1974

Design of a minicomputer with hardware indexing.

H. S. Sodhi
University of Windsor

DESIGN OF A MINICOMPUTER WITH HARDWARE INDEXING

by

H. S. SODHI

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical Engineering
in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario

1974

ABSTRACT

This work describes the design and implementation of adding hardware indexing to the PDP-8/S computer.

For educational purposes, it is desirable to have an index register, which is a powerful tool in providing greater programming flexibility. Originally the PDP-8/S had a limited facility for address modification in the form of auto-indexing. In the modified version, this has been replaced by a more useful indexing feature, the addition of a 12-bit index register. Four basic instructions for loading, adding, storing and incrementing with a conditional skip have been provided for the index register. Two additional bits are required due to the index register: one for specifying the presence or absence of index address modification and the second to provide additional operation codes to manipulate the contents of the index register; as a result the memory "page" size has been reduced from 128 words to 32 words. This is not a serious handicap for instructional purposes. However, a switch is incorporated so that the computer can be used in the normal mode, thereby, retaining the larger "page" size but without the special indexing feature.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. P.A.V. Thomas for his supervision and valuable suggestions. Furthermore, the author's thanks are extended to the National Research Council of Canada for its generous financial support.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 DESIRABILITY OF AN INDEX REGISTER

As minicomputers are becoming more widely available at very low cost, they can be used as a convenient teaching aid to demonstrate the various hardware features and also for teaching machine language and assembly language programming rather than using a larger computer with some higher level language. The minicomputer has the added advantage that the student can operate the machine with "hands on" and thus interact with the computer. As the console lights flicker, there is a certain fascination at having complete control over the logical mechanism under your fingertips.

Most of the minicomputers perform the same kinds of arithmetic and logic, work with the same kinds of computer peripherals, and use the same kinds of computer languages as traditional computers but few of the earlier machines had an index register (for address modification). The possible reasons for the sacrifice of programming flexibility with indexing are (a) the desire to keep the cost of a minicomputer low and (b) the smaller word length in most of the earlier minicomputers.

The PDP-8/S was chosen to add an index register because it has a reasonable memory and instruction size

and was available for mainly instructional purposes.

1.2 IMPLICATIONS:

In order to achieve the objective of adding index register, two major changes were effected:

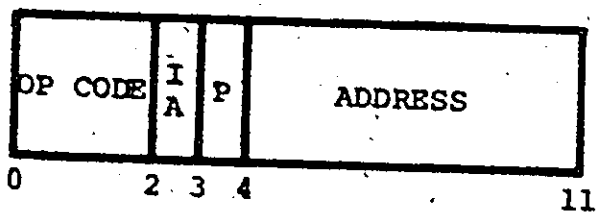
(a) changes in the instruction word structure;

(b) addition of the index register.

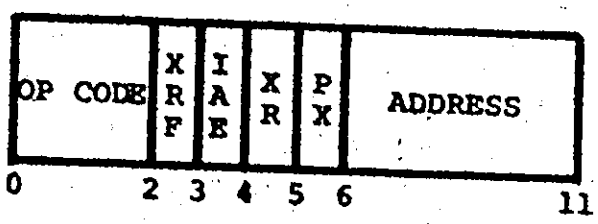
1.2(a) Changes in the Instruction Word Structure:

In the normal mode, the structure of an instruction word is as shown in figure 1(a). Bits (0-2) are provided for the operation code. Hence, eight basic commands are available. Seven bits (5-11) are allocated for address. Two remaining bits, 3 and 4, convey the address mode. Out of these two bits, single indirect addressing is indicated by bit 3 being 1 and bit 4 determines whether the current or the base page is being accessed.

Due to the modifications for educational mode (with the addition of the index register), the bit configuration of the instruction word is changed as shown in figure 1(b). The first three bits (0-2) are still used for the operation code. The fourth bit called the index register function (XRF), in combination with the operation code bits, produces the extra commands needed due to the addition of the index register. Now in place of two in the normal form, three bits are used for the addressing mode. The indirect bit (IA) is shifted one to the right and is represented by bit 4, called IAE - indirect addressing for



(a) NORMAL



(b) EDUCATIONAL

FIGURE I: BIT CONFIGURATION OF INSTRUCTION WORD

educational mode. Bit 5 called XR, indicates whether indexing is required or not. Bit 6 is used for paging and is called PX. This leaves five bits (7-11) for memory addressing.

As can be seen, in the educational form the address portion is reduced to five bits as compared to seven bits in the normal form. Although it would tend to appear as a major limitation, on careful examination it is not because of two main reasons:

(i) Using the indirect addressing, even with this reduced page size, 4K memory programs can still be written, thereby, using the full memory.

(ii) For educational purposes, despite the smaller address portion of the instruction, all the original features plus the indexing capabilities of PDP-8/S can be demonstrated.

However, to make the modified version more versatile, a switch is provided to select either the educational or the normal form. In the educational mode, the extra indexing facilities are available but reduced page size. On the other hand, in the normal form, there are all the original facilities of the PDP-8/S and thereby retaining the larger page size and ability to run standard PDP-8/S programs.

1.2(b) Addition of the Index Register

In the original PDP-8/S, to get around the problem of automatic address modification, there existed an auto-indexing feature. The auto-indexing facility was invoked under the following conditions:

(1) cells 10_8 through 17_8 were referenced.

(2) they were referenced indirectly.

This arrangement had three main restrictions:

(i) The auto-indexing could be done only if cells (10_8-17_8) were referenced indirectly.

(ii) The auto-indexing could not be avoided if cells (10_8-17_8) were referenced indirectly.

