THE PDP 9 AS A CONTROL ELEMENT FOR A PARALLEL COMPUTER

by

Samuel A. Miller

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1968

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_____ 10 May 1962

F. J. HILL — Date

Associate Professor of Electrical Engineering

ACKNOWLEDGEMENT

The author is particularily grateful to Dr. F. J. Hill whose guidance during the research leading to this thesis and advice during the preparation of the initial and final manuscripts was invaluable.

TABLE OF CONTENTS

TABLE OF CONTENTS— Continued

LIST OF ILLUSTRATIONS

LIST OF TABLES

ABSTRACT

The feasibility of designing a parallel entry computer to be controlled by the PDP 9 computer was investigated. An array of eight processing elements was considered; the design of a processing element was completed and the cost of the hardware for the processing element was determined to be $2350. A problem with parallel characteristics was programmed on the PDP 9 computer in the assembly and Fortran languages and the same problem was programmed on the parallel computer. The speed advantage was 1.8-1 compared with the PDP 9 assembly language program and 150-1 compared with the Fortran program. It was determined that with minor changes in the control sequencer, the speed advantages of the parallel computer could be increased to 10-1 compared with the fastest PDP 9 program.

CHAPTER 1

INTRODUCTION

Rapid changes in the field of computer science have resulted in more efficient data processing and have enabled the user of the computer to solve large, complex problems more accurately and more efficiently. These changes have been the result of great technological strides in allied sciences which led to improvements in computer hardware and from the development of efficient programming techniques. However, the basic organization of the digital computer today is little changed from that of the first computer ever built.

Even though the modern digital computer can efficiently solve complex problems, certain classes of problems still require extensive computation time and computer solution may therefore be uneconomical, even though sophisticated programming techniques are used.

Some of these time consuming problems have the property of parallelism, that is they can be reduced to sets of identical sub-problems such that each sub-problem may be solved simultaneously and independently or almost independently. Typical problems of this class are those involving numerical solution of partial differential equations, systems of ordinary differential equations, and operations

on matrices. Problems of this nature occur in areas such as photoreconnaissance, weather predicting and pattern recognition.

In order to solve such problems efficiently, the concept of parallel computation has been considered. The conventional digital computer is able to perform only one operation at a time; in order to solve a problem, the central processor must perform a sequence of operations step by step until the problem has been solved. Instructions are executed sequentially under command of the control unit which exercises control over all elements of the computer. The concept of parallel computation provides for an array of independent processors each with a memory and arithmetic capability and all under the control of a central control unit. The physical organization of such a computer corresponds to the geometrical interpretation of this class of problem. Each processor or "processing element" of the array independently solves one of the sub-problems of the main problem; however, all processors may solve a sub-problem simultaneously and all are under control of a central control unit.

Two parallel computers have been designed, the Solomon and Illiac IV with the latter being under construction. It has been judged that the savings in computation time when using parallel computers are orders of magnitude compared with even the fastest modern digital computer so that savings in computation time can more than offset the increased cost of hardware. The Illiac and Solomon computers are however, large scale parallel computing machines. This

thesis will investigate the possibility of using parallel computation in conjunction with a small general purpose computer.

The purpose of this thesis therefore is to:

1. Investigate the feasibility of a parallel computer using an array of eight processing elements interfaced with a small general purpose computer (the PDP 9) using the PDP 9 to perform most control functions for the computer. An independent control sequencer (the subject of another thesis) will interface the PDP 9 with the processing elements.

2. Design a processing element with basic arithmetic capability and to estimate the cost of eight such processing elements.

3. Program on the PDP 9 computer a typical problem of the class previously described and to program the same problem on the parallel computer in order to compare the computation times assuming the memory cycle time of the processing element would be the same as that of the PDP 9.

CHAPTER 2

THE PARALLEL COMPUTER

The parallel computer proposed in this thesis consists of eight processing elements (PE's) each with basic arithmetic capability and independent memory under control of one control sequencer. The PDP 9 computer acts as the memory for the parallel computer control sequencer and the PDP 9 provides address modification and indexing where required. Instructions to be executed by the parallel computer are transfered via the PDP 9 I/O bus through an interface to the parallel computer.

The accumulator of each PE is tied to the PDP 9 I/O bus through its own interface to facilitate independent data transfer between any PE and the PDP 9. A broadcast register, also tied to the PDP 9 I/O bus, is common to all processing elements to enable data transfer from the PDP 9 to an arbitrary number of processing elements.

Processing elements are interconnected as an in-line array. Each processing element has the capability to receive data from the memory of its neighbor to the left or right as well as from its own memory. It may also receive data from the broadcast register. The source from which a PE receives its data is determined by two bits in the instruction word designated the "routing bits".

Each processing element has the facility to accept or reject a command from the control unit. This feature, known as modal control, facilitates the independent operation of each processing element. The programmer has the facility to change modes by means of special instructions to keep the majority of processing elements in continuous operation. Modal control is effected by four bits of the instruction word designated by "mode bits".

Each processing element is designed to operate in fixed point binary arithmetic with a memory of 256 words and a word length of 18 bits. Processing elements are designed using Digital Equipment Corporation logic cards so the system performance from a logic standpoint will be comparable with that of the PDP 9. Memory design and associated memory logic are not included; however, it is assumed that a memory with performance comparable to that of the PDP 9 could be procured.

Ten interface units between the PDP 9 and the parallel computer are required. The design of these units is completely described in PDP 9 User's Handbook and is therefore not discussed in this thesis. Each unit of the parallel computer which is connected to the PDP 9 through an independent interface must be assigned a device number for use by the PDP 9 input - output transfer instruction.

The system block diagram shown in Figure 2.1 displays the salient features of the parallel computer. A detailed description of the processing elements appears in Chapter 4 and the description and design of the control sequencer is the subject of another thesis.

These features give the parallel computer the basic capability of a large, independent parallel computer and provide the programmer with sufficient latitude to program most problems which have parallel characteristics.

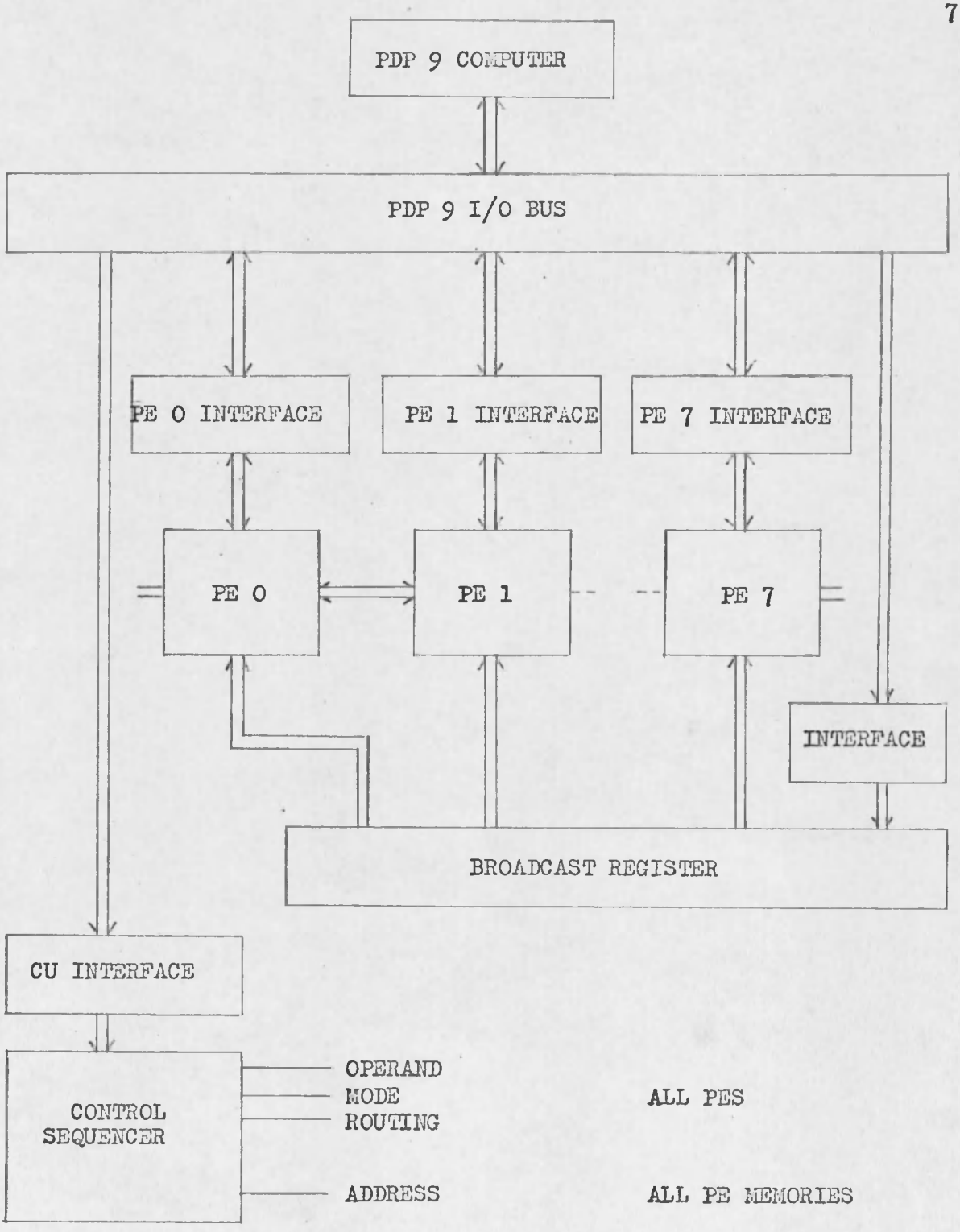


Figure 2.1. Parallel Computer System Diagram

CHAPTER 3

PARALLEL COMPUTER INSTRUCTIONS

The instruction word of the PDP 9 computer is 18 bits long. Since the parallel computer is to be directly compatible with the PDP 9, its instruction word will also consists of 18 bits.

The format for the parallel computer instruction word is shown in Figure 3.1. The bit arrangement as specified in Figure 3.1 applies to all instructions with the exception of the shift, examine and set mode instructions each of which has a distinctive format.

Operand assignments for the parallel computer instructions are shown in Table 3.1 and routing and modal assignments are shown in Table 3.2.

The mode bits of the instruction word designate any of the 16 combinations of the four modes 0, 1, 2 and 3. Therefore, if the modes designated by the instruction were 12 those processing elements whose mode states were 1 or 2 would accept the instruction word.

The routing bits designate internal, left, right or broadcast indicating from where the processing element is to receive its information. The broadcast register is common to all processing elements and each processing element has access to its own memory or to the memories of its left or right neighbor under control of a routing command.

The address portion of the instruction word consists of 8 bits which will admit a memory size of 256 eighteen bit words.

Instruction List

The parallel computer is designed to execute a total of ten instructions. This repertoire provides the computer with the capability to perform basic arithmetic, logical, shift and accumulator test operations. Other instructions are under consideration.

Clear Add

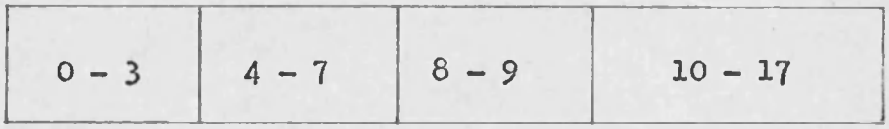
This instruction clears the processing element accumulator and places the contents of the memory location specified by the address field in the accumulator. The routing bits will specify which processing element memory (or the broadcast register) is the information source. Only those processing elements whose mode state agrees with the mode states specified by the instruction word will respond to the instruction.

Add

Add the contents of the location specified by the address field to the accumulator. Routing and modal control apply to this instruction so that a processing element may obtain information from any of four sources and may accept or reject the instruction.

Subtract

The contents of the location specified by the address field are subtracted from the accumulator. Routing and mode states must



Bits		
	0 - 3	Operand
	4 - 7	Mode
	8 - 9	Routing
	10 - 17	Address

Figure 3.1. Parallel Computer Instruction Word

Table 3.1

Operand Assignments
Parallel Computer Instruction Word

Instruction	Operand (Octal)
Clear Add	00
Add	01
Subtract	02
Logical AND	03
Exclusive OR	04
Load Mode	05
Store Mode	06
Store Accumulator	07
Examine	10
Set Mode	11
Shift	12

Table 3.2

Mode and Routing Assignments
Parallel Computer Instruction Word

Mode (Bits 4 - 7)	Mode Number
0000	None
1000	0
0100	1
0010	2
0001	3
0011	2,3
0101	1,3
0110	1,2
0111	1,2,3
1001	0,3
1010	0,2
1011	0,2,3
1100	0,1
1101	0,1,3
1110	0,1,2
1111	0,1,2,3

Routing (Bits 8 - 9)	Route
00	Internal
01	Left
10	Right
11	Broadcast

be specified in the instruction.

Logical AND

The logical operation AND is performed between the contents of the addressed location and the contents of the accumulator on a bit by bit basis. Routing and modal control apply.

Exclusive OR

The logical operation exclusive OR is performed on a bit by bit basis between the contents of the addressed register and the contents of the accumulator. Routing and mode information must be specified.

Load Mode

This operation sets the mode flip flops in the processing elements to the state specified by the contents of the addressed location. Routing for this instruction must be internal. The mode state of the processing element must agree with one of the modes specified by the instruction word.

Store Mode

This instruction obtains the processing element mode state and stores it in the specified address. The two least significant bits of the data word will contain the mode; the remainder of the bits in the data word will be zero. Routing must be internal and modes must be specified.

Store Accumulator

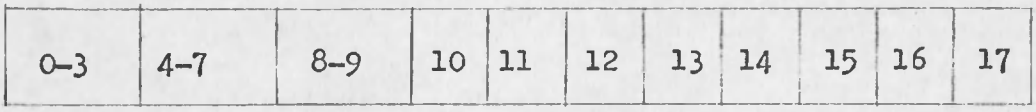
The contents of the accumulator are stored in the specified address. Routing must be internal. Modal control applies to this instruction.

All instructions previously discussed have the format as shown in Figure 3.1. For these instructions, all bits of the instruction word must be specified with the exception that if broadcast routing is applied the address field need not be specified since the broadcast register is not addressable. In the case of the store accumulator, store mode, and load mode instructions routing is always internal.

Three instructions of those available to the programmer have special formats. These instructions are the examine, set mode and shift instructions and the formats are shown in Figures 3.2, 3.3, and 3.4.

Examine

The examine instruction tests the state of the accumulator for one of eight different conditions as is indicated in Figure 3.2. Routing bits are immaterial for this instruction but mode states must be specified. One of bits 10 - 17 must be set to the binary state "one" with each particular bit of bits 10 - 17 specifying a different test. Inclusive "ORing", that is setting more than one of bits 10 - 17, is not permitted. Bits 0 - 7 of the broadcast

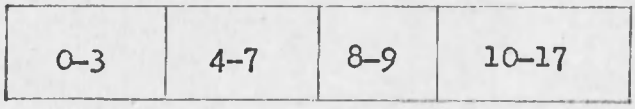


Bits	Function
0-3	Operand 1000
4-7	Mode
8-9	Immaterial

Bit	State of Accumulator
10	All 1's
11	All 0's
12	Greater than 0
13	Greater or equal to 0
14	Equal to 0
15	Less than 0
16	Less than or equal to 0
17	Not equal to 0

Only one of bits 10-17 may be 1

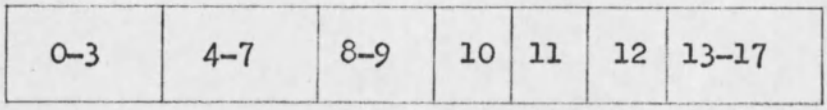
Figure 3.2. Examine Instruction - Format



Bits	Function
0-3	Operand 1001
4-7	Mode
10-17	Immaterial

Broadcast	Bit 8	Bit 9	Mode
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	2
0	1	1	3
1	0	1	3
1	1	1	2

Figure 3.3 Set Mode Instruction - Format



Bits	Function
0-3	Operand 1010
4-7	Mode
8-9	Routing
10	Include sign bit in shift = 1
11	Left shift =0 Right shift = 1
12	Rotate acc =0 Shift over end = 1
13-17	Shift count

Figure 3.4. Shift Instruction - Format

register record the results of the test with each of the eight bits recording the result for a different processing element. A successful test results in a binary "one" being placed in the appropriate bit of the broadcast register. For example, if the accumulator of PE #6 is to be tested for "equal to zero", bit 14 of the instruction word is set to "1" and the mode of PE #6 must be one of the modes specified by the instruction word. If, after execution of the instruction, the test was successful, i.e., the accumulator was zero, a "1" is placed in bit 6 of the broadcast register. If the test was unsuccessful, a zero is placed in this bit. If a processing element does not participate in the test, the corresponding bit in the broadcast register remains zero.

