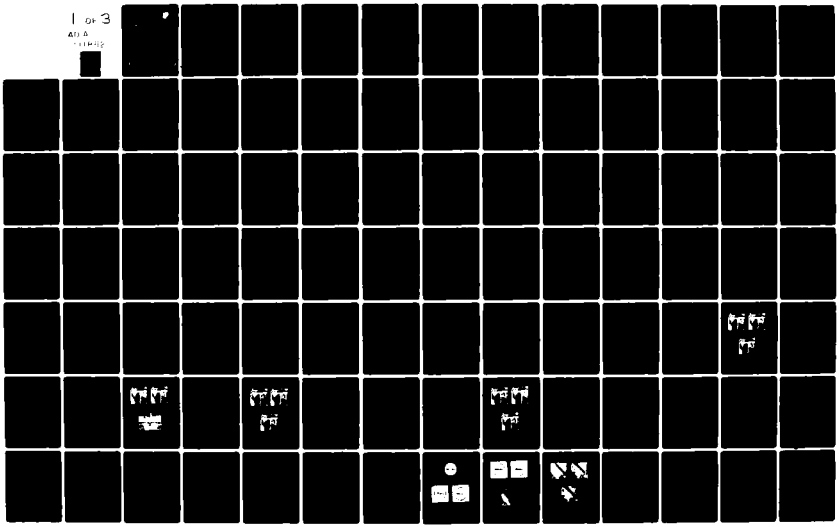
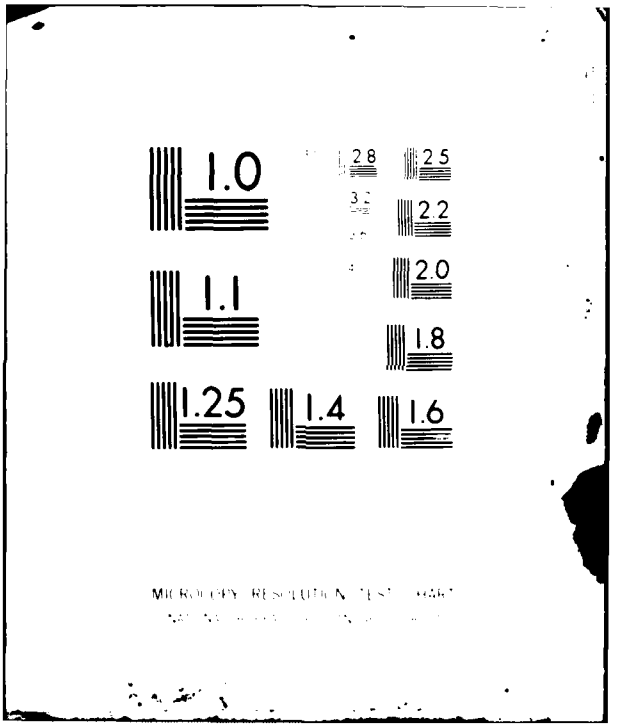
AD-A111 882 MISSISSIPPI STATE UNIV MISSISSIPPI STATE DEPT OF ELEC--ETC F/G 5/8
THE APPLICATION OF SPECIAL COMPUTING TECHNIQUES TO SPEED-UP IMA--ETC(U)
DEC 81 R W MCLAREN, W D MCFARLAND F30602-80-C-0032
UNCLASSIFIED RADC-TR-81-230 NL

1 OF 3
AD A
111882




1.0
2.8
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

(12)

AD A111882

RADC-TR-81-230
Final Technical Report
December 1981

THE APPLICATION OF SPECIAL COMPUTING TECHNIQUES TO SPEED-UP IMAGE FEATURE EXTRACTION AND PROCESSING TECHNIQUES

University of Missouri-Columbia

Robert W. McLaren
William D. McFarland

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-230 has been reviewed and is approved for publication.

APPROVED:

FREDERICK W. RAHRIG
Project Engineer

APPROVED:

OWEN R. LAWTER, Colonel, USAF
Chief, Intelligence and Reconnaissance Division

FOR THE COMMANDER:

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRRE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-230 AD-A111882	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE APPLICATION OF SPECIAL COMPUTING TECHNIQUES TO SPEED-UP IMAGE FEATURE EXTRACTION AND PROCESSING TECHNIQUES	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 9 Jan 80 - 1 June 81
	6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Robert W. McLaren William D. McFarland	8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0032
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Missouri-Columbia Department of Electrical Engineering Columbia MO 65201	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941836
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRE) Griffiss AFB NY 13441	12. REPORT DATE December 1981
	13. NUMBER OF PAGES 208
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Frederick W. Rahrig (IRRE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing
Feature Extraction
High Speed Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Advances in solid state technology have created vast amounts of digital information for image processing. It has become evident that conventional serial type processors are no longer sufficient to handle and manipulate these vast amounts of data for feature extraction and other image processing routines in general. The results of this report indicate that reductions in processing times of one to two orders of magnitude can be obtained for certain algorithms when implemented on special machine architectures.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The increasing capability of specialized digital processors has led to the possibility of implementing a large number of image processing functions in near real time.

TABLE OF CONTENTS

SECTION Page

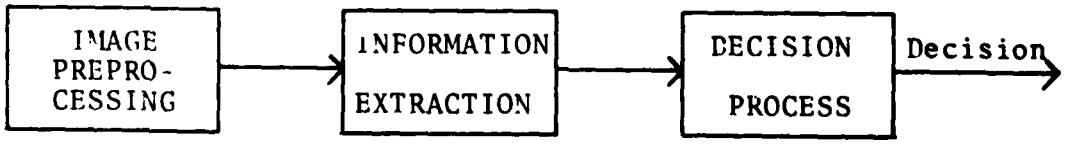
SECTION Page

I. Introduction

1.1 Digital Image Feature Extraction and Processing

Digital image processing, the manipulation of images by computer, is a relatively recent development which has received an ever increasing amount of attention in terms of techniques applied, special processors, and the range of applications. Examples of this include medical diagnosis (x-rays or computer-aided tomography), aerial surveillance (agriculture, forestry, and land-use planning), such as LANDSAT, and military (navigation, target evaluation, mapping, and the like). Digital imagery contains a great amount of information or features whose extraction and processing can be useful in serving many different applications including those indicated above. The high-speed automated processing of imagery by computer is a crucial factor in the effective implementation of an advanced computer image exploitation facility. The increasing capability of digital computers has led to the possibility of implementing a large number of image processing functions in near "real-time," a result which is essential to establishing a near-production type facility. Digital image processing hardware and software are being developed and used by the government, industry, and universities. A significant government application is found in the military- surveillance, terrain identification, and defense mapping.

Digital image processing refers to any process or procedure which is applied by a digital computer to an image in digital form (sampled and quantitized), regardless of the source. A basic system example is illustrated in Fig. 1.1 below:



The overall exploitation procedure consists of image preprocessing, information extraction, and image decision making. Image preprocessing in a narrower interpretation consists of techniques

for improving or enhancing the image for information extraction or for the display of an image to a human observer. These techniques include the use of transforms for filtering, restoration of images, and image segmentation that precede further processing. Preprocessing can also include image "normalization" such as histogram modification, geometric corrections, control of scale and resolution, and image combining (e.g., subtraction or correlation), as well as techniques for noise supression and data clustering.

Another significant step or stage in image processing and exploitation is feature extraction. The objectives and particular techniques applied in this stage often overlap those of preprocessing, depending upon the information extracted. Feature extraction can be viewed as an interpretation of the image in terms of specified features. These features include shapes, boundaries, edges, textures, and the like [1] - [17]. The extraction or detection of such features is often considered to be a part of an image preprocessing stage in image processing which leads to another image preprocessing step called segmentation [18] - [21]. The relative predominance and subsequent interpretation or significance of given features is highly dependent on the type of imagery. This imagery arises from such sources as LANDSAT, aerial photography, FLIR, and RADAR, some of which can be black and white or color. There has been a great deal of work done on the feature extraction problem as a part of the preprocessing stage [22] - [24]. This work includes that on thresholding [25], [26], edge or boundary following [3], [27], shape detection [16], [28], relaxation labeling [29] - [31], and other approaches to segmentation [32], [33]. Other preprocessing techniques include "averaging" and the application of various transforms [34], [35].

In an overall automated image exploitation facility, there is an emphasis on factors in addition to those illustrated in Fig. 1.1, which are significant to the efficiency and effective-

ness of the overall system. These factors consist of image storage and retrieval, image representation and display, image information manipulation, user interaction, and image generation; Fig. 1.2 illustrates these additional factors.

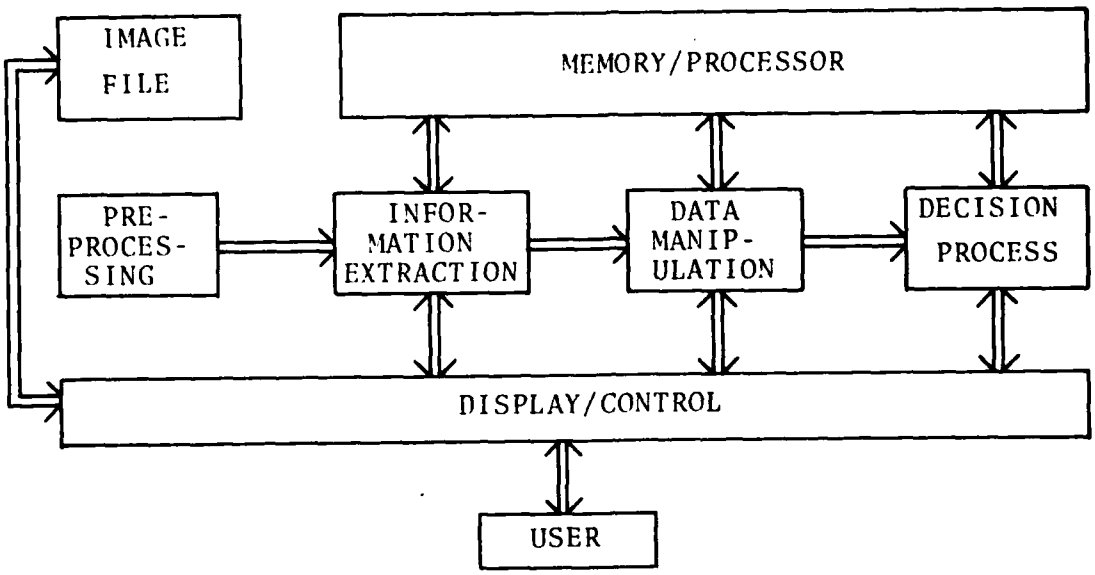


Fig. 1.2 Illustration of a More Complete Image Processing System

Particular configurations of such a system, including hardware, software, displays, information extraction and the decision scheme or process are application dependent. The algorithms and procedures that are applied to implement various image processing techniques are a function of the objectives of the image processing system user.

In particular, there has been a set of algorithms developed under the OLPARS program at RADC, although other projects or systems at RADC were using algorithms similar to these (in the AFES system, for example). These algorithms include histogram modification, averaging, boundary inclosure, enhancement and other feature extraction and processing techniques. These techniques are well-developed and have proven useful as applied to various types of Imagery. These software techniques serve as

a basis for more sophisticated image processing built around a PDP 11/70. This system represents a near production system which handles a higher throughput for various types of imagery. This system has increased speed, memory size and terminal sophistication than previous systems. Of interest here is the processing speed of image feature extraction and processing techniques and its sensitivity to certain computer architectures. The capabilities of the basic AFES configuration to handle higher throughput rates can be significantly expanded by reducing the computation times associated with the implementation of image feature extraction and processing techniques.

1.2 Project Objectives

The objective of this project is to demonstrate the increased computational capability of special computer architectures with application to selected algorithms that would demonstrate a speed-up that would hopefully imply increased throughput. In particular, the machine to demonstrate the proposed speed-up is a STARAN array processor. This machine was chosen because: 1) RADC has one that is operational and thus has an interest in it, and 2) the accessibility of a STARAN to the UMC researchers at Goodyear Aerospace along with expert help in its use from consultants at Goodyear. The idea is to increase computational efficiency and effectiveness through the use of this special image processing architecture. In particular, the STARAN serves as a representative of a special architecture amenable to speeding-up certain image feature extraction and processing techniques. The main thrust of this project is to provide results to support the decision to add or utilize a special architecture in an image processing facility. That is, rather than proposing a specific system configuration, the results of this project serve as a demonstration that a special computer architecture can effect a significant increase in computational speed and thus throughput when applied to selected algorithms for image feature extraction and processing.

1.3 Project Resources

The successful completion of the objectives of this project has depended upon the availability of specific resources. One of these resources is represented by computers; the project employed the use of a PDP 11/50 and a PDP 11/34 located at UMC and the STARAN facility, including the PDP 11/20 host computer located at Goodyear Aerospace in Akron, Ohio. Both computer systems at UMC are RK05 disc-based systems as was the system at Goodyear. All facilities had provision for hard-copy output and a high-resolution display unit (Comtal or Ramtek). Another resource has been manpower. This project has involved the part-time efforts of two faculty members and the part-time help of one to two research assistants (graduate students), depending on their availability over the span of the project. Finally, there was a subcontract with Goodyear Aerospace to provide us with two resources: 1) use of the STARAN facility for program debugging, testing and validation, and 2) personnel or consultants to provide help with the use of the STARAN facility and the actual set-up and demonstration of STARAN operation using programs developed at UMC.

1.4 Project Overview

The objective in seeking to speed-up image feature extraction and processing techniques, rather than just establishing faster processing (especially by sacrificing information content or accuracy), emphasizes techniques to significantly increase the throughput in a total image processing system, especially for a near production system for image interpretation and evaluation. The STARAN array processor represents the means for implementing such a speed-up.

The software to implement selected image processing algorithms on the STARAN array processor were developed by the UMC research team. Two computers at UMC were used for this purpose. Software was developed but could not be debugged or tested at UMC. The role of Goodyear Aerospace was to provide expertise on

the use of the STARAN and provide us time on the STARAN for debugging, testing, and evaluation of programs, along with a hard copy of the results and the display of appropriate images. The research team at UMC and colleagues at Goodyear Aerospace cooperated closely to achieve the goals of this project. This cooperation assumed several forms: 1) several visits of the UMC research team to Goodyear for discussion and use of the STARAN, 2) telephone conversations with colleagues at Goodyear, and 3) a remote terminal hook-up from the STARAN facility at Goodyear to a terminal at UMC via a phone line. After debugging and testing the software for selected algorithms, the results of run times, including the input/output, and display results were obtained with the use of the STARAN facility. Original images and images processed at the RADC STARAN facility would be used for a demonstration of part of the results obtained on this project.

For the most part, software for implementation on the STARAN was developed by research assistants at UMC supervised by faculty members. One of the longer-term or expected outcomes of the work described here is a more general one than using a particular computer architecture and applying or utilizing its speed for application to selected algorithms. Although the processor is a particular one, and the degree of computational speed-up depends on the amenability of the chosen algorithms to speed-up by that processor, the objective is to use the results obtained as a basis for predicting or recommending a more thorough study of special processors for increased throughput (at least an order of magnitude).

II. Problem Formulation

2.1 Image Processing and the Need for Faster Computation

The effective and efficient operation of a near production or image processing facility involves a number of overlapping or interdependent tasks. These tasks include image storage and

retrieval, image enhancement, feature extraction, image recognition, and user display functions (interactive mode). The problem is compounded by the following factors: 1) increased resolution or picture size, say 1024 x 1024 or larger, 2) increased gray-level quantization, 3) complex algorithms involving large image segments, 4) multiple images of the same target- MSS or color, for example, and 5) rapid image handling for near real-time interaction by a user at a display. For example, for a large resolution image, say 4000 x 4000 pixels and 8 bits/pixel, the storage requirements are almost 10^8 bits/image. Then, for say 1000 images, one is considering about 10^{11} bits, which exceeds the capacity range of most current mass storage facilities, at least for near real-time access. This project addresses only a segment of the overall throughput problem- the processing of images, image by image by high-speed image feature extraction and processing techniques. If the computer architecture that provides the speed-up in computation time is coupled or matched with fast I/O transfers along with image compression techniques, the system throughput will significantly improve. At the image feature extraction and processing level, it is expected that these methods will speed-up processing time by at least an order of magnitude. The first problem is to select the image feature extraction and processing techniques to match the chosen computer architecture in order to realize its full capacity.

2.2 Justification for the Selected Algorithms

The variety of image feature extraction and processing techniques covers a wide range. Examples include segmentation, texture identification, noise reduction, thresholding, edge enhancement, image transformation, and the like. Among the choices of algorithms are ones with the common characteristics of being "useful" and at the same time amenable to speed-up through the use of special computer architectures. One can classify the techniques of interest here into two broad categories: 1) feature extraction, and 2) image processing (enhancement). Feature

extraction techniques are concerned with the extraction of specific information from the entire image or from particular sections of it, such as edges, particular shaped objects, lines, pixel statistics (mean, variance, or texture). The main purpose for feature extraction is usually preparation for image recognition. Techniques associated with image preprocessing, in contrast, are applied so as to improve the image, preparing it for feature extraction. Examples of this include histogram modification, noise reduction or elimination, and edge sharpening. Another purpose of such preprocessing is to enhance the user display, such as on-line image manipulation (preprocessing techniques can be called-up), manuscript generation, and image compression and storage. Often the distinction of these two types of processing is not clear, such as with the application of transforms (Fourier, Walsch, etc.), which could precede filtering and then enhancement, or the spectrum could be used to generate a set of features.

Essentially, there were two algorithms which were chosen for implementation on the STARAN array processor. These are: 1) an edge detection (gradient) method, PTEDGE, for detecting or outlining edges, and 2) several techniques designed for image noise reduction or removal: a) MODAL, a technique for pixel replacement, b) ODDPX, another pixel replacement technique, and c) SIMNB, a noise removal technique. Each of these techniques itself involves several subroutines in the implementation. These techniques were developed and executed on a conventional (serial) machine (PDP 11/45); this is reported on in [36]. However, in that study, it appears that the actual application of the techniques to sample imagery was not done, or at least no "before and after" imagery were presented. Thus, the two selected techniques involve: 1) an edge detection or sharpening method, and 2) several variations of noise reduction or removal methods. Available software for these programs already existed in assembly language.

All of these computer-oriented techniques are local tech-

niques; that it, the operations performed on or applied to individual pixels depend on the gray levels of adjacent or immediate surrounding (neighborhood) pixels. Thus, the identification and modification of the gray level of a pixel are made on the basis of the gray levels of neighboring pixels. Being "local" techniques, they are highly amenable to parallel processing; this means that each small area of an entire image can be assigned to a separate processing element for computation. Then, the entire image can be processed in a time close to that what a single area might require for processing. Some computing architectures, such as an array processor, handle the parallel processing in a line-by-line format.

2.3 Justification and Need for Special Computer Architecture

In various places in this report, the term "conventional" or serial machine will be used. This refers to what later is a class of computers called SISD (single instruction stream, single data stream). A "conventional" machine executes the instructions in sequence (serial) and in an image processing application, the processing is implemented pixel-by-pixel, area-by-area, or line-by-line. Thus, depending upon the computations applied to each pixel, area, or line, the total processing time is proportional to the number of pixels, areas or lines to process. This represents a very efficient approach to image processing. Then, the image throughput depends on or is limited by, for a given image (size and number of gray levels), the total sum of computation times associated with a given algorithm. Thus, it is proposed that the parallelism of special computer architectures be taken advantage of for the processing of the image sections. The main advantage of special architectures can occur when it is matched to the algorithm; then, the algorithm can be broken-down and restructured so as to take advantage of the special architecture-parallel tasking, vector operations, and the like. Thus, the key element here is the ability to restructure an existing, chosen algorithm to match a given architecture.

III. Review of Special Computing Architectures

3.1 General Overview

Advances in computer architecture, matching software, circuit design, device fabrication, as well as storage and retrieval techniques have resulted in an increased processing speed in modern digital computers. This, in turn, has provided the basis for a corresponding increase in image processing speeds. Hardware and software architectures are intimately related and difficult to separate. Hardware/software system design to bring about increased image processing speed and improved throughput involves a number of trade-offs, including speed, precision, reliability, flexibility for expansion, modes of operation, ease of use and interactive capabilities. The relative weights and/or constraints imposed on these factors is strongly application dependent. Here, the application is digital image processing, an image being represented by an array of pixels, whose gray levels have been quantitized. The need for high speed is readily apparent when real-time or near real-time image processing is required, especially in a near production environment. One example of near real-time image processing is in digital television [37]. A frame rate of 30/sec. and a horizontal line scan of 63.5 microsec. (with the remaining 13.5 microsec. being used for retrace). With a minimum horizontal resolution of about 500 pixels/line, there would be 50/500 = 100 nanosec. processing time/pixel. This imposes a sever constraint or requirement on the computing architecture.

Many applications do not require the extraordinarily high processing speeds referred to previously. Digital images from weather or reconnaissance satellites may not require 30 processed images/sec. However, time-delays, even for a single frame, may not be acceptable due to rapid movement of storm systems or targets. Short processing times are important even for objects that do not appear to be changing. Image processing requirements

depend on image information content. For example, a single frame of LANDSAT imagery containes 30 x 10^6 bytes of information. The storage of 1000 such images would require some 3 x 10^{11} bits of storage. To handle a high throughput with such a data base, memories could be arranged in a hierarchical or functional manner, a buffer memory could be used (10^8 bytes, say) along with a temporary working space (memory) and a high-speed data transfer rate.

In order to implement digital image processing techniques and satisfy reasonable goals regarding throughput, while being able to manipulate images and implement various sophisticated algorithms in near real-time, it is proposed to utilize advanced computer architectures and organization. These architectures include parallel processing, associative (array) processing, and multiprocessing. Computer architectures can be classified based on the properties of the data and instruction streams. This has led to 4 categories of computer architectures. These are summarized below.

	SD: Single Data Stream	MD: Multiple Data Stream
SI: Single Instruction Stream	Unit Processor	Parallel Process. And Associative Proc.
MI: Multiple Instruction Stream	Pipeline Processor	Multi-processor/ Multicomputer

Table 3.1 A Classification of Generic Processor Architectures

Using this table, one can consider the following architectures:

1) SISD (Single Instruction stream/Single Data stream); uniprocessor (example: IBM 370).
2) MISD (Multiple Instruction stream/Single Data stream); pipeline (example: CDC Star 100).
3) SIMD (Single Instruction stream/Multiple Data stream); paral-

lel or array processor (examples: ILLIAC IV or STARAN).

4. MIMD (Multiple Instruction stream/Multiple Data stream); multiprocessor (example: UNIVAC 1108).
5. SIMD and MIMD combined.

The architecture for a "typical" array or parallel processor is shown in Fig. 3.1 below, (SIMD).

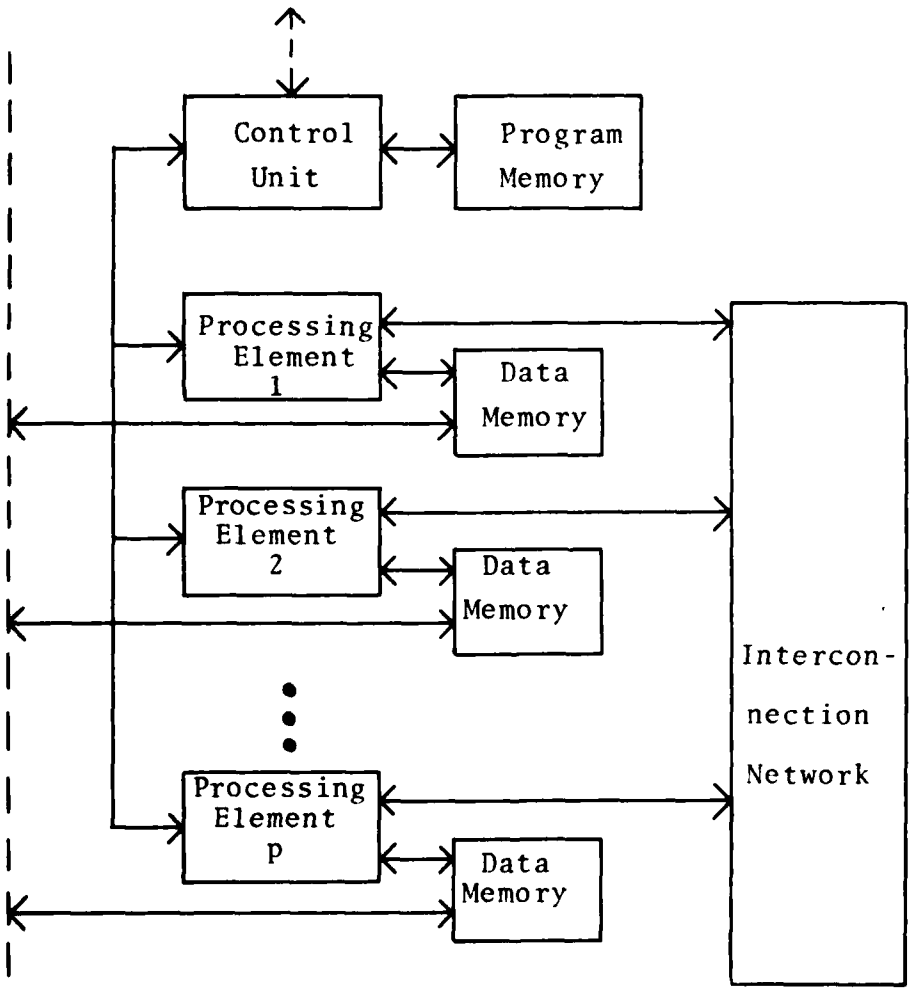


Fig. 3.1 Block Diagram of a Parallel Processor

The diagram in Fig. 3.2 shows a typical associative processor configuration (SIMD).

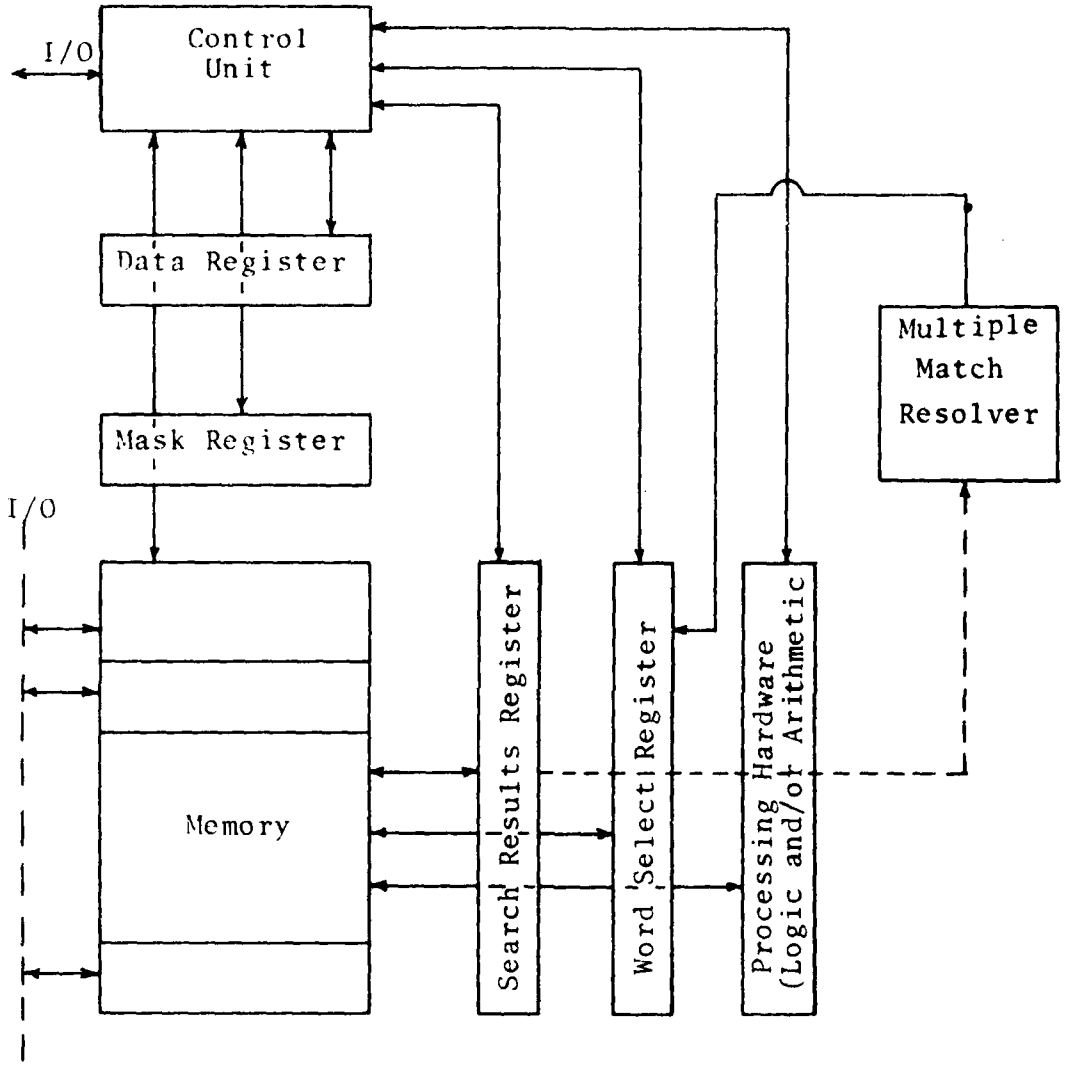


Fig. 3.2 Block Diagram of Associative Processor

Next, three different types of high-speed processors that vary significantly in architecture will be discussed. The ILLIAC IV and AP-120B are introduced in relatively brief form, while the main emphasis is on the STARAN associative processor.

IV. High-Speed Processors

4.1 ILLIAC IV

4.1.1 Architecture

The first high speed processor to be examined is the ILLIAC IV. Its SIMD architecture and topological relationship between processing units result in the ILLIAC IV being classified as an array processor [38]. The design follows that of the SOLOMON computer which was one of the earliest processors designed with a high degree of parallelism [39].

Four types of units make up the ILLIAC IV system configuration as seen in Figure 4.1. The first unit, a Burroughs B6700, functions as a host computer. It provides for user interfacing and program assembly. An I/O controller is the next unit. It provides for data and instruction transfers between mass storage and the arrays. Mass storage is provided by the third unit, a disk file. The disk file allows storage of large amounts of data and instructions. Such storage is particularly useful when the system is supporting many users. The final unit is the array which consists of a control unit (CU) and 64 processing units (PU). Data storage, instruction storage, arithmetic and logic operations are all performed by this unit.

The ILLIAC IV contains four arrays. Each of the arrays is designed to operate independently or in conjunction with each other. This is accomplished by having a controller for each array and also providing a system of interconnections between the control units and the processing units. The result is a variation of all four arrays acting independently, two sets of two arrays acting together or all four arrays acting as one large

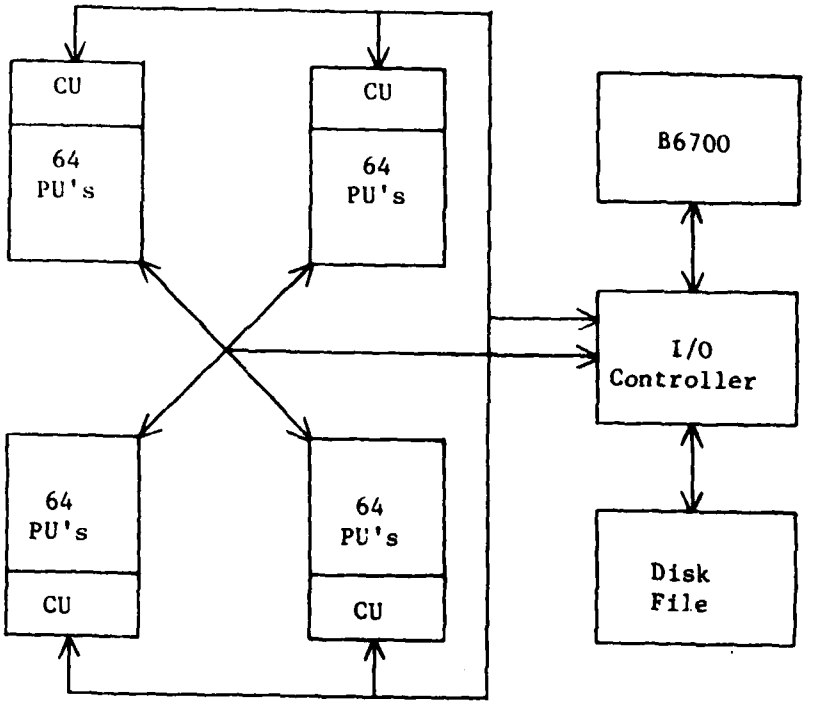
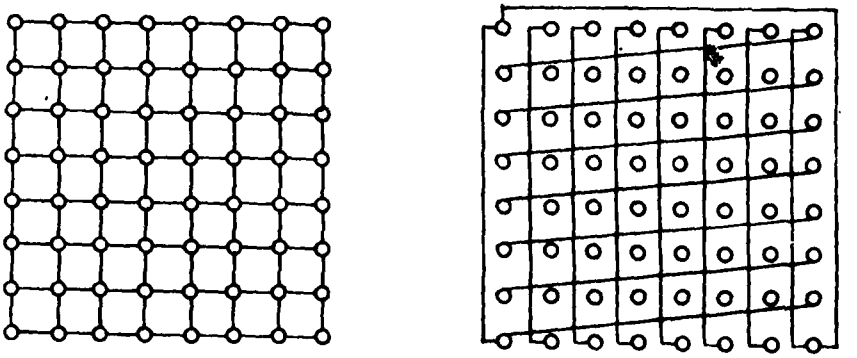


Figure 4.1 ILLIAC IV System Configuration



(a) Adjacent PU's (b) Nonadjacent PU's

Figure 4.2 PU Connections

Sixty-four processing units (PU) are connected together to make up an array. Each PU is connected to four neighbors. This allows the results of one calculation to be used in another calculation without lengthy data transfers. The interconnections can be seen in Figures 4.2 (a) and 4.2 (b). Each processing unit is represented by a small circle and its interconnection with other processing units by a line. Connections shown in both figures exist simultaneously; however, they are shown in separate figures to avoid confusion.

Instructions are both transferred to and in some instances executed by the control unit (CU), Figure 4.3. The instructions, 32 bits long, are read from the processing element memory (PEM), Figure 4.4, in blocks of eight words (16 instructions). They are stored in the instruction buffer which holds up to 128 instructions. Each instruction is then transferred to the advanced instruction station (ADVAST), where it is decoded. This station determines whether the instruction is to be executed in the CU or the PU.

Control unit instructions are executed immediately without being transferred to the final queue (FINQ). A typical example of a control unit instruction would be a jump instruction. This would require a change in the program counter, part of the CU, and would not interfere with the processing units.

If an instruction is to be executed by the processing units, it is transferred from the advance instruction station (ADVAST) to the final queue (FINQ). Here the instruction awaits transmission to the final station (FINST) or the broadcast data and address register. Data intended for use by all or many of the processing units is transferred to each PU from the broadcast data and address register. This is useful for adding a constant value to several or all PU's at the same time. Instructions entering the final station are decoded further and transmitted to the processing units as control signals.

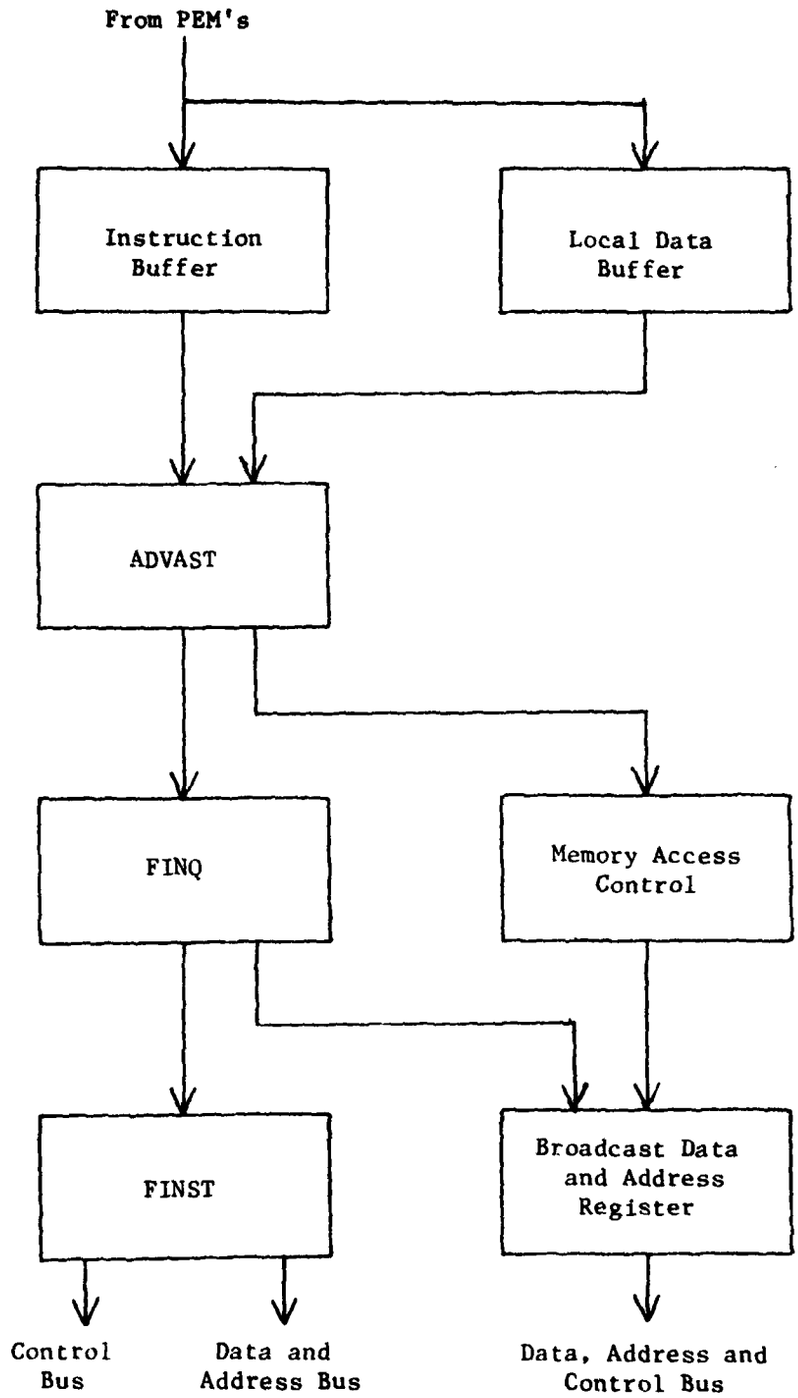


Figure 4.3 ILLIAC IV Control Unit

The control unit has a limited arithmetic capability. This provides for such operations as address indexing. An index value stored in a processing element memory (PEM) location could be transferred to the local data buffer of the CU. The data would then go to the advanced instruction station which contains the arithmetic unit. After this data is added to an address previously sent to the ADVAST from the instruction buffer, the result is transferred to the memory access control. The address is then transferred to the broadcast data and address register where it is used to address a processing element memory.

Each of the processing units (PU) which have been mentioned so far, consists of a processing element (PE) and a processing element memory (PEM) shown in Figure 4.4 [38], [40]. Data and control signals enter the register block (RB) of the PE. Data can be stored in the RB, which contains three storage registers and an accumulator, for use in arithmetic instructions which use the arithmetic logic unit (ALU). Transfers are made between the ALU and the RB to provide the needed feedback for operations such as shifting.

Addressing the PEM is accomplished by transferring the address from the CU or the PEM into the register block. The PU can then take advantage of the ALU to perform address modification. The address can be indexed, if desired, and then transferred to the address register. From there, it is used to address the processing element memory.

Arithmetic operations are performed in the ALU of the processing units. The ALU operates on a 64-bit word which can be divided into two 32-bit or eight 8-bit segments. The 64-bit word and 32-bit segments all have flags which can be used to determine if an operation is to take place. This flag is not operable for the 8-bit segments, which prevents these segments from participating in simultaneous conditional operations. Each ALU can operate independently or in conjunction with neighboring processing units. Since each PU has its own memory, PEM,

simultaneous memory accesses by all PU's can occur.

The ILLIAC IV is capable of performing up to 256 simultaneous arithmetic operations involving 64-bit words. Its architecture results in a machine capable of greatly surpassing sequential computers in a wide variety of operations.

4.1.2 Applications to Image Processing

The ILLIAC IV can be used in a wide variety of image processing applications. The discussion in this section, however, will be limited to three areas. They are table lookup, convolution, and Fourier analysis [41].

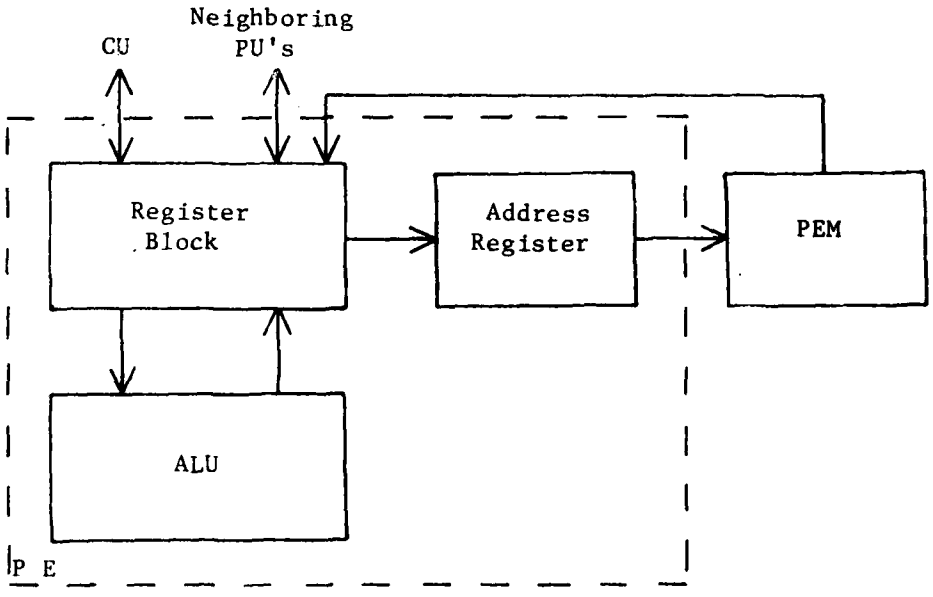


Figure 4.4 ILLIAC IV Processing Unit

Table lookup can be used to threshold an image. This process changes a pixel gray level to a predetermined value when the pixel gray level falls within certain limits [42]. Pixel gray level values can be stored in the processing element memory. Then up to 512 pixels can simultaneously be assigned gray level values based on the limits they fall between.

Convolution can be used for smoothing, filtering, and edge detection [42]. Smoothing results when high frequencies are removed from the image. This will yield an image with high frequency noise filtered out; however, it may also result in blurring. Filtering can be used to enhance high frequencies and produce sharper images. Edge detection may be used to enhance the edges in an image. The convolution of a slowly changing gray level with an impulse results in a faster changing gray level with some overshoot. The result is edge enhancement.

Fourier analysis involves converting signals from the time domain or spatial domain to the frequency domain and back again. This conversion frequently requires complex operations in which the calculated conversion value of one pixel is required to calculate the next pixel. Once the conversion is accomplished, filtering in the frequency domain can result in edge emphasis (high pass) or noise removed (low pass).

4.1.3 Advantages/Disadvantages for Image Processing

The architecture of the ILLIAC IV is highly suited for image processing. Each of the four arrays consists of 64 processing units which yield a total of 256 PU's. Each processing unit uses a 64-bit word which can be separated into two 32-bit or eight 8-bit segments. The segments can each be treated separately, resulting in a total configuration of up to 2,048 eight-bit pixels. This in combination with indexing, high speed memory (250 nanoseconds), high speed processing (400 nanoseconds multiply) and the topological nature of the arrays, are the main advantages in using this computer for image processing.

Disadvantages of the ILLIAC IV include its expense and the flag bits used by the PU's. The large cost, approximately 30 million dollars, is caused by the massive amount of electronics required to perform complex calculations in each processing unit [45]. Every processing unit requires around 10^4 ECL gates and 2,048 words of 250 nanoseconds memory which results in over 2.5 million ECL gates and over 4 M bytes of memory [39]. The ILLIAC IV was constructed in the mid-1960's, using the technology of the 1960's. The result was an extremely costly and large machine.

The flag bits mentioned earlier are a disadvantage because they have limited application. These bits can be used to determine if an operation is to take place in a processing unit. For example, an array may contain some data which must have a constant added to it, while the other data cannot be changed. This can be done by broadcasting data to all processing units in an array and setting the flag bit only on those data words which should have the constant added to them. The flag bit can be set for each 64-bit word in the processing element memory (PEM) or the two 32-bit segments. The flag cannot be set for the eight 8-bit segments which change a possible 2,048 simultaneous operations to 512.

In summary, the ILLIAC IV is a high speed array processor consisting of four arrays. Each array consists of 64 processing units which can be combined to give up to 256 64-bit computations simultaneously. Due to the high cost and mid-1960's technology, few ILLIAC IV's have been constructed. Access to an ILLIAC IV is limited primarily to users of the ARPA network.

4.2 AP-120B

4.2.1 Architecture

The AP-120B is a parallel pipelined processor [43]. Figure 4.5 illustrates the parallel pipelined concept. Processes A, B, and C can take place simultaneously, hence they are called paral-

lel processes. Processes D and E can also occur simultaneously; however, data entering E must first be processed by D. The pipelined concept implies that when data enters E to be processed, new data may also enter D.

