UNIVAC® 1107

GENERAL DESCRIPTION

The UNIVAC 1107 THIN-FILM Memory Computer

GENERAL DESCRIPTION

The UNIVAC 1107 THIN-FILM Memory Computer

Contents


UNIVAC
1107

1. UNIVAC 1107 Thin-Film Memory Computer

The UNIVAC® 1107 Thin-Film Memory Computer represents the most significant departure from conventional data-processing systems since the introduction of solid-state circuitry. For the first time, a thin-film memory device is used in a commercially available computing system.

The many controls which must be exercised for high efficiency input-output, concurrent computation, and internal transfers are amalgamated in the heart of the computer. The busy computer crossroads, technically designated as *registers,* have been placed in the logic of this machine so that peripheral equipment can run at full speed with little interference to running programs.

This ultra high-speed heart of the computer has been made possible by another UNIVAC *first* – **Thin-Film Memory.** UNIVAC Thin-Film Memory is made by deposition of magnetic alloys under high vacuum in layers so thin that magnetization can be switched by rotation within time intervals of several nanoseconds. Remington Rand UNIVAC's new technological breakthrough provides multiple accumulators, multiple index registers, and multiple input-output control registers. These multiple registers permit "housekeeping" steps to proceed in parallel with the main program and offer the advantages of multi-address logic where such logic is most efficient (for example, Search instructions).

Basically, the UNIVAC 1107 is an advanced solid-state data-processing system designed and developed to provide reliable solutions to complex problems. This computer system is well suited to off-line, on-line, and real-time problems in commercial, scientific, and military applications. A highly versatile input-output section and a large internal memory, backed by a powerful instruction repertoire, provide the UNIVAC 1107 with unequalled data-processing capabilities.

UNIVAC 1107 FEATURES

Included among the many features of the UNIVAC 1107 Data-Processing System are:

- **A thin-film control memory—the most advanced storage device on the market today — used for arithmetic and index registers, for input-output access control, for other special controls, and for auxiliary storage.**
- **300-nanosecond (0.3-microsecond) access time for thin-film memory, with a complete cycle time of 600 nanoseconds.**
- **A ferrite-core memory for instructions and operands, available in capacities of 16,384 words in one bank; or of 16,384, 32,768, 49,152 or 65,536 words in two separately accessed banks.**
- **2-microsecond effective cycle time for core storage (overlapping of two banks).**
- **36-bit words in both thin-film and core memories.**
- **An instruction word format that provides for indexing, automatic index-register incrementation, partial word transfers, and indirect addressing, along with a current operand reference and specification of an arithmetic register.**
- **An extremely powerful instruction repertoire, including fixed and floating-point, integer, and fractional arithmetic.**
- **16 input channels and 16 output channels, capable of concurrent input-output transmissions up to 250,000 words per second (1,500,000 characters per second), without direct supervision by the main program.**

- **Automatic programming: ALGOL and COBOL compiling programs and a FORTRAN translating program.**
- **An Executive Routine, capable of integrating routines for multiple programs.**
- **Compatibility with existing UNIVAC systems is maintained through Uniservo IIA, 90-column and 80-column punched-card, and paper-tape peripheral units. Versatile off-line communication with peripheral units can be accomplished by including a UNIVAC Solid-State or UNIVAC STEP system as a satellite computer.**

TYPICAL APPLICATIONS

In line with UNIVAC's leadership in developing and manufacturing computing systems of advanced logical design, the UNIVAC 1107 offers the most advanced data-processing capability now available. This general-purpose system can efficiently and economically handle a wide range of technical applications, such as:

- **Tactical data systems**
- **Command and control systems**
- **Digital communication and switching systems**
- **Data reduction and analysis**
- **Logistics**
- **Scientific computation**
- **Traffic control**
- **Reservation systems**
- **Computational analysis**
- **Inventory and scheduling systems**
- **Intelligence systems**
- **Systems simulation**
- **Missile and satellite dynamics**
- **Process control**

MODULARITY

Because the storage capacity and the number of input-output channels activated are optional, the user can select a UNIVAC 1107 System that will meet his *immediate* processing and cost requirements. The system selected can then be expanded at a rate consistent with the quantity and complexity of applications.

Compatibility with a wide range of commercial, scientific, and military peripheral equipment—of both advanced and standard design—complements the basic building-block characteristic. Consequently, the UNIVAC 1107 System can be varied on the basis of size, components, or applications. In any particular application a configuration can be chosen that will provide a well-balanced system with unprecedented growth potential.

Along with modular construction, the UNIVAC 1107's unique input-output section – designed to be adaptable to new peripheral equipment – assures the user of a data-processing system that will keep pace with the computer industry far into the foreseeable future. This section can connect the Central Computer with many different types of peripheral units, including other Central Computers.

PERIPHERAL EQUIPMENT

The list of peripheral equipment compatible with the UNIVAC 1107 Thin-Film Memory Computer includes:

Standard Peripheral Equipment:

Magnetic Drums
Magnetic-Tape Units
Punched-Card Units
High-Speed Printers
Paper-Tape Subsystems
Supervisory Console Auxiliaries

Special Peripheral Equipment:

Analog-to-Digital and Digital-to-Analog Converters
Electronic Printers
Displays, Plotters, and Keysets
Multiplex and Switching Units
Special Real-Time On-Line Systems
Mass-Storage Units
Other Off-Line Systems
Other Computers

OPERATIONS IN BRIEF

UNIVAC 1107 internal operations are performed in the parallel binary mode. Each computer word, in thin-film control memory and core memory, contains 36 bits. Instructions normally include the address of both an operand and an arithmetic register, and may specify indexing, incrementing or decrementing, indirect addressing, and field selection. Direct communication between internal memory and peripheral equipment may be scheduled over 16 sets of bidirectional communication paths consisting of 16 input channels and 16 output channels.

Memory

Regardless of the core-memory capacity selected by a user (capacities range from 16,384 to 65,536 words), every UNIVAC 1107 System employs a separate thin-film control memory. This memory, which is the latest development in storage techniques, consists of an array of thin magnetic films. The time required to obtain information from the UNIVAC 1107's thin-film memory is only 300 nanoseconds (0.3 microsecond). Very high operating speeds can be achieved because the thin-film control memory allows parallelism and sophisticated logic. Each instruction *does more work.*

In addition to providing auxiliary storage locations, the control memory furnishes:

15 Index Registers
16 Arithmetic Registers*
36 Special Control Registers

Instructions

The instruction repertoire encompasses both fixed and floating-point arithmetic. Fixed-point instructions, in turn, provide for integer and fractional arithmetic. Provision has also been made for partial word transfers, partial compares, repeated search operations, and masking. Special add and subtract instructions perform parallel addition or subtraction of two or three fields within a single data word. To provide fast programming of double-precision arithmetic, special features have been incorporated in the arithmetic section.

Four arithmetic-register addresses overlap index-register locations.

Input-Output Channels

The UNIVAC 1107's input-output channels have been paired to meet the requirement that standard peripheral equipment accommodate bidirectional data transfers. Up to 16 input channels and 16 output channels can be used for direct communication between peripheral equipment and internal memory.

Program Interrupt

Seventy-four interrupt signals, governing input-output operations and various contingency and error conditions, include internal and external interrupts for every channel. In effect, each interrupt causes a jump from the main program to an associated subroutine. This subroutine may set up input or output transmissions, prepare the computer for error diagnostic routines, or perform any other function the programmer may assign to it.

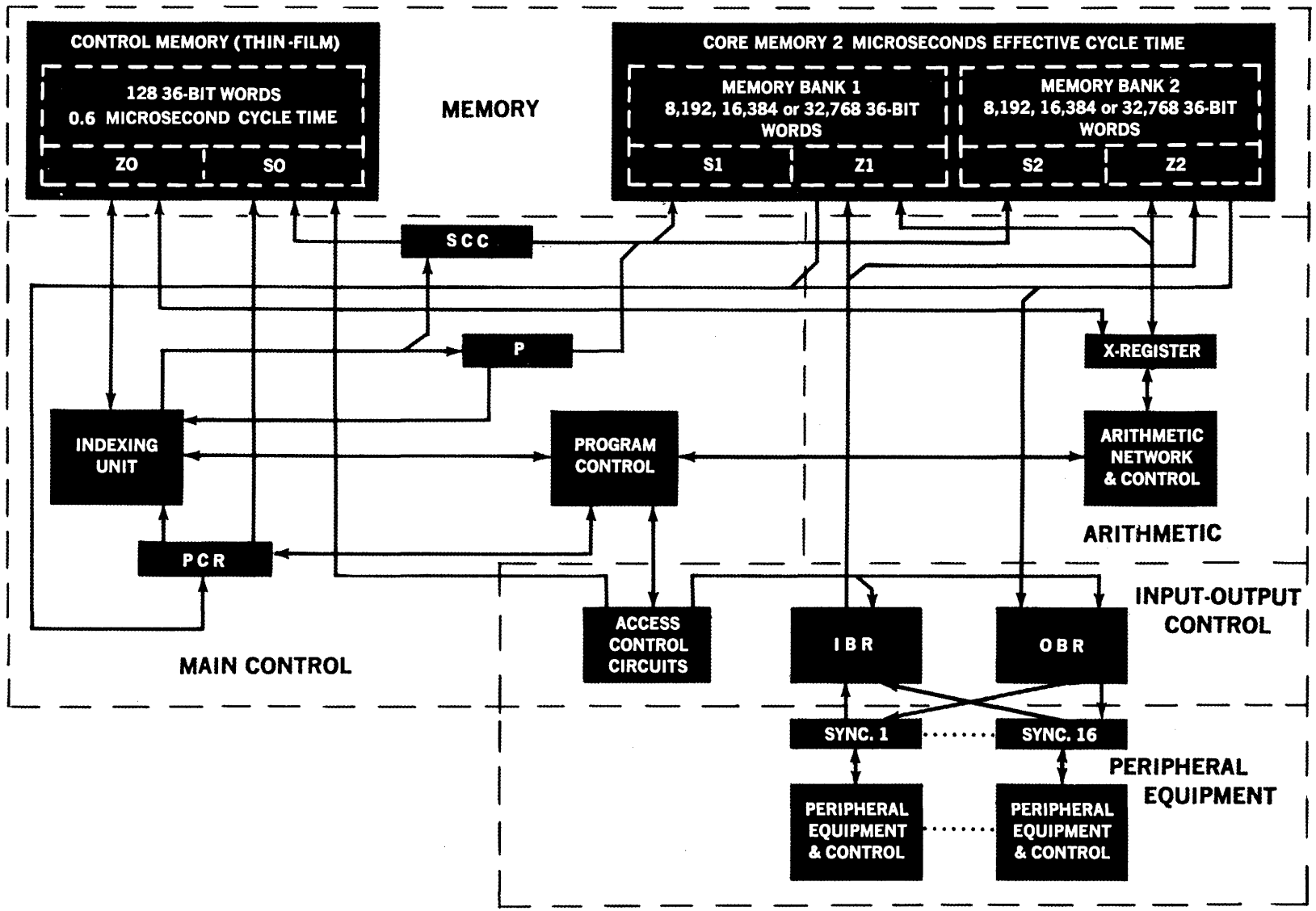


Figure 2-1. Block Diagram of the UNIVAC 1107 Thin-Film Memory Computer

2. Central Computer—Storage

The Central Computer in the UNIVAC 1107 System comprises four major sections: storage, control, arithmetic, and input-output. A block diagram of the Central Computer is presented in Figure 2–1.

The storage section consists of control memory and core memory, along with their associated address, transfer, and control circuits. Memory is addressed via storage class control *(SCC)* from the indexing unit, program control register *(PCR)*, and control registers located in the control memory. Each address is decoded by *SCC* to determine whether reference is to be made to control memory or to core memory bank one or two. The address is then transmitted to the appropriate storage address register *(S0, S1,* or *S2)*, and the corresponding memory reference is initiated.

CONTROL MEMORY

The 128-word control memory consists of deposited magnetic films. The film array has word selection and operates in the parallel mode. Read access for any address is 300 nanoseconds; complete cycle time is 600 nanoseconds.

CORE MEMORY

Core memory consists of modular arrays of ferrite cores with coincident current selection. Capacity options are 16,384 words in one bank, or 16,384, 32,768, 49,152 or 65,536 words in two banks. Read access time for any address is 1.8 microseconds; complete cycle time is 4.0 microseconds. The two banks function as separate units which are overlapped to provide an effective cycle time of 2 microseconds.

Writing into specified regions of core memory may be suppressed by a special memory write lockout instruction (Function Code 72-11). This core-memory lockout can protect any 8 memory regions (which could include the entire core memory) from undesired writing operations. The single address 201 (311 octal) containing the external status word cannot be locked out, since it must always be available for the external interrupts.

STORAGE ALLOCATION

The first 202 of the 65,536 addresses are used as shown in Table 2–1. Core memory addresses have the lower range in bank one and the higher range in bank two.

The addresses 00000-00127 (000000-000177 octal) are the 128 words of control memory when specified by the special designators in the instruction word or by the execution address. The addresses 00000-00127 designate the first 128 words of the first core-bank when these addresses are specified by the *P*-register for an instruction reference and by the *V*-address of the input-output access control word for input and output transfers (Figure 2–2).

