	003858	003859	003860	003861	003862	003863	003864	003865	003866	003867	003868	003869	003870	003871
00F2	003872	003873	003874	003875	003876	003877	003878	003879	003880	003881	003882	003883	003884	003885	003886	003887
00F3	003888	003889	003890	003891	003892	003893	003894	003895	003896	003897	003898	003899	003900	003901	003902	003903
00F4	003904	003905	003906	003907	003908	003909	003910	003911	003912	003913	003914	003915	003916	003917	003918	003919
00F5	003920	003921	003922	003923	003924	003925	003926	003927	003928	003929	003930	003931	003932	003933	003934	003935
00F6	003936	003937	003938	003939	003940	003941	003942	003943	003944	003945	003946	003947	003948	003949	003950	003951
00F7	003952	003953	003954	003955	003956	003957	003958	003959	003960	003961	003962	003963	003964	003965	003966	003967
00F8	003968	003969	003970	003971	003972	003973	003974	003975	003976	003977	003978	003979	003980	003981	003982	003983
00F9	003984	003985	003986	003987	003988	003989	003990	003991	003992	003993	003994	003995	003996	003997	003998	003999
00FA	004000	004001	004002	004003	004004	004005	004006	004007	004008	004009	004010	004011	004012	004013	004014	004015
00FB	004016	004017	004018	004019	004020	004021	004022	004023	004024	004025	004026	004027	004028	004029	004030	004031
00FC	004032	004033	004034	004035	004036	004037	004038	004039	004040	004041	004042	004043	004044	004045	004046	004047
00FD	004048	004049	004050	004051	004052	004053	004054	004055	004056	004057	004058	004059	004060	004061	004062	004063
00FE	004064	004065	004066	004067	004068	004069	004070	004071	004072	004073	004074	004075	004076	004077	004078	004079
00FF	004080	004081	004082	004083	004084	004085	004086	004087	004088	004089	004090	004091	004092	004093	004094	004095

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100	004096	004097	004098	004099	004100	004101	004102	004103	004104	004105	004106	004107	004108	004109	004110	004111
0101	004112	004113	004114	004115	004116	004117	004118	004119	004120	004121	004122	004123	004124	004125	004126	004127
0102	004128	004129	004130	004131	004132	004133	004134	004135	004136	004137	004138	004139	004140	004141	004142	004143
0103	004144	004145	004146	004147	004148	004149	004150	004151	004152	004153	004154	004155	004156	004157	004158	004159
0104	004160	004161	004162	004163	004164	004165	004166	004167	004168	004169	004170	004171	004172	004173	004174	004175
0105	004176	004177	004178	004179	004180	004181	004182	004183	004184	004185	004186	004187	004188	004189	004190	004191
0106	004192	004193	004194	004195	004196	004197	004198	004199	004200	004201	004202	004203	004204	004205	004206	004207
0107	004208	004209	004210	004211	004212	004213	004214	004215	004216	004217	004218	004219	004220	004221	004222	004223
0108	004224	004225	004226	004227	004228	004229	004230	004231	004232	004233	004234	004235	004236q	004237	004238	004239
0109	004240	004241	004242	004243	004244	004245	004246	004247	004248	004249	004250	004251	004252	004253	004254	004255
010A	004256	004257	004258	004259	004260	004261	004262	004263	004264	004265	004266	004267	004268	004269	004270	004271
010B	004272	004273	004274	004275	004276	004277	004278	004279	004280	004281	004282	004283	004284	004285	004286	004287
010C	004288	004289	004290	004291	004292	004293	004294	004295	004296	004297	004298	004299	004300	004301	004302	004303
010D	004304	004305	004306	004307	004308	004309	004310	004311	004312	004313	004314	004315	004316	004317	004318	004319
010E	004320	004321	004322	004323	004324	004325	004326	004327	004328	004329	004330	004331	004332	004333	004334	004335
010F	004336	004337	004338	004339	004340	004341	004342	004343	004344	004345	004346	004347	004348	004349	004350	004351

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0110	004352	004353	004354	004355	004356	004357	004358	004359	004360	004361	004362	004363	004364	004365	004366	004367
0111	004368	004369	004370	004371	004372	004373	004374	004375	004376	004377	004378	004379	004380	004381	004382	004383
0112	004384	004385	004386	004387	004388	004389	004390	004391	004392	004393	004394	004395	004396	004397	004398	004399
0113	004400	004401	004402	004403	004404	004405	004406	004407	004408	004409	004410	004411	004412	004413	004414	004415
0114	004416	004417	004418	004419	004420	004421	004422	004423	004424	004425	004426	004427	004428	004429	004430	004431
0115	004432	004433	004434	004435	004436	004437	004438	004439	004440	004441	004442	004443	004444	004445	004446	004447
0116	004448	004449	004450	004451	004452	004453	004454	004455	004456	004457	004458	004459	004460	004461	004462	004463
0117	004464	004465	004466	004467	004468	004469	004470	004471	004472	004473	004474	004475	004476	004477	004478	004479
0118	004480	004481	004482	004483	004484	004485	004486	004487	004488	004489	004490	004491	004492	004493	004494	004495
0119	004496	004497	004498	004499	004500	004501	004502	004503	004504	004505	004506	004507	004508	004509	004510	004511
011A	004512	004513	004514	004515	004516	004517	004518	004519	004520	004521	004522	004523	004524	004525	004526	004527
011B	004528	004529	004530	004531	004532	004533	004534	004535	004536	004537	004538	004539	004540	004541	004542	004543
011C	004544	004545	004546	004547	004548	004549	004550	004551	004552	004553	004554	004555	004556	004557	004558	004559
011D	004560	004561	004562	004563	004564	004565	004566	004567	004568	004569	004570	004571	004572	004573	004574	004575
011E	004576	004577	004578	004579	004580	004581	004582	004583	004584	004585	004586	004587	004588	004589	004590	004591
011F	004592	004593	004594	004595	004596	004597	004598	004599	004600	004601	004602	004603	004604	004605	004606	004607

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0120	004608	004609	004610	004611	004612	004613	004614	004615	004616	004617	004618	004619	004620	004621	004622	004623
0121	004624	004625	004626	004627	004628	004629	004630	004631	004632	004633	004634	004635	004636	004637	004638	004639
0122	004640	004641	004642	004643	004644	004645	004646	004647	004648	004649	004650	004651	004652	004653	004654	004655
0123	004656	004657	004658	004659	004660	004661	004662	004663	004664	004665	004666	004667	004668	004669	004670	004671
0124	004672	004673	004674	004675	004676	004677	004678	004679	004680	004681	004682	004683	004684	004685	004686	004687
0125	004688	004689	004690	004691	004692	004693	004694	004695	004696	004697	004698	004699	004700	004701	004702	004703
0126	004704	004705	004706	004707	004708	004709	004710	004711	004712	004713	004714	004715	004716	004717	004718	004719
0127	004720	004721	004722	004723	004724	004725	004726	004727	004728	004729	004730	004731	004732	004733	004734	004735
0128	004736	004737	004738	004739	004740	004741	004742	004743	004744	004745	004746	004747	004748	004749	004750	004751
0129	004752	004753	004754	004755	004756	004757	004758	004759	004760	004761	004762	004763	004764	004765	004766	004767
012A	004768	004769	004770	004771	004772	004773	004774	004775	004776	004777	004778	004779	004780	004781	004782	004783
012B	004784	004785	004786	004787	004788	004789	004790	004791	004792	004793	004794	004795	004796	004797	004798	004799
012C	004800	004801	004802	004803	004804	004805	004806	004807	004808	004809	004810	004811	004812	004813	004814	004815
012D	004816	004817	004818	004819	004820	004821	004822	004823	004824	004825	004826	004827	004828	004829	004830	004831
012E	004832	004833	004834	004835	004836	004837	004838	004839	004840	004841	004842	004843	004844	004845	004846	004847
012F	004848	004849	004850	004851	004852	004853	004854	004855	004856	004857	004858	004859	004860	004861	004862	004863