Figure 4.6 is a diagram of a typical AP-120B system. Unlike the other computers discussed in this thesis, the AP-120B must have a host computer. The host is a general purpose sequential computer. Its main functions are handling operating system overhead, user interfacing, and performing data manipulation required in preparation for the AP-120B. Data is transferred between computers via DMA cycle stealing and control signals are passed along through the I/O interface.

The architecture of the parallel pipelined processor is shown in Figure 4.7 [44]. Data, instructions, and control signal are received from the host through the I/O interface. Instructions are then stored in the program memory. Data can be stored in the table memory, data pad X, data pad Y, main data memory, or the integer block.

When instructions in the program memory are ready for execution, they are transferred to the control buffer which generates control signals used in the AP-120B. The result can be a combination of up to all ten of the following operations: [43]

1. Floating-point add
2. Floating-point multiply
3. Fetch or store from main data memory
4. Read accumulator
5. Read accumulator
6. Store accumulator
7. Store accumulator
8. Conditional branching
9. Fetch from table memory
10. Integer block

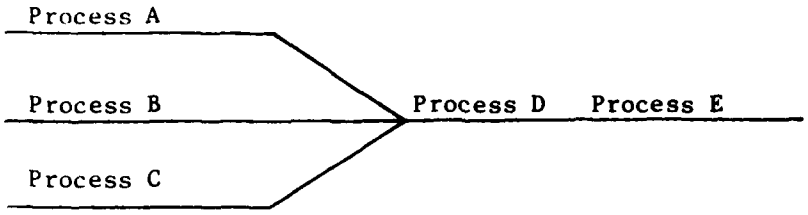


Figure 4.5 Parallel Pipeline Concept

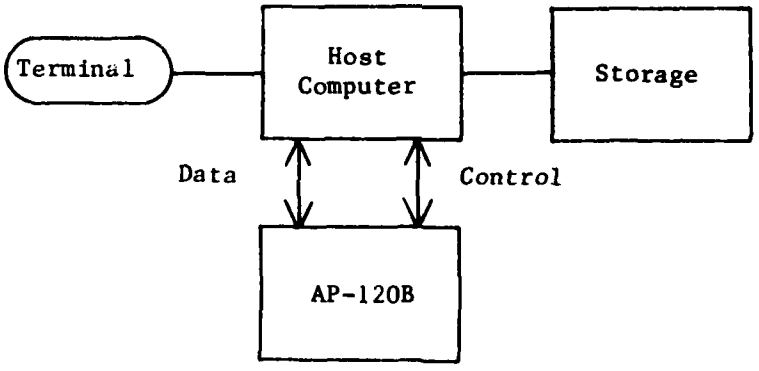


Figure 4.6 Typical AP-120B Configuration

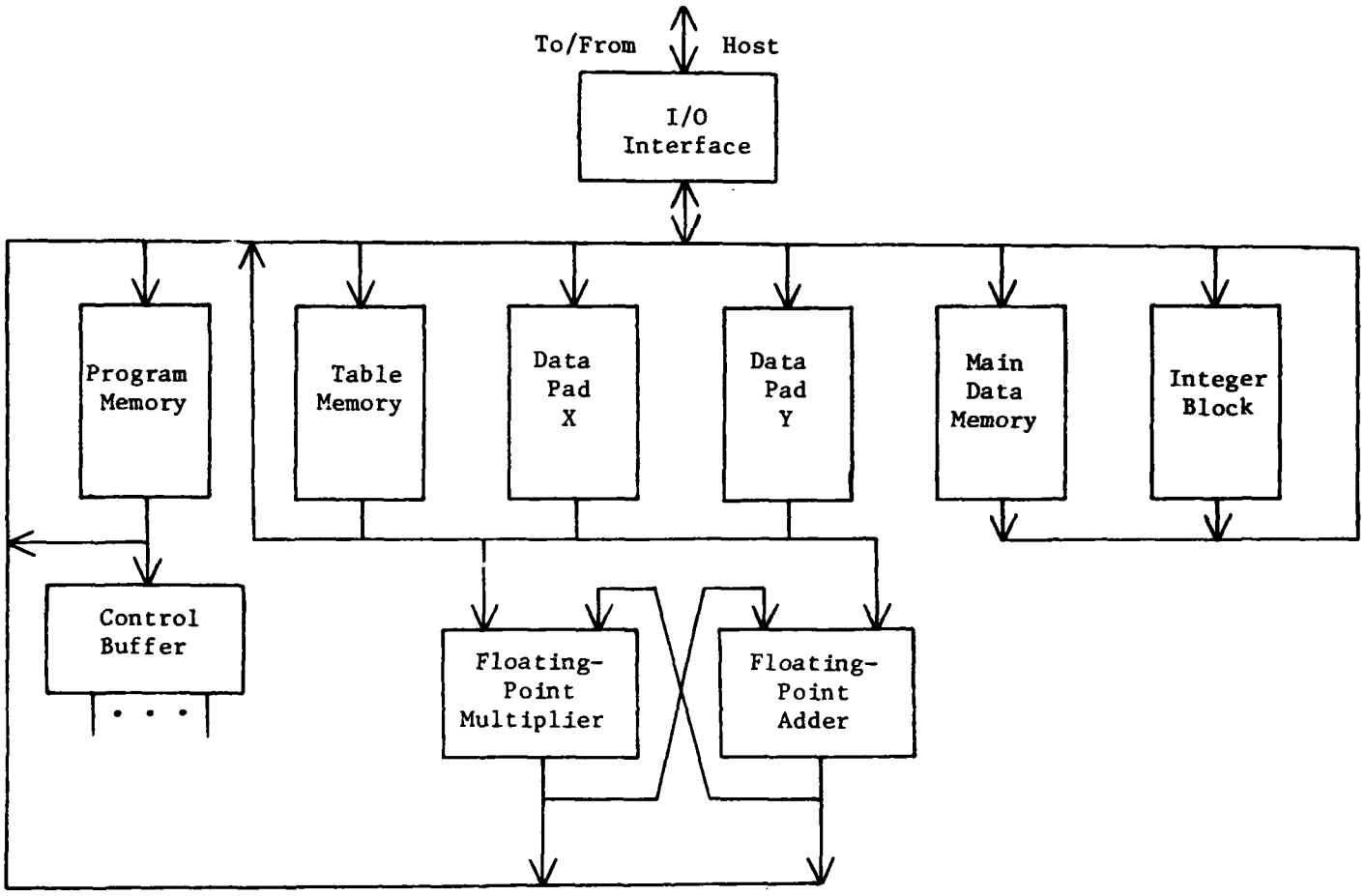


Figure 4.7 AP-120B Architecture

The program memory is constructed with bipolar semiconductor memories. It is available in 1K word increments and expandable up to 4K words. The instruction word size is 64 bits, which permits up to 10 different operations to be executed concurrently.

The table memory uses bipolar technology for both RAM's and ROM's. The ROM's occupy up to 4K words and are used to store constants and sine/cosine tables. The RAM's are available in 1K increments, bringing the total table memory capacity to 64K words. Each data word is 38 bits long to provide increased accuracy in floating-point operations. The table memory can be used to store frequently used constants.

Data pad X and data pad Y are floating-point accumulators. Each pad has 32 floating-point accumulators which are 38 bits long. Data pad X and Y may be concurrently used for source and destination registers. The result is four accumulators (two source and two destination) capable of being accessed in one instruction.

The main data memory consists of up to 512K words of MOS memory. Direct addressing is limited to 64K; however, paging techniques will result in 512K addressable words. This memory is designed to store floating-point numbers which result in its 38-bit word length.

The integer block contains sixteen 16-bit integer registers and an integer arithmetic logic unit. This unit is used for addressing functions and integer arithmetic.

4.2.2 Applications to Image Processing

The AP-120B parallel pipelined processor performs efficiently on calculations. Its high speed floating-point adder and multiplier yield rapid calculations required for such processes as Fourier Analysis, Convolution and Correlation.

The parallel pipelined architecture used by the AP-120B is ideal for many image processing applications. One example is

Fourier Analysis, which can be accomplished with the use of the Fast Fourier Transform. This transform requires many variables to be multiplied. This can be done in parallel. The results of the multiplications must also be added. This must also be done to the next group of variables which results in a pipeline movement of data.

4.2.3 Advantages/Disadvantages for Image Processing

The parallel pipelined architecture provides a large speed increase over traditional sequential machines. Fourier Analysis is accomplished in less time with the AP-120B than with general purpose computers. Performing a 512 x 512 Fast Fourier Transform of an image can be accomplished in 1.55 seconds on the AP-120B whereas a general purpose computer may require in excess of 30 minutes [43].

The AP-120B was designed to be a high speed processor for scientific use. To accomplish this, floating-point hardware became the basis for the processor. Many image processing applications, however, do not require floating-point hardware. Large arrays with primitive arithmetic units would fit these applications better. Thresholding an image could be dome much more rapidly on a computer capable of operating on a large number of pixels at one time than it could be done on the AP-120B.

The AP-120B appears to be the most cost-effective of the three processors discussed in this thesis. Its price is in the one hundred thousand dollar range and its speed is around 3.5 million floating-point operations per second [45].

In summary, the AP-120B is a high-performance, relatively low-cost parallel pipeline processor. It is designed to be connected to a host computer which controls the overall system. The AP-120B became commercially available in the mid-1970's. As a result of the use of newer technology, MSI, LSI and higher speed memories, and the AP-120 architecture, its price and performance have resulted in many units being sold commercially.

Another variation on the AP-120B is a later model by the same manufacturer, the FP-100. This machine does not have the same capabilities, but is more cost-effective and is well matched to host machines such as a PDP 11/23.

4.3 Other Architecture

Other examples of special machines for picture processing which are much faster than conventional machines are the CLIP series, PPM and PICAP. The CDC Flexible Processor and the TOSPICS are two additional examples of more powerful machines, which are similar in their treatment of image processing tasks. A feature of some of these more powerful machines is having a special type of memory, CAM (content addressable memory [51]).

Special purpose computer architecture for digital image processing can be partitioned into two broad classes, bit-plane processing and distributed processing. The bit-plane approach uses Boolean operators as processors on primarily binary images. The distributed computing approach appears to have more computational capability. Most of the existing machines designed for parallel and array processing have the disadvantage of not being "reconfigurable," whereas a variety of digital image processing tasks would greatly benefit from this feature. A multi-processor configuration should be considered; it may be useful to combine the capabilities of both parallelism and array processing. Four principal areas where system performance can be improved for specific applications are: 1) devices and circuits, 2) system architecture, 3) system organization, and 4) system software. Performance characteristics include throughput, flexibility, availability, and reliability. The essential characteristics of a multiprocessor are as follows:

1) contains two or more processors of approximately comparable capabilities
2) all processors share access to common memory
3) all processors share access to input/output channels, control units and devices

4) entire system is controlled by one operating system providing interaction among processors and their programs at the job, task, step, and data set element levels

A key to classifying such structures is the interconnection subsystem- its topology and operations. Three organizations of this subsystem are common: 1) time-shared on common bus, 2) cross-bar switch matrix, and 3) multiport memories. A cross-bar configuration must be capable of resolving conflict situations. The essential structure of a multi-processor system consists of a host computer such as a PDP 11/70 or 11/780, multiple processors, shared memory, local memories, a mass storage memory, inter-processor connections to link processors, memory and I/O, and multiport memories. A basic, but general multi-processor organization is illustrated in Fig. 4.8.

The RCA 215 is an example of a cross-bar multiprocessor, while the Univac 1108 is an example of a multiport memory multiprocessor. The concept and use of multiport memories are represented at the chip level by special processors, such as the Intel 2920 and the AMD S2811.

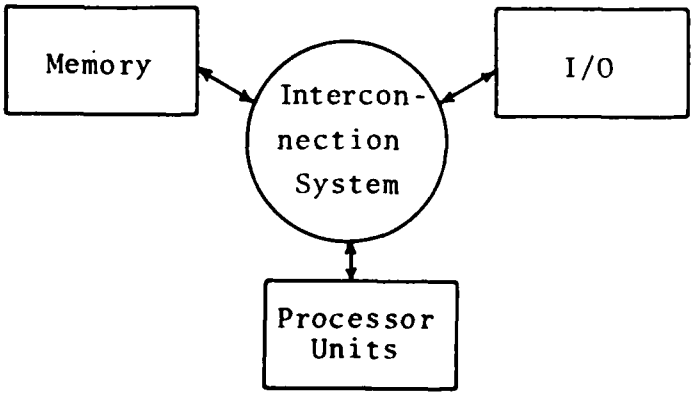


Fig. 4.8 Basic Multiprocessor Organization

Another approach to special architecture for high-speed image processing is the use of bit-slice architecture. Here, the computational tasks are separated from the control and memory tasks. Special chips are designed as the computational units 2-bit, 4-bit, or 8-bit (one byte) slices; they are designed with high-speed technology devices (bipolar Schotky or ECL). These high-speed computational units are then coupled together for form n-bits (8, 12, 16, 32, as required). Computational tasks are performed by special programs (microprogrammed) stored in a ROM (Control ROM). A fast, common memory is used for storing the image processing algorithms. Register to register transfer times of data within these devices are in the nanosec. range. Byte slice devices are now available. The advances of this approach include the high speed characteristics of the bit or byte slice devices and the ease of combining the basic units to form larger combinations as required.

4.4 The STARAN

4.4.1 Introduction

One of the special computer architectures emphasized here in regard to speeding-up image feature extraction and processing techniques is the STARAN. It is emphasized here because this machine was used to demonstrate the speed-up possible with an array-type computer architecture.

4.4.2 Architecture

The STARAN associateive processor (AP) is a parallel processor designed to provide efficient computations of parallel operations. Its associative architecture is derived from the content addressable arrays. In its normal configuration, Figure 4.9, STARAN is connected to a sequential controller. This controller provides a method to load AP programs, a user interface with the AP, a program assembly and debug capability, and control over the AP [46]. The host computer can provide overall system control for time sharing operation.

The STARAN, on which the work for this project was implemented, is located at the Goodyear Aerospace Corporation (GAC) in Akron, Ohio. A PDP 11/20 is used as the sequential controller or host computer at GAC. Mass storage is provided by two DEC RK05 disk drives and two tape drives. The user interface is a DECwriter. A tape drive is employed for the imagery input, while a COMTAL display is used for the imagery output. Later on in the project period, a remote link was established from the STARAN facility at GAC to a remote terminal at the University via a telephone link.

STARAN architecture consists of a control memory unit (CMU) and associative arrays, Fig. 4.10. The minimum configuration requires the CMU and one array. The maximum configuration would consist of up to 32 arrays; however, the STARAN at Goodyear Aerospace Corporation, consists of only two arrays.

The control memory unit interfaces the sequential controller to the associative arrays. Data and instructions are normally transferred from the sequential controller storage devices to the CMU where they are stored. Data can be moved through a 32-bit common register in the CMU to the associative arrays for processing and then back to the CMU. Instructions are executed in the CMU.

The associative arrays consist of a multidimensional access (MDA) memory, response store registers, and a shift network. These units are used for storage of the 256 256-bit words, arithmetic and logic operations, and flags to allow content addressability. Parallel I/O directly into and out of the arrays is also available in some STARAN associative processors.

A more detailed description of the STARAN architecture is presented in Section 5.1.

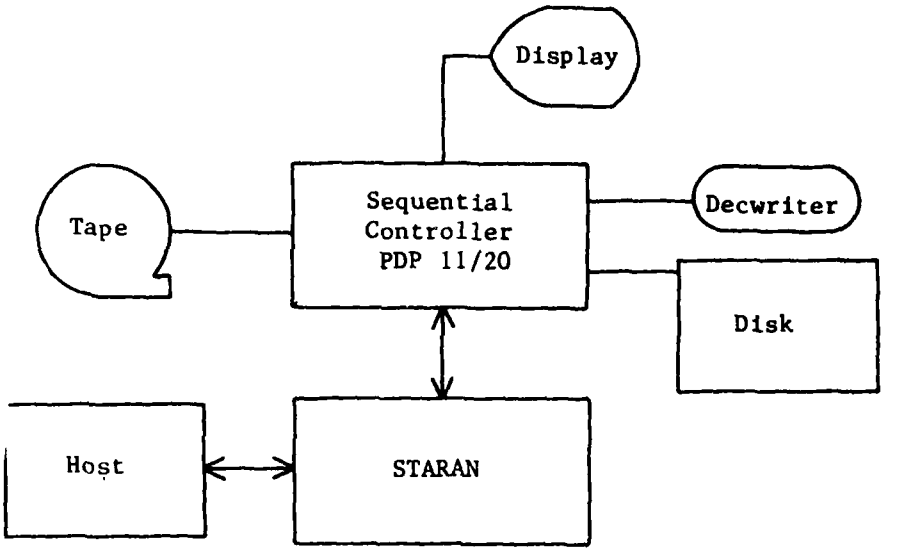


Figure 4.9 STARAN System Configuration

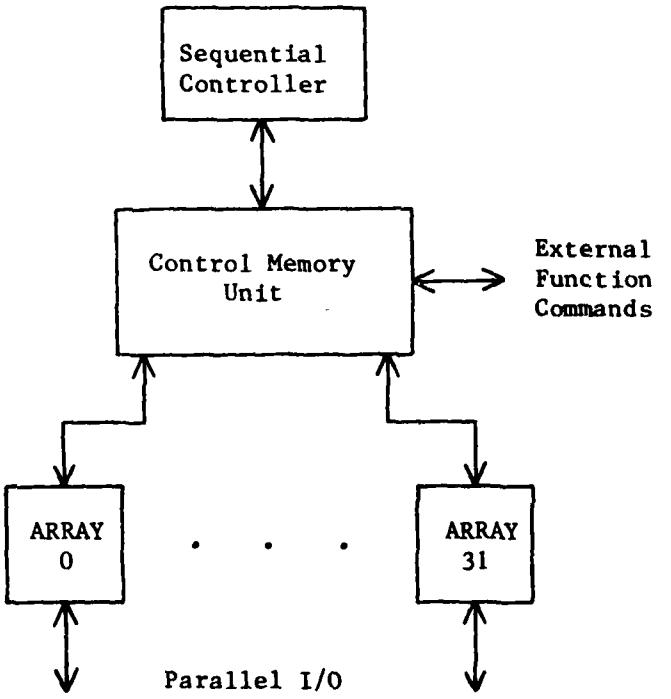


Figure 4.10 STARAN Architecture

4.4.3 Applications to Image Processing

The STARAN associative processor has proven to be a powerful computer capable of handling many applications. It has been used extensively by the Rome Air Development Center to investigate its usefulness in such areas as Advanced Warning and Control Systems (AWACS) tracking systems and Air Traffic Control systems [50]. For image processing application, STARAN is particularly useful in areas requiring manipulations of pixels. Examples would be histograms, convolution, noise removal, and thresholding. Such techniques can be carried out without the use of floating-point numbers for which the STARAN is not ideally suited.

Examining the application of STARAN to producing a histrogram will demonstrate the power of the associative arrays. Since an array consists of 256 words, it can hold up to 256 pixels, storing only one pixel per word. Two arrays will hold up to 512 pixels which allows simultaneous comparisons of all 512 pixels with a specific pixel. The number of matches can be determined, which results in a pixel count for the histogram. Only 512 such operations are required to complete the histogram of a 512 x 512 image. On a sequential computer, 262, 144 such operations would be required.

4.4.4 Advantages/Disadvantages for Image Processing

The two greatest advantages of the STARAN are the large arrays and the powerful associative instruction set. The arrays, 256 x 256 bits, can provide arithmetic operations on at least 256 pixels at the same time. A system which has 32 arrays could, for example, check 8,192 pixels for a particular gray level value all at one time with the execution of one instruction. The associative instruction set allows complex manipulation of data in the MDA memory, response store registers and the common registers.

The disadvantages of STARAN include that of most high performance processors, cost- approximately 3/4 of a million dollars. Several other areas need improvement, also. Data transfers from

the control memory unit to the associative arrays must go through the common register. This is a 32-bit register which forms a bottleneck when trying to transfer large amounts of data to the arrays. An attempt to overcome this was made by providing for parallel I/O directly to into the arrays. This method, however, is not available on all STARAN associative processors.

Associative arrays are constructed with 256 words of memory per array. Each word can perform a primitive arithmetic and logic operation. These operations are not efficient for multiplication and division, requiring additional execution time. Floating-point arithmetic requires additional software which results in a further slowing of execution speeds. This disadvantage could prove significant for algorithms such as the Fast Fourier Transform.

In summary, the STARAN's associative arrays provide a powerful means of processing parallel integer oriented operations. The lack of floating-point hardware does significantly reduce the speed of this machine for many scientific applications.

V. STARAN Associative Processor

5.1 Architecture

A brief discussion of the STARAN architecture was given in Section 4.4.2. This section will expand on that introduction by examining more closely the control memory unit and the associative arrays.

The control memory unit can be divided into 10 areas, Figure 5.1, as follows:

Page 0. This is a memory that uses bipolar technology to achieve fast access. It contains 512 32-bit words. Page memory is used for instruction storage only. This page is used primarily for a library of microprograms that are frequently required in STARAN programs. Page 0 contains hexadecimal addresses 000 through 1FF.

Page 1. This page has the same amount and type of memory as page 0; however, it is intended for STARAN programs about to be executed. It contains hexadecimal addresses 200 through 3FF.

Page 2. This page is functionally identical to Page 1. Its address space is hexadecimal 400 through 5FF.

HSDB. The high speed data buffer is a 512 32-bit word bipolar memory. It is intended for data storage requiring frequent access. Its address space is hexadecimal 600 through 7FF.

Bulk Core Memory. This memory is nonvolatile core which is intended for program and data storage. The standard configuration contains 16K 32-bit words occupying hexadecimal addresses 8,000 through BFFF.

Port Switch Logic. This unit acts as a switch to connect the program pager to page 0, 1, or 2 for a memory write operation. It also can connect page 0, 1, or 2 to the AP control or sequential controller for a memory read.

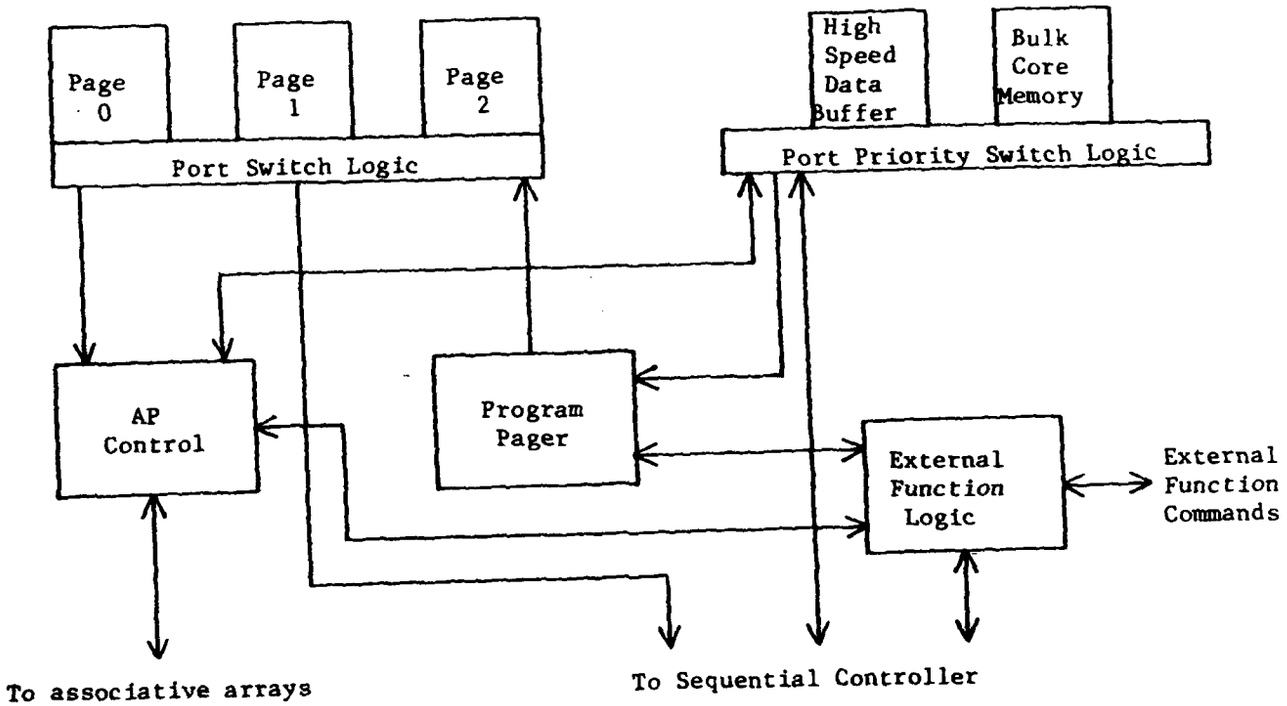


Figure 5.1 Control Memory Unit

Port Priority Switch Logic. This unit determines the priority of the port switches and sends out control signals to allow port activation. It also provides switched paths for the high speed data buffer to the AP control and sequential controller. Switched paths also exist from the bulk core memory to the AP control, program pager and the sequential controller.

AP Control. The associative processor control is designed to control the STARAN arrays. Instructions which are fetched from bulk core memory, page memory, or external logic are decoded and executed in the AP control or the associative arrays, Figure 5.2.

Program Pager. The program pager is connected to the bulk core memory and page memories via port switches. Its function is to load the high speed page memory with programs stored in the lower speed bulk core memory. This allows for large program storage in bulk core and fast program access in page memory. Programs should be written to require a sufficient amount of execution time in page memory to allow one of the unused page memories to be loaded by the program pager.

External Function Logic. This unit transfers control and status lines of some STARAN elements for external use. Resetting and clearing various registers and flags can be accomplished externally. AP control status is also monitored. The AP control unit, Figure 5.2, will be discussed in detail to allow a better understanding of the programs found later in this thesis.

Data enters the AP control from the memory in the control memory unit. It is transferred to one or more of the following registers:

1. Common Register (C)
2. Array Select Register (AS)
3. Field Length Counter (FL1 or FL2)
4. Field Pointer (FP1, FP2, of FP3)
5. Field Pointer Extra (FPE)
6. Block Length Counter (BL)

7. Data Pointer (DP)
8. Program Counter (PC)
9. Interrupt Mask (IMASK)
10. Instruction Register

Data can also be written back to memory from these registers.

Instructions from the page or bulk core memory are transferred to the instruction register (IR). In the IR instructions can be modified by data entering the adder. For example, address modification by a register is possible in the instruction, LR C,0(DP) which loads register C with the contents of the memory location in the CMU addressed by register DP. After any necessary instruction modifications are made, the IR outputs control signals for use within the AP control unit.

The following is a brief description of the functional blocks shown in Figure 5.2.

Bus Logic. This unit provides logic to interface the AP control to the CMU memory.

Instruction Register. It modifies, if necessary, and then decodes the 32-bit instruction words received from CMU memory. It also outputs control signals for use within the AP control unit.

Common Register. It provides a transfer path for 32-bit words between memory and associative arrays.

Array Select Register. Each bit in this register is an associative array enable bit.

FL1, FL2. These field length counters are 8-bit registers that can be decremented and checked for a zero value with a branch instruction.

FP1, FP2, FP3, FPE. The field pointers are 8-bit registers which can be incremented or decremented. They are frequently used to point to a location in the associative array.

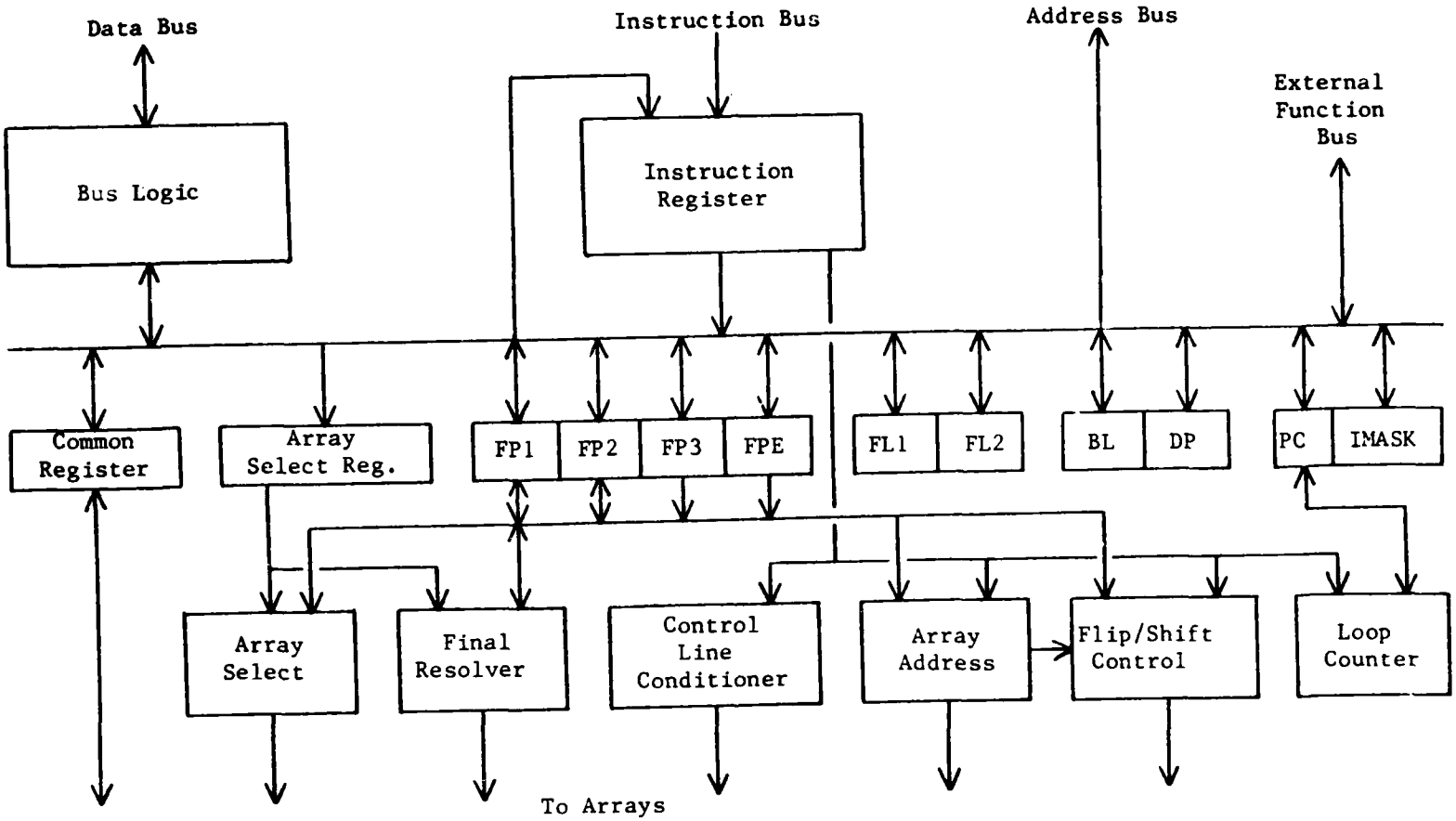


Figure 5.2 AP Control

BL. The block length counter is a 16-bit register which can be decremented.

DP. The data pointer is a 16-bit register which is frequently used in combination with the BL counter to step through a block of data. It can be incremented or decremented.

PC. The program counter is a 16-bit register used to address the next instruction in the CMU memory.

IMASK. This register is four bits long and used to enable interrupts.

Array Select. This register uses either the array select register (AS) or the field pointer FP1, to select which array or arrays are active.

Final Resolver. This register determines the first responder set (Y response store register) and returns the array address to FP1 and the word address to FP2.

Control Line Conditioner. The conditioner generates control signals required to obtain addresses in the final resolver.

Array Address. This unit generates the address mode used in each array, bit column or word.

Flip/Shift Control. This unit generates control signals used to provide multidimensional access of the arrays.

Loop Counter. Two 16-bit registers and a comparator make up this unit. One register holds the starting address of the loop. The other register holds the final address of the loop. When the comparator indicates the PC and the final address register are the same, the starting address is loaded in the PC to continue the loop. The loop is repeated as many times as specified in the instruction.

The STARAN contains from one to 32 arrays. Each of these arrays contains a multidimensional access memory which has 256 words that are 256 bits long. The major units in the arrays are

shown in Figure 5.3.

Control signals, data, and addresses are transferred from the control memory unit to the array control in the associative array. The control signals are used for the resolver (Y response store), the response store registers, the shift network, and the multi-dimensional access (MDA) memory. Addresses are used to locate words, bit columns, or fields in the array.

The response store registers are 256 words long and one bit wide. The M response store register is used for temporary storage and also as a mask to enable array words to participate in an arithmetic, logic, or move instructions. The X register is used for temporary storage. The Y register is used for temporary storage, but it also functions as a resolver. When operating in this mode, the Y register has a bit set corresponding to each word in the MDA memory which meets conditions required by the instruction. For example, the instruction EQC sets each bit in the Y response store register if the specified fields in the common register (C) and the array field are equal. The response store registers are also used to perform arithmetic and logic operations in the array.

The shift network allows the data in a response store register to be shifted and loaded to another response store register or into the MDA memory.

The MDA memory is 256 x 256 bits. It is segmented into fields, words, and bit column addresses as shown in Figure 5.4. The field address shown in this figure is not fixed in length, but can range from one to 256 bits long. STARAN instructions use three types of addresses in performing data manipulations in the array.

5.2 Instruction Set

The STARAN instruction set can be grouped into three areas. The first area is the control memory unit instructions. These instructions correspond to sequential computer instructions in

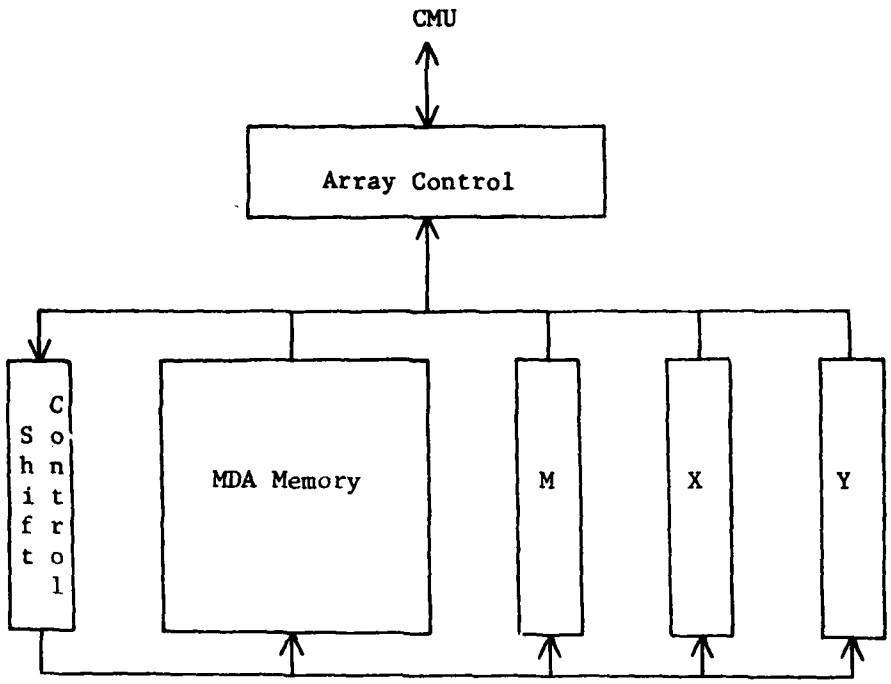


Figure 5.3 Associative Array

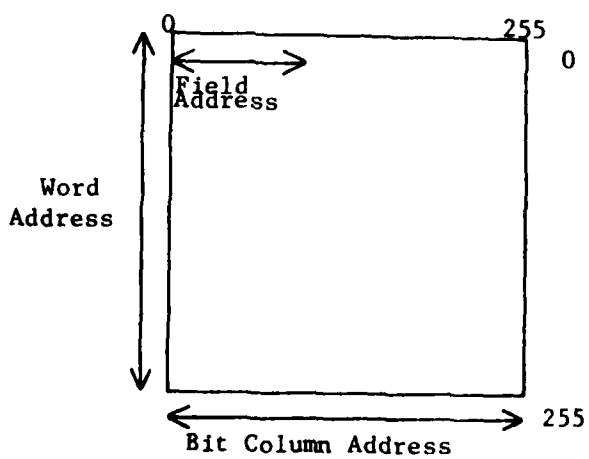


Figure 5.4 Associative Array Addresses

most cases. Branching, register manipulation, and control instructions are included in this group. The second area is the associative array instructions. They are uniquely STARAN instructions which takes advantage of the multidimensional access memory. The final area is the parallel input and output instructions. This area is designed to take advantage of facilities which support high speed parallel I/O to and from the arrays. Parallel I/O instructions were not used in the research for this thesis. They are not supported by the STARAN for which this research has been done and will not be discussed further.

Control memory unit instructions can be grouped into three types. The first type is branch instructions, the second type is register instructions, and the third type is control instructions. The instructions required for the associative array are grouped into six types. These types are branch, load, store, move, searches, and arithmetic.

A brief description is given in Appendix A of most of the instructions used in Appendices B, C, and D [47]. They include both control memory unit instructions and associative array instructions.

The use of several instructions will follow. They are store, arithmetic, search, and load instructions and are listed in Appendix A.

A store instruction moves data from one location in the array to another or from the CMU to the array. If it is desired to move data from the CMU memory to the associative array memory, the SC c,d instruction can be used. First, data must be moved from the CMU memory to the common register via a load register instruction. Next, the SC c,d instruction can be executed.

The SC c,d instruction affects only those arrays that are enabled by the array select register and only those array words which have their mask bit (M response store register) set. The parameter 'c' is used to specify a field in the common register.

This register is 32 bits long and the field can be from 1 to 32 bits long. A field designation of (4,20) indicates that starting at bit number 4, the fifth bit, 20 bits in the common register are used. The parameter 'd' is used to specify a field in the associative array.

An example of this instruction is SC (0,8),(249,8). This instruction moves the first byte of the common register to the last byte in each array word which has its mask bit set.

The arithmetic instruction, ADC a,b,c, will be considered next. If data from the CMU memory is to be added to a field in the array, this instruction can be used. Data will first have to be moved from the CMU memory to the common register. Then the instruction ADC a,b,c will add field 'b' of the common register to field 'a' of the array and store the result in field 'c' of the array. Again the array and array words must be enabled to participate in this instruction.

A search instruction can be used to determine if fields in an array meet certain conditions. The instruction GEC a,b can be used to determine if the contents of field 'a' is greater than field 'b'. Field 'a' is in the common register and field 'b' is in the array. Only arrays that are enabled and words that have their mask bit set participate in the instruction. If the conditions are met, then a bit is set in the Y response store register corresponding to the word in which the conditions were met.

It may be desired to move the contents of the array fields which met the above criteria, back into the CMU. This can be done with the load instruction LCM c,d. First, the link pointer (FP12) is set equal to the array field location which has a Y response store register bit set. This can be done with the STEP instruction [47]. Then instruction LCM is executed which moves field 'd' of the array to field 'c' in the common register. The array field is determined by the contents of FP12 which was determined by the contents of the Y response store register.

5.3 Comparison with Sequential Computer Instruction Set

STARAN has instructions which are both parallel and sequential in nature. The instructions for the control memory unit are sequential and very similar to those found in a typical sequential computer such as the PDP 11/45. The instructions for the associative arrays are parallel in nature and bear little resemblance to those of a sequential computer.

The control memory unit has been seen to have branch instructions and register instructions. These instructions provide little more than the minimum necessary to carry out elementary data manipulation. While the STARAN can use direct, register, and autoindexing address modes, it lacks the power of indirect addressing which is found in many sequential computers. This adds to the weakness of the STARAN instruction set.

An almost total lack of arithmetic and logic instructions further weakens the control memory unit's usefulness. Complementing, both 1's and 2's, incrementing and decrementing is the extent of these instructions. Sequential computers have a wide variety of arithmetic and logic instructions. The PDP 11/45 uses instructions such as ADD, SUB, MUL, and DIV, which result in addition, subtraction, multiplication, and division, respectively [48]. Such functions are not always of a parallel nature and to process them on the STARAN requires the additional step of moving them into the associate array.

The reasons for selecting and using the STARAN for implementing the speed-up of image feature extraction and processing techniques are two-fold. 1) The STARAN is an excellent example of a particular type of high-speed architecture- associative arrays that are content addressable. 2) The STARAN has been available to RADC in either a stand-along mode or through MULTICS, and this can be viewed as providing motivation for the interest at RADC in using special architectures for speed-up. At the same time, a preestablished contact with a colleague at Goodyear

Aerospace led to a joint working relationship between the research team at UMC and colleagues at Goodyear. This relationship included the use of their STARAN computer for programming, debugging, testing, and evaluation, as well as help regarding its use. Thus, the selection of the STARAN to establish the advantages of special computer architectures for speeding-up image processing was a very practical one from the point of view of availability as well as its potential use by RADC. At the same time, however, this project can be viewed as a demonstration project in which the choice of the STARAN is not as important as demonstrating that a special computer architecture can provide a significant speed-up of image feature extraction and processing techniques. This is exactly what this project has established.

VI. Project Description and Procedures

6.1 Achieving the Required Background and Experience

This section briefly describes the preparation and general approach by the UMC research team on this project in order to achieve its objectives. The original research team (who also completed most of the programming work) consisted of two faculty members and two research assistants. All have had experience in image processing and recognition. The two faculty members had worked with other image processing projects involving the processing of medical imagery and LANDSAT imagery, while the research assistants have had a very practical background in software development for image processing as well as in the use of image processing and analysis equipment at UMC. However, this project has presented some special challenges in terms of using a special computer and taking advantage of its architecture for speeding-up image extraction and processing techniques. Therefore, it was necessary to learn the basic structure and programming language (APPLE and MAPLE) so as to be able to restructure programs prepared for conventional or serial machines for execution on the STARAN as well as test and evaluate the programs developed.

Material obtained from Goodyear Aerospace provided background on the STARAN, including history, structure, and operation (programming). With this material and direct help from colleagues at Goodyear Aerospace during a visit there, we were able to prepare programs that would eventually run successfully on the STARAN. For the most part, throughout the project, programs that were developed at UMC were debugged, tested, and evaluated during visits to Goodyear Aerospace by two or more of the UMC research team.

6.2 Selection of Algorithms

As experience was gained in the operation of the STARAN, the UMC project team was also evaluating and reviewing various existing algorithms in order to select two or three that would have a high potential for speed-up when executed on the STARAN. One source of material describes some programs developed for PDP 11/45, but not actually demonstrated with displayed imagery; another source was the OLPAR programs which were successfully used by RADC. The criteria for selection of algorithms were two-fold: 1) to select "useful" algorithms in the sense of commonly used or significant algorithms, and 2) select algorithms that would be amenable to significant speed-up when executed on the STARAN. Two such algorithms were then selected: a) an edge gradient technique, and b) a noice reduction or elimination technique. The next step was to change the form of these algorithms by breaking them down into elemental operations.

6.3 Role of UMC Computing Facility

The University of Missouri-Columbia Image Analysis Laboratory provided one of the main facilities to implement the goals of the project. The equipment essentially consisted of a PDP 11/50 minicomputer with 88 k words of memory and operates under RSX-11M. On-line computer peripherals include a RP03 disk drive, two cartridge disk drives, nine track, 800 bpi tape drive, card reader, line printer, and four CRT terminals. On-line image analysis peripherals include a Spatial Data Computer Eye 108

television digitizer, a Dicommed 50B image dissector scanner, a Ramtex color display (256 x 256 x 12) and a Ramtex black and white display (512 x 512 x 8), a joystick interfaced through the computer eye and a Graf Pen x-y tablet. Also used was a black and white display built around a Data Disc fixed-head per track disk drive, a second Computer EYE, and a Hewlet-Packard x-y plotter. In addition, a PDP 11/34 minicomputer was available for use, with one fixed head and one removable head disk (RK05). Also, the University Computer Network (Amdahl 470/V7 and IBM 3031) was available for larger data processing needs such as measurement selection on large data bases. Images and data can be transferred via a 9-track magnetic tape from the image analysis laboratory to the Computer Network. Fig. 6.1 illustrates the essential equipment available to the UMC research team for carrying out the objectives of this project.

The initial role of the UMC computing facilities was to simply test and display the results of the selected algorithms on particular imagery. This initial study helped to provide a basis for selecting the algorithms by demonstrating their effectiveness and efficiency in achieving their objective on selected imagery. This was accomplished using the original, available form of the selected algorithms- assembly language level. The effectiveness was measured by "before and after" displays, and at the same time, algorithm parameters were varied to determine and optimize their effect. Then, with the algorithms selected, the effect of sensitivity, thresholding, amount and type of noise removed, and the like, could be studied. Another purpose was to prepare (but unable to test) programs in APPLE (or MAPLE) for execution at the Goodyear Aerospace STARAN facility. A related task or function using equipment at UMC was to prepare the developed software in the correct format and media for direct use at Goodyear and eventually at RADC for program execution. At a later point in the project, the sample imagery used for illustrating the selected algorithms included an example image provided by RADC.