DECIMAL ADDRESS	OCTAL ADDRESS	FUNCTION		
00000	000000	Unassigned		CONTROL MEMORY
00001-00015	000001-000017	Index Registers (15)		
00012-00027	000014-000033	Arithmetic Registers (16)*		
00028-00031	000034-000037	Unassigned		
00032-00047	000040-000057	Input Access-Control Words (16)		
00048-00063	000060-000077	Output Access-Control Words (16)		
00064	000100	Real-Time Clock	R Registers	
00065	000101	Repeat Counter		
00066	000102	M Register		
00067	000103	T-Register (temporary storage for P)		
00068-00079	000104-000117	Additional Special Registers		
00080-00127	000120-000177	Unassigned		
00128-00143	000200-000217	External Request Interrupts (16)		CORE MEMORY
00144-00159	000220-000237	Input Data Termination Interrupts (16)		
00160-00175	000240-000257	Output Data Termination Interrupts (16)		
00176-00191	000260-000277	Function Termination Interrupts (16)		
00192-00199	000300-000307	Error Interrupts (8)		
00200	000310	Real-Time Clock Interrupt		
00201	000311	External Status Word		
00202	000312	External Synchronization Interrupt		
00203-65535	000313-177777	Unassigned Core-Memory Addresses		

*Overlap 4 index registers.

Table 2-1. Address Assignments

NAME	INPUT-OUTPUT ACCESS CONTROL WORD		
FUNCTION	G	W	V
BITS	35 34	33 18	17 00

G (2 bits) 00—Increment V
01—Inhibit Increment V
10—Decrement V
11—Inhibit Decrement V
W (16 bits) —Number of words remaining to be transferred
V (18 bits) —Address of current data transfer

Figure 2-2. Format of the Input-Output Access Control Word

3. Central Computer—Control

The control memory provides special storage assignments for index registers, arithmetic registers, input-output access control, a mask register, the real-time clock, a repeat count, temporary program address storage, and auxiliary storage.

CONTROL-MEMORY REGISTERS

The *B*-registers, the *A*-registers, and the *R*-registers have what is termed "two-address accessibility." That is, they may be referenced directly either by the operand-address portion of an instruction or by a special designator contained in certain instructions. This feature is further explained in the description of the instruction word format.

Index Registers

Fifteen index registers (or *B*-registers) are used for operand address modification, index counts, and modifier incrementation. The right half of the index word contains the 18-bit address modifier. The left half of the index word contains the 18-bit increment (decrement). The modifier is added to the base execution address of the instruction to obtain the effective address. A control bit in each instruction then specifies whether the increment is to be added to the modifier. Special index register instructions test the modifier. The indexing unit is an 18-bit arithmetic unit which can perform addition and subtraction in one's complement arithmetic. This unit handles both address modification and incrementation (decrementing) of the modifier in parallel with the main arithmetic sequence. The index word format is shown in Figure 3-1.

Arithmetic Registers

Sixteen arithmetic registers in the control memory provide interim high-speed storage for arithmetic operations. From a programming viewpoint, these arithmetic registers (*A*-registers) function as 16 accumulators. The actual accumulation is performed in the arithmetic section with the result stored in the *A*-register(s) specified by the instruction. Because of the overlap of addresses (Table 2–1), four *A*-registers can be directly designated as index registers.

- **With four arithmetic registers capable of functioning as index registers, the UNIVAC 1107 System can perform highly sophisticated address modification.**

Input-Output Access Control Registers

Thirty-two control-memory addresses are used for storing input and output access

NAME	INDEX REGISTER WORD	
FUNCTION	△	Q
BITS	35 18	17 00

△ (18 bits)—Increment or Decrement
Q (18 bits)—Modifier

Figure 3-1. Format of the Index Word

control words. The function of these words will be discussed in the input-output section.

R-Registers

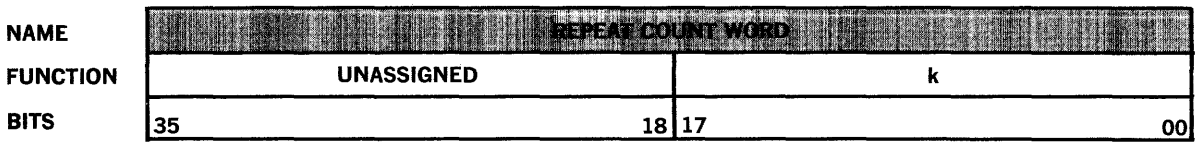
Sixteen control-memory addresses are designated as "*R*-registers." Four of the 16 *R*-registers are assigned specific functions as explained below; the remaining 12 may be used in any way the programmer desires (except for instruction storage).

Of the four specially assigned *R*-registers, one is used for the real-time clock. The 36-bit number contained in this register is decremented by one every millisecond (the exact granularity being 2^{-10} seconds). When the count is reduced to zero, an internal interrupt occurs which causes the program to jump to address 200 (octal 310). The direct two-address accessibility of the clock simplifies presetting the count, resetting it during operation, and subsequently reading it out.

The second of the four assigned *R*-registers is used for the repeat count, containing the number of times a repeated instruction is to be executed (k). The format of the repeat count word is shown in Figure 3–2. Each time a repeated instruction is executed the k is reduced by one. The repeat function terminates when k is reduced to zero. The unassigned left half of the repeat count word may be used to "park" the total number of times an instruction is to be repeated.

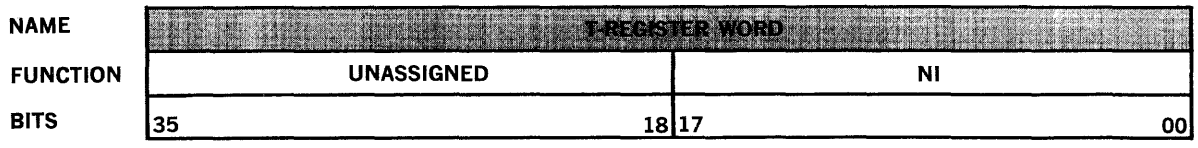
The third assigned *R*-register is the *M*- or masking register. The *M*-register contains the mask used in certain logical and test instructions (Function Codes 43 and 71).

The fourth assigned *R*-register is the *T*-register. This register is used as a temporary parking register to hold the address of the next instruction, *NI*, during the execution of a repeated instruction. During the execution of nonrepeated instructions, this address is not referenced by main control. The format of the *T*-register word is shown in Figure 3–3.



k (18 bits) —Number of times instruction is to be executed

Figure 3-2. Format of the Repeat Count Word



NI (18 bits) —Address of the next instruction

Figure 3-3. Format of the T-Register Word

Control Circuits

The main control and timing circuits supply control signals which synchronize the execution of the instructions. The particular instruction to be executed is determined by the contents of the program-control register.

INDEXING UNIT

The indexing unit, containing an adder and sensing circuits, is shared by program control and input-output control. Program control uses the indexing unit to advance the *P*-register, to "count down" *k* in the repeat count register, to control repeated sequences, and to perform the indexing operations. The indexing operations include the addition of the modifier to the base address to obtain the effective execution address, the addition of the increment to the modifier, and various tests of the modifier. Input-output control uses the indexing unit to increment (decrement) the data-transfer address and to decrement the word count.

INTERRUPTS

An interrupt is a special control signal which diverts the "attention" of the computer to "consider" an extraordinary event or set of circumstances; that is, it causes program control to be transferred to a special subroutine which corresponds to the "stimulus." Many levels of control can be exercised by the numerous forms of interrupts provided. The interrupts from external sources serve primarily to synchronize the computer program with the readiness of peripheral devices, including other computers, to transmit or receive data. Internal interrupts serve primarily to synchronize the computer program with the termination of input-output transfers and to signal the occurrence of an error.

An interrupt causes the next instruction to be procured from a fixed address corresponding to the interrupt source. This fixed address serves as a subroutine entrance by containing a *return jump* instruction. The *return jump* instruction transfers the contents of *P*, which is the address of the instruction which otherwise would have been executed, to the first address of the subroutine, thereby providing the subroutine exit. Program control is then transferred to the second address of the subroutine.

Several classes of interrupts are provided. An external-request interrupt with a fixed address for subroutine entrance corresponds to each of the 16 input channels. The 16 external-request interrupts enable external devices connected to the input channels to demand the attention of the computer as required. An internal interrupt with a fixed address for subroutine entrance corresponds to each of the 16 input access control words, 16 output access control words, and 16 external function words. An internal interrupt with a fixed address for subroutine entrance is provided for the real-time clock.

An additional external interrupt with a fixed address for subroutine entrance is provided for real-time system synchronization. This interrupt is independent of the input-output channels. The synchronization interrupt accepts signals from an external generator of any desired frequency. The external generator may be a supplementary real-time clock for the computer or the master clock of a multi-computer complex. The fixed address assignments for the various classes of interrupts are shown in Table 2–1.

The occurrence of an interrupt causes a lockout to prohibit the occurrence of further external interrupts until the instruction, *enable all external interrupts,* is executed. The 16 external input interrupts may also be disabled under program control (Function Code 75).

ERROR DETECTION

Certain errors are detected by the computer and cause internal-error interrupts without stopping computer operation. An internal interrupt with a corresponding fixed ad-

dress is provided for each of eight types of internally detected errors. These interrupts cannot be disabled or locked out. Upon detection of an error, control is transferred to an appropriate error address (subroutine entrance) in a manner similar to the other interrupts. Action is taken by the subroutine and control is then returned to the main program.

The error detection includes illegal function code and attempted write into a locked-out area of memory; these are indicated to the computer operator as program faults. Detection of arithmetic errors includes floating-point characteristic overflow and underflow and stated-point-divide overflow. Parity checks are also made on peripheral equipment.

INSTRUCTION FORMAT

The operation of the computer is controlled by a program of instructions stored in memory. Each instruction is read from memory, normally from sequential addresses, and then is transmitted to the program control register for interpretation. Each instruction consists of several parts called designators. These are identified by letters and are listed below as they appear from left to right in the instruction word. (See Figure 3–4.)

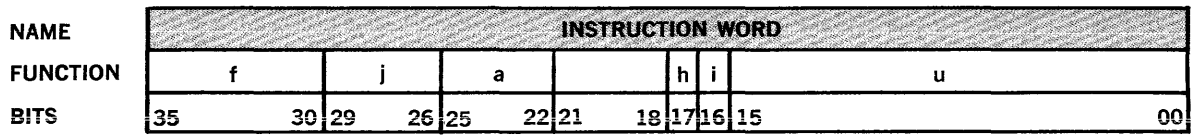
The execution of an arithmetic instruction includes the following steps:

1. Read out (B_b) from film memory specified by the *b* designator of the current instruction.
2. Store the result of the previous instruction at A_a of thin-film memory specified by *a* of the previous instruction.
3. Add $(B)_Q$ to u where b $\neq$ 0 for current instruction.
4. Store the second part of the result of the previous instruction at $A_a + 1$ in film memory at the address one higher than that used in Step 2, if applicable.
5. Initiate access to the operand with the effective address obtained from Step 3. The storage reference may be made to film memory or core memory.
6. Initiate access to NI. This storage reference is made to core memory.
7. Read out (A_a) from film memory specified by *a* of the current instruction.
8. Add $(B)_\Delta$ to $(B)_Q$ if specified by *h* of the current instruction.
9. Store index register word at the film memory address used in Step 1.
10. Initiate arithmetic operation.
11. Initiate input-output word transfers if requested and then proceed to Step 1.

The above steps take 4.0 microseconds for most instructions when the operand references are made either to film memory or to the alternate core memory bank. Instruction execution time is extended by 4.0 microseconds when the operand reference is made to the same bank as the instruction reference. Most arithmetic operations are performed in less than two microseconds. Certain arithmetic operations require more than two microseconds and thereby extend the instruction times beyond 4.0 microseconds. Each input-output word transfer increases the instruction time by 4.0 microseconds, with the exception of extended arithmetic instructions, where input-output word transfers can occur simultaneously with the arithmetic sequences. For example, two input-output word transfers and the *Floating Multiply* instruction take 12.7 microseconds total. Three input-output word transfers with this instruction take 16.0 microseconds total.

When the A-register used in Step 2 above is the sixteenth A-register, the second part of the result is stored at the next address 00028 (00034 octal) in Step 4.

When core memory references (Steps 5, 6, and 11) are made to addresses outside the core memory modules installed, the reference is made to an address with fewer significant bits in the specified memory bank.



f (6 bits)—Function Code
j (4 bits)—Operand Interpretation or Minor Function Code*
a (4 bits)—A-Register Designator, I/O Channel Number, B-Register or R Register Designator*
b (4 bits)—B-Register Designator
h (1 bit)—Incrementation Designator
i (1 bit)—Indirect Addressing Designator
u (16 bits)—Base Operand Address
*Instruction determines usage.

Figure 3-4. Format of the Instruction Word

The program control overlaps as much of the instruction sequencing as practical. For example, in an arithmetic instruction which augments an A-register, the initial content of the specified A-register is transferred to the arithmetic unit concurrently with an operand reference to core memory. Similarly, the steps for procurement of the next instruction are undertaken before the sequencing of the current instruction is completed.