(iii) The address could be modified by an increment of only 1.

To alleviate these problems, in the educational form, auto-indexing has been replaced by a more flexible indexing feature. In this mode, the first two restrictions are completely eliminated. For the third, we can have an increment from 0 to 31_{10}.

Just like all the other registers, the index register is a 12-bit register. In order to command and address the index register (XR), the instruction repertoire has been expanded with the addition of the four basic instructions given in table I.

Table I

MNEMONIC	OP CODE	XRF	MEANING
LDX	0	1	Load XR from memory
ADX	1	1	Add to XR from memory
ISX	2	1	Increment the contents of XR by address portion of instruction and skip if zero
DCX	3	1	Deposit the contents of XR in memory and clear XR

INSTRUCTION SET FOR THE INDEX REGISTER.

Chapter II

GENERAL DESIGN CONSIDERATIONS

The design of the hardware indexing facility of the PDP-8/S can be considered in two main parts:

(1) The logic design for the changed instruction word structure and decoding of the index register instructions.

(2) The timing modifications.

2.1 DECODER DESIGN:

Figure 2 gives the block diagram of the decoder design. It is possible to describe the design by discussing in general terms the various sections of the block diagram.

Mode control is achieved by a switch and a flip-flop. The output of the flip-flop produces either the educational mode or the normal mode.

The normal mode combines with the instruction register (IR) to produce the following bit configurations:

0 - 2 operation code

3 indirect addressing

4 paging

5 - 11 memory address.

Thus the original structure of the instruction word is kept.

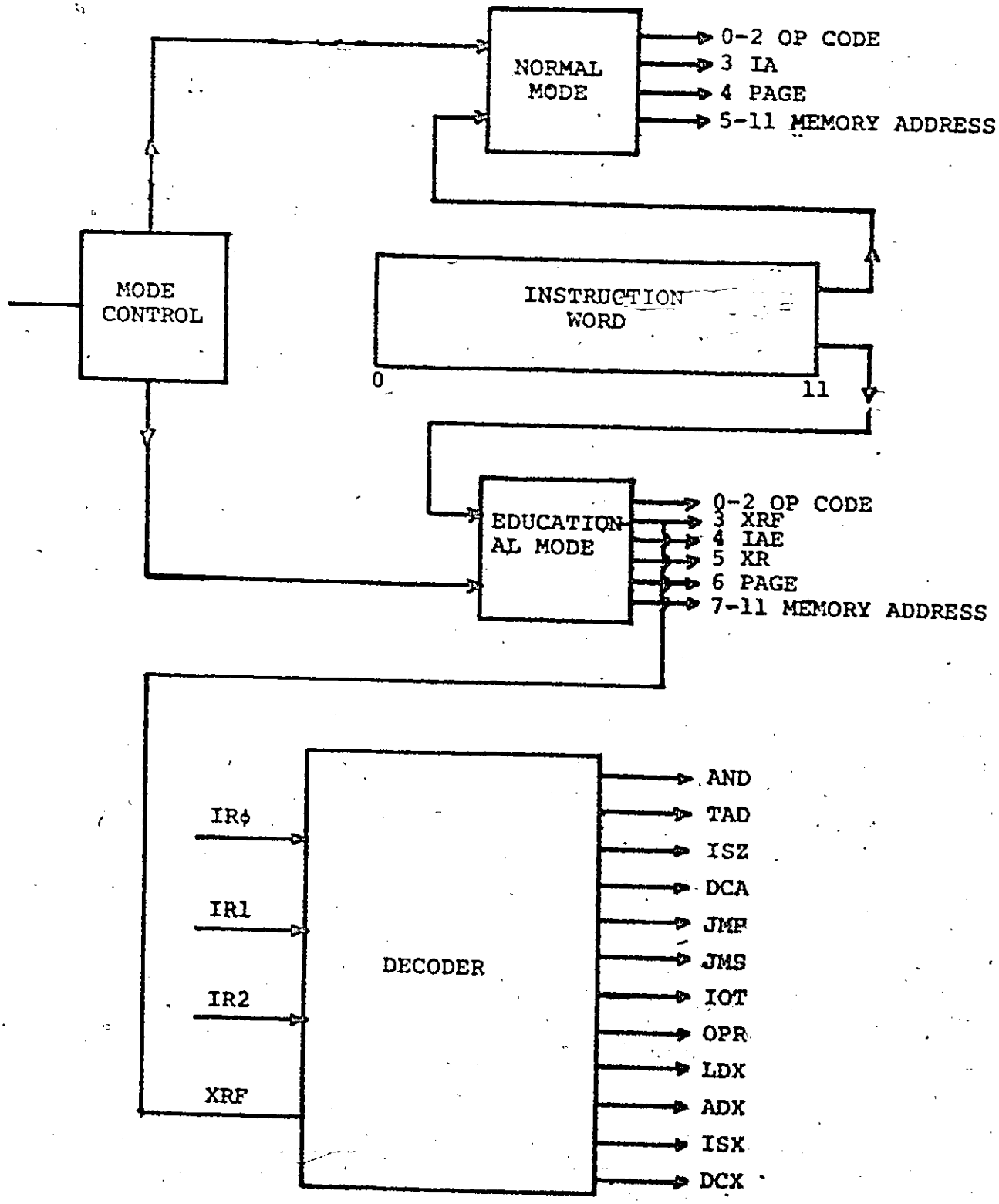


FIGURE 2: DECODER BLOCK DIAGRAM

The educational mode in conjunction with the instruction word provides the extra control bits needed for the index register instructions and to specify the presence or absence of indexing. It alters the word structure in the following fashion:

0 - 2 operation code

3 index register function

4 indirect addressing

5 indexing

6 paging

7 - 11 memory address.