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0130	004864	004865	004866	004867	004868	004869	004870	004871	004872	004873	004874	004875	004876	004877	004878	004879
0131	004880	004881	004882	004883	004884	004885	004886	004887	004888	004889	004890	004891	004892	004893	004894	004895
0132	004896	004897	004898	004899	004900	004901	004902	004903	004904	004905	004906	004907	004908	004909	004910	004911
C133	004912	004913	004914	004915	004916	004917	004918	004919	004920	004921	004922	004923	004924	004925	004926	004927
C134	004928	004929	004930	004931	004932	004933	004934	004935	004936	004937	004938	004939	004940	004941	004942	004943
0135	004944	004945	004946	004947	004948	004949	004950	004951	004952	004953	004954	004955	004956	004957	004958	004959
0136	004960	004961	004962	004963	004964	004965	004966	004967	004968	004969	004970	004971	004972	004973	004974	004975
0137	004976	004977	004978	004979	004980	004981	004982	004983	004984	004985	004986	004987	004988	004989	004990	004991
0138	004992	004993	004994	004995	004996	004997	004998	004999	005000	005001	005002	005003	005004	005005	005006	005007
0139	005008	005009	005010	005011	005012	005013	005014	005015	005016	005017	005018	005019	005020	005021	005022	005023
013A	005024	005025	005026	005027	005028	005029	005030	005031	005032	005033	005034	005035	005036	005037	005038	005039
013B	005040	005041	005042	005043	005044	005045	005046	005047	005048	005049	005050	005051	005052	005053	005054	005055
013C	005056	005057	005058	005059	005060	005061	005062	005063	005064	005065	005066	005067	005068	005069	005070	005071
013D	005072	005073	005074	005075	005076	005077	005078	005079	005080	005081	005082	005083	005084	005085	005086	005087
013E	005088	005089	005090	005091	005092	005093	005094	005095	005096	005097	005098	005099	005100	005101	005102	005103
013F	005104	005105	005106	005107	005108	005109	005110	005111	005112	005113	005114	005115	005116	005117	005118	005119

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0140	005120	005121	005122	005123	005124	005125	005126	005127	005128	005129	005130	005131	005132	005133	005134	005135
0141	005136	005137	005138	005139	005140	005141	005142	005143	005144	005145	005146	005147	005148	005149	005150	005151
0142	005152	005153	005154	005155	005156	005157	005158	005159	005160	005161	005162	005163	005164	005165	005166	005167
0143	005168	005169	005170	005171	005172	005173	005174	005175	005176	005177	005178	005179	005180	005181	005182	005183
0144	005184	005185	005186	005187	005188	005189	005190	005191	005192	005193	005194	005195	005196	005197	005198	005199
0145	005200	005201	005202	005203	005204	005205	005206	005207	005208	005209	005210	005211	005212	005213	005214	005215
0146	005216	005217	005218	005219	005220	005221	005222	005223	005224	005225	005226	005227	005228	005229	005230	005231
0147	005232	005233	005234	005235	005236	005237	005238	005239	005240	005241	005242	005243	005244	005245	005246	005247
0148	005248	005249	005250	005251	005252	005253	005254	005255	005256	005257	005258	005259	005260	005261	005262	005263
0149	005264	005265	005266	005267	005268	005269	005270	005271	005272	005273	005274	005275	005276	005277	005278	005279
014A	005280	005281	005282	005283	005284	005285	005286	005287	005288	005289	005290	005291	005292	005293	005294	005295
014B	005296	005297	005298	005299	005300	005301	005302	005303	005304	005305	005306	005307	005308	005309	005310	005311
014C	005312	005313	005314	005315	005316	005317	005318	005319	005320	005321	005322	005323	005324	005325	005326	005327
014D	005328	005329	005330	005331	005332	005333	005334	005335	005336	005337	005338	005339	005340	005341	005342	005343
014E	005344	005345	005346	005347	005348	005349	005350	005351	005352	005353	005354	005355	005356	005357	005358	005359
014F	005360	005361	005362	005363	005364	005365	005366	005367	005368	005369	005370	005371	005372	005373	005374	005375

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0150	005376	005377	005378	005379	005380	005381	005382	005383	005384	005385	005386	005387	005388	005389	005390	005391
0151	005392	005393	005394	005395	005396	005397	005398	005399	005400	005401	005402	005403	005404	005405	005406	005407
0152	005408	005409	005410	005411	005412	005413	005414	005415	005416	005417	005418	005419	005420	005421	005422	005423
0153	005424	005425	005426	008427	005428	005429	005430	005431	005432	005433	005434	005435	005436	005437	005438	005439
0154	005440	005441	005442	005443	005444	005445	005446	005447	005448	005449	005450	005451	005452	005453	005454	005455
0155	005456	005457	005458	005459	005460	005461	005462	005463	005464	005465	005466	005467	005468	005469	005470	005471
0156	005472	005473	005474	005475	005476	005477	005478	005479	005480	005481	005482	005483	005484	005485	005486	005487
0157	005488	005489	005490	005491	005492	005493	005494	005495	005496	005497	005498	005499	005500	005501	005502	005503
0158	005504	005505	005506	005507	005508	005509	005510	006511	005512	005513	005514	005515	005516	005517	005518	005519
0159	005520	005521	005522	005523	005524	005525	005526	005527	005528	005529	005530	005531	005532	005533	005534	005535
015A	005536	005537	005538	005539	005540	005541	005542	0055a3	005544	005545	005546	005547	005548	005549	005550	005551
015B	005552	005553	005554	005555	005556	005557	005558	005559	005560	005561	005562	005563	005564	005565	005566	005567
015C	005568	005569	005570	005571	005572	C05573	005574	005575	005576	005577	005578	005579	005580	005581	005582	005583
015D	005584	005585	005586	005587	005588	005589	005590	005591	005592	005593	005594	005595	005596	005597	005598	005599
015E	005600	005601	005602	005603	005604	005605	005606	005607	005608	005609	005610	005611	005612	005613	005614	005615
015F	005616	005617	005618	005619	005620	005621	005622	005623	005624	005625	005626	005627	005628	005629	005630	005631