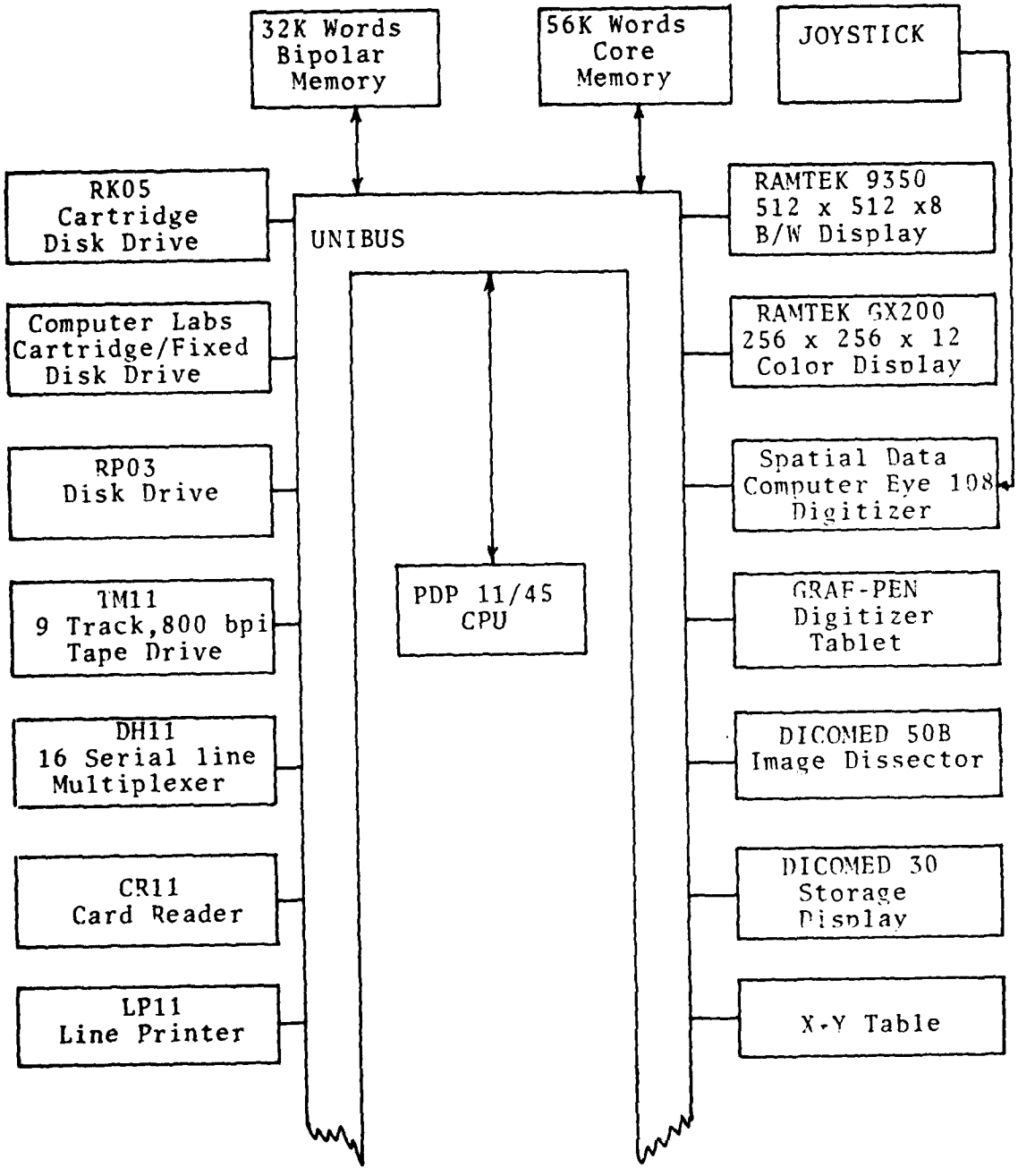


Fig. 6.1 The Image Analysis Laboratory at UMC

Another purpose of the UMC facilities was to evaluate results obtained during visits to Goodyear Aerospace in terms of processed imagery ("before and after") to verify that the required processing and effects of design parameters had taken place, and so that the next visit for using the STARAN could be made more effective. Finally, another purpose of using the facilities at UMC was to take photographs of the selected "before and after" imagery from the display equipment to provide a more viewable and permanent record of results.

6.4 Use of the STARAN Computer at Goodyear Aerospace

Part of the work completed on this project in achieving its objectives was implemented through a subcontract with Goodyear Aerospace Corporation at Akron, Ohio. Frequent contact was maintained between the research team at UMC and colleagues at Goodyear during the development of the programs to be executed on the STARAN. The major purpose in establishing this working relationship with Goodyear Aerospace (regarding the project) was, of course, to learn how to use the STARAN and use the facility for program modifications, testing and evaluation. They provided background material on the STARAN for study early in the project period and prior to the first visit by the UMC research team to the STARAN facility at Goodyear. During the first visit to Goodyear, the research team was provided with further background "lec tures" on the STARAN and was given a demonstration of the use of the STARAN facility. Frequent consultation with colleagues at Goodyear was necessary in regard to basic questions, programming problems, arranging for visits, and setting-up a remote terminal. The second segment of the joint effort concerned the actual use of the STARAN facility. This facility was used extensively during subsequent visits by the UMC research team for program modification, testing, and evaluation. In addition, the facility was also used when a terminal was set up at UMC as a remote terminal with access to the STARAN. Thus, the cooperation of Goodyear Aerospace in providing consulting help and availability of

the STARAN facility was essential for successful achievement of the project goals.

6.5 Image Description

Basically, the imagery used in this project is represented by a 512 x 512 pixel black and white image, each pixel defined by an 8-bit gray-level. Essentially, three types or topics were used for the imagery in order to illustrate the effects of the applied image feature extraction and processing techniques. These were: 1) an example of LANDSAT imagery, 2) a camera lens cap, and 3) an aerial image of a runway provided by RADC. For different purposes, it was necessary that the imagery be available on different media. For processing and for transfer to and from UMC and the STARAN facility, it was most useful to use the RK05 disc system (RL0L and RL02 are now replacing this system). However, for longer-term storage and availability/transfer from one machine or image processing to another not having a common disk system, it is useful to store programs on a nine-track unformatted magnetic tape. The STARAN facility and the UMC computer facility used different displays.

6.6 Procedures for Software Development and Testing

In order to achieve the objectives of this project, the following sequence of steps or procedures was followed:

1) Select algorithms for analysis.
2) Analyze the selected algorithms in detail using flowcharts.
3) Test the selected algorithms on particular images.
4) Vary the design parameters to provide a basis to select usable values.
5) Translate the selected algorithms into assembly level programs for execution on the STARAN (APPLE/MAPLE).
6) Debug and modify, test, and evalue programs on STARAN.
7) Display and document results.

6.7 Basis of Comparison

One of the central objectives of this project is to establish the speed advantage of a special computer architecture over a conventional or serial machine in terms of image feature extrac-

tion and processing. Thus, essentially one can simply consider the "run" times for a given algorithm on the serial machine and on the STARAN. This project shows a significant speed-up of run-times for the STARAN. In general, the run times include I/O time and actual execution time directly associated with the algorithm. With the STARAN, the I/O transfers were relatively slow because the host machine was a serial machine. Thus, it may be more meaningful to separate the I/O times from the algorithm execution times. In this way, the speed advantage of a special architecture in executing a given algorithm would be more apparent. The I/O, in fact, if handled by a conventional machine, would certainly slow-down the through-put of a "near" production system. In general, the solution to this problem would be to use a special host machine to handle the I/O operations campatible with the special processor. For the STARAN, this would require a 256 bit I/O transfer register as an interface. As the results indicate, there is a significant speed-up for implementing or executing the selected algorithms for image feature extraction and processing techniques as compared with conventional machines.

VII. Presentation of Results: Image Noise Reduction

7.1 Modal Technique

7.1.1 Description

The modal technique is a method for removing noise from an image [36]. The process works by examining neighbors of each pixel to determine if the pixel should be replaced. A 3 x 3 neighborhood is used, resulting in eight neighboring pixels and the pixel being considered for replacement, the center pixel. Any pixels that do not have eight neighbors are not replaced. This prevents the edge points of an image from being changed.

When a pixel is being considered for replacement, all neighbors and the center pixel are compared. If two or more pixels have the same gray level value, they are grouped into a mode. The mode contains the gray level value and a frequency

count to indicate how many pixels are in the mode. After all comparisons have been made, the center pixel is replaced with the gray level of the mode with the highest frequency count.

When more than one mode has the same frequency count, the mode is selected by priority. A mode is prioritized according to which pixels are in it. The priorities [illegible] ne 3 x 3 neighborhood from highest to lowest are the center, top left, top middle, top right, middle left, middle right, bottom left, and bottom middle.

If no mode occurs in a neighborhood, the center point is not changed.

7.1.2 STARAN Implementation

A STARAN implementation of the modal technique appears in Appendix B. The following discussion is based on this program. Figure 7.1 is a flowchart of this program and should be referred to when reading this section.

The processing discussed in Section 7.1.1 takes place in the associative arrays. Buffers are set up in arrays 0 and 1 to allow for storage and comparisons. Since the image being processed is 512 x 512, and an array is 256 x 256, two arrays are used to store an image line.

Each pixel is in a separate word of the array as shown in Appendix B. This allows load, store, search, move, or arithmetic instructions to perform simultaneously on an entire line of the image.

Arrays 0 and 1 are loaded with the first, second, and third image lines. Each line has 8-bit pixels so they occupy a field of eight bits. For line one, the field is (0,8) which means the field's most significant bit is in bit column zero, and the field is eight bits long. Line two has a field of (8,8) and line three has a field of (16,8). Since all the instructions requiring inputs from these fields are logical and not arithmetic, a sign bit is not required.

Another buffer used in the array is the compare buffer. This buffer contains all the pixels in the neighborhood and the center pixel. The center pixel is the one from the second line stored in the array. Nine pixels are in the compare buffer which is in field (24,72).

The frequency count buffer is composed of two types of entries. The first entry is the frequency count discussed in Section 7.1.1 and the next is the pixel gray level value. All pixels in the neighborhood except the bottom right pixel are in this buffer, including the center pixel. The buffer contains eight pixels, 64 bits, and frequency count tags, 32 bits, which are in field (96,96).

The final buffer is the output line. It is eight bits long and is in field (192,8).

The first segment in the modal program initialized the input and output devices. This is done by setting up the tran block for the input, mag tape, and the output, COMTAL display.

Image data is read in from the magnetic tape on the PDP 11/20. The data on tape has had every 16-bit half word swapped and every byte swapped from the original image. This results in a byte sequence of 43218765... when 12345678... was the original sequence. The swapping is necessary if STARAN instructions are not used to swap the data input. The STARAN interfaces are configured in such a way that they swap the data; this requires the program swap the data or the input data be swapped to compensate.

The amount of data input to the STARAN memory depends on the record size of the magnetic tape. The value used for this program was 16,384 bytes.

After 16,384 bytes of the image has been input, 32 image lines, buffer pointers are initialized. These pointers indicate when the input buffer, IBUFF, is empty, when the output buffer, OBUFF, is full, and when the last image line has been input, LIF. Data input from the mag tape is stored in the IBUFF and data to

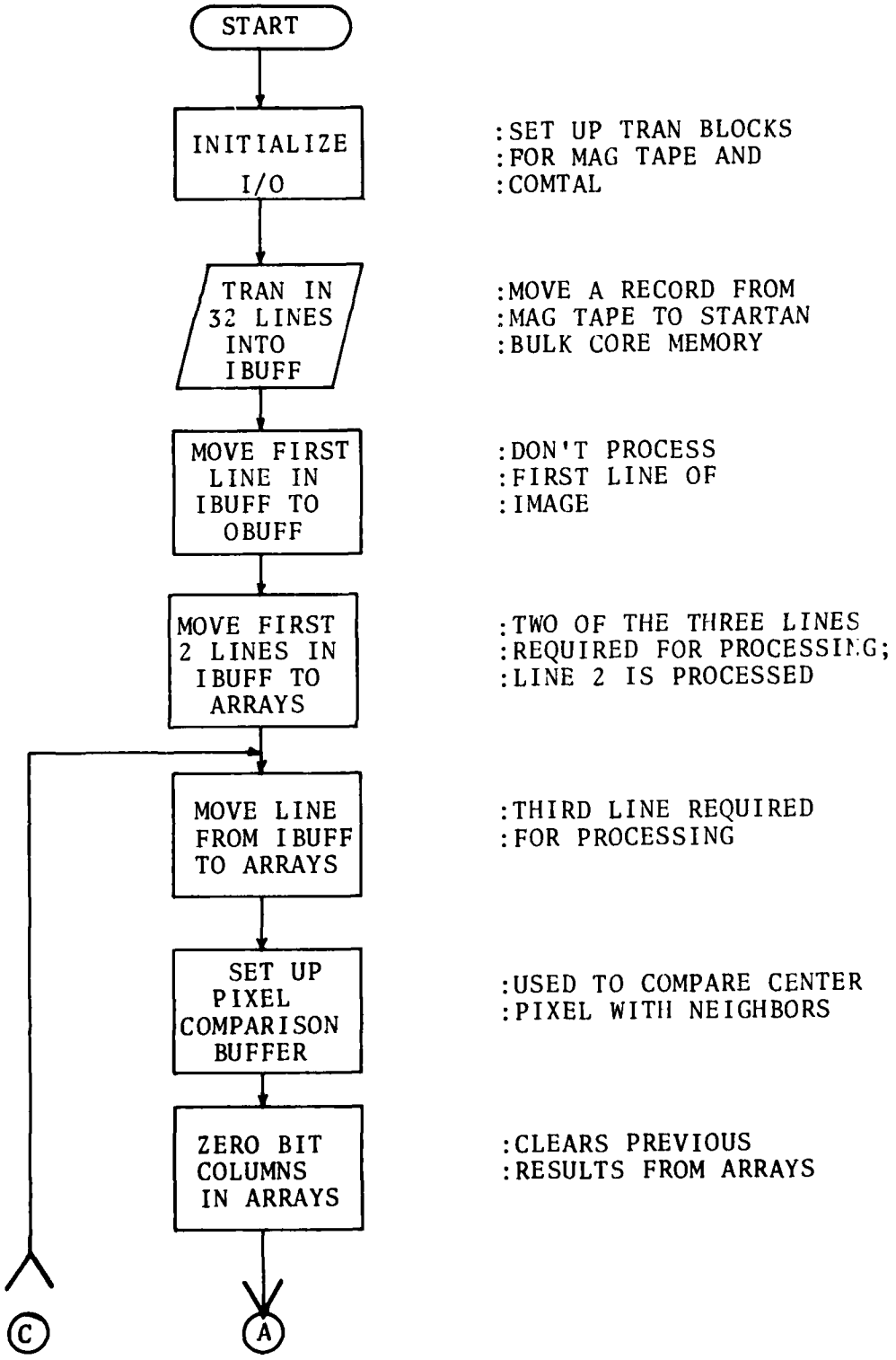


Figure 7.1 MODAL Flowchart

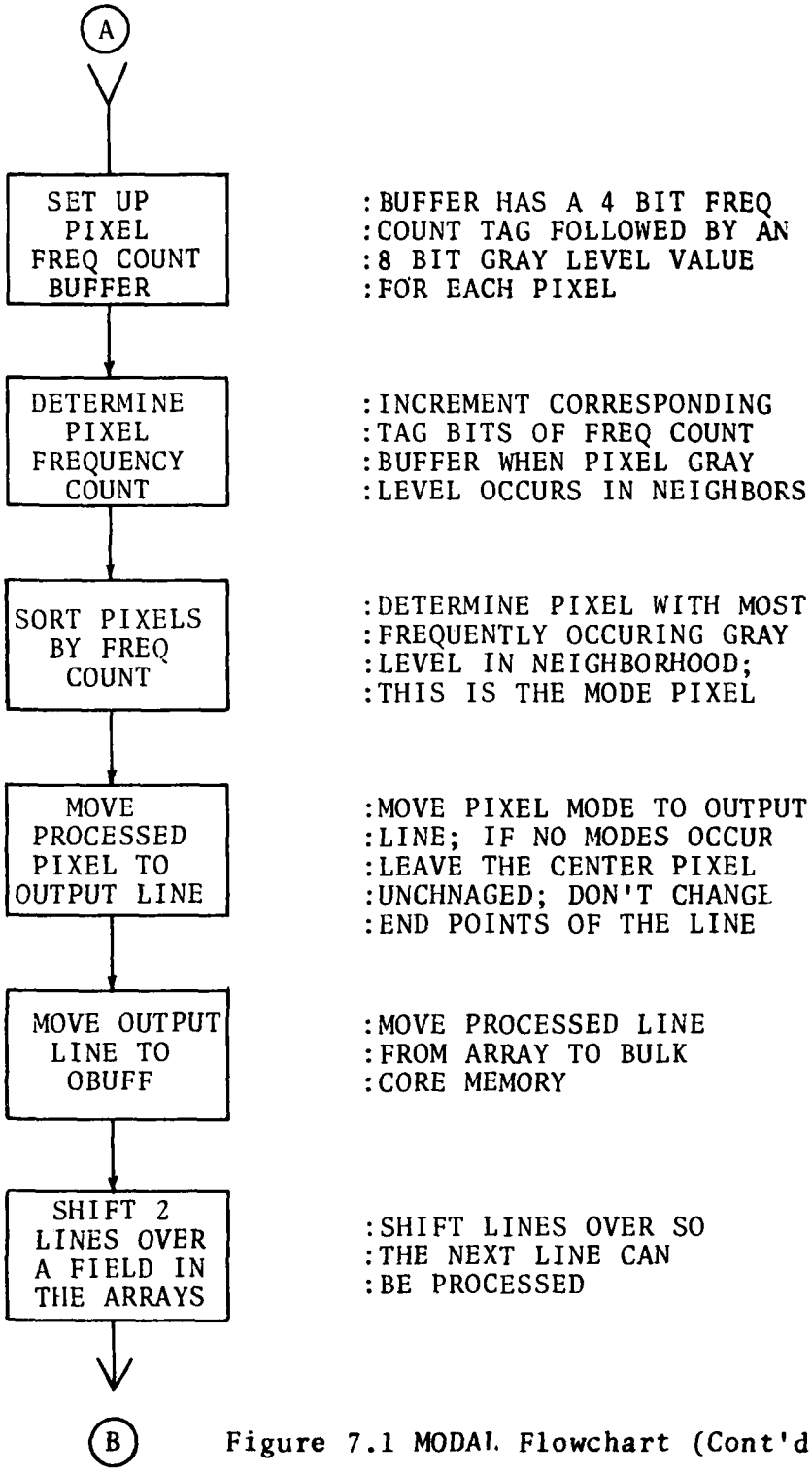


Figure 7.1 MODAL Flowchart (Cont'd)

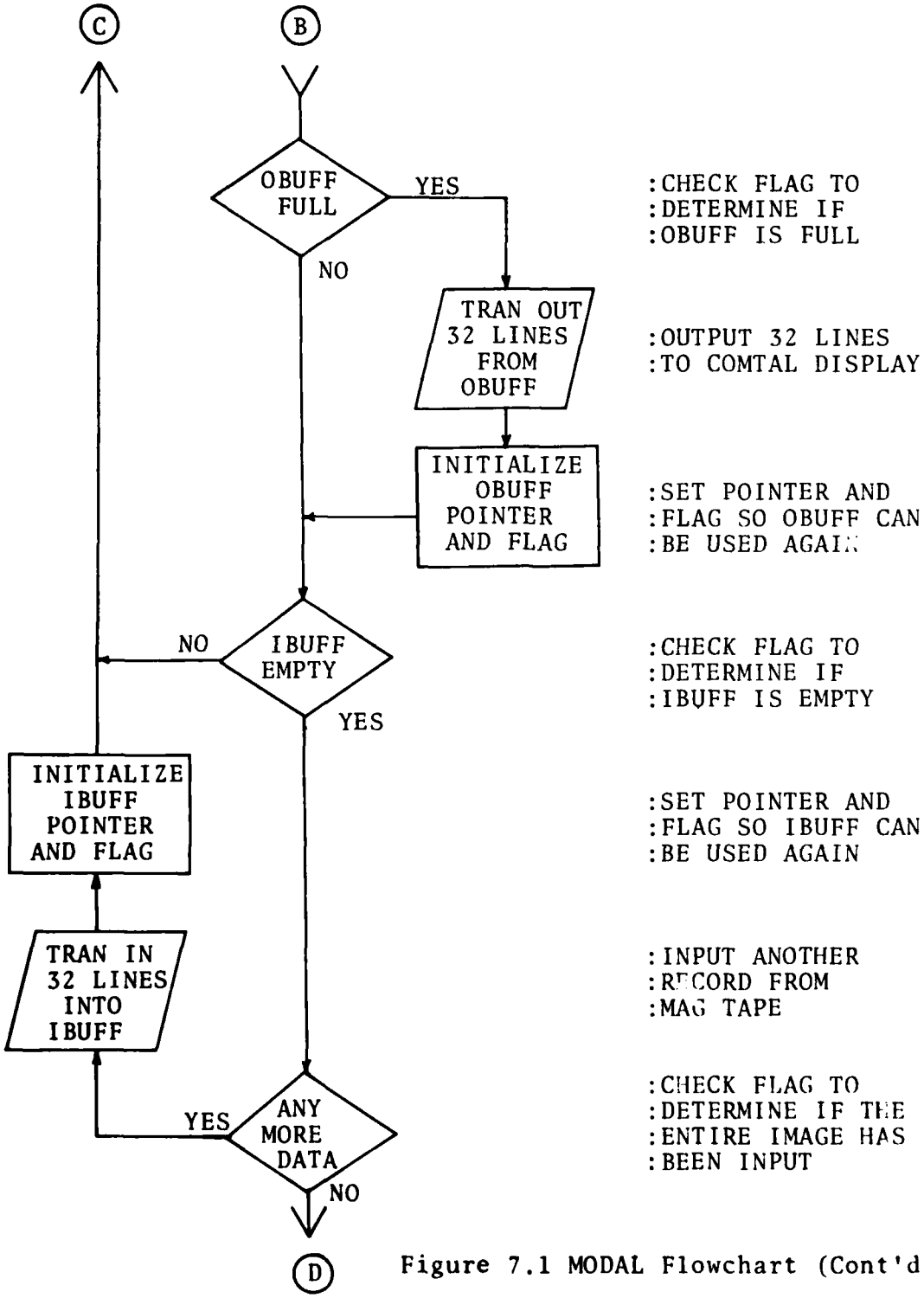


Figure 7.1 MODAL Flowchart (Cont'd)

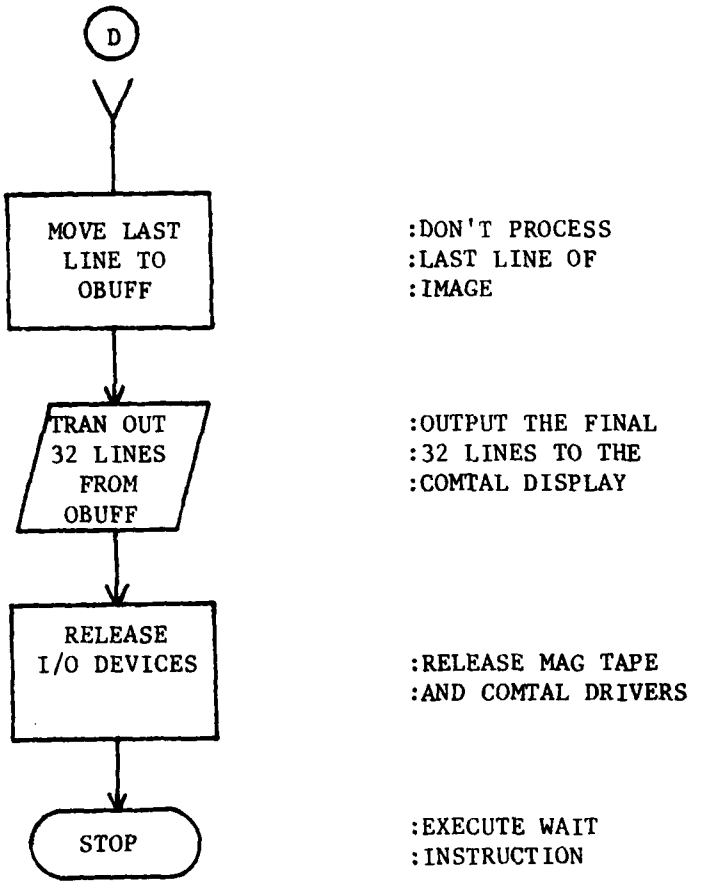


Figure 7.1 MODAL Flowchart (Cont'd)

be output to the COMTAL display is stored in the OBUFF.

The first image line is moved from IBUFF to OBUFF. Since the edge pixels of the image do not have complete neighborhoods, these pixels are not processed. Next, the first two lines from the IBUFF are moved into the associative arrays.

The following step marks the beginning of a program loop which is continued until all image lines have been loaded into the arrays.

Another line from IBUFF is moved into the arrays. This line occupies field (16,8) in the arrays, and is the final line required before processing the second line in the arrays.

Buffers are set up in the arrays to prepare for processing. The comparison buffer is set up in field (24,72). Field (96,96) is then cleared and the frequency count buffer is set up in it.

Pixels in the comparison buffer are now compared with each other, and the frequency counts are stored in the frequency count tag. After this is complete, the frequency count tags are sorted along with their corresponding pixel values. The most frequently occurring gray levels are moved to the output line in the arrays.

The output line is moved to the OBUFF with a pixel swap taking place to ensure proper display on the COMTAL. After the move, image lines 2 and 3 in the array are moved over one field to occupy lines 1 and 2. This allows a new line to be read into the arrays.

The OBUFF is checked next to determine if it is full. If it is, the OBUFF will have its data output to the COMTAL display.

The IBUFF will be checked to determine if it is empty. If it is, more data will be read in, if available; if not, a new line will be moved into the array. When the IBUFF is empty, a flag will be checked to determine if the last image line has been read from the mag tape. If it has, the final line will be output to the COMTAL display and the program will halt. If it has not,

a new record will be read from the mag tape into the array.

Processing will start over when a new line is moved into the arrays.

7.1.3 Results and Evaluation

A digitized image of a photograph was used to test the MODAL program. The digitized image, Figure 4.2 (a), had random noise added to it. Five percent of the pixels had a random gray level, -100 to 100, added to them. Values over 255 were set equal to 255. Values under 0 were set to 0. Figure 7.2 (b) is a photograph of the noisy image.

The MODAL program was run with the noisy image for input. After processing, the output, Figure 7.2 (c), had a significantly lower noise level. The river, left to right on the photograph, is almost entirely free of noise. The tree line, top to bottom, has much less noise in it; however, the improvements were not as great as that seen for the river. Significant improvements are seen in the fields, also, which appear to have undergone about the same amount of noise removal as the tree line.

Noise removal was seen to be greater in the river than anywhere else on the image. Examination of this image shows the main difference in this area is a uniform gray level. The modal technique uses the most frequently occurring gray level. One, two, three, or four noisy pixels in a neighborhood and the uniform gray level, the river, would be in five pixels and, therefore be chosen as the correct value. If the noisy pixels are not the same value, which would be expected for random noise but not for some types of noise, up to seven noise pixels could be in a neighborhood, and the correct pixel value would be chosen.

The reason noise was not removed from the tree line and fields as well as it was from the river, is the variation in gray levels. Two or more gray levels must be exactly the same to be a mode. Since the gray levels can vary from 0 to 255, there are many times when adjacent gray levels are not equal.

(a) Digitized Image

(b) Noise Added

(c) Modal Technique Used on Noisy Image

Figure 7.2 Images with 512 x 512 Resolution

7.2 Odd Pixel

7.2.1 Description

Odd pixel noise reduction uses a 3 x 3 neighborhood to determine if the center pixel is to be replaced [36]. Two different replacement modes are used. In mode 0, the 8 neighboring pixels are added together and divided by 8 to get their average. If this average differs from the center pixel by more than a user specified threshold (THRES), the center point is replaced by the average, otherwise the center point is not changed. In mode 1, the user specifies an additional parameter, the number of neighbors (NON). The number of neighbors varies from 1 to 8. This parameter is the number of neighbors that must differ from the threshold if the center pixel is to be replaced. The number of neighbors actually differing by the threshold amount may be greater than NON, but if it is less, the center point will not change.

7.2.2 STARAN Implementation

The input and output of image data along with the data transfers to and from the arrays are the same as for the MODAL program. The processing is different, as can be seen in Figure 7.3, and will be presented in this section.

Array buffers are different in the ODDPX program, Appendix C, than in the MODAL program. A description of these buffers is in Appendix C.

Mode 0 is implemented on the STARAN by adding all the neighbors together at one time. Each array word has a center pixel in it and the sum of the eight neighbors. The sum is stored in SUMBUF. A comparison is made between the average, sum shifted 3 bits, and the center pixel. If the difference exceeds the user specified threshold (THRES), the average is moved to OUTBUF in place of the center pixel.

If the difference is equal or less than the threshold, the

center pixel is moved to OUTBUF. OUTBUF is used the same way in the ODDPX program as output line is in the MODAL program.

Mode 1 is implemented by adding together all the neighbors that differ from the center pixel by more than the threshold. The sum is stored in SUMBUF and the number of neighbors that exceed the threshold is stored in CNTBUF. if CNTBUF equals or exceeds the user specified parameter NON, then the average of the value in SUMBUF is stored in OUTBUF. If this condition is not met, the center pixel is moved to the OUTBUF.

7.2.3 Results and Evaluation

The ODDPX program processed the same image used by the MODAL program, Figure 4.2 (b).

Processing in mode 0 with a threshold of 25, produced the image seen in Figure 7.4 (a). Noise appears to be almost totally absent in this photograph. The photograph also appears slightly blurred.

Absence of noise in the processed photograph is due to the averaging nature of the processing. The center pixel will tend to blend in more with its neighbors. This also results in blurring of edges. One white spot in the lower right corner of this photograph appears to be caused by the film processing since it is irregular shaped and does not appear in the original or noisy photograph.

When mode 1 was used for processing several different values of NON, some interesting results occurred. With a threshold of 25 and NON=1, the program resulted in the images in Figure 7.4 (b) and (c). The result of trying to choose a new center pixel based on any one pixel, is noise amplification. For each noisy pixel, the surrounding eight neighbors are made equal to it and the noise point is changed to the previous value of its neighbors. This is evident in the Figure 7.4 (c) which is an enlargement of Figure 7.4 (b). The process involved in amplifying the noise is seen in Figure 7.5 The value 1 is the noisy pixel and c is the

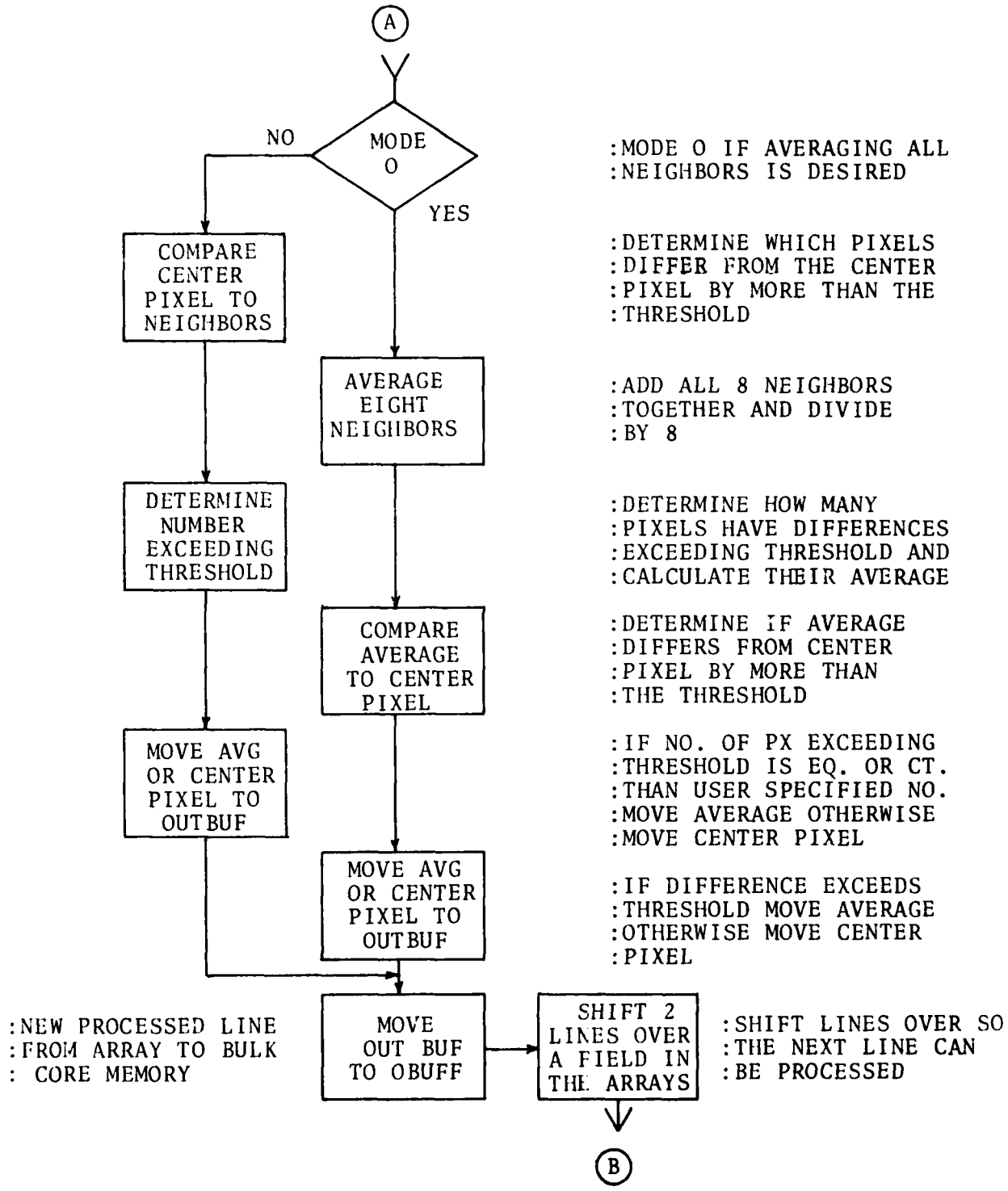


FIGURE 7.3 ODDPX FLOWCHART

(a) ODDPX THRES=25, Averaging

(b) ODDPX THRES=25, NON=1

(c) ODDPX THRES=25, NON=1, Enlarged

Figure 7.4 Images with 512 x 512 Resolution

center point. In Figure 7.5 (a) the center point does not have any noise points as neighbors. The result is no change in the center pixel. In Figure 7.5 (b), the neighborhood has one noisy pixel which differs from the center pixel and other neighbors. The result is a center pixel which is the average of those pixels differing, 1. As we move around the noisy pixel, we make the value of all its neighbors equal to it. In Figure 7.5 (c), all eight neighbors differ, and their average is 0, which results in the noisy pixel being replaced by the neighbor's without noise.

Using mode 1 with NON=3 and THRES=25 results in the image in Figure 7.6 (a). This image is absent of most of the noisy pixels. The areas where noisy pixels are appear to be larger than the original pixels. This is caused by the low value of NON which enlarges those areas with many adjacent noisy pixels. The edge of this image is rough for the same reason.

When NON=5, Figure 7.6 (b), the noisy pixels are absent and the edges are not pitted. When NON=8, Figure 7.6 (c), noisy pixels appear again in the image. This is caused by more than one noisy pixel in the neighborhood or by neighbors whose differences exceed the threshold.

0	0	0		
0	C	0	0	0
0	0	0	1	0
		0	0	0

(a) No Change

U	0	0	
0	C	0	0
0	0	1	0
	0	0	0

(b) Neighbor Changed to Noise

0	0	0
0	C	0
0	0	0

(c) Noise Changed to Neighbor

Figure 7.5 Noise Amplification

(a) ODDPX THRES=25, NON=3

(b) ODDPX THRES=25, NON=5

(c) ODDPX THRES=25, NON=8

Figure 7.6 Images with 512 x 512 Resolution

7.3 Similar Neighbors Technique

7.3.1 Description

Similar neighbor noise removal uses a 3 x 3 neighborhood. Each neighbor is compared to the center pixel. If the center pixel minus the neighbor is greater than the user specified threshold, the neighbor will be similar if high noise is to be removed. If the neighbor minus the center pixel is greater than the threshold, the neighbor will be similar if low noise is to be removed.

The user must specify whether low or high noise is to be removed. When similar neighbors are determined, there must be at least two of them adjacent for the center point to remain unchanged. If at least two are not adjacent, the center pixel will be replaced with the average of the dissimilar neighbors.

7.3.2 STARAN Implementation

The input and output of data into the STARAN in this program is identical to the previous programs.

A description of the array buffers for the program SIMNB is in Appendix D.

The processing, as seen in Figure 7.7, begins similar to mode 1 of OLDPX. The number of pixels whose difference exceeds the threshold is determined. Actual calculations depend on whether low or high noise is to be removed. When similar neighbors are determined, a comparison is made to determine if any are adjacent. If any are, the center pixel is moved to OUTBUF. If two or more adjacent similar pixels are not found, the average of the dissimlar neighbors is moved to OUTBUF.

7.3.3 Results and Evaluation

With the image of Figure 7.2 (b) as the input, the program SIMNB was run. A threshold of 25 and low noise set for removal gave the result seen in Figure 7.8 (a). Removal of low noise appears to be complete with this technique. The image is clear

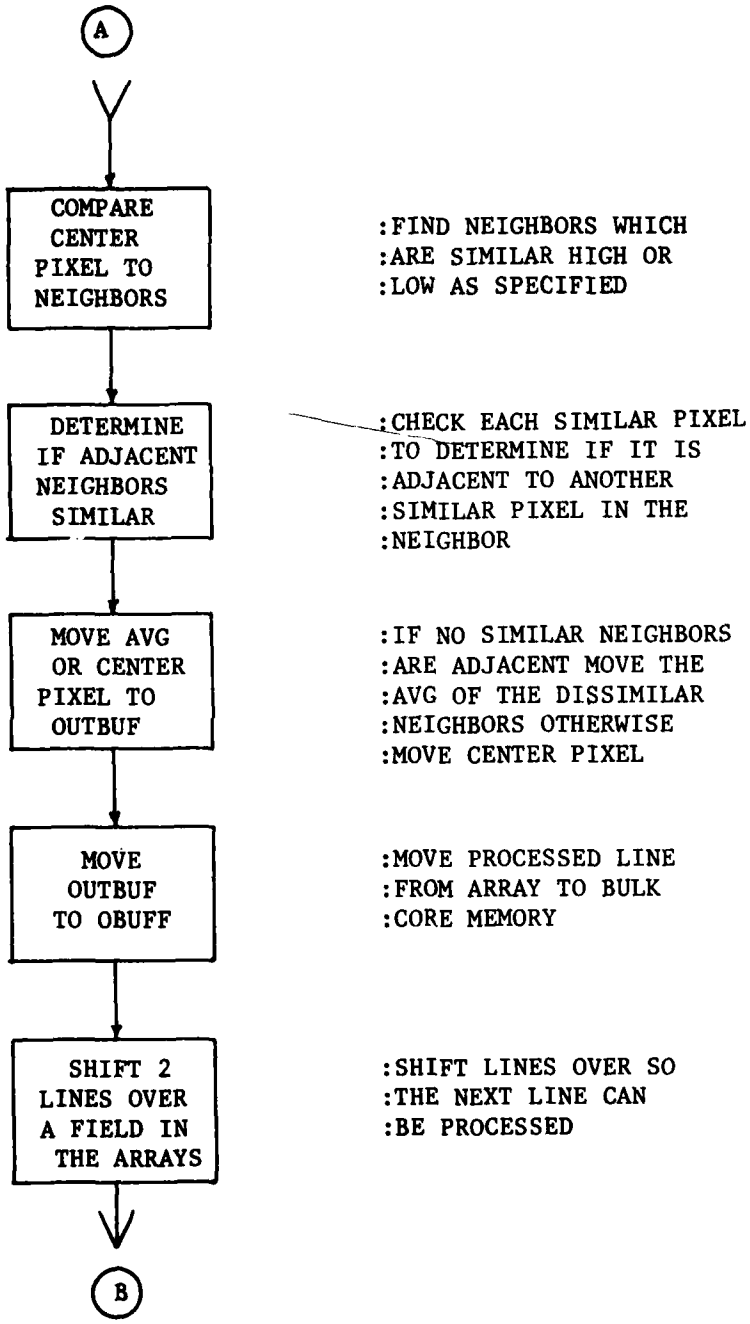


Figure 7.7 SIMNB Flowchart

and sharp.

Using a threshold of 25 and specifying high noise remov results in Figure 7.2 (b). The rivers, tree line and fields appear to be absent of noisy pixels and blurring.

Combining the two operations yields a picture with both high and low noise removed, Figure 7.8 (c). This image is clear and has sharp edges.

Requiring two adjacent pixels to be similar, as this technique does, results in excellent noise removal characteristics. When two adjacent neighbors do not contain noise, the gray level difference between them is usually small, except at the edges. This small difference is usually within the user specified threshold if a reasonable value is chosen. If an edge exists in the 3 x 3 neighborhood, it will usually cause at least two adjacent edge points in the neighborhood. The result of this would be to make no changes to the center pixel.

The similar neighbors technique efficiently removed the noisy pixels from the image. When combined with the sharpness of the image, this appears to be a valuable noise removal technique. The method of requiring two or more adjacent similar neighbors is especially good for preserving edges.

(a) SIMNB THRES=25, Low Noise Removed

(b) SIMNB THRES=25, High Noise Removed

(c) SIMNB THRES=25, Low and High Noise Removed

Figure 7.8 Images with 512 x 512 Resolution

7.4 Comparisons and Conclusions

Execution of image processing algorithms has been done on a PDP 11/45 in the U.S. Air Force's On-Line Pattern Analysis and Recognition System (OLPARS). This system uses two 9-track 800 bpi tape drives for image input and output. Three algorithms, written in assembly language, in OLPARS (MODREP, ODDOT, AND ODDLIN), were run, using the image shown in Figure 4.2 (b). The total run time was measured and based on a worst case tape drive speed, 25 inches per second, execution and I/O times were calculated. The time required to read or write an image to tape is approximately 512 x 512/(800x25)=13 seconds.

NAME		TIME		
Program	OLPARS	Execution	I/O	Total
MODAL	MODREP	4 min, 34 sec	26 sec	5 min, 0 sec
ODDPX	ODDDOT	2 min, 1 sec	26 sec	2 min, 27 sec
SIMNB	ODDLIN	2 min, 22 sec	26 sec	2 min, 48 sec

Programs were also written in FORTRAN that performed the same functions. A multi-platter moving head disk was used for storing image data which resulted in an insignificant amount of I/O time when compared to the execution time. The programs were run in the Electrical Engineering Image Analysis Laboratory at the University of Missouri on a PDP 11/50. The processing times are as follows:

Program Name	Total Time
MODAL	30 min
ODDPX	15 min
SIMNB	15 min

Appendices B, C, and D contain listings of programs executed at Goodyear Aerospace Corporation on the STARAN associative processor. These programs performed the same functions as those mentioned in OLPARS. Each STARAN program contained less than 512 words, which allowed them to be placed in page memory to increase execution speed. The times listed below were measured for

each program:

Program Name	Execution	TIME I/O	Total
MODAL	.998 sec	15.6 sec	16.6 sec
ODDPX	1.08 sec	15.6 sec	16.7 sec
SIMNB	1.12 sec	15.6 sec	16.7 sec

A sizable difference in execution time is seen between the OLPARS programs and the STARAN programs. The execution time ranges from (2x60+1)/1.08=112 to (4x60+34)/.998=275 times faster when the STARAN is used. The total times, however, do not reflect this large speed difference due to the slow I/O devices, the tape drives. Approximately (15.6/16.6)x100%=94% of the STARAN's time is spent waiting on I/O in the MODAL program. This compares to (26/5x60) x 100%=8.7% of the PDP 11/45's time. The resulting total time speed up is only (2x60+1)/16.7=7 to (4x60+34)/16.6=17 times faster with the STARAN.

While a speed up of 275 times is impressive, the STARAN still falls short of real-time processing when parallel I/O facilities are not used. Processing times of 16 seconds do, however, offer help for those bogged down with numerous images to process.

Adding additional arrays to the STARAN would further decrease the execution time of the programs. If a full 32 arrays were available, an increase in speed of approximately 32/2=16 times would be made. This would, however, require considerable expense.

Using parallel I/O would increase throughput to the arrays. This would allow 256 input and 256 output lines for each array. Data could be input directly into the STARAN arrays without being stored in the control memory unit memory. Parallel I/O along with the STARAN's submicrosecond execution time would lend itself to some applications in real-time processing.

When considering the STARAN's cost versus performance, the 750 thousand dollar price tag may well prevent many commercial users from examining it closer. While this computer may cost 10 to 20 times as much as a PDP 11/45, it can deliver over 200 times

the processing speed. This factor alone may not be enough to compensate for the high price.

VIII. Presentation of Results: Edge Detection

8.1 Description

The edge detection (PTEDGE) technique is a method for determining or defining edge points in a digital image. The first step is to determine gradients within the image which exceed a threshold specified by the user. A gradient test is made within a 3 x 3 neighborhood about each point. If the gray-level difference between the center point and any user-specified neighbors is greater than the threshold, the point corresponding in position to the center point is defined as an edge point.