Function Code, *f* (6 Bits)

The left-hand 6 bits of the instruction word designate the function to be performed by that instruction. In some instructions, where the normal meaning of j is not applicable, j is combined with f as a 10-bit function code. Values of f (or f and j) which are not meaningful are fault conditions which cause an error interrupt, and the program will jump to a fixed memory address which is the entrance to an error routine.

Operand Interpretation, *j* (4 Bits)

The j designator normally defines that part of the operand that is to be transferred to and from the arithmetic unit. These values of j are given in Figures 3-5 and 3-6 .

A-Register Designator, *a* (4 Bits)

During arithmetic instructions, this designator specifies any one of the 16 A-registers. These fast registers are also directly addressable by the operand address. In some instructions such as *block transfer*, load B_a, and *test modifier*, the a-designator specifies the index registers (using the notation B_a). For these instructions a specifies one of 16 index registers (for example, a = 0 specifies address 00000). The input-output control instructions use a to specify the channel to be used. In other instructions, a designates an R-register. In one instruction, *index jump*, a and j taken together designate any one of the 128 words of control memory.

B-Register Designator, *b* (4 Bits)

The b-designator of the instruction specifies which of the 15 index registers, if any, is to take part in the modification of the operand address designator, u. No address modification occurs for $b = 0$.

Incrementation Designator, *h* (1 Bit)

This designator specifies incrementation of the modifier of the index register designated by b as follows:

$h = 0$, No incrementation
$h = 1$, $(B)_Q + (B)_\Delta \rightarrow (B)_Q$

Indirect Addressing Designator, *i* (1 Bit)

This designator specifies either normal or indirect addressing of the operand. Indirect addressing denotes that the *effective address* of the operand is contained in the memory location specified by $U = u + (B)_Q$;

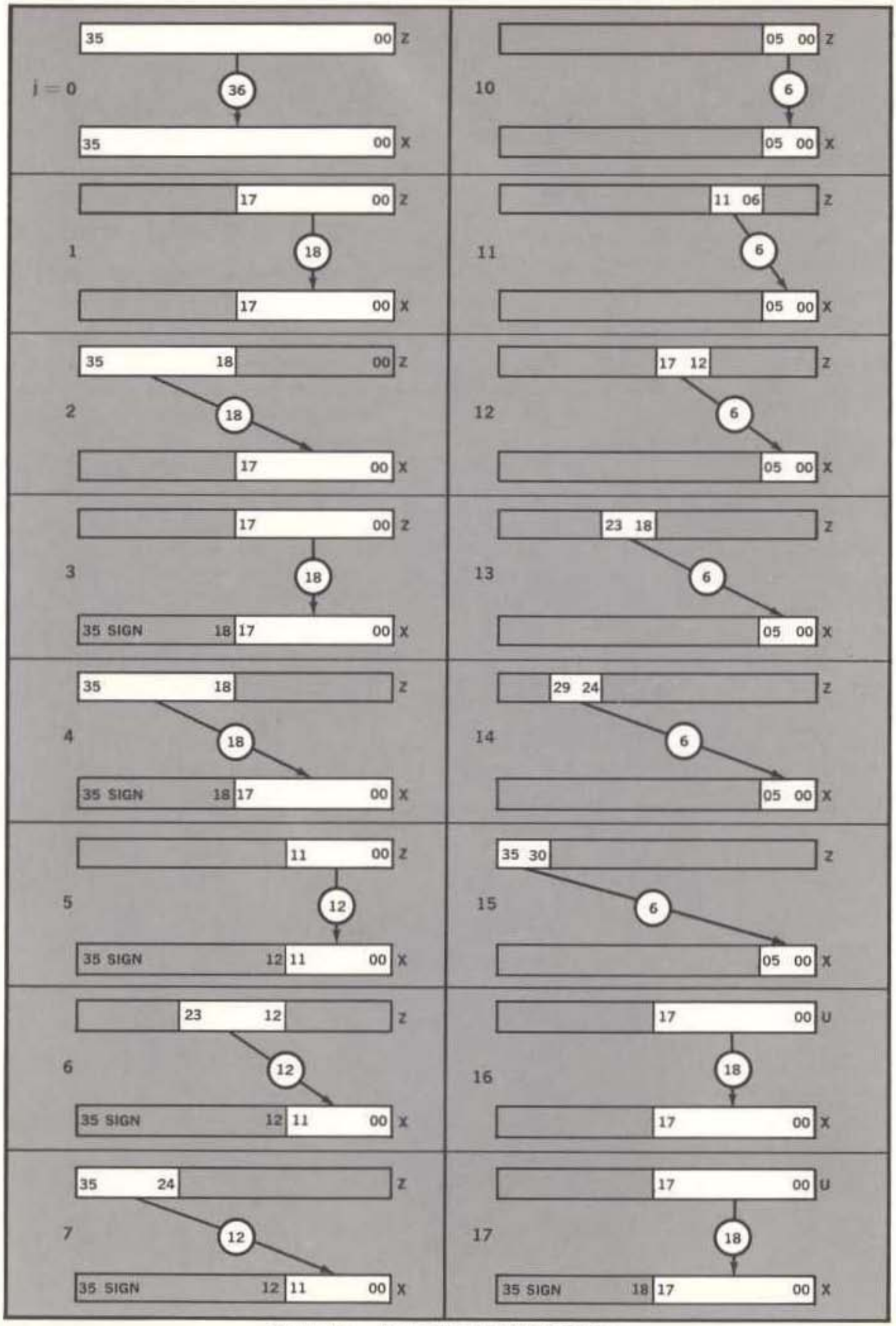


Figure 3-5. Data Paths to Arithmetic Section

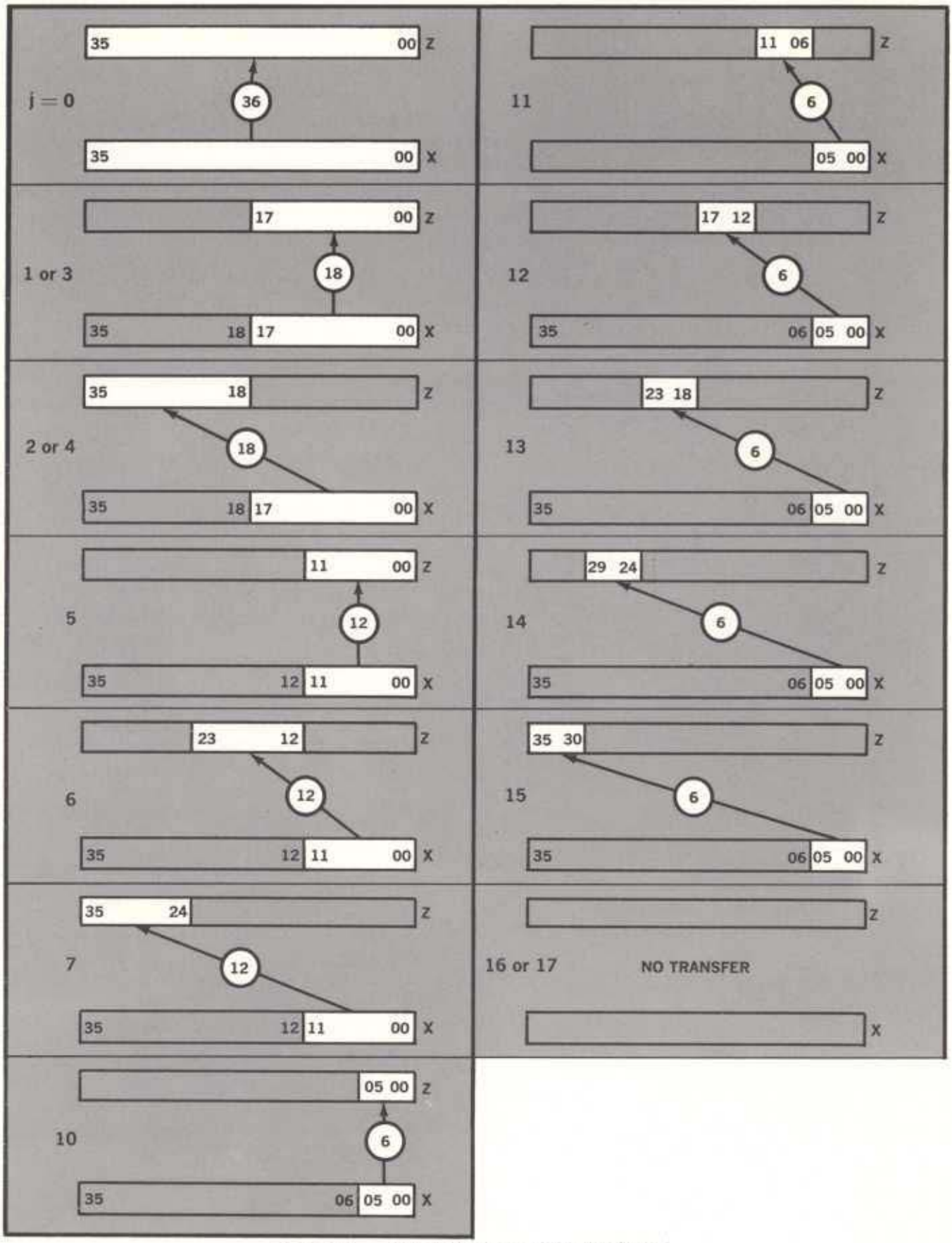


Figure 3-6. Data Paths *from* Arithmetic Section

that is, U is the address of an address. If the indirect address indicator thus obtained from the right 22 bits of the word at the U address indicates indirect addressing is to be used, the process is repeated; that is, the indirect addressing is cascaded. The specification of indirect addressing is as follows:

$i = 0$, Direct addressing applies. The *effective address* is $U = u + (B)_Q$.

$i = 1$, Indirect addressing applies. The *effective address* is provided by the right 22 bits of the memory location given by $U = u + (B)_Q$. Since the b, h, i, and u portions of the program control register are replaced, the indexing, incrementing, and indirect addressing can be cascaded.

Base Operand Address, *u* (16 Bits)

The operand address, made up of the base address plus any specified modification, designates which of the possible 65,536 storage locations will be referred to in the execution of the instruction. Most instructions refer to a storage location once during the execution of an instruction. When j equals 16 or 17 (octal), the effective operand is taken directly from the instruction word. The instruction does not reference the address of an operand but instead contains the operand within itself. U also provides a shift count.

REPEATED SEQUENCES

Thirteen instructions operate in a repeated sequence mode: *block transfer* and *repeated search*. The number of times the function is to be executed is specified by k, which is retained in the repeat count register.

As part of the preparation for the execution of a repeated sequence, the repeat count register is loaded with the *load* R_a instruction. During the repeated sequence mode, the instruction is retained in *PCR* and P is not consulted for the next instruction. Each time the function is executed, k is counted down by one and tested for zero. If k does not equal zero, the function is repeated. When k equals 0, the normal sequence of procuring the next instruction from P is resumed.

An interrupt which occurs during a repeated sequence temporarily halts the repeated sequence mode and causes a jump to the fixed address corresponding to the interrupt. This termination is performed in a manner that allows the return from the interrupt subroutine to reinitiate the repeated sequence from the point of interruption. When an interrupt temporarily halts a repeated sequence, P is set back to the address of the instruction which set up the repeated sequence mode.

In the termination of a *search* instruction by a jump or skip condition, P is treated in the normal manner.

The contents of the applicable control registers prior, during, and subsequent to execution of a repeated instruction are summarized as follows:

Prior to repeated sequence
P-register: P.
Repeat count register: k, number of times instruction is to be executed.
T-register: Not applicable.

During repeated sequence
P-register: k, number of times instruction is to be executed.
Repeat count register: k = initial k.
T-register: Temporary storage of P.

After repeat termination
P-register: P.
Repeat count register: k = number of times remaining in sequence (for normal termination, $k = 0$; for jump or interrupt termination, $k \geq 0$).*
T-register: Not applicable.

SUPERVISORY CONSOLE

The Supervisory Console includes the operator's control panel, a keyboard and type-printer, and a control unit for the keyboard and type-printer. Optionally, a Paper-Tape

**If the initial value of k were also parked in the unused half of the repeat count word, this information would be retained unchanged.*

Reader and Punch may be connected to the computer through the same control unit. Information transfer between the computer and any single device is performed on a character basis over the input channel and output channel assigned to the console auxiliaries. Two switches mounted on the control unit permit selection of the Paper-Tape Reader or the keyboard and the Paper-Tape Punch or the type-printer.

Operator's Control Panel

This panel provides direct operator communication with the computer. The manual controls and indicators provided allow the operator to:

1. Stop computer program execution but allow the input-output section to continue operation.
2. Clear all computer registers except those in the input-output section.
3. Master clear all computer registers including those in the input-output section.
4. Start computer program execution. The program will start with the instruction located at the address indicated by the six octal digit indicators of the *P*-register.
5. Set the desired address into the *P*-register for starting program execution.
6. Set any of the fifteen (15) *selective jump* switches. An indicator is lit when the corresponding jump is selected. These jumps can be selected while the computer is running.
7. Set any of the four (4) *selective stop* switches. When the selection is made, the upper half of the corresponding selection indicator is lit, and when the stop is made the corresponding stop indicator is lit. These stops can be selected while the computer is running.
8. Read in an initial loading (bootstrap) program from input channels.
9. Select one of the sixteen (16) channels for the initial load of the bootstrap program.