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0160	005632	005633	005634	005635	005636	006637q	005638	005639	005640	005641	005642	005643	005644	005645	005646q	005647
0161	005648	005649	005650	005651	005652	005653	005654	005655	005656	005657	005658	005659	005660	005661	005662	005663
0162	005664	005665	005666	005667	005668	005669	005670	005671	005672	005673	005674	005675	005676	005677	005678	005679
0163	005680	005681	005682	005683	005684	005685	005686	005687	005688	005689	005690	005691	005692	005693	005694	005695
0164	005696	005697	005698	005699	005700	005701	005702	005703	005704	005705	005706	005707	005708	005709	005710	005711
0165	005712	005713	005714	005715	005716	005717	005718	005719	005720	005721	005722	005723	005724	005725	005726	005727
0166	005728	005729	005730	005731	005732	005733	005734	005735	005736	005737	005738	005739	005740	005741	005742	005743
0167	005744	005745	005746	005747	005748	005749	005750	005751	005752	005753	005754	005755	005756	005757	005758	005759
0168	005760	005761	005762	005763	005764	005765	005766	005767	005768	005769	005770	005771	005772	005773	005774	005775
0169	005776	005777	005778	005779	005780	005781	005782	005783	005784	005785	005786	005787	005788	005789	005790	005791
016A	005792	005793	005794	005795	005796	005797	005798	005799	005800	005801	005802	005803	005804	005805	005806	005807
016B	005808	005809	005810	005811	005812	005813	005814	005815	005816	005817	005818	005819	005820	005821	005822	005823
016C	005824	005825	005826	005827	005828	005829	005830	005831	005832	005833	005834	005835	005836	005837	005838	005839
016D	005840	005841	005842	005843	005844	005845	005846	005847	005848	005849	005850	005851	005852	005853	005854	005855
016E	005856	005857	005858	005859	005860	005861	005862	005863	005864	005865	005866	005867	005868	005869	005870	005871
016F	005872	005873	005874	005875	005876	005877	005878	005879	005880	005881	005882	005883	005884	005885	005886	005887

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0170	005888	005889	005890	005891	005892	005893	005894	005895	005896	005897	005898	005899	005900	005901	005902	005903
0171	005904	005905	005908	005907	005908	005909	005910	005911	005912	005913	005914	005915	005916	005917	005918	005919
0172	005920	005921	005922	005923	005924	005925	005926	005927	005928	005929	005930	005931	005932	005933	005934	005935
0173	005936	005937	005938	005939	005940	005941	005942	005943	005944	005945	005946	005947	005948	005949	005950	005951
0174	005952	005953	005954	005955	005956	005957	005958	005959	005960	005961	005962	005963	005964	005965	005966	005967
0175	005968	005969	005970	005971	005972	005973	005974	005975	005976	005977	005978	005979	005980	005981	005982	005983
0176	005984	005985	005986	005987	005988	005989	005990	005991	005992	005993	005994	005995	005996	005997	005998	005999
0177	006000	006001	006002	006003	006004	006005	006006	006007	006008	006009	006010	006011	006012	006013	006014	006015
0178	006016	006017	006018	006019	006020	006021	006022	006023	006024	006025	006026	006027	006028	006029	006030	006031
0179	006032	006033	006034	006035	006036	006037	006038	006039	006040	006041	006042	006043	006044	006045	006046	006047
017A	006048	006049	006050	006051	006052	006053	006054	006055	006056	006057	006058	006059	006060	006061	006062	006063
017B	006064	006065	006066	006067	006068	006069	006070	006071	006072	006073	006074	006075	006076	006077	006078	006079
017C	006080	006081	006082	006083	006084	006085	006086	006087	006088	006089	006090	006091	006092	006093	006094	006095
017D	006096	006097	006098	006099	006100	006101	006102	006103	006104	006105	006106	006107	006108	006109	006110	006111
017E	006112	006113	006114	006115	006116	006117	006118	006119	006120	006121	006122	006123	006124	006125	006126	006127
017F	006128	006129	006130	006131	006132	006133	006134	006135	006136	006137	006138	006139	006140	006141	006142	006143

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0180	006144	006145	006146	006147	006148	006149	006150	006151	006152	006153	006154	006155	006156	006157	006158	006159
0181	006160	006161	006162	006163	006164	006165	006166	006167	006168	006169	006170	006171	006172	006173	006174	006175
0182	006176	006177	006178	006179	006180	006181	006182	006183	006184	006185	006186	006187	006188	006189	006190	006191
0183	006192	006193	006194	006195	006196	006197	006198	006199	006200	006201	006202	006203	006204	006205	006206	006207
0184	006208	006209	006210	006211	006212	006213	006214	006215	006216	006217	006218	006219	006220	006221	006222	006223
0185	006224	006225	006726	006227	006728	006229	006230	006231	006232	006233	006234	006235	006236	006237	006238	006239
0186	006240	006241	006242	006243	006244	006245	006246	006247	006248	006249	006250	006251	006252	006253	006254	006255
0187	006256	006257	006258	006259	006260	006261	006262	006263	006264	006265	006266	006267	006268	006269	006270	006271
0188	006272	006273	006274	006275	006276	006277	006278	006279	006280	006281	006282	006283	006284	006285	006286	006287
0189	006288	006289	006290	006291	006292	006293	006294	006295	006296	006297	006298	006299	006300	006301	006302	006303
018A	006304	006305	006306	006307	006308	006309	006310	006311	006312	006313	006314	006315	006316	006317	006318	006319
018B	006320	006321	006322	006323	006324	006325	006326	006327	006328	006329	006330	006331	006332	006333	006334	006335
018C	006336	006337	006338	006339	006340	006341	006342	006343	006344	006345	006346	006347	006348	006349	006350	006351
018D	006352	006353	006354	006355	006356	006357	006358	006359	006360	006361	006362	006363	006364	006365	006366	006367
018E	006368	006369	006370	006371	006372	006373	006374	006375	006376	006377	006378	006379	006380	006381	006382	006383
018F	006384	006385	006386	006387	006388	006389	006390	006391	006392	006393	006394	006395	006396	006397	006398	006399

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0190	006400	006401	006402	006403	006404	006405	006406	006407	006408	006409	006410	006411	006412	006413	006414	006415
0191	006416	006417	006418	006419	006420	006421	006422	006423	006424	006425	006426	006427	006428	006429	006430	006431
0192	006432	006433	006434	006435	006436	006437	006438	006439	006440	006441	006442	006443	006444	006445	006446	006447
0193	006448	006449	006450	006451	006452	006453	006454	006455	006456	006457	006458	006459	006460	006461	006462	006463
0194	006464	006465	006466	006467	006468	006469	006470	006471	006472	006473	006474	006475	006476	006477	006478	006479
0195	006480	006481	006482	006483	006484	006485	006486	006487	006488	006489	006490	006491	006492	006493	006494	006495
0196	006496	006497	006498	006499	006500	006501	006502	006503	006504	006505	006506	006507	006508	006509	006510	006511
0197	006512	006513	006514	006515	006516	006517	006518	006519	006520	006521	006522	006523	006524	006525	006526	006527
0198	006528	006529	006530	006531	006532	006533	006534	006535	006536	006537	006538	006539	006540	006541	006542	006543
0199	006544	006545	006546	006547	006548	006549	006550	006551	006552	006553	006554	006556	006556	006557	006558	006559
019A	006560	006561	006562	006563	006564	006565	006566	006567	006568	006569	006570	006571	006572	006573	006574	006575
019B	006576	006577	006578	006579	006580	006581	006582	006583	006584	006585	006586	006587	006588	006589	006590	006591
019C	006592	006593	006594	006595	006596	006597	006598	006599	006600	006601	006602	006603	006604	006605	006606	006607
019D	006608	006609	006610	006611	006612	006613	006614	006615	006616	006617	006618	006619	006620	006621	006622	006623
019E	006624	006625	006626	006627	006628	006629	006630	006631	006632	006633	006634	006635	006636	006637	006638	006639
019F	006640	006641	006642	006643	006644	006645	006646	006647	006648	006649	006650	006651	006652	006653	006654	006655