8.2 Algorithm

The original image to be processed for edge detection has a resolution of 512 x 512; this means 512 pixels/line and 512 lines/image. Any particular pixel can be specified by line number and element. A 3 x 3 neighborhood is used to implement the edge detection procedure. In order to establish a comparison between the center point and its neighbors, the neighbors are numbered as shown in Fig. 8.1. The neighbor number, k, must be specified if

6	5	4	line i
7	x	3	line i+1
8	1	2	line i+2
j	j+1	j+2	

x = center point

Fig. 8.1 Assighment of Neighbors for Center Point x

a comparison is to be made x and neighbor k. The resultant edge detected image would have pixels that satisfy the criterion,

$$\left| g(x) - g(k) \right| \geq \text{Threshold}$$

where g(x) and g(k) are the gray levels of the center point and the neighbor k, respectively. For this application, the associative processor of the STARAN has the control memory arranged as shown in Fig. 8.2.

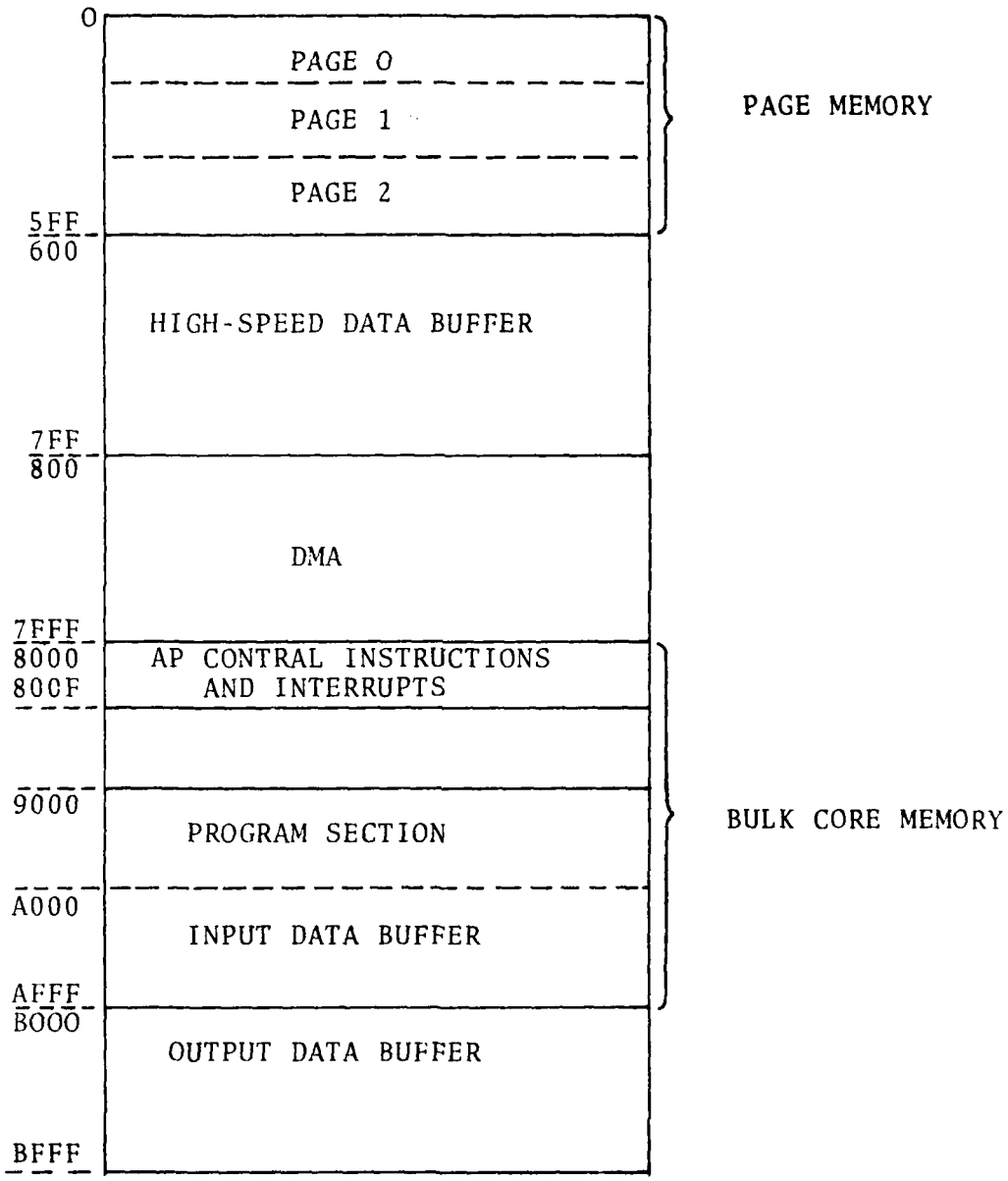


Fig. 8.2 Associative Processor (AP) Control Memory Arrangement

8.3 STARAN Implementation

The next several Figures illustrate the data paths, the arrangement of data and the sequence of tasks which the STARAN would carry out in the implementation of this algorithm. The data paths followed are shown in Fig. 8.3.

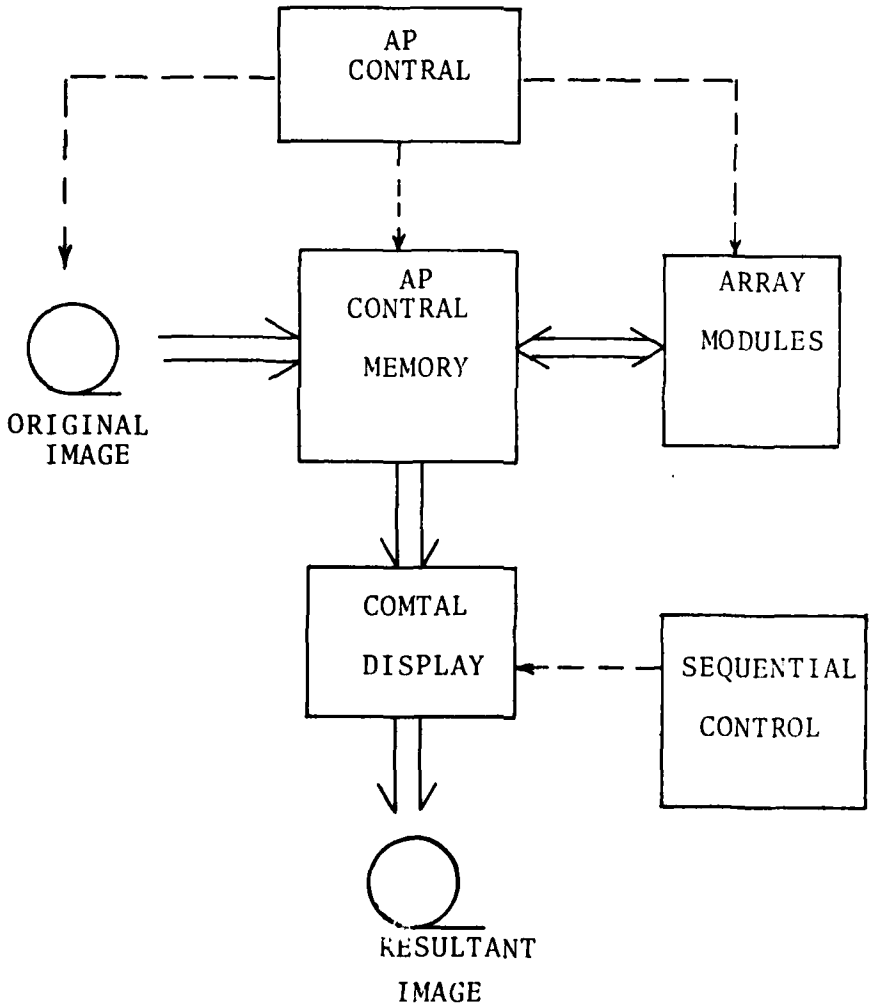
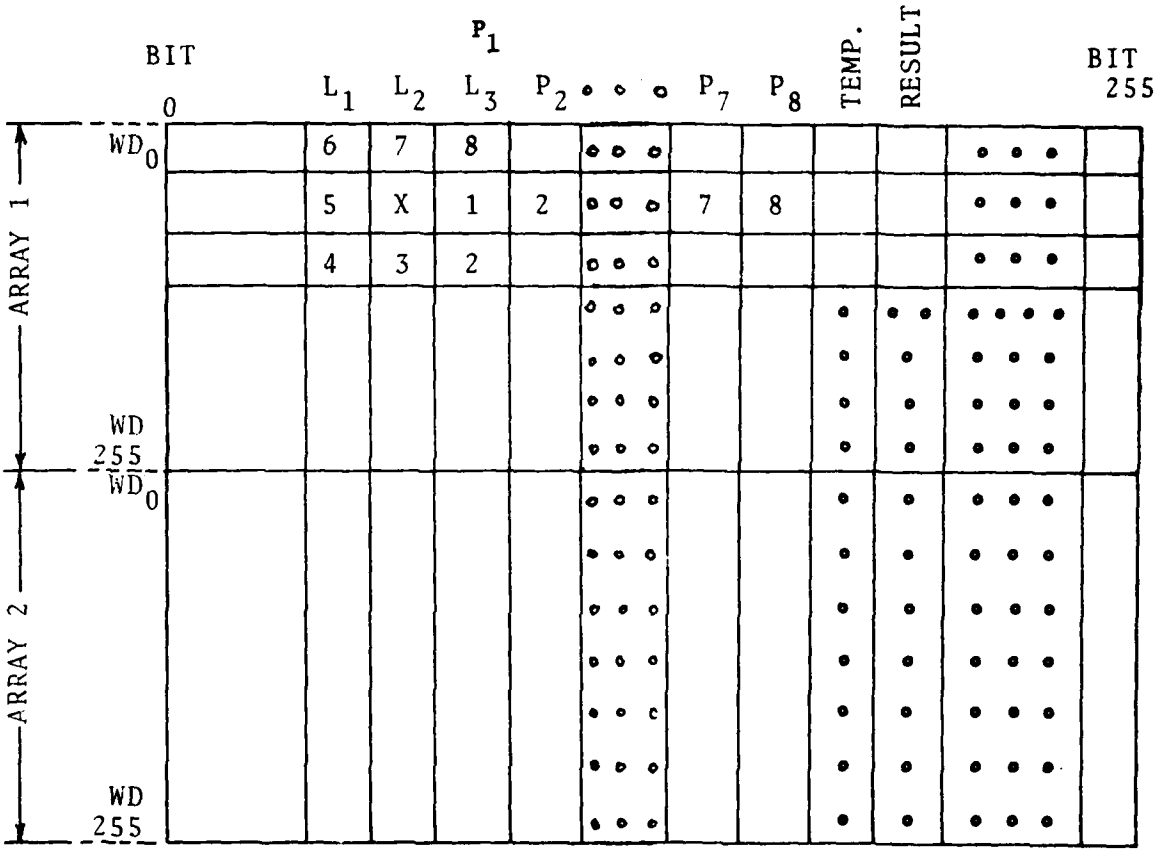


Fig. 8.3 Data Paths in the STARAN

Fig. 8.4 shows the arrangement of array modules in the STARAN as set-up for the execution of the PTEDGE program. The required arrangment of image data on the (input) tape is shown in Fig. 8.5.



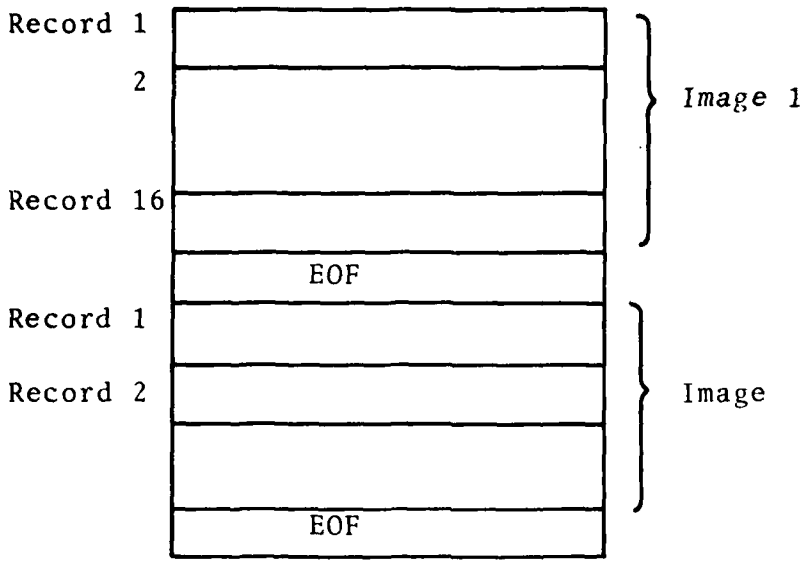
L_1, L_2, L_3 : Store 3 consecutive image lines; each field is 8 bits long (bits 8 - 31).

P_1, P_2, ..., P_8: Store the neighbors of each center point (bits 24 - 87).

Temp. : Store temporary result of comparison (bits 88 - 95).

Result : Store result of overall comparison (bits 96 - 103).

Fig. 8.4 Arrangement of array modules in the STARAN



Each record has 16,384 bytes; this is equivalent to 32 512-pixel image lines; 16 records are needed to store a 512 x 512 image.

Fig. 8.5 Data Structure on the Tape (Input Data)

A flow-chart for the point edge detection program to be executed on the STARAN is shown in Fig. 8.6. A program listing for the PTEDGE technique is given in Appendix E.

8.4 Memory Map and User Options

For the PTEDGE algorithm, program execution relative to bulk core uses the following locations:

a. Locations 6Ø8 - 616 (in high-speed data buffer): all variables and constants.

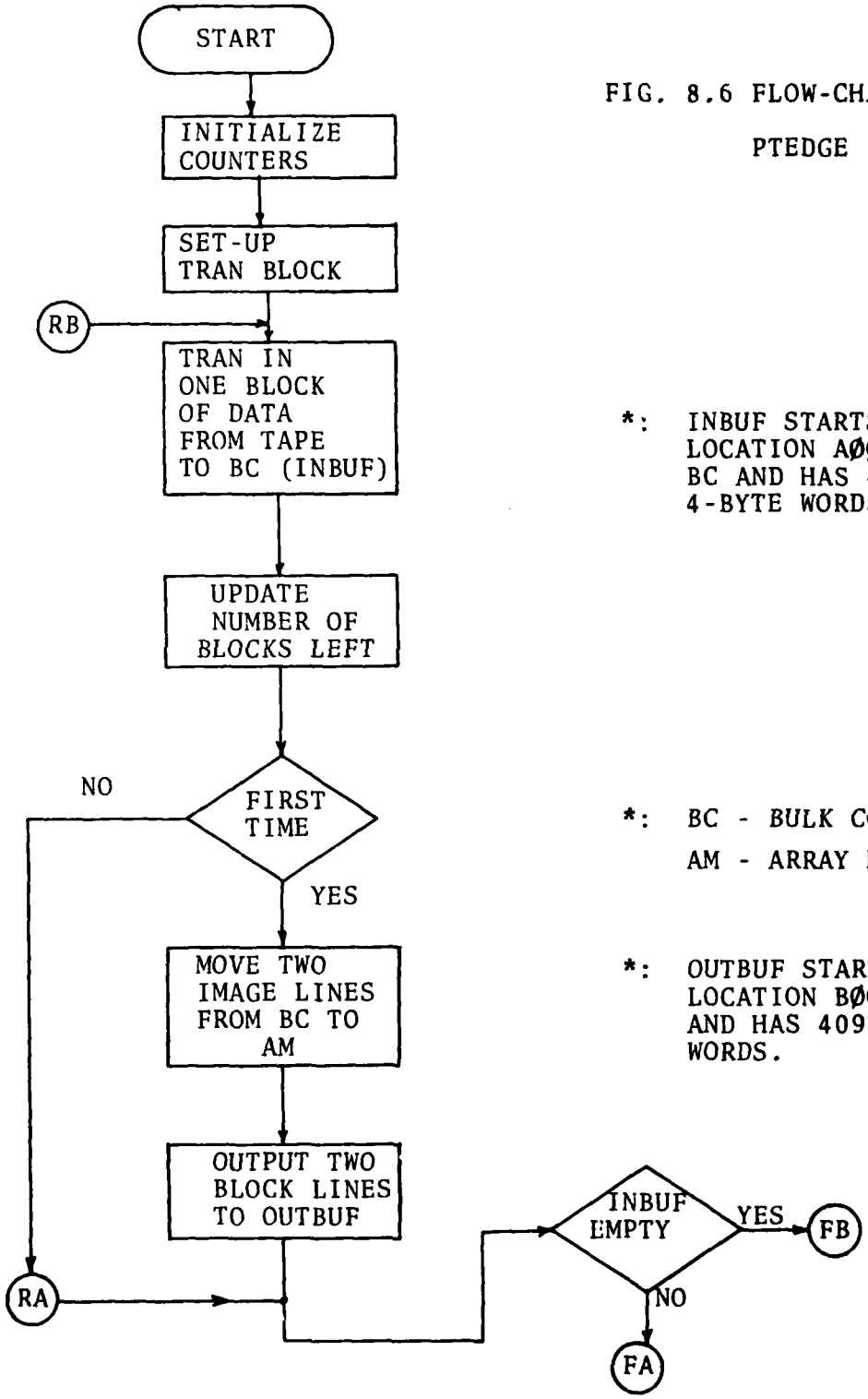


FIG. 8.6 FLOW-CHART FOR PTEDGE

*: INBUF STARTS AT LOCATION AØØØ IN BC AND HAS 4096 4-BYTE WORDS.

*: BC - BULK COR_
AM - ARRAY MEMORY

*: OUTBUF STARTS AT LOCATION BØØØ IN BC AND HAS 4096 4-BYTE WORDS.

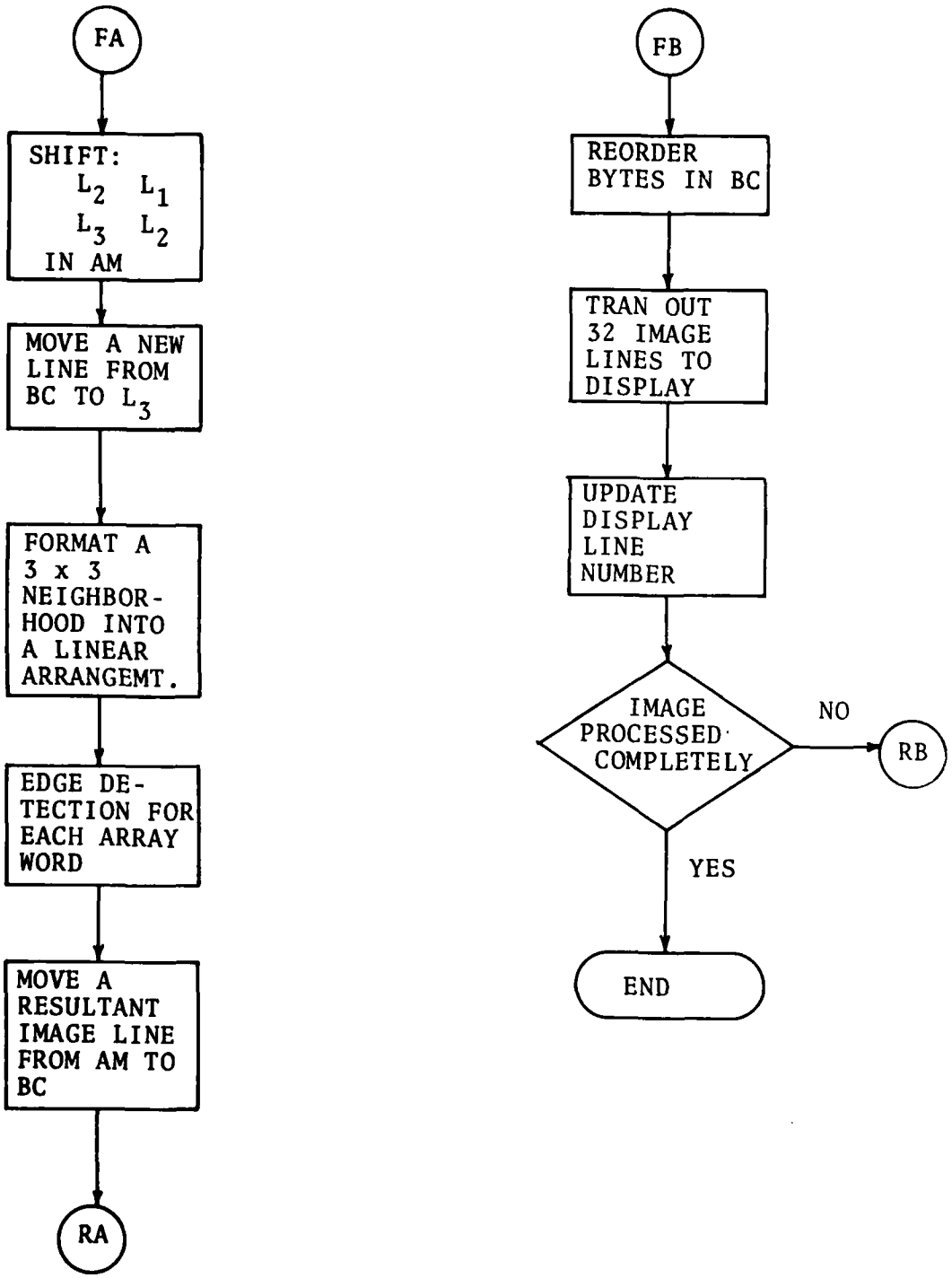


Fig. 8.6 Flow-chart for PTEDGE (Cont'd)

b. Locations 9ØØØ - 928Ø: store program.

c. Locations AØØØ - AFFF: store input data (from tape).

d. Locations BØØØ - BFFF: store processed data (ready for Comtal display).

Program execution relative to page memory uses the following locations:

a. Locations 0000 - Ø28D: store programs.

b. Other locations are retained.

Contents to be changed use the following locations:

a. Location 60C contains the threshold value.

b. Locations 60F - 616 contain the selected neighbors; the selected neighbors are declared by loading the value 1 into its corresponding locations.

8.5 Program Descriptions

The overall program which implements the execution of the PTEDGE technique can be described in terms of the following subprograms:

a. PTEDGE: This is the main program which controls the logical flow during implementation of the Point Edge Detection procedure.

b. EDGE: This subroutine tests the gradient between any selected neighbor and the center point. If the gradients are greater than the threshold, a value of 255 is assigned to the corresponding position in the RESULT field. Otherwise, a value of 0 is assigned.

c. MOVE: This subroutine moves field L_2 to L_1 and L_3 to L_2 after one image line has been processed.

d. LINEIN: This subroutine packs an image line from Bulk Core memory and loads it into the Array memory.

e. LINEBOUT: This subroutine packs the contents of the Array memory, which represents an image line, and moves them to Bulk Core.

f. FORMAT: This subroutine formats a 3 x 3 neighborhood into a linear arrangement in order to provide the Subroutine EDGE with workable data.

g. STORAGE: This subroutine defines all the variables and constants. The addresses of input buffer are also specified.

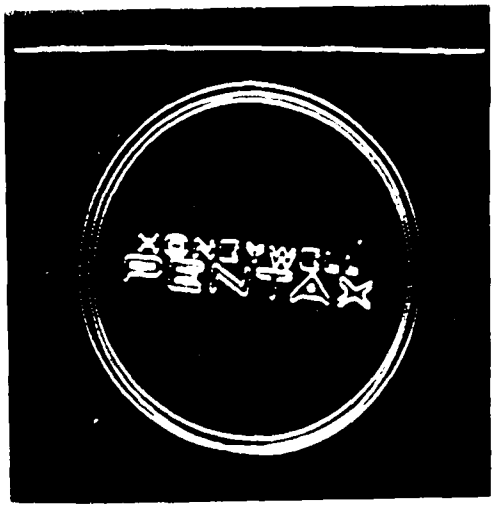
h. ALLIO: This subroutine defines the parameter blocks which are involved in TRAN I/O.

8.6 Results and Evaluation

A digitized image was used to test the PTEDGE program. The original image, Fig. 8.7 (a), is the digitized image of a "lense cap." The outline or shape of the lense cap as well as the lettering on the cap represent potential edges (to the viewer) for detection. The next four figures, Fig. 8.7 (b), 8.7 (c), 8.7 (d), and 8.7 (e), show the results of applying the PTEDGE technique to the image of Fig. 8.7 (a). Each of these four figures, Figs. 8.7 (b) - 8.7 (e), represents a variation in the application of the PTEDGE program. Fig. 8.7 (b) shows the result of detecting horizontal edges, Fig. 8.7 (c) shows vertical edges, Fig. 8.7 (d) shows edges at 45 degrees (to the right), and Fig. 8.7 (e) shows edges in all directions. The white, straight, almost horizontal line near the top of the images shown in Figs. 8.7 (b) - 8.7 (e) is an artifact. By examining the results of applying the PTEDGE program, one would conclude that the program is working. Note, for example, that for a given direction of edge detection, edges that are parallel to that direction almost disappear, while edges that are at right angles to that direction show a definite sharpening. The PTEDGE program was also applied to the digital image of a runway (provided by RADC); the original is shown in Fig. 8.8 (a). Fig. 8.8 (b) shows the result of applying this program in the vertical direction, Fig. 8.8 (c) shows similar results for edges along a 45 degree angle (to the left), and Fig. 8.8 (d) shows similar results for applying the program in all directions. Fig. 8.8 (c) also shows a horizontal artifact structure near the middle of the image.

(a) Original Image

(b) Horizontal Edges



(c) Vertical Edges

Fig. 8.7 Results of Applying the PTIDGI Program

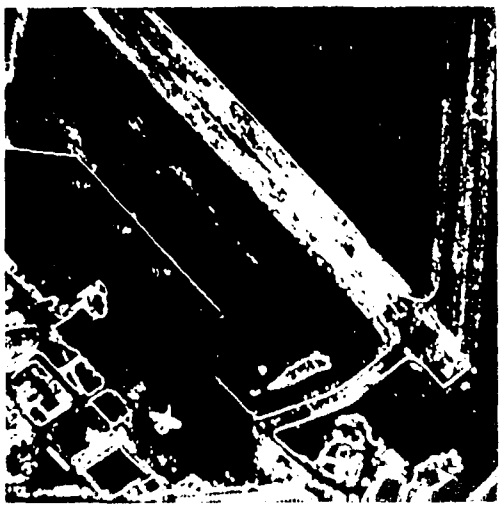
(d) 45° Edges

(e) Edges in All Directions

Fig. 8.7 Results of Applying the PTEDGE Program

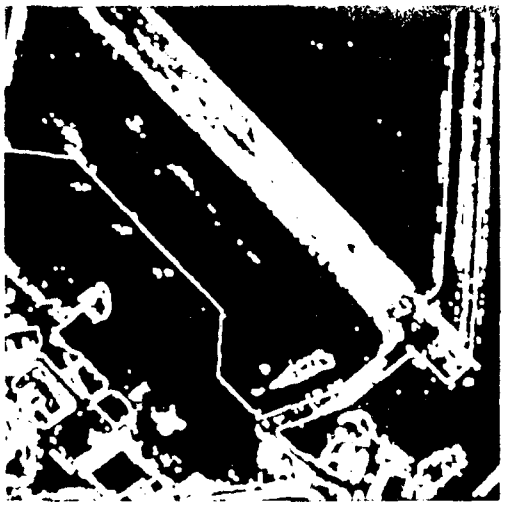
(a) Original Image

Fig. 8.8 Results of Applying the PTEDGE Program to Runway



(b) Vertical Edges

[illegible] at 45° (to the left)



(d) Edges in All Directions

Fig. 8.8 Results of Applying the [illegible] Program to Runway

8.7 Time Efficiency

The table shown below, Table 8.1, shows the execution times associated with each one of a number of tasks that are necessary for the implementation of the PTEDGE program by the STARAN computer. From an examination of this table, several conclusions can be made. 1) By comparing these results with similar ones obtained for the programs associated with the nosie reduction programs discussed in Section VII, there is a definite speed advantage in using the STARAN by one to two orders of magnitude. 2) The most time consuming tasks are TRANIN and TRANOUT, which are associated with getting the data in and out of the processor; this reinforces the need for a parallel I/O port (256 bits long) to take advantage of the associative processor. 3) There is a definite speed advantage in loading the program into page memory instead of bulk core; this was an expected result. 4) An 8-point edge detection scheme requires more execution time than for a one point edge detection scheme, but the difference is only about 2%.

Program Sect. \ Program Load In:		Page Memory (sec)	Bulk Memory (sec)
Overall Efficiency	8 points	17.88.09219	21.8283354
	1 point	17.5888866	21.3121987
EDGE	8 points	0.1392855	0.623975
	1 point	0.0509124	0.1654172
TRANIN		7.89336855	6.609249
TRANOUT		8.5948942	8.2893537
LINEIN		0.4318520	2.1508331
LINEOUT		0.3844740	1.5908665
FORMAT		0.0342776	0.1977352
MOVE		0.0053604	0.0290251

Table 8.1 Time Efficiency for Tasks Associated with the Implementation of the PTEDGE Program.

IX. Conclusions/Future Work

As emphasized earlier, the STARAN was chosen for 1) its special architecture to speed-up image feature extraction and processing, and 2) accessibility and interest to the Air Force. The results obtained in this project clearly demonstrate that a special computer architecture does have a speed advantage. However, the STARAN itself does not take advantage of current technology, and, of course, a host computer more closely matched to the capability of the STARAN would increase the throughput. In particular, if the host of "control" computer had a 256 bit I/O register, the data transfers in and out of storage would be more efficient, and throughput would be improved. The results indicate that with the use of a special computer architecture such as the STARAN, the speed-up in execution time is on the order of two orders of magnitude. When I/O time is included, however, the speed advantage is only about one order of magnitude. In both cases, the basis for comparison is a PDP 11/45, using assembly-level programming.

The emphasis in this project has been on the use of the STARAN to demonstrate the speed advantage of special computer architectures. Also, several structures other than that of the STARAN were investigated with the objective to determine their amenability to speed-up image feature extraction and processing. As a result, it is recommended that three additional structures be studied further to determine their advantages and disadvantages in an image processing environment and that they be applied to specific algorithms. One of these structures is a pipeline structure; when this is coupled with a multiport memory, there is a definite indication that it can result in a speed-up of image processing. Another structure which has a strong potential for the speed-up of image processing is a multiprocessor structure, where multiple processors operate in parallel on the same image. The interprocessor connections can vary considerably.

References

1. E.S. Deutsch and J.R. Fram, "A Quantitative Study of the Orientation Bias of Some Edge Detection Schemes," IEEE Trans. on Computers, vol. C-27, no. 3, March, 1978.

2. S.D. Shapiro, "Transform Method of Curve Detection for Textured Image Data," IEEE Trans. on Computers, vol. C-27, no. 3, March, 1978.

3. L.S. Davis, et al., "On Models for Line Detection," University of Maryland Computer Science Center Report, TR-258, August, 1973.

4. R. Nevatia, "Locating Object Boundaries in Textured Environments," IEEE Trans. on Computers, vol. C-25, no. 11, November, 1976.

5. L.S. Davis, S. Johns, and J.K. Aggarwal, "Texture Analysis Using Generalized Co-Occurrence Matrices," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 3, July, 1979.

6. T. Pavlidis, "The Use of a Syntactic Shape Analyzer for Contour Matching," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 3, July, 1979.

7. H. Wechslet and M. Kodode, "A New Edge Detection Technique and its Implementation," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-7, no. 12, 1977.

8. H. Wechsler and M. Kodode, "A Random Walk Procedure for Texture Discrimination," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 3, July, 1979.

9. J. Birk, R. Kelley, N. Chen, and L. Wilson, "Image Feature Extraction Using Diameter Limited Gradient Direction Histograms," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 2, April, 1979.

10. O.D. Faugeras and W.K. Pratt, "Decorrelation Methods for Texture Feature Extraction," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 4, July, 1980.

11. R.W. Conners and C.A. Harlow, "A Theoretical Comparison of Texture Algorithms," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 3, May, 1980.

12. K. Deguchi and I. Morishita, "Texture Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques," IEEE Trans/Computers, August, 1978.

13. C.M. Bjorklund and T. Pavlidis, "Global Shape Analysis by k-Syntactic Similarity," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 2, March, 1981.

14. T.H. Hong, C.R. Dyer, and A. Rosenfeld, "Texture Primitive Extraction Using an Edge-Based Approach, IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-10, no. 11, November 1980.

15. T. Ichikawa, "A Pyramidal Representation of Images and its Feature Extraction Facility," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 3, May, 1981.

16. L.G. Shapiro, "A Structural Model of Shape," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 2, March, 1980.

17. T. Pavlidis and F. Ali, "A Hierarchical Syntactic Shape Analyzer," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, Jan., 1979.

18. J. Sklansky, "Image Segmentation and Feature Extraction," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-8, no. 4, April, 1978.

19. P.M. Narendra and M. Goldberg, "Image Segmentation with Directed Trees," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 2, March, 1980.

20. W.A. Perkins, "Area Segmentation of Images Using Edge Points," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 1, 1980.

21. K. Price and R. Reddy, "Matching Segments of Images," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, Jan., 1979.

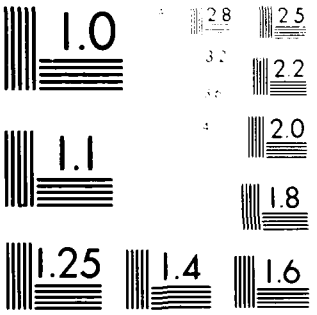
22. E. Persoon and K.S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-7, no. 3, March, 1977.

23. K.S. Shanmugam, F.M. Dickey, and J.A. Green, "An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, Jan., 1979.

24. R. Machuca and A.L. Gilbert, "Finding Edges and Noisy Scenes," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, Jan., 1979.

AD-A111 882 MISSISSIPPI STATE UNIV MISSISSIPPI STATE DEPT OF ELEC--ETC F/G 5/8
THE APPLICATION OF SPECIAL COMPUTING TECHNIQUES TO SPEED-UP IMA--ETC(U)
DEC 81 R W MCLAREN, W D MCFARLAND F30602-80-C-0032
UNCLASSIFIED RADC-TR-81-230 NL

2 OF 3
AD A
111882



MICROCOPY RESOLUTION TEST CHART

25. J.S. Weszka and A. Rosenfeld, "Threshold Evaluation Techniques," IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-8, no. 8, Aug., 1978.

26. C.R. Dyer and A. Rosenfeld, "Thinning Algorithms for Gray-Scale Pictures," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, Jan., 1979.

27. J.M. Prager, "Extracting and Labeling Boundary Segments in Natural Scenes," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 1, Jan., 1980.

28. L.S. Davis and T.C. Henderson, "Hierarchical Constraint Processes for Shape Analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 3, May, 1981.

29. L.S. Davis, "Shape Matching Using Relaxation Techniques," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. Pami-1, no. 1, Jan., 1979.

30. J.A. Richards, D.A. Landgrebe, and P.H. Swain, "Pixel Labeling by Supervised Probabilistic Relaxation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 2, March, 1981.

31. A.J. Danker and A. Rosenfeld, "Blob Detection by Relaxation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 1, Jan., 1981.

32. N. Ahuja and A. Rosenfeld, "Mosaic Models for Textures," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 1, Jan., 1981.

33. L.G. Shapiro and R.M. Haralick, "Decomposition of Two-Dimensional Shapes by Graph Theoretic Clustering," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, Jan., 1979.

34. P.M. Narendra, "A Separable Median Filter for Image Noise Smoothing," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-3, no. 1, Jan., 1981.

35. A.K. Jain, "A Sinusoidal Family of Unitary Transforms," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 4, Oct., 1979.

36. "Image Processing System Software- User's Manual," RADC Technical Report, TR-79-52, vol. 1, June, 1979.

37. R.H. Stafford, Digital Television, John Wiley, 1980.

38. K.J. Thurber, Large Scale Computer Architecture, Hayden, New Jersey, 1976.

39. G.H. Barnes, "The ILLIAC IV Computer," IEEE Trans. on Computers, vol. C-17, no. 8, Aug., 1968.

40. R.L. Davis, "The ILLIAC IV Processing Element," IEEE Trans. on Computers, vol. C-18, no. 9, Sept., 1969.

41. D.J. Kuck, "ILLIAC IV Software and Application Programming," IEEE Trans. on Computers, vol. C-17, no. 8, Aug., 1968.

42. A. Rosenfeld, Digital Picture Processing, Academic Press, New York, 1976.

43. "AP-120B Array Processor," Floating Point Systems, Inc., Form 7244, Rev. 1, March, 1978.

44. "Array Processor Math Library- Part I," Floating Point Systems Inc., 1978.

45. J.N. Palasco, Floating Point Systems, Inc., "Slide Presentation," June, 1979.

46. Reference Manual, Goodyear Aerospace Corp., GER-15636B, Sept., 1974.

47. APPLE Programming Manual, Goodyear Aerospace Corp., Akron, Ohio, GER-15637B, Sept., 1974.

48. PDP 11 Processor Handbook, Digital Equipment Corp., Maynard, Mass., 1978.

49. D.J. Kuck, High-Speed Computer and Algorithm Organization, Academic Press, New York, 1977.

50. G. Goos, Parallel Processing, Springer-Verlag, New York, 1975.

51. C.C. Foster, Content Addressable Parallel Processors, Van Nostrand, 1976.

Additional References on Special Computer Architecture

1. J. Weglarz, "Multiprocessor Scheduling with Memory Allocation- A Deterministic Approach," IEEE Trans. on Computers, vol. C-29, no. 8, Aug., 1978.

2. C.L. Wu, and T.Y. Feng, "On a Class of Multistage Interconnection Networks," IEEE Trans. on Computers, vol. C-29, no. 8, Aug., 1980.

3. D. Nassimi and S. Sahni, "Data Broadcasting in SIMD Computers," IEEE Trans. on Computers, vol. C-30, no. s, Feb., 1981.

4. J.A. Arulpragasam, R.A. Giggi, R.F. Lary, D.T. Sullivan, and C.C. Wu, "Modular Minicomputers Using Microprocessors," IEEE Trans. on Computers, vol. C-29, no. 2, Feb., 1980.

5. R.W. Holgate and R.N. Ibbett, "An Analysis of Instruction-Fetching Strategies in Pipelined Computers," IEEE Trans. on Computers, vol. C-29, no. 4, April, 1980.

6. J. Bruno, J.W. Jones, III, and K. So, "Deterministic Scheduling with Pipelined Processors," IEEE Trans. on Computers, vol. C-29, no. 4, April, 1980.

7. A.P. Reeves, "A Systematically Designed Binary Array Processor," IEEE Trans. on Computers, vol. C-29, no. 4, April, 1980.

8. D.D. Gajski, "Parallel Compressors," IEEE Trans. on Computers, vol. C-29, no. 5, May, 1980.

9. R.A. Finkel and M.H. Solomon, "Processor Interconnection Strategies," IEEE Trans. on Computers, vol. C-29, no. 5, May, 1980.

10. D.A. Padua, D.J. Kuck, and D.H. Lawie, "High-Speed Multiprocessors and Compilation Techniques," IEEE Trans. on Computers, vol. C-29, no. 9, Sept., 1980.

11. F.T. Fung and H.C. Torng, "On the Analysis of Memory Conflicts and Bus Contentions in Multiple-Microprocessor Systems," IEEE Trans. on Computers, vol. C-28, no. 1, Jan., 1979.

12. S.B. Wu and M.T. Liu, "A Cluster Structure as an Interconnection Network for Large Multimicrocomputer Systems," IEEE Trans. on Computers, vol. C-30, no. 4, April, 1981.

APPENDIX A

STARAN INSTRUCTION SET

Control Memory Unit Instructions

Branch

B a(r)±k,c

This instruction is an unconditional branch. Entry 'a' is either a symbol or a constant. Entry 'r' is optional and may be register RO through R7 or DP. Entry 'k' is optional and is a simple expression modifying a(r). Entry 'c' is a control digit which modifies BL or DP registers as follows:

c	Function
1	Decrement BL
2	Decrement DP
3	Decrement BL and increment DP
4	Decrement DP
5	Decrement BL and DP

BAL,r_1 a(r)±k,c

This is a branch and link instruction. The program branches to a location determined by a(r)±k. When the instruction B 0(r_1) is encountered, a branch to the address in r_1 is executed. This address is that of the instruction immediately following the BAL,r_1 instruction.

BNZ,r_1 a(r)±k,c

This instruction executes a branch if the value of register r_1 is not equal to zero.

BZ,r_1 a(r)±k,c

This instruction executes a branch is the value of register r_1 is equal to zero.

LOOP,a_1 a(r)±k

This instruction will loop through a program segment starting with the instruction LOOP and ending at address a(r)±k. The number of loops is 'a_1' which can be a simple address expression.

RPT,a

The repeat instruction executes the instruction following it 'a' times.

Register

DECR b

The contents of register 'b' is decremented by 1.

INCR b

The contents of register 'b' is incremented by 1.

LI,s b,a

The immediate data 'a' is loaded into register 'b'. Entry 's' is optional and specifies the number of left end-around byte shifts before loading.

LR,s b,a(r)±k,c

Register 'b' is loaded with the contents of memory location a(r)±k.

SR,s b,a(r)±k,c

The contents of register 'b' is stored at location a(r)±k.

Control

WAIT

This instruction sets the processor to an inactive state.

Associative Array Instructions

Load

CLR a

This instruction changes all bits in response store register 'a' to zero.

L,w a,b

This instruction loads the response store register 'a' with the source 'b'. Entry 'a' can be response store register X, Y, or M. Entry 'b' can be response store register X, Y, or M, a simple address expression, an associative field expression, a field pointer, a link pointer or a resolver value. When an address, associative field expression or a field pointer is used, a bit column or word is loaded into the response store register 'a' depending on entry 'w'. Entry 'w' is optional and is used to indicate word mode access of the array. When a link pointer (FP12) is used for entry 'b' FP1 points to the array and FP2 to the word or bit column.

LN,w a,b

This isntruction operates the same way the L,w a,b does except the data loaded in 'a' is complemented.

LCM c,d

This instruction loads field 'c' in the common register with field 'd' in the array. The link pointer (FP12) points to the word which will be used in the associative array.

LCW e

This instruction will load the common register with one of eight 32-bit blocks from response store register X or Y. Entry 'e' designates the register

and block number, for example, X(1) is the X response store register bits 32 through 63.

ROT a,b,c

This instruction will rotate the selected response store register or common register, entry 'a'. Entry 'b' indicates the number of end-around bit positions to be rotated. If 'b' is negative, the rotation is left, otherwise it is right. Entry 'c' is optional and indicates the modulus to be rotated.

SET a

This instruction changes all bits in response store register 'a' to one.

Store

S,w a,b

This instruction stores response store register 'a' into destination 'b'. Entries 'a', 'b', and 'w' are the same as those used in instruction L,w a,b.

SC c,d

This instruction will store field 'c' of the common register into field 'd' of the associative memory or response store register 'd'. If the associative memory is used, response store register M is used as a mask register. Only array words with mask bits set participate in the operation.

SCW c,d

This instruction is the same as SC c,d except the associative memory word is pointed to by the link pointer (FP12).

Searches

EQF a,b

This instruction sets bits in the Y response store register if corresponding mask bits (M response store) are set and the contents of the array field 'a' is equal to the contents of array field 'b'.

GEC a,b

This instruction sets bits in the Y response store register if corresponding mask bits (M response store) are set and the contents of array field 'a' is greater than or equal to the contents of common register field 'b'.

GTC a,b

This instruction is the same as GEC except field 'a' must be greater than field 'b'.

LEC a,b

This instruction is the same as GEC except field 'a' must be less than or equal to field 'b'.

LTC a,b

This instruction is the same as GEC except field 'a' must be less than field 'b'.

LTF a,b

This instruction sets bits in the Y response store register if corresponding mask bits (M resopnse store) are set and the contents of array field 'a' is less than the contents of array field 'b'.

Move

MVF a,b

This instruction moves the contents of array field

'a' to array field 'b' in each word of the array which has its mask (M resonse store) bit set.

MVNF a,b

This instruction operates the same as MVF except the two's complement of field 'a' is moved.

Arithmetic

ADC a,b,c

This instruction adds field 'a' in the common register to field 'b' in the array and stores the result in field 'c' of the array. This operation takes place in each word that has its mask bit set.

ADF a,b,c

This instruction is the same as ADC except all the fields 'a', 'b' and 'c' are in the array.

DVF a,b,c

This instruction divides field 'a' by field 'b' and stores the result in field 'c'. All fields are in the array. Each word with a mask bit set participates.

SBF a,b,c

This instruction subtracts field 'b' from field 'a' and stores the results in field 'c'. All fields are in the array. Each word with a mask bit set participates.