In addition to the above manual controls and indicators, four (4) fault or status indicators are provided. These are:

1. *Computer Status*—for example, excessive temperature, poor voltage regulation.
2. *Program Faults*—for example, illegal operation code, illegal memory access (memory lockout).
3. *Peripheral Equipment Fault*—for example, loss of power in a channel synchronizer or control unit, disconnected cable.
4. *Initial Loading (bootstrap) Fault*—error occurring during the loading of the Bootstrap program.

Keyboard

The keyboard is an assembly consisting of a standard four-bank typewriter keyboard. The keyboard can generate 64 basic characters, among which are the 51 characters of COBOL (COmmon Business Oriented Language).

Type-Printer

The type-printer is a 10-character-per-second printer. It is capable of printing the 26 alphabetic characters, 10 numeric characters, and 19 special printable characters of the basic 64-character code. It will also respond to 9 control codes (space, carriage return, and so forth). All 51 of the COBOL Characters are printed.

INITIAL LOADING

An automatic bootstrap operation has been provided as a means for initial loading of programs and for program restoration. This operation facilitates the reading of 160 words from peripheral equipments such as magnetic drums, magnetic tapes, punched cards, and paper tape. Upon request for the bootstrap operation by the operator, an external function word is sent to the peripheral equipment. An input

transfer is established with a monitor for 160 words. The peripheral equipment terminates any current operation and starts transmitting data from address zero, if a drum; or block zero, if magnetic tape; from the first available punched card; or from the first frame, if paper tape.

When all 160 words have been entered into the computer, the main program is transferred to the program contained within these 160 words via the input-data-termination interrupt.

Should an error such as parity occur during the transfer of the bootstrap operation, an error indicator is lit on the operator's console. The operator can reinstate the bootstrap operation again on the same peripheral equipment over the same input channel, or he may select another peripheral equipment and input channel.

4. Central Computer-Arithmetic

The arithmetic section includes an adder, temporary storage registers, a sequence counter, shift matrix, threshold-sensing circuits, and arithmetic-sequence control circuits. The adder is a 36-bit one's complement subtractive adder (mod 2^{36}-1). The counter is employed during multiply and divide operations. The threshold-sensing circuits determine equality and relative magnitude of the contents of registers specified by the instructions. The arithmetic-sequence control circuits govern the execution of the algorithms for addition, subtraction, multiplication, division, shifting, and testing relative magnitudes.

As may be seen from the instruction repertoire, a complete range of fixed- and floating-point arithmetic operations is provided. Addition and subtraction of two or three fields (vector components) within the data word can be performed in the same time as an ordinary add instruction.

Three very important features make the arithmetic section unique.

1. Word transmissions between the arithmetic section and core memory can be directly segmented into halves, thirds, and sixths.
2. Operands and results are retained in 16 directly addressable arithmetic registers (A-registers) in the thin-film control memory.
3. Results of arithmetic operations are retained in thin-film memory in double-precision form. This feature, through the use of thin-film accumulators, facilitates programming double-precision arithmetic operations.

PARTIAL TRANSFERS

A four-bit portion of the instruction designated j, specifies the mode of transmission to or from the arithmetic section. Figure 3-5 shows the data paths *to* the arithmetic section from the storage transfer register (Z) of core memory. When j is 16 or 17 (octal), the effective operand is taken directly from the instruction word. (That is, the instruction does not reference the address of an operand but instead contains the operand within itself.) All partial words enter the lower-order positions of the X-register and, with the exception of $j = 1$, 2, or 10–16 (octal), the sign is extended to the higher-order positions.

- **The automatic shifting of partial words to lower-order positions in the X-register is extremely valuable both in terms of programming and processing. With this feature, computation can frequently be performed immediately after the partial words have been transferred, without first calling for "housekeeping," such as shift instructions.**

Figure 3-6 shows the data paths for data transfers *from* the arithmetic section to core memory as specified by j. In this case the rest of the core word is not changed.

In instructions where the partial word designation has no significance (f = 70, 71, 72, 73, 74, 75, 76, octal), the j code is treated as part of the regular function code to specify various instructions.

ARITHMETIC REGISTERS

A four-bit portion of the instruction, designated a, specifies one of the 16 high-speed

A-registers in control memory. Specification of an *A*-register as well as a regular execution address provides much of the flexibility of multiple address instructions.

Arithmetic operations are performed in the arithmetic unit utilizing temporary storage registers. These registers retain no initial or final results from one instruction to another. All such results are retained in the *A*-registers specified by the instructions. The *A*-registers function as 16 accumulators from the programming point of view. The fixed- and floating-point word formats are shown in Figure 4-1.

DOUBLE PRECISION ARITHMETIC

Most arithmetic instructions preserve two word intermediate results. In the case of stated-point multiplication, a double-length product is stored in the *A*-registers of control memory for integer and fractional operations. Integer and fractional division are performed upon a double-length dividend with the remainder preserved for use as the dividend in the next instruction if desired. An overflow indication from additions is retained for programmed tests as desired.

The floating-point instructions for addition, subtraction, and multiplication always store a two-word result. Data words need not be normalized. Floating-point division requires that the divisor have at least as many significant bits as the dividend. Floating-point addition provides the most significant word normalized and stored in an *A*-register and the least significant word without normalization stored at the next higher addressed *A*-register. Both results contain their appropriate characteristics. A full double-precision addition can be obtained from four single-precision additions.

The floating-point product of two normalized numbers produces either 53 or 54 bits of product. Wherever 53 occurs a *left shift* of one results. Therefore, the product of normalized values yields normalized results, while the product of unnormalized values yields unnormalized results. In either case, the characteristics of the two word results always differ by 27. Three floating-point multiplications and additions will yield a double-precision result except for the extreme right portion in the register containing the least significant half.

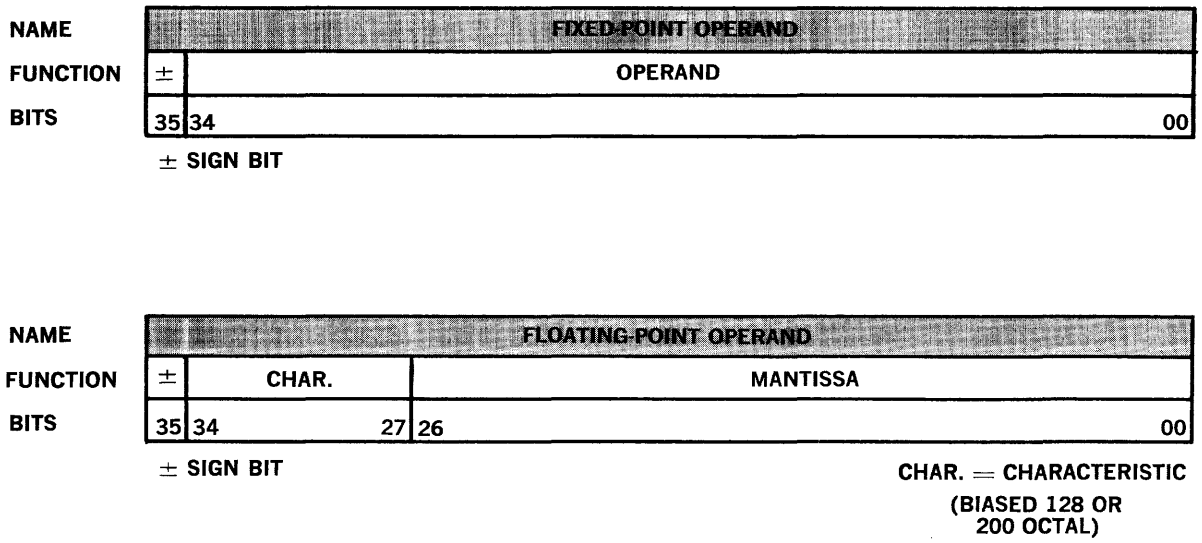


Figure 4-1. Format of Fixed-Point and Floating-Point Words

5. Central Computer—Input-Output

GENERAL CHARACTERISTICS

The input-output section provides the data paths and control circuits necessary for direct communication between the core memory banks and the peripheral equipment. A communication path between computer and peripheral unit is initially established by the main computer program. Thereafter, the individual transfers are governed by the input-output access-control circuits, which monitor the number of words to be transferred and specify the addresses in core memory to and from which the data are transmitted. As previously mentioned, the access-control circuits service the requests of each external unit until the specified sequence of word transfers has been completed. Thus these circuits free the remaining sections of the computer enabling them to continue with the execution of the main program.

Working in conjunction with the access control circuits, priority control circuits resolve situations where two or more external equipments *simultaneously* attempt to communicate with the computer. In each case, priority is given to the one with the lowest channel number.

Up to 16 output and 16 input channels can be in concurrent operation. However, the number that can be handled efficiently is a function of the data transfer rate of the peripheral equipment. (In general, faster transfer rates tend to reduce the number of channels that can be handled efficiently.) In most cases, both an input and an output channel are used with the same peripheral equipment to provide bidirectional data transmission. The capability of the input-output section to provide separate and distinct control over input-output channels accommodates specialized peripheral devices in real-time applications. Concurrent word transfers are multiplexed to provide a maximum communication rate of 250,000 words per second (1,500,000 characters per second). Conventional off-line operations, such as card-to-tape conversion, can be performed as on-line operations with negligible interruption of other programs. In this type of operation, the information flows to and from an assigned memory area, serving as a transfer buffer.

FUNCTIONAL OPERATION

The computer establishes communication with the peripheral equipments by sending control codes (external function words) over the output data lines. These external function words are distinguished from data words because they are accompanied by a signal over the external function line, a special control cable between the computer and the peripheral equipment. Through the use of external function words, the computer, in normal operation under program control, can activate or deactivate peripheral units.

The peripheral equipments can, if desired, generate interrupt requests. Such an interrupt request causes an unconditional jump in the main program to an address in core memory associated with the external interrupt for that particular communication channel. This address, in turn, contains an instruction which transfers control to a program subroutine.

The normal control signals used to transfer words to and from the computer are shown in Table 5-1. The lines linking the computer to the peripheral equipment are illustrated in Figure 5-1.

NAME	ORIGIN	MEANING	EFFECT
Output Request	Output Unit	Output unit is ready to receive next output word.	Computer forms next output word and sends output acknowledge.
Output Acknowledge	Computer	Computer has transmitted next output word.	Initiates cycle whereby output unit accepts and processes output word.
Input Request	Input Unit	Input unit is ready to transmit next input word.	Computer accepts input word and sends input acknowledge.
Input Acknowledge	Computer	Computer has received last input word.	Initiates cycle whereby input unit produces next input word.
Interrupt	Peripheral Unit	Peripheral unit requires output from (or input to) computer.	Causes program to jump to interrupt subroutine for that channel.
External Function	Computer	Word on data lines is an external function word.	Peripheral control unit decodes function word.

NOTE: The terms "output" and "input" are referenced with respect to the computer.

Table 5-1. Input-Output Control Signals

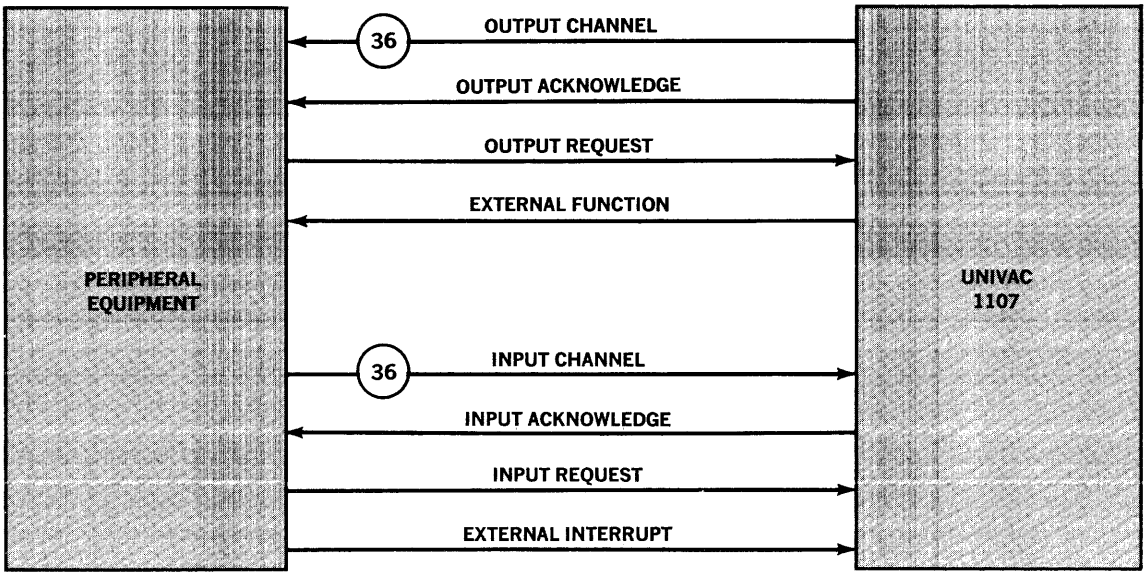


Figure 5-1. Typical Interconnection Between Peripheral Equipment and the UNIVAC 1107 Computer

Information is transferred to or from the computer's core memory in blocks of data. A block of data is defined as a series of words having consecutive memory addresses starting with a program-determined first word and ending with a program-determined last word. Block length is limited only by the size of the memory and the addresses of the first and last words of the block.