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01A0	006656	006657	006658	006659	006660	006661	006662	006663	006664	006665	006666	006667	006668	006669	006670	006671
01A1	006672	006673	006674	006675	006676	006677	006678	006679	006680	006681	006682	006683	006684	006685	006686	006687
01A2	006688	006689	006690	006691	006692	006693	006694	006695	006696	006697	006698	006699	006700	006701	006702	006703
01A3	006704	006705	006706	006707	006708	006709	006710	006711	006712	006713	006714	006715	006716	006717	006718	006719
01A4	006720	006721	006722	006723	006724	006725	006726	006727	006728	006729	006730	006731	006732	006733	006734	006735
01A5	006736	006737	006738	006739	006740	006741	006742	006743	006744	006745	006746	006747	006748	006749	006750	006751
01A6	006752	006753	006754	006755	006756	006757	006758	006759	006760	006761	006762	006763	006764	006765	006766	006767
01A7	006768	006769	006770	006771	006772	006773	006774	006775	006776	006777	006778	006779	006780	006781	006782	006783
01A8	006784	006785	006786	006787	006788	006789	006790	006791	006792	006793	006794	006795	006796	006797	006798	006799
01A9	006800	006801	006802	006803	006804	006805	006806	006807	006808	006809	006810	006811	006812	006813	006814	006815
01AA	006816	006817	006818	006819	006820	006821	006822	006823	006824	006825	006826	006827	006828	006829	006830	006831
01AB	006832	006833	006834	006835	006836	006837	006838	006839	006840	006841	006842	006843	006844	006845	006846	006847
01AC	006848	006849	006850	006851	006852	006853	006854	006855	006856	006857	006858	006859	006860	006861	006862	006863
01AD	006864	006865	006866	006867	006868	006869	006870	006871	006872	006873	006874	006875	006876	006877	006878	006879
01AE	006880	006881	006882	006883	006884	006885	006886	006887	006888	006889	006890	006891	006892	006893	006894	006895
01AF	006896	006897	006898	006899	006900	006901	006902	006903	006904	006905	006906	006907	006908	006909	006910	006911

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01B0	006912	006913	006914	006915	006916	006917	006918	006919	006920	006921	006922	006923	006924	006925	006926	006927
01B1	006928	006929	006930	006931	006932	006933	006934	006935	006936	006937	006938	006939	006940	006941	006942	006943
01B2	006944	006945	006946	006947	006948	006949	006950	006951	006952	006953	006954	006955	006956	006957	006958	006959
01B3	006960	006961	006962	006963	006964	006965	006966	006967	006968	006969	006970	006971	006972	006973	006974	006975
01B4	006976	006977	006978	006979	006980	006981	006982	006983	006984	006985	006986	006987	006988	006989	006990	006991
01B5	006992	006993	006994	006995	006996	006997	006998	006999	007000	007001	007002	007003	007004	007005	007006	007007
01B6	007008	007009	007010	007011	007012	007013	007014	007015	007016	007017	007018	007019	007020	007021	007022	007023
01B7	007024	007025	007026	007027	007028	007029	007030	007031	007032	007033	007034	007035	007036	007037	007038	007039
01B8	007040	007041	007042	007043	007044	007045	007046	007047	007048	007049	007050	007051	007052	007053	007054	007055
01B9	007056	007057	007058	007059	007060	007061	007062	007063	007064	007065	007066	007067	007068	007069	007070	007071
01BA	007072	007073	007074	007075	007076	007077	007078	007079	007080	007081	007082	007083	007084	007085	007086	007087
01BB	007088	007089	007090	007091	007092	007093	007094	007095	007096	007097	007098	007099	007100	007101	007102	007103
01BC	007104	007105	007106	007107	007108	007109	007110	007111	007112	007113	007114	007115	007116	007117	007118	007119
01BD	007120	007121	007122	007123	007124	007125	007126	007127	007128	007129	007130	007131	007132	007133	007134	007135
01BE	007136	007137	007138	007139	007140	007141	007142	007143	007144	007145	007146	007147	007148	007149	007150	007151
01BF	007152	007153	007154	007155	007156	007157	007158	007159	007160	007161	007162	007163	007164	007165	007166	007167

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01C0	007168	007169	007170	007171	007172	007173	007174	007175	007176	007177	007178	007179	007180	007181	007182	007183
01C1	007184	007185	007186	007187	007188	007189	007190	007191	007192	007193	007194	007195	007196	007197	007198	007199
01C2	007200	007201	007202	007203	007204	007205	007206	007207	007208	007209	007210	007211	007212	007213	007214	007215
01C3	007216	007217	007218	007219	007220	007221	007222	007223	007224	007225	007226	007227	007228	007229	007230	007231
01C4	007232	007233	007234	007235	007236	007237	007238	007239	007240	007241	007242	007243	007244	007245	007246	007247
01C5	007248	007249	007250	007251	007252	007253	007254	007255	007256	007257	007258	007259	007260	007261	007262	007263
01C6	007264	007265	007266	007267	007268	007269	007270	007271	007272	007273	007274	007275	007276	007277	007278	007279
01C7	007280	007281	007282	007283	007284	007285	007286	007287	007288	007289	007290	007291	007292	007293	007294	007295
01C8	007296	007297	007298	007299	007300	007301	007302	007303	007304	007305	007306	007307	007308	007309	007310	007311
01C9	007312	007313	007314	007315	007316	007317	007318	007319	007320	007321	007322	007323	007324	007325	007326	007327
01CA	007328	007329	007330	007331	007332	007333	007334	007335	007336	007337	007338	007339	007340	007341	007342	007343
01CB	007344	007345	007346	007347	007348	007349	007350	007351	007352	007353	007354	007355	007356	007357	007358	007359
01CC	007360	007361	007362	007363	007364	007365	007366	007367	007368	007369	007370	007371	007372	007373	007374	007375
01CD	007376	007377	007378	007379	007380	007381	007382	007383	007384	007385	007386	007387	007388	007389	007390	007391
01CE	007392	007393	007394	007395	007396	007397	007398	007399	007400	007401	007402	007403	007404	007405	007406	007407
01CF	007408	007409	007410	007411	007412	007413	007414	007415	007416	007417	007418	007419	007420	007421	007422	007423

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01D0	007424	007425	007426	007427	007428	007429	007430	007431	007432	007433	007434	007435	007436	007437	007438	007439
01D1	007440	007441	007442	007443	007444	007445	007446	007447	007448	007449	007450	007451	007452	007453	007454	007455
01D2	007456	007457	007458	007459	007460	007461	007462	007463	007464	007465	007466	007467	007468	007469	007470	007471
01D3	007472	007473	007474	007475	007476	007477	007478	007479	007480	007481	007482	007483	007484	007485	007486	007487
01D4	007488	007489	007490	007491	007492	007493	007494	007495	007496	007497	007498	007499	007500	007501	007502	007503
01D5	007504	007505	007506	007507	007508	007509	007510	007511	007512	007513	007514	007515	007516	007517	007518	007519
01D6	007520	007521	007522	007523	007524	007525	007526	007527	007528	007529	007530	007531	007532	007533	007534	007535
01D7	007536	007537	007538	007539	007540	007541	007542	007543	007544	007545	007546	007547	007548	007549	007550	007551
01D8	007552	007553	007554	007555	007556	007557	007558	007559	007560	007561	007562	007563	007564	007565	007566	007567
01D9	007568	007569	007570	007571	007572	007573	007574	007575	007576	007577	007578	007579	007580	007581	007582	007583
01DA	007584	007585	007586	007587	007588	007589	007590	007591	007592	007593	007594	007595	007596	007597	007598	007599
01DB	007600	007601	007602	007603	007604	007605	007606	007607	007608	007609	007610	007611	007612	007613	007614	007615
01DC	007616	007617	007618	007619	007620	007621	007622	007623	007624	007625	007626	007627	007628	007629	007630	007631
01DD	007632	007633	007634	007635	007636	007637	007638	007639	007640	007641	007642	007643	007644	007645	007646	007647
01DE	007648	007649	007650	007651	007652	007653	007654	007655	007656	007657	007658	007659	007660	007661	007662	007663
01DF	007664	007665	007666	007667	007668	007669	007670	007671	007672	007673	007674	007675	007676	007677	007678	007679