The length of the instruction register is increased from four to six bits because now it has to hold two additional bits for generating the various instructions and effecting the address modification.

In the educational form, the four high order bits - the operation code and XRF bit, instead of the three in the original PDP-8/S are fed to the decoder to produce the basic PDP-8/S and index register instructions.

2.2 TIMING MODIFICATIONS:

In the unmodified version, each word-time is made up of 14-bit times (numbered ϕϕ through 13) during which the clock generates a string of fourteen bit pulses 750 ns apart. The first twelve pulses perform whatever serial operations are required on one or more 12-bit words. The thirteenth pulse performs most of the individual

operations that are required for any instruction, checks parity and requests a memory cycle if one is required. When a memory cycle is requested, the processor clock stops, and memory goes through its cycle controlled by its own clock. Upon completion of the cycle, the processor clock restarts at the fourteenth pulse, which determines the transition to the next word-time. When the computer is stopped by the program or the operator, it does so between the thirteenth and fourteenth bit times, i.e., at time 13 but following the completion of the memory cycle if one is requested.

The word-time in which a word is processed depends upon what type of information it represents (an instruction, an address, an operand) and what functions must be performed on it. There are six word-times: Fetch, Index, Defer, Execute, End, Break. Although the execution of an instruction begins in the Fetch time, the program must start in the End time, which determines the location of the first instruction and requests a memory cycle to retrieve it. In other words, Fetch time does not fetch the instruction. Fetch time processes the instruction just retrieved from memory. It transfers the instruction code to the instruction register for decoding, transfers the address part to the memory address register (MA), and increments the program counter (PC) so that it will point to the next location.

At the end of Fetch time, the processor requests memory access if the instruction is indirectly addressed or requires the retrieval of an operand. If an auto-indexing location (10_8-17_8) is indirectly addressed, the processor then enters the Index time, for any other indirect addressing it enters the Defer time; otherwise it goes to the Execute time. In Index time the address that has been retrieved is incremented by one and written back in memory. The processor then enters the Defer time to move the new address to MA and requests a memory cycle, if any operand must be retrieved. The actual logical, arithmetic or program control operation specified by the instruction is then performed in the Execute time, which requests access if an operand must be deposited in memory. In the End time the processor determines the location of the next instruction and fetches it. In some cases, the Execute and End times are simultaneous; if they are not, the processor returns to Fetch time unless a program interrupt has been requested. In this case, the processor enters the Break time.

In the modified version, most of the sequences are the same as described above except that a few changes have been made for the indexing feature. The auto-indexing has been replaced by more useful indexing facilities. The Index time originally used for auto-indexing is now utilized for indexing. It is possible to explain the

timing flow by discussing the various sections in the timing diagram of figure 3.

The program starts in End time WTE, which finds out the location of the first instruction and requests a memory cycle to retrieve it. The flow after the End time, depends upon the possible values of bits four and five and can be considered in four parts. The first two parts represent the conditions existing in the normal form.

2.2(a) No Indexing or Indirect Addressing

When both bits 4 and 5 are 0, it indicates that this instruction does not require indexing or indirect addressing. In Fetch time (WTF), the operation code is transferred to the instruction register for decoding, the address portion to the MA and the program counter is incremented by one so that it will point to the next location. At the end of the Fetch time, if required an operand is retrieved from the memory. Next follows the Execute time (WTX), during which the logical, arithmetic or control operation specified by the instruction is performed. The processor control is then passed on automatically to the End time.

2.2(b) Only Indirect Addressing

In any instruction, this kind of situation is indicated by bit 4 being 1 and bit 5 as 0. At the end of the Fetch time the processor requests memory access to get the effective address given by the contents of the