As indicated in the repertoire of instructions, a block transfer can be established in one of two ways: with a monitor or without a monitor. "With a monitor" means that an internal interrupt will be generated at the conclusion of the transfer; "without a monitor" means that this interrupt will be inhibited.

The word-by-word transfer of data to or from the computer is governed by an input-output "access-control word." One such word is assigned to each channel. Input and output access-control words have the same format. (See Figure 2-2.)

The output-access-control word serves two purposes: it controls the transfer of output data words, and it controls the transfer of the external function words previously mentioned. The input-access-control word governs only the transfer of input data words.

As shown in Figure 2-2, the access-control word consists of three parts: G is the "transfer mode designator," the use of which is explained below; V is the starting address of the data transfer; and, W is the number of words to be transferred.

The computer program initiates an input-output transfer in the following manner:

1. The program compiles a list of external function words necessary to define the desired operation to the peripheral equipment. This list may be one or more words long.
2. An **initiate monitored function mode** instruction is executed, which begins the transmission of the list to the peripheral equipment. The rate at which the external function words are transmitted is governed by the output request signal rate of the peripheral equipment. A tape control unit, for example, may receive instructions to initiate read, initiate write, search tape, or rewind the specified tape unit.
3. After the last external function word is transmitted, an internal interrupt (resulting from the monitor) occurs which notifies the program that data transmission may begin. Normally, the main program will then jump to a subroutine which initiates data transfer.
4. As part of the subroutine which initiates data transmission, there occurs an **initiate output** (or **input**) **mode** instruction which transmits an output (or input) access control word to the location in control memory corresponding to the particular channel. The **initiate output** (or **input**) **mode** instruction may or may not specify a monitor. Once the instruction is given, data transfers occur at the natural rate of the peripheral equipment.

The first data transfer references address V, as specified in the access-control word. The second transfer references address V ($G = 01$ or 11), address $V + 1$ ($G = 00$), or address $V - 1$ ($G = 10$). Thus the condition of G determines whether the same address is referenced, or whether addresses in ascending or descending order are referenced.

Referencing the same address for successive data transfers is an unusual feature which requires explanation. Assume for example that some input (or output) unit has a natural rate much slower than that of the computer. Successive input (or output) words may then easily be processed by the main program in real-time without using more than one address in core memory. (Such operation demands that the main program be synchronized in some manner with the natural rate of the peripheral equipment.) Or again assume that the same word is to be written repeatedly on a peripheral device; setting G equal to 11 or 01 will accomplish the function.

As the successive data transfers occur, W, which specifies the number of words involved in the transfer, is decremented. When W reaches zero, the transfer is terminated. If a monitor was established, an internal interrupt for that channel is generated.

The input-output section is capable of controlling 16 input and 16 output transmis-

sions at the same time. Therefore, one unit of special peripheral equipment may use an input channel while a different unit uses the corresponding output channel for simplex operations. Other special peripheral equipment, using both an input channel and the corresponding output channel, can be operated with simultaneous input and output transfers in a full-duplex mode of communication. The standard peripheral units discussed below use an input channel and the corresponding output channel for bidirectional communication, but not at the same time, in a half-duplex mode of communication. Different priority for input and output communication with a peripheral subsystem can be achieved with the use of input and output channels of different number.

The description above indicates the general philosophy of the input-output section. An examination of the instruction repertoire (Function Code 75) will reveal the various detailed options available to the programmers when dealing with input-output problems. The characteristics of the input-output section enhance the system's capability to run multiple programs.

6. Peripheral Equipment

The input-output section of the UNIVAC 1107 is capable of utilizing a large variety of peripheral equipment. Some units of peripheral equipment may be used to provide a hierarchy of auxiliary storage; for example, drums, mass storage, and tapes. Other units may serve as input or output (origin or destination) equipment; for example, card units, printers, and paper-tape subsystem. Other units may serve as information links to other systems. With the adaptable input-output section, the computer can communicate with many real-time devices, such as analog-to-digital and digital-to-analog converters, keysets, printing telegraph equipment, digital communication terminals, digitized radar and tracking systems, display systems, other sensing and instrumentation systems, and other information-processing systems.

The standard peripheral equipment units use an output channel and the corresponding input channel to effect bidirectional communication. The typical interconnection of standard peripheral equipment to the UNIVAC 1107 is shown in Figure 5-1. The output channel (36 lines) provides function words and data words to the peripheral equipment. The input channel (36 lines) provides data words and status, including error conditions, to the computer from the peripheral equipment. The High-Speed Printer uses an input channel to report status information to the computer and uses an output channel to receive data and external functions.

Each subsystem of peripheral equipment consists of one or more peripheral devices of the same type, a peripheral control unit for the particular type of device, and a channel synchronizer. The channel synchronizer, which is a logical part of the control unit, provides the proper interface between the peripheral gear and the computer. Since this function is common to each peripheral control unit, the channel synchronizer is a common compact unit of similar design for each control unit.

The channel synchronizer accepts words from the output channel of the computer and sends character elements of the words to the peripheral control unit. Communication is bidirectional; the channel synchronizer assembles characters from the control unit and sends words to the input channel of the computer. In addition to the assembly-disassembly of words, the channel synchronizer performs many control operations common to the peripheral control units. These operations include primary interpretation of the function word, search by comparing an identifier with data read from an external device, and providing the computer with information about the status of the external equipment.

Multiple computer operation is readily accommodated with the channel synchronizer. Two computers can also communicate directly with a channel synchronizer and thereby share a peripheral complex. A modified control unit operates the peripheral equipment in response to the programmed control words received over the input-output channel from each computer. Figure 6-1 shows two computers sharing the same peripheral equipment. Each of the computers can obtain the use of a shared unit of equipment when available or can capture

the unit from the other computer with programmed external functions. These standard features can be extended to cover more sophisticated switching of peripheral units with multiplexing units tailored to the installation requirements.

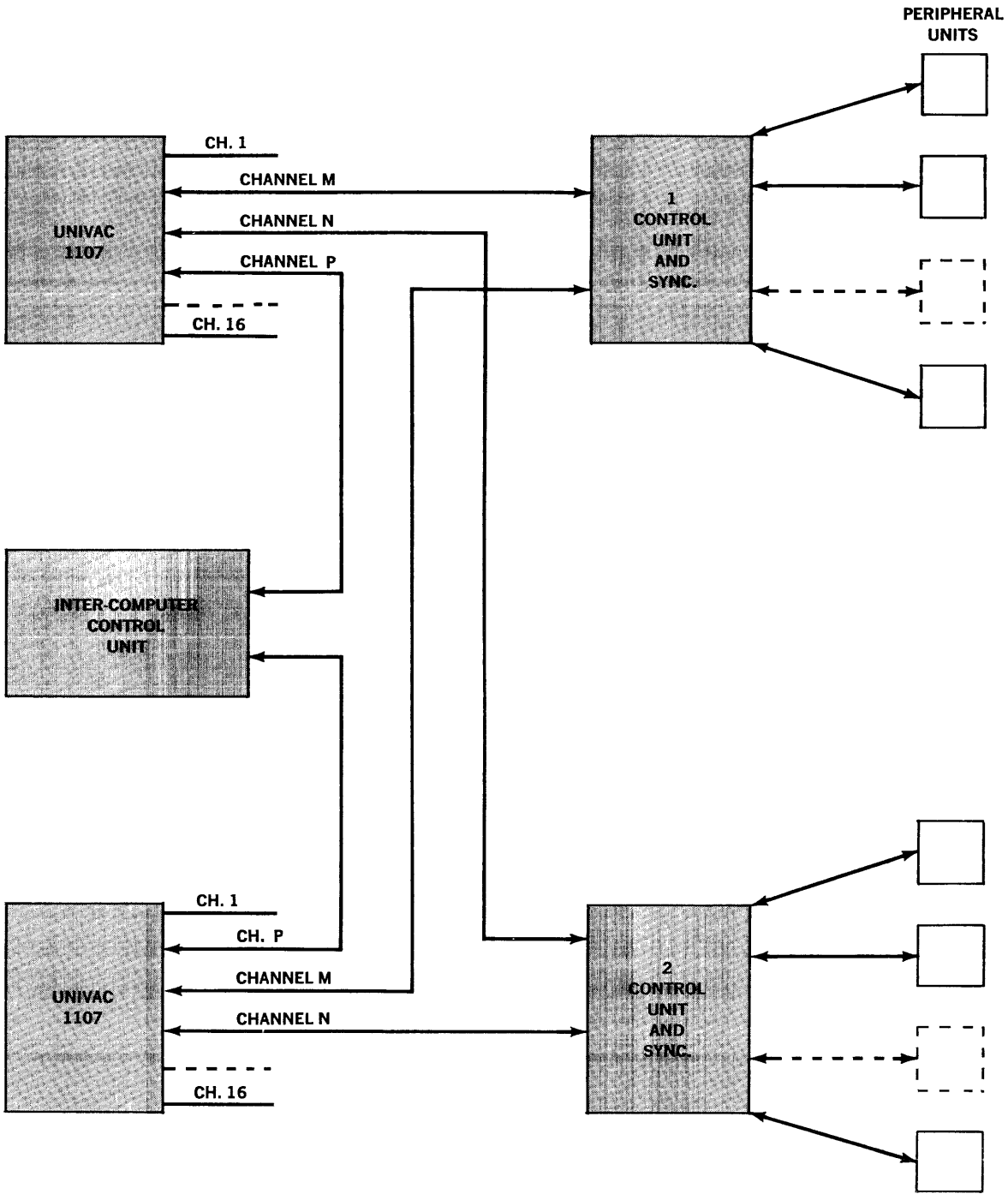


Figure 6-1. Two 1107 Computers Sharing the Same Peripheral Equipment and Communicating Directly with Each Other

Computer-to-computer communication is accomplished with an inter-computer control unit. This unit provides full-duplex communication as shown in Figure 6-1.

The standard UNIVAC 1107 peripheral equipment subsystems are:

- **Magnetic Drum**
- **Magnetic Tape**
  120kc character rate RRU format
  25kc character rate RRU format
  200 characters per inch IBM format
- **High-Speed Printer**
- **Punched Cards**
- **Paper Tape**

MAGNETIC-DRUM SUBSYSTEM

A magnetic-drum subsystem comprises one drum control unit and from 1 to 8 FH-880 or FH-500 drum storage units. This system provides the computer with a large-volume, medium-access memory in optional capacities of from 262,144 to 6,291,456 36-bit words. Up to 15 drum subsystems may directly communicate with the computer, giving it up to more than 94 million words of random-access storage. This amounts to over half a billion 6-bit characters.

Programmed instructions initiate the transfer of any number of words, up to full capacity of core memory. Transfers may begin at any location and thereafter continue sequentially. Words may be transferred between consecutive drum and core memory addresses, since the drum word interval is 16.5 microseconds. Transfers between drum and core memory begin at the time the angular position of the drum coincides with the specified starting address.

The average access time for the initial word is 17 milliseconds for the FH-880 and 8.5 milliseconds for the FH-500, followed by a 16.5-microsecond interval between succeeding words.

A programmed search instruction will initiate an off-line drum word search. This search may be started at any predetermined address and will continue until a find is realized, the complete drum file is searched, or the search is discontinued by programmed intervention. The search area may also be limited by the initial program instruction. When the search key word is found, the transfer of data to core memory will begin automatically.

When initiating a new program in the computer by a bootstrap routine, the magnetic drum can be utilized for program storage.

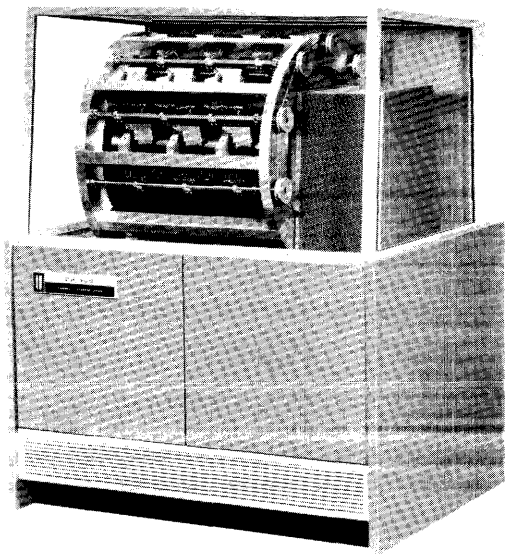


Figure 6-2. Flying Head Magnetic-Drum Unit

Magnetic Drum FH-880

The flying head magnetic-drum unit (see Figure 6-2) provides a large volume of random-access storage. Since the unit is word-addressable, the word transfer rate to the computer may be varied as desired by word-interlacing the data. Pertinent characteristics of the FH-880 are listed below:

Character capacity: 4,718,592 characters
Word capacity: 786,432 words
Bit density: 490 bits per inch
Access time:
    Maximum: 34 milliseconds
    Average: 17 milliseconds
Character transfer rate (maximum): 360kc
Word transfer rate (maximum): 60kc
Type of recording: RZ (return-to-zero)

Magnetic Drum FH-500

This flying-head magnetic-drum unit provides a medium amount of random-access storage with an access speed twice as fast as the FH-880. Because the unit is word-addressable, the word transfer rate to the computer may be varied as desired by the word-interlacing of the data. Pertinent operational characteristics of the FH-500 are listed below:

Character capacity: 1,572,864 characters
Word capacity: 262,144 words
Bit density: 467 bits per inch
Access time:
    Maximum: 17 milliseconds
    Average: 8.5 milliseconds
Character transfer rate (maximum): 375kc
Word transfer rate (maximum): 62.5kc
Type of recording: RZ (return-to-zero)

MAGNETIC-TAPE SUBSYSTEM

A Magnetic-Tape Subsystem consists of a Magnetic-Tape Control Unit, a Uniservo power supply, and up to 12 Uniservo magnetic-tape handlers. Three different Magnetic-Tape Subsystems are available. These are:

Uniservo IIA (Remington Rand UNIVAC Format)
Uniservo IIA (IBM Format)
Uniservo III

A Magnetic-Tape Subsystem performs the following operations:

Read forward or backward; write forward; search forward or backward on the first word read of a block; rewind and rewind with interlock.