Set Mode

This instruction is one which is executed on the basis of the examine instruction. When used, this instruction will normally follow the examine instruction in the instruction sequence. It may, however, be used at any time as long as the broadcast register retains the result of the previous examine instruction. The format for this instruction is shown in Figure 3.3. The mode of a processing element is set according to the contents of its assigned bit in the broadcast register (which has been set on the basis of the examine instruction) and according to the contents of bits 8 and 9 of the instruction word. The address field in this instruction is irrelevent but modes must be specified in the instruction word.

Shift

In this command, routing and modes must be specified. The format for the instruction is shown in Figure 3.4. Bit 10 specifies whether the sign bit participates in the shift and bits 11 and 12 specify the shift direction and whether the accumulator is to be rotated. Bits 13 - 17 specify the shift count.

Proposed Instructions

Two instructions have been proposed but have not been implemented in the design. These commands are iterate and search. It proposed that the iterate instruction be designed to cause a sequence of n instructions to be repeated m times with n and m being specified in the instruction word. The search instruction would be designed to scan all or a portion of memory for a word identical to the content of the accumulator. The use of the iterate command is obvious and the search command could be used for comparison tests such as testing for convergence.

Mnemonics

In order to facilitate programming, a set of mnemonics has been devised for the operand and routing bits as shown in Table 3.3. Operand mnemonics consists of four letters to distinguish them from PDP 9 mnemonics which contain three letters. The following instruction illustrates the use of mnemonics:

PCLA 12 L 100

This instruction is interpreted to mean clear the accumulator and

Table 3.3

Parallel Computer Mnemonics

Operand:

PCLA	-	clear add
PADD	-	add
PSUB	-	subtract
PAND	-	logical AND
PXOR	-	exclusive OR
PLDM	-	load mode
PSTM	-	store mode
PSTA	-	store accumulator
PEXA	-	examine
PSEM	-	set mode
RSHF	-	right shift
LSHF	-	left shift

Routing:

I	-	internal
L	-	left
R	-	right
B	-	broadcast

General:

C	-	control

add to the accumulator the contents of location 100 from the memory of the left neighbor. Only those processing elements whose mode state is 1 or 2 will accept the instruction. Mnemonics for other commands are written in a similar manner.

Since PDP 9 instructions will also be used in programming, a selected group of PDP 9 mnemonics is listed in Table 3.4 for reference.

Table 3.4

PDP 9 Mnemonics

LAC 100	-	Load the accumulator with the contents of location 100
LAC (100	-	Load the accumulator with the octal number 100
LAC I 100	-	Load the accumulator with the contents of the location whose address is stored in location 100
DAC 100	-	Deposit accumulator contents in location 100
ADD	-	Add, one's complement
TAD	-	Add, two's complement
SAD 100	-	Skip the next instruction if the accumulator differs from the contents of location 100
ISZ 100	-	Increment the contents of location 100 and skip the next instruction if these contents are zero
HLT	-	Halt
JMP 100	-	Obtain the next instruction from location 100
XCT 100	-	Execute the instruction stored in location 100
NOP	-	No operation
IOT Y	-	Input/output transfer to or from device number Y

CHAPTER 4

PROCESSING ELEMENT DESIGN

The basic computing unit in the parallel computer is the processing element. The processing element is designed to perform fixed point binary arithmetic using a data word of 18 bits length and the execute the instructions designated in Chapter 2.

The essential features of the processing element are its capability to accept or reject instructions (modal control) and its ability to receive data from four points: internal memory, the memory of its left or right neighbor and the broadcast register.

The design outline in this chapter is an elementary design only; not included are the design of the memory and associated logic and the accumulator design which has been included in the design of the control sequencer.

A block diagram of the processing element is shown in Figure 4.1. Data are routed from one of three memories or the broadcast register into one of the four 18 bit AND gates which has been enabled by the routing signal from the control sequencer. The AND gates drive an OR circuit which gates the data into the adder or into the logic required to implement the AND and the exclusive OR operations. A diagram showing the link between the memory and the adder is shown in Figure 4.2. This drawing shows the connection for one bit only;

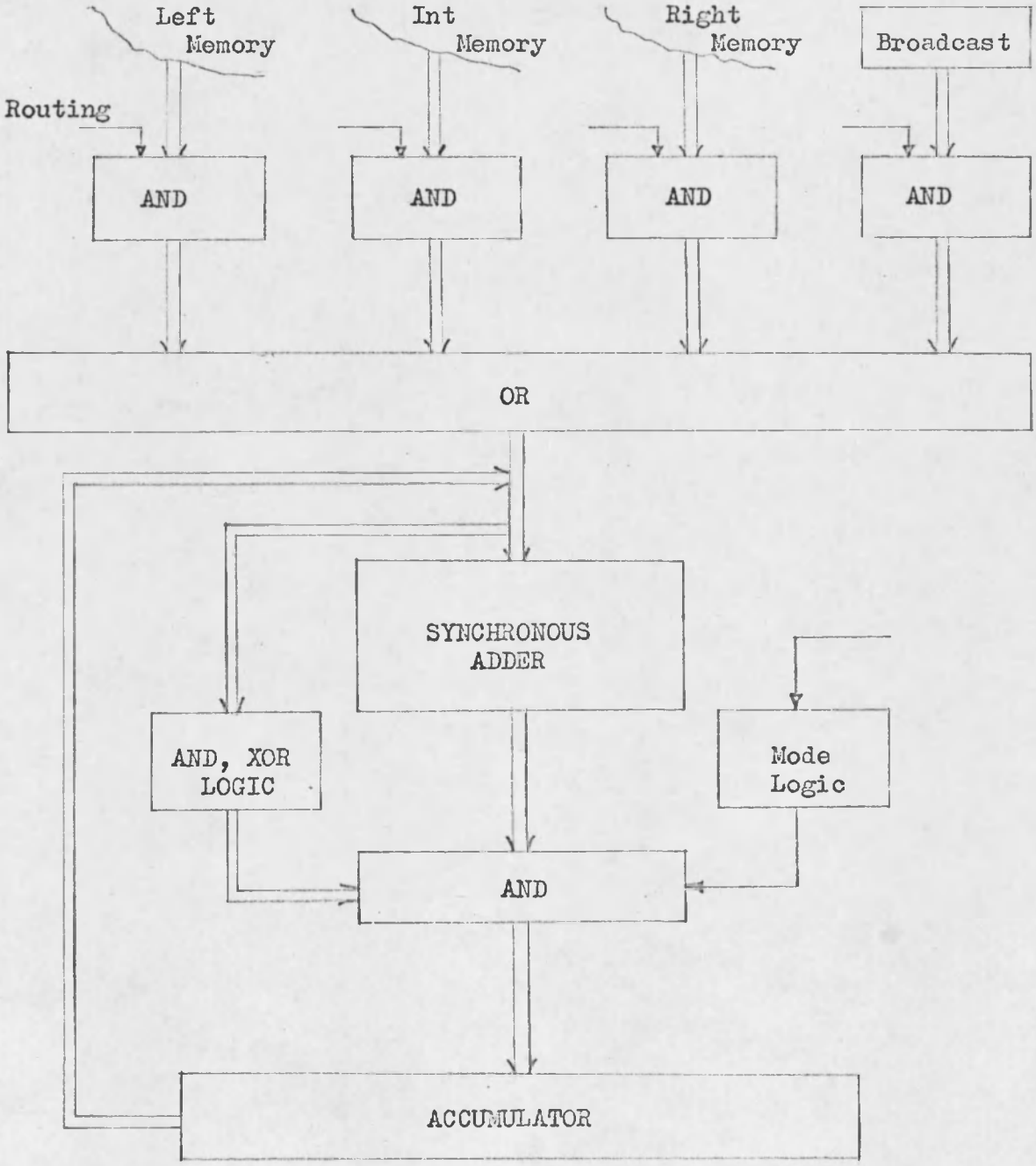


Figure 4.1. Processing Element – System Diagram

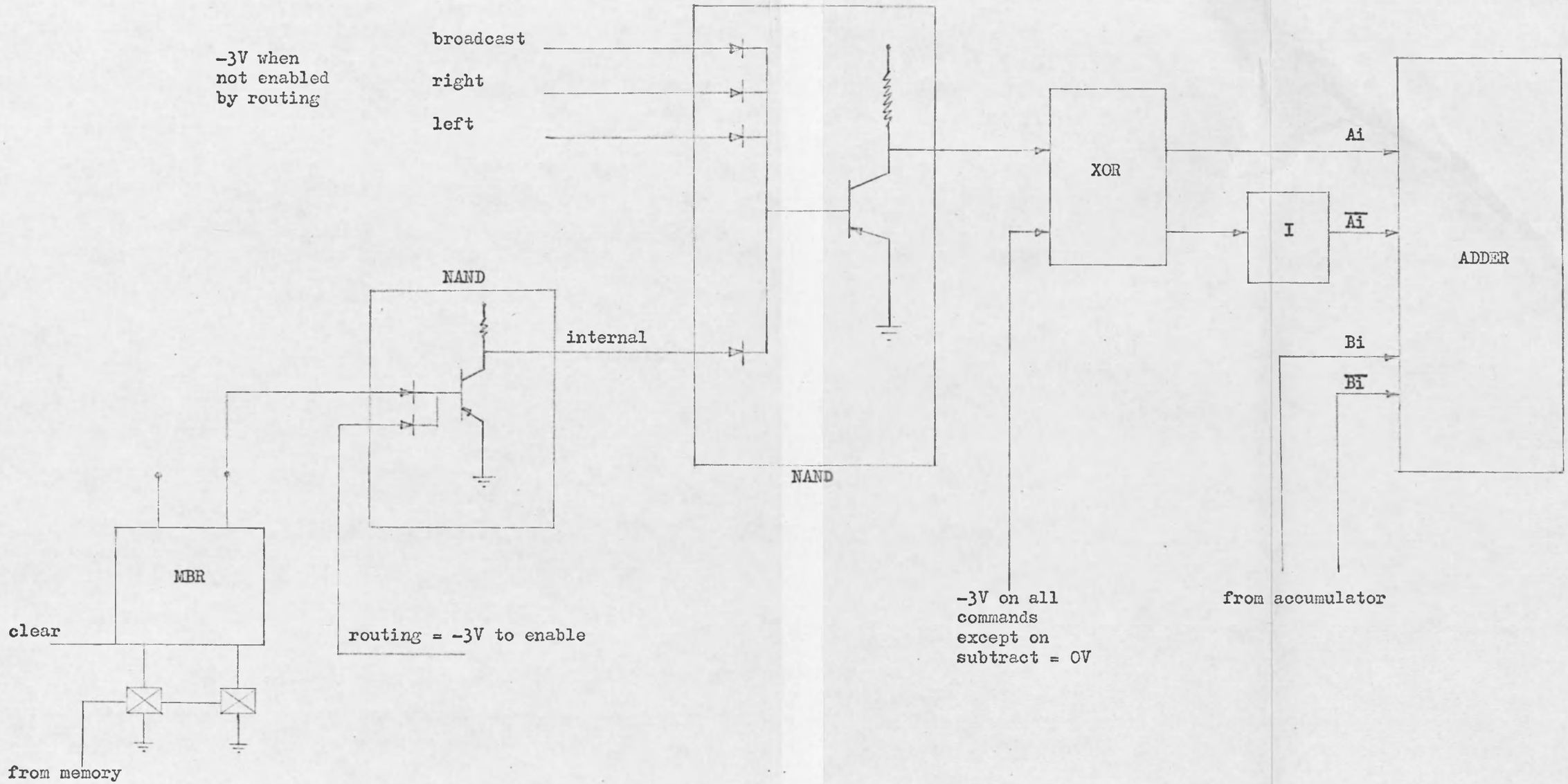


Figure 4.2. Memory - Adder Link - One Bit

the circuit was realized using two NAND gates. Also shown at the adder input is the logic used to implement the subtract operation.

The adder used is the synchronous adder shown in Figure 4.3.

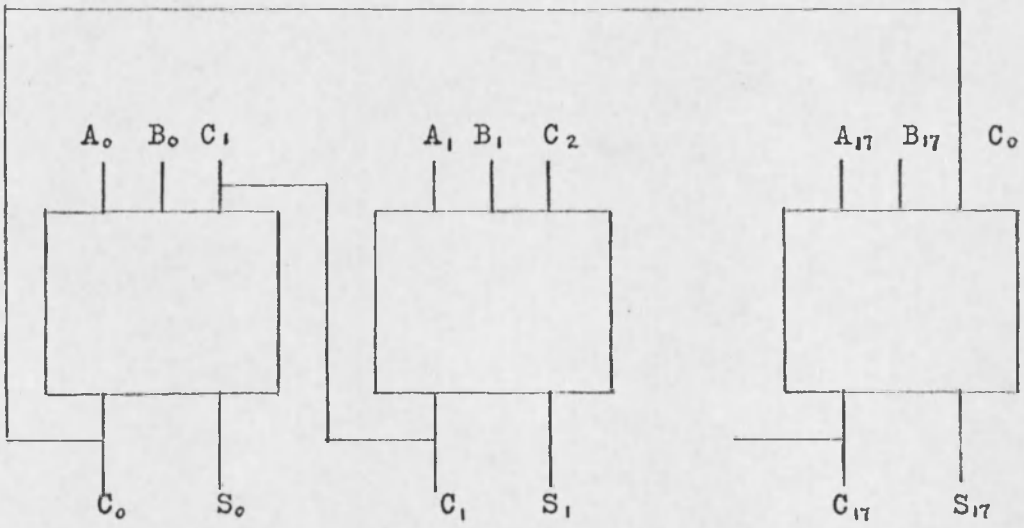


Figure 4.3. Synchronous Adder

Although this is the basic adder, it was chosen because of the long I/O transfer time from the PDP 9 which will allow a full 18 bit carry propogate (18 bit propogate time is 1.3 usec). The standard NAND gate realization of one stage of the adder is shown in Appendix A. The adder consists of combinational logic, the output of which drives an 18 bit AND gate which is gated on or off by the mode logic. It should be noted that in this system, the logic preceding this AND gate is in continuous operation, whether or not a particular

instruction is designated for a specific processing element. Only the mode logic determines whether the results of a specific operation will reach the accumulator. The subtract operation is performed in 1's complement arithmetic; therefore, end around carry will occur if negative and positive numbers are added and the positive number is greater in magnitude. End around carry also occurs if two negative numbers are added together.

The output of the adder which would normally be fed directly to an accumulator register is fed to an AND gate (Figure 4.1) which is enabled by the mode signal. The mode gate is enabled only if the mode state of the processing element flip flops is a subset of the modes specified by the instruction word. The interconnection of the adder with the accumulator register is shown in Figure 4.4 which also shows the input to the mode flip flops and the logic required to implement the logical AND and the exclusive OR operations.