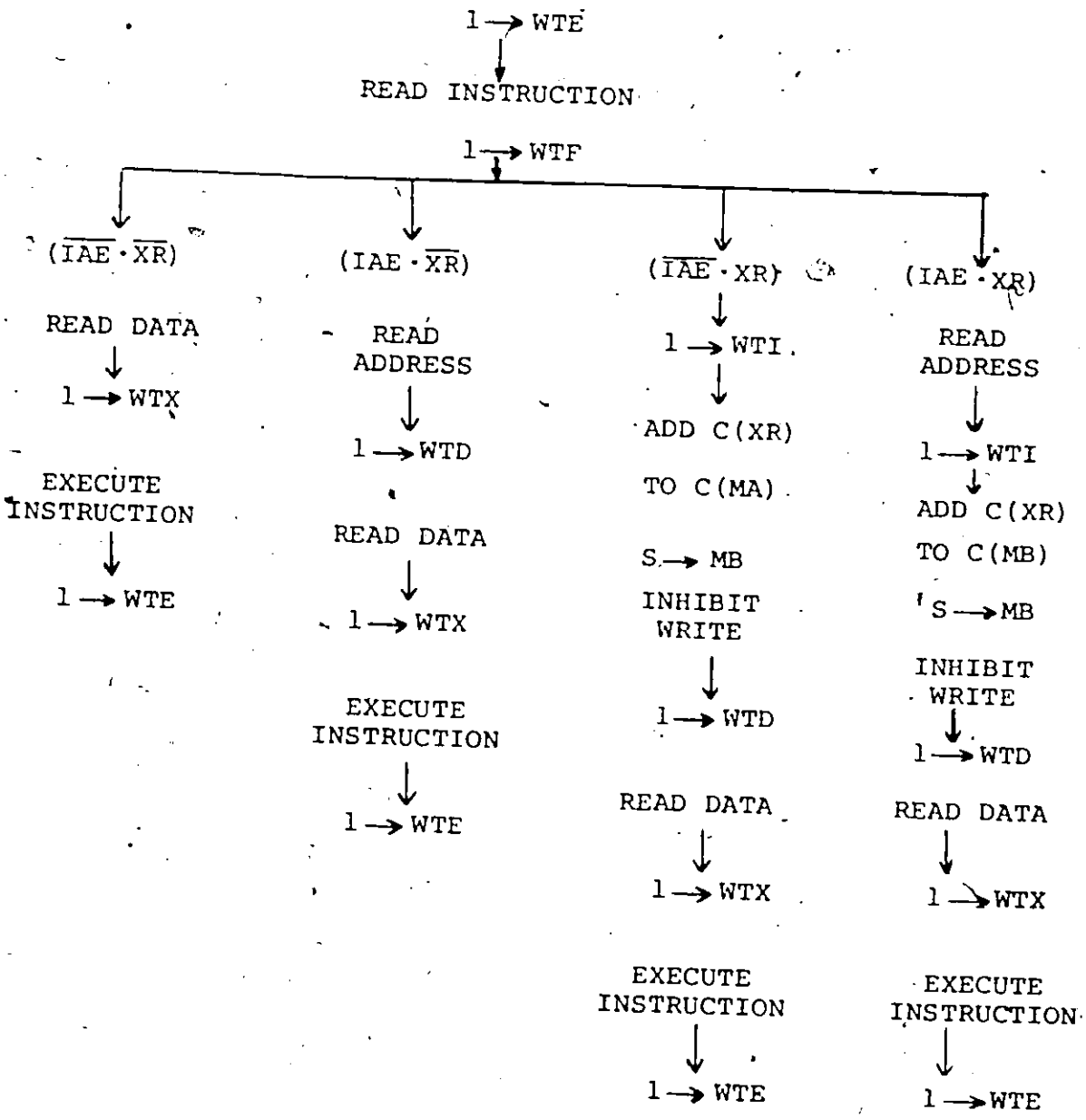


FIGURE 3: WORD TIMING DIAGRAM

memory location pointed to by the address portion of the instruction. The processor then enters the Defer time (WTD) to move the new address to the MA and request a memory cycle, if an operand must be retrieved. In the Execute time, the desired operation is performed and control is transferred to the End time.

2.2(c) Only Indexing

In any instruction, this condition exists when bit 4 is 0 and bit 5 is 1. After the various operations are performed in the Fetch time the processor enters the Index time (WTI). In the Index time the contents of the index register are added to those of the memory address register and the resulting sum is stored in the memory buffer register (MB). For auto-indexing, the incremented address was written back in the memory at the end of the Index time. But now this writing back into the memory is inhibited and the processor enters the Defer time. The effective address is moved to the MA and a memory cycle is requested, if an operand must be retrieved. After the address modification has been achieved in the Index and Defer times, the processor goes to the Execute time to perform the actual logical or arithmetic operation specified by the instruction. The processor then goes to the End time.

2.2(d) Both Indexing and Indirect Addressing

When both bits 4 and 5 are 1, it indicates that in this instruction indexing as well as indirect addressing is

desired. In this type of situation, there are two possibilities, i.e., indexing after indirect addressing or vice versa. Both are useful features but indirect addressing followed by indexing has been implemented in the educational version of the PDP-8/S. In the Fetch time the processor transfers operation code to the IR for decoding, the address part to the MA and increments the PC. At the end of the Fetch time, it requests memory access to read the address (the contents of the memory location pointed to by the address portion of the instruction) because the instruction is indirectly addressed. The processor then enters the Index time. During this time the contents of the index register are added to those of the memory buffer register (MB) and the sum is put into MB. Writing back of this address into the memory at the end of the Index time is inhibited. The control then passes on to the Defer time and the data stored in location pointed by the effective address is read. The actual arithmetic, logical or control operation is performed in the Execute time as usual before going into the End time. The End time determines the location of the next instruction and requests a memory cycle to retrieve it.

Chapter III

DETAILED DESIGN

Before describing the actual detailed design, it would be appropriate to briefly explain the organization of the drawings and the logic. All the implementation has been carried out using NAND logic. The related original drawings of the PDP-8/S have been attached in the appendix for ready reference. The logic network that existed before modification are shown in broken lines and all the added circuitry is shown in solid lines. The output terminals of a flip-flop are drawn twice using signs to show the polarities associated with either state of the circuit. They are drawn in such a way to eliminate excessive line crossing in showing the shift connections from one bit of a register to another.

3.1 INDEX REGISTER

This register is used to store the number required to modify the address of an instruction. By necessity the storage is non-destructive because the decision to exit from a loop is usually based upon the number stored in the index register. The index registers, most commonly, are made of the same length as the address field of an instruction. But in this case, the index register is 12-bit because it is convenient from a design point of view. Being of

full length, it can also serve as a temporary general purpose register. The circuit diagram of the index register XR, and various functions controlling it are shown in figure 4.