Once an operation is initiated by the program, the Magnetic-Tape Control unit supervises the transfer of data between the selected magnetic-tape unit and core memory without further intervention by the Central Computer. Character transfer rates of 12.5 kc, 20 kc, 25 kc, and 120 kc are available depending on the Magnetic-Tape Subsystem selected.

Additional general features of the Magnetic-Tape Subsystems are:

Variable block length recording format.
Detection of bad spots on tape (in RRU format).
Parity checking on each tape frame.
The ability to check the tape handler functions.

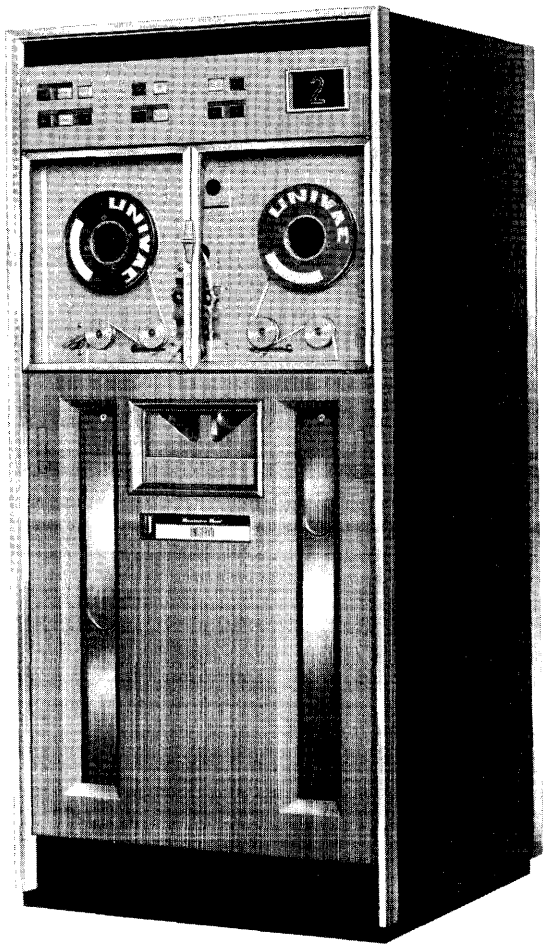


Figure 6-3. Uniservo IIA

Uniservo IIA Subsystem (RRU Format)

Up to 12 Model 72 Uniservo IIA Tape Handlers (see Figure 6-3) are used in this medium-speed tape-storage system of the 1107 Computer. The Uniservo IIA unit records information on either plastic or metal tape in the Remington Rand UNIVAC standard format. In this tape subsystem, only one tape unit may read or write at a given time, but any number of units may rewind while a read or a write operation is in process. Variable block-length format,

character parity checks, and bad-spot detection are provided with this subsystem. General specifications of the Uniservo IIA unit are listed below:

Tape speed: 100 inches per second
Recording density: 125 or 250 bits per inch
Interblock spacing: 1.05 inches
Character transfer rate: 12.5 or 25 kc
Tape length: 2400' plastic or 1500' metal tape
Tape width: 0.5 inch
Type of recording: Return to zero

Uniservo IIA Subsystem (IBM Format)

This tape subsystem provides the 1107 Computer with the ability to read and record on magnetic tape in the IBM 729-II Tape Format. A maximum of 12 Uniservo IIA tape handlers, modified for the IBM format, may be used with this subsystem. Only one tape unit may read or write at a given time, but any number of tape units may rewind while a read or write operation is in process. Variable block-length format, character parity-checking, and start and end-of-block sensing are provided with this subsystem.

General specifications of the Uniservo IIA unit (IBM Format) are as follows:

Tape speed: 100 inches per second
Recording density: 200 bits per inch
Character transfer rate: 20kc
Tape length: 2400' plastic tape
Tape width: 0.498±.002 inch
Type of recording: Non-return to zero

Uniservo III Subsystem

The Uniservo III high-performance tape handler provides the 1107 Computer with a 120kc character transfer rate tape system. Up to 12 Uniservo III tape handlers may be used with this tape system. To provide increased flexibility this tape subsystem is capable of performing two read operations or a read and a write operation simultaneously. Two Uniservo III units may be read at the same time also. Two input and two output 1107 channels are used to provide this flexibility. Up to 7 high-performance tape subsystems can be used on the 1107 Computer.

Other features of this subsystem are variable block-length recording, tape bad-spot detection, write-check mode of recording, and tape-frame parity-checking. Specifications for the Uniservo III subsystem are:

Tape speed: 100 inches per second
Recording density: 1,000 bits per inch
Interblock spacing: 0.75 inches
Character transfer rate: 120kc
Tape length: 2400' plastic tape
Tape width: 0.498±.002 inch
Type of recording: Pulse-phase modulation

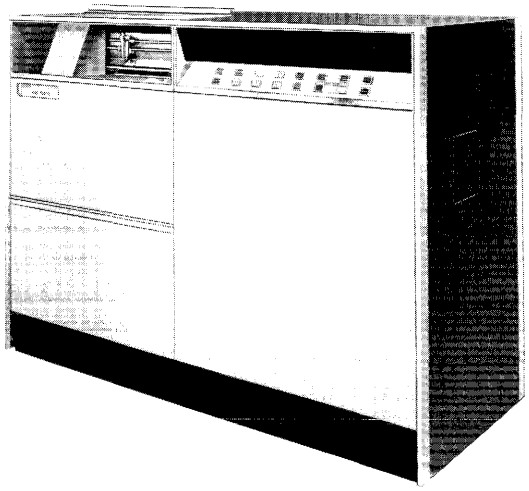


Figure 6-4. High-Speed Printer

HIGH-SPEED PRINTER SUBSYSTEM

The High-Speed Printer subsystem consists of a Control Unit and one or two High-Speed Printers. The High-Speed Printer is under the complete control of the High-Speed Printer Control Unit performing whatever functions the latter demands of it. These functions include paper spacing of 0 to 63 lines, selection of one of the two units, and print. The control unit may alternately control either of two High-Speed Printers.

Two models of the High-Speed Printer are available with the High-Speed Printer subsystem. They are the Model 46 and the Model 151 (see Figure 6-4). The Model 46 prints at a rate of 600 lines per minute and has been used successfully by UNIVAC Computer Systems for several years. The Model 151 is a new solid-state High-Speed Printer

with a printing rate of 700 lines per minute. It is considerably smaller than conventional High-Speed Printers. The characteristics of both the Model 46 High-Speed Printer and the Model 151 High-Speed Printer are reflected in the following specifications:

Line capacity: 128 characters (maximum)
Available characters: 26 Alpha, 10 Numeric, and 15 Special Characters
Number of carbons: At least 4 with 12 lb. stock
Paper stock: Any standard sprocket-fed paper, 4-27 inches wide and up to card stock in weight
Horizontal spacing: 10 characters per inch
Vertical spacing: 6 lines per inch, single-spaced

PUNCHED-CARD SUBSYSTEM

The Punched-Card Subsystem consists of a Punched-Card Control Unit, a Card-Reader, and a Card-Punch. Two subsystems are available, the Medium-Speed System and the High-Performance System. Either 90-column or 80-column equipment may be supplied with either subsystem. Supplied with the subsystem is a card buffer. This buffer allows concurrent operation of the Card-Reader and Card-Punch at the maximum card-handling rates of the devices.

Once an operation is initiated by the program, the Punched-Card Control Unit supervises the transfer of data between the Reader and/or Punch and the Computer Core Memory without further intervention. Checking is provided in both the read and punch mode.

Medium-Performance Subsystem

In this subsystem, the card-reader brush-reads cards at two stations at a rate of 600 per minute. The second reading station is used as a checking station, and the unit has three output stackers. The card-punch for this subsystem block-punches cards at a rate of 150 cards per minute. A post-punch reading station is utilized for checking purposes. This punch has two output stackers.

High-Performance Subsystem

In this subsystem the cards are read by photocells in a dual sensing system at a rate of 700 cards per minute. Dual sensing is utilized to provide a read-check operation. The punch used in this subsystem row-punches the cards at a rate of 300 cards per minute. Checking is accomplished by reading the card at a post-punch reading station. This punch has two output stackers.

PAPER-TAPE SUBSYSTEM

There are two optional paper-tape subsystems available. One subsystem uses a one-character-per-computer-word transfer. The other subsystem utilizes the word assembly-disassembly features of the channel synchronizer. Control and operation of the former subsystem is discussed under the Supervisory Console section. The latter subsystem is discussed below. Either subsystem is capable of handling 5-, 6-, 7-, or 8-level paper tape.

The paper-tape subsystem consists of a paper-tape control unit and one or two paper-tape readers and punches (see Figure 6-6). Odd or even parity-checking and generation are provided for seven and eight-level paper tape. Verification of all punched data is provided by a post-read station on the punch. Selection of parity and tape levels is provided at the reader and punch.

Paper-Tape Reader

The paper-tape reader is capable of reading up to eight channels at a rate of 400 frames per second. The unit may be operated in either "start-stop" or "free-running" modes. The unit will accept tape widths of $^{11}/_{16}$ in., $^{7}/_{8}$ in., or 1 in.

Optional paper-tape readers are available with operational speeds of over 1,000 frames per second.

Paper-Tape Punch

The Paper-Tape Punch is capable of perforating up to eight channels at a rate of 110 frames per second. The unit will accept tape widths of $^{11}/_{16}$ in., $^{7}/_{8}$ in., or 1 in. A post-read station is provided so that the punched data can be verified.

Optional paper tape punches are available with operational speeds of 300 frames per second.

MASS-STORAGE FILE SUBSYSTEM

A large random-access storage system will be incorporated in the 1107 system, providing access time in the range of 100 milliseconds, with a bit capacity of 600 million bits and a word transfer rate of approximately 10,000 words per second. From one to eight mass-storage units may be attached via a control unit to an input and output channel on the computer.

Programmed instructions initiate the reading or writing of any number of words to or from this mass-storage unit. Information will be stored in serial fashion on the magnetic medium and separated into sectors on each track. Reading may begin at any word location and continue until the specified number of words has been transferred to core memory. Writing must start at the beginning of a sector. A parity-check bit is added when writing and is used as a check bit during a read operation.

7. Systems Programming

The following programs will be provided for the 1107 Computer System:

ALGOL—An algorithmic language compiler. The specifications are those developed jointly by the ACM Committee on Programming Languages and the GAMM* Committee on Programming. The report was published in the *Communications of the ACM,* May and July, 1960.

FORTRAN—A translator which will accept problems written in FORTRAN language and translate them to a format which may then be processed. This will permit problems previously coded in FORTRAN to be run on the 1107 without revision.

COBOL—A data-processing compiler. It is for programs stated in the English language of COBOL 1960—as defined in the Government Printing Office publication dated April, 1960. Additional standard COBOL language and compiler features will be supplied consistent with the release of official specifications.

SIMULATOR—A routine to execute interpretatively the instruction repertoire of the 1107 on an 1103A, 1103AS, and 1105 will be provided. This will be used in checking programs written for the 1107 prior to the availability of the 1107 computer.

BASIC UTILITY LIBRARY

An Assembly System

An advanced computer-oriented mnemonic code assembly system will be provided. This routine will accept 1107 instructions containing mnemonic function codes and designators, and symbolic operand addresses and translate these instructions to an absolute or relative form ready for loading and operation on the 1107 Computer. The assembler will contain a means of easily correcting the source code, allocating assembled programs, and a side-by-side output of source and assembled programs. Also included will be facilities for incorporating library routines in the assembled program.

(1) Phase I development consists of preparation of the assembly program to operate on the UNIVAC 1103A Computer. The established 1107 assembly language will be accepted as input and 1107 machine code will be produced as the object program.

(2) Phase II development consists of the preparation of the assembly program to operate on the UNIVAC 1107 Computer and to produce UNIVAC 1107 code as the object program.