The mode logic diagram is shown in Figure 4.5. The mode gate enable signal is determined by both the state of the mode flip flops and the status of bits 4-7 of the instruction word. This logic provides a mode enable signal to those processing elements whose mode states agree with the modes specified by the instruction word. The logic diagram as shown in Figures 4.2 - 4.5 implement all instructions except store accumulator, examine, set mode and shift.

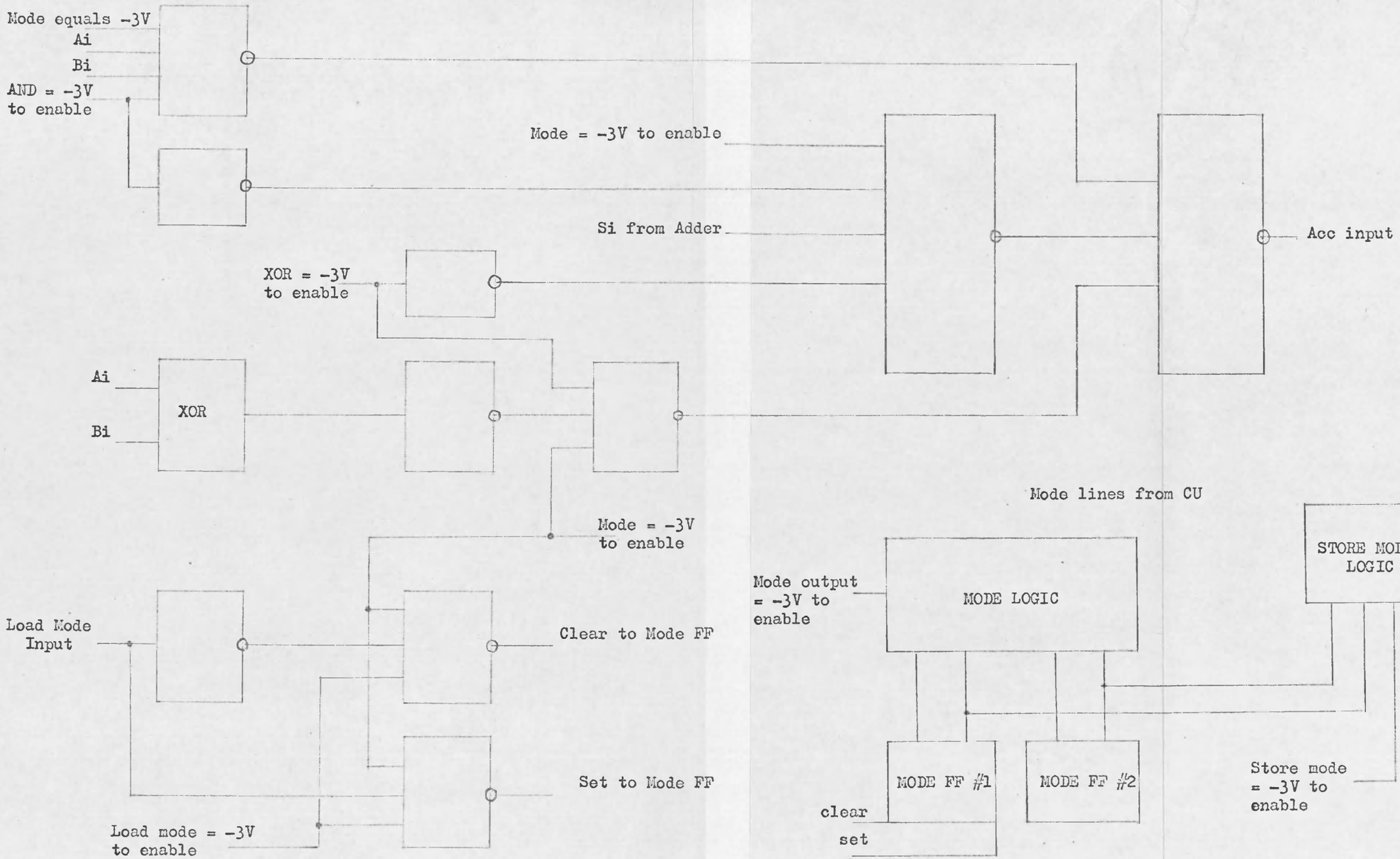


Figure 4.4. Adder - Accumulator Interconnection

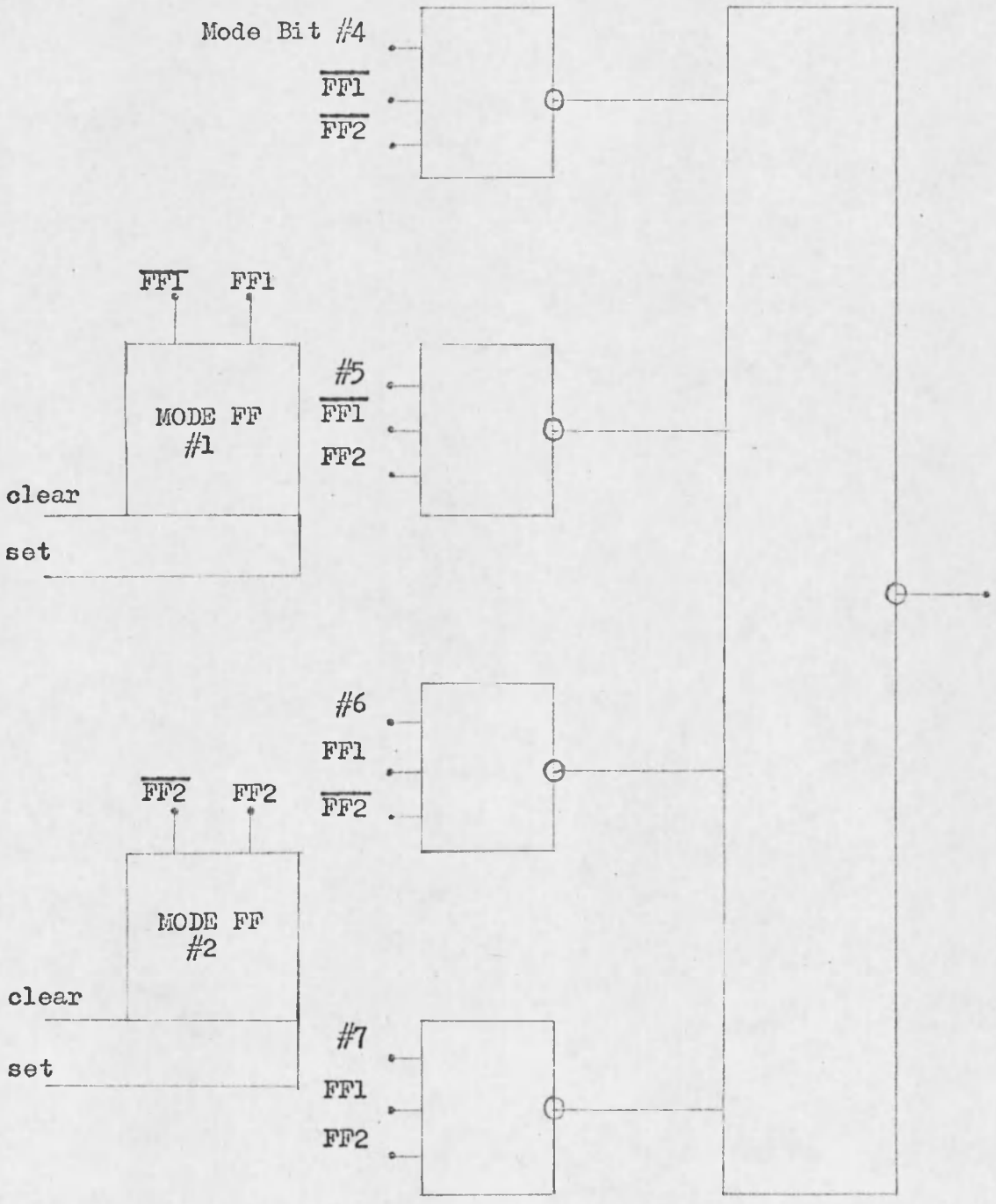


Figure 4.5. Mode Logic Diagram

Cost

The logic diagrams are implemented using Digital Equipment Corporation logic cards. Although this is not the most economical solution, this system was chosen to make the parallel computer directly compatible with the PDP 9. The cost breakdown is shown in Table 4.1. The broadcast register, accumulator and memory buffer costs are not included. The total cost of circuit cards for one processing element is $2,370.00.

Table 4.1

Processing Element Cost

Memory - Adder Link

Circuit	Total	Cards
R111 NAND	72	24
R001 Diode Net	36	5 1/7
R131 XOR	18	4 1/2
R603 Pulse Amp	2	2/3
R107 Inverter	18	2 4/7

Adder

R111 NAND	162	54
R107 Inverter	18	2 4/7
R001 Diode Net	126	18

End Around Carry

R111 NAND	1	1/3
R603 Pulse Amp	1	1/3

Load Mode Logic, Mode Flip Flops

R107 Inverter	2	2/7
R111 NAND	2	2/3
R001 Diode Net	2	2/7
R200 SC FF	2	2

Adder - Accumulator Link

R107 Inverter	57	7 5/7
R111 NAND	72	24
R001 Diode Net	108	15 3/7
R131 XOR	18	4 1/2

Mode Logic

R111 NAND	6	1 2/3
R001 Diode Net	8	6/7
R603 Pulse Amp	2	2/3

Store Mode Logic

R107 Inverter	2	2/7
R111 NAND	2	2/3

Table 4.1
(Continued)
Processing Element Cost

	Total	
R001	40 @ $ 4.00	$ 160.00
R107	14 @ $24.00	$ 336.00
R111	106 @ $14.00	$1484.00
R131	9 @ $35.00	$ 315.00
R200	2 @ $ 9.50	$ 19.00
R603	2 @ $28.00	$ 56.00
	Total Cost	$2370.00

CHAPTER 5

PDP 9 COMPUTER PROGRAMMING

The purpose of designing a parallel computer is to provide an increase in computation speed which is significant enough to offset the cost of increased hardware. Problems which have parallel characteristics are ideally suited for programming on this type of computer.

In order to form a basis for comparison of the parallel computer with a conventional computer, the solution of Poisson's and Laplace's equations, problems with parallel characteristics, was chosen for programming on the PDP 9. Programming of the problems on the PDP 9 allowed both an evaluation of the programming technique and provided a standard for comparing computation times. The problem was programmed in PDP 9 assembly language and Fortran IV.

Poisson's equation in two dimensions

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y)$$

becomes Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

for $f(x,y) = 0$

The solution of these equations by means of a digital computer has been the subject of much discussion in literature on numerical analysis and various methods of computer solution are still under investigation.

A method popularly used to solve these equations numerically is the method of finite differences. One of the many schemes available to solve equations by finite differences is the method of successive corrections, also known as successive relaxation or successive displacements.

In solving a partial differential equation by this method we consider a grid of points distributed over the limits of x and y separated by a distance Δh (Figure 5.1):

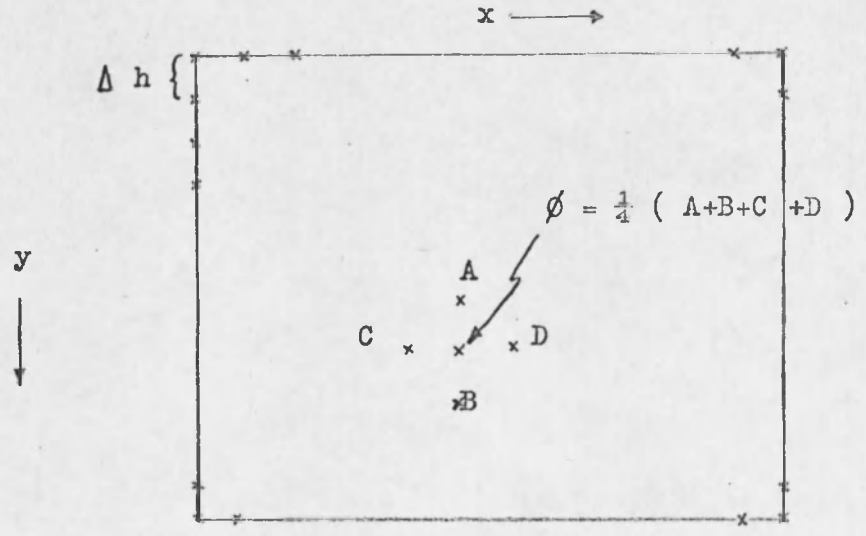


Figure 5.1. Partial Differential Equation – Finite Difference Method

The perimeter of the grid represents the known boundary conditions and, if the number of points in the grid is sufficiently large (i.e., Δh is small), we may approximate the value of any point ϕ by

$$\phi = \tfrac{1}{4} (A+B+C+D) + \tfrac{1}{4} h^2 f(x,y)$$

for Poisson's equation and

$$\phi = \tfrac{1}{4} (A+B+C+D)$$

Laplace's equation.

This is a partial difference equation, the derivation of which is presented in many texts on numerical analysis. To solve for all points in the matrix we iteratively compute the value of every point in the grid until the solution converges to within some desired limit. The error introduced by substitution of the difference equation for the differential equation is of the order of $2/3\, \Delta h^4$ provided there are no discontinuities in the solution. Under these conditions the solution is stable, that is the difference equation always converges. The separation, Δh, for the difference equation is determined by both the integration limits and the matrix dimension.

This method is not the best method by which to solve the equation. Another method is the method of successive overrelaxation in which the rate of convergence is measured and an overcorrecting factor is applied to improve convergence. Rate of convergence may also be improved by using different iteration paths such as

alternate row and column iteration. These equations may also be solved numerically by entirely different techniques such as Monte Carlo methods and by using Fourier transform techniques.

A number of programs were written for the PDP 9 in both assembly and Fortran languages. The purpose of these programs was both to gain familiarity with the PDP 9 computer and to investigate the accuracy of the successive corrections method for solving partial differential equations. The major effort was spent on programming in assembly language to gain the advantage of the rapid execution time of assembly language instructions by the PDP 9. Boundary conditions for the various problems were chosen to avoid scaling problems attendant in fixed point computations.

Both Poisson's and Laplace's equation were programmed in assembly language over a 63x63 and an 8x8 matrix. The boundary conditions in all cases were chosen to be constant, and in the case of Poisson's equation, the factor f (x,y) was also chosen to be constant. The programs were allowed to run until the computed value of a predesignated point in the matrix was the same on the n'th and (n-1) iterations through the matrix. The program for the solution of Laplace's and Poisson's equation was designed to output the computed value of the point at the center of the matrix every 100th iteration through the matrix. The program for the 8x8 matrix printed the entire matrix every 20 iterations.

Examination of the results of these programs revealed a computation error of approximately 2% in the final value of the point at the center of the matrix. The source of this error was determined to be the shifting of significant bits over the accumulator limits during the solution of the difference equation. As the solution converged to the final value, the error was introduced at the beginning of an iteration path and was propagated across the matrix dimension. This can be readily demonstrated by the example $\frac{1}{4}$ (A+B+C+D) = $\frac{1}{4}$ (15) = 3.75. The single precision fixed point computation however, yields the value 3. This problem was overcome by adding the value 1 on every alternate iteration through the matrix. This yielded solutions accurate to ±1 count for all cases. However, this increased the computation time for the solution of Laplace's equation over the 63x63 matrix from 10 seconds to 10.8 seconds per 100 iterations.

Computer outputs for Laplace's and Poisson's equation over a 63x63 and an 8x8 matrix are shown in Appendix B. Included are sample outputs for various boundary conditions and Poisson factors. Outputs for the same program, modified as previously described, are shown for the same matrix dimensions in Appendix C.