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01E0	007680	007681	007682	007683	007684	007685	007686	007687	007688	007689	007690	007691	007692	007693	007694	007695
01E1	007696	007697	007698	007699	007700	007701	007702	007703	007704	007705	007706	007707	007708	007709	007710	007711
01E2	007712	007713	007714	007715	007716	007717	007718	007719	007720	007721	007722	007723	007724	007725	007726	007727
01E3	007728	007729	007730	007731	007732	007733	007734	007735	007736	007737	007738	007739	007740	007741	007742	007743
01E4	007744	007745	007746	007747	007748	007749	007750	007751	007752	007753	007754	007755	007756	007757	007758	007759
01E5	007760	007761	007762	007763	007764	007765	007766	007767	007768	007769	007770	007771	007772	007773	007774	007775
01E6	007776	007777	007778	007779	007780	007781	007782	007783	007784	007785	007786	007787	007788	007789	007790	007791
01E7	007792	007793	007794	007795	007796	007797	007798	007799	007800	007801	007802	007803	007804	007805	007806	007807
01E8	007808	007809	007810	007811	007812	007813	007814	007815	007816	007817	007818	007819	007820	007821	007822	007823
01E9	007824	007825	007826	007827	007828	007829	007830	007831	007832	007833	007834	007835	007836	007837	007838	007839
01EA	007840	007841	007842	007843	007844	007845	007846	007847	007848	007849	007850	007851	007852	007853	007854	007856
01EB	007856	007857	007858	007859	007860	007861	007862	007863	007864	007865	007866	007867	007868	007869	007870	007871
01EC	007872	007873	007874	007875	007876	007877	007878	007879	007880	007881	007882	007883	007884	007885	007886	007887
01ED	007888	007889	007890	007891	007892	007893	007894	007895	007896	007897	007898	007899	007900	007901	007902	007903
01EE	007904	007905	007906	007907	007908	007909	007910	007911	007912	007913	007914	007915	007916	007917	007918	007919
01EF	007920	007921	007922	007923	007924	007925	007926	007927	007928	007929	007930	007931	007932	007933	007934	007935

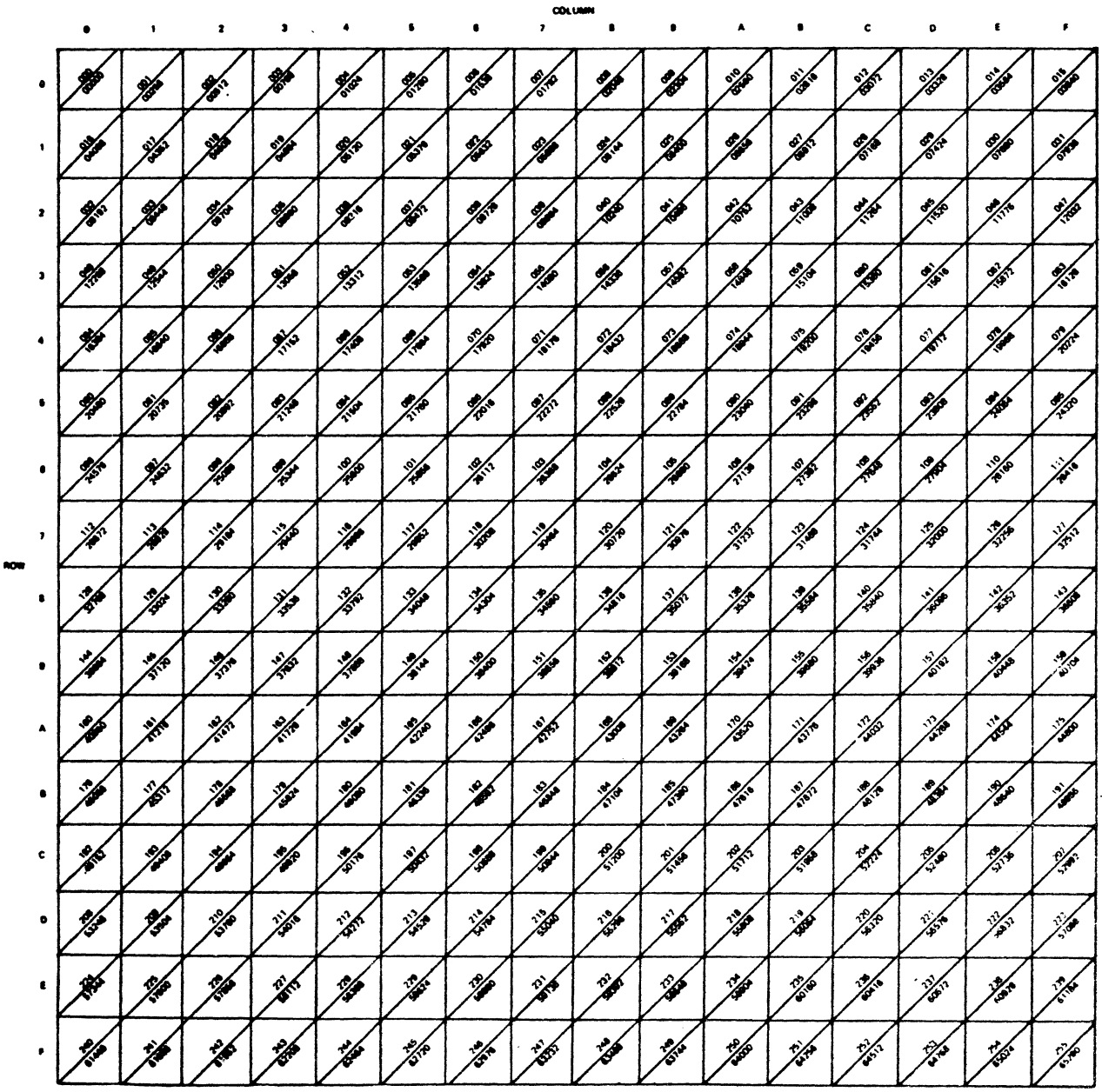
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01F0	007936	007937	007938	007939	007940	007941	007942	007943	007944	007945	007946	007947	007948	007949	007950	007951
01F1	007952	007953	007954	007955	007956	007957	007958	007959	007960	007961	007962	007963	007964	007965	007966	007967
01F2	007968	007969	007970	007971	007972	007973	007974	007975	007976	007977	007978	007979	007980	007981	007982	007983
01F3	007984	007985	007986	007987	007988	007989	007990	007991	007992	007993	007994	007995	007996	007997	007998	007999
01F4	008000	008001	008002	008003	008004	008005	008006	008007	008008	008009	008010	008011	008012	008013	008014	008015
01F5	008016	008017	008018	008019	008020	008021	008022	008023	008024	008025	008026	008027	008028	008029	008030	008031
01F6	008032	008033	008034	008035	008036	008037	008038	008039	008040	008041	008042	008043	008044	008045	008046	008047
01F7	008048	008049	008050	008051	008052	008053	008054	008055	008056	008057	008058	008059	008060	008061	008062	008063
01F8	008064	008065	008066	008067	008068	008069	008070	008071	008072	008073	008074	008075	008076	008077	008078	008079
01F9	008080	008081	008082	008083	008084	008085	008086	008087	008088	008089	008090	008091	008092	008093	008094	008095
01FA	008096	008097	008098	008099	008100	008101	008102	008103	008104	008105	008106	008107	008108	008109	008110	008111
01FB	008112	008113	008114	008115	008116	008117	008118	008119	008120	008121	008122	008123	008124	008125	008126	008127
01FC	008128	008129	008130	008131	008132	008133	008134	008135	008136	008137	008138	008139	008140	008141	008142	008143
01FD	008144	008145	008146	008147	008148	008149	008150	008151	008152	008153	008154	008155	008156	008157	008158	008159
01FE	008160	008161	008162	008163	008164	008165	008166	008167	008168	008169	008170	008171	008172	008173	008174	008175
01FF	008176	008177	008178	008179	008180	008181	008182	008183	008184	008185	008186	008187	008188	008189	008190	008191