UNIVAC 1107 Executive System

A routine to accomplish automatically the execution of runs for a previously determined computer schedule, to extract the programs that are to be executed and position these in their operating locations, to provide for the time-sharing of several programs operating in parallel, and to incorporate checking features to be carried out during the problem run.

Sort-Merge Program

Routines to arrange random items in an ordered sequence and/or to combine two or more ordered sequences into a single sequence based on information contained in specified fields of each item.

**Gesellschaft für Angewandte Mathematik und Mechanik.*

Input-Output Routines

A set of routines to perform the principal input and output functions for standard peripheral equipment.

Debugging Aids

A set of routines which provides a means of utilizing the computer to assist the programer in debugging his programs. Among the routines included are the following:

Changed-Word Post Mortem—A routine to compare the contents of program or data areas with a selected image area.

Address Reference Search—A routine to detect all words in the computer memory which reference a particular address.

Dump Selected Memory Area—A routine to provide the contents of all locations within a specified memory area.

Function Evaluation Routines

A set of commonly used mathematical routines. The initial set of routines will include sine, cosine, tangent, arcsine, arccosine, arctangent, square root, natural logarithm, and exponential. These routines will be written in fixed and floating point.

Librarian Routine

A routine to be used for building and maintaining a library of subroutines. It will be capable of inserting, deleting, or changing the routines of the library, and of extracting routines for use in a programming system. This routine allows the contents of the library to be changed with ease according to customer requirements.

8. Instruction Repertoire

Listed below are the instructions constituting the repertoire of the UNIVAC 1107, together with a symbolic description of their functions, execution times, and mnemonic codes of the 1107 assembly system.

GLOSSARY OF SYMBOLS AND TERMS

Grouped here for convenience and easy reference are the abbreviations and symbols used throughout the foregoing discussion of the UNIVAC 1107.

( ) — Indicates "the contents of" the address given within the parentheses.

u — The address, or base address, in the right-hand 16 bits of the instruction word.

U — The effective address of the operand to be used in the operation. It also serves as the shift count in shift instructions. $U = u + (B)_Q$ if no indirect addressing is indicated. If indirect addressing is indicated, $u + (B)_Q$ is the address at which U may be obtained.

A_n, $(U)_n$ — The subscript n indicates the bit number under discussion.

$( )_{17\text{-}00}$ — The subscript numbers represent the range of bit positions considered in the word whose address is given within the parentheses. For example:
$( )_{\text{right half}} = ( )_{17\text{-}00}$
$( )_{\text{left half}} = ( )_{35\text{-}18}$
The bits are always numbered from right to left.

B_Q — Modifier portion of index register, $B_{17\text{-}00}$.

B_Δ — Increment portion of index register, $B_{35\text{-}18}$, used to increment the modifier B_Q.

b — b designator (bits 21-18) of the instruction word.

a — The a-designator (bits 25-22 of the instruction word). In arithmetic instructions, a designates one of the A-registers, in input-output instructions, a designates an input or output channel; in certain other instructions, a designates a B-register or an R-register.

R — Refers to a group of special registers.

M — Mask register (an R-register).

$( )'$ — The prime on a quantity represents the one's complement of that quantity.

NI — Next Instruction.

P — Program address count in the P-register.

$\rightarrow$ — Transfer the word (or words) shown at the left of the arrow to the address (or addresses) shown at the right of the arrow.

$( ) \odot ( )$ — The logical product, or logical AND, is defined by the table:

	0	1
0	0	0
1	0	1

$( ) \oplus ( )$ — The logical sum, also called the Inclusive OR:

	0	1
0	0	1
1	1	1

$( ) \overline{\oplus} ( )$ — The logical difference, controlled complement, add-without-carry, Exclusive OR:

	0	1
0	0	1
1	1	0

INSTRUCTION REPERTOIRE

f	j	NAME	DESCRIPTION		EXECUTION TIME IN μ SEC. Alternate Core Banks	EXECUTION TIME IN μ SEC. Same Core Bank	MNEMONIC CODE
01	0-17	Store Positive	$(A) \rightarrow U$		4.0	8.0	STP
02		Store Negative	$-(A) \rightarrow U$		4.0	8.0	STN
03		Store Magnitude	$\lvert(A)\rvert \rightarrow U$		4.0	8.0	STM
04		Store R_a	$(R_a) \rightarrow U$		4.0	8.0	STR
05		Store Zero	$0 \rightarrow U$ (Clear U)		4.0	8.0	STZ
06		Store B_a	$(B_a) \rightarrow U$		4.0	8.0	STB
10		Load Positive	$(U) \rightarrow A$		4.0	8.0	LDP
11		Load Negative	$-(U) \rightarrow A$		4.0	8.0	LDN
12		Load Positive Magnitude	$\lvert(U)\rvert \rightarrow A$		4.0	8.0	LDM
13		Load Negative Magnitude	$-\lvert(U)\rvert \rightarrow A$		4.0	8.0	LNM
14		Add	$(A) + (U) \rightarrow A$		4.0	8.0	ADD
15		Subtract	$(A) - (U) \rightarrow A$		4.0	8.0	SUB
16		Add Magnitude	$(A) + \lvert(U)\rvert \rightarrow A$		4.0	8.0	ADM
17		Subtract Magnitude	$(A) - \lvert(U)\rvert \rightarrow A$		4.0	8.0	SBM
20		Add and Load	$(A) + (U) \rightarrow A + 1$		4.0	8.0	ADL
21		Subtract and Load	$(A) - (U) \rightarrow A + 1$		4.0	8.0	SBL
22†		Block Transfer	$(V_1)_i \rightarrow (V_2)_i$ repeated k times. Initial V_1 address is $u + (B_b)_{17-0}$, and subsequent addresses are formed by incrementation by $(B_b)_{35-18}$. Similarly, V_2 addresses are $u + (B_a)_{17-0}$ incremented by $(B_a)_{35-18}$.		8.0	8.0	BTR
23		Load R_a	$(U) \rightarrow R_a$		4.0	8.0	LDR
24		Add to B_a	$(B_a) + (U) \rightarrow B_a$		4.0	8.0	ADB
25		Subtract from B_a	$(B_a) - (U) \rightarrow B_a$		4.0	8.0	SBB
26		Load B_a Modifier Only	$(U) \rightarrow B_{a17-0}$		4.0	8.0	LBM
27		Load B_a	$(U) \rightarrow B_a$		4.0	8.0	LDB
30		Multiply Integer	$(A) \cdot (U) \rightarrow A, A + 1$		12.7	16.7	MPI
31		Multiply Single (Integer)	$(A) \cdot (U) \rightarrow A$		12.0	16.0	MPS
32		Multiply Fractional	$(A) \cdot (U) \rightarrow A, A + 1$		13.3	17.3	MPF
34		Divide Integer	$(A, A + 1) \div (U)$; Quotient $\rightarrow A$, Remainder $\rightarrow A + 1$		31.0	35.0	DVI
35		Divide Single and Load (Fractional)	$(A) \div (U)$; Quotient $\rightarrow A + 1$, No Remainder		31.0	35.0	DVL
36		Divide Fractional	$(A, A + 1) \div (U)$; Quotient $\rightarrow A$, Remainder $\rightarrow A + 1$		31.0	35.0	DVF
40		Selective Set	$(A) \rightarrow A + 1$. Then set $(A + 1)_n$ for $(U)_n = 1$ i.e., $(A) \oplus (U) \rightarrow A + 1$		4.0	8.0	SSE
41		Selective Complement	$(A) \rightarrow A + 1$. Then complement $(A + 1)_n$ for $(U)_n = 1$ i.e., $(A) \overline{\oplus} (U) \rightarrow A + 1$		4.0	8.0	SCP
42		Selective Clear	$(A) \rightarrow A + 1$. Then clear $(A + 1)_n$ for $(U)_n = 0$ i.e., $(A) \odot (U) \rightarrow A + 1$		4.0	8.0	SCL
43		Selective Substitute	$(A) \rightarrow A + 1$. Then $(U)_n \rightarrow (A + 1)_n$ for $(M)_n = 1$ i.e., $(A) \odot (M)' + (U) \odot (M) \rightarrow A + 1$		4.7	8.7	SSU
44		Selective Even Parity Test	If $[(A) \odot (U)]$ is even parity, Skip NI	No Skip	6.0	10.0	SEP
				Skip	10.0	14.0	
45		Selective Odd Parity Test	If $[(A) \odot (U)]$ is odd parity, Skip NI	No Skip	6.0	10.0	SOP
				Skip	10.0	14.0	
47		Test Modifier	If $(B_a)_{17-0} < (U)$, take NI; If $(B_a)_{17-0} \geq (U)$, Skip. In either case, $(B_a)_{17-0} + (B_a)_{35-18} \rightarrow B_{a17-0}$	No Skip	4.7	8.7	TMO
				Skip	8.7	12.7	
50		Test Zero	Skip NI if $(U) = 0$	No Skip	4.0	8.0	TZR
				Skip	8.0	12.0	
51		Test Not Zero	Skip NI if $(U) \neq 0$	No Skip	4.0	8.0	TNZ
				Skip	8.0	12.0	
52		Test Equal	Skip NI if $(U) = (A)$	No Skip	4.0	8.0	TEQ
				Skip	8.0	12.0	
53		Test Not Equal	Skip NI if $(U) \neq (A)$	No Skip	4.0	8.0	TNE
				Skip	8.0	12.0	
54		Test Less Than or Equal	Skip NI if $(U) \leq (A)$	No Skip	4.0	8.0	TLE
				Skip	8.0	12.0	
55		Test Greater Than	Skip NI if $(U) > (A)$	No Skip	4.0	8.0	TGR
				Skip	8.0	12.0	
56		Test Within Limits	Skip NI if $(A) < (U) \leq (A + 1)$ (Note: $(A) < (A + 1)$)	No Skip	4.7	8.7	TWL
				Skip	8.7	12.7	
57		Test Outside Limits	Skip NI if $(U) \leq (A)$ or $(U) > (A + 1)$ (Note: $(A) \leq (A + 1)$)	No Skip	4.7	8.7	TOL
				Skip	8.7	12.7	

† All repeat operations (22, 62-67, 71) take 16 μ sec. combined setup and termination time.

INSTRUCTION REPERTOIRE

f	j	NAME	DESCRIPTION		EXECUTION TIME IN μ SEC. Alternate Core Banks	EXECUTION TIME IN μ SEC. Same Core Bank	MNEMONIC CODE
60	0-17	Test Positive	Skip NI if (U) $\geq$ 0	No Skip	4.0	8.0	TPO
				Skip	8.0	12.0	
61		Test Negative	Skip NI if (U) $<$ 0	No Skip	4.0	8.0	TNG
				Skip	8.0	12.0	
62†		Search Equal	Skip NI if $(U)_i = (A)$	No Skip	4.0	4.0	SEQ
			Repeated k times	Skip	4.0	4.0	
63†		Search Not Equal	Skip NI if $(U)_i \neq (A)$	No Skip	4.0	4.0	SNE
			Repeated k times	Skip	4.0	4.0	
64†		Search Less Than or Equal	Skip NI if $(U)_i \leq (A)$	No Skip	4.0	4.0	SLE
			Repeated k times	Skip	4.0	4.0	
65†		Search Greater Than	Skip NI if $(U)_i > (A)$	No Skip	4.0	4.0	SGR
				Skip	4.0	4.0	
66†		Search Within Limits	Skip NI if $(A) < (U)_i \leq (A + 1)$	No Skip	4.7	4.7	SWL
			(Note: $(A) < (A + 1)$)	Skip	4.7	4.7	
67†		Search Outside Limits	Skip NI if $(U)_i \leq (A)$ or $(U)_i > (A+1)$	No Skip	4.7	4.7	SOL
			(Note: $(A) < (A + 1)$)	Skip	4.7	4.7	
70		Index Jump	If $(CM)_{ja} > 0$, Jump to U	No Jump	8.0	8.0	IXJP
			$(CM)_{ja} < 0$, Take NI	Jump	4.0	4.0	
			Then $(CM)_{ja} - 1 \rightarrow CM_{ja}$				
			NOTE: j in this instruction serves with the a-designator to specify any one of the 128 words of Control Memory.				
71†	*						
	00	Masked Search Equal	Skip NI if $(U)_i \odot (M) = (A) \odot (M)$	No Skip	4.0	4.0	MSEQ
			Repeated k times	Skip	4.0	4.0	
	01	Masked Search Not Equal	Skip NI if $(U)_i \odot (M) \neq (A) \odot (M)$	No Skip	4.0	4.0	MSNE
			Repeated k times	Skip	4.0	4.0	
	02	Masked Search Less Than or Equal	Skip NI if $(U)_i \odot (M) \leq (A) \odot (M)$	No Skip	4.0	4.0	MSLE
			Repeated k times	Skip	4.0	4.0	
	03	Masked Search Greater Than	Skip NI if $(U)_i \odot (M) > (A) \odot (M)$	No Skip	4.0	4.0	MSGR
			Repeated k times	Skip	4.0	4.0	
	04	Masked Search Within Limits	Skip NI if $(A) \odot (M) < (U)_i \odot (M) \leq (A + 1) \odot (M)$	No Skip	4.7	4.7	MSWL
			(Note: $(A) \odot (M) < (A + 1) \odot (M)$)	Skip	4.7	4.7	
			Repeated k times				
	05	Masked Search Outside Limits	Skip NI if $(U)_i \odot (M) \leq (A)$ or $(U)_i \odot (M) > (A + 1)$	No Skip	4.7	4.7	MSOL
			(Note: $(A) \odot (M) < (A + 1) \odot (M)$)	Skip	4.7	4.7	
			Repeated k times				
72	*						
	00	Wait for Interrupt	The computer program sequence stops (i.e., P is not advanced). The wait condition is removed by an interrupt.		4.0		WAIT
	01	Return Jump	$(P) \rightarrow U_{17-0}$ and Jump to U + 1		8.0	8.0	RTJP
	02	Positive Bit Control Jump	If $(A)_{35} = 0$, Jump to U	No Jump	4.0	4.0	PBJP
			Shift (A) left one in either case	Jump	8.0	8.0	
	03	Negative Bit Control Jump	If $(A)_{35} = 1$, Jump to U	No Jump	4.0	4.0	NBJP
			Shift (A) left one in either case	Jump	8.0	8.0	
	04	Add Halves	$(A)_{17-0} + (U)_{17-0} \rightarrow A_{17-0}$		4.0	8.0	ADDH
			$(A)_{35-18} + (U)_{35-18} \rightarrow A_{35-18}$				
	05	Subtract Halves	$(A)_{17-0} - (U)_{17-0} \rightarrow A_{17-0}$		4.0	8.0	SUBH
			$(A)_{35-18} - (U)_{35-18} \rightarrow A_{35-18}$				
	06	Add Thirds	$(A)_{35-24} + (U)_{35-24} \rightarrow A_{35-24}$		4.0	8.0	ADDT
			$(A)_{23-12} + (U)_{23-12} \rightarrow A_{23-12}$				
			$(A)_{11-0} + (U)_{11-0} \rightarrow A_{11-0}$				
	07	Subtract Thirds	$(A)_{35-24} - (U)_{35-24} \rightarrow A_{35-24}$		4.0	8.0	SUBT
			$(A)_{23-12} - (U)_{23-12} \rightarrow A_{23-12}$				
			$(A)_{11-0} - (U)_{11-0} \rightarrow A_{11-0}$				
	10	Execute Remote Instruction	Execute the Instruction at U		4.0	— + Execution Time	EXRI
	11	Load Memory Lockout Register	$U_{5-0} \rightarrow$ MLR		4.0	—	LMLR
			For $U_0 = 1$ lockout 0—4095				
			$U_1 = 1$ lockout 4096—8191				
			$U_2 = 1$ lockout 8192—16383				
			$U_3 = 1$ lockout 16384—32767				
			$U_4 = 1$ lockout applies to 1st BANK				
			$U_5 = 1$ lockout applies to 2nd BANK				
73	*						
	00	Single Right Circular Shift‡	Shift (A) right U places circularly		4.0		SCSH
	01	Double Right Circular Shift	Shift (A, A + 1) right U places circularly		4.0		DCSH
	02	Single Right Logical Shift	Shift (A) right U places, end off; fill with zeros (Max. Shift = 36)		4.0		SLSH

*j serves as part of the Function Code

† All repeat operations (22, 62-67, 71) take 16 μ sec. combined setup and termination time.

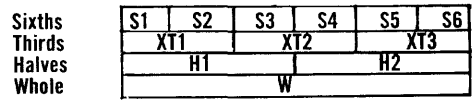
‡Instruction execution time is independent of the number of shifts performed (e.g. a shift of 72 takes 4 microseconds). There are no memory references in the first six shift instructions, 73 00 — 73 05; consequently, the distinction between **alternate** core banks and the **same** core bank is irrelevant.

INSTRUCTION REPERTOIRE

f	j	NAME	DESCRIPTION		EXECUTION TIME IN μ SEC. Alternate Core Banks	EXECUTION TIME IN μ SEC. Same Core Bank	MNEMONIC CODE
	03	Double Right Logical Shift	Shift (A, A + 1) right U places, end off; fill with zeros. (Max. Shift = 72)		4.0		DLSH
	04	Single Right Arithmetic Shift	Shift (A) right U places, end off; fill with sign bits.		4.0		SASH
	05	Double Right Arithmetic Shift	Shift (A, A + 1) right U places, end off; fill with sign bits. (Max. Shift = 72)		4.0		DASH
	06	Scale Factor Shift	(U) → A, shift A left circularly until $A_{35} \neq A_{34}$ or until A has been shifted 36 times. Store the scaled quantity in A and the number of shifts that occurred in A + 1.		6.0	10.0	SFSH
74	*						
	00	Zero Jump	Jump to U if (A) = 0	No Jump	4.0	4.0	ZRJP
				Jump	8.0	8.0	
	01	Non-zero Jump	Jump to U if (A) ≠ 0	No Jump	4.0	4.0	NZJP
				Jump	8.0	8.0	
	02	Positive Jump	Jump to U if (A) ≥ 0	No Jump	4.0	4.0	POJP
				Jump	8.0	8.0	
	03	Negative Jump	Jump to U if (A) < 0	No Jump	4.0	4.0	NGJP
				Jump	8.0	8.0	
	04	Console Selective Jump	Jump to U if A = key setting on console (1 of 15)		4.0	4.0	CSJP
	05	Selective Stop Jump	Stop if A = stop key setting on console (1 of 4), always jump to U		4.0	4.0	SSJP
	06	No Operation	Do Nothing; continue with NI		4.0	4.0	NOOP
	07	Enable All Input—Output Interrupts and Jump	Jump to U and permit interrupts to occur		4.0	4.0	EIJP
	10	Even Jump	Jump to U if $(A)_0 = 0$	No Jump	4.0	4.0	EVJP
				Jump	8.0	8.0	
	11	Odd Jump	Jump to U if $(A)_0 = 1$	No Jump	4.0	4.0	ODJP
				Jump	8.0	8.0	
	12	Modifier Jump	If $(B_a)_{17-0} > 0$, Jump to U	No Jump	4.0	4.0	MOJP
			If $(B_a)_{17-0} \leq 0$, Take NI	Jump	8.0	8.0	
			In either case $(B_a)_{17-0} + (B_a)_{35-18} \rightarrow B_{a17-0}$				
	13	Load Modifier and Jump	(P) → $(B_a)_{17-0}$ and Jump to U		4.0	4.0	LMJP
	14	Overflow Jump	Jump to U if overflow cond. is set		4.0	4.0	OVJP
	15	No-Overflow Jump	Jump to U if overflow cond. is not set		4.0	4.0	NOJP
	16	Carry Jump	Jump to U if carry cond. is set		4.0	4.0	CYJP
	17	No-Carry Jump	Jump to U if carry cond. is not set		4.0	4.0	NCJP
75	*						
	00	Initiate Input Mode	(U) → input control word a, and initiate input mode on channel a.		4.0	8.0	IIPM
	01	Initiate Monitored Input Mode	(U) → input control word a, and initiate input mode on channel a with monitor.		4.0	8.0	IMIM
	02	Input Mode Jump	Jump to U if channel a is in the input mode.		4.0	4.0	IMJP
	03	Terminate Input Mode	Terminate input mode on channel a.		4.0	4.0	TIPM
	04	Initiate Output Mode	(U) → output control word a, and initiate output mode on channel a.		4.0	8.0	IOPM
	05	Initiate Monitored Output Mode	(U) → output control word a, and initiate output mode on channel a with monitor.		4.0	8.0	IMOM
	06	Output Mode Jump	Jump to U if channel a is in the output mode.		4.0	4.0	OMJP
	07	Terminate Output Mode	Terminate output mode on channel a.		4.0	4.0	TOPM
	10	Initiate Function Mode	(U) → output control word a, and initiate function mode on channel a.		4.0	8.0	IFNM
	11	Initiate Monitored Function Mode	(U) → output control word a, and initiate function mode on channel a with monitor.		4.0	8.0	IMFM
	12	Function Mode Jump	Jump to U if channel a is in the function mode.		4.0	4.0	FMJP
	13	Force External Transfer	Request external function or output word on channel a.		4.0	4.0	FEXT
	14	Enable All External Interrupts	All external interrupts are permitted to occur.		4.0	4.0	EAEI
	15	Disable All External Interrupts	All external interrupts are prevented from occurring.		4.0	4.0	DAEI
	16	Enable Single External Interrupt	An external interrupt on channel a is permitted to occur.		4.0	4.0	ESEI
	17	Disable Single External Interrupt	An external interrupt on channel a is prevented from occurring.		4.0	4.0	DSEI
76	*						
	00	Floating Add	(A) + (U) → A, A + 1		14.0	18.0	FLAD
	01	Floating Subtract	(A) − (U) → A, A + 1		14.0	18.0	FLSB
	02	Floating Multiply	(A) • (U) → A, A + 1		12.7	16.7	FLMP
	03	Floating Divide	(A) ÷ (U); Quotient → A, Remainder → A + 1		26.0	30.0	FLDV
	04	Floating Point Unpack	Unpack (U), store mantissa in A + 1 and store the biased characteristic in A		4.7	8.7	FLUP
	05	Floating Point Normalize Pack	Form the packed normalized number from mantissa stored in U and from biased characteristic in A, and store at A + 1		6.7	10.7	FLNP
	06	Floating Characteristic Difference Magnitude	Absolute value of $\lvert(A)_{34-27}\rvert - \lvert(U)_{34-27}\rvert \rightarrow A + 1$		4.0	8.0	FLCM
	07	Floating Characteristic Difference	$\lvert(A)_{34-27}\rvert - \lvert(U)_{34-27}\rvert \rightarrow A + 1$		4.0	8.0	FLCD

*j serves as part of the Function Code

In the mnemonic column:
Normal *J* designator



An *X* placed before *H*1 or *H*2 extends the sign bit when transferring *to* arithmetic, halfwords being the only section of a word for which the sign extension may be specified. Thirds are always extended and sixths never.

In transfers *from* arithmetic, the signs are never extended. In this case *T*1, *T*2, and *T*3 must be used for the one-third word transfer.

In addition one may transfer the lower half of the current instruction *to* arithmetic either extended or not. These are indicated *UOP* and *XUOP*. The *h* and *i* designators are not interpreted in the usual way in this case; they are merely bits of the number transferred.

9. Application Notes

EQUIPMENT CONFIGURATION

A typical system configuration may consist of a Central Computer with:

32,768 words of magnetic core memory
7 magnetic-tape units
1 magnetic-drum unit
1 High-Speed Printer
1 Supervisory Console
1 Paper-Tape Reader and Punch Unit

This set of equipment would occupy less than 200 square feet of floor area of an operational area of approximately 1,000 square feet and dissipates approximately 42 kw of power.

UTILIZATION OF PERIPHERAL EQUIPMENT

The maximum input-output transfer rate is limited by the cycle time of core memory. With a basic memory cycle time of 4.0 microseconds, the transfer rate of a memory bank is 250,000 words per second. This word rate is the maximum instantaneous input-output transfer rate for a memory bank. However, in a practical system, since the memory time is required for other purposes, such as to obtain instructions and operands, this transfer rate cannot be fully realized. In writing a specific program, the maximum concurrent peripheral input-output transfer rate must be considered so that it does not overload the system.

A representative example of a configuration of peripheral equipment operating concurrently is given below:

1 magnetic-drum channel	60,000 words per sec.
1 magnetic-tape channel at 120,000 char. per sec.	20,000 words per sec.
1 magnetic-tape channel at 25,000 char. per sec.	4,167 words per sec.
1 High-Speed Printer channel (while loading buffer)	4,166 words per sec.
Total	88,333 words per sec.

The computer can easily handle this transfer rate since it is only about one-third the maximum transfer rate. When the input-output transfer rate is greater than one-half the maximum rate of a memory bank, special consideration must be given to factors such as the amount of internal data-processing required, the data and control word transfers which are peculiar to each peripheral unit, and the over-all duty cycle of such a combination of transfers. The transfer-data rate given in the above example would not be sustained for more than a few thousand words unless the operation performed was essentially buffering an external unit to external unit transfer with the input and output rates approximately in balance.

The above word rate for the printer is the instantaneous rate to fill the 128-character buffer. The average word rate for the printer is 233 words per second. Similarly, transfers with the drum and tapes are usually performed as finite block transfers which also make the average word rate significantly lower.

MAINTENANCE

The 1107 Computer circuits use advanced techniques of solid-state design, with a positive-OR circuit forming the basic module. Conservative operating speeds provide the functional capabilities with the reliability necessary for real-time data processing.

Special-purpose support equipment is available in addition to the normal complement of standard test equipment such as oscilloscopes and meters. A chassis test unit pro-

vides the facility to test each circuit module contained within the system. The down time of the system is minimized by the use of diagnostic routines with marginal checking procedures in regular preventive maintenance programs. Potential sources of trouble in the modules are thereby detected and the modules replaced. Detailed diagnosis and repair are performed with the aid of the test unit. As a result, the UNIVAC 1107 System is maintained on a plane of very high reliability, consistent with the outstanding performance of the Central Computer and its memory modules.

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

PRINTED IN U.S.A. UT2440