A special problem, Poisson's equation $\nabla^2\phi = -2$ in two dimensions, was programmed on a 25x25 matrix to demonstrate the accuracy of the finite difference method. This problem was programmed in Fortran IV and is shown in Appendix D along with output. The known solution is given for comparison purposes.

Two special programs were written to solve Laplace's equation over a 32x32 matrix. These programs are not designed to provide data output, but rather are to be used for direct computation time comparison with an identical problem written for the parallel computer.

PDP 9 Program - Assembly Language

A program was written in PDP 9 assembly language solve Laplace's equation over a 32x32 matrix. The flow chart for the PDP 9 program is shown in Figure 5.2 and the program listing is shown in Figure 5.3.

The problem is organized in a manner similar to program written for the parallel computer (Chapter 6). Effective use is made of the autoindex feature on the PDP 9 to provide address modification for solving the difference equation. End of row and end of matrix detection is provided to facilitate reloading of autoindex registers for the next iteration path. A test for convergence to the final value is made at the end of each iteration through the entire matrix. The solution is considered to have converged if the value of any point ϕ in the matrix on the n'th iteration is equal to the value of the same point on the (n-1) iteration.

In order to obtain a direct timing comparison with the program written for the parallel computer, the program will perform 100 iterations through the entire matrix before coming to a HALT.

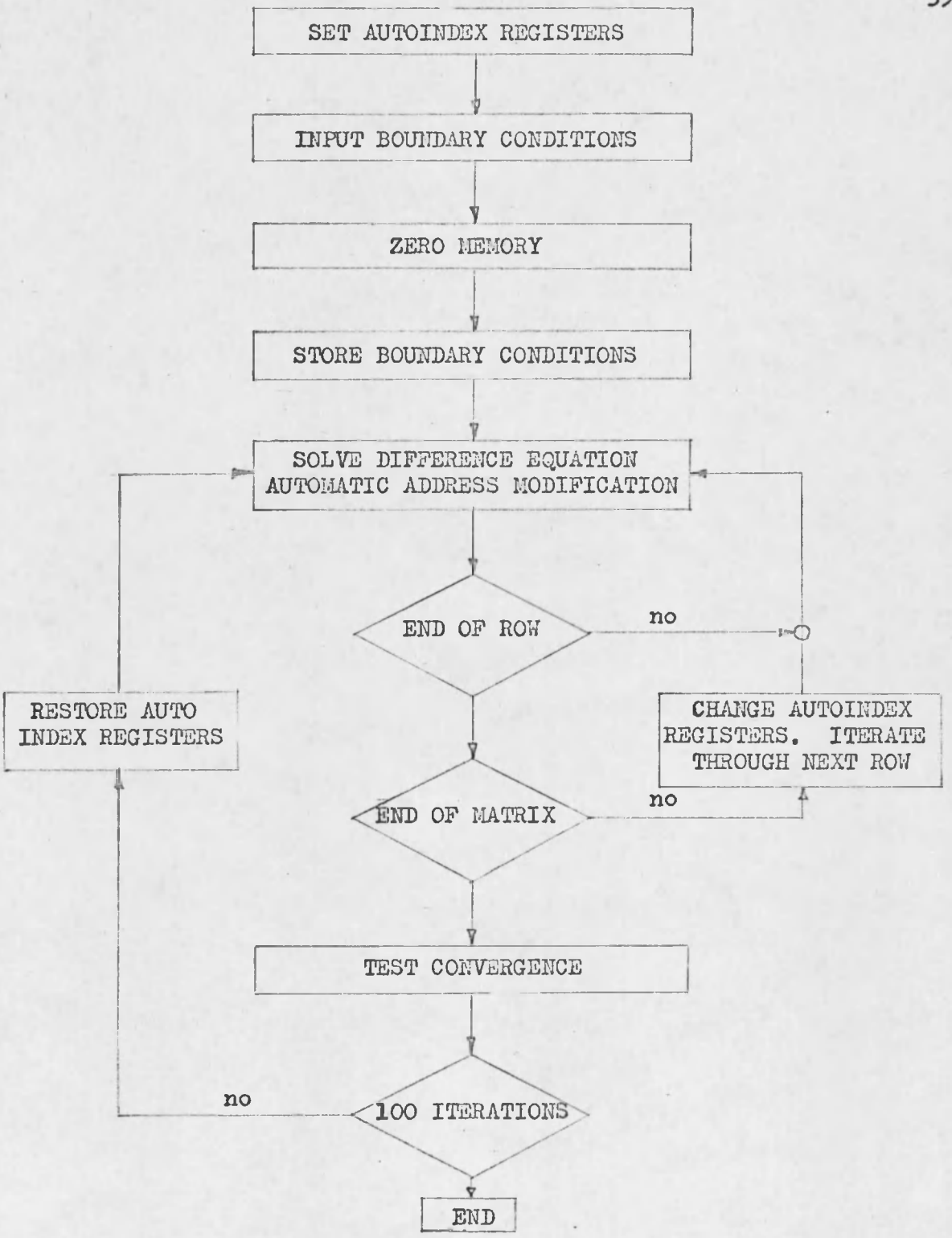


Figure 5.2. Flow Chart - PDP 9 Program

```
LAPLACE'S EQUATION 32X32
/FOR COMPARISON WITH PARALLEL COMPUTER PROGRAM
/INITIALIZE AUTO INDEX REGISTERS. AUTO INDEX REGISTERS
/CONTAIN LOCATIONS OF PHI'S TO BE USED IN DIFFERENCE
/EQUATIONS
10/       100      177       201       300
14/       200
10200/
/READ IN K (K = 0 FOR LAPLACE'S EQUATION). READ IN
/BOUNDARY CONDITIONS
          DECIN    DAC 10177  /K
          DECIN    DAC 10176  /UPPER BOUNDARY
          DECIN    DAC 10175  /LOWER BOUNDARY
          DECIN    DAC 10174  /LEFT
          DECIN    DAC 10173  /RIGHT
/ZERO MEMORY FROM LOC 100-4077  (2048 LOCATIONS)
S,        DZM I Y  ISZ Y      LAC Y
          SAD (4100           SKP       JMP S
/SET ALL BOUNDARY CONDITIONS
          LAC (100 DAC Y
UBOUND,   LAC 10176
          DAC I Y  ISZ Y      LAC Y
          SAD (140 SKP        JMP UBOUND
          LAC (3100           DAC Y
LBOUND,   LAC 10175
          DAC I Y  ISZ Y      LAC Y
          SAD (4100           SKP       JMP LBOUND
          LAC (100 DAC Y
LEFT,     LAC 10174
          DAC I Y  LAC Y      ADD (100  DAC Y
          SAD (4100           SKP       JMP LEFT
          LAC (137 DAC Y
RIGHT,    LAC 10173
          DAC I Y  LAC Y      ADD (100  DAC Y
          SAD (4137           SKP       JMP RIGHT
/ITERATE THROUGH 32X32 MATRIX, COORDINATES 100, 137,
/4000, 4037
/SOLVE 5 POINT DIFFERENCE EQUATION
A,        CLL
          LAC I 10 ADD I 11   ADD I 12  ADD I 13
          RCR      RCR
          DAC I 14
/TEST FOR END OF ROW AND END OF MATRIX
          LAC 12   SAD ROWLIM
          SKP      JMP A      SAD LASTRO          JMP C
/IF ROW HAS BEEN COMPLETED BUT MATRIX HAS NOT, MODIFY
/AUTO INDEX REGISTERS TO START ITERATION THRU NEXT ROW
B,        LAC X0   ADD (100   DAC 10    DAC X0
```

Figure 5.3. Program Listing - Assembly Language

```
          LAC X1    ADD (100    DAC 11     DAC X1
          LAC X2    ADD (100    DAC 12     DAC X2
          LAC X3    ADD (100    DAC 13     DAC X3
          LAC X4    ADD (100    DAC 14     DAC X4
          LAC ROWLIM            ADD (100   DAC ROWLIM
          JMP A
/AFTER ITERATING THRU ENTIRE MATRIX,
/RELOAD AUTO INDEX REGISTERS.
/INCREMENT ITERATION COUNT
C,        LAC X00   DAC 10      DAC X0
          LAC X01   DAC 11      DAC X1
          LAC X02   DAC 12      DAC X2
          LAC X03   DAC 13      DAC X3
          LAC X04   DAC 14      DAC X4
          LAC X05   DAC ROWLIM
/TEST FOR CONVERGENCE TO FINAL VALUE.
/STOP OR CONTINUE ITERATION.
D,        LAC 1517  SAD Z       NOP        DAC Z
          ISZ J     JMP A       HLT
/REGISTER CONTENTS
X0,       100       X00,        100
X1,       177       X01,        177
X2,       201       X02,        201
X3,       300       X03,        300
X4,       200       X04,        200
X05,      237       ROWLIM,     237
Y,        100
LASTRO,   3737      Z,          0
J,        -143
PAUSE 10200
```

Figure 5.3. Continued

PDP 9 Program Timing

Timing for the PDP 9 program to solve Laplace's equation is as follows (times expressed in term of PDP 9 cycle times):

To zero memory locations for the matrix	20,480 cycles
Setting boundary conditions	
Upper	392
Lower	388
Left	452
Right	448
Iteration	
Through one row	769
For 29 rows	22,301
For row #30 (including convergence test)	766
Total time:	
Zeroing memory, setting boundary conditions	22,160 cycles
100 iterations through 30 rows	2,328,860

or, at one microsecond per cycle, approximately 2.33 seconds.

Since there are no I/O interrupts in this program, the calculated time of 2.33 seconds corresponds to the actual running time of the program.

PDP 9 Program - Fortran IV

The program shown in Figure 5.4 is written in Fortran IV to solve Laplace's equation on a 32x32 matrix. The program is written for timing comparison only and has no data output. The program uses the PDP 9 clock for timing and outputs the total number of clock

```
C LAPLACE EQUATION 32X32
C FOR COMPARISON WITH PARALLEL COMPUTER PROGRAM
C
C INITIALIZE CLOCK
C
      DIMENSION U(32,32)
      CALL CLKON
      N=0
      X=0
      DO 1 I=1,32
      DO 1 J=1,32
      U(I,J)=0
1     CONTINUE
C
C SET BOUNDARIES
C
      DO 2 J=1,32
      U(1,J)=30000.
      U(32,J)=30000.
      U(J,1)=30000.
      U(J,32)=30000.
2     CONTINUE
C
C ITERATE
C
      DO 4 K=1,100
      DO 3 I=2,31
      DO 3 J=2,31
      U(I,J)=(.25)*(U(I,J-1)+U(I,J+1)+U(I-1,J)+
     1U(I+1,J))
3     CONTINUE
      IF(X-U(17,17))4,4,4
4     X=U(17,17)
      CALL CLKRD(N)
      CALL CLKOFF
C
C TYPE TOTAL CLOCK PULSES @ 1/60 SEC/PULSE
C
      WRITE (4,5)N
5     FORMAT (31H TOTAL NUMBER OF CLOCK
     1PULSES =,I7////)
      STOP
      END
```

Figure 5.4 Program Listing - Fortran IV

pulses (1/60 second per pulse) for 100 iterations through the matrix.

This program required 201 seconds to run to completion (Fortran IV floating point add time is 418 usec and floating multiply time is 470 usec).

A similar program, written in PDP 9 Fortran II required approximately 10 minutes of computation time.

CHAPTER 6

PARALLEL COMPUTER PROGRAMMING

An important application for which the parallel computer is well suited is the solution of partial differential equations. These equations appear in the description of physical processes such as hydrodynamics, the theory of elasticity, electromagnetism and quantum mechanics. The solution of these equation describe possible physical reactions that have to be fixed by means of boundary conditions.

The problem considered here is the two dimension case of Poisson's equation for f (x,y)=0. Poisson's equation

$$\frac{\partial 2u}{\partial x2} + \frac{\partial 2u}{\partial y2} = f\ (x,y)$$

then becomes Laplace's equation

$$\frac{\partial 2u}{\partial x2} + \frac{\partial 2u}{\partial y2} = 0$$

for the two dimension case.

Laplace's equation can be solved by the method of successive corrections by replacing the partial differential equation with the partial difference equation

$$\phi = \tfrac{1}{4}\ (\phi a, +\phi b+\phi c+\phi d)$$

and iteratively computing the solution to this equation over a matrix of arbitrary dimension.

The application of the parallel computation technique to this type of problem is readily apparent, since the difference equation is the same for each point in the grid. Ideally, therefore, for an n x n grid we would have n^2 processing elements, each solving the difference equation for a specific point with continuous data transfer between a given processing element and its four adjacent neighbors. Since this is not possible, we break the grid into eight equal sections with each processing element working on a section simultaneously. Each processor is therefore in continuous operation and maximum utilization of the parallel computer is effected.

The problem chosen for illustration of the programming of the parallel computer is the solution of Laplace's equation in two dimensions with constant boundary conditions. A grid or matrix of 32x32 points is used to solve the equation. If Δh between any two points of the grid in the x and y directions is assumed to be 1/31, the integration limits are 0-1. With constant boundary conditions, computer solution is of course not necessary; however, the method of solution is applicable irrespective of boundary values.

To solve this problem, the 32x32 matrix is broken into eight sub-matrices each of dimension 4x32. Each processing element is responsible for computation of 120 points in the grid (128 less 8 boundary values) with the exception of the two outer elements. These elements compute 90 points (128 less 38 boundary values). The geometry of the problem is displayed in Figure 6.1.

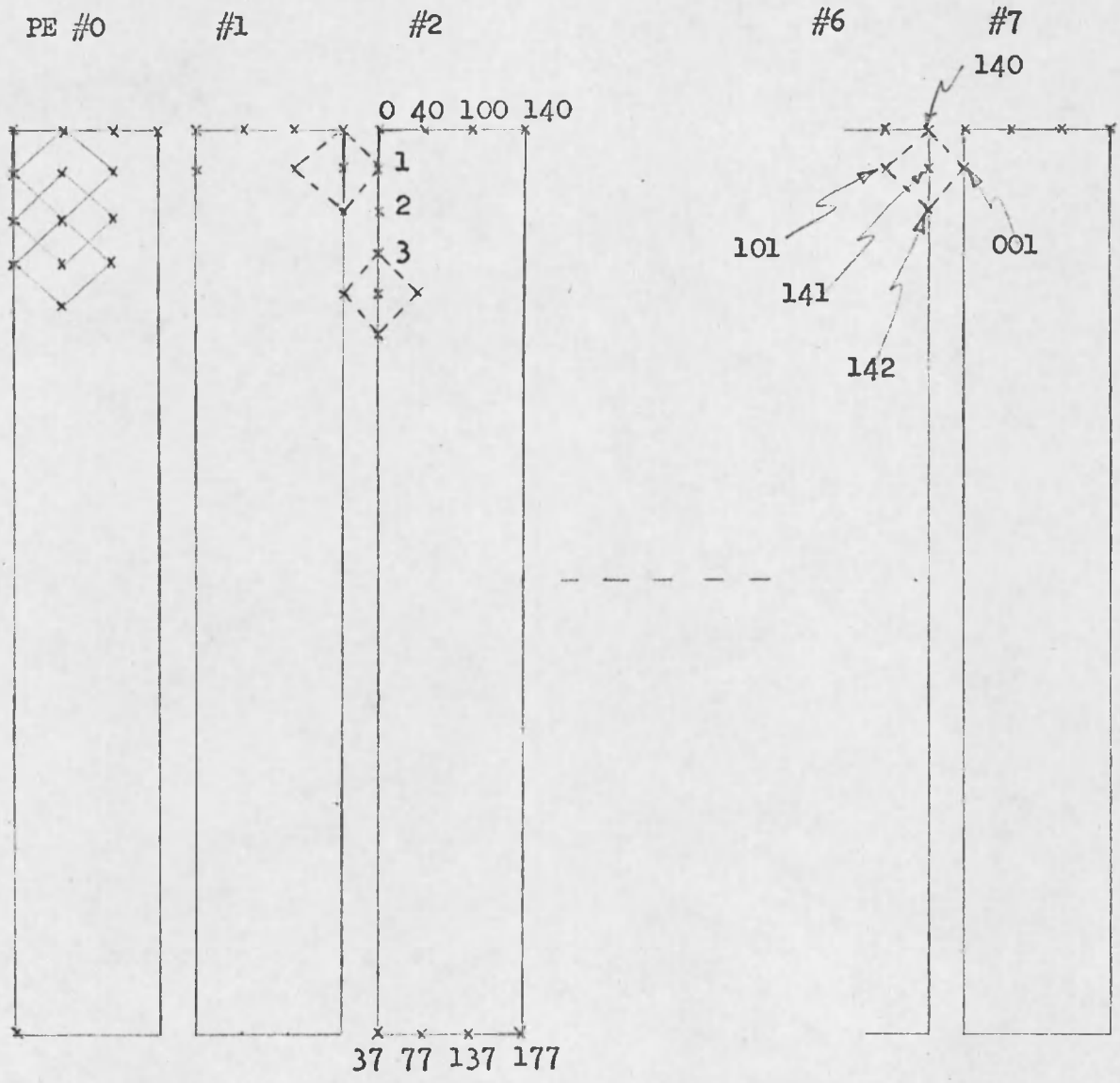


Figure 6.1. Distribution of a 32x32 Matrix in PE Memory for Solution of Laplace's Equation

The numbers refer to the octal address in each PE memory. Each address contains a number which corresponds to the value of that point as computed from the 4 point difference equation. The geometric position of any point over the 32x32 matrix is therefore identified as an octal address in the memory of one of the eight processing elements.