APPENDIX C

HEXADECIMAL CONVERSION TABLE

Converting to hexadecimal may be simplified by using the following table.

To convert $(61275)_{10}$ to hexadecimal, using the table: the table entry closest to, but not greater than, $(61275)_{10}$ is $(61184)_{10}$, which equals $(EF00)_{16}$ from the table. Subtracting 61,184 from the original number $(61275\text{-}61184)_{10}$ leaves a remainder of $(91)_{10}$, which equals $(5B)_{16}$. Therefore, $(61275)_{10} = (EF5B)_{16}$.



APPENDIX D

HEXADECIMAL ADDITIONS

In the following Hexadecimal Addition Table, all values represent the result of an addition of a hexadecimal character from the column across the top and the column down the left side. The result of the addition is found where the two characters to be added intersect within the table. All values above the slanted line represent the result of an addition with no carry generated; all those values below the slanted line represent the result of an addition with a carry of one generated into the next character position of the hexadecimal result.

HEXADECIMAL ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1
3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2
4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3
5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4
6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5
7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6
8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A
C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B
D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C
E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D
F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E

APPENDIX E

NUMERICAL INFORMATION

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 380 086 994 171 142 578 125

APPENDIX F

TABLE OF POWERS OF SIXTEEN

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 × 10
16	1	0.62500 00000 00000 00000 × 10^{-1}
256	2	0.39062 50000 00000 00000 × 10^{-2}
4 096	3	0.24414 06250 00000 00000 × 10^{-3}
65 536	4	0.15258 78906 25000 00000 × 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 × 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 × 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 × 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 × 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 × 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 × 10^{-12}
17 592 186 044 416	11	0.56843 41886 08080 14870 × 10^{-13}
281 474 976 710 656	12	0.35527 13678 80050 09294 × 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 × 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 × 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 × 10^{-18}

TABLE OF POWERS OF TEN

10^n	n	10^{-n}
1	0	1.0000 0000 0000 0000
A	1	0.1999 9999 9999 999A
64	2	0.28F5 C28F 5C28 F5C3 × 16^{-1}
3E8	3	0.4189 374B C6A7 EF9E × 16^{-2}
2710	4	0.68DB 8BAC 710C B296 × 16^{-3}
1 86A0	5	0.A7C5 AC47 1B47 8423 × 16^{-4}
F 4240	6	0.10C6 F7A0 B5ED 8D37 × 16^{-4}
98 9680	7	0.1AD7 F29A BCAF 4858 × 16^{-5}
5F5 E100	8	0.2AF3 1DC4 6118 73BF × 16^{-6}
3B9A CA00	9	0.44B8 2FA0 9B5A 52CC × 16^{-7}
2 540B E400	10	0.6DF3 7F67 5EF6 EADF × 16^{-8}
17 4876 E800	11	0.AFEB FF0B CB24 AAFF × 16^{-9}
E8 D4A5 1000	12	0.1197 9981 2DEA 1119 × 16^{-9}
918 4E72 A000	13	0.1C25 C268 4976 81C2 × 16^{-10}
5AF3 107A 4000	14	0.2D09 370D 4257 3604 × 16^{-11}
3 8D7E A4C6 8000	15	0.480E BE7B 9D58 566D × 16^{-12}
23 86F2 6FC1 0000	16	0.734A CA5F 6226 F0AE × 16^{-13}
163 4578 5D8A 0000	17	0.B877 AA32 36A4 B449 × 16^{-14}
DF0 B6B3 A764 0000	18	0.1272 5DD1 D243 ABA1 × 16^{-14}
8AC7 2304 89E8 0000	19	0.1D83 C94F B6D2 AC35 × 16^{-15}

APPENDIX G

ASCII INTERCHANGE CODE SET WITH CARD PUNCH CODES

Row	Col	0	1	2	3	4	5	6	7
Bit Positions 4 5 6 7		0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP No punch	0 0	@ 8-4	P 11-7	. 8-1	p 12-11-7
0001	1	SOH 12-9-1	DCI 11-9-1	! 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2
0100	4	EOT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8
1010	A	LF 0-9-5	SUB 9-8-7	* 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9
1011	B	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0
1100	C	FF 12-9-8-4	FS 11-9-8-4	, 0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	¦ 12-11
1101	D	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-2	m 12-11-4	} 11-0
1110	E	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	^ 11-8-7	n 12-11-5	~ 11-0-1
1111	F	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	_ 0-8-5	o 12-11-6	DEL 12-9-7

Some positions in the ASCII code chart may have a different graphic representation on various devices as:

ASCII	IBM 029
!	¦
[	¢
]	!
^	>

Control Characters:

NUL	–	Null	DC3	–	Device Control 3
SOH	–	Start of Heading (CC)	DC4	–	Device Control 4 (stop)
STX	–	Start of Text (CC)	NAK	–	Negative Acknowledge (CC)
ETX	–	End of Text (CC)	SYN	–	Synchronous Idle (CC)
EOT	–	End of Transmission (CC)	ETB	–	End of Transmission Block (CC)
ENQ	–	Enquiry (CC)	CAN	–	Cancel
ACK	–	Acknowledge (CC)	EM	–	End of Medium
BEL	–	Bell (audible or attention signal)	SS	–	Start of Special Sequence
BS	–	Backspace (FE)	ESC	–	Escape
HT	–	Horizontal Tabulation (punch card skip) (FE)	FS	–	File Separator (IS)
LF	–	Line Feed (FE)	GS	–	Group Separator (IS)
VT	–	Vertical Tabulation (FE)	RS	–	Record Separator (IS)
FF	–	Form Feed (FE)	US	–	Unit Separator (IS)
CR	–	Carriage Return (FE)	DEL	–	Delete
SO	–	Shift Out	SP	–	Space (normally nonprinting)
SI	–	Shift In	(CC)	–	Communication Control
DLE	–	Data Link Escape (CC)	(FE)	–	Format Effector
DC1	–	Device Control 1	(IS)	–	Information Separator
DC2	–	Device Control 2			