This register is similar to the other registers of the PDP-8/S, i.e., it is basically a right shift 12-bit register. The R202 flip-flop was considered versatile enough to meet these simple requirements. The DCD gates at the gated set and gated clear terminals are used for shifting by cross-connecting the output of the previous flip-flop to the level inputs of the DCD gates of the following flip-flop. Thus, when the pulse inputs of these DCD gates are pulsed, the shifting of the contents of the one flip-flop to the next is achieved. The function -(WTX·XRF + WTI) is formed because the contents of the XR need to be shifted right during the Execute time (WTX) to perform the various index register associated instructions or during the Index time (WTI) when the contents of the XR are to be added to the address field of the instruction. The clock pulses A(ϕϕ-11) as the pulse input and the function -(WTX·XRF + WTI) as the level input to DCD gate of the pulse amplifier R603 produce the index register shift pulses (XRSH). XRSH is a shift pulse that shifts the contents of the XR to the right by one bit. The logic net at the bottom left produces the functions which allow the sum from the serial adder to be shifted into the XR for LDX, ADX and ISX instruction during the Execute time

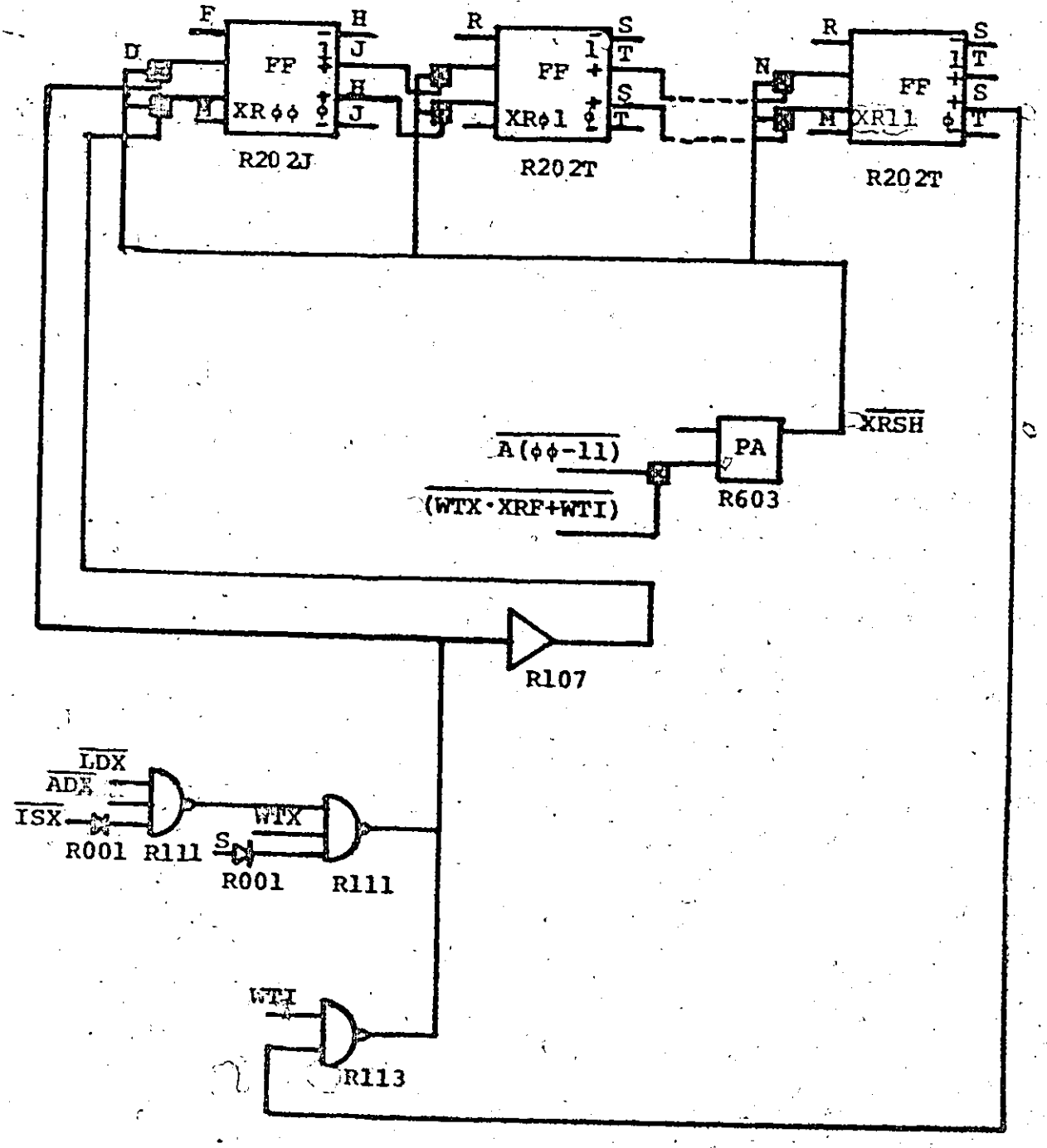


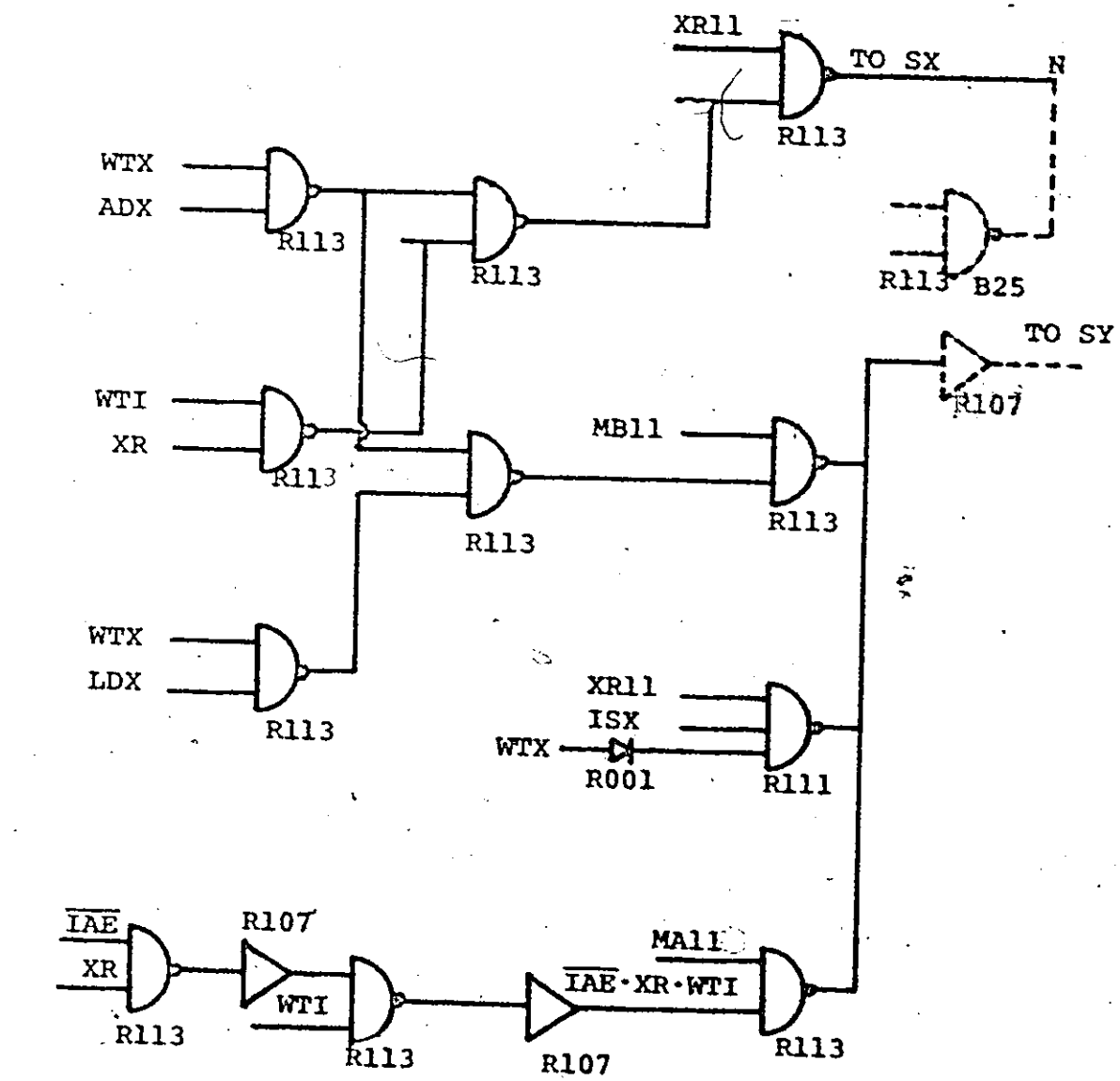
FIGURE 4: INDEX REGISTER

(WTX). For the Index time (WTI) the contents of XR are re-circulated by ANDING WTI with the ouput of XR.

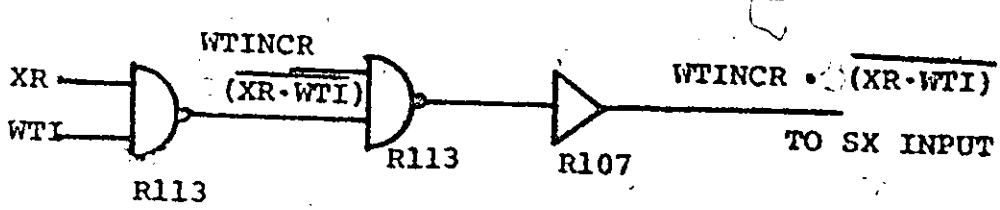
3.2 ADDER