Iteration is programmed by columns. The unknown values of eight points are computed simultaneously and thus 240 points are computed by one iteration through the column dimension of the matrix.

The value of intercommunication between adjacent processing elements is also apparent by examination of the "edges" of the sub-matrices where it can be seen that a PE requires data from the memory of an adjacent PE to compute the value of a point on the "edge".

In order to demonstrate the programming technique the series of instructions listed below will solve the four point difference equation for the matrix point identified by location 141 in each PE memory (Figure 6.1 - PE #6). The adjacent points which determine this value are identified by addresses 101, 140 and 142 in each PE internal memory and by address 001 in the memory of the processing element's right neighbor. Also, processing element #7 does not have to solve for this point since in this PE location 141 is a boundary value which does not change.

Since PE's 0-6 must solve this problem in parallel, the modes of PE's 0-6 must be a subset of the modes designated by the instruction word. The routing portion of the instruction word must designate

internal or right indicating whether the information is to come from the PE's own memory or from its right neighbor.

The equation to be solved is $\phi = \frac{1}{4}(\phi a+\phi b+\phi c+\phi d)$ and the instruction list (assuming all PE's but #7 will respond to modes 0 or 3) is:

```
PCLA   03   I   140      / Clear the accumulator and
                         / add the contents of location
                         / 140 from internal memory - PE's
                         / 0-6

PADD   03   I   140      / Add the contents of location
                         / 101 internal

PADD   03   I   142      / Add the contents of location
                         / 142 internal

PADD   03   R   001      / Add the contents of location
                         / 001 from the right memory

RSHF   03       2        / Right shift the accumulator two
                         / places (divide by 4)

PSTA   03   I   141      / Store the accumulator in
                         / location 141 internal routing
```

It is apparent then, that to solve for every point in the matrix, it is necessary only to modify the address and routing portions of these instructions and to develop an iteration scheme. Since there is no provision for indexing in the parallel computer, instruction modification and control of iteration will be accomplished by the PDP 9.

The program to solve this problem is written for the PDP 9 since the PDP 9 exercises control over the parallel computer by transfering data and instructions and by indexing parallel computer instructions.

The program requires that there be two lists stored in PDP 9 memory. The first of these, DATA, is a list of constants and PE modes interspersed with PE instructions. The PDP 9 will IOT a mode or a constant to one of the PE accumulators or to the broadcast register. The PDP 9 will then IOT appropriate instructions to the control unit which will cause these modes or constants to be stored by each PE. The sequence of constant, instruction, instruction in the list DATA is repeated for each different constant and mode until all constants and modes required for the solution of the problem have been stored by all processing elements.

The list DATA also contains boundary conditions, with each boundary being followed by a sequence of instructions. A boundary condition is I/O transferred to the broadcast register. This boundary condition is to be stored in PE memory. Two instructions required for this purpose are transferred to the control sequencer. The first instruction loads the PE accumulator from the broadcast register and the second stores the contents of the PE accumulator. This sequence is repeated until all boundary conditions have been transferred.

For the special case of the left and right boundary conditions which are to be stored only in PE's #0 and 7, the PDP 9 transfers instructions to change the modes of PE's #0 and 7. The PDP 9 then transfers instructions which cause PE's #0 and 7 to store the left and right boundary conditions. A subroutine CHANGE modifies and IO

transfers these instructions to store the left and right boundary conditions in sequential locations in PE's #0 and 7.

The second list stored in the PDP 9 memory is a list of processing element instructions which are to be sequentially transferred to the parallel computer. Instructions in LIST are those which are to be executed by the parallel computer to solve the problem. A PDP 9 subroutine, MODIFY, provides address modification and IOT for any sequence of instructions which is to be repeated to effect iteration. LIST also contains PE instructions necessary for PE mode changes when required.

PDP 9 mnemonics are used in both lists to identify the end of a sequence of instructions which is to be repeated by the processing elements with only address modification. These mnemonics are XCT and NOP. When the PDP 9 encounters these mnemonics, a jump to a subroutine occurs. The subroutine modifies an instruction and then I/O transfers it to the control unit. All instructions in the sequence are modified and transferred in this manner. This process is repeated for the required number of modifications to each instruction at which time control is returned to the main program which continues the transfer of other instructions.

The PDP 9 program to provide data transfer consists of the following parts:

1. A program START which transfers constants, modes, boundary conditions and instructions from the list DATA; associated with this

program is the subroutine CHANGE used for address modification.

2. A program PE INST and associated subroutine MODIFY which transfers and/or modifies an instruction sequence from LIST for execution of the main program by the parallel computer.

3. A program CONVERGE which tests for convergence of the solution after a complete iteration. If the solution has converged, control is transferred to the program OUT DATA which outputs the final results; if the solution has not converged, the program restores the autoindex registers and instructions which have been modified and returns control to PE INST to continue the transfer of instructions.

The problem is charted in the flow diagrams shown in Figures 6.2, 6.3 and 6.4. The first of these diagrams is a basic chart only; the remaining flow diagrams describe the problem in detail. The program listing is detailed in Appendix E.

Parallel Computer Program
Computation Time - Laplace Equation

Timing for the parallel computer program is as follows (all times expressed in terms of PDP 9 cycle time):

To transfer modes and store in PE internal memory	30 cycles
To transfer constants & store in PE internal memory	54
To transfer upperboundary	64
To transfer lowerboundary	69
Mode changes PE's 0 & 7	14
Transfer & store left boundary	352