APPENDIX B

MODAL PROGRAM

```
                ;
                ;
                ;
                MODAL  START
                       EXTRN   LINKBK1,LINKBK2,TRAN1,TRAN2
                       EXTRN   LINKWD1,LINKWD2
                       ENTRY   ERRTN1,ERRTN2,ERRTN3
                       ENTRY   ERRTN4,ERRTN5
                ;
                ;
                ;
                ;      DAVID M. CRAWFORD
                ;      RESEARCH ASSISTANT
                ;      ELECTRICAL ENGINEERING DEPT.
                ;      UNIVERSITY OF MISSOURI - COLUMBIA
                ;      16 MAY 1980
                ;      REVISION: 16 JULY 1980
                ;      REVISION: 22 JULY 1980
                ;
                ;
                ;      THIS PROGRAM IS DESIGNED TO PERFORM NOISE
                ;      REDUCTION ON IMAGES BY USING A MODAL RE-
                ;      PLACEMENT TECHNIQUE.  THE IMAGE, 512 X 512
                ;      PIXELS, IS READ FROM MAGNETIC TAPE AND PRO-
                ;      CESSED BY STARAN.  THE NEW IMAGE IS THEN
                ;      OUTPUT TO THE COMTAL DISPLAY.
                ;
                ;      MODAL REPLACEMENT-
                ;
                ;      THIS TECHNIQUE USES A 3X3 NEIGHBORHOOD.  ANY
                ;      GRAY LEVELS IN THE NEIGHBORHOOD THAT OCCUR
                ;      MORE THAN ONCE ARE GROUPED INTO MODES.  THE
                ;      MORE OFTEN THE GRAY LEVEL OCCURS THE HIGHER
                ;      THE FREQUENCY COUNT OF THE MODE.  THE HIGH-
                ;      EST FREQUENCY COUNT CORRESPONDS TO THE
                ;      SELECTED MODE.  IF SEVERAL MODES HAVE THE
                ;      SAME FREQUENCY COUNT A PRIORITY SYSTEM
                ;      IS USED AS FOLLOWS;
                ;      CENTER, TOP LEFT, TOP CENTER, TOP RIGHT,
                ;      MIDDLE LEFT, ETC.  THIS IS ILLUSTRATED
                ;      BELOW.
                ;
                ;              1 2 3 . . . . FIRST LINE
                ;              4 5 6 . . . . SECOND LINE
                ;              7 8 9 . . . . THIRD LINE
                ;
                ;              PRIORITY - 5,1,2,3,4,6,7,8
                ;
                ;      IF NO MODES OCCUR THE CENTER PIXEL IS
                ;      CHOSEN AS THE MODAL VALUE.
                ;
```

```
                    ;
                    ;       THE PROGRAM OPERATES WITH THREE LINES STORED
                    ;       IN ARRAYS 0 AND 1, FIELDS (0,8), (8,8), (16,8).
                    ;       THE FIRST PIXEL OF THE FIRST LINE IS STORED IN
                    ;       FIELD (0,8) OF WORD 0 IN ARRAY 0.  THE LAST
                    ;       PIXEL OF THE FIRST LINE IS IN FIELD (0,8) OF
                    ;       WORD 255 IN ARRAY 1.  THE SECOND LINE IS IN
                    ;       FIELD (8,8) AND THE THIRD IN FIELD (16,8).
                    ;       THE MODAL REPLACEMENT VALUE IS DETERMINED FOR
                    ;       THE LINE IN FIELD (8,8).  A COMPARISON BUFFER
                    ;       (24,72) IS SET UP IN THE ARRAYS IN THE SAME
                    ;       WORD WHICH CONTAINS THE PIXEL BEING CHECKED
                    ;       FOR REPLACEMENT.  COMPARISON BUFFER IS USED TO
                    ;       DETERMINE FREQUENCY COUNT.  THIS COUNT IS STORE
                    ;       IN A TAG NEXT TO ITS GRAY LEVEL VALUE IN THE
                    ;       FREQUENCY COUNT BUFFER (96,96).  SORTING THE
                    ;       TAG VALUES RESULTS IN THE MODAL REPLACEMENT
                    ;       VALUE WHICH IS STORED IN THE OUTPUT LINE FIELD
                    ;       (192,8) AND OUTPUT TO THE COMTAL.  ALL EDGE
                    ;       POINTS OF THE IMAGE ARE OUTPUT UNALTERED.
                    ;
                    ;
                    ;
                    ;
                    ;       THE ILLUSTRATION BELOW SHOWS WHERE THE
                    ;       BUFFERS ARE IN THE ARRAYS.  ARRAY 0 IS
                    ;       SHOWN WHICH CONTAINS THE FIRST HALF OF
                    ;       THE IMAGE LINE.  ARRAY 1 IS SIMILAR.
                    ;
                    ;
                    ****************************************
                    * 1 * 4 * 7 *                          *
                    ******************************         *
                    * 2 * 5 * 8 *147258369*51234678*       *
                    ******************************         *
                    * 3 * 6 * 9 *         *        *       *
                    ************ COMPARE *  FREQ  *       *
                    *   *   *   *  BUFFER    COUNT *       *
                    *   *   *   *           BUFFER *       *
                    *   *   *   *                  *       *
                    * F * S * T *                  * O *
                    * I * E * H *                  * U *
                    * R * C * I *                  * T *
                    * S * O * R *                  * P *
                    * T * N * D *                  * U *
                    *   * D *   *                  * T *
                    *   *   *   *                  *   *
                    * L * L * L *                  * L *
                    * I * I * I *                  * I *
                    * N * N * N *                  * N *
                    * E * E * E *                  * E *
                    *   *   *   *                  *   *
                    ;
                    ;
                    ;
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00003 V
                    038B ERRTN1 EQU     ERR             ERROR RETURN
                    038B ERRTN2 EQU     ERR             ERROR RETURN
                    038B ERRTN3 EQU     ERR             ERROR RETURN
                    038B ERRTN4 EQU     ERR             ERROR RETURN
                    038B ERRTN5 EQU     ERR             ERROR RETURN
                    0000 BLKNUM EQU     0               I/O BLCK NUMBER
                    A000 IBUFF  EQU     X'A000'         IBUFF ADDRESS
                    B000 OBUFF  EQU     X'B000'         OBUFF ADDRESS
                    1000 IBSIZE EQU     4096            INPUT BUFFER SIZE
                    1000 OBSIZE EQU     4096            OUTPUT BUFFER SIZE
                    0020 LIB    EQU     IBSIZE/128      IMAGE LINES IN IBUFF
                    0020 LOB    EQU     OBSIZE/128      IMAGE LINES IN OBUFF
                    0010 BLKS   EQU     512/LIB         NO. OF INPUT BLOCKS
                    0080 SP     EQU     128             32 BIT SEGMENTS PER LIN
                    1000 MXOBDP EQU     OBSIZE          MAX OBUFF DP VALUE
                    1000 MXIBDP EQU     IBSIZE          MAX IBUFF DP VALUE
                    0000 MODAL  EQU     $
 0000 0000 3660C000         LI,2    AS,X'C000'      SELECT ARRAYS 0 AND 1
                      ;
                      ;     INITIALIZE INPUT -- MAG TAPE
                      ;
 0001 0001 74201000         LI      CH,IBSIZE
 0002 0002 32000000         LI      CL,BLKNUM
 0003 0003 30010001         SR      C,TRAN2+1
 0004 0004 74200000         LI      CH,0
 0005 0005 3200A000         LI      CL,IBUFF
 0006 0006 30010003         SR      C,TRAN2+3
                            INIT    LINKBK2
 0007 0007 72800000
 0008 0008 34A00014
 0009 0009 30C18010
                      ;
                      ;     INITIALIZE OUTPUT -- COMTAL
                      ;
 000A 000A 74201000         LI      CH,OBSIZE
 000B 000B 32000000         LI      CL,BLKNUM
 000C 000C 30010001         SR      C,TRAN1+1
 000D 000D 74200000         LI      CH,0
 000E 000E 3200B000         LI      CL,OBUFF
 000F 000F 30010003         SR      C,TRAN1+3
                            INIT    LINKBK1
 0010 0010 72800000
 0011 0011 34A00014
 0012 0012 30C18010
                      ;
                      ;     INPUT LINES TO IBUFF (NO. = LIB)
                      ;
                            TRAN    TRAN2
 0013 0013 72800000
 0014 0014 34A00016
 0015 0015 30C18010
                            IOWAIT  LINKWD2
 0016 0016 72800000
 0017 0017 74A00000
 0018 0018 37200000
 0019 0019 30C18010
```

```
    1                       ;
    2                       ;       INITIALIZE BUFFER POINTERS
    3                       ;
    4 001A 001A 34A01000            LI      BL,MXIBDP       INIT IBUFF EMPTY FLAG
W   5 001B 001B 30810613            SR      BL,IBEF
    6 001C 001C 34A01000            LI      BL,MXOBDP       INIT OBUFF EMPTY FLAG
W   7 001D 001D 30810612            SR      BL,OBEF
    8 001E 001E 34A00010            LI      BL,BLKS         INIT LAST IBUFF FLAG
W   9 001F 001F 30810614            SR      BL,LIF
   10                       ;
   11                       ;       MOVE FIRST LINE IN IBUFF TO OBUFF
   12                       ;
   13 0020 0020 73C00000            LI      FP12,0
   14 0021 0021 32800000            LI      DP,0
   15 0022 0022 34810612            LR      BL,OBEF
   16 0023 0023 3F7F002C            LOOP,SP LINE1           LINE ONE TO IBUFF
   17 0024 0024 3602A000            LR      C,IBUFF(DP)     LOAD 4 PX IN C REG
   18 0025 0025 400077A1            CLR     X               PIXEL SWAP
   19 0026 0026 420099A0            SC      X(0)            1,2,3,4 TO 4,3,2,1
   20 0027 0027 400888BB            ROT     X,-8,16
      0028 0028 400088BB
   21 0029 0029 401088BB            ROT     X,-16,32
      002A 002A 400088BB
   22 002B 002B 21C0A0FB            LCW     X(0)
   23 002C 002C 3005B000 LINE1      SR      C,OBUFF(DP),3   STORE 4 PX IN OBUFF
W  24 002D 002D 30810610            SR      DP,OBDP
W  25 002E 002E 30810612            SR      BL,OBEF
   26                       ;
   27                       ;       MOVE FIRST TWO LINES IN
   28                       ;       IBUFF TO THE ARRAYS
   29                       ;
   30 002F 002F 32800000            LI      DP,0
   31 0030 0030 34810613            LR      BL,IBEF
   32 0031 0031 33C00000            LI      FP12,0          ARRAY WORD POINTER
   33 0032 0032 3F7F0050            LOOP,SP L1              FIRST LINE IN ARRAY
   34 0033 0033 3605A000            LR      C,IBUFF(DP),3   LOAD 4 PX IN C REG
   35 0034 0034 400088A1            SCW     (0,8),(0,8)     PX TO ARRAY
      0035 0035 4FC0A03F
      0036 0036 57C00000
      0037 0037 08000003
   36 0038 0038 01E00001            INCR    FP12            NEXT WORD
   37 0039 0039 400088A1            SCW     (8,8),(0,8)     PX TO ARRAY
      003A 003A 4FC0A03F
      003B 003B 42008840
      003C 003C 40F8885A
      003D 003D 4000885A
      003E 003E 57C00002
      003F 003F 08000003
   38 0040 0040 01E00001            INCR    FP12            NEXT WORD
   39 0041 0041 400088A1            SCW     (16,8),(0,8)    PX TO ARRAY
      0042 0042 4FC0A03F
      0043 0043 42008840
      0044 0044 40F0885A
      0045 0045 4000885A
      0046 0046 57C00002
      0047 0047 08000003
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00005 V
    1 0048 0048 01E00001        INCR    FP12            NEXT WORD
    2 0049 0049 400088A1        SCW     (24,8),(0,8)    PX TO ARRAY
      004A 004A 4FC0A03F
      004B 004B 42008840
      004C 004C 40E0885A
      004D 004D 40F8885A
      004E 004E 57C00002
      004F 004F 08000003
    3 0050 0050 01E00001 L1     INCR    FP12            NEXT WORD
    4 0051 0051 33C00000        LI      FP12,0          ARRAY WORD POINTER
    5 0052 0052 3F7F006F        LOOP,SP L2              2ND LINE IN ARRAY
    6 0053 0053 3605A000        LR      C,IBUFF(DP),3   LOAD 4 PX IN C REG
    7 0054 0054 400088A1        SCW     (0,8),(8,8)     PX TO ARRAY
      0055 0055 4FC0A87F
      0056 0056 42008840
      0057 0057 40F83852
      0058 0058 57C00002
      0059 0059 08000003
    8 005A 005A 01E00001        INCR    FP12            NEXT WORD
    9 005B 005B 400088A1        SCW     (8,8),(8,8)     PX TO ARRAY
      005C 005C 4FC0A87F
      005D 005D 57C00000
      005E 005E 08000003
   10 005F 005F 01E00001        INCR    FP12            NEXT WORD
   11 0060 0060 400088A1        SCW     (16,8),(8,8)    PX TO ARRAY
      0061 0061 4FC0A87F
      0062 0062 42008840
      0063 0063 40F8885A
      0064 0064 4000885A
      0065 0065 57C00002
      0066 0066 08000003
   12 0067 0067 01E00001        INCR    FP12            NEXT WORD
   13 0068 0068 400088A1        SCW     (24,8),(8,8)    PX TO ARRAY
      0069 0069 4FC0A87F
      006A 006A 42008840
      006B 006B 40F0885A
      006C 006C 4000885A
      006D 006D 57C00002
      006E 006E 08000003
   14 006F 006F 01E00001 L2     INCR    FP12            NEXT WORD
W  15 0070 0070 30810611        SR      DP,IBDP
W  16 0071 0071 30810613        SR      BL,IBEF
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00006 V
    1                        ;
    2                        ;      MOVE ANOTHER LINE FROM IBUFF TO THE ARRAY
    3                        ;
    4 0072 0072 32810611 NXLINE LR      DP,IBDP
    5 0073 0073 34810613        LR      BL,IBEF
    6 0074 0074 33C00000        LI      FP12,0           ARRAY WORD POINTER
    7 0075 0075 3F7F0091        LOOP,SP LINE             MOVE LINE IN ARRAY
    8 0076 0076 3605A000        LR      C,IBUFF(DP),3    LOAD 4 PX IN C REG
    9 0077 0077 400088A1        SCW     (0,8),(16,8)     PX TO ARRAY
      0078 0078 4FC0B0BF
      0079 0079 42008840
      007A 007A 40F08852
      007B 007B 57C00002
      007C 007C 08000003
   10 007D 007D 01E00001        INCR    FP12             NEXT WORD
   11 007E 007E 400088A1        SCW     (8,8),(16,8)     PX TO ARRAY
      007F 007F 4FC0B0BF
      0080 0080 42008840
      0081 0081 40F88852
      0082 0082 57C00002
      0083 0083 08000003
   12 0084 0084 01E00001        INCR    FP12             NEXT WORD
   13 0085 0085 400088A1        SCW     (16,8),(16,8)    PX TO ARRAY
      0086 0086 4FC0B0BF
      0087 0087 57C00000
      0088 0088 08000003
   14 0089 0089 01E00001        INCR    FP12             NEXT WORD
   15 008A 008A 400088A1        SCW     (24,8),(16,8)    PX TO ARRAY
      008B 008B 4FC0B0BF
      008C 008C 42008840
      008D 008D 40F8885A
      008E 008E 4000885A
      008F 008F 57C00002
      0090 0090 08000003
   16 0091 0091 01E00001 LINE   INCR    FP12             NEXT WORD
W  17 0092 0092 30810611        SR      DP,IBDP
W  18 0093 0093 30810613        SR      BL,IBEF
```

```
                        ;
                        ;       SET UP PIXEL COMPARISON BUFFER(24,72) IN THE
                        ;       ARRAY FIELDS (24,24),(48,24) AND (72,24)
                        ;
 0094 0094 73900000             LI      FP1,0
 0095 0095 73400018             LI      FP2,24
 0096 0096 35A00048             LI      FP3,72
 0097 0097 3F17009F             LOOP,24 SHFT            MOVE FIELD (0,24)
 0098 0098 430088A5             L       X,FP1
 0099 0099 40FF88B3             ROT     X,1             DOWN A WORD TO
 009A 009A 5B400003             S       X,FP2           FIELD (24,24) AND
 009B 009B 40FE888B             ROT     X,-2            UP A WORD TO
 009C 009C 400088BB
 009D 009D 18800003             S       X,FP3           FIELD (72,24)
 009E 009E 01700001             INCR    FP1,FP2         NEXT BIT COLUMN
 009F 009F 01A00001 SHFT        INCR    FP3
 00A0 00A0 73C00000             LI      FP12,0          ARRAY 0 WORD 0
 00A1 00A1 47C088A5             LCM     (0,24),(24,24)  TO
 00A2 00A2 40F888B3
 00A3 00A3 65C1A0BB
 00A4 00A4 73C00100             LI      FP12,X'0100'    ARRAY 1 WORD 0
 00A5 00A5 400088A1             SCW     (0,24),(24,24)
 00A6 00A6 4FC0A0BF
 00A7 00A7 40008841
 00A8 00A8 40F8885A
 00A9 00A9 40E0885A
 00AA 00AA 48000002
 00AB 00AB 48C00001
 00AC 00AC 42008840
 00AD 00AD 40F8885A
 00AE 00AE 40E0885A
 00AF 00AF 57C00002
 00B0 00B0 48000003
 00B1 00B1 73C001FF             LI      FP12,X'01FF'    ARRAY 1 WORD 255
 00B2 00B2 47C088A5             LCM     (0,24),(72,24)  TO
 00B3 00B3 401888B3
 00B4 00B4 65C280BB
 00B5 00B5 73C000FF             LI      FP12,X'00FF'    ARRAY 0 WORD 255
 00B6 00B6 400088A1             SCW     (0,24),(72,24)
 00B7 00B7 4FC2A8FF
 00B8 00B8 42408840
 00B9 00B9 40F88852
 00BA 00BA 57C00002
 00BB 00BB 48000003
 00BC 00BC 4000BBA1             SET     M
 00BD 00BD 48000003
 00BE 00BE 37904717             MVF     (0,24),(48,24)
 00BF 00BF 3F1700C2
 00C0 00C0 433488A5
 00C1 00C1 4B800001
 00C2 00C2 13A40003
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00008 V
    1                        ;
    2                        ;       CLEAR FIELD (96,96) FOR USE BY
    3                        ;       THE FREQUENCY COUNT BUFFER
    4                        ;
    5 00C3 00C3 74200000             LI      CH,0
    6 00C4 00C4 72000000             LI      CL,0
    7 00C5 00C5 33C01F7F             SC      (0,32),(96,32)
      00C6 00C6 3F1F00C8
      00C7 00C7 4B40B7A1
      00C8 00C8 13740003
    8 00C9 00C9 33C01F9F             SC      (0,32),(128,32)
      00CA 00CA 3F1F00CC
      00CB 00CB 4B40B7A1
      00CC 00CC 13740003
    9 00CD 00CD 33C01FBF             SC      (0,32),(160,32)
      00CE 00CE 3F1F00D0
      00CF 00CF 4B40B7A1
      00D0 00D0 13740003
```

```
  1                        ;
  2                        ;       SET UP FREQUENCY COUNT BUFFER
  3                        ;
  4 00D1 00D1 37906B3F         MVF     (56,8),(100,8)  PX 5 TO POSITION 1
    00D2 00D2 3F0700D5
    00D3 00D3 433488A5
    00D4 00D4 4B800001
    00D5 00D5 13A40003
  5 00D6 00D6 3790771F         MVF     (24,8),(112,8)  PX 1 TO POSITION 2
    00D7 00D7 3F0700DA
    00D8 00D8 433488A5
    00D9 00D9 4B800001
    00DA 00DA 13A40003
  6 00DB 00DB 37908337         MVF     (48,8),(124,8)  PX 2 TO POSITION 3
    00DC 00DC 3F0700DF
    00DD 00DD 433488A5
    00DE 00DE 4B800001
    00DF 00DF 13A40003
  7 00E0 00E0 37908F4F         MVF     (72,8),(136,8)  PX 3 TO POSITION 4
    00E1 00E1 3F0700E4
    00E2 00E2 433488A5
    00E3 00E3 4B800001
    00E4 00E4 13A40003
  8 00E5 00E5 37909B27         MVF     (32,8),(148,8)  PX 4 TO POSITION 5
    00E6 00E6 3F0700E9
    00E7 00E7 433488A5
    00E8 00E8 4B800001
    00E9 00E9 13A40003
  9 00EA 00EA 3790A757         MVF     (80,8),(160,8)  PX 6 TO POSITION 6
    00EB 00EB 3F0700EE
    00EC 00EC 433488A5
    00ED 00ED 4B800001
    00EE 00EE 13A40003
 10 00EF 00EF 3790B32F         MVF     (40,8),(172,8)  PX 7 TO POSITION 7
    00F0 00F0 3F0700F3
    00F1 00F1 433488A5
    00F2 00F2 4B800001
    00F3 00F3 13A40003
 11 00F4 00F4 3790BF47         MVF     (64,8),(184,8)  PX 8 TO POSITION 8
    00F5 00F5 3F0700F8
    00F6 00F6 433488A5
    00F7 00F7 4B800001
    00F8 00F8 13A40003
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00010 V
    1                        ;
    2                        ;       DETERMINE THE PIXEL FREQUENCY COUNT
    3                        ;       AND STORE IT IN THE TAG
    4                        ;
    5 00F9 00F9 72000001             LI      CL,1            COUNT INCREMENTER
    6 00FA 00FA 77906B77             EQF     (100,8),(112,8) COMPARE PX 5 TO PX 1
      00FB 00FB 00008841
      00FC 00FC 3F0700FF
      00FD 00FD 43A488A5
      00FE 00FE 433400A5
      00FF 00FF 00002243
    7 0100 0100 48000002             L       M,Y
    8 0101 0101 75E00363             ADC     (96,4),(28,4),(96,4)       INCR TAG?
      0102 0102 73C01F63
      0103 0103 37200303
      0104 0104 2C000000
    9 0105 0105 4000BBA1             SET     M
      0106 0106 48000003
   10 0107 0107 77906B83             EQF     (100,8),(124,8) COMPARE PX 5 TO PX 2
      0108 0108 00008841
      0109 0109 3F07010C
      010A 010A 43A488A5
      010B 010B 433400A5
      010C 010C 00002243
   11 010D 010D 48000002             L       M,Y
   12 010E 010E 75E00363             ADC     (96,4),(28,4),(96,4)       INCR TAG?
      010F 010F 73C01F63
      0110 0110 37200303
      0111 0111 2C000000
   13 0112 0112 4000BBA1             SET     M
      0113 0113 48000003
   14 0114 0114 77906B8F             EQF     (100,8),(136,8) COMPARE PX 5 TO PX 3
      0115 0115 00008841
      0116 0116 3F070119
      0117 0117 43A488A5
      0118 0118 433400A5
      0119 0119 00002243
   15 011A 011A 48000002             L       M,Y
   16 011B 011B 75E00363             ADC     (96,4),(28,4),(96,4)       INCR TAG?
      011C 011C 73C01F63
      011D 011D 37200303
      011E 011E 2C000000
   17 011F 011F 4000BBA1             SET     M
      0120 0120 48000003
   18 0121 0121 77906B9B             EQF     (100,8),(148,8) COMPARE PX 5 TO PX 4
      0122 0122 00008841
      0123 0123 3F070126
      0124 0124 43A488A5
      0125 0125 433400A5
      0126 0126 00002243
   19 0127 0127 48000002             L       M,Y
   20 0128 0128 75E00363             ADC     (96,4),(28,4),(96,4)       INCR TAG?
      0129 0129 73C01F63
      012A 012A 37200303
      012B 012B 2C000000
   21 012C 012C 4000BBA1             SET     M
      012D 012D 08000003
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00011 V
    1 012E 012E 77906BA7        EQF     (100,8),(160,8) COMPARE PX 5 TO PX 6
      012F 012F 00008841
      0130 0130 3F070133
      0131 0131 43A488A5
      0132 0132 433400A5
      0133 0133 00002243
    2 0134 0134 48000002        L       M,Y
    3 0135 0135 75E00363        ADC     (96,4),(28,4),(96,4)       INCR TAG?
      0136 0136 73C01F63
      0137 0137 37200303
      0138 0138 2C000000
    4 0139 0139 4000BBA1        SET     M
      013A 013A 48000003
    5 013B 013B 77906BB3        EQF     (100,8),(172,8) COMPARE PX 5 TO PX 7
      013C 013C 00008841
      013D 013D 3F070140
      013E 013E 43A488A5
      013F 013F 433400A5
      0140 0140 00002243
    6 0141 0141 48000002        L       M,Y
    7 0142 0142 75E00363        ADC     (96,4),(28,4),(96,4)       INCR TAG?
      0143 0143 73C01F63
      0144 0144 37200303
      0145 0145 2C000000
    8 0146 0146 4000BBA1        SET     M
      0147 0147 48000003
    9 0148 0148 77906BBF        EQF     (100,8),(184,8) COMPARE PX 5 TO PX 8
      0149 0149 00008841
      014A 014A 3F07014D
      014B 014B 43A488A5
      014C 014C 433400A5
      014D 014D 00002243
   10 014E 014E 48000002        L       M,Y
   11 014F 014F 75E00363        ADC     (96,4),(28,4),(96,4)       INCR TAG?
      0150 0150 73C01F63
      0151 0151 37200303
      0152 0152 2C000000
   12 0153 0153 4000BBA1        SET     M
      0154 0154 48000003
   13 0155 0155 77906B5F        EQF     (100,8),(88,8)  COMPARE PX 5 TO PX 9
      0156 0156 00008841
      0157 0157 3F07015A
      0158 0158 43A488A5
      0159 0159 433400A5
      015A 015A 00002243
   14 015B 015B 48000002        L       M,Y
   15 015C 015C 75E00363        ADC     (96,4),(28,4),(96,4)       INCR TAG?
      015D 015D 73C01F63
      015E 015E 37200303
      015F 015F 2C000000
   16 0160 0160 4000BBA1        SET M
      0161 0161 48000003
   17 0162 0162 77907783        EQF     (112,8),(124,8) COMPARE PX 1 TO PX 2
      0163 0163 00008841
      0164 0164 3F070167
      0165 0165 43A488A5
      0166 0166 433400A5
      0167 0167 00002243
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00012 V
    1 0168 0168 48000002        L       M,Y
    2 0169 0169 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      016A 016A 73C01F6F
      016B 016B 37200303
      016C 016C 2C000000
    3 016D 016D 4000BBA1        SET     M
      016E 016E 48000003
    4 016F 016F 7790778F        EQF     (112,8),(136,8) COMPARE PX 1 TO PX 3
      0170 0170 00008841
      0171 0171 3F070174
      0172 0172 43A488A5
      0173 0173 433400A5
      0174 0174 00002243
    5 0175 0175 48000002        L       M,Y
    6 0176 0176 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      0177 0177 73C01F6F
      0178 0178 37200303
      0179 0179 2C000000
    7 017A 017A 4000BBA1        SET     M
      017B 017B 48000003
    8 017C 017C 7790779B        EQF     (112,8),(148,8) COMPARE PX 1 TO PX 4
      017D 017D 00008841
      017E 017E 3F070181
      017F 017F 43A488A5
      0180 0180 433400A5
      0181 0181 00002243
    9 0182 0182 48000002        L       M,Y
   10 0183 0183 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      0184 0184 73C01F6F
      0185 0185 37200303
      0186 0186 2C000000
   11 0187 0187 4000BBA1        SET     M
      0188 0188 48000003
   12 0189 0189 779077A7        EQF     (112,8),(160,8) COMPARE PX 1 TO PX 6
      018A 018A 00008841
      018B 018B 3F07018E
      018C 018C 43A488A5
      018D 018D 433400A5
      018E 018E 00002243
   13 018F 018F 48000002        L       M,Y
   14 0190 0190 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      0191 0191 73C01F6F
      0192 0192 37200303
      0193 0193 2C000000
   15 0194 0194 4000BBA1        SET     M
      0195 0195 48000003
   16 0196 0196 779077B3        EQF     (112,8),(172,8) COMPARE PX 1 TO PX 7
      0197 0197 00008841
      0198 0198 3F07019B
      0199 0199 43A488A5
      019A 019A 433400A5
      019B 019B 00002243
   17 019C 019C 48000002        L       M,Y
   18 019D 019D 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      019E 019E 73C01F6F
      019F 019F 37200303
      01A0 01A0 2C000000
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00013 V
    1 01A1 01A1 4000BBA1        SET     M
      01A2 01A2 48000003
    2 01A3 01A3 779077BF        EQF     (112,8),(184,8) COMPARE PX 1 TO PX 8
      01A4 01A4 00008841
      01A5 01A5 3F0701A8
      01A6 01A6 43A488A5
      01A7 01A7 433400A5
      01A8 01A8 00002243
    3 01A9 01A9 48000002        L       M,Y
    4 01AA 01AA 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      01AB 01AB 73C01F6F
      01AC 01AC 37200303
      01AD 01AD 2C000000
    5 01AE 01AE 4000BBA1        SET     M
      01AF 01AF 48000003
    6 01B0 01B0 7790775F        EQF     (112,8),(88,8)  COMPARE PX 1 TO PX 9
      01B1 01B1 00008841
      01B2 01B2 3F0701B5
      01B3 01B3 43A488A5
      01B4 01B4 433400A5
      01B5 01B5 00002243
    7 01B6 01B6 48000002        L       M,Y
    8 01B7 01B7 75E0036F        ADC     (108,4),(28,4),(108,4)     INCR TAG?
      01B8 01B8 73C01F6F
      01B9 01B9 37200303
      01BA 01BA 2C000000
    9 01BB 01BB 4000BBA1        SET     M
      01BC 01BC 48000003
   10 01BD 01BD 7790838F        EQF     (124,8),(136,8) COMPARE PX 2 TO PX 3
      01BE 01BE 00008841
      01BF 01BF 3F0701C2
      01C0 01C0 43A488A5
      01C1 01C1 433400A5
      01C2 01C2 00002243
   11 01C3 01C3 48000002        L       M,Y
   12 01C4 01C4 75E0037B        ADC     (120,4),(28,4),(120,4)     INCR TAG?
      01C5 01C5 73C01F7B
      01C6 01C6 37200303
      01C7 01C7 2C000000
   13 01C8 01C8 4000BBA1        SET     M
      01C9 01C9 48000003
   14 01CA 01CA 7790839B        EQF     (124,8),(148,8) COMPARE PX 2 TO PX 4
      01CB 01CB 00008841
      01CC 01CC 3F0701CF
      01CD 01CD 43A488A5
      01CE 01CE 433400A5
      01CF 01CF 00002243
   15 01D0 01D0 48000002        L       M,Y
   16 01D1 01D1 75E0037B        ADC     (120,4),(28,4),(120,4)     INCR TAG?
      01D2 01D2 73C01F7B
      01D3 01D3 37200303
      01D4 01D4 2C000000
   17 01D5 01D5 4000BBA1        SET     M
      01D6 01D6 08000003
```

```
  1 01D7 01D7 779083A7        EQF     (124,8),(160,8) COMPARE PX 2 TO PX 6
    01D8 01D8 00008841
    01D9 01D9 3F0701DC
    01DA 01DA 43A488A5
    01DB 01DB 433400A5
    01DC 01DC 00002243
  2 01DD 01DD 48000002        L       M,Y
  3 01DE 01DE 75E0037B        ADC     (120,4),(28,4),(120,4)     INCR TAG?
    01DF 01DF 73C01F7B
    01E0 01E0 37200303
    01E1 01E1 2C000000
  4 01E2 01E2 4000BBA1        SET     M
    01E3 01E3 48000003
  5 01E4 01E4 779083B3        EQF     (124,8),(172,8) COMPARE PX 2 TO PX 7
    01E5 01E5 00008841
    01E6 01E6 3F0701E9
    01E7 01E7 43A488A5
    01E8 01E8 433400A5
    01E9 01E9 00002243
  6 01EA 01EA 48000002        L       M,Y
  7 01EB 01EB 75E0037B        ADC     (120,4),(28,4),(120,4)     INCR TAG?
    01EC 01EC 73C01F7B
    01ED 01ED 37200303
    01EE 01EE 2C000000
  8 01EF 01EF 4000BBA1        SET     M
    01F0 01F0 48000003
  9 01F1 01F1 779083BF        EQF     (124,8),(184,8) COMPARE PX 2 TO PX 8
    01F2 01F2 00008841
    01F3 01F3 3F0701F6
    01F4 01F4 43A488A5
    01F5 01F5 433400A5
    01F6 01F6 00002243
 10 01F7 01F7 48000002        L        M,Y
 11 01F8 01F8 75E0037B        ADC     (120,4),(28,4),(120,4)     INCR TAG?
    01F9 01F9 73C01F7B
    01FA 01FA 37200303
    01FB 01FB 2C000000
 12 01FC 01FC 4000BBA1        SET     M
    01FD 01FD 48000003
 13 01FE 01FE 7790835F        EQF     (124,8),(88,8)  COMPARE PX 2 TO PX 9
    01FF 01FF 00008841
    0200 0200 3F070203
    0201 0201 43A488A5
    0202 0202 433400A5
    0203 0203 00002243
 14 0204 0204 48000002        L       M,Y
 15 0205 0205 75E0037B        ADC     (120,4),(28,4),(120,4)     INCR TAG?
    0206 0206 73C01F7B
    0207 0207 37200303
    0208 0208 2C000000
 16 0209 0209 4000BBA1        SET     M
    020A 020A 48000003
 17 020B 020B 77908F9B        EQF     (136,8),(148,8) COMP'JE PX 3 TO PX 4
    020C 020C 00008841
    020D 020D 3F070210
    020E 020E 43A488A5
    020F 020F 433400A5
    0210 0210 00002243
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00015 V
    1 0211 0211 48000002        L       M,Y
    2 0212 0212 75E00387        ADC     (132,4),(28,4),(132,4)     INCR TAG?
      0213 0213 73C01F87
      0214 0214 37200303
      0215 0215 2C000000
    3 0216 0216 4000BBA1        SET     M
      0217 0217 48000003
    4 0218 0218 77908FA7        EQF     (136,8),(160,8) COMPARE PX 3 TO PX 6
      0219 0219 00008841
      021A 021A 3F07021D
      021B 021B 43A488A5
      021C 021C 433400A5
      021D 021D 00002243
    5 021E 021E 48000002        L       M,Y
    6 021F 021F 75E00387        ADC     (132,4),(28,4),(132,4)     INCR TAG?
      0220 0220 73C01F87
      0221 0221 37200303
      0222 0222 2C000000
    7 0223 0223 4000BBA1        SET     M
      0224 0224 48000003
    8 0225 0225 77908FB3        EQF     (136,8),(172,8) COMPARE PX 3 TO PX 7
      0226 0226 00008841
      0227 0227 3F07022A
      0228 0228 43A488A5
      0229 0229 433400A5
      022A 022A 00002243
    9 022B 022B 48000002        L       M,Y
   10 022C 022C 75E00387        ADC     (132,4),(28,4),(132,4)     INCR TAG?
      022D 022D 73C01F87
      022E 022E 37200303
      022F 022F 2C000000
   11 0230 0230 4000BBA1        SET     M
      0231 0231 48000003
   12 0232 0232 77908FBF        EQF     (136,8),(184,8) COMPARE PX 3 TO PX 8
      0233 0233 00008841
      0234 0234 3F070237
      0235 0235 43A488A5
      0236 0236 433400A5
      0237 0237 00002243
   13 0238 0238 48000002        L       M,Y
   14 0239 0239 75E00387        ADC     (132,4),(28,4),(132,4)     INCR TAG?
      023A 023A 73C01F87
      023B 023B 37200303
      023C 023C 2C000000
   15 023D 023D 4000BBA1        SET     M
      023E 023E 48000003
   16 023F 023F 77908F5F        EQF     (136,8),(88,8)  COMPARE PX 3 TO PX 9
      0240 0240 00008841
      0241 0241 3F070244
      0242 0242 43A488A5
      0243 0243 433400A5
      0244 0244 00002243
   17 0245 0245 48000002        L       M,Y
   18 0246 0246 75E00387        ADC     (132,4),(28,4),(132,4)     INCR TAG?
      0247 0247 73C01F87
      0248 0248 37200303
      0249 0249 2C000000
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00016 V
    1 024A 024A 4000BBA1        SET     M
      024B 024B 48000003
    2 024C 024C 77909BA7        EQF     (148,8),(160,8) COMPARE PX 4 TO PX 6
      024D 024D 00008841
      024E 024E 3F070251
      024F 024F 43A488A5
      0250 0250 433400A5
      0251 0251 00002243
    3 0252 0252 48000002        L       M,Y
    4 0253 0253 75E00393        ADC     (144,4),(28,4),(144,4)     INCR TAG?
      0254 0254 73C01F93
      0255 0255 37200303
      0256 0256 2C000000
    5 0257 0257 4000BBA1        SET     M
      0258 0258 48000003
    6 0259 0259 77909BB3        EQF     (148,8),(172,8) COMPARE PX 4 TO PX 7
      025A 025A 00008841
      025B 025B 3F07025E
      025C 025C 43A488A5
      025D 025D 433400A5
      025E 025E 00002243
    7 025F 025F 48000002        L       M,Y
    8 0260 0260 75E00393        ADC     (144,4),(28,4),(144,4)     INCR TAG?
      0261 0261 73C01F93
      0262 0262 37200303
      0263 0263 2C000000
    9 0264 0264 4000BBA1        SET     M
      0265 0265 48000003
   10 0266 0266 77909BBF        EQF     (148,8),(184,8) COMPARE PX 4 TO PX 8
      0267 0267 00008841
      0268 0268 3F07026B
      0269 0269 43A488A5
      026A 026A 433400A5
      026B 026B 00002243
   11 026C 026C 48000002        L       M,Y
   12 026D 026D 75E00393        ADC     (144,4),(28,4),(144,4)     INCR TAG?
      026E 026E 73C01F93
      026F 026F 37200303
      0270 0270 2C000000
   13 0271 0271 4000BBA1        SET     M
      0272 0272 48000003
   14 0273 0273 77909B5F        EQF     (148,8),(88,8)  COMPARE PX 4 TO PX 9
      0274 0274 00008841
      0275 0275 3F070278
      0276 0276 43A488A5
      0277 0277 433400A5
      0278 0278 00002243
   15 0279 0279 48000002        L       M,Y
   16 027A 027A 75E00393        ADC     (144,4),(28,4),(144,4)     INCR TAG?
      027B 027B 73C01F93
      027C 027C 37200303
      027D 027D 2C000000
   17 027E 027E 4000BBA1        SET     M
      027F 027F 08000003
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 0/017 V
    1 0280 0280 7790A7B3        EQF     (160,8) (172,8) COMPARE PX 6 TO PX 7
      0281 0281 00008841
      0282 0282 3F070285
      0283 0283 43A488A5
      0284 0284 433400A5
      0285 0285 00002243
    2 0286 0286 48000002        L       M,Y
    3 0287 0287 75E0039F        ADC     (156,4),(28,4),(156,4)     INCR TAG?
      0288 0288 73C01F9F
      0289 0289 37200303
      028A 028A 2C000000
    4 028B 028B 4000BBA1        SET     M
      028C 028C 48000003
    5 028D 028D 7790A7BF        EQF     (160,8),(184,8) COMPARE PX 6 TO PX 8
      028E 028E 00008841
      028F 028F 3F070292
      0290 0290 43A488A5
      0291 0291 433400A5
      0292 0292 00002243
    6 0293 0293 48000002        L       M,Y
    7 0294 0294 75E0039F        ADC     (156,4),(28,4),(156,4)     INCR TAG?
      0295 0295 73C01F9F
      0296 0296 37200303
      0297 0297 2C000000
    8 0298 0298 4000BBA1        SET     M
      0299 0299 48000003
    9 029A 029A 7790A75F        EQF     (160,8),(88,8)  COMPARE PX 6 TO PX 9
      029B 029B 00008841
      029C 029C 3F07029F
      029D 029D 43A488A5
      029E 029E 433400A5
      029F 029F 00002243
   10 02A0 02A0 48000002        L       M,Y
   11 02A1 02A1 75E0039F        ADC     (156,4),(28,4),(156,4)     INCR TAG?
      02A2 02A2 73C01F9F
      02A3 02A3 37200303
      02A4 02A4 2C000000
   12 02A5 02A5 4000BBA1        SET     M
      02A6 02A6 48000003
   13 02A7 02A7 7790B3BF        EQF     (172,8),(184,8) COMPARE PX 7 TO PX 8
      02A8 02A8 00008841
      02A9 02A9 3F0702AC
      02AA 02AA 43A488A5
      02AB 02AB 433400A5
      02AC 02AC 00002243
   14 02AD 02AD 48000002        L       M,Y
   15 02AE 02AE 75E003AB        ADC     (168,4),(28,4),(168,4)     INCR TAG?
      02AF 02AF 73C01FAB
      02B0 02B0 37200303
      02B1 02B1 2C000000
   16 02B2 02B2 4000BBA1        SET     M
      02B3 02B3 48000003
   17 02B4 02B4 7790B35F        EQF     (172,8),(88,8)  COMPARE PX 7 TO PX 9
      02B5 02B5 00008841
      02B6 02B6 3F0702B9
      02B7 02B7 43A488A5
      02B8 02B8 433400A5
      02B9 02B9 00002243
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00018 V
    1 02BA 02BA 48000002        L       M,Y
    2 02BB 02BB 75E003AB        ADC     (168,4),(28,4),(168,4)     INCR TAG?
      02BC 02BC 73C01FAB
      02BD 02BD 37200303
      02BE 02BE 2C000000
    3 02BF 02BF 4000BBA1        SET     M .
      02C0 02C0 48000003
    4 02C1 02C1 7790BF5F        EQF     (184,8),(88,8)  COMPARE PX 8 TO PX 9
      02C2 02C2 00008841
      02C3 02C3 3F0702C6
      02C4 02C4 43A488A5
      02C5 02C5 433400A5
      02C6 02C6 00002243
    5 02C7 02C7 48000002        L       M,Y
    6 02C8 02C8 75E003B7        ADC     (180,4),(28,4),(180,4)     INCR TAG?
      02C9 02C9 73C01FB7
      02CA 02CA 37200303
      02CB 02CB 2C000000
```

```
 1                        ;
 2                        ;       SORT PIXELS BY FREQUENCY COUNT TAG
 3                        ;
 4 02CC 02CC 4000BBA1          SET     M
   02CD 02CD 48000003
 5 02CE 02CE 77907B6F          LTF     (108,4),(120,4) PX 1 TAG < PX 2 TAG
   02CF 02CF 35700001
   02D0 02D0 2C010000
 6 02D1 02D1 48000002          L       M,Y
 7 02D2 02D2 37907783          MVF     (120,12),(108,12)       IF SO, SWAP
   02D3 02D3 3F0B02D6
   02D4 02D4 433488A5
   02D5 02D5 4B800001
   02D6 02D6 13A40003
 8 02D7 02D7 4000BBA1          SET     M
   02D8 02D8 48000003
 9 02D9 02D9 7790876F          LTF     (108,4),(132,4) PX 1 TAG < PX 3 TAG
   02DA 02DA 35700001
   02DB 02DB 2C010000
10 02DC 02DC 48000002          L       M,Y
11 02DD 02DD 3790778F          MVF     (132,12),(108,12)       IF SO, SWAP
   02DE 02DE 3F0B02E1
   02DF 02DF 433488A5
   02E0 02E0 4B800001
   02E1 02E1 13A40003
12 02E2 02E2 4000BBA1          SET     M
   02E3 02E3 48000003
13 02E4 02E4 7790936F          LTF     (108,4),(144,4) PX 1 TAG < PX 4 TAG
   02E5 02E5 35700001
   02E6 02E6 2C010000
14 02E7 02E7 48000002          L       M,Y
15 02E8 02E8 3790779B          MVF     (144,12),(108,12)       IF SO, SWAP
   02E9 02E9 3F0B02EC
   02EA 02EA 433488A5
   02EB 02EB 4B800001
   02EC 02EC 13A40003
16 02ED 02ED 4000BBA1          SET     M
   02EE 02EE 48000003
17 02EF 02EF 77909F6F          LTF     (108,4),(156,4) PX 1 TAG < PX 6 TAG
   02F0 02F0 35700001
   02F1 02F1 2C010000
18 02F2 02F2 48000002          L       M,Y
19 02F3 02F3 379077A7          MVF     (156,12),(108,12)       IF SO, SWAP
   02F4 02F4 3F0B02F7
   02F5 02F5 433488A5
   02F6 02F6 4B800001
   02F7 02F7 13A40003
20 02F8 02F8 4000BBA1          SET     M
   02F9 02F9 48000003
21 02FA 02FA 7790AB6F          LTF     (108,4),(168,4) PX 1 TAG < PX 7 TAG
   02FB 02FB 35700001
   02FC 02FC 2C010000
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00020 V
    1 02FD 02FD 48000002        L       M,Y
    2 02FE 02FE 379077B3        MVF     (168,12),(108,12)       IF SO, SWAP
      02FF 02FF 3F0B0302
      0300 0300 433488A5
      0301 0301 4B800001
      0302 0302 13A40003
    3 0303 0303 4000BBA1        SET     M
      0304 0304 48000003
    4 0305 0305 7790B76F        LTF     (108,4),(180,4) PX 1 TAG < PX 8 TAG
      0306 0306 35700001
      0307 0307 2C010000
    5 0308 0308 48000002        L       M,Y
    6 0309 0309 379077BF        MVF     (180,12),(108,12)       IF SO, SWAP
      030A 030A 3F0B030D
      030B 030B 433488A5
      030C 030C 4B800001
      030D 030D 13A40003
```

```
                        ;
                        ;       SET UP OUTPUT LINE IN ARRAY
                        ;
 030E 030E 4000BBA1             SET     M                   LOAD ARRAY OUTPUT
 030F 030F 48000003
 0310 0310 3790C76B             MVF     (100,8),(192,8)     LINE WITH PX 5
 0311 0311 3F070314
 0312 0312 433488A5
 0313 0313 4B800001
 0314 0314 13A40003
 0315 0315 77906F63             LTF     (96,4),(108,4)      PX 5 TAG < PX 1 TAG
 0316 0316 35700001
 0317 0317 2C010000
 0318 0318 48000002             L       M,Y
 0319 0319 3790C777             MVF     (112,8),(192,8)     IF SO, NEW PIXEL
 031A 031A 3F07031D
 031B 031B 433488A5
 031C 031C 4B800001
 031D 031D 13A40003
 031E 031E 73C00000             LI      FP12,0              ARRAY WORD POINTER
 031F 031F 47C088A5             LCM     (0,8),(100,8)       DO NOT ALTER
 0320 0320 401C88BB
 0321 0321 401088BB
 0322 0322 65C3803B
 0323 0323 400088A1             SCW     (0,8),(192,8)       FIRST PX
 0324 0324 4FC6A03F
 0325 0325 57C60000
 0326 0326 48000003
 0327 0327 73C001FF             LI      FP12,X'01FF'        AND
 0328 0328 47C088A5             LCM     (0,8),(100,8)
 0329 0329 401C88BB
 032A 032A 401088BB
 032B 032B 65C3803B
 032C 032C 400088A1             SCW     (0,8),(192,8)       LAST PX OF LINE
 032D 032D 4FC6A03F
 032E 032E 57C60000
 032F 032F 08000003
```

```
    1                       ;
    2                       ;       MOVE PROCESSED LINE TO OBUFF WITH
    3                       ;       PIXEL SWAP (1,2,3,4 TO 4,3,2,1)
    4                       ;
    5 0330 0330 33C00000            LI      FP12,0          ARRAY WORD POINTER
    6 0331 0331 32810610            LR      DP,OBDP
    7 0332 0332 34810612            LR      BL,OBEF
    8 0333 0333 3F7F033E            LOOP,SP LNOUT           MOVE LINE TO OBUFF
    9 0334 0334 47C088A5            LCM     (24,8),(192,8)  LOAD C REG WITH PX
      0335 0335 401888B3
      0336 0336 25C698FB
   10 0337 0337 01E00001            INCR    FP12            NEXT WORD
   11 0338 0338 27C690BD            LCM     (16,8),(192,8)  LOAD C REG WITH PX
   12 0339 0339 01E00001            INCR    FP12            NEXT WORD
   13 033A 033A 27C6687D            LCM     (8,8),(192,8)   LOAD C REG WITH PX
   14 033B 033B 01E00001            INCR    FP12            NEXT WORD
   15 033C 033C 27C6A03D            LCM     (0,8),(192,8)   LOAD C REG WITH PX
   16 033D 033D 01E00001            INCR    FP12            NEXT WORD
   17 033E 033E 3005B000 LNOUT      SR      C,OBUFF(DP),3   STORE 4 PX IN OBUFF
W  18 033F 033F 30810610            SR      DP,OBDP
W  19 0340 0340 30810612            SR      BL,OBEF
   20                       ;
   21                       ;       SHIFT TWO IMAGE LINES OVER A FIELD IN
   22                       ;       THE ARRAYS TO PREPARE FOR A NEW LINE.
   23                       ;
   24 0341 0341 4000BBA1            SET     M
      0342 0342 48000003
   25 0343 0343 3790070F            MVF     (8,8),(0,8)     2ND FIELD TO 1ST
      0344 0344 3F070347
      0345 0345 433488A5
      0346 0346 4B800001
      0347 0347 13A40003
   26 0348 0348 37900F17            MVF     (16,8),(8,8)    3RD FIELD TO 2ND
      0349 0349 3F07034C
      034A 034A 433488A5
      034B 034B 4B800001
      034C 034C 13A40003
```

```
    1                       ;
    2                       ;       IF OBUFF IS FULL OUTPUT TO COMTAL
    3                       ;
    4 034D 034D 34810612            LR      BL,OBEF         IS OBUFF FULL?
    5 034E 034E 2911035E            BNZ,BL  OBNF            IF NOT CHECK IBUFF
    6                               TRAN    TRAN1           IF SO, OUTPUT
      034F 034F 72800000
      0350 0350 34A00016
      0351 0351 30C18010
    8                               IOWAIT  LINKWD1
      0352 0352 72800000
      0353 0353 74A00000
      0354 0354 37200000
      0355 0355 30C18010
    9 0356 0356 36810001            LR      (BL,DP),TRAN1+1 UPDATE TRAN OUT
   10 0357 0357 3E1F0358            RPT,LOB
   11 0358 0358 28140001            INCR    DP
   12 0359 0359 30810001            SR      (BL,DP),TRAN1+1
   13 035A 035A 32800000            LI      DP,0            RE-INITIALIZE
W  14 035B 035B 30810610            SR      DP,OBDP         OBUFF DATA POINTER
   15 035C 035C 34A01000            LI      BL,MXOBDP       AND
W  16 035D 035D 30810612            SR      BL,OBEF         OBUFF EMPTY FLAG
   17                       ;
   18                       ;       IF IBUFF IS NOT EMPTY GET NEXT LINE
   19                       ;
   20 035E 035E 34810613 OBNF       LR      BL,IBEF         IS IBUFF EMPTY?
   21 035F 035F 29110072            BNZ,BL  NXLINE          IF NOT, NEXT LINE
   22                       ;
   23                       ;       IF ENTIRE IMAGE HAS NOT BEEN INPUT
   24                       ;       MOVE MORE DATA INTO IBUFF
   25                       ;
   26 0360 0360 34810614            LR      BL,LIF          HAS ENTIRE IMAGE
   27 0361 0361 01030001            DECR    BL              BEEN INPUT
W  28 0362 0362 30810614            SR      BL,LIF
   29 0363 0363 29010370            BZ,BL   DONE            IF SO, GO TO DONE
   30                               TRAN    TRAN2           IF NOT, INPUT
      0364 0364 72800000
      0365 0365 34A00016
      0366 0366 30C18010
   32                               IOWAIT  LINKWD2         MORE IMAGE
      0367 0367 72800000
      0368 0368 74A00000
      0369 0369 37200000
      036A 036A 30C18010
   33 036B 036B 32800000            LI      DP,0            RE-INITIALIZE
W  34 036C 036C 30810611            SR      DP,IBDP         IBUFF DATA POINTER
   35 036D 036D 34A01000            LI      BL,MXIBDP       AND
W  36 036E 036E 30810613            SR      BL,IBEF         IBUFF EMPTY FLAG
   37 036F 036F 28010072            B       NXLINE          PROCESS NEXT LINE
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00024 V
    1                        ;
    2                        ;       MOVE LAST LINE TO OBUFF AND
    3                        ;       OUTPUT OBUFF TO COMTAL
    4                        ;
    5 0370 0370 3F7F037D DONE    LOOP,SP LTLINE          MOVE LINE TO OBUFF
    6 0371 0371 32810611         LR      DP,IBDP
    7 0372 0372 3604A000         LR      C,IBUFF(DP),2   LOAD 4 PX IN C REG
W   8 0373 0373 30810611         SR      DP,IBDP
    9 0374 0374 400077A1         CLR     X               PIXEL SWAP
   10 0375 0375 420099A0         SC      X(0)            1,2,3,4 TO 4,3,2,1
   11 0376 0376 400888BB         ROT     X,-8,16
      0377 0377 400088BB
   12 0378 0378 40108BBB         ROT     X,-16,32
      0379 0379 40008BBB
   13 037A 037A 21C0A0FB         LCW     X(0)
   14 037B 037B 32810610         LR      DP,OBDP
   15 037C 037C 3004B000         SR      C,OBUFF(DP),2   STORE 4 PX IN OBUFF
W  16 037D 037D 30810610 LTLINE SR      DP,OBDP
   17                            TRAN    TRAN1           OUTPUT FINAL OBUFF
      037E 037E 72800000
      037F 037F 34A00016
      0380 0380 30C18010
   19                            IOWAIT  LINKWD1
      0381 0381 72800000
      0382 0382 74A00000
      0383 0383 37200000
      0384 0384 30C18010
   20                            RLSE    LINKWD1
      0385 0385 72800000
      0386 0386 34A00018
      0387 0387 30C18010
   22                            RLSE    LINKWD2
      0388 0388 72800000
      0389 0389 34A00018
      038A 038A 30C18010
   24 038B 038B 38002000 ERR     WAIT
   25                0610        ORG     X'0610',A       HIGH SPEED DATA BUFFER
   26                0610 OBDP   DS                      OBUFF POINTER STORAGE
   27                0611 IBDP   DS                      IBUFF POINTER STORAGE
   28                0612 OBEF   DS                      OBUFF EMPTY FLAG
   29                0613 IBEF   DS                      IBUFF EMPTY FLAG
   30                0614 LIF    DS                      LAST IBUFF FLAG
   31                0000        END     MODAL
```

```
MODAL     APPLE V04-00 24-JUL-80 20:17:30 PAGE 00025 V
ERRORS DETECTED: 00000
WARNINGS DETECTED: 00018
```