The serial adder produces the sum of a pair of numbers one bit at a time. For each step, the circuit uses three inputs: two are bits from summands, the third is the carry-in from the previous step. At each step, two outputs are formed, a bit of the sum S, and the carry out C which is stored in a carry flip-flop for the next step. The modifications for the educational version on the original adder (drawing attached in appendix) circuits are shown in figure 5.

The input SX is generated by the upper left net. This input comes from the index register, for ADX and XR during WTI or AC for TAD. But in any other case in which addition is actually performed, SX is true only during BTϕϕ and hence has the effect of adding one to the number received as the SY input. BTϕϕ is gated-in by the function WTINCR for any operation that requires incrementing. When indexing is desired, this increment by 1 should not take place. This is achieved by ANDING WTINCR with -(XR·WTI) as shown. When there is no SX input, the number shifted out at S is identical to that shifted in at SY. This procedure is used to complement AC, to test it for zero, and to transfer it to PC.



MODIFICATIONS TO ADDER



INHIBIT WTINCR

FIGURE 5

The bottom net produces the input SY. As a result of modification, number shifted into SY is, from MB in LDX, from XR in ISX. Both these inputs occur for the Execute time (WTX). For the educational mode, when only indexing is desired the contents of the MA are shifted into SY. Function ($\overline{IAE}$·XR·WTI) which ensures that this input from the MA into SY occurs only for indexing is produced as shown by the logical net at the bottom left.

3.3 INSTRUCTION REGISTER (IR)

Before going into modifications of the index register itself, we need to produce mode control lists. This is shown in figure 6. NORMAL and EDUCATIONAL bits are obtained with the simple circuit combination of a R204 flip-flop and a switch.