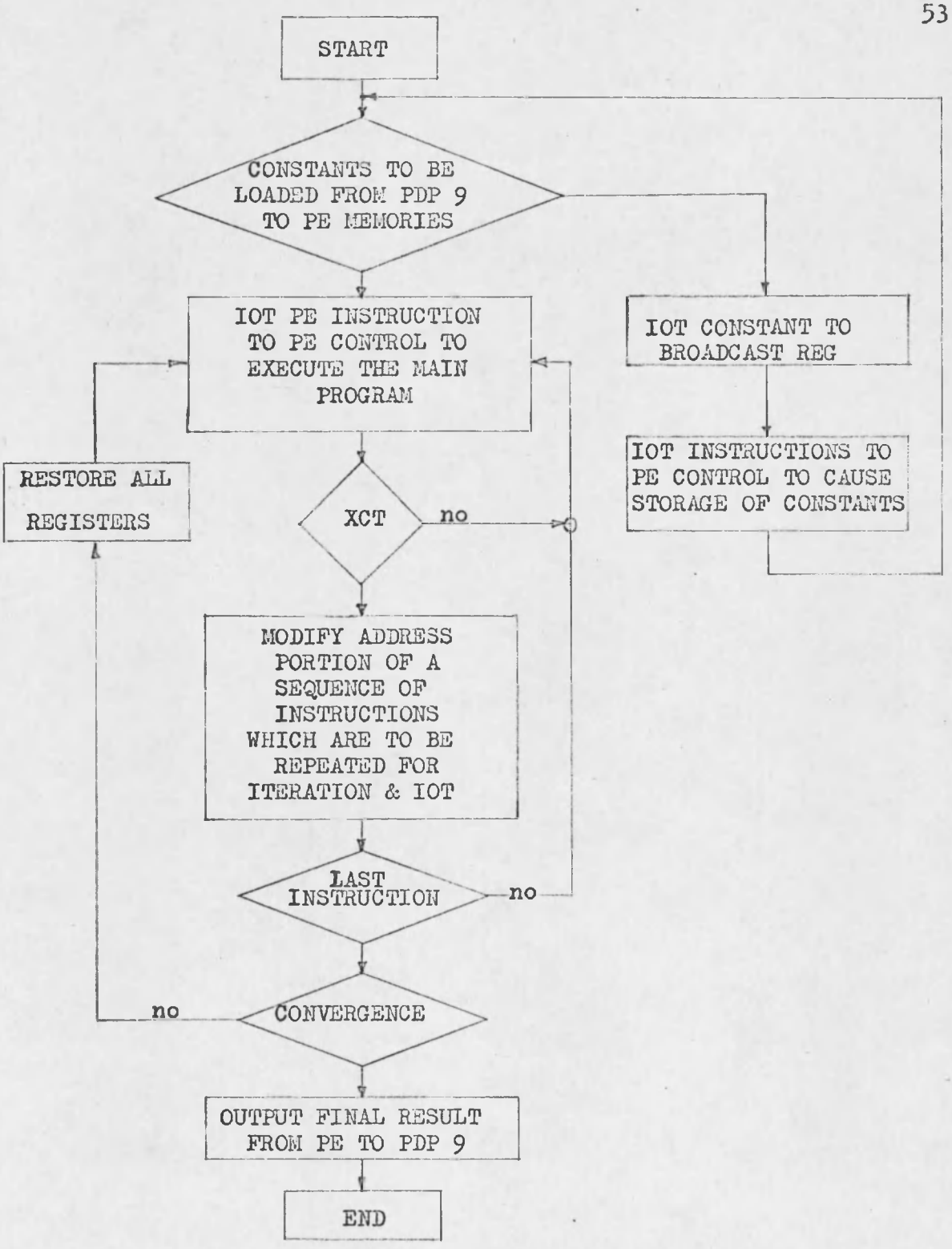


Figure 6.2. Flow Chart - Parallel Computer Program

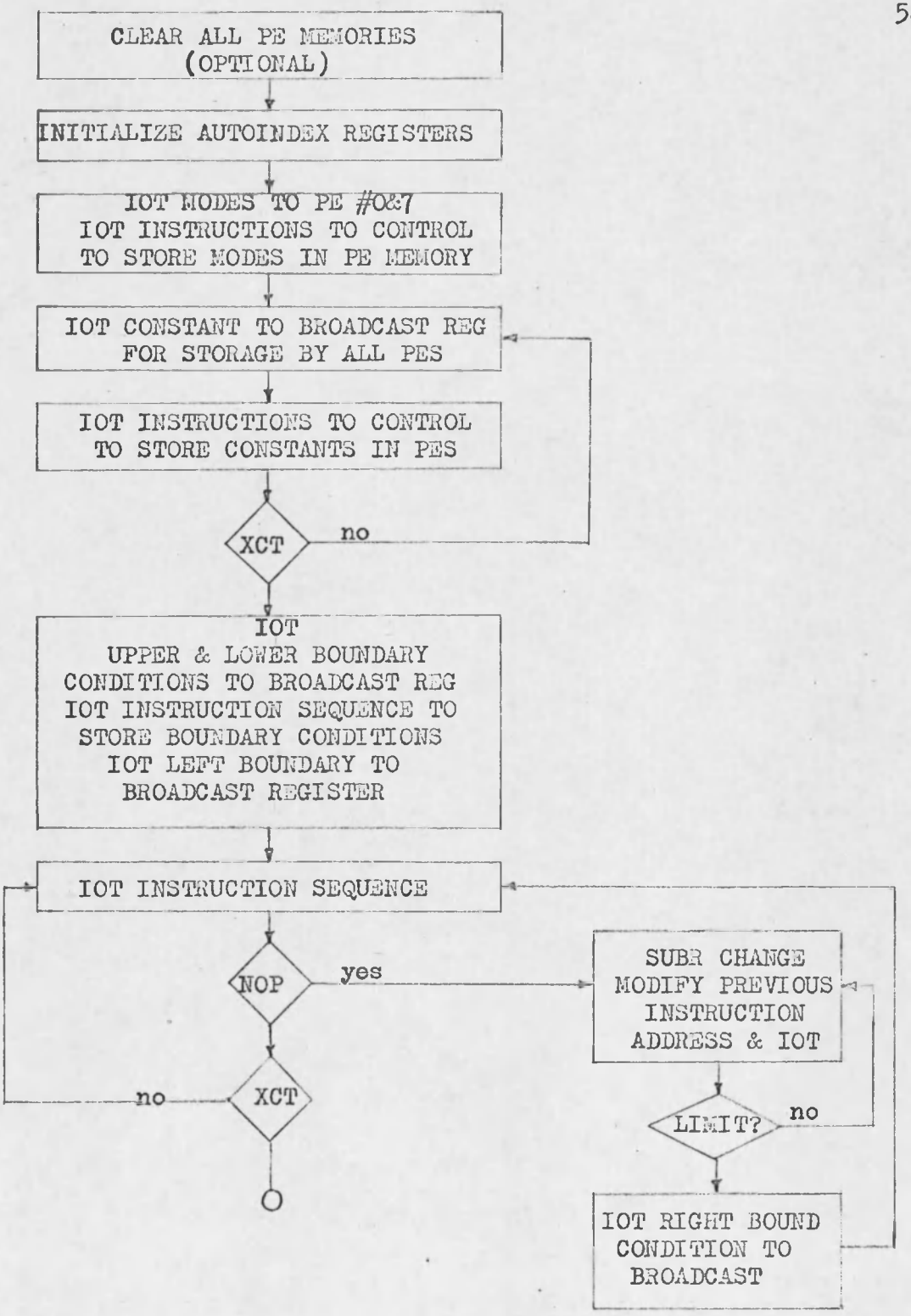


Figure 6.3. Flow Chart - Parallel Computer Program

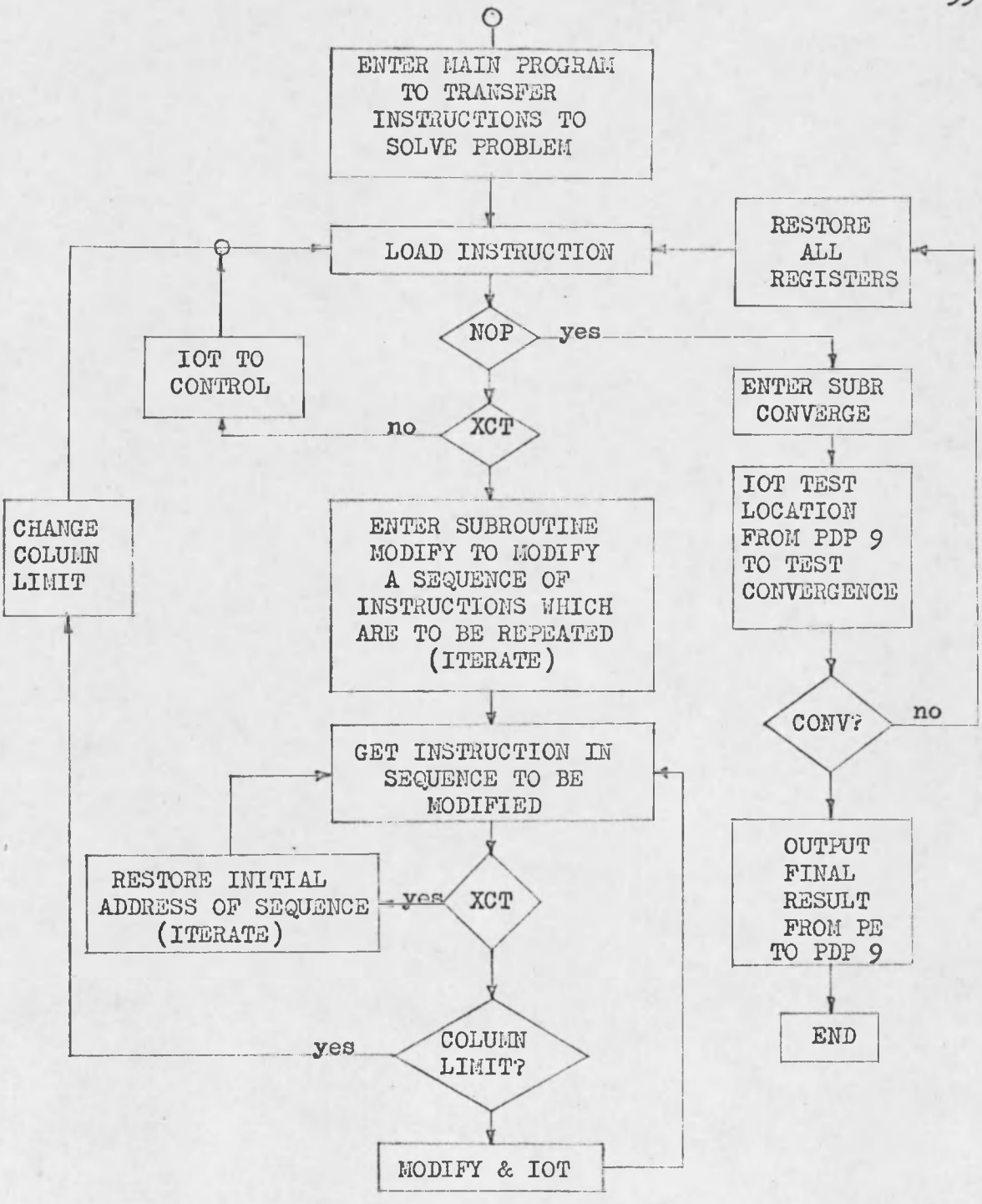


Figure 6.4. Flow Chart - Parallel Computer Program

Program modification	13
Transfer & store right boundary	359
	955 cycles

Total time to execute program START & subroutine CHANGE to transfer all modes, constants and boundary conditions as well as instructions from the list DATA is 955 cycles.

To output six instructions to solve the difference equation for one point (PE's 1-7)	80 cycles
To address modify and IOT to solve for the next 29 points in a column = (106x29)	3078
Mode change PE's 0 and 7	12
Iteration through the next column(all PE's)=(80+3078)	3158
Iteration through the next column (all PE's)	3158
Mode change PE #7	12
Iteration through last column (PE's 0-6)	3156
End of one iteration through matrix	12666 cycles
Convergence test	15
Register restoration	11
	12692 cycles

On last iteration only	12666 cycles
Convergence test	14
Data return to PDP 9	14
	12694 cycles

Total Timing for 100 iterations

12692x99	1,256,508
Iteration #100	12,694
DATA transfer	955
Clearing PE memory (if used)	2,321
	1,272,478 cycles

Therefore, assuming 100 iterations are required for convergence, the total time required to solve the problem is 1.273 seconds (one cycle time = one microsecond) with the final result being stored in a predesignated location in PDP 9. This time is to be compared with the PDP 9 assembly language program which required 2.33 seconds and the Fortran IV program which required 201 seconds.

The time required to transfer six instructions which will solve the difference equation for one point is 80 cycles. In order to solve for the next 29 points in a column, this sequence of instructions is address modified and each instruction is I/O transferred after it has been modified. This process requires 3078 cycles for the 29 points; of this total, only 116 cycles are I/O transfer time with the remaining 2962 cycles being required by the PDP 9 address modification subroutine. PDP 9 computation time not associated with the actual computation of the problem thus accounts for some 90% of the total computing time.

If the parallel computer control unit had six storage registers and an indexing facility, the total computation time could be significantly reduced. Assuming such a facility were available and, allowing two cycles for indexing by the parallel computer (during which time another instruction is being executed by the processing elements), the total computation time for 29 points would be 12x29 = 348 cycles. Adding 80 cycles for intitial I/O transfer and allowing 72 cycles for column limit tests, etc, the total computation time to solve for 30 points in one column is 500 cycles.

The total computation time for 100 iterations through four columns would then be 200,000 cycles. Allowing a conservative 30,000 cycles for the remainder of the PDP 9 program, the maximum computation time for this problem would be 0.23 seconds compared with the 1.273 seconds required by the program as now written. Compared with the 2.33 seconds required by the PDP 9 program, a significant improvement of 10 to 1 computation time could be realized.

CHAPTER 7

CONCLUSION

Rapid advances in technology have resulted in the design of faster, larger computers and in the development of more efficient data processing. However, practical limitations in hardware design have nearly been reached and future designs will require different design techniques to effectively increase computation speed.

The concept of parallel computation affords the possibility of great increase in computation speed at moderate cost.

The cost of implementing the design for eight processing elements for a parallel computer as presented in this thesis is approximately $20,000. The addition of a control sequencer, interface units and elements of the processing elements not included in this thesis will raise the total cost to implement the parallel computer to the order of $30,000.

The programming of a problem with parallel characteristics was investigated using the PDP 9. A computation error introduced by the use of fixed point programming was discovered and rectified by the use of a different computation technique so that such problems can be solved correctly to the accuracy limits of the PDP 9.

A special problem, the solution of Laplace's equation over 2 32x32 matrix was solved in PDP 9 Fortran II and IV and PDP 9

assembly language; this special problem was also programmed for the parallel computer. The computation time for the Fortran II program was 10 minutes; in Fortran IV, the computation time was 201 seconds and the assembly language program required 2.33 seconds. The calculated computation time for the parallel computer program was 1.273 seconds, an improvement 1.5 to 1 over the fastest PDP 9 computation time.

Although the speed advantage of 1.5-1 would not justify the cost of hardware, it was determined that the addition of 6-10 memory registers for the control unit and the addition of a facility to index parallel computer instructions could reduce the computation time for the same problem from 1.273 seconds to approximately 0.23 seconds.

The addition of these capabilities could be accomplished at very small cost relative to the total cost of $30,000 and would result in a significant improvement in computation speed of 10 to 1 compared with the fastest PDP 9 program.

Significant improvements in computation speeds afforded by the parallel computer would greatly assist agencies such as AEC who require hundreds of hours of computation time solving partial differential equations using FORTRAN and similar methods for programming these problems.

APPENDIX A

SYNCHRONOUS ADDER - I'TH STAGE

Figure A.1 shows the I'th stage for the synchronous adder. This circuit is a standard NAND realization which yields the sum and carry signals Si+1 and Ci+1 given the inputs Ai, Bi, Ci and their complements. The cost of the adder portion of the processing element was based on the use of this circuit.

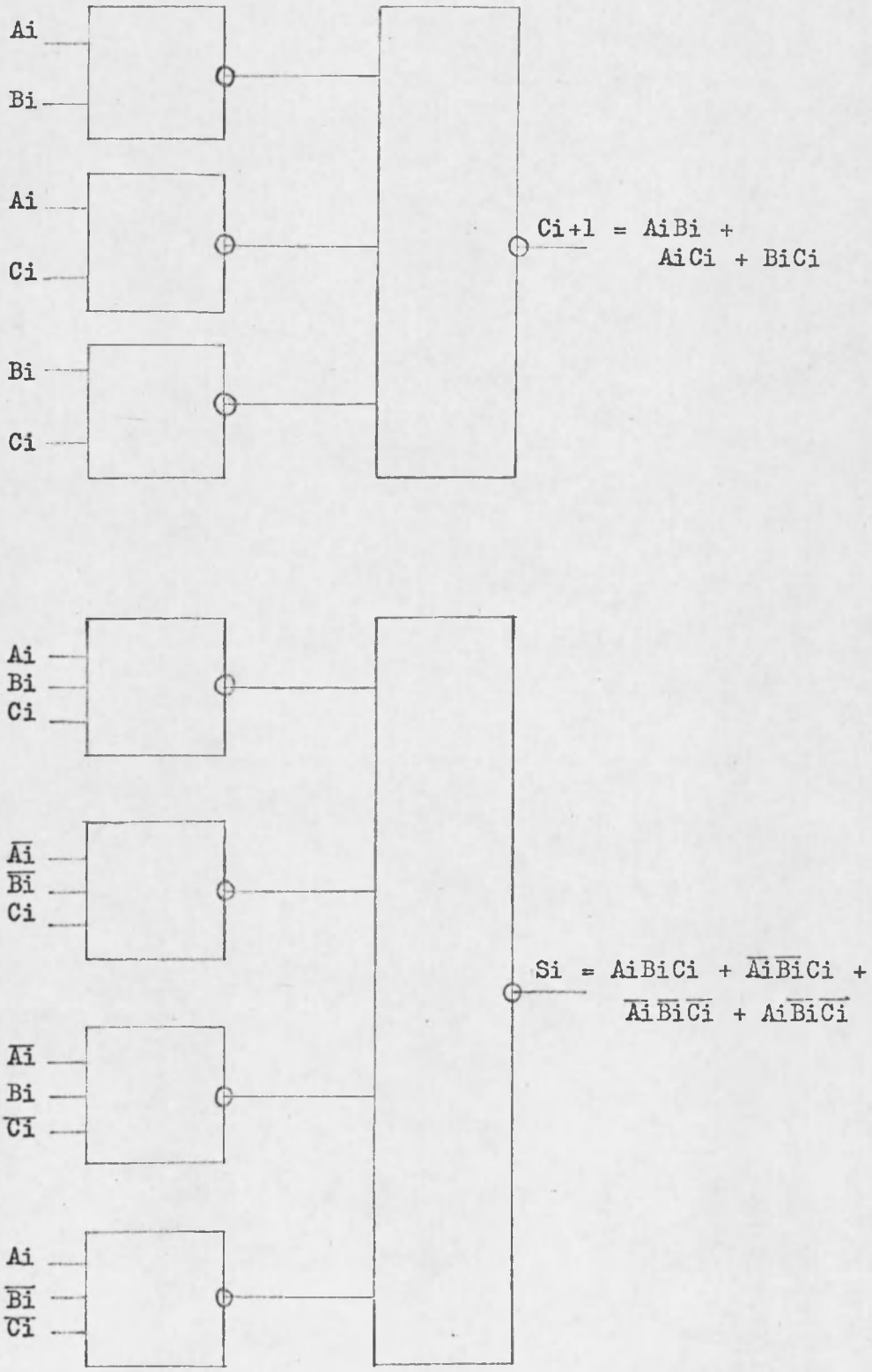


Figure A.1. Synchronous Adder - I'th Stage

APPENDIX B

PDP 9 COMPUTER OUTPUTS - LAPLACE/POISSON EQUATIONS

Figures B.1 - B.6 are solutions to Laplace's and Poisson's equations using two different matrix dimensions. Boundary conditions were identical and were set to the constant value 30,000. The other initial values of the matrix were zero. The difference equation solved was

$$\phi = \frac{1}{4} (A+B+C+D) + \frac{1}{4} \Delta h^2 f (x,y)$$

where the factor $\frac{1}{4} \Delta h^2 f (x,y)$ is designated the Poisson constant. Solutions were obtained for Poisson constants set to the values 0 (yielding Laplace's equation), 1 and 10.

Figures B.1 - B.3 are the computed outputs for the equations over an 8x8 matrix. The entire matrix was printed every 20 iterations. The last printout is the final solution after convergence. Examination of Figure B.1 shows an error in the final result. The results presented in Figures B.2 and B.3 are also in error.

Figures B.4 - B.6 are the computed solutions over a 63x63 matrix for the same Poisson constants. The program printed the computed value of the center of the matrix every 100 iterations. Computation time was 10 seconds per 100 iterations and the final results are in error by approximately 2%.

```
0          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT
                  OUTPUT MATRIX EVERY 20 ITERATIONS

030000   030000   030000   030000   030000   030000   030000   030000
030000   029747   029588   029537   029582   029697   029848   030000
030000   029588   029331   029247   029321   029508   029753   030000
030000   029537   029247   029153   029236   029447   029723   030000
030000   029582   029321   029236   029311   029501   029750   030000
030000   029697   029508   029447   029501   029639   029819   030000
030000   029848   029753   029723   029750   029819   029909   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   029995   029991   029990   029991   029993   029996   030000
030000   029991   029986   029984   029985   029989   029994   030000
030000   029990   029984   029982   029983   029987   029993   030000
030000   029991   029985   029983   029984   029988   029993   030000
030000   029993   029989   029987   029988   029991   029995   030000
030000   029996   029994   029993   029993   029995   029997   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   029998   029997   029996   029996   029997   029998   030000
030000   029997   029995   029994   029994   029995   029997   030000
030000   029996   029994   029993   029993   029994   029996   030000
030000   029996   029994   029993   029993   029994   029996   030000
030000   029997   029995   029994   029994   029995   029997   030000
030000   029998   029997   029996   029996   029997   029998   030000
030000   030000   030000   030000   030000   030000   030000   030000
```

Figure B.1. Laplace/Poisson Equation Solution

```
1          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT
                  OUTPUT MATRIX EVERY 20 ITERATIONS

030000   030000   030000   030000   030000   030000   030000   030000
030000   029751   029594   029544   029589   029703   029852   030000
030000   029594   029340   029258   029332   029518   029760   030000
030000   029544   029258   029166   029249   029458   029730   030000
030000   029589   029332   029249   029324   029513   029757   030000
030000   029703   029518   029458   029513   029649   029825   030000
030000   029852   029760   029730   029757   029825   029913   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   029999   029998   029998   029998   029999   030000   030000
030000   029998   029996   029996   029997   029999   030000   030000
030000   029998   029996   029996   029997   029999   030000   030000
030000   029998   029997   029997   029998   030000   030001   030000
030000   029999   029999   029999   030000   030001   030001   030000
030000   030000   030000   030000   030001   030001   030001   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030002   030003   030003   030003   030003   030002   030000
030000   030003   030005   030005   030005   030005   030003   030000
030000   030003   030005   030006   030006   030005   030003   030000
030000   030003   030005   030006   030006   030005   030003   030000
030000   030003   030005   030005   030005   030005   030003   030000

030000   030002   030003   030003   030003   030003   030002   030000
030000   030000   030000   030000   030000   030000   030000   030000
```

Figure B.2. Laplace/Poisson Equation Solution

```
10          /POISSON CONSTANT
30000       /UPPER BOUNDARY
30000       /LOWER
30000       /LEFT
30000       /RIGHT
                    OUTPUT MATRIX EVERY 20 ITERATIONS

030000   030000   030000   030000   030000   030000   030000   030000
030000   029788   029651   029609   029654   029760   029890   030000
030000   029651   029429   029362   029436   029607   029817   030000
030000   029609   029362   029288   029371   029563   029796   030000
030000   029654   029436   029371   029446   029617   029823   030000
030000   029760   029607   029563   029617   029739   029883   030000
030000   029890   029817   029796   029823   029883   029951   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030037   030056   030064   030065   030057   030038   030000
030000   030056   030087   030102   030103   030090   030058   030000
030000   030064   030102   030120   030121   030105   030067   030000
030000   030065   030103   030121   030122   030106   030068   030000
030000   030057   030090   030105   030106   030092   030059   030000
030000   030038   030058   030067   030068   030059   030039   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030040   030061   030070   030070   030061   030040   030000
030000   030061   030096   030111   030111   030096   030061   030000
030000   030070   030111   030130   030130   030111   030070   030000
030000   030070   030111   030130   030130   030111   030070   030000
030000   030061   030096   030111   030111   030096   030061   030000
030000   030040   030061   030070   030070   030061   030040   030000
030000   030000   030000   030000   030000   030000   030000   030000
```

Figure B.3. Laplace/Poisson Equation Solution

```
0          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT


OUTPUT EVERY 100 ITERATIONS


000000
000265
003272
008029
012558
016313
019287
021613
023415
024812
025891
026726
027374
027878
028265
028565
028780
028980
029088
029188
029288
029382
029385


TOTAL NUMBER OF ITERATIONS = 2200
```

Figure B.4. Laplace/Poisson Equation Solution

```
1          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT

OUTPUT EVERY 100 ITERATIONS

000000
000457
003639
008557
013221
017082
020137
022523
024376
025813
026920
027784
028446
028968
029366
029666
029892
030092
030214
030314
030414
030495

TOTAL NUMBER OF ITERATIONS = 2100
```

Figure B.5. Laplace/Poisson Equation Solution

```
10          /POISSON CONSTANT
30000       /UPPER BOUNDARY
30000       /LOWER
30000       /LEFT
30000       /RIGHT

OUTPUT EVERY 100 ITERATIONS

000000
002252
007142
013514
019343
024117
027884
030820
033094
034857
036217
037274
038089
038726
039209
039585
039885
040093
040287
040397
040497
040597
040671
040702
040704

TOTAL NUMBER OF ITERATIONS = 2400
```

Figure B.6. Laplace/Poisson Equation Solution

APPENDIX C

PDP 9 COMPUTER OUTPUTS - LAPLACE/POISSON EQUATIONS

The computer outputs shown in Figures C.1 - C.6 are solutions to the same problems described in Appendix B with the exception that these programs were modified to eliminate the errors in the final result. The programs were modified to add the value 1 to the computed solutions on every alternate iteration. This yielded results accurate to ± 1 in the least significant digit.