APPENDIX C

ODDPX PROGRAM

```
                ;
                ;
                ;
                ODDPX  START
                       EXTRN   LINKBK1,LINKBK2,TRAN1,TRAN2
                       EXTRN   LINKWD1,LINKWD2
                       ENTRY   ERRTN1,ERRTN2,ERRTN3
                       ENTRY   ERRTN4,ERRTN5
                ;
                ;
                ;
                ;      DAVID M. CRAWFORD
                ;      RESEARCH ASSISTANT
                ;      ELECTRICAL ENGINEERING DEPT.
                ;      UNIVERSITY OF MISSOURI - COLUMBIA
                ;      16 MAY 1980
                ;      REVISION: 16 JULY 1980
                ;      REVISION: 22 JULY 1980
                ;
                ;
                ;      THIS PROGRAM IS DESIGNED TO PERFORM NOISE
                ;      REDUCTION ON IMAGES BY USING AN ODD PIXEL
                ;      REPLACEMENT TECHNIQUE.  THE IMAGE, 512 X 512
                ;      PIXELS, IS READ FROM MAGNETIC TAPE AND PRO-
                ;      CESSED BY STARAN.  THE NEW IMAGE IS THEN
                ;      OUTPUT TO THE COMTAL DISPLAY.
                ;
                ;      ODD PIXEL REPLACEMENT-
                ;
                ;      THIS TECHNIQUE USES A 3X3 NEIGHBORHOOD.
                ;      TWO MODES OF OPERATION ARE USED:
                ;      MODE 0 - THE NEIGHBORS IN THE NEIGHBORHOOD
                ;               ARE AVERAGED BY ADDING THEM TOGETHER
                ;               AND DIVIDING BY EIGHT.  IF THIS
                ;               AVERAGE DIFFERS FROM THE CENTER
                ;               PIXEL OF THE NEIGHBORHOOD BY MORE
                ;               THAN A USER SPECIFIED THRESHOLD
                ;               (THRES) THE CENTER PIXEL IS REPLACED
                ;               BY THE AVERAGE.
                ;      MODE 1 - EIGHT NEIGHBORING PIXELS ARE EACH
                ;               COMPARED TO THE CENTER PIXEL.  THE
                ;               NUMBER OF NEIGHBORS THAT EXCEED
                ;               A USER SPECIFIED THRESHOLD(THRES)
                ;               ARE DETERMINED.  IF THIS NUMBER
                ;               EQUALS OR EXCEEDS THE AMOUNT
                ;               SPECIFIED BY THE USER (NON) THE
                ;               CENTER PIXEL WILL BE REPLACED BY
                ;               THE AVERAGE OF THOSE NEIGHBORS
                ;               EXCEEDING THE THRESHOLD.
                ;
                ;
```

```
;
;       THE PROGRAM OPERATES WITH THREE LINES STORED
;       IN ARRAYS 0 AND 1, FIELDS (0,9), (9,9), (18,9).
;       THE FIRST PIXEL OF THE FIRST LINE IS STORED IN
;       FIELD (0,9) OF WORD 0 IN ARRAY 0.  THE LAST
;       PIXEL OF THE FIRST LINE IS IN FIELD (0,9) OF
;       WORD 255 IN ARRAY 1.  THE SECOND LINE IS IN
;       FIELD (9,9) AND THE THIRD IN FIELD (18,9).
;       THE ODD PIXEL VALUE IS DETERMINED FOR THE
;       LINE IN FIELD(9,9).
;
;
;       THE FOLLOWING BUFFERS ARE SET UP IN THE
;       ARRAYS IN THE SAME WORD WHICH CONTAINS
;       THE PIXEL BEING CHECK TO DETERMINE IF
;       IT IS NOISE.
;
;
;       COMPARISON BUFFER     CMPBUF  (27,54)
;       SUM BUFFER            SUMBUF  (81,12)
;       DIFFERENCE BUFFER     DIFBUF  (93,9)
;       COUNT BUFFER          CNTBUF  (102,5)
;       OUTPUT BUFFER         OUTBUF  (107,8)
;
;
;       CMPBUF - USED TO STORE NEIGHBORS IN THE
;                SAME ARRAY WORD AS THE CENTER
;                PIXEL FOR COMPARISON IN MODE 0
;                AND MODE 1.
;       SUMBUF - USED TO STORE THE SUM OF ALL
;                EIGHT NEIGHBORS FOR MODE 0 OR
;                THE SUM OF ALL NEIGHBORS WHICH
;                DIFFER FROM THE CENTER PIXEL
;                BY MORE THAN THE THRESHOLD
;                FOR MODE 1.
;       DIFBUF - USED TO STORE THE DIFFERENCE
;                BETWEEN THE AVERAGE AND THE
;                CENTER PIXEL FOR MODE 0 OR
;                THE DIFFERENCE BETWEEN A NEIGH-
;                BORING PIXEL AND THE CENTER
;                PIXEL FOR MODE 1.
;       CNTBUF - USED TO STORE THE NUMBER OF
;                NEIGHBORS THAT DIFFER FROM THE
;                CENTER PIXEL BY MORE THAN THE
;                USER SPECIFIED THRESHOLD(THRES)
;                FOR MODE 1.
;       OUTBUF - USED TO STORE THE PROCESSED
;                PIXEL TO BE OUTPUT FOR MODE 0
;                AND MODE 1.
;
;       ALL EDGE POINTS ARE OUTPUT UNALTERED.
;
```

```
                 024C ERRTN1 EQU     ERR              ERROR RETURN
                 024C ERRTN2 EQU     ERR              ERROR RETURN
                 024C ERRTN3 EQU     ERR              ERROR RETURN
                 024C ERRTN4 EQU     ERR              ERROR RETURN
                 024C ERRTN5 EQU     ERR              ERROR RETURN
                 0000 BLKNUM EQU     0                I/O BLOCK NUMBER
                 A000 IBUFF  EQU     X'A000'          IBUFF ADDRESS
                 B000 OBUFF  EQU     X'B000'          OBUFF ADDRESS
                 1000 IBSIZE EQU     4096             INPUT BUFFER SIZE
                 1000 OBSIZE EQU     4096             OUTPUT BUFFER SIZE
                 0020 LIB    EQU     IBSIZE/128       IMAGE LINES IN IBUFF
                 0020 LOB    EQU     OBSIZE/128       IMAGE LINES IN OBUFF
                 0010 BLKS   EQU     512/LIB          NO. OF INPUT BLOCKS
                 0080 SP     EQU     128              32 BIT SEGMENTS PER LIN
                 0003 NON    EQU     3                NUMBER OF NEIGHBORS
                 0001 MODE   EQU     1                MODE 0 = 0,  MODE 1 = 1
                 0019 THRES  EQU     25               THRESHOLD
                 1000 MXOBDP EQU     OBSIZE           MAX OBUFF DP VALUE
                 1000 MXIBDP EQU     IBSIZE           MAX IBUFF DP VALUE
                 0000 ODDPX  EQU     $
0000 0000 3660C000       LI,2    AS,X'C000'       SELECT ARRAYS 0 AND 1
                     ;
                     ;   INITIALIZE INPUT -- MAG TAPE
                     ;
0001 0001 74201000       LI      CH,IBSIZE
0002 0002 32000000       LI      CL,BLKNUM
0003 0003 30010001       SR      C,TRAN2+1
0004 0004 74200000       LI      CH,0
0005 0005 3200A000       LI      CL,IBUFF
0006 0006 30010003       SR      C,TRAN2+3
                         INIT    LINKBK2
0007 0007 72800000
0008 0008 34A00014
0009 0009 30C18010
                     ;
                     ;   INITIALIZE OUTPUT -- COMTAL
                     ;
000A 000A 74201000       LI      CH,OBSIZE
000B 000B 32000000       LI      CL,BLKNUM
000C 000C 30010001       SR      C,TRAN1+1
000D 000D 74200000       LI      CH,0
000E 000E 3200B000       LI      CL,OBUFF
000F 000F 30010003       SR      C,TRAN1+3
                         INIT    LINKBK1
0010 0010 72800000
0011 0011 34A00014
0012 0012 30C18010
```

```
    1                       ;
    2                       ;       INPUT LINES TO IBUFF (NO. = LIB)
    3                       ;
    4                               TRAN    TRAN2
      0013 0013 72800000
      0014 0014 34A00016
      0015 0015 30C18010
    6                               IOWAIT  LINKWD2
      0016 0016 72800000
      0017 0017 74A00000
      0018 0018 37200000
      0019 0019 30C18010
    7                       ;
    8                       ;       INITIALIZE BUFFER POINTERS
    9                       ;
   10 001A 001A 34A01000            LI      BL,MXIBDP       INIT IBUFF EMPTY FLAG
W  11 001B 001B 30810613            SR      BL,IBEF
   12 001C 001C 34A01000            LI      BL,MXOBDP       INIT OBUFF EMPTY FLAG
W  13 001D 001D 30810612            SR      BL,OBEF
   14 001E 001E 34A00010            LI      BL,BLKS         INIT LAST IBUFF FLAG
W  15 001F 001F 30810614            SR      BL,LIF
   16                       ;
   17                       ;       MOVE FIRST LINE IN IBUFF TO OBUFF
   18                       ;
   19 0020 0020 73C00000            LI      FP12,0
   20 0021 0021 32800000            LI      DP,0
   21 0022 0022 34810612            LR      BL,OBEF
   22 0023 0023 3F7F002C            LOOP,SP LINE1           LINE ONE TO IBUFF
   23 0024 0024 3602A000            LR      C,IBUFF(DP)     LOAD 4 PX IN C REG
   24 0025 0025 400077A1            CLR     X               PIXEL SWAP
   25 0026 0026 420099A0            SC      X(0)            1,2,3,4 TO 4,3,2,1
   26 0027 0027 400888BB            ROT     X,-8,16
      0028 0028 400088BB
   27 0029 0029 401088BB            ROT     X,-16,32
      002A 002A 400088BB
   28 002B 002B 21C0A0FB            LCW     X(0)
   29 002C 002C 3005B000 LINE1      SR      C,OBUFF(DP),3   STORE 4 PX IN OBUFF
W  30 002D 002D 30810610            SR      DP,OBDP
W  31 002E 002E 30810612            SR      BL,OBEF
   32                       ;
   33                       ;       CLEAR FIELD (0,32) FOR USE
   34                       ;       BY THE INPUT DATA
   35                       ;
   36 002F 002F 74200000            LI      CH,0
   37 0030 0030 72000000            LI      CL,0
   38 0031 0031 33C01F1F            SC      (0,32),,0,32)
      0032 0032 3F1F0034
      0033 0033 4B40B7A1
      0034 0034 13740003
```

```
ODDPX     APPLE V04-00 24-JUL-80 21:05:11 PAGE 00005 V
  1                        ;
  2                        ;       MOVE FIRST TWO LINES IN
  3                        ;       IBUFF TO THE ARRAYS
  4                        ;
  5 0035 0035 32800000             LI      DP,0
  6 0036 0036 34810613             LR      BL,IBEF
  7 0037 0037 33C00000             LI      FP12,0          ARRAY WORD POINTER
  8 0038 0038 3F7F0059             LOOP,SP L1              FIRST LINE IN ARRAY
  9 0039 0039 3605A000             LR      C,IBUFF(DP),3   LOAD 4 PX IN C REG
 10 003A 003A 400088A1             SCW     (0,8),(1,8)     PX TO ARRAY
    003B 003B 4FC0A147
    003C 003C 42008840
    003D 003D 40FF8852
    003E 003E 57C00002
    003F 003F 08000003
 11 0040 0040 01E00001             INCR    FP12            NEXT WORD
 12 0041 0041 400088A1             SCW     (8,8),(1,8)     PX TO ARRAY
    0042 0042 4FC0A147
    0043 0043 42008840
    0044 0044 40F8885A
    0045 0045 40FF885A
    0046 0046 57C00002
    0047 0047 08000003
 13 0048 0048 01E00001             INCR    FP12            NEXT WORD
 14 0049 0049 400088A1             SCW     (16,8),(1,8)    PX TO ARRAY
    004A 004A 4FC0A147
    004B 004B 42008840
    004C 004C 40F0885A
    004D 004D 40FF885A
    004E 004E 57C00002
    004F 004F 08000003
 15 0050 0050 01E00001             INCR    FP12            NEXT WORD
 16 0051 0051 400088A1             SCW     (24,8),(1,8)    PX TO ARRAY
    0052 0052 4FC0A147
    0053 0053 42008840
    0054 0054 40E0885A
    0055 0055 40FF885A
    0056 0056 40F88852
    0057 0057 57C00002
    0058 0058 08000003
 17 0059 0059 01E00001 L1          INCR    FP12            NEXT WORD
```

```
ODDPX     APPLE V04-00 24-JUL-80 21:05:11 PAGE 00006 V
    1 005A 005A 33C00000          LI      FP12,0            ARRAY WORD POINTER
    2 005B 005B 3F7F007B          LOOP,SP L2                2ND LINE IN ARRAY
    3 005C 005C 3605A000          LR      C,IBUFF(DP),3     LOAD 4 PX IN C REG
    4 005D 005D 400088A1          SCW     (0,8),(10,8)      PX TO ARRAY
      005E 005E 4FC0AA8F
      005F 005F 42008840
      0060 0060 40FE8852
      0061 0061 40F88852
      0062 0062 57C00002
      0063 0063 08000003
    5 0064 0064 01E00001          INCR    FP12              NEXT WORD
    6 0065 0065 400088A1          SCW     (8,8),(10,8)      PX TO ARRAY
      0066 0066 4FC0AA8F
      0067 0067 42008840
      0068 0068 40FE8852
      0069 0069 57C00002
      006A 006A 08000003
    7 006B 006B 01E00001          INCR    FP12              NEXT WORD
    8 006C 006C 400088A1          SCW     (16,8),(10,8)     PX TO ARRAY
      006D 006D 4FC0AA8F
      006E 006E 42008840
      006F 006F 40F8885A
      0070 0070 40FE885A
      0071 0071 57C00002
      0072 0072 08000003
    9 0073 0073 01E00001          INCR    FP12              NEXT WORD
   10 0074 0074 400088A1          SCW     (24,8),(10,8)     PX TO ARRAY
      0075 0075 4FC0AA8F
      0076 0076 42008840
      0077 0077 40F0885A
      0078 0078 40FE885A
      0079 0079 57C00002
      007A 007A 08000003
   11 007B 007B 01E00001 L2       INCR    FP12              NEXT WORD
W  12 007C 007C 30810611          SR      DP,IBDP
W  13 007D 007D 30810613          SR      BL,IBEF
```

```
ODDPX     APPLE V04-00 24-JUL-80 21:05:11 PAGE 00007 V
    1                        ;
    2                        ;      MOVE ANOTHER LINE FROM IBUFF TO THE ARRAY
    3                        ;
    4 007E 007E 32810611 NXLINE LR      DP,IBDP
    5 007F 007F 34810613        LR      BL,IBEF
    6 0080 0080 33C00000        LI      FP12,0            ARRAY WORD POINTER
    7 0081 0081 3F7F00A5        LOOP,SP LINE              MOVE LINE IN APRAY
    8 0082 0082 3605A000        LR      C,IBUFF(DP),3     LOAD 4 PX IN C REG
    9 0083 0083 400088A1        SCW     (0,8),(19,8)      PX TO ARRAY
      0084 0084 4FC0B3D7
      0085 0085 42008840
      0086 0086 40FC8852
      0087 0087 40FF885A
      0088 0088 40F0885A
      0089 0089 57C00002
      008A 008A 08000003
   10 008B 008B 01E00001        INCR    FP12              NEXT WORD
   11 008C 008C 400088A1        SCW     (8,8),(19,8)      PX TO ARRAY
      008D 008D 4FC0B3D7
      008E 008E 42008840
      008F 008F 40FC885A
      0090 0090 40FF8852
      0091 0091 40F0885A
      0092 0092 57C00002
      0093 0093 08000003
   12 0094 0094 01E00001        INCR    FP12              NEXT WORD
   13 0095 0095 400088A1        SCW     (16,8),(19,8)     PX TO ARRAY
      0096 0096 4FC0B3D7
      0097 0097 42008840
      0098 0098 40FF885A
      0099 0099 40FC885A
      009A 009A 57C00002
      009B 009B 08000003
   14 009C 009C 01E00001        INCR    FP12              NEXT WORD
   15 009D 009D 400088A1        SCW     (24,8),(19,8)     PX TO ARRAY
      009E 009E 4FC0B3D7
      009F 009F 42008840
      00A0 00A0 40FC885A
      00A1 00A1 40FF8852
      00A2 00A2 4000885A
      00A3 00A3 57C00002
      00A4 00A4 08000003
   16 00A5 00A5 01E00001 LINE   INCR    FP12              NEXT WORD
W  17 00A6 00A6 30810611        SR      DP,IBDP
W  18 00A7 00A7 30810613        SR      BL,IBEF
```

```
                        ;
                        ;       SET UP PIXEL COMPARISON BUFFER(27,54) IN
                        ;       THE ARRAY FIELDS (27,27) AND (54,27).
                        ;
00A8 00A8 73900000              LI      FP1,0
00A9 00A9 7340001B              LI      FP2,27
00AA 00AA 35A00036              LI      FP3,54
00AB 00AB 3F1A00B3              LOOP,27 SHFT            MOVE FIELD (0,27)
00AC 00AC 430088A5              L       X,FP1
00AD 00AD 40FF88B3              ROT     X,1             DOWN A WORD TO
00AE 00AE 5B400003              S       X,FP2           FIELD (27,27) AND
00AF 00AF 40FE88BB              ROT     X,-2            UP A WORD TO
00B0 00B0 400088BB
00B1 00B1 1B800003              S       X,FP3           FIELD (54,27)
00B2 00B2 01700001              INCR    FP1,FP2         NEXT BIT CO.UMN
00B3 00B3 01A00001 SHFT         INCR    FP3
00B4 00B4 73C00000              LI      FP12,0          ARRAY 0 WORD 0
00B5 00B5 47C088A5              LCM     (0,27),(27,27)  TO
00B6 00B6 40F888B3
00B7 00B7 401C88BB
00B8 00B8 401F88BB
00B9 00B9 65C1A0D3
00BA 00BA 73C00100              LI      FP12,X'0100'    ARRAY 1 WORD 0
00BB 00BB 400088A1              SCM     (0,27),(27,27)
00BC 00BC 4FC0A0D7
00BD 00BD 40008841
00BE 00BE 40FC885A
00BF 00BF 40FF8852
00C0 00C0 40E0885A
00C1 00C1 48000002
00C2 00C2 4BC00001
00C3 00C3 42008840
00C4 00C4 40FC885A
00C5 00C5 40FF8852
00C6 00C6 40E0885A
00C7 00C7 57C00002
00C8 00C8 48000003
00C9 00C9 73C001FF              LI      FP12,X'01FF'    ARRAY 1 WORD 255
00CA 00CA 47C088A5              LCM     (0,27),(54,27)  TO
00CB 00CB 40FE88B3
00CC 00CC 40F888B3
00CD 00CD 65C2A0D3
00CE 00CE 73C000FF              LI      FP12,X'00FF'    ARRAY 0 WORD 255
00CF 00CF 400088A1              SCM     (0,27),(54,27)
00D0 00D0 4FC1A0D7
00D1 00D1 40008841
00D2 00D2 40F8885A
00D3 00D3 40FE8852
00D4 00D4 40E0885A
00D5 00D5 48000002
00D6 00D6 4BC00001
00D7 00D7 42208840
00D8 00D8 40F8885A
00D9 00D9 40FE8852
00DA 00DA 40E0885A
00DB 00DB 57C00002
00DC 00DC 08000003
```

```
 1 00DD 00DD 4000BBA1        SET     M
   00DE 00DE 08000003
 2                     ;
 3                     ;     CLEAR FIELD (81,32) FOR
 4                     ;     USE BY BUFFER FIELDS
 5                     ;
 6 00DF 00DF 74200000        LI      CH,0
 7 00E0 00E0 72000000        LI      CL,0
 8 00E1 00E1 33C01F70        SC      (0,32),(81,32)
   00E2 00E2 3F1F00E4
   00E3 00E3 4B40B7A1
   00E4 00E4 13740003
 9                     ;
10                     ;     CHECK MODE OF OPERATION
11                     ;
12 00E5 00E5 32800001        LI      DP,MODE
13 00E6 00E6 2951012F        BNZ,DP  METH2
14                     ;
15                     ;     ADD 8 NEIGHBORS AND STORE IN SUMBUF(81,12)
16                     ;
17 00E7 00E7 4000BBA1        SET     M
   00E8 00E8 48000003
18 00E9 00E9 75E0085C        ADF     (0,9),(81,12),(81,12)  ADD PX2
   00EA 00EA 73C0085C
   00EB 00EB 37200B0B
   00EC 00EC 2C000000
19 00ED 00ED 75E0085C        ADF     (18,9),(81,12),(81,12) ADD PX8
   00EE 00EE 73C01A5C
   00EF 00EF 37200B0B
   00F0 00F0 2C000000
20 00F1 00F1 75E0085C        ADF     (27,9),(81,12),(81,12) ADD PX1
   00F2 00F2 73C0235C
   00F3 00F3 37200B0B
   00F4 00F4 2C000000
21 00F5 00F5 75E0085C        ADF     (36,9),(81,12),(81,12) ADD PX4
   00F6 00F6 73C02C5C
   00F7 00F7 37200B0B
   00F8 00F8 2C000000
22 00F9 00F9 75E0085C        ADF     (45,9),(81,12),(81,12) ADD PX7
   00FA 00FA 73C0355C
   00FB 00FB 37200B0B
   00FC 00FC 2C000000
23 00FD 00FD 75E0085C        ADF     (54,9),(81,12),(81,12) ADD PX3
   00FE 00FE 73C03E5C
   00FF 00FF 37200B0B
   0100 0100 2C000000
24 0101 0101 75E0085C        ADF     (63,9),(81,12),(81,12) ADD PX6
   0102 0102 73C0475C
   0103 0103 37200B0B
   0104 0104 2C000000
25 0105 0105 75E0085C        ADF     (72,9),(81,12),(81,12) ADD PX9
   0106 0106 73C0505C
   0107 0107 37200B0B
   0108 0108 2C000000
```

```
                        ;
                        ;       CALCULATE THE NUMBER OF PIXELS DIFFERING FROM
                        ;       THE CENTER PIXEL BY MORE THAN THE THRESHOLD.
                        ;       STORE VALUE IN CNTBUF(102,5).  ADD THOSE GRAY
                        ;       LEVELS TOGETHER THAT DIFFER BY MORE THAN THE
                        ;       THRESHOLD AND STORE THEM IN SUMBUF(81,12).
                        ;       SET SIMFLG(115,16) FOR THOSE NEIGHBORS THAT
                        ;       ARE DISSIMILAR.
                        ;
00E9 00E9 74200001      LI      CH,1                    ADD 1
00EA 00EA 72000019      LI      CL,THRES                THRESHOLD
00EB 00EB 4000BBA1      SET     M
00EC 00EC 48000003
00ED 00ED 75E00008      SBF     (8,9),(9,9),(93,9)      PX2-PX5
00EE 00EE 73C01165
00EF 00EF 37200808
00F0 00F0 2C000000
00F1 00F1 2C2102C7      BAL,R2  LIMITS
00F2 00F2 75E0085C      ADF     (8,9),(81,12),(81,12)   ADD TO SUMBUF
00F3 00F3 73C0085C
00F4 00F4 37200B0B
00F5 00F5 2C000000
00F6 00F6 75E0046A      ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
00F7 00F7 73C00F6A
00F8 00F8 37200404
00F9 00F9 2C000000
00FA 00FA 75E00174      ADC     (115,2),(14,2),(115,2)  SET SIMFLG
00FB 00FB 73C00F74
00FC 00FC 37200101
00FD 00FD 2C000000
00FE 00FE 4000BBA1      SET     M
00FF 00FF 48000003
0100 0100 75E0081A      SBF     (18,9),(9,9),(93,9)     PX8-PX5
0101 0101 73C01165
0102 0102 37200808
0103 0103 2C000000
0104 0104 2C2102C7      BAL,R2  LIMITS
0105 0105 75E0085C      ADF     (18,9),(81,12),(81,12)  ADD TO SUMBUF
0106 0106 73C01A5C
0107 0107 37200B0B
0108 0108 2C000000
0109 0109 75E0046A      ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
010A 010A 73C00F6A
010B 010B 37200404
010C 010C 2C000000
010D 010D 75E00176      ADC     (117,2),(14,2),(117,2)  SET SIMFLG
010E 010E 73C00F76
010F 010F 37200101
0110 0110 2C000000
0111 0111 4000BBA1      SET     M
0112 0112 48000003
0113 0113 75E00823      SBF     (27,9),(9,9),(93,9)     PX1-PX5
0114 0114 73C01165
0115 0115 37200808
0116 0116 2C000000
0117 0117 2C2102C7      BAL,R2  LIMITS
```

```
                                ;
                                ;       CALCULATE THE NUMBER OF NEIGHBORING PIXELS
                                ;       DIFFERING FROM THE CENTER PIXEL BY MORE THAN
                                ;       THE THRESHOLD.  STORE VALUE IN CNTBUF(102,5).
                                ;       ADD THOSE NEIGHBORS TOGETHER THAT DIFFER BY
                                ;       MORE THAN THE THRESHOLD AND STORE THEM IN
                                ;       SUMBUF(81,12).
                                ;
 012F 012F 74200001 METH2  LI       CH,1                      ADD 1
 0130 0130 72000019        LI       CL,THRES                  THRESHOLD
 0131 0131 4000BBA1        SET      M
 0132 0132 48000003
 0133 0133 75E00808        SBF      (0,9),(9,9),(93,9)        PX2-PX5
 0134 0134 73C01165
 0135 0135 37200808
 0136 0136 2C000000
 0137 0137 2C21024D        BAL,R2   LIMITS
 0138 0138 75E0085C        ADF      (0,9),(81,12),(81,12)     ADD TO SUMBUF
 0139 0139 73C0085C
 013A 013A 37200B0B
 013B 013B 2C000000
 013C 013C 75E0046A        ADC      (102,5),(11,5),(102,5)    INCR CNTBUF
 013D 013D 73C00F6A
 013E 013E 37200404
 013F 013F 2C000000
 0140 0140 4000BBA1        SET      M
 0141 0141 48000003
 0142 0142 75E0081A        SBF      (18,9),(9,9),(93,9)       PX8-PX5
 0143 0143 73C01165
 0144 0144 37200808
 0145 0145 2C000000
 0146 0146 2C21024D        BAL,R2   LIMITS
 0147 0147 75E0085C        ADF      (18,9),(81,12),(81,12)    ADD TO SUMBUF
 0148 0148 73C01A5C
 0149 0149 37200B0B
 014A 014A 2C000000
 014B 014B 75E0046A        ADC      (102,5),(11,5),(102,5)    INCR CNTBUF
 014C 014C 73C00F6A
 014D 014D 37200404
 014E 014E 2C000000
 014F 014F 4000BBA1        SET      M
 0150 0150 48000003
 0151 0151 75E00823        SBF      (27,9),(9,9),(93,9)       PX1-PX5
 0152 0152 73C01165
 0153 0153 37200808
 0154 0154 2C000000
 0155 0155 2C21024D        BAL,R2   LIMITS
 0156 0156 75E0085C        ADF      (27,9),(81,12),(81,12)    ADD TO SUMBUF
 0157 0157 73C0235C
 0158 0158 37200B0B
 0159 0159 2C000000
 015A 015A 75E0046A        ADC      (102,5),(11,5),(102,5)    INCR CNTBUF
 015B 015B 73C00F6A
 015C 015C 37200404
 015D 015D 2C000000
 015E 015E 4000BBA1        SET      M
 015F 015F 08000003
```

```
  1 0160 0160 75E0082C        SBF     (36,9),(9,9),(93,9)      PX4-PX5
    0161 0161 73C01165
    0162 0162 37200808
    0163 0163 2C000000
  2 0164 0164 2C21024D        BAL,R2  LIMITS
  3 0165 0165 75E0085C        ADF     (36,9),(81,12),(81,12)   ADD TO SUMBUF
    0166 0166 73C02C5C
    0167 0167 37200B0B
    0168 0168 2C000000
  4 0169 0169 75E0046A        ADC     (102,5),(11,5),(102,5)   INCR CNTBUF
    016A 016A 73C00F6A
    016B 016B 37200404
    016C 016C 2C000000
  5 016D 016D 4000BBA1        SET     M
    016E 016E 48000003
  6 016F 016F 75E00835        SBF     (45,9),(9,9),(93,9)      PX7-PX5
    0170 0170 73C01165
    0171 0171 37200808
    0172 0172 2C000000
  7 0173 0173 2C21024D        BAL,R2  LIMITS
  8 0174 0174 75E0085C        ADF     (45,9),(81,12),(81,12)   ADD TO SUMBUF
    0175 0175 73C0355C
    0176 0176 37200B0B
    0177 0177 2C000000
  9 0178 0178 75E0046A        ADC     (102,5),(11,5),(102,5)   INCR CNTBUF
    0179 0179 73C00F6A
    017A 017A 37200404
    017B 017B 2C000000
 10 017C 017C 4000BBA1        SET     M
    017D 017D 48000003
 11 017E 017E 75E0083E        SBF     (54,9),(9,9),(93,9)      PX3-PX5
    017F 017F 73C01165
    0180 0180 37200808
    0181 0181 2C000000
 12 0182 0182 2C21024D        BAL,R2  LIMITS
 13 0183 0183 75E0085C        ADF     (54,9),(81,12),(81,12)   ADD TO SUMBUF
    0184 0184 73C03E5C
    0185 0185 37200B0B
    0186 0186 2C000000
 14 0187 0187 75E0046A        ADC     (102,5),(11,5),(102,5)   INCR CNTBUF
    0188 0188 73C00F6A
    0189 0189 37200404
    018A 018A 2C000000
 15 018B 018B 4000BBA1        SET     M
    018C 018C 48000003
 16 018D 018D 75E00847        SBF     (63,9),(9,9),(93,9)      PX6-PX5
    018E 018E 73C01165
    018F 018F 37200808
    0190 0190 2C000000
 17 0191 0191 2C21024D        BAL,R2  LIMITS
 18 0192 0192 75E0085C        ADF     (63,9),(81,12),(81,12)   ADD TO SUMBUF
    0193 0193 73C0475C
    0194 0194 37200B0B
    0195 0195 2C000000
```

```
ODDPX     APPLE V04-00 24-JUL-80 21:05:11 PAGE 00013 V
    1 0196 0196 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
      0197 0197 73C00F6A
      0198 0198 37200404
      0199 0199 2C000000
    2 019A 019A 4000BBA1        SET     M
      019B 019B 48000003
    3 019C 019C 75E00850        SBF     (72,9),(9,9),(93,9)     PX9-PX5
      019D 019D 73C01165
      019E 019E 37200808
      019F 019F 2C000000
    4 01A0 01A0 2C21024D        BAL,R2  LIMITS
    5 01A1 01A1 75E0085C        ADF     (72,9),(81,12),(81,12)  ADD TO SUMBUF
      01A2 01A2 73C0505C
      01A3 01A3 37200B0B
      01A4 01A4 2C000000
    6 01A5 01A5 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
      01A6 01A6 73C00F6A
      01A7 01A7 37200404
      01A8 01A8 2C000000
```

```
                        ;
                        ;       IF ENOUGH PIXELS DIFFERED BY MORE THEN THE
                        ;       THRESHOLD REPLACE CENTER PIXEL WITH THE
                        ;       AVERAGE OF THOSE PIXELS DIFFERING.
                        ;
 01A9 01A9 4000BBA1             SET     M
 01AA 01AA 48000003
 01AB 01AB 74200003             LI      CH,NON          NO. OF NEIGHBORS
 01AC 01AC 40007741             CLR     Y
 01AD 01AD 77906A0F             GEC     (102,5),(11,5)
 01AE 01AE 03B48B45
 01AF 01AF 3E0201B0
 01B0 01B0 03B42945
 01B1 01B1 43801645
 01B2 01B2 40002241
 01B3 01B3 48000002             L       M,Y
 01B4 01B4 37905051             DVF     (81,12),(102,5),(79,14)    AVERAGE
 01B5 01B5 030088A5
 01B6 01B6 3F0001B8
 01B7 01B7 0B800001
 01B8 01B8 13A40003
 01B9 01B9 75E00350
 01BA 01BA 73C04F66
 01BB 01BB 77200855
 01BC 01BC 34A06A04
 01BD 01BD 2C000000
 01BE 01BE 4A4F0001
 01BF 01BF 424F4445
 01C0 01C0 524F0002
 01C1 01C1 37907257             MVF     (80,8),(107,8)   AVERAGE TO OUTBUF
 01C2 01C2 3F0701C5
 01C3 01C3 433488A5
 01C4 01C4 4B800001
 01C5 01C5 13A40003
 01C6 01C6 40008841             L       Y,M
 01C7 01C7 400044A2             LN      M,Y
 01C8 01C8 48000003
 01C9 01C9 37907211             MVF     (10,8),(107,8) CENTER PX TO OUTBUF
 01CA 01CA 3F0701CD
 01CB 01CB 433488A5
 01CC 01CC 4B800001
 01CD 01CD 13A40003
```

```
                        ;
                        ;       MOVE FIRST PIXEL AND LAST PIXEL
                        ;       UNALTERED TO OUTBUF(107,8)
                        ;
01CE 01CE 73C00000 LN   LI      FP12,0          ARRAY 0 WORD 0
01CF 01CF 47C088A5      LCM     (0,8),(10,8)
01D0 01D0 401E88BB
01D1 01D1 401888BB
01D2 01D2 65C0803B
01D3 01D3 400088A1      SCW     (0,8),(107,8)   FIRST PX OF LINE
01D4 01D4 4FC3AB97
01D5 01D5 42608840
01D6 01D6 40FC885A
01D7 01D7 40FF8852
01D8 01D8 40F0885A
01D9 01D9 57C00002
01DA 01DA 48000003
01DB 01DB 73C001FF      LI      FP12,X'01FF'    ARRAY 1 WORD 255
01DC 01DC 47C088A5      LCM     (0,8),(10,8)
01DD 01DD 401E88BB
01DE 01DE 401888BB
01DF 01DF 65C0803B
01E0 01E0 400088A1      SCW     (0,8),(107,8)   LAST PX OF LINE
01E1 01E1 4FC3AB97
01E2 01E2 42608840
01E3 01E3 40FC885A
01E4 01E4 40FF8852
01E5 01E5 40F0885A
01E6 01E6 57C00002
01E7 01E7 08000003
                        ;
                        ;       MOVE PROCESSED LINE TO OBUFF WITH
                        ;       PIXEL SWAP (1,2,3,4 TO 4,3,2,1)
                        ;
01E8 01E8 33C00000      LI      FP12,0          ARRAY WORD POINTER
01E9 01E9 32810610      LR      DP,OBDP
01EA 01EA 34810612      LR      BL,OBEF
01EB 01EB 3F7F01FF      LOOP,SP LNOUT           MOVE LINE TO OBUFF
01EC 01EC 47C088A5      LCM     (24,8),(107,8)  LOAD C REG WITH PX
01ED 01ED 401F88B3
01EE 01EE 401C88B3
01EF 01EF 25C378FB
01F0 01F0 01E00001      INCR    FP12            NEXT WORD
01F1 01F1 47C088A5      LCM     (16,8),(107,8)  LOAD C REG WITH PX
01F2 01F2 401F88B3
01F3 01F3 25C350BB
01F4 01F4 01E00001      INCR    FP12            NEXT WORD
01F5 01F5 47C088A5      LCM     (8,8),(107,8)   LOAD C REG WITH PX
01F6 01F6 401C88BB
01F7 01F7 401F88BB
01F8 01F8 25C3A87B
01F9 01F9 01E00001      INCR    FP12            NEXT WORD
01FA 01FA 47C088A5      LCM     (0,8),(107,8)   LOAD C REG WITH PX
01FB 01FB 401F88B3
01FC 01FC 401C88B3
01FD 01FD 25C3803B
01FE 01FE 01E00001      INCR    FP12            NEXT WORD
01FF 01FF 3005B000 LNOUT SR     C,OBUFF(DP),3   STORE 4 PX IN OBUFF
```

```
W   1 0200 0200 30810610          SR       DP,OBDP
W   2 0201 0201 30810612          SR       BL,OBEF
    3                       ;
    4                       ;     SHIFT TWO IMAGE LINES OVER A FIELD IN
    5                       ;     THE ARRAYS TO PREPARE FOR A NEW LINE.
    6                       ;
    7 0202 0202 4000BBA1          SET      M
      0203 0203 48000003
    8 0204 0204 37900811          MVF      (9,9),(0,9)      2ND FIELD TO 1ST
      0205 0205 3F080208
      0206 0206 433488A5
      0207 0207 4B800001
      0208 0208 13A40003
    9 0209 0209 3790111A          MVF      (18,9),(9,9)     3RD FIELD TO 2ND
      020A 020A 3F08020D
      020B 020B 433488A5
      020C 020C 4B800001
      020D 020D 13A40003
   10                       ;
   11                       ;     IF OBUFF IS FULL OUTPUT TO COMTAL
   12                       ;
   13 020E 020E 34810612          LR       BL,OBEF          IS OBUFF FULL?
   14 020F 020F 2911021F          BNZ,BL   OBNF             IF NOT CHECK IBUFF
   15                             TRAN     TRAN1            IF SO, OUTPUT
      0210 0210 72800000
      0211 0211 34A00016
      0212 0212 30C18010
   17                             IOWAIT   LINKWDI
      0213 0213 72800000
      0214 0214 74A00000
      0215 0215 37200000
      0216 0216 30C18010
   18 0217 0217 36810001          LR       (BL,DP),TRAN1+1 UPDATE TRAN OUT
   19 0218 0218 3E1F0219          RPT,LOB
   20 0219 0219 28140001          INCR     DP
   21 021A 021A 30810001          SR       (BL,DP),TRAN1+1
   22 021B 021B 32800000          LI       DP,0             RE-INITIALIZE
W  23 021C 021C 30810610          SR       DP,OBDP          OBUFF DATA POINTER
   24 021D 021D 34A01000          LI       BL,MXOBDP        AND
W  25 021E 021E 30810612          SR       BL,OBEF          OBUFF EMPTY FLAG
   26                       ;
   27                       ;     IF IBUFF IS NOT EMPTY GET NEXT LINE
   28                       ;
   29 021F 021F 34810613 OBNF     LR       BL,IBEF          IS IBUFF EMPTY?
   30 0220 0220 2911007E          BNZ,BL   NXLINE           IF NOT, NEXT LINE
```

```
    1                        ;
    2                        ;       IF ENTIRE IMAGE HAS NOT BEEN INPUT
    3                        ;       MOVE MORE DATA INTO IBUFF
    4                        ;
    5 0221 0221 34810614             LR      BL,LIF          HAS ENTIRE IMAGE
    6 0222 0222 01030001             DECR    BL              BEEN INPUT
W   7 0223 0223 30810614             SR      BL,LIF
    8 0224 0224 29010231             BZ,BL   DONE            IF SO, GO TO DONE
    9                                TRAN    TRAN2           IF NOT, INPUT
      0225 0225 72800000
      0226 0226 34A00016
      0227 0227 30C18010
   11                                IOWAIT  LINKWD2         MORE IMAGE
      0228 0228 72800000
      0229 0229 74A00000
      022A 022A 37200000
      022B 022B 30C18010
   12 022C 022C 32800000             LI      DP,0            RE-INITIALIZE
W  13 022D 022D 30810611             SR      DP,IBDP         IBUFF DATA POINTER
   14 022E 022E 34A01000             LI      BL,MXIBDP       AND
W  15 022F 022F 30810613             SR      BL,IBEF         IBUFF EMPTY FLAG
   16 0230 0230 2801007E             B       NXLINE          PROCESS NEXT LINE
   17                        ;
   18                        ;       MOVE LAST LINE TO OBUFF AND
   19                        ;       OUTPUT OBUFF TO COMTAL
   20                        ;
   21 0231 0231 3F7F023E DONE        LOOP,SP LTLINE          MOVE LINE TO OBUFF
   22 0232 0232 32810611             LR      DP,IBDP
   23 0233 0233 3604A000             LR      C,IBUFF(DP),2   LOAD 4 PX IN C REG
W  24 0234 0234 30810611             SR      DP,IBDP
   25 0235 0235 400077A1             CLR     X               PIXEL SWAP
   26 0236 0236 420099A0             SC      X(0)            1,2,3,4 TO 4,3,2,1
   27 0237 0237 400888BB             ROT     X,-8,16
      0238 0238 400088BB
   28 0239 0239 401088BB             ROT     X,-16,32
      023A 023A 400088BB
   29 023B 023B 21C0A0FB             LCW     X(0)
   30 023C 023C 32810610             LR      DP,OBDP
   31 023D 023D 3004B000             SR      C,OBUFF(DP),2   STORE 4 PX IN OBUFF
W  32 023E 023E 30810610 LTLINE      SR      DP,OBDP
   33                                TRAN    TRAN1           OUTPUT FINAL OBUFF
      023F 023F 72800000
      0240 0240 34A00016
      0241 0241 30C18010
   35                                IOWAIT  LINKWD1
      0242 0242 72800000
      0243 0243 74A00000
      0244 0244 37200000
      0245 0245 30C18010
   36                                RLSE    LINKWD1
      0246 0246 72800000
      0247 0247 34A00018
      0248 0248 30C18010
   38                                RLSE    LINKWD2
      0249 0249 72800000
      024A 024A 34A00018
      024B 024B 30C18010
   40 024C 024C 38002000 ERR         WAIT
```

```
ODDPX     APPLE V04-00 24-JUL-80 21:05:11 PAGE 00018 V
                        ;
                        ;       CHECK DIFBUF(93,9) TO DETERMINE IF
                        ;       THE THRESHOLD HAS BEEN EXCEEDED
                        ;
 024D 024D 40007741 LIMITS CLR     Y
 024E 024E 7790651F        LEC     (93,9),(23,9)   DIFBUF<THRES
 024F 024F 03B4B445
 0250 0250 3E060251
 0251 0251 03B41645
 0252 0252 43802945
 0253 0253 40002241
 0254 0254 48000002        L       M,Y
 0255 0255 75E00865        MVNF    (93,9),(93,9)   DIFBUF=-DIFBUF
 0256 0256 33400065
 0257 0257 2C010000
 0258 0258 40007741        CLR     Y
 0259 0259 7790651F        LEC     (93,9),(23,9)   -DIFBUF<THRES
 025A 025A 03B4B445
 025B 025B 3E06025C
 025C 025C 03B41645
 025D 025D 43802945
 025E 025E 40002241
 025F 025F 400044A2        LN      M,Y
 0260 0260 08000003
 0261 0261 280A0000        B       0(R2)
                0610       ORG     X'0610',A       HIGH SPEED DATA BUFFER
                0610 OBDP  DS                      OBUFF POINTER STORAGE
                0611 IBDP  DS                      IBUFF POINTER STORAGE
                0612 OBEF  DS                      OBUFF EMPTY FLAG
                0613 IBEF  DS                      IBUFF EMPTY FLAG
                0614 LIF   DS                      LAST IBUFF FLAG
                0000       END     ODDPX
```

```
ODDPX     APPLE V04-00 24-JUL-80 21:05:11 PAGE 00019 V
ERRORS DETECTED: 00000
WARNINGS DETECTED: 00018
```