Originally, the instruction register held only the operation code and indirect bit during the processing of each instruction. But IR has now been extended from 4 to 6-bits to hold two extra bits: XRF and XR. The altered configuration of the IR is shown in figure 7. R2ϕ2 flip-flop was considered versatile enough to serve as storage for the two additional bits and be able to right shift the contents of the IR for the execution of any instruction. IR is cleared only in special circumstances, such as beginning of a console operation or a program break. After each instruction is fetched from the memory, the contents of the MB are shifted into IR by the standard

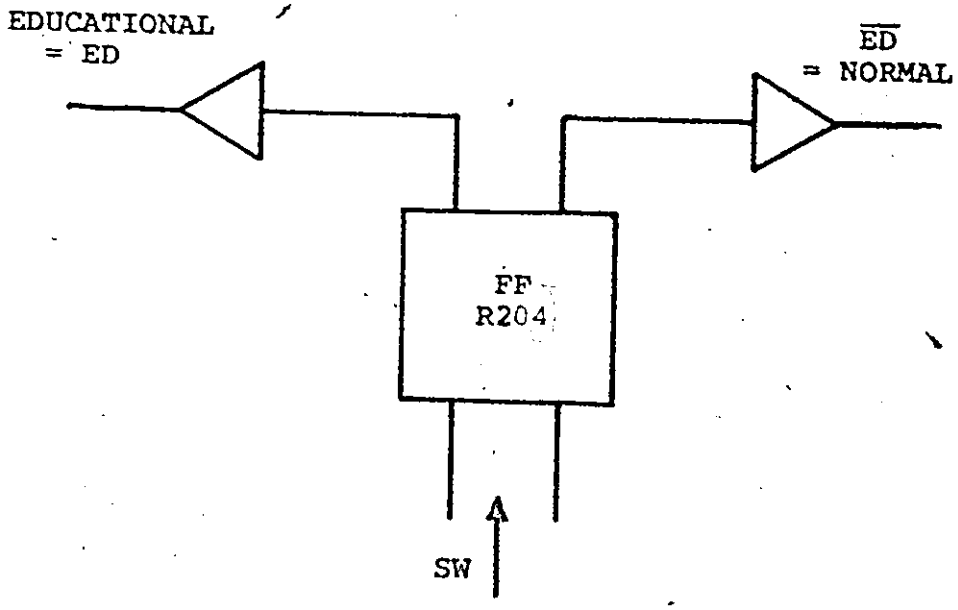


FIGURE 6: MODE CONTROL

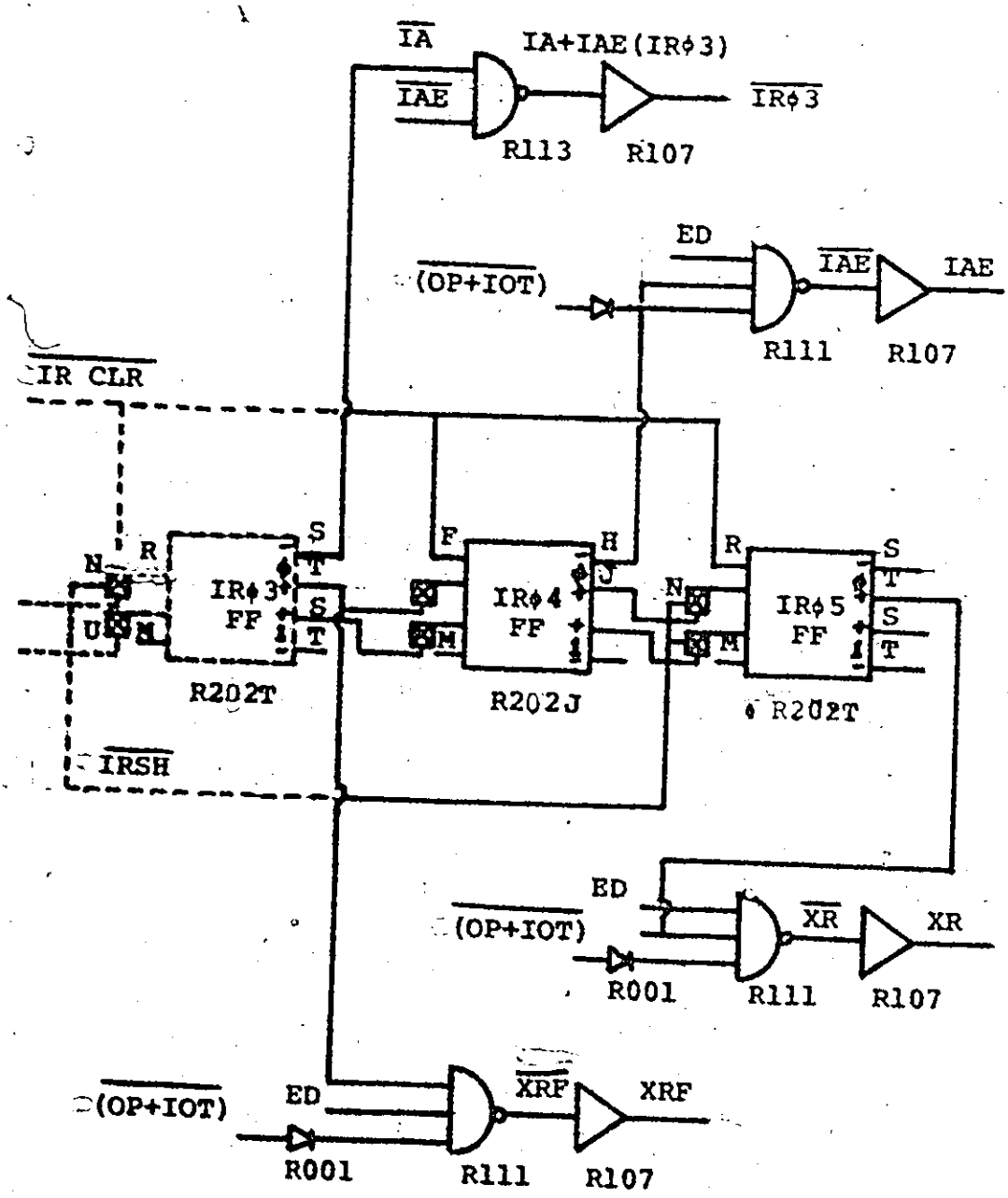


FIGURE 7: MODIFICATIONS TO IR

12-bit shift, which therefore leaves the left 6-bits of the word in the IR and drops out the address part at the right. Three combinations of a NAND(R111) and invertor (R107) gates produce XRF - bit required for index register instructions, IAE - indirect bit for educational mode and XR - indexing bit. In the normal form, we want IRϕ3 bit to act as indirect bit while in the educational form IR04 is used as an indirect bit. These two bits are ORed together as shown and output is fed to the pins where IRϕ3 was originally the only input. This way indirect addressing takes place when IRϕ3 bit is 1 or IR04 bit is 1 depending upon whether operating in the normal or the educational mode.

3.4 DECODING CIRCUITS FOR THE INDEX REGISTER INSTRUCTIONS

There are four simple logic networks to produce the required additional instructions for the index register. All four networks are similar; considering the one for decoding ISX - increment by immediate address and skip if the contents of XR are zero. This is generated by combining ISZ and XRF as shown in figure 8.

ISX is required when the index register function bit XRF is 1. At the top, XRF is NANDED with the inverted output of the decoder (DCDR) to produce ISX. But in order to retain ISZ for the normal mode and for educational mode, as part of the complete instruction repertoire, the network at the bottom is utilized.

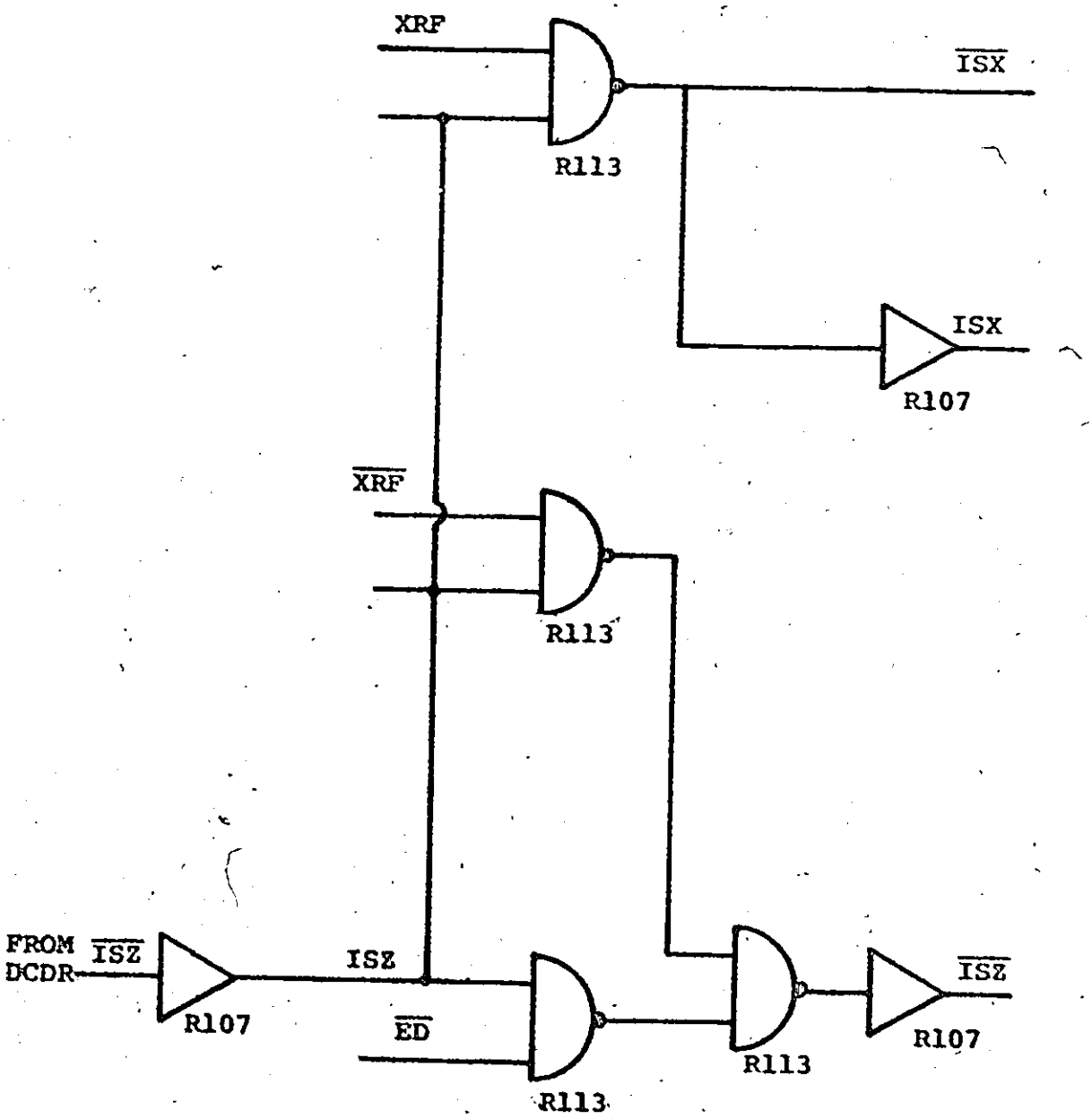


FIGURE 8: DECODING CIRCUIT FOR ISX