Figures C.1 - C.3 are solutions to the equations on an 8x8 matrix and Figures C.4 - C.6 are solutions over a 63x63 matrix.

```
0          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT
                  OUTPUT MATRIX EVERY 20 ITERATIONS

030000   030000   030000   030000   030000   030000   030000   030000
030000   029748   029591   029540   029586   029700   029851   030000
030000   029590   029336   029253   029327   029513   029758   030000
030000   029539   029253   029160   029243   029453   029728   030000
030000   029584   029326   029242   029318   029507   029755   030000
030000   029699   029513   029452   029508   029644   029823   030000
030000   029849   029757   029726   029754   029822   029912   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   029996   029995   029994   029995   029996   029999   030000
030000   029994   029992   029990   029992   029994   029998   030000
030000   029993   029991   029989   029991   029993   029998   030000
030000   029994   029992   029990   029992   029994   029998   030000
030000   029996   029995   029993   029995   029996   029999   030000
030000   029998   029998   029997   029998   029998   030000   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   029999   030000   029999   030000   029999   030000   030000
030000   029999   030000   029999   030000   029999   030000   030000
030000   029999   030000   029999   030000   029999   030000   030000
030000   029999   030000   029999   030000   029999   030000   030000
030000   029999   030000   029999   030000   029999   030000   030000
030000   029999   030000   029999   030000   029999   030000   030000
030000   030000   030000   030000   030000   030000   030000   030000
```

Figure C.1. Laplace/Poisson Equation Solution

```
1           /POISSON CONSTANT
30000       /UPPER BOUNDARY
30000       /LOWER
30000       /LEFT
30000       /RIGHT
                    OUTPUT MATRIX EVERY 20 ITERATIONS

030000   030000   030000   030000   030000   030000   030000   030000
030000   029752   029598   029547   029593   029706   029855   030000
030000   029597   029346   029264   029339   029523   029764   030000
030000   029546   029265   029173   029257   029464   029735   030000
030000   029592   029338   029256   029332   029518   029762   030000
030000   029706   029524   029464   029519   029653   029829   030000
030000   029854   029764   029734   029761   029828   029916   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030000   030001   030001   030002   030002   030003   030000
030000   030000   030001   030001   030003   030003   030004   030000
030000   030000   030002   030002   030004   030004   030005   030000
030000   030001   030003   030003   030005   030005   030005   030000
030000   030002   030004   030004   030006   030005   030005   030000
030000   030002   030004   030004   030005   030004   030004   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030003   030006   030006   030007   030006   030005   030000
030000   030005   030010   030010   030011   030009   030007   030000
030000   030006   030011   030012   030013   030011   030008   030000
030000   030006   030011   030012   030013   030011   030008   030000
030000   030005   030010   030010   030011   030009   030007   030000
030000   030003   030006   030006   030007   030006   030005   030000
030000   030000   030000   030000   030000   030000   030000   030000
```

Figure C.2. Laplace/Poisson Equation Solution

```
10          /POISSON CONSTANT
30000       /UPPER BOUNDARY
30000       /LOWER
30000       /LEFT
30000       /RIGHT
                    OUTPUT MATRIX EVERY 20 ITERATIONS

030000   030000   030000   030000   030000   030000   030000   030000
030000   029789   029654   029612   029658   029763   029892   030000
030000   029653   029434   029367   029442   029612   029821   030000
030000   029611   029368   029294   029378   029568   029800   030000
030000   029657   029442   029377   029453   029622   029827   030000
030000   029763   029613   029568   029623   029743   029886   030000
030000   029891   029820   029799   029827   029885   029953   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030038   030059   030067   030069   030060   030041   030000
030000   030058   030092   030106   030109   030094   030062   030000
030000   030067   030107   030125   030128   030110   030071   030000
030000   030067   030108   030126   030129   030111   030072   030000
030000   030059   030095   030110   030112   030096   030063   030000
030000   030040   030062   030070   030071   030062   030042   030000
030000   030000   030000   030000   030000   030000   030000   030000

030000   030000   030000   030000   030000   030000   030000   030000
030000   030041   030064   030073   030074   030064   030043   030000
030000   030063   030101   030116   030117   030100   030065   030000
030000   030073   030117   030136   030137   030116   030074   030000
030000   030073   030117   030136   030137   030116   030074   030000
030000   030063   030101   030116   030117   030100   030065   030000
030000   030041   030064   030073   030074   030064   030043   030000
030000   030000   030000   030000   030000   030000   030000   030000
```

Figure C.3. Laplace/Poisson Equation Solution

```
Ø          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT

OUTPUT EVERY 100 ITERATIONS

000000
000368
003478
008315
012909
016711
019724
022077
023906
025325
026426
027274
027926
028434
028825
029125
029345
029545
029661
029761
029861
029961
030000

TOTAL NUMBER OF ITERATIONS = 2200
```

Figure C.4. Laplace/Poisson Equation Solution

```
1          /POISSON CONSTANT
30000      /UPPER BOUNDARY
30000      /LOWER
30000      /LEFT
30000      /RIGHT


OUTPUT EVERY 100 ITERATIONS


000000
000555
003833
008831
013561
017471
020567
022984
024861
026312
027436
028307
028974
029500
029904
030205
030432
030632
030736
030836
030936
031036
031060


TOTAL NUMBER OF ITERATIONS = 2100
```

Figure C.5. Laplace/Poisson Equation Solution

```
10          /POISSON CONSTANT
30000       /UPPER BOUNDARY
30000       /LOWER
30000       /LEFT
30000       /RIGHT

OUTPUT EVERY 100 ITERATIONS

000000
002351
007335
013787
019683
024508
028314
031279
033579
035359
036735
037802
038625
039266
039756
040137
040437
040648
040843
040956
041056
041156
041229
041265
041267

TOTAL NUMBER OF ITERATIONS = 2500
```

Figure C.6. Laplace/Poisson Equation Solution

APPENDIX D

POISSON'S EQUATION - SPECIAL PROBLEM

Figure D.1 is a program listing written in Fortran IV to solve Poisson's equation $\nabla^2\phi = -2$ in two dimensions. Boundary conditions are zero and integration limits are 0-1. The problem is programmed over a matrix dimension of 25x25 ($\Delta h = 0.0416$).

The purpose of programming this problem was to demonstrate the accuracy of the successive relaxation method for solving partial differential equations. The computed solution for the potential at the center of the matrix is given in Figure D.2. The known solution, accurate to 5 decimals, is given for comparison purposes along with the total computation time.

```
C POISSON'S EQUATION
C
      DIMENSION U(25,25)
      WRITE(4,1)
1     FORMAT (33H TWO DIMENSIONAL POISSON EQUATION/
     125H BOUNDARY CONDITIONS ZERO/23H INTEGRATION
     2LIMITS Ø-1/
     340H THE CORRECT ANSWER FOR THE POTENTIAL AT/
     436H THE CENTRE OF THE SQUARE IS Ø.14734/////
     540H OUTPUT EVERY 1ØØ ITERATIONS THRU MATRIX///)
C
C
      CALL CLKON
      DO 2 I=1,25
      DO 2 J=1,25
2     U(I,J)=Ø
      N=Ø
      Y=2.*.Ø4166666*.Ø4166666
3     X=U(13,13)
      DO 8 K=1,1ØØ
      DO 4 I=2,24
      DO 4 J=2,24
4     U(I,J)=(.25)*(U(I,J-1)+U(I,J+1)+U(I+1,J)+
     1U(I-1,J)+Y)
8     CONTINUE
      WRITE(4,5)U(13,13)
5     FORMAT (F15.5)
      IF(X-U(13,13))3,6,3
6     CALL CLKRD(N)
      CALL CLKOFF
C
C TYPE TOTAL CLOCK PULSES AT 1/6Ø SEC/PULSE
C
      WRITE(4,7)N
7     FORMAT(///21H TOTAL CLOCK PULSES = I7///)
      STOP
      END
```

Figure D.1. Poisson's Equation - Program Listing

```
TWO DIMENSIONAL POISSON EQUATION, ∇²U=-2
BOUNDARY CONDITIONS ZERO
INTEGRATION LIMITS Ø-1
THE CORRECT ANSWER FOR THE POTENTIAL AT
THE CENTER OF THE SQUARE IS Ø.14734
OUTPUT EVERY 1ØØ ITERATIONS THROUGH MATRIX
          Ø.11754
          Ø.14183
          Ø.14619
          Ø.14697
          Ø.14711
          Ø.14714
          Ø.14714
          Ø.14714
          Ø.14714
          Ø.14714
TOTAL CLOCK PULSES = 82169
TOTAL COMPUTATION TIME = 22.8 MINUTES
RESULT CONVERGED TO 5 DECIMALS AFTER 13.7 MINUTES
STOP   ØØØØØØ
```

Figure D.2. Poisson's Equation - Program Solution

APPENDIX E

PARALLEL COMPUTER PROGRAM

The parallel computer program was written for the PDP 9 computer to solve Laplace's equation over a 32x32 matrix. The program was written to develop a method for programming a problem with parallel characteristics and to provide a timing comparison with the same problem as written for the PDP 9 computer.