APPENDIX D

SIMNB PROGRAM

```
                    ;
                    ;
                    ;
                    SIMNB  START
                           EXTRN   LINKBK1,LINKBK2,TRAN1,TRAN2
                           EXTRN   LINKWD1,LINKWD2
                           ENTRY   ERRTN1,ERRTN2,ERRTN3
                           ENTRY   ERRTN4,ERRTN5
                    ;
                    ;
                    ;
                    ;      DAVID M. CRAWFORD
                    ;      RESEARCH ASSISTANT
                    ;      ELECTRICAL ENGINEERING DEPT.
                    ;      UNIVERSITY OF MISSOURI - COLUMBIA
                    ;      15 JULY 1980
                    ;      REVISION: 22 JULY 1980
                    ;
                    ;
                    ;      THIS PROGRAM IS DESIGNED TO PERFORM NOISE
                    ;      REDUCTION ON IMAGES BY USING A SIMILAR
                    ;      NEIGHBOR REPLACEMENT TECHNIQUE.  THE IMAGE
                    ;      512 X 512 PIXELS, IS READ FROM MAGNETIC
                    ;      TAPE AND PROCESSED BY STARAN.  THE NEW
                    ;      IMAGE IS THEN OUTPUT TO THE [illegible]
                    ;
                    ;      SIMILAR NEIGHBOR REPLACEMENT
                    ;
                    ;      THIS TECHNIQUE USES A [illegible]
                    ;      EACH PIXEL IN THE NOISE [illegible] PIXEL
                    ;      TO THE CENTER PIXEL TO DETERMINE IF IT IS
                    ;      SIMILAR.  IF IT IS THEN A SIMILAR FLAG [illegible]
                    ;      (115,16) IS SET.  THE SIMILAR FLAG [illegible]
                    ;      PARED TO DETERMINE IF ANY ADJACENT NEIGHBORS
                    ;      ARE SIMILAR.  IF SO THE CENTER PIXEL IS NOT
                    ;      CHANGED OTHERWISE THE CENTER PIXEL IS RE-
                    ;      PLACED WITH THE AVERAGE OF THOSE NEIGHBORS
                    ;      WHICH ARE NOT SIMILAR.
                    ;      SIMILARITY IS DETERMINED BY TWO USER SPE-
                    ;      FIED PARAMETERS.  IF HIGH NOISE(NOISE=1) IS TO
                    ;      BE REMOVED ALL NEIGHBORING PIXELS FOR W[illegible]
                    ;        CENTER POINT - NEIGHBORING PIXEL - T[illegible]
                    ;      IS TRUE WILL BE SIMILAR.  IF LOW NOISE(NOISE [illegible]
                    ;      IS TO BE REMOVED ALL NEIGHBORING PIXELS [illegible]
                    ;      WHICH
                    ;        NEIGHBORING PIXEL - CENTER PIXEL > THRESHOLD
                    ;      IS TRUE WILL BE SIMILAR.  NOISE AND THRES ARE
                    ;      THE TWO USER SPECIFIED PARAMETERS
                    ;      THE PROGRAM OPERATES WITH THREE LINES STORED
                    ;      IN ARRAYS 0 AND 1. FIELDS (0,9), (9,9), (15,[illegible]
                    ;      THE FIRST PIXEL OF THE FIRST LINE IS STORED IN
                    ;      FIELD (0,9) OF WORD 0 IN ARRAY 0.  THE LAST
                    ;      PIXEL OF THE FIRST LINE IS IN FIELD (0,9)
                    ;      OF WORD 255 IN ARRAY 1.  THE SECOND LINE IS
                    ;      IN FIELD (9,9) AND THE THIRD IN FIELD (15,9).
                    ;      THE SIMILAR NEIGHBOR VALUE IS DETERMINED FOR
                    ;      THE LINE IN FIELD (9,9).
                    ;
```

```
;
;       THE FOLLOWING BUFFERS ARE SET UP IN THE
;       ARRAYS IN THE SAME WORD WHICH CONTAINS
;       THE PIXEL BEING CHECKED TO DETERMINE IF
;       IT IS NOISE.
;
;
;       COMPARISON BUFFER       CMPBUF  (27,54)
;       SUM BUFFER              SUMBUF  (81,12)
;       DIFFERENCE BUFFER       DIFBUF  (93,9)
;       COUNT BUFFER            CNTBUF  (102,5)
;       OUTPUT BUFFER           OUTBUF  (107,8)
;       SIMILAR FLAG            SIMFLG  (115,16)
;       CHANGE FLAG             CHGFLG  (131,5)
;
;
;       CMPBUF - USED TO STORE NEIGHBORS IN THE
;                SAME ARRAY WORD AS THE CENTER
;                PIXEL FOR COMPARISON.
;       SUMBUF - USED TO STORE THE SUM OF THOSE
;                NEIGHBORS WHICH ARE NOT SIMILAR.
;       DIFBUF - USED TO STORE THE DIFFERENCE
;                BETWEEN EACH NEIGHBOR AND THE
;                CENTER PIXEL.
;       CNTBUF - USED TO STORE THE NUMBER OF
;                DISSIMILAR NEIGHBORS.
;       OUTBUF - USED TO STORE THE PROCESSED
;                PIXEL TO BE OUTPUT.
;       SIMFLG - A FLAG FOR EACH NEIGHBOR WHICH
;                IS EQUAL TO 0 IF SIMILAR AND
;                1 IF DISSIMILAR.
;       CHGFLG - A FLAG WHOSE VALUE IS GREATER
;                THAN ZERO IF THE CENTER PIXEL
;                IS TO BE REPLACED BY THE AVER-
;                AGE OF THE DISSIMILAR NEIGHBORS.
;
;       ALL EDGE POINTS ARE OUTPUT UNALTERED.
;
```

```
                 02C6 ERRTN1 EQU     ERR              ERROR RETURN
                 02C6 ERRTN2 EQU     ERR              ERROR RETURN
                 02C6 ERRTN3 EQU     ERR              ERROR RETURN
                 02C6 ERRTN4 EQU     ERR              ERROR RETURN
                 02C6 ERRTN5 EQU     ERR              ERROR RETURN
                 0000 BLKNUM EQU     0                I/O BLOCK NUMBER
                 A000 IBUFF  EQU     X'A000'          IBUFF ADDRESS
                 B000 OBUFF  EQU     X'B000'          OBUFF ADDRESS
                 1000 IBSIZE EQU     4096             INPUT BUFFER SIZE
                 1000 OBSIZE EQU     4096             OUTPUT BUFFER SIZE
                 0020 LIB    EQU     IBSIZE/128       IMAGE LINES IN IBUFF
                 0020 LOB    EQU     OBSIZE/128       IMAGE LINES IN OBUFF
                 0010 BLKS   EQU     512/LIB          NO. OF INPUT BLOCKS
                 0080 SP     EQU     128              32 BIT SEGMENTS PER LIN
                 0000 NOISE  EQU     0                LOW=0  HIGH=1
                 0019 THRES  EQU     25               THRESHOLD
                 1000 MXOBDP EQU     OBSIZE           MAX OBUFF DP VALUE
                 1000 MXIBDP EQU     IBSIZE           MAX IBUFF DP VALUE
                 0000 SIMNB  EQU     $
 0000 0000 3660C000          LI,2    AS,X'C000'       SELECT ARRAYS 0 AND 1
                     ;
                     ;       INITIALIZE INPUT -- MAG TAPE
                     ;
 0001 0001 74201000          LI      CH,IBSIZE
 0002 0002 32000000          LI      CL,BLKNUM
 0003 0003 30010001          SR      C,TRAN2+1
 0004 0004 74200000          LI      CH,0
 0005 0005 3200A000          LI      CL,IBUFF
 0006 0006 30010003          SR      C,TRAN2+3
                             INIT    LINKBK2
 0007 0007 72800000
 0008 0008 34A00014
 0009 0009 30C18010
                     ;
                     ;       INITIALIZE OUTPUT -- COMTAL
                     ;
 000A 000A 74201000          LI      CH,OBSIZE
 000B 000B 32000000          LI      CL,BLKNUM
 000C 000C 30010001          SR      C,TRAN1+1
 000D 000D 74200000          LI      CH,0
 000E 000E 3200B000          LI      CL,OBUFF
 000F 000F 30010003          SR      C,TRAN1+3
                             INIT    LINKBK1
 0010 0010 72800000
 0011 0011 34A00014
 0012 0012 30C18010
                     ;
                     ;       INPUT LINES TO IBUFF (NO. = LIB)
                     ;
                             TRAN    TRAN2
 0013 0013 72800000
 0014 0014 34A00016
 0015 0015 30C18010
                             IOWAIT  LINKWD2
 0016 0016 72800000
 0017 0017 74A00000
 0018 0018 37200000
 0019 0019 30C18010
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00004 V
    1                        ;
    2                        ;       INITIALIZE BUFFER POINTERS
    3                        ;
    4 001A 001A 34A01000         LI      BL,MXIBDP       INIT IBUFF EMPTY FLAG
W   5 001B 001B 30810613         SR      BL,IBEF
    6 001C 001C 34A01000         LI      BL,MXOBDP       INIT OBUFF EMPTY FLAG
W   7 001D 001D 30810612         SR      BL,OBEF
    8 001E 001E 34A00010         LI      BL,BLKS         INIT LAST IBUFF FLAG
W   9 001F 001F 30810614         SR      BL,LIF
   10                        ;
   11                        ;       MOVE FIRST LINE IN IBUFF TO OBUFF
   12                        ;
   13 0020 0020 73C00000         LI      FP12,0
   14 0021 0021 32800000         LI      DP,0
   15 0022 0022 34810612         LR      BL,OBEF
   16 0023 0023 3F7F002C         LOOP,SP LINE1            LINE ONE TO IBUFF
   17 0024 0024 3602A000         LR      C,IBUFF(DP)      LOAD 4 PX IN C REG
   18 0025 0025 400077A1         CLR     X                PIXEL SWAP
   19 0026 0026 420099A0         SC      X(0)             1,2,3,4 TO 4,3,2,1
   20 0027 0027 400888BB         ROT     X,-8,16
      0028 0028 400088BB
   21 0029 0029 401088BB         ROT     X,-16,32
      002A 002A 400088BB
   22 002B 002B 21C0A0FB         LCW     X(0)
   23 002C 002C 3005B000 LINE1   SR      C,OBUFF(DP),3    STORE 4 PX IN OBUFF
W  24 002D 002D 30810610         SR      DP,OBDP
W  25 002E 002E 30810612         SR      BL,OBEF
   26                        ;
   27                        ;       CLEAR FIELD (0,32) FOR USE
   28                        ;       BY THE INPUT DATA
   29                        ;
   30 002F 002F 74200000         LI      CH,0
   31 0030 0030 72000000         LI      CL,0
   32 0031 0031 33C01F1F         SC      (0,32),(0,32)
      0032 0032 3F1F0034
      0033 0033 4B40B7A1
      0034 0034 13740003
```

```
                        ;
                        ;       MOVE FIRST TWO LINES IN
                        ;       IBUFF TO THE ARRAYS
                        ;
0035 0035 32800000              LI      DP,0
0036 0036 34810613              LR      BL,IBEF
0037 0037 33C00000              LI      FP12,0          ARRAY WORD POINTER
0038 0038 3F7F0059              LOOP,SP L1              FIRST LINE IN ARRAY
0039 0039 3605A000              LR      C,IBUFF(DP),3   LOAD 4 PX IN C REG
003A 003A 400088A1              SCW     (0,8),(1,8)     PX TO ARRAY
003B 003B 4FC0A147
003C 003C 42008840
003D 003D 40FF8852
003E 003E 57C00002
003F 003F 08000003
0040 0040 01E00001              INCR    FP12            NEXT WORD
0041 0041 400088A1              SCW     (8,8),(1,8)     PX TO ARRAY
0042 0042 4FC0A147
0043 0043 42008840
0044 0044 40F8885A
0045 0045 40FF885A
0046 0046 57C00002
0047 0047 08000003
0048 0048 01E00001              INCR    FP12            NEXT WORD
0049 0049 400088A1              SCW     (16,8),(1,8)    PX TO ARRAY
004A 004A 4FC0A147
004B 004B 42008840
004C 004C 40F0885A
004D 004D 40FF885A
004E 004E 57C00002
004F 004F 08000003
0050 0050 01E00001              INCR    FP12            NEXT WORD
0051 0051 400088A1              SCW     (24,8),(1,8)    PX TO ARRAY
0052 0052 4FC0A147
0053 0053 42008840
0054 0054 40E0885A
0055 0055 40FF885A
0056 0056 40F88852
0057 0057 57C00002
0058 0058 08000003
0059 0059 01E00001 L1           INCR    FP12            NEXT WORD
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00006 V
    1 005A 005A 33C00000          LI       FP12,0            ARRAY WORD POINTER
    2 005B 005B 3F7F007B          LOOP,SP  L2                2ND LINE IN ARRAY
    3 005C 005C 3605A000          LR       C,IBUFF(DP),3     LOAD 4 PX IN C REG
    4 005D 005D 400088A1          SCW      (0,8),(10,8)      PX TO ARRAY
      005E 005E 4FC0AA8F
      005F 005F 42008840
      0060 0060 40FE8852
      0061 0061 40F88852
      0062 0062 57C00002
      0063 0063 08000003
    5 0064 0064 01E00001          INCR     FP12              NEXT WORD
    6 0065 0065 400088A1          SCW      (8,8),(10,8)      PX TO ARRAY
      0066 0066 4FC0AA8F
      0067 0067 42008840
      0068 0068 40FE8852
      0069 0069 57C00002
      006A 006A 08000003
    7 006B 006B 01E00001          INCR     FP12              NEXT WORD
    8 006C 006C 400088A1          SCW      (16,8),(10,8)     PX TO ARRAY
      006D 006D 4FC0AA8F
      006E 006E 42008840
      006F 006F 40F8885A
      0070 0070 40FE885A
      0071 0071 57C00002
      0072 0072 08000003
    9 0073 0073 01E00001          INCR     FP12              NEXT WORD
   10 0074 0074 400088A1          SCW      (24,8),(10,8)     PX TO ARRAY
      0075 0075 4FC0AA8F
      0076 0076 42008840
      0077 0077 40F0885A
      0078 0078 40FE885A
      0079 0079 57C00002
      007A 007A 08000003
   11 007B 007B 01E00001 L2       INCR     FP12              NEXT WORD
W  12 007C 007C 30810611          SR       DP,IBDP
W  13 007D 007D 30810613          SR       BL,IBEF
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00007 V
    1                        ;
    2                        ;       MOVE ANOTHER LINE FROM IBUFF TO THE ARRAY
    3                        ;
    4 007E 007E 32810611 NXLINE LR       DP,IBDP
    5 007F 007F 34810613        LR       BL,IBEF
    6 0080 0080 33C00000        LI       FP12,0            ARRAY WORD POINTER
    7 0081 0081 3F7F00A5        LOOP,SP LINE               MOVE LINE IN ARRAY
    8 0082 0082 3605A000        LR       C,IBUFF(DP),3     LOAD 4 PX IN C REG
    9 0083 0083 400088A1        SCW      (0,8),(19,8)      PX TO ARRAY
      0084 0084 4FC0B3D7
      0085 0085 42008840
      0086 0086 40FC8852
      0087 0087 40FF885A
      0088 0088 40F0885A
      0089 0089 57C00002
      008A 008A 08000003
   10 008B 008B 01E00001        INCR     FP12              NEXT WORD
   11 008C 008C 400088A1        SCW      (8,8),(19,8)      PX TO ARRAY
      008D 008D 4FC0B3D7
      008E 008E 42008840
      008F 008F 40FC885A
      0090 0090 40FF8852
      0091 0091 40F0885A
      0092 0092 57C00002
      0093 0093 08000003
   12 0094 0094 01E00001        INCR     FP12              NEXT WORD
   13 0095 0095 400088A1        SCW      (16,8),(19,8)     PX TO ARRAY
      0096 0096 4FC0B3D7
      0097 0097 42008840
      0098 0098 40FF885A
      0099 0099 40FC885A
      009A 009A 57C00002
      009B 009B 08000003
   14 009C 009C 01E00001        INCR     FP12              NEXT WORD
   15 009D 009D 400088A1        SCW      (24,8),(19,8)     PX TO ARRAY
      009E 009E 4FC0B3D7
      009F 009F 42008840
      00A0 00A0 40FC885A
      00A1 00A1 40FF8852
      00A2 00A2 4000885A
      00A3 00A3 57C00002
      00A4 00A4 08000003
   16 00A5 00A5 01E00001 LINE   INCR     FP12              NEXT WORD
W  17 00A6 00A6 30810611        SR       DP,IBDP
W  18 00A7 00A7 30810613        SR       BL,IBEF
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00008 V
                        ;
                        ;       SET UP PIXEL COMPARISON BUFFER(27,54) IN
                        ;       THE ARRAY FIELDS (27,27) AND (54,27).
                        ;
 00A8 00A8 73900000             LI      FP1,0
 00A9 00A9 7340001B             LI      FP2,27
 00AA 00AA 35A00036             LI      FP3,54
 00AB 00AB 3F1A00B3             LOOP,27 SHFT            MOVE FIELD (0,27)
 00AC 00AC 430088A5             L       X,FP1
 00AD 00AD 40FF88B3             ROT     X,1             DOWN A WORD TO
 00AE 00AE 5B400003             S       X,FP2           FIELD (27,27) AND
 00AF 00AF 40FE88BB             ROT     X,-2            UP A WORD TO
 00B0 00B0 400088BB
 00B1 00B1 18800003             S       X,FP3           FIELD (54,27)
 00B2 00B2 01700001             INCR    FP1,FP2         NEXT BIT CO.UMN
 00B3 00B3 01A00001 SHFT        INCR    FP3
 00B4 00B4 73C00000             LI      FP12,0          ARRAY 0 WORD 0
 00B5 00B5 47C088A5             LCM     (0,27),(27,27)  TO
 00B6 00B6 40F888B3
 00B7 00B7 401C88BB
 00B8 00B8 401F88BB
 00B9 00B9 65C1A0D3
 00BA 00BA 73C00100             LI      FP12,X'0100'    ARRAY 1 WORD 0
 00BB 00BB 400088A1             SCW     (0,27),(27,27)
 00BC 00BC 4FC0A0D7
 00BD 00BD 40008841
 00BE 00BE 40FC885A
 00BF 00BF 40FF8852
 00C0 00C0 40E0885A
 00C1 00C1 48000002
 00C2 00C2 4BC00001
 00C3 00C3 42008840
 00C4 00C4 40FC885A
 00C5 00C5 40FF8852
 00C6 00C6 40E0885A
 00C7 00C7 57C00002
 00C8 00C8 48000003
 00C9 00C9 73C001FF             LI      FP12,X'01FF'    ARRAY 1 WORD 255
 00CA 00CA 47C088A5             LCM     (0,27),(54,27)  TO
 00CB 00CB 40FE88B3
 00CC 00CC 40F888B3
 00CD 00CD 65C2A0D3
 00CE 00CE 73C000FF             LI      FP12,X'00FF'    ARRAY 0 WORD 255
 00CF 00CF 400088A1             SCW     (0,27),(54,27)
 00D0 00D0 4FC1A0D7
 00D1 00D1 40008841
 00D2 00D2 40F8885A
 00D3 00D3 40FE8852
 00D4 00D4 40E0885A
 00D5 00D5 48000002
 00D6 00D6 4BC00001
 00D7 00D7 42208840
 00D8 00D8 40F8885A
 00D9 00D9 40FE8852
 00DA 00DA 40E0885A
 00DB 00DB 57C00002
 00DC 00DC 08000003
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00009 V
    1 00DD 00DD 4000BBA1          SET     M
      00DE 00DE 08000003
    2                       ;
    3                       ;     CLEAR FIELD (81,32) AND (113,32)
    4                       ;     FOR USE BY BUFFER FIELDS
    5                       ;
    6 00DF 00DF 74200000          LI      CH,0
    7 00E0 00E0 72000000          LI      CL,0
    8 00E1 00E1 33C01F70          SC      (0,32),(81,32)
      00E2 00E2 3F1F00E4
      00E3 00E3 4B40B7A1
      00E4 00E4 13740003
    9 00E5 00E5 33C01F90          SC      (0,32),(113,32)
      00E6 00E6 3F1F00E8
      00E7 00E7 4B40B7A1
      00E8 00E8 13740003
```

```
                        ;
                        ;       CALCULATE THE NUMBER OF PIXELS DIFFERING FROM
                        ;       THE CENTER PIXEL BY MORE THAN THE THRESHOLD.
                        ;       STORE VALUE IN CNTBUF(102,5).  ADD THOSE GRAY
                        ;       LEVELS TOGETHER THAT DIFFER BY MORE THAN THE
                        ;       THRESHOLD AND STORE THEM IN SUMBUF(81,12).
                        ;       SET SIMFLG(115,16) FOR THOSE NEIGHBORS THAT
                        ;       ARE DISSIMILAR.
                        ;
 00E9 00E9 74200001             LI      CH,1                    ADD 1
 00EA 00EA 72000019             LI      CL,THRES                THRESHOLD
 00EB 00EB 4000BBA1             SET     M
 00EC 00EC 48000003
 00ED 00ED 75E00808             SBF     (0,9),(9,9),(93,9)      PX2-PX5
 00EE 00EE 73C01165
 00EF 00EF 37200808
 00F0 00F0 2C000000
 00F1 00F1 2C2102C7             BAL,R2  LIMITS
 00F2 00F2 75E0085C             ADF     (0,9),(81,12),(81,12)   ADD TO SUMBUF
 00F3 00F3 73C0085C
 00F4 00F4 37200B0B
 00F5 00F5 2C000000
 00F6 00F6 75E0046A             ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
 00F7 00F7 73C00F6A
 00F8 00F8 37200404
 00F9 00F9 2C000000
 00FA 00FA 75E00174             ADC     (115,2),(14,2),(115,2)  SET SIMFLG
 00FB 00FB 73C00F74
 00FC 00FC 37200101
 00FD 00FD 2C000000
 00FE 00FE 4000BBA1             SET     M
 00FF 00FF 48000003
 0100 0100 75E0081A             SBF     (18,9),(9,9),(93,9)     PX8-PX5
 0101 0101 73C01165
 0102 0102 37200808
 0103 0103 2C000000
 0104 0104 2C2102C7             BAL,R2  LIMITS
 0105 0105 75E0085C             ADF     (18,9),(81,12),(81,12)  ADD TO SUMBUF
 0106 0106 73C01A5C
 0107 0107 37200B0B
 0108 0108 2C000000
 0109 0109 75E0046A             ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
 010A 010A 73C00F6A
 010B 010B 37200404
 010C 010C 2C000000
 010D 010D 75E00176             ADC     (117,2),(14,2),(117,2)  SET SIMFLG
 010E 010E 73C00F76
 010F 010F 37200101
 0110 0110 2C000000
 0111 0111 4000BBA1             SET     M
 0112 0112 48000003
 0113 0113 75E00823             SBF     (27,9),(9,9),(93,9)     PX1-PX5
 0114 0114 73C01165
 0115 0115 37200808
 0116 0116 2C000000
 0117 0117 2C2102C7             BAL,R2  LIMITS
```

```
  1 0118 0118 75E0085C        ADF     (27,9),(81,12),(81,12)  ADD TO SUMBUF
    0119 0119 73C0235C
    011A 011A 37200B0B
    011B 011B 2C000000
  2 011C 011C 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
    011D 011D 73C00F6A
    011E 011E 37200404
    011F 011F 2C000000
  3 0120 0120 75E00178        ADC     (119,2),(14,2),(119,2)  SET SIMFLG
    0121 0121 73C00F78
    0122 0122 37200101
    0123 0123 2C000000
  4 0124 0124 4000BBA1        SET     M
    0125 0125 48000003
  5 0126 0126 75E0082C        SBF     (36,9),(9,9),(93,9)     PX4-PX5
    0127 0127 73C01165
    0128 0128 37200808
    0129 0129 2C000000
  6 012A 012A 2C2102C7        BAL,R2  LIMITS
  7 012B 012B 75E0085C        ADF     (36,9),(81,12),(81,12)  ADD TO SUMBUF
    012C 012C 73C02C5C
    012D 012D 37200B0B
    012E 012E 2C000000
  8 012F 012F 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
    0130 0130 73C00F6A
    0131 0131 37200404
    0132 0132 2C000000
  9 0133 0133 75E0017A        ADC     (121,2),(14,2),(121,2)  SET SIMFLG
    0134 0134 73C00F7A
    0135 0135 37200101
    0136 0136 2C000000
 10 0137 0137 4000BBA1        SET     M
    0138 0138 48000003
 11 0139 0139 75E00835        SBF     (45,9),(9,9),(93,9)     PX7-PX5
    013A 013A 73C01165
    013B 013B 37200808
    013C 013C 2C000000
 12 013D 013D 2C2102C7        BAL,R2  LIMITS
 13 013E 013E 75E0085C        ADF     (45,9),(81,12),(81,12)  ADD TO SUMBUF
    013F 013F 73C0355C
    0140 0140 37200B0B
    0141 0141 2C000000
 14 0142 0142 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
    0143 0143 73C00F6A
    0144 0144 37200404
    0145 0145 2C000000
 15 0146 0146 75E0017C        ADC     (123,2),(14,2),(123,2)  SET SIMFLG
    0147 0147 73C00F7C
    0148 0148 37200101
    0149 0149 2C000000
 16 014A 014A 4000BBA1        SET     M
    014B 014B 48000003
 17 014C 014C 75E0083E        SBF     (54,9),(9,9),(93,9)     PX3-PX5
    014D 014D 73C01165
    014E 014E 37200808
    014F 014F 2C000000
 18 0150 0150 2C2102C7        BAL,R2  LIMITS
```

```
  1 0151 0151 75E0085C        ADF     (54,9),(81,12),(81,12)  ADD TO SUMBUF
    0152 0152 73C03E5C
    0153 0153 37200B0B
    0154 0154 2C000000
  2 0155 0155 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
    0156 0156 73C00F6A
    0157 0157 37200404
    0158 0158 2C000000
  3 0159 0159 75E0017E        ADC     (125,2),(14,2),(125,2)  SET SIMFLG
    015A 015A 73C00F7E
    015B 015B 37200101
    015C 015C 2C000000
  4 015D 015D 4000BBA1        SET     M
    015E 015E 48000003
  5 015F 015F 75E00847        SBF     (63,9),(9,9),(93,9)     PX6-PX5
    0160 0160 73C01165
    0161 0161 37200808
    0162 0162 2C000000
  6 0163 0163 2C2102C7        BAL,R2  LIMITS
  7 0164 0164 75E0085C        ADF     (63,9),(81,12),(81,12)  ADD TO SUMBUF
    0165 0165 73C0475C
    0166 0166 37200B0B
    0167 0167 2C000000
  8 0168 0168 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
    0169 0169 73C00F6A
    016A 016A 37200404
    016B 016B 2C000000
  9 016C 016C 75E00180        ADC     (127,2),(14,2),(127,2)  SET SIMFLG
    016D 016D 73C00F80
    016E 016E 37200101
    016F 016F 2C000000
 10 0170 0170 4000BBA1        SET     M
    0171 0171 48000003
 11 0172 0172 75E00850        SBF     (72,9),(9,9),(93,9)     PX9-PX5
    0173 0173 73C01165
    0174 0174 37200808
    0175 0175 2C000000
 12 0176 0176 2C2102C7        BAL,R2  LIMITS
 13 0177 0177 75E0085C        ADF     (72,9),(81,12),(81,12)  ADD TO SUMBUF
    0178 0178 73C0505C
    0179 0179 37200B0B
    017A 017A 2C000000
 14 017B 017B 75E0046A        ADC     (102,5),(11,5),(102,5)  INCR CNTBUF
    017C 017C 73C00F6A
    017D 017D 37200404
    017E 017E 2C000000
 15 017F 017F 75E00182        ADC     (129,2),(14,2),(129,2)  SET SIMFLG
    0180 0180 73C00F82
    0181 0181 37200101
    0182 0182 2C000000
```

```
                        ;
                        ;       MAKE CHANGE FLAG GREATER THAN ZERO
                        ;       IF ADJACENT SIMILAR NEIGHBORS EXIST
                        ;
 0183 0183 74200001             LI      CH,1                    ADD 1
 0184 0184 72000000             LI      CL,0
 0185 0185 4000BBA1             SET     M
 0186 0186 48000003
 0187 0187 40007741             CLR     Y
 0188 0188 77907478             EQF     (115,2),(119,2)  PX 2 AND 1 SIMFLG EQ
 0189 0189 00008841
 018A 018A 3F01018D
 018B 018B 43A488A5
 018C 018C 433400A5
 018D 018D 00002243
 018E 018E 48000002             L       M,Y
 018F 018F 40007741             CLR     Y
 0190 0190 40008841             EQC     (115,2),(30,2)   PX 2 SIMILAR?
 0191 0191 3790741F
 0192 0192 3E010193
 0193 0193 03B42645
 0194 0194 48000002             L       M,Y
 0195 0195 75E00487             ADC     (131,5),(11,5),(131,5) INCR CHGFLG
 0196 0196 73C00F87
 0197 0197 37200404
 0198 0198 2C000000
 0199 0199 4000BBA1             SET     M
 019A 019A 48000003
 019B 019B 40007741             CLR     Y
 019C 019C 7790747E             EQF     (115,2),(125,2)  PX 2 AND 3 SIMFLG EQ
 019D 019D 00008841
 019E 019E 3F0101A1
 019F 019F 43A488A5
 01A0 01A0 433400A5
 01A1 01A1 00002243
 01A2 01A2 48000002             L       M,Y
 01A3 01A3 40007741             CLR     Y
 01A4 01A4 40008841             EQC     (115,2),(30,2)   PX 2 SIMILAR?
 01A5 01A5 3790741F
 01A6 01A6 3E0101A7
 01A7 01A7 03B42645
 01A8 01A8 48000002             L       M,Y
 01A9 01A9 75E00487             ADC     (131,5),(11,5),(131,5) INCR CHGFLG
 01AA 01AA 73C00F87
 01AB 01AB 37200404
 01AC 01AC 2C000000
 01AD 01AD 4000BBA1             SET     M
 01AE 01AE 48000003
 01AF 01AF 40007741             CLR     Y
 01B0 01B0 77907682             EQF     (117,2),(129,2)  PX 8 AND 9 SIMFLG EQ
 01B1 01B1 00008841
 01B2 01B2 3F0101B5
 01B3 01B3 43A488A5
 01B4 01B4 433400A5
 01B5 01B5 00002243
 01B6 01B6 48000002             L       M,Y
 01B7 01B7 00007741             CLR     Y
```

```
  1 01B8 01B8 40008841        EQC     (117,2),(30,2)   PX 8 SIMILAR?
    01B9 01B9 3790761F
    01BA 01BA 3E0101BB
    01BB 01BB 03B42645
  2 01BC 01BC 48000002        L       M,Y
  3 01BD 01BD 75E00487        ADC     (131,5),(11,5),(131,5) INCR CHGFLG
    01BE 01BE 73C00F87
    01BF 01BF 37200404
    01C0 01C0 2C000000
  4 01C1 01C1 4000BBA1        SET     M
    01C2 01C2 48000003
  5 01C3 01C3 40007741        CLR     Y
  6 01C4 01C4 7790767C        EQF     (117,2),(123,2)  PX 8 AND 7 SIMFLG EQ
    01C5 01C5 00008841
    01C6 01C6 3F0101C9
    01C7 01C7 43A488A5
    01C8 01C8 433400A5
    01C9 01C9 00002243
  7 01CA 01CA 48000002        L       M,Y
  8 01CB 01CB 40007741        CLR     Y
  9 01CC 01CC 40008841        EQC     (117,2),(30,2)   PX 8 SIMILAR?
    01CD 01CD 3790761F
    01CE 01CE 3E0101CF
    01CF 01CF 03B42645
 10 01D0 01D0 48000002        L       M,Y
 11 01D1 01D1 75E00487        ADC     (131,5),(11,5),(131,5) INCR CHGFLG
    01D2 01D2 73C00F87
    01D3 01D3 37200404
    01D4 01D4 2C000000
 12 01D5 01D5 4000BBA1        SET     M
    01D6 01D6 48000003
 13 01D7 01D7 40007741        CLR     Y
 14 01D8 01D8 7790787A        EQF     (119,2),(121,2)  PX 1 AND 4 SIMFLG EQ
    01D9 01D9 00008841
    01DA 01DA 3F0101DD
    01DB 01DB 43A488A5
    01DC 01DC 433400A5
    01DD 01DD 00002243
 15 01DE 01DE 48000002        L       M,Y
 16 01DF 01DF 40007741        CLR     Y
 17 01E0 01E0 40008841        EQC     (119,2),(30,2)   PX 1 SIMILAR?
    01E1 01E1 3790781F
    01E2 01E2 3E0101E3
    01E3 01E3 03B42645
 18 01E4 01E4 48000002        L       M,Y
 19 01E5 01E5 75E00487        ADC     (131,5),(11,5),(131,5) INCR CHGFLG
    01E6 01E6 73C00F87
    01E7 01E7 37200404
    01E8 01E8 2C000000
 20 01E9 01E9 4000BBA1        SET     M
    01EA 01EA 48000003
 21 01EB 01EB 40007741        CLR     Y
 22 01EC 01EC 77907A7C        EQF     (121,2),(123,2)  PX 4 AND 7 SIMFLG EQ
    01ED 01ED 00008841
    01EE 01EE 3F0101F1
    01EF 01EF 43A488A5
    01F0 01F0 433400A5
    01F1 01F1 00002243
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00015 V
    1 01F2 01F2 48000002        L       M,Y
    2 01F3 01F3 40007741        CLR     Y
    3 01F4 01F4 40008841        EQC     (121,2),(30,2)   PX 4 SIMILAR?
      01F5 01F5 37907A1F
      01F6 01F6 3E0101F7
      01F7 01F7 03B42645
    4 01F8 01F8 48000002        L       M,Y
    5 01F9 01F9 75E00487        ADC     (131,5),(11,5),(131,5) INCR CHGFLG
      01FA 01FA 73C00F87
      01FB 01FB 37200404
      01FC 01FC 2C000000
    6 01FD 01FD 4000BBA1        SET     M
      01FE 01FE 48000003
    7 01FF 01FF 40007741        CLR     Y
    8 0200 0200 77907E80        EQF     (125,2),(127,2)  PX 3 AND 6 SIMFLG EQ
      0201 0201 00008841
      0202 0202 3F010205
      0203 0203 43A488A5
      0204 0204 433400A5
      0205 0205 00002243
    9 0206 0206 48000002        L       M,Y
   10 0207 0207 40007741        CLR     Y
   11 0208 0208 40008841        EQC     (125,2),(30,2)   PX 3 SIMILAR?
      0209 0209 37907E1F
      020A 020A 3E01020B
      020B 020B 03B42645
   12 020C 020C 48000002        L       M,Y
   13 020D 020D 75E00487        ADC     (131,5),(11,5),(131,5) INCR CHGFLG
      020E 020E 73C00F87
      020F 020F 37200404
      0210 0210 2C000000
   14 0211 0211 4000BBA1        SET     M
      0212 0212 48000003
   15 0213 0213 40007741        CLR     Y
   16 0214 0214 77908082        EQF     (127,2),(129,2)  PX 6 AND 9 SIMFLG EQ
      0215 0215 00008841
      0216 0216 3F010219
      0217 0217 43A488A5
      0218 0218 433400A5
      0219 0219 00002243
   17 021A 021A 48000002        L       M,Y
   18 021B 021B 40007741        CLR     Y
   19 021C 021C 40008841        EQC     (127,2),(30,2)   PX 6 SIMILAR?
      021D 021D 3790801F
      021E 021E 3E01021F
      021F 021F 03B42645
   20 0220 0220 48000002        L       M,Y
   21 0221 0221 75E00487        ADC     (131,5),(11,5),(131,5) INCR CHGFLG
      0222 0222 73C00F87
      0223 0223 37200404
      0224 0224 2C000000
```

```
 1                          ;
 2                          ;       IF ADJACENT SIMILAR NEIGHBORS DO NOT
 3                          ;       EXIST REPLACE CENTER PIXEL WITH THE
 4                          ;       AVERAGE OF THOSE PIXELS DIFFERING.
 5                          ;
 6 0225 0225 74200000               LI      CH,0
 7 0226 0226 4000BBA1               SET     M
   0227 0227 48000003
 8 0228 0228 40007741               CLR     Y
 9 0229 0229 40008841               EQC     (131,5),(11,5) ANY SIMILAR NEIGHBORS?
   022A 022A 3790870F
   022B 022B 3E04022C
   022C 022C 03B42645
10 022D 022D 48000002               L       M,Y
11 022E 022E 37905051               DVF     (81,12),(102,5),(79,14) AVERAGE
   022F 022F 030088A5
   0230 0230 3F000232
   0231 0231 0B800001
   0232 0232 13A40003
   0233 0233 75E00350
   0234 0234 73C04F66
   0235 0235 77200855
   0236 0236 34A06A04
   0237 0237 2C000000
   0238 0238 4A4F0001
   0239 0239 424F4445
   023A 023A 524F0002
12 023B 023B 37907257               MVF     (80,8),(107,8)  AVERAGE TO OUTBUF
   023C 023C 3F07023F
   023D 023D 433488A5
   023E 023E 4B800001
   023F 023F 13A40003
13 0240 0240 40008841               L       Y,M
14 0241 0241 400044A2               LN      M,Y
   0242 0242 48000003
15 0243 0243 37907211               MVF     (10,8),(107,8)  CENTER PX TO OUTBUF
   0244 0244 3F070247
   0245 0245 433488A5
   0246 0246 4B800001
   0247 0247 13A40003
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00017 V
   1                        ;
   2                        ;       MOVE FIRST PIXEL AND LAST PIXEL
   3                        ;       UNALTERED TO OUTBUF(107,8)
   4                        ;
   5 0248 0248 73C00000         LI      FP12,0          ARRAY 0 WORD 0
   6 0249 0249 47C088A5         LCM     (0,8),(10,8)
     024A 024A 401E88BB
     024B 024B 401888BB
     024C 024C 65C0803B
   7 024D 024D 400088A1         SCW     (0,8),(107,8)   FIRST PX OF LINE
     024E 024E 4FC3AB97
     024F 024F 42608840
     0250 0250 40FC885A
     0251 0251 40FF8852
     0252 0252 40F0885A
     0253 0253 57C00002
     0254 0254 48000003
   8 0255 0255 73C001FF         LI      FP12,X'01FF'    ARRAY 1 WORD 255
   9 0256 0256 47C088A5         LCM     (0,8),(10,8)
     0257 0257 401E88BB
     0258 0258 401888BB
     0259 0259 65C0803B
  10 025A 025A 400088A1         SCW     (0,8),(107,8)   LAST PX OF LINE
     025B 025B 4FC3AB97
     025C 025C 42608840
     025D 025D 40FC885A
     025E 025E 40FF8852
     025F 025F 40F0885A
     0260 0260 57C00002
     0261 0261 08000003
```

```
   1                        ;
   2                        ;       MOVE PROCESSED LINE TO OBUFF WITH
   3                        ;       PIXEL SWAP (1,2,3,4 TO 4,3,2,1)
   4                        ;
   5 0262 0262 33C00000             LI      FP12,0          ARRAY WORD POINTER
   6 0263 0263 32810610             LR      DP,OBDP
   7 0264 0264 34810612             LR      BL,OBEF
   8 0265 0265 3F7F0279             LOOP,SP LNOUT           MOVE LINE TO OBUFF
   9 0266 0266 47C088A5             LCM     (24,8),(107,8)  LOAD C REG WITH PX
     0267 0267 401F88B3
     0268 0268 401C88B3
     0269 0269 25C378FB
  10 026A 026A 01E00001             INCR    FP12            NEXT WORD
  11 026B 026B 47C088A5             LCM     (16,8),(107,8)  LOAD C REG WITH PX
     026C 026C 401F88B3
     026D 026D 25C350BB
  12 026E 026E 01E00001             INCR    FP12            NEXT WORD
  13 026F 026F 47C088A5             LCM     (8,8),(107,8)   LOAD C REG WITH PX
     0270 0270 401C88BB
     0271 0271 401F88BB
     0272 0272 25C3A87B
  14 0273 0273 01E00001             INCR    FP12            NEXT WORD
  15 0274 0274 47C088A5             LCM     (0,8),(107,8)   LOAD C REG WITH PX
     0275 0275 401F88B3
     0276 0276 401C88B3
     0277 0277 25C3803B
  16 0278 0278 01E00001             INCR    FP12            NEXT WORD
  17 0279 0279 30058000 LNOUT       SR      C,OBUFF(DP),3   STORE 4 PX IN OBUFF
W 18 027A 027A 30810610             SR      DP,OBDP
W 19 027B 027B 30810612             SR      BL,OBEF
  20                        ;
  21                        ;       SHIFT TWO IMAGE LINES OVER A FIELD IN
  22                        ;       THE ARRAYS TO PREPARE FOR A NEW LINE.
  23                        ;
  24 027C 027C 4000BBA1             SET     M
     027D 027D 48000003
  25 027E 027E 37900811             MVF     (9,9),(0,9)     2ND FIELD TO 1ST
     027F 027F 3F080282
     0280 0280 433488A5
     0281 0281 4B800001
     0282 0282 13A40003
  26 0283 0283 3790111A             MVF     (18,9),(9,9)    3RD FIELD TO 2ND
     0284 0284 3F080287
     0285 0285 433488A5
     0286 0286 4B800001
     0287 0287 13A40003
```

```
   1                          ;
   2                          ;       IF OBUFF IS FULL OUTPUT TO COMTAL
   3                          ;
   4 0288 0288 34810612               LR      BL,OBEF          IS OBUFF FULL?
   5 0289 0289 29110299               BNZ,BL  OBNF             IF NOT CHECK IBUFF
   6                                  TRAN    TRAN1            IF SO, OUTPUT
     028A 028A 72800000
     028B 028B 34A00016
     028C 028C 30C18010
   8                                  IOWAIT  LINKWD1
     028D 028D 72800000
     028E 028E 74A00000
     028F 028F 37200000
     0290 0290 30C18010
   9 0291 0291 36810001               LR      (BL,DP),TRAN1+1 UPDATE TRAN OUT
  10 0292 0292 3E1F0293               RPT,LOB
  11 0293 0293 28140001               INCR    DP
  12 0294 0294 30810001               SR      (BL,DP),TRAN1+1
  13 0295 0295 32800000               LI      DP,0             RE-INITIALIZE
W 14 0296 0296 30810610               SR      DP,OBDP          OBUFF DATA POINTER
  15 0297 0297 34A01000               LI      BL,MXOBDP        AND
W 16 0298 0298 30810612               SR      BL,OBEF          OBUFF EMPTY FLAG
  17                          ;
  18                          ;       IF IBUFF IS NOT EMPTY GET NEXT LINE
  19                          ;
  20 0299 0299 34810613 OBNF          LR      BL,IBEF          IS IBUFF EMPTY?
  21 029A 029A 2911007E               BNZ,BL  NXLINE           IF NOT, NEXT LINE
  22                          ;
  23                          ;       IF ENTIRE IMAGE HAS NOT BEEN INPUT
  24                          ;       MOVE MORE DATA INTO IBUFF
  25                          ;
  26 029B 029B 34810614               LR      BL,LIF           HAS ENTIRE IMAGE
  27 029C 029C 01030001               DECR    BL               BEEN INPUT
W 28 029D 029D 30810614               SR      BL,LIF
  29 029E 029E 290102AB               BZ,BL   DONE             IF SO, GO TO DONE
  30                                  TRAN    TRAN2            IF NOT, INPUT
     029F 029F 72800000
     02A0 02A0 34A00016
     02A1 02A1 30C18010
  32                                  IOWAIT  LINKWD2          MORE IMAGE
     02A2 02A2 72800000
     02A3 02A3 74A00000
     02A4 02A4 37200000
     02A5 02A5 30C18010
  33 02A6 02A6 32800000               LI      DP,0             RE-INITIALIZE
W 34 02A7 02A7 30810611               SR      DP,IBDP          IBUFF DATA POINTER
  35 02A8 02A8 34A01000               LI      BL,MXIBDP        AND
W 36 02A9 02A9 30810613               SR      BL,IBEF          IBUFF EMPTY FLAG
  37 02AA 02AA 2801007E               B       NXLINE           PROCESS NEXT LINE
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00020 V
    1                        ;
    2                        ;       MOVE LAST LINE TO OBUFF AND
    3                        ;       OUTPUT OBUFF TO COMTAL
    4                        ;
    5 02AB 02AB 3F7F02B8 DONE    LOOP,SP LTLINE          MOVE LINE TO OBUFF
    6 02AC 02AC 32810611         LR      DP,IBDP
    7 02AD 02AD 3604A000         LR      C,IBUFF(DP),2   LOAD 4 PX IN C REG
W   8 02AE 02AE 30810611         SR      DP,IBDP
    9 02AF 02AF 400077A1         CLR     X               PIXEL SWAP
   10 02B0 02B0 420099A0         SC      X(0)            1,2,3,4 TO 4,3,2,1
   11 02B1 02B1 400888BB         ROT     X,-8,16
      02B2 02B2 400088BB
   12 02B3 02B3 401088BB         ROT     X,-16,32
      02B4 02B4 400088BB
   13 02B5 02B5 21C0A0FB         LCW     X(0)
   14 02B6 02B6 32810610         LR      DP,OBDP
   15 02B7 02B7 3004B000         SR      C,OBUFF(DP),2   STORE 4 PX IN OBUFF
W  16 02B8 02B8 30810610 LTLINE SR      DP,OBDP
   17                            TRAN    TRAN1           OUTPUT FINAL OBUFF
      02B9 02B9 72800000
      02BA 02BA 34A00016
      02BB 02BB 30C18010
   19                            IOWAIT  LINKWD1
      02BC 02BC 72800000
      02BD 02BD 74A00000
      02BE 02BE 37200000
      02BF 02BF 30C18010
   20                            RLSE    LINKWD1
      02C0 02C0 72800000
      02C1 02C1 34A00018
      02C2 02C2 30C18010
   22                            RLSE    LINKWD2
      02C3 02C3 72800000
      02C4 02C4 34A00018
      02C5 02C5 30C18010
   24 02C6 02C6 38002000 ERR    WAIT
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00021 V
   1                        ;
   2                        ;       CHECK DIFBUF(93,9) TO DETERMINE IF THE
   3                        ;       THRESHOLD HAS BEEN EXCEEDED CHECK
   4                        ;       FOR LOW OR HIGH NOISE AS SPECIFIED.
   5                        ;
   6 02C7 02C7 40007741 LIMITS CLR      Y
   7 02C8 02C8 32800000         LI      DP,NOISE
   8 02C9 02C9 295102D1         BNZ,DP  HIGH
   9 02CA 02CA 7790651F         GTC     (93,9),(23,9)   DIFBUF>THRES
     02CB 02CB 03B47845
     02CC 02CC 3E0602CD
     02CD 02CD 03B42945
     02CE 02CE 43801645
     02CF 02CF 00002241
  10 02D0 02D0 280102DA         B       A1
  11 02D1 02D1 75E00865 HIGH   MVNF    (93,9),(93,9)   DIFBUF=-DIFBUF
     02D2 02D2 33400065
     02D3 02D3 2C010000
  12 02D4 02D4 7790651F         GTC     (93,9),(23,9)   DIFBUF>THRES
     02D5 02D5 03B47845
     02D6 02D6 3E0602D7
     02D7 02D7 03B42945
     02D8 02D8 43801645
     02D9 02D9 40002241
  13 02DA 02DA 08000002 A1     L       M,Y
  14 02DB 02DB 280A0000         B       0(R2)
  15                0610        ORG     X'0610',A       HIGH SPEED DATA BUFFER
  16                0610 OBDP   DS                      OBUFF POINTER STORAGE
  17                0611 IBDP   DS                      IBUFF POINTER STORAGE
  18                0612 OBEF   DS                      OBUFF EMPTY FLAG
  19                0613 IBEF   DS                      IBUFF EMPTY FLAG
  20                0614 LIF    DS                      LAST IBUFF FLAG
  21                0000        END     SIMNB
```

```
SIMNB     APPLE V04-00 24-JUL-80 21:47:54 PAGE 00022 V
ERRORS DETECTED: 00000
WARNINGS DETECTED: 00018
```

APPENDIX E

PTEDGE PROGRAM

```
                                PTEDGE    START
                                          ENTRY     ERROR
                                          ENTRY     ERRTN1,ERRTN2,ERRTN3,ERRTN4,ERR1
                                          EXTRN     INITS,TRANINS,TRANOUTS,RELEASES
                                          EXTRN     BLKNUM,GETAD,PUTAD,LINEIN,LINEOUT
                                          EXTRN     CONDITION,FLBLKCNT,INBUFF,OUTBUFF
                                          EXTRN     TRANTTIN,TRANOUT,WDCNT
                                          EXTRN     EDGE,FORMAT,MOVE
                                ;
                                ;         SET UP TRAN PARAMETERS
                                ;
0000 0000 74200000              PTEDGE    LI        CH,0
0001 0001 32000008                        LI        CL,8
0002 0002 30010000                        SR        C,FLBLKCNT
0003 0003 74200000                        LI        CH,WDCNT
0004 0004 32000000                        LI        CL,BLKNUM
0005 0005 30010001                        SR        C,TRANTTIN+1
0006 0006 34202000                        LI        CH,8192
0007 0007 30010001                        SR        C,TRANOUT+1
                                ;
                                ;         OPEN TRAN FILES
                                ;
0008 0008 2C710000                        BAL,R7    INITS
                                ;
                                ;         TRAN IN 2 BLOCKS FROM TAPE TO MC
                                ;
0009 0009 74200000              MNEXTBUF  LI        CH,0
000A 000A 32000000                        LI        CL,INBUFF
000B 000B 30010003                        SR        C,TRANTTIN+3
000C 000C 2C710000                        BAL,R7    TRANINS
000D 000D 74200000                        LI        CH,0
000E 000E 32001000                        LI        CL,INBUFF+4096
000F 000F 30010003                        SR        C,TRANTTIN+3
0010 0010 2C710000                        BAL,R7    TRANINS
```

```
                        ;
                        ;          INITIALIZE OFFSETS FOR INPUFF AND OUTPUFF
                        ;
0011 0011 74200000              LI        CH,0
0012 0012 32000000              LI        CL,0
0013 0013 30010000              SR        C,GETAD
0014 0014 30010000              SR        C,PUTAD
                        ;
                        ;          UPDATE TOTAL NUMBER OF TIMES TO READ
                        ;          DATA FROM THE TAPE
0015 0015 36810000              LR        (BL,DP),FLMLKCNT
0016 0016 30800018              LRR       C,(BL,DP)
0017 0017 28160001              DECR      DP
0018 0018 30810000              SR        (BL,DP),FLMLKCNT
0019 0019 3390001C              LI        FP1,28
001A 001A 2801001C              BPS       MBEGIN
001B 001B 2801002C              B         MABEADI
                        ;
                        ;          THIS SECTION ONLY FOR THE VERY BEGINING
                        ;          OF THE PROCESS.
                        ;          MOVE THE FIRST 2 LINES INTO ARRAY FIELD
                        ;          L2 AND L3
                        ;
001C 001C 74200000 MBEGIN       LI        CH,0
```

```
 1 0010 0010 32000003            LI        CL,3
 2 001E 001E 30010000            SR        C,CONDITION
 3 001F 001F 2C710000            BAL,R7    LINEIN
 4 0020 0020 74200000            LI        CH,0
 5 0021 0021 32000001            LI        CL,1
 6 0022 0022 30010000            SR        C,CONDITION
 7 0023 0023 2C710000            BAL,R7    LINEIN
 8                        ;
 9                        ;      BLACK OUT TWO LINES IN OUTBUFF
10                        ;
11 0024 0024 74200000            LI        CH,0
12 0025 0025 72000000            LI        CL,0
13 0026 0026 32800000            LI        DP,0
14 0027 0027 3EFF0028            RPT,256
15 0028 0028 30040000            SR        C,OUTBUFF(DP),2
16                        ;
17                        ;      INITIALIZE OFFSET OF OUTBUFF
18                        ;
19 0029 0029 74200000            LI        CH,0
20 002A 002A 32000100            LI        CL,256
21 002B 002B 30010000            SR        C,PUTAD
22                        ;
23                        ;      CHECK IF INBUFF IS EMPTY
24                        ;
25 002C 002C 36010000 MAHEAD1    LR        C,GETAD
26 002D 002D 33900012            LI        RP1,18
27 002E 002E 28C10041            BRZ       MAHEAD2
28                        ;
29                        ;      REORDER DATA IN BC FOR DUMPING TO COMTAL
30                        ;      FROM 0123 TO 3210
31                        ;
```

```
  32 002F 002F 32802000                LI        DP,8192
  33 0030 0030 3602FFFF  MSWAP         LR        C,OUTBUFF(DP)-1
  34 0031 0031 30100001F               LRK,1     F,C
T 35 0032 0032 30300006                LRR,3     FP1,C
T 36 0033 0033 30300015                LRR,3     FL1,C
  37 0034 0034 31C6FFFF                SR        F,OUTBUFF(DP)-1,4
  38 0035 0035 29510030                BNZ,DP    MSWAP
  39                         ;
  40                         ;         TRAN OUT 64 IMAGE LINES
  41                         ;
  42 0036 0036 74200000                LI        CH,0
  43 0037 0037 32000000                LI        CL,OUTBUFF
  44 0038 0038 30010003                SR        C,TRANOUT+3
  45 0039 0039 2C710000                BAL,R7    TRANOUTS
  46 003A 003A 36810001                LR        (BL,DP),TRANOUT+1
  47                         ;
  48                         ;         UPDATE THE IMAGE LINE NUMBER
  49                         ;
  50 003B 003B 3E3F003C                RPT,64
  51 003C 003C 28140001                INCR      DP
  52 003D 003D 30810001                SR        (BL,DP),TRANOUT+1
  53 003E 003E 36810000                LR        (BL,DP),FLBLKCNT
  54 003F 003F 2941004A                BZ,DP     MEND
  55 0040 0040 28010009                B         MNEXTBUF
  56                         ;
  57                         ;         SHIFT L2 TO L1, L3 TO L2 AND
  58                         ;         MOVE A NEW LINE FROM BC TO L3 OF AM.
```

```
                          ;
0041 0041 2C710000 MAHEAD2   BAL,R7    MOVE
0042 0042 74200000           LI        CH,0
0043 0043 32000001           LI        CL,1
0044 0044 30010000           SR        C,CONDITION
0045 0045 2C710000           BAL,R7    LINEIN
                          ;
                          ;          FORMAT THE FIELD FROM A 3 X 3 MATRIX
                          ;
0045 0046 2C710000           BAL,R7    FORMAT
                          ;
                          ;          DO EDGE DETECTION
                          ;
0047 0047 2C710000           BAL,R7    EDGE
                          ;
                          ;          MOVE A RESULTANT LINE FROM AE TO BC
                          ;
0048 0048 2C710000           BAL,R7    LINEOUT
0049 0049 28010020           B         MAHEAD1
                          ;
                          ;          ONCE FINISH, RELEASE FILES
                          ;
                     004A MEND       EQU       $
004A 004A 2C710000           BAL,R7    RELEASES
                          ;
                          ;          ERROR TRAP
                          ;
004B 004B 38002000 ERROR     SAI
                     004B ERRIN1     EQU       ERROR
                     004B ERRIN2     EQU       ERROR
                     004B ERRTP3     EQU       ERROR
                     004B ERRTP4     EQU       ERROR
                     004B ERRTP5     EQU       ERROR
                     0000            END       PTEDGE
```

ERRORS DETECTED: 00000
WARNINGS DETECTED: 00002

```
                        EDGE      START
                        ;
                        ;   THE RADIENTS BETWEEN THE CENTER AND
                        ;   THE NEIGHBORING POINTS, WHICH ARE PICKED,
                        ;   OF 1 ARE PARTIAL  BE CALCULATED
                        ;
                                  ENTRY     EDGE
                                  EXTRN     THRSHLD,PIEXPT
                                  EXTRN     TOTAL,SLOT
                0808 L1           DF        8,8
                1008 L2           DF        16,8
                1808 L3           DF        24,8
                1808 P1           DF        24,8
                2008 P2           DF        32,8
                2808 P3           DF        40,8
                3008 P4           DF        48,8
                3808 P5           DF        56,8
                4008 P6           DF        64,8
                4808 P7           DF        72,8
                5008 P8           DF        80,8
                5809 TEMP         DF        88,4
                6108 RESULT       DF        97,8
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                0000 END;         END       S
```

```
30 0000 0000 40008BA1              SPT       F
   0001 0001 48000003
31 0002 0002 74200000              LI        CH,0
32 0003 0003 32000000              LI        CL,0
33 0004 0004 30010000              SR        C,TOTAL
34 0005 0005 30010000              SR        C,ELEMENT
35 0006 0006 30010000 REP  A1      LR        C,TOTAL
36 0007 0007 33900001C             LI        FP1,28
37 0008 0008 26D100A5              BCS       FAKEUP
38 0009 0009 36810000              LR        (PL,DP),ELEMENT
39 000A 000A 36070000              LR        C,SLOT(DP)
40 000B 000B 33900001F             LI        FP1,31
41 000C 000C 26C1009E              BNZ       UPDATE
42                         ;
43                         ;  IF CURRENT ELEMENT IS PICKED
44                         ;  CALCULATE GRADIENT
45                         ;
46 0000 0000 280200BC              B         CASE(DP)
47                         ;
48                         ;          [illegible]
49                         ;
50 000E 000E 75E00717 P  CES1      SLF       [illegible]
   000F 000F 73C01F0C
   0010 0010 372007DE
   0011 0011 2C000000
51 0012 0012 73C05058              AVAF      TEMP,TEMP
   0013 0013 35F00860
   0014 0014 2C010000
52 0015 0015 30010000              LR        C,THRSHLD
```

```
 1 0016 0016 7790601F              GTC       TEMP,(23,9)
   0017 0017 03B47845
   0018 0018 3E060019
   0019 0019 03B42945
   001A 001A 43B01645
   001B 001B 40002241
 2 001C 001C 400088A1              EXOR      M,Y
   001D 001D 4000CCA2
   001E 001E 08000003
 3 001F 001F 2B01009E              B         UPDATE
 4                          ;
 5                          ;  SECOND ELEMENT
 6                          ;
 7 0020 0020 75E00717 PROCESS2     SPF       L2,P2,TEMP
   0021 0021 73C02760
   0022 0022 37200708
   0023 0023 2C000000
 8 0024 0024 73C06058              MVAF      TEMP,TEMP
   0025 0025 35E00860
   0026 0026 2C010000
 9 0027 0027 36010000              GR        C,THRSHLD
10 0028 0028 7790601F              GTC       TEMP,(23,9)
   0029 0029 03B47845
   002A 002A 3E06002B
   002B 002B 03B42945
   002C 002C 43B01645
   002D 002D 40002241
11 002E 002E 400088A1              EXOR      M,Y
   002F 002F 4000CCA2
   0030 0030 08000003
12 0031 0031 2B01009E              B         UPDATE
13                          ;
14                          ;  THIRD ELEMENT
15                          ;
16 0032 0032 75E00717 PROCESS3     SPF       L2,P3,TEMP
   0033 0033 73C02F60
   0034 0034 37200708
   0035 0035 2C000000
17 0036 0036 73C06058              MVAF      TEMP,TEMP
   0037 0037 35E00860
   0038 0038 2C010000
18 0039 0039 36010000              GR        C,THRSHLD
19 003A 003A 7790601F              GTC       TEMP,(23,9)
   003B 003B 03B47845
   003C 003C 3E06003D
   003D 003D 03B42945
   003E 003E 43B01645
   003F 003F 40002241
```

```
20 0040 0040 40008881                    EXOR      M,Y
   0041 0041 4000CCA2
   0042 0042 08000003
21 0043 0043 2801009E                    B         UPDATE
22                          ;
23                          ;
24                          ;   FOURTH ELEMENT
25                          ;
26 0044 0041 75E00717 PROCESS4   SHF       L2,P4,TEMP
   0045 0045 73C03760
```



```
   0046 0046 37260708
   0047 0047 2C000000
 1 0048 0048 73C06058                    MVAF      TEMP,TEMP
   0049 0049 35E00860
   004A 004A 2C010000
 2 004B 004B 36010000                    LR        C,THRSHLD
 3 004C 004C 7790601F                    GTC       TEMP,(23,9)
   004D 004D 03847845
   004E 004E 3E06004F
   004F 004F 03842945
   0050 0050 43801645
   0051 0051 40002241
 4 0052 0052 40008881                    EXOR      M,Y
   0053 0053 4000CCA2
   0054 0054 08000003
 5 0055 0055 2801009E                    B         UPDATE
 6                          ;
 7                          ;   FIFTH ELEMENT
 8                          ;
 9 0056 0056 75E00717 PROCESS5    SHF       L2,P5,TEMP
   0057 0057 73C03F60
   0058 0058 37200708
   0059 0059 2C000000
10 005A 005A 73C06058                    MVAF      TEMP,TEMP
   005B 005B 35E00860
   005C 005C 2C010000
11 005D 005D 36010000                    LR        C,THRSHLD
12 005E 005E 7790601F                    GTC       TEMP,(23,9)
   005F 005F 03847845
   0060 0060 3E060061
   0061 0061 03842945
   0062 0062 43801645
   0063 0063 40002241
```

Cop
Per

```
13 0064 0064 40008841              LXOR      F,Y
   0065 0065 4000CCA2
   0066 0066 08000003
14 0067 0067 2801009F              B         UPDATE
15                          ;
16                          ;  SIXTH ELEMENT
17                          ;
18 0068 0068 75E00717 PROCESS6     SHF       D2,P6,TEMP
   0069 0069 73C04760
   006A 006A 37200708
   006B 006B 2C000000
19 006C 006C 73C06058              MVAF      TEMP,TEMP
   006D 006D 35E00860
   006E 006E 2C010000
20 006F 006F 36010000              LR        C,THRSHLD
21 0070 0070 7790601F              GTC       TEMP,(23,9)
   0071 0071 03B47845
   0072 0072 3E060073
   0073 0073 03B42945
   0074 0074 43801645
   0075 0075 40002241
22 0076 0076 400088A1              LXOR      M,Y
   0077 0077 4000CCA2
   0078 0078 08000003
23 0079 0079 2801009F              B         UPDATE
```

EDGE APPLE V04-00 16-JUL-[illegible] [illegible]:39:14 PAGE 00004

```
 1                          ;
 2                          ;  SEVENTH ELEMENT
 3                          ;
 4 007A 007A 75E00717 PROCESS7      SHF       D2,P7,TEMP
   007B 007B 73C04E60
   007C 007C 37200708
   007D 007D 2C000000
 5 007E 007E 73C06058               MVAF      TEMP,TEMP
   007F 007F 35F00860
   0080 0080 2C010000
 6 0081 0081 36010000               LR        C,THRSHLD
 7 0082 0082 7790601F               GTC       TEMP,(23,9)
   0083 0083 03B47845
   0084 0084 3E060085
   0085 0085 03B42945
   0086 0086 43801645
   0087 0087 40002241
 8 0088 0088 400088A1               LXOR      M,Y
   0089 0089 4000CCA2
   008A 008A 08000003
 9 008B 008B 2801009F               B         UPDATE
```

```
10                                  ;
11                                  ;   [illegible] ELEMENT
12                                  ;
13 008C 008C 75E00717 P[illegible]      SEF       L2,PR,TEMP
   008D 008D 73C05760
   008E 008E 37200708
   008F 008F 2C000000
14 0090 0090 73C06058               MVAF      TEMP,TEMP
   0091 0091 35E00860
   0092 0092 2C010000
15 0093 0093 36010000               LF        C,[illegible]
16 0094 0094 7790601F               GIC       TEMP,(23,9)
   0095 0095 03E47845
   0096 0096 3E060097
   0097 0097 03E42945
   0098 0098 43801645
   0099 0099 40002241
17 009A 009A 4000CCA1               EXOR      M,Y
   009B 009B 4000CCA2
   009C 009C 08000003
18 009D 009D 2801009F               B         UPDATE
19                                  ;
20                                  ;   UPDATE SECTION
21                                  ;
22                  009E [illegible]     FOD       S
23 009E 009E 36810000               LR        (PE,DP),TOTAL
24 009F 009F 28140001               INCR      DP
25 00A0 00A0 30810000               SR        (PE,DP),TOTAL
26 00A1 00A1 36810000               LR        (PE,DP),[illegible]
27 00A2 00A2 28140001               INCR      DP
28 00A3 00A3 30810000               SR        (PE,DP),[illegible]
29 00A4 00A4 28010006               P         [illegible]
30                                  ;
31                                  ;   KEEP THE RESULT FOR OUTPUT
32                                  ;
33 00A5 00A5 4000CCA1 [illegible]   SET       [illegible]
   00A6 00A6 08000003
```

```
 1 00A7 00A7 36010000            LR        C,THRSHLD
 2 00A8 00A8 7790601F            LTC       TEMP,(23,9)
   00A9 00A9 03B44745
   00AA 00AA 3E0600AB
   00AB 00AB 03B41645
   00AC 00AC 43802945
   00AD 00AD 40002241
 3 00AE 00AE 48000002            L         M,Y
 4 00AF 00AF 76000000            LI        C,0
 5 00B0 00B0 33C01F68            SC        (24,8),RESULT
   00B1 00B1 3F0700B3
   00B2 00B2 4B40B7A1
   00B3 00B3 13740003
 6 00B4 00B4 400044A2            LN        M,Y
   00B5 00B5 48000003
 7 00B6 00B6 760000FF            LI        C,255
 8 00B7 00B7 33C01F68            SC        (24,8),RESULT
   00B8 00B8 3F0700BA
   00B9 00B9 4B40B7A1
   00BA 00BA 13740003
 9 00BB 00BB 280F0000            B         0(,7)
10                        ;
11                        ;   THE ADDRESSES TO BE JUMPED TO
12                        ;
13 00BC 00BC 2801000E CASE        B         PROCES1
14 00BD 00BD 28010020             B         PROCES2
15 00BE 00BE 28010032             B         PROCES3
16 00BF 00BF 28010044             B         PROCES4
17 00C0 00C0 28010056             B         PROCES5
18 00C1 00C1 28010068             B         PROCES6
19 00C2 00C2 2801007A             B         PROCES7
20 00C3 00C3 2801008C             B         PROCES8
21                   FFFF         END
```



```
ERRORS DETECTED: 00000
WARNINGS DETECTED: 00000
```

```
MOVE        APPLE V04-00 16-JUL-80 04:40:21 PAGE 00001

                            MOVE        START
                            ;
                            ;           THIS SUBROUTINE MOVES FIELD L2 TO L1,
                            ;                   L3 TO L2
                            ;
                                                        MOVE
                      0808 L1                           8,8
                      1008 L2                           16,8
                      1808 L3                           24,8
                            ;
                            ;
                      0000 MOVE                         S
0000 0000 40008BA1                                      M
0001 0001 46000003
0002 0002 37900F17                                      L2,L1
0003 0003 3E070006
0004 0004 433488A5
0005 0005 4E800001
0006 0006 13A40003
0007 0007 3790171F                                      L3,L2
0008 0008 3E070008
0009 0009 433488A5
000A 000A 4E800001
000B 000B 13A40003
000C 000C 28020000                                      0(R7)
                      FFFF                              END


MOVE        APPLE V04-00 16-JUL-80 04:40:21 PAGE 00002

ERRORS DETECTED: 00000
WARNINGS DETECTED: 00000
```

```
LINEIN    APPLE V04-00 16-JUL-80 0[illegible]:11:14 PAGE 00001

                                LINEIN    [illegible]
                                ;
                                ;   MOVE A [illegible]   FROM BC TO AM
                                ;
                                          [illegible] LINEIN
                                          [illegible]      CONDI' 100',GETAD,ERROR,[illegible]
                                ;
                                ;
                                ;
                        0008 L1         DE        8,8
                        1008 L2         DE        16,8
                        1808 L3         DE        24,8
                        0008 C1         DE        0,8
                        0808 C2         DE        8,8
                        1008 C3         DE        16,8
                        1808 C4         DE        24,8
                                ;
                                ;
                                ;
                        0000 LINEIN     EQU       $
0000 0000 75700080                      LI        R[illegible],128
0001 0001 4000B8A1                      SRI       R
0002 0002 08000003
0003 0003 32810000                      LR        DP,GETAD
0004 0004 33C00000                      LI        RR12,0
0005 0005 34A10000                      LR,       RL,C'[illegible]
0006 0006 01030001                      DEC       BL
0007 0007 29010008                      BZ,       CASE1
0008 0008 01030001                      DEC       BL
0009 0009 2901002C                      BZ,       CASE2
000A 000A 01030001                      DEC       BL
000B 000B 2901004C                      BZ,       CASE3
000C 000C 2B010000                      B         ERROR
```

```
 33                                  ;
 34                                  ;   MOVE A L1    TO L3
 35                                  ;
 36 000D 000D 36040000 CASE1        LR          1,INBUFF(DP),2
 37 000F 000E 40008BA1              SCW         C1,L3
    000F 000F 4FC0B8FF
    0010 0010 42008840
    0011 0011 40F8885A
    0012 0012 40F0885A
    0013 0013 57C00002
    0014 0014 08000003
 38 0015 0015 01E00001              INCR        FP12
 39 0016 0016 40008BA1              SCW         C2,L3
    0017 0017 4FC0B8FF
    0018 0018 42008840
    0019 0019 40F08852
    001A 001A 57C00002
    001B 001B 08000003
 40 001C 001C 01E00001              INCR        FP12
 41 001D 001D 40008BA1              SCW         C3,L3
    001E 001E 4FC0B8FF
    001F 001F 42008840
    0020 0020 40F88852
    0021 0021 57C00002
    0022 0022 08000003
```



```
  1 0023 0023 01E00001              INCR        FP12
  2 0024 0024 40008BA1              SCW         C4,L3
    0025 0025 4FC0B8FF
    0026 0026 57C00000
    0027 0027 08000003
  3 0028 0028 01E00001              INCR        FP12
  4 0029 0029 01010001              DECR        FL1
  5 002A 002A 2891000D              BNZ,=L1     CASE1
  6 002B 002B 28010008              B           RETURN
  7                                 ;
  8                                 ;   MOVE A LINE TO L1
  9                                 ;
 10 002C 002C 36040000 CASE2        LR          C,INBUFF(DP),2
 11 002D 002D 40008BA1              SCW         C1,L1
    002E 002E 4FC0A87F
    002F 002F 42008840
    0030 0030 40F88852
    0031 0031 57C00002
    0032 0032 08000003
 12 0033 0033 01E00001              INCR        FP12
 13 0034 0034 40008BA1              SCW         C2,L1
    0035 0035 4FC0A87F
    0036 0036 57C00000
    0037 0037 08000003
 14 0038 0038 01E00001              INCR        FP12
```

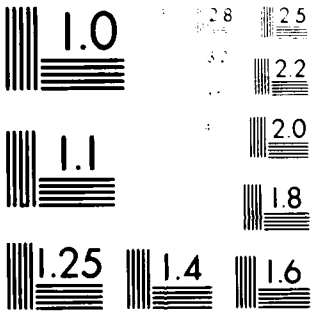
```
15 0039 0039 40008BA1                SCW        C3,L1
   003A 003A 4FC0A67F
   003B 003B 42008840
   003C 003C 40F8885A
   003D 003D 4000885A
   003E 003E 57C00002
   003F 003F 08000003
16 0040 0040 01E00001                INCR       FP12
17 0041 0041 40008BA1                SCW        C4,L1
   0042 0042 4FC0A87F
   0043 0043 42008840
   0044 0044 40F0885A
   0045 0045 4000885A
   0046 0045 57C00002
   0047 0047 08000003
18 0048 0048 01E00001                INCR       FP12
19 0049 0049 01010001                DECR       FL1
20 004A 004A 2891002C                BNZ,FL1    CASE2
21 004B 004B 2801006B                B          RETURN
22                          ;
23                          ;  MOVE A LINE L1 TO L2
24                          ;
25 004C 004C 36040000 CASE3          LR         R,INBUFF(DP),2
26 004D 004D 40008BA1                SCW        C1,L2
   004E 004E 4FC0B8BF
   004F 004F 42008840
   0050 0050 40F08852
   0051 0051 57C00002
   0052 0052 08000003
27 0053 0053 01E00001                INCR       FP12
28 0054 0054 40008BA1                SCW        C2,L2
   0055 0055 4FC0B8BF
   0056 0056 42008840
```

```
LINEIN     APPLE V04-00 16-JUL-80 04:41:14 PAGE 0003

      0057 0057 40F88852
      0058 0058 57C00002
      0059 0059 08000003
    1 005A 005A 01E00001            INCR       FP12
    2 005B 005B 40008BA1            SCW        C4,L2
      005C 005C 4FC080BF
      005D 005D 57C00000
      005E 005E 08000003
    3 005F 005F 01F00001            INCR       FP12
    4 0060 0060 40008BA1            SCW        C4,L2
      0061 0061 4FC080BF
      0062 0062 42008840
      0063 0063 40F8885A
      0064 0064 4000885A
      0065 0065 57C00002
      0066 0066 08000003
    5 0067 0067 01E00001            INCR       FP12
    6 0068 0068 01010001            DECR       FL1
    7 0069 0069 2891004C            BEZ,FL1    CASE3
    8 006A 006A 2801006B            B          RETURN
    9                       ;
   10                       ;
   11                       ;
   12 006B 006B 34A00000 RETURN     LI          1,0
   13 006C 006C 30810000            SF         (L,DP),GPLAB
   14 006D 006D 280F0000            B          X(F7)
   15                FFFF           END


LINEIN     APPLE V04-00 16-JUL-80 04:41:14 PAGE 0004

ERRORS DETECTED: 00000
  RNINGS DETECTED: 00000
```

AD-A111 882 MISSISSIPPI STATE UNIV MISSISSIPPI STATE DEPT OF ELEC--ETC F/G 5/8
THE APPLICATION OF SPECIAL COMPUTING TECHNIQUES TO SPEED-UP IMA--ETC(U)
DEC 81 R W MCLAREN, W D MCFARLAND F30602-80-C-0032
UNCLASSIFIED RADC-TR-81-230 NL

3 OF 3
AD A
111882

END
DATE
FILMED
04-82
DTIC



```
                                LINEOUT   START
                                ;
                                ;         MOVE RESULT FROM A4 TO BC
                                ;
                                          ENTRY     LINEO
                                          EXTRN     PUTAD, LTEOFF
                                ;
                                ;
                                ;
                        6106    RESULT    DF        97,8
                        0008    C1        DF        0,8
                        0808    C2        DF        8,8
                        1008    C3        DF        16,8
                        1808    C4        DF        24,8
                                ;
                                ;
                                ;
                        0000    LINEOUT   EQU       $
0000 0000 34A00080              LI        BL,128
0001 0001 42810000              LR        DE,PUTAD
0002 0002 73C00000              LI        FP12,0
0003 0003 47C088A5      REPEAT  LCM       C1,RESULT
0004 0004 401F8885
0005 0005 401088FF
0006 0006 25C38038
0007 0007 01E00001              INCR      FP12
0008 0008 47C088A5              LCM       C2,RESULT
0009 0009 401F8884
000A 000A 401C8858
000B 000B 25C34878
000C 000C 01E00001              INCR      FP12
000D 000D 47C088A5              LCM       C3,RESULT
000E 000E 401F88EF
000F 000F 401888E8
0010 0010 25C37088
0011 0011 01E00001              INCR      FP12
0012 0012 47C088A5              LCM       C4,RESULT
0013 0013 401F8888
0014 0014 401888B8
0015 0015 25C398F8
0016 0016 01E00001              INCR      FP12
0017 0017 30050000              SR        C,OUTBUFF(DE),3
0018 0018 29110003              BNZ,BL    REPEAT
0019 0019 34A00000              LI        BL,0
001A 001A 40810000              SR        (BL,DE),PUTAD
001B 001B 280F0000              B         0(R7)
                        FFFF    END
```

```
FORMAT     APPLE V04-00 16-JUL-80 04:43:27 PAGE 00001

                          FORMAT     START
                          ;
                          ;  FORMAT A 3*3 MATRIX INTO A LINEAR STRUCTURE, I.e.,
                          ;  FORMAT   6   7   8
                          ;            5   X   1
                          ;            4   3   2
                          ;
                          ;  INTO  1   2   3   4   5   6   7   8
                          ;
                                     ENTRY     FORMAT
                          ;
                          ;
                     0808 L1         DF        6,8
                     1008 L2         DF        16,8
                     1608 L3         DF        24,8
                     1608 P1         DF        24,6
                     2008 P2         DF        32,6
                     2808 P3         DF        40,8
                     3008 P4         DF        48,8
                     3808 P5         DF        56,8
                     4008 P6         DF        64,8
                     4808 P7         DF        72,8
                     5008 P8         DF        80,8
                          ;
                          ;  MOVE ELEMENTS 2,3,AND 4 TO POSITIONS
                          ;
                     0000 FORMAT     EQU       $
0000 0000 40001543                   SET       M
     0001 0001 48060003
0002 0002 37902718                   MVF       L3,P2
     0003 0003 3E670006
     0004 0004 43348845
     0005 0005 4E800001
     0006 0006 13A40003
```

```
30 0007 0007 37902F17          MVF     L2,P3
   0008 0008 3F070008
   0009 0009 43348BA5
   000A 000A 48800001
   000B 000B 13A40004
31 000C 000C 37903 0F          MVF     L1,P4
   000D 000D 3F079010
   000E 000E 43348BA5
   000F 000F 48800001
   0010 0010 13A40003
32                        ;
33                        ;  SHIFT UP ELEMENTS 2,3, AND 4 ONE LINE
34                        ;
35 0011 0011 73C00100          LI      FP12,256
36 0012 0012 67C1A08D          LCM     (0,24),(32,24)
37 0013 0013 73C00000          LI      FP12,0
38 0014 0014 40008841          SCW     (0,24),(32,24)
   0015 0015 4FC1A08F
   0016 0016 57C10000
   0017 0017 48000003
39 0018 0018 73900020          LI      FP1,32
40 0019 0019 33400020          LI      FP2,32
41 001A 001A 3F17001E          LOOP,24 FSHIFTUP
42 001B 001B 43308845          L       Y,FP1+
```

```
 1 001C 001C 40FF865A              ROT       Y,-1
   001D 001D 40008858
 2 001E 001E 1B600002 FSHIFTUP     S         X,FP2+
 3                    ;
 4                    ;  MOVE ELEMENTS 6,7,AND 8 TO POSITION
 5                    ;
 6 001F 001F 37405710              MVF       (8,24),(64,24)
   0020 0020 3F170023
   0021 0021 43348885
   0022 0022 41500001
   0023 0023 13A40003
 7                    ;
 8                    ;  SHIFT DOWN ELEMENTS 6,7,AND 8 ONE LINE
 9                    ;
10 0024 0024 73C000FF              LI        FP12,255
11 0025 0025 57C2A0F0              LCM       (0,24),(64,24)
12 0026 0026 73C001FF              LI        FP12,511
13 0027 0027 40008881              SCW       (0,24),(64,24)
   0028 0028 4FC2A0FF
   0029 0029 57C20000
   002A 002A 48000003
14 002B 002B 73900040              LI        FP1,64
15 002C 002C 33400040              LI        FP2,64
16 002D 002D 3F170030              LOOP,24   FSHIFTDN
17 002E 002E 43308845              L         Y,FP1+
18 002F 002F 40FF8852              ROT       Y,1
19 0030 0030 5B600002 FSHIFTDN     S         Y,FP2+
20 0031 0031 37403F0F              MVF       L1,P5
   0032 0032 3F070035
   0033 0033 43348885
   0034 0034 40800001
   0035 0035 13A40003
21 0036 0036 2B0F0000              B         0(97)
22                FFFF             END
```

```
STORAGE   APEEF V04-00 16-JUL-80 04:44:04 PAGE 00001

                                STORAGE   START
                                ;
                                ;         THIS SEGMENT IS FOR CONSTANT AND
                                ;         RESERVED STORAGE.  IT IS PUT IN
                                ;         HIGH SPEED DATA BUFFER (HSDB)
                                ;
                                          ENTRY     WDCNT,BLKNUM,INBUFF,OUTBUFF,GETA
                                          ENTRY     CONDITION,ELEMENCNT,THRSHLD,TOTAL,
                        0608              ORG       HSDB,A
                        1000 WDCNT        EQU       4096
                        0000 BLKNUM       EQU       0
                        A000 INBUFF       EQU       X'A000'
                        C000 OUTBUFF      EQU       X'C000'
                        0608 GETAD        DS
                        0609 PUTAD        DS
                        060A CONDITION    DS
                        060B ELEMENCNT    DS
060C 060C 00000064      THRSHLD           DC        100
0600 0600 00000000      TOTAL             DC        0
060F 060E 00000000      ELEMENT           DC        0
060E 060E 00000001      SLOT              DC,X      1
0610 0610 00000001
0611 0611 00000001
0612 0612 00000001
0613 0613 00000001
0614 0614 00000001

*** LISTING OF REMAINDER OF DC SUPPRESSED ***

                        FFFF              END


                                          ERRORS DETECTED: 00000
                                          WARNINGS DETECTED: 00000
```

```
                                ALLIO     START
                                          ENTRY    INITS,TRANINS,TRANOUTS,RELEASES
                                          ENTRY    POINTIN,POINTOUT,TRANITIN,TRANOUT
                                          ENTRY    LNKWDIN,LNKWDOUT
                                ;
                                ;
                        0000    INITS     EQU      S
                                          INIT     POINTIN
   0000 0000 72808180
   0001 0001 34A00014
   0002 0002 30C18010
                                ;
                                          INIT     POINTOUT
   0003 0003 7280818C
   0004 0004 34A00014
   0005 0005 30C18010
   0006 0006 280F0000                     B        0(R7)
                                ;
                                ;
                        0007    TRANINS   EQU      S
                                          TRAN     TRANITIN
   0007 0007 72808184
   0008 0008 34A00016
   0009 0009 30C18010
                                ;
                                          LOWAIT   LNKWDIN
   000A 000A 7280819C
   000B 000B 74A60000
   000C 000C 37209000
   000D 000D 30C18010
   000E 000E 280F0000                     B        0(R7)
                                ;
                                ;
                        000F    TRANOUTS  EQU      S
                                          TRAN     TRANOUT
```

```
   000F 000F 7280819C
   0010 0010 34A00018
   0011 0011 30C18010
27                          ;
28                                        IOWAIT    LNKWDOUT
   0012 0012 72808188
   0013 0013 74A00000
   0014 0014 37200000
   0015 0015 30C18010
29 0016 0016 280F0000                     B         0(R7)
30                          ;
31                          ;
32                  0017 RELEASES         EQU       S
33                                        RLSE      LNKWDIN
   0017 0017 7280819C
   0018 0018 34A00018
   0019 0019 30C18010
35                          ;
36                                        RLSE      LNKWDOUT
   001A 001A 72808188
   001B 001B 34A00018
   001C 001C 30C18010
38 001D 001D 280F0000                     B         0(R7)
```

```
1                           ;
2                           ;
3                   81A0 POINTIN    EQU       X'81A0'
4                   818C POINTOUT   EQU       X'818C'
5                   81A4 TRANITIN   EQU       X'81A4'
6                   8190 TRANOUT    EQU       X'8190'
7                   819C LNKWDIN    EQU       X'819C'
8                   8188 LNKWDOUT   EQU       X'8188'
9                   FFFF            END
```

ERRORS DETECTED: 00000
WARNINGS DETECTED: 00000

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

ATE

LMED

8