32/70 SERIES INSTRUCTIONS BY OP CODE

OP CODE	MNEMONIC	DESCRIPTION	PAGE
0000	HALT	HALT	6-188
0001	WAIT	WAIT	6-189
0002	NOP	NO OPERATION	6-190
0003	LCS	LOAD CONTROL SWITCHES	6-184
0004	ES	EXTEND SIGN	6-168
0005	RND	ROUND REGISTER	6-169
0006	BEI	BLOCK EXTERNAL INTERRUPTS	6-210
0007	UEI	UNBLOCK EXTERNAL INTERRUPTS	6-211
0008	EAE	ENABLE ARITHMETIC EXCEPTION TRAP	6-196
0009	RDSTS	READ CPU STATUS WORD	6-194
000A	SIPU	START IPU	6-190A
000D	SEA	SET EXTENDED ADDRESSING	6-59
000E	DAE	DISABLE ARITHMETIC EXCEPTION TRAP	6-197
000F	CEA	CLEAR EXTENDED ADDRESSING	6-60
0400	ANR	AND REGISTER AND REGISTER	6-99
0800	ORR	OR REGISTER AND REGISTER	6-104
0808	ORRM	OR REGISTER AND REGISTER MASKED	6-105
0C00	EOR	EXCLUSIVE OR REGISTER AND REGISTER	6-110
0C00	ZR	ZERO REGISTER	6-43
0C08	EORM	EXCLUSIVE OR REGISTER AND REGISTER MASKED	6-111
1000	CAR	COMPARE ARITHMETIC WITH REGISTER	6-87
1400	CMR	COMPARE MASKED WITH REGISTER	6-93
1800	SBR	SET BIT IN REGISTER	6-129
1C00	ZBR	ZERO BIT IN REGISTER	6-130
2000	ABR	ADD BIT IN REGISTER	6-133
2400	TBR	TEST BIT IN REGISTER	6-135
2800	TRSW	TRANSFER REGISTER TO PSWR	6-57
2C00	TRR	TRANSFER REGISTER TO REGISTER	6-47
2C03	TRC	TRANSFER REGISTER COMPLEMENT	6-53
2C04	TRN	TRANSFER REGISTER NEGATIVE	6-51
2C05	XCR	EXCHANGE REGISTERS	6-55
2C07	LMAP	LOAD MAP	6-61
2C08	TRRM	TRANSFER REGISTER TO REGISTER MASKED	6-48
2C09	SETCPU	SET CPU MODE	6-193
2C0A	TMAPR	TRANSFER MAP TO REGISTER	6-62
2C0B	TRCM	TRANSFER REGISTER COMPLEMENT MASKED	6-54
2C0C	TRNM	TRANSFER REGISTER NEGATIVE MASKED	6-52
2C0D	XCRM	EXCHANGE REGISTERS MASKED	6-56
2C0E	TRSC	TRANSFER REGISTER TO SCRATCHPAD	6-46
2C0F	TSCR	TRANSFER SCRATCHPAD TO REGISTER	6-45
3000	CALM	CALL MONITOR	6-191
3400	LA	LOAD ADDRESS	6-25
3800	ADR	ADD REGISTER TO REGISTER	6-144
3808	ADRM	ADD REGISTER TO REGISTER MASKED	6-145
3C00	SUR	SUBTRACT REGISTER FROM REGISTER	6-155
3C08	SURM	SUBTRACT REGISTER FROM REGISTER MASKED	6-156
4000	MPR	MULTIPLY REGISTER BY REGISTER	6-161
4400	DVR	DIVIDE REGISTER BY REGISTER	6-166
6000	NOR	NORMALIZE	6-113
6400	NORD	NORMALIZE DOUBLE	6-114
6800	SCZ	SHIFT AND COUNT ZEROS	6-115
6C00	SRA	SHIFT RIGHT ARITHMETIC	6-121
6C40	SLA	SHIFT LEFT ARITHMETIC	6-116
7000	SRL	SHIFT RIGHT LOGICAL	6-122
7040	SLL	SHIFT LEFT LOGICAL	6-117
7400	SRC	SHIFT RIGHT CIRCULAR	6-123
7440	SLC	SHIFT LEFT CIRCULAR	6-118
7800	SRAD	SHIFT RIGHT ARITHMETIC DOUBLE	6-124
7C00	SRLD	SHIFT RIGHT LOGICAL DOUBLE	6-125
7C40	SLLD	SHIFT LEFT LOGICAL DOUBLE	6-120
8000	LEAR	LOAD EFFECTIVE ADDRESS REAL	6-24
8400	ANMH	AND MEMORY HALFWORD	6-96
8400	ANMW	AND MEMORY WORD	6-97
8400	ANMD	AND MEMORY DOUBLEWORD	6-98
8408	ANMB	AND MEMORY BYTE	6-95
8800	ORMH	OR MEMORY HALFWORD	6-101
8800	ORMW	OR MEMORY WORD	6-102
8800	ORMD	OR MEMORY DOUBLEWORD	6-103
8808	ORMB	OR MEMORY BYTE	6-100
8C00	EOMH	EXCLUSIVE OR MEMORY HALFWORD	6-107
8C00	EOMW	EXCLUSIVE OR MEMORY WORD	6-108
8C00	EOMD	EXCLUSIVE OR MEMORY DOUBLEWORD	6-109
8C08	EOMB	EXCLUSIVE OR MEMORY BYTE	6-106
9000	CAMH	COMPARE ARITHMETIC WITH MEMORY HALFWORD	6-84
9000	CAMW	COMPARE ARITHMETIC WITH MEMORY WORD	6-85
9000	CAMD	COMPARE ARITHMETIC WITH MEMORY DOUBLEWORD	6-86
9008	CAMB	COMPARE ARITHMETIC WITH MEMORY BYTE	6-83
9400	CMMH	COMPARE MASKED WITH MEMORY HALFWORD	6-90
9400	CMMW	COMPARE MASKED WITH MEMORY WORD	6-91
9400	CMMD	COMPARE MASKED WITH MEMORY DOUBLEWORD	6-92
9408	CMMB	COMPARE MASKED WITH MEMORY BYTE	6-89
9808	SBM	SET BIT IN MEMORY	6-128
9C08	ZBM	ZERO BIT IN MEMORY	6-130
A008	ABM	ADD BIT IN MEMORY	6-132
A408	TBM	TEST BIT IN MEMORY	6-134
A800	EXM	EXECUTE MEMORY	6-187
AC00	LH	LOAD HALFWORD	6-11
AC00	LW	LOAD WORD	6-12
AC00	LD	LOAD DOUBLEWORD	6-13
AC08	LB	LOAD BYTE	6-10

OP CODE	MNEMONIC	DESCRIPTION	PAGE
B000	LMH	LOAD MASKED HALFWORD	6-15
B000	LMW	LOAD MASKED WORD	6-16
B000	LMD	LOAD MASKED DOUBLEWORD	6-17
B008	LMB	LOAD MASKED BYTE	6-14
B400	LNH	LOAD NEGATIVE HALFWORD	6-19
B400	LNW	LOAD NEGATIVE WORD	6-20
B400	LND	LOAD NEGATIVE DOUBLEWORD	6-21
B408	LNB	LOAD NEGATIVE BYTE	6-18
B800	ADMH	ADD MEMORY HALFWORD	6-96
B800	ADMW	ADD MEMORY WORD	6-97
B800	ADMD	ADD MEMORY DOUBLEWORD	6-98
B808	ADMB	ADD MEMORY BYTE	6-95
BC00	SUMH	SUBTRACT MEMORY HALFWORD	6-152
BC00	SUMW	SUBTRACT MEMORY WORD	6-153
BC00	SUMD	SUBTRACT MEMORY DOUBLEWORD	6-154
BC08	SUMB	SUBTRACT MEMORY BYTE	6-151
C000	MPMH	MULTIPLY BY MEMORY HALFWORD	6-159
C000	MPMW	MULTIPLY BY MEMORY WORD	6-160
C008	MPMB	MULTIPLY BY MEMORY BYTE	6-158
C400	DVMH	DIVIDE BY MEMORY HALFWORD	6-164
C400	DVMW	DIVIDE BY MEMORY WORD	6-165
C408	DVMB	DIVIDE BY MEMORY BYTE	6-163
C800	LI	LOAD IMMEDIATE	6-22
C801	ADI	ADD IMMEDIATE	6-150
C802	SUI	SUBTRACT IMMEDIATE	6-157
C803	MPI	MULTIPLY IMMEDIATE	6-162
C804	DVI	DIVIDE IMMEDIATE	6-167
C805	CI	COMPARE IMMEDIATE	6-88
C806	SVC	SUPERVISOR CALL	6-192
C807	EXRR	EXECUTE REGISTER RIGHT	6-186
C807	EXR	EXECUTE REGISTER	6-185
CC00	LF	LOAD FILE	6-28
D000	LEA	LOAD EFFECTIVE ADDRESS	6-23
D400	STH	STORE HALFWORD	6-30
D400	STW	STORE WORD	6-31
D400	STD	STORE DOUBLEWORD	6-32
D408	STB	STORE BYTE	6-29
D800	STMH	STORE MASKED HALFWORD	6-34
D800	STMW	STORE MASKED WORD	6-35
D800	STMD	STORE MASKED DOUBLEWORD	6-36
D808	STMB	STORE MASKED BYTE	6-33
DC00	STF	STORE FILE	6-37
E000	SUFW	SUBTRACT FLOATING-POINT WORD	6-174
E000	SUFD	SUBTRACT FLOATING-POINT DOUBLEWORD	6-175
E008	ADFW	ADD FLOATING-POINT WORD	6-172
E008	ADFD	ADD FLOATING-POINT DOUBLEWORD	6-173
E400	DVFW	DIVIDE FLOATING-POINT WORD	6-178
E400	DVFD	DIVIDE FLOATING-POINT DOUBLEWORD	6-179
E408	MPFW	MULTIPLY FLOATING-POINT WORD	6-176
E408	MPFD	MULTIPLY FLOATING-POINT DOUBLEWORD	6-177
E800	ARMH	ADD REGISTER TO MEMORY HALFWORD	6-147
E800	ARMW	ADD REGISTER TO MEMORY WORD	6-148
E800	ARMD	ADD REGISTER TO MEMORY DOUBLEWORD	6-149
E808	ARMB	ADD REGISTER TO MEMORY BYTE	6-146
EC00	BU	BRANCH UNCONDITIONALLY	6-72
EC00	BCT	BRANCH CONDITION TRUE	6-74
F000	BCF	BRANCH CONDITION FALSE	6-73
F000	BFT	BRANCH FUNCTION TRUE	6-75
F400	BIB	BRANCH AFTER INCREMENTING BYTE	6-77
F420	BIH	BRANCH AFTER INCREMENTING HALFWORD	6-78
F440	BIW	BRANCH AFTER INCREMENTING WORD	6-79
F460	BID	BRANCH AFTER INCREMENTING DOUBLEWORD	6-80
F800	ZMH	ZERO MEMORY HALFWORD	6-40
F800	ZMW	ZERO MEMORY WORD	6-41
F800	ZMD	ZERO MEMORY DOUBLEWORD	6-42
F808	ZMB	ZERO MEMORY BYTE	6-39
F880	BL	BRANCH AND LINK	6-76
F900	BRI	BRANCH AND RESET INTERRUPT	6-181
F980	LPSD	LOAD PROGRAM STATUS DOUBLEWORD	6-182
FA80	LPSDCM	LOAD PROGRAM STATUS DOUBLEWORD AND CHANGE MAP	6-183
FC00	EI	ENABLE INTERRUPT	6-201
FC01	DI	DISABLE INTERRUPT	6-204
FC02	RI	REQUEST INTERRUPT	6-202
FC03	AI	ACTIVATE INTERRUPT	6-203
FC04	DAI	DEACTIVATE INTERRUPT	6-205
FC05	TD	TEST DEVICE	6-216
FC06	CD	COMMAND DEVICE	6-215
FC17	SIO	START I/O	6-217
FC1F	TIO	TEST I/O	6-218
FC27	STPIO	STOP I/O	6-219
FC2F	RSCHNL	RESET CHANNEL	6-220
FC37	HIO	HALT I/O	6-221
FC3F	GRIO	GRAB CONTROLLER	6-222
FC47	RSCTL	RESET CONTROLLER	6-223
FC4F	ECWCS	ENABLE CHANNEL WCS LOAD	6-224
FC5F	WCWCS	WRITE CHANNEL WCS	6-225
FC67	ECI	ENABLE CHANNEL INTERRUPT	6-207
FC6F	DCI	DISABLE CHANNEL INTERRUPT	6-208
FC77	ACI	ACTIVATE CHANNEL INTERRUPT	6-206
FC7F	DACI	DEACTIVATE CHANNEL INTERRUPT	6-209

Reader's Comment Form

Date _____

Manual Title: _____

Publication Number _____

- **How do you use this publication?**

  ☐ As an introduction to the subject.
  ☐ Emergency Maintenance.
  ☐ Other _____

- **Is the material:**

	Yes	No
Easy to read?	☐	☐
Well organized?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Accurate?	☐	☐
Suitable for its intended use?	☐	☐

- **Please check the items that describe your position:**

☐ Customer Personnel
☐ SYSTEMS Personnel
☐ Engineer
☐ Sales Representative
☐ Technician
☐ Field Service
☐ Operator
☐ Programmer
☐ Instructor
☐ Trainee
☐ Other _____

- **Please check specific criticism(s), give page number(s), and explain below:**

☐ Clarification on page(s) _____
☐ Addition on page(s) _____
☐ Deletion on page(s) _____
☐ Error on page(s) _____

Explanation:

Your Name: _____

Your Company: _____

402-02 (4/81)
Printed in USA

May we have your comments?

This publication is one of a series of SYSTEMS Engineering Laboratories technical documents written to serve each of a wide variety of users. Your completion of the attached form will aid SYSTEMS in the continued production of complete, easily referenced material in each of these various technical publications.

Thank You

Fold and Staple for Mailing

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 2356 FORT LAUDERDALE, FLORIDA

POSTAGE WILL BE PAID BY ADDRESSEE

SYSTEMS Engineering Laboratories
6901 West Sunrise Boulevard
Fort Lauderdale, Florida 33313

Fold and Staple for Mailing

Proven COMPUTER Performance

SYSTEMS

A Subsidiary of GOULD INC.

6901 WEST SUNRISE BLVD., FT. LAUDERDALE, FLORIDA 33313