The program assumes only that the instruction list required by the parallel computer has been stored in PDP 9 memory by another program.

```
/"DATA" IS A LIST OF CONSTANTS & BOUNDARY CONDITIONS INTERSPERSED
/WITH PE INSTRUCTIONS. "LIST" IS A LIST OF PE INSTRUCTIONS
/WHICH EXECUTE THE MAIN PROGRAM.

10/             LIST - 1                /PDP 9 AUTOINDEX REGISTERS
11/             DATA - 1                /THESE REGISTERS, WHEN
12/             LIST - 1                /INDIRECTLY ADDRESSED, CAUSE
                                        /THEIR CONTENTS TO BE
                                        /INCREMENTED BY ONE AND THE
                                        /INCREMENTED NUMBER IS ADDRESSED.

/TRANSFER ALL CONSTANTS REQUIRED BY THE
/PROCESSING ELEMENTS DURING THE COMPUTATION TO
/THE BROADCAST REGISTER FOR STORAGE IN THE
/PE MEMORIES.  THESE CONSTANT ARE LOCATED
/IN PDP 9 MEMORY STARTING AT LOCATION DATA
/EACH CONSTANT IS FOLLOWED BY THE
/INSTRUCTIONS WHICH ARE TRANSFERED TO THE
/CONTROL UNIT AND CAUSE THE CONSTANT
/TO BE STORED.  THE LIST IS TERMINATED
/BY A HLT.

/ALL MODES ARE INITIALLIZED FIRST.  FOR THIS
/PROGRAM, ALL PE'S ARE TO RESPOND TO MODE
/0; PE#0 IS TO RESPOND TO MODE 1
/AND PE#7 IS TO RESPOND TO MODE 2
/FOR SETTING BOUNDARY CONDITIONS.

START,          LAC I 11                /LOAD THE CONTENTS OF LOCATION
                SAD (HLT                /DATA, IF THE CONTENTS OF DATA
                JMP PE INST             /ARE HLT, EXECUTE MAIN PROGRAM

                IOT #0                  /IOT MODE TO PE #0

                LAC I 11                /LOAD DATA +1
                IOT C                   /IOT INSTRUCTION TO CONTROL
                                        /UNIT TO STORE MODE FOR PE #0

                LAC I 11                /LOAD DATA +2
                IOT #7                  /IOT MODE TO PE #7
                LAC I 11                /LOAD DATA +3
                IOT C                   /IOT 2 INSTRUCTIONS TO CONTROL
                                        /UNIT TO STORE MODE FOR PE #7

/MODES CAN BE SET AS FOLLOWS WHEN REQUIRED
/               PE #0   -   MODE 1
/               PE #1-6 -   MODE 0
/               PE #7   -   MODE 2
```

```
GO,         LAC I 11          /LOAD DATA +4 WHICH CONTAINS
                              /THE FIRST CONSTANT FOR USE
                              /BY ALL PE'S

            SAD (XCT
            JMP GO1           /LIST OF CONSTANTS TERMINATED
                              /BY XCT

GO+3,       IOT B             /IOT CONSTANT TO BROADCAST
                              /REGISTER

            LAC I 11          /THE NEXT TWO WORDS IN DATA
            IOT C             /LIST ARE INSTRUCTIONS WHICH
            LAC I 11          /WILL STORE BROADCAST REGISTER
            IOT C             /IN PE MEMORIES

            JMP GO            /CONTINUE TRANSFER OF CONSTANTS
                              /TO PE'S UNTIL XCT IS
                              /ENCOUNTERED

/TRANSFER BOUNDARY CONDITIONS

GO1,        LAC I 11          /LOAD UPPER OR LOWER BOUNDARY
            IOT B             /CONDITION & TRANSFER TO
                              /BROADCAST

            LAC (-5           /THE NEXT FIVE WORDS IN DATA
            DAC TEMP          /ARE INSTRUCTIONS TO STORE THE
GO1+4,      ISZ TEMP          /BOUNDARY CONDITION
GO1+5,      JMP GO2
            LAC I 11
            IOT C
            JMP GO1+4

GO2,        LAC (JMP GO3      /TRANSFER LOWER BOUNDARY
            DAC GO1+5         /FOLLOWED BY FIVE INSTRUCTIONS
            JMP GO1

GO3,        LAC I 11          /CHANGE MODE OF PE#0 TO 1 AND
            IOT C             /PE #7 TO 2
            LAC I 11
            IOT C
GO3+4       LAC I 11          /TRANSFER LEFT OR RIGHT BOUNDARY
            SAD (HLT          /TO BROADCAST REGISTER
            JMP PE INST
            IOT B
```

```
G03+10,     LAC I 11          /TRANSFER INSTRUCTIONS TO
            SAD (NOP          /CONTROL WHICH WILL STORE THE
            JMP CHANGE        /LEFT OR RIGHT BOUNDARY.
            IOT C             /CHANGE IS A SUBROUTINE WHICH
            JMP G03+10        /WILL MODIFY & IOT INSTRUCTIONS
                              /TO STORE LEFT OR RIGHT
                              /BOUNDARIES.  DATA IS TERMINATED
                              /BY A HLT.  THE SUBROUTINE
                              /CHANGE RETURNS CONTROL TO G03+4

/THIS SUBROUTINE MODIFIES INSTRUCTIONS
/DATA +22 AND DATA +37 FOR STORAGE OF
/LEFT BOUNDARY CONDITIONS IN PE #0 IN
/ADDRESSES  36 AND RIGHT BOUNDARY CONDITIONS
/IN PE #7 ADDRESSES 141-176.  THE SUBROUTINE
/IS ENTERED WHEN THE WORD NOP IS ENCOUNTERED
/IN THE DATA LIST.

CHANGE,     LAC (-35          /2'S COMPLEMENT OF NUMBER OF
            DAC TEMP          /POINTS OCCUPIED BY LEFT OR
CHANGE +2,  ISZ TEMP          /RIGHT BOUNDARIES
            JMP D

CHANGE +4,  ISZ DATA +33
            LAC DATA +33
            IOT C
            JMP CHANGE +2

D,          LAC (ISZ DATA +37
            DAC CHANGE +4
            LAC (LAC DATA +37
            DAC CHANGE +5
            LAC (JMP G03+4
            DAC D
            JMP G03+4
```

```
/MAIN PROGRAM
/THIS PROGRAM SEQUENTIALLY TRANSFERS INSTRUCTIONS
/TO THE CONTROL UNIT FOR EXECUTION BY THE
/PROCESSING ELEMENTS.  THESE INSTRUCTIONS ARE
/STORED IN PDP 9 MEMORY BEGINNING WITH LOCATION
/"LIST".

/WHEN THE INSTRUCTION XCT IS ENCOUNTERED IN
/LIST, THIS INDICATES THAT A SEQUENCE OF
/INSTRUCTION IS TO BE REPEATED, WITH THE
/EXCEPTION THAT ADDRESSES ARE TO BE MODIFIED.
/THIS CAUSES CONTROL TO BE TRANSFERED TO A
/PDP 9 SUBROUTINE WHICH MODIFIES THE
/INSTRUCTIONS AND CONTINUES THE DATA TRANSFER.

/THE SEQUENCE LIST IS ALSO TERMINATED BY A
/"NOP" TRANSFERRING CONTROL TO A ROUTINE WHICH
/TRANSFERS INSTRUCTIONS TO TEST FOR CONVERGENCE.

PE INST,    LAC I 10          /LOAD A PE INSTRUCTION FROM
            SAD (NOP          /LIST
            JMP CONVERGE      /ON NOP, TEST FOR CONVERGENCE
            SAD (XCT          /XCT INDICATES A SEQUENCE OF
            JMP MODIFY        /INSTRUCTIONS IS TO BE
                              /MODIFIED AND REPEATED

            IOT C             /IOT PE INSTRUCTION TO CONTROL

            JMP PE INST       /CONTINUE TRANSFER

/THIS SUBROUTINE WILL MODIFY A SEQUENCE
/OF INSTRUCTIONS CONTAINED IN LIST.  THESE
/INSTRUCTIONS ARE ONE WHICH ARE TO BE
/REPEATED WITH ONLY A CHANGE IN ADDRESS.
/THE SUBROUTINE WILL IOT THE INSTRUCTION
/AS SOON AS IT HAS BEEN MODIFIED.

MODIFY,     LAC 12            /TEMPORARILY STORE 12
            DAC TEMP

MODIFY +2,  LAC I 12          /LOAD INSTRUCTION TO BE MODIFIED
            SAD (RSHF 2       /RSH 2 IS NOT TO BE MODIFIED
            JMP .+6           /IOT RSH 2 WITHOUT MODIFYING

            SAD(XCT           /END OF INSTRUCTION SEQUENCE
            JMP E1
MODIFY +6,  ADD (1            /MODIFY INSTRUCTION ADDRESS
```

```
MODIFY +7,      SAD (PSTA I 40          /END OF COLUMN
                JMP E                   /YES.  CHANGE COLUMN LIMIT
                IOT C                   /NO.  MODIFY & IOT

E,              LAC (ADD (1
                DAC MODIFY +6
                LAC MODIFY +7           /NEW COLUMN LIMIT
                ADD (40
                DAC MODIFY +7
                JMP PE INST

E1,             LAC TEMP                /RESTORE REGISTER 12 & CONTINUE
                ISZ MODIFY +6
                DAC 12
                JMP MODIFY +2
```

/THIS SUBROUTINE TRANSFERS A SEQUENCE OF
/INSTRUCTIONS WHICH TEST FOR CONVERGENCE OF
/THE SOLUTION. IF THE SOLUTION HAS CONVERGED,
/IT CAUSES A STOP; IF THE SOLUTION HAS NOT
/CONVERGED IT RESTORES THE INSTRUCTIONS WHICH
/HAVE BEEN MODIFIED. IT ALSO RESTORES
/AUTOINDEX REGISTERS AND RETURNS CONTROL TO
/THE MAIN PROGRAM (PE INST).

```
CONVERGE,       LAC I 10                /LOAD LIST +40
                IOT C                   /TO CONTROL
                IOT #1                  /RETURN CONTENTS OF 158 IN PE #0
                                        /TO PDP 9 ACC
                SAD (COMPARE            /IF THE ACCUMULATOR IS THE SAME
                JMP OUT DATA            /AS COMPARE THE RESULT HAS
                                        /CONVERGED

                DAC COMPARE             /NEW VALUE OF COMPARE

F,              LAC (LIST -1            /RESTORE AUTOINDEX REGISTERS
                DAC 10                  /AND RETURN TO MAIN PROGRAM
                DAC 12
                LAC (PSTA I 40
                DAC MODIFY +7
                JMP PE INST
```

/OUT DATA OUTPUTS FINAL RESULT TO PDP 9

```
OUT DATA,       LAC I 10                /OUTPUT FINAL INSTRUCTION IN LIST
                IOT C                   /WHICH STORES THE VALUE OF ANY
                                        /POINT OF THE MATRIX IN A PE
                                        /ACCUMULATOR
```

```
IOT #5                  /IOT TO PDP 9 FROM A PE (#5)
DAC FINAL
HLT
```

```
/DATA IS THE FIRST LOCATION OF A SEQUENCE
/CONTAINING CONSTANTS INTERSPERSED WITH INSTRUCTIONS.
/A CONSTANT IS IOT TO THE BROADCAST
/REGISTER OR THE PE ACCUMULATORS FOR
/TRANSFER TO PE MEMORY BY INSTRUCTIONS
/WHICH FOLLOW THE CONSTANT IN THE SEQUENCE
/DATA.  CONSTANTS TO BE STORED INCLUDE
/THE NUMBERS 0 AND 1 AND ALL BOUNDARY
/CONDITIONS.  MODES WHICH ARE TO BE SET FOR
/PE'S 0 & 7 ARE ALSO TRANSFERED FROM
/THIS LIST, EACH MODE BEING FOLLOWED BY
/TWO INSTRUCTIONS WHICH STORE AND LOAD THE MODE.

/IN THE CASE OF THE LEFT AND RIGHT
/BOUNDARIES WHICH ARE STORED IN SEQUENTIAL
/MEMORY LOCATIONS IN PE'S 0 & 7 A SUBROUTINE
/"CHANGE" MODIFIES THE ADDRESS PORTIONS OF
/THE INSTRUCTIONS WHICH ARE TO BE REPEATED
/IN ORDER TO STORE THE BOUNDARY CONDITIONS.
/THIS SUBROUTINE IS ENTERED WHEN THE
/INSTRUCTION NOP IS ENCOUNTERED IN THE DATA LIST.
/THE INSTRUCTION XCT IS USED TO DELIMIT BETWEEN
/CONSTANTS AND BOUNDARY CONDITIONS.

DATA,       000001          /MODE FOR PE #0 TO BE LOADED
                            /INTO PE #0 ACCUMULATOR
            PSTM 00 I 200   /MODE TORE MODE INTERNAL LOCATION
                            /INTO 00.  NOTE THAT ALL PE'S
                            /WILL RESPOND TO THIS
                            /INSTRUCTION; HOWEVER, PE'S 1-7
                            /HAVE 0 IN THEIR ACCUMULATORS.

            000002
            PSTM 00 I 201   /MODE FOR PE #7

            000000          /THE CONSTANT 0 WHICH IS IOT
                            /TO BROADCAST
            PCLA 00 B -     /ALL PE'S ADD 0 TO THEIR ACC
                            /FROM BROADCAST ADDRESS IRRELEVENT

            PSTA 00 I 210   /ALL PE'S STORE 0 IN LOCATION
                            /210

            000001          /CONSTANT 1 TO BE IOT TO
                            /BROADCAST
```

```
                PCLA 00 B-
                PSTA 00 I 211       /ALL PE'S STORE 1 IN LOCATION
                XCT                 /211.  XCT TERMINATES CONSTANTS
                                    /TO BE TRANSFERED.

                UPPERBOUND          /UPPERBOUNDARY CONDITION IOT
                                    /TO BROADCAST
                PCLA 00 B-          /ALL PE'S LOAD UPPERBOUND IN
                                    /THEIR ACCUMULATORS
                PSTA 00 I 000
                PSTA 00 I 040       /ALL PE'S STORE UPPERBOUND IN
                PSTA 00 I 100       /LOCATIONS, 0, 40, 100 & 140
                PSTA 00 I 140

                LOWERBOUND          /IOT TO BROADCAST IOT
                PCLA 00 B-          /INSTRUCTIONS TO CONTROL
                PSTA 00 I 037
                PSTA PP I 077       /ALL PE'S STORE LOWERBOUND IN
                PSTA 00 I 137       /LOCATIONS 37, 77, 137, 177
                PSTA 00 I 177

                PLDM 00 I 200       /PE#0 CHANGES MODE TO 1 & PE #7
                PLDM 00 I 201       /CHANGES TO MODE 2

                LEFTBOUND           /STORE LEFT BOUNDARY IN PE #1
                PCLA 31 B-          /ONLY.  NOP TRANSFERS CONTROL
DATA +33,       PSTA 31 I 001       /TO SUBROUTINE TO MODIFY DATA
                NOP                 /+33 ADDRESS

                RIGHTBOUND          /STORE RIGHT BOUNDARY IN PE #7
                PCLA 32 B-          /ADDRESSES 141-177
DATA +37        PSTA 32 I 141
                NOP

DATA +41        HLT                 /THIS TERMINATES THE TRANSFER
                                    /OF CONSTANTS TO THE PROCESSING
                                    /ELEMENTS.
```

```
/LIST IS A SEQUENCE OF INSTRUCTIONS WHICH
/SOLVE THE PARTIAL DIFFERENTIAL EQUATION.
/FOUR BLOCKS OF SIX EQUATIONS WHICH, WHEN
/TRANSFERED TO THE PROCESSING ELEMENTS,
/SOLVE THE DIFFERENCE EQUATIONS OVER A
/32X32 GRID.  EACH BLOCK OF SIX INSTRUCTIONS
/IS TERMINATED BY XCT WHICH CAUSES ENTRANCE
/TO SUBROUTINE MODIFY WHICH MODIFIES THE
/ADDRESS PORTION OF FIVE OR THE SIX
/INSTRUCTIONS AND IOT THE MODIFIED
/INSTRUCTIONS TO EFFECT ITERATIONS THROUGH
/A COLUMN OF THE 32X32 GRID.

/LIST ALSO CONTAINS AN INSTRUCTION WHICH
/TESTS FOR CONVERGENCE TO THE FINAL SOLUTION

/INSTRUCTIONS IN THIS SEQUENCE ARE TRANSFERED
/BY THE MAIN PROGRAM (PE INST) AND THE
/SUBROUTINE MODIFY PRIOR TO ENTRY OF THE
/MAIN PROGRAM WHICH OUTPUTS THE INSTRUCTIONS
/IN LIST, THE MODES OF PE'S 0 AND 7 HAVE
/BEEN SET TO 1 AND 2 RESPECTIVELY.  ALL
/OTHER MODES ARE SET TO 0.

LIST,       PCLA 02 I 041      /THESE SIX INSTRUCTIONS ARE
            PADD 02 I 000      /EXECUTED BY PE'S 1-7 TO
            PADD 02 I 002      /SOLVE THE DIFFERENCE EQUATION
            PADD 02 L 141      /FOR ONE POINT IN PE #0, THIS
            RSHF 02     2      /IS A BOUNDARY POINT XCT
            PSTA 02 I 001      /ENTERS THE SUBROUTINE MODIFY
            XCT

            PLDM 12 I 210      /PE'S 0&7 CHANGE THEIR MODE
                               /TO 0

LIST +10,   PCLA 00 I 040      /ALL PE'S EXECUTE THE SEQUENCE
            PADD 00 I 001      /WHICH WILL SOLVE THE DIFFERENCE
            PADD 00 I 102      /EQUATION OVER ANOTHER COLUMN
            PADD 00 I 042
            RSHF 00     2
            PSTA 00 I 041
            XCT

LIST +17,   PCLA 00 I 100      /NEXT COLUMN OF MATRIX
            PADD 00 I  41      /ALL PE'S
            PADD 00 I 102
            PADD 00 I 141
```

```
                RSHF 00     2
                PSTA 00 I 101
                XCT

LIST +26        PLDM 00 I 201       /ALL PE'S HAVE 0 IN LOC 201
                                    /EXCEPT #7 THIS PE LOADS
                                    /MODE 2.

LIST +27        PCLA 00 I 140       /LAST COLUMN, PE'S 0-6 ONLY.
                PADD 00 I 101       /THIS IS A BOUNDARY POINT IN
                PADD 00 I 142       /PE #7
                PADD 00 R 001
                RSHF 00     2
                PSTA 00 I 141

LIST +35        XCT
                NOP

                PLDM 00 I 200       /PE #0 LOADS MODE 1

LIST +40        PCLA 11 I 158       /PE #0 WILL IOT THIS LOCATION
                                    /TO PDP 9 ACC AND SUBROUTINE
                                    /CONVERGE WILL TEST FOR
                                    /CONVERGENCE

LIST +41        PCLA                /LOAD ACC WITH ANY DESIRED POINT
                                    /IN MATRIX FOR FINAL RESULT

/IN THE PREVIOUS PROGRAM IT WAS ASSUMED THAT ALL LOCATIONS IN
/PE MEMORY REQUIRED FOR DATA STORAGE WERE CLEARED.  IF THEY
/WERE NOT, THE PROGRAM STARTS AT "CLEAR" INSTEAD OF AT "START"
/THIS PROGRAM CLEARS PE MEMORY.  ADDITIONAL TIME IS 2321
/CYCLES.  THIS PROGRAM TRANSFERS CONTROL TO START ON COMPLETION.

CLEAR,          CLA
                IOT B               /IOT 0 TO BROADCAST
                LAC CLMEM
                IOT C               /ALL PE'S CLEAR THEIR ACC
                LAC CLMEM +1
CLEAR +5,       SAD (PSTA 00 I 256  /LAST PE MEMORY ADDRESS +1
                JMP START
                IOT C
                ADD (1
                JMP CLEAR +5

CLMEM,          PCLA 00 B-          /ALL PE'S LOAD ZERO IN THEIR
                                    /ACCUMULATOR
                PSTA 00 I 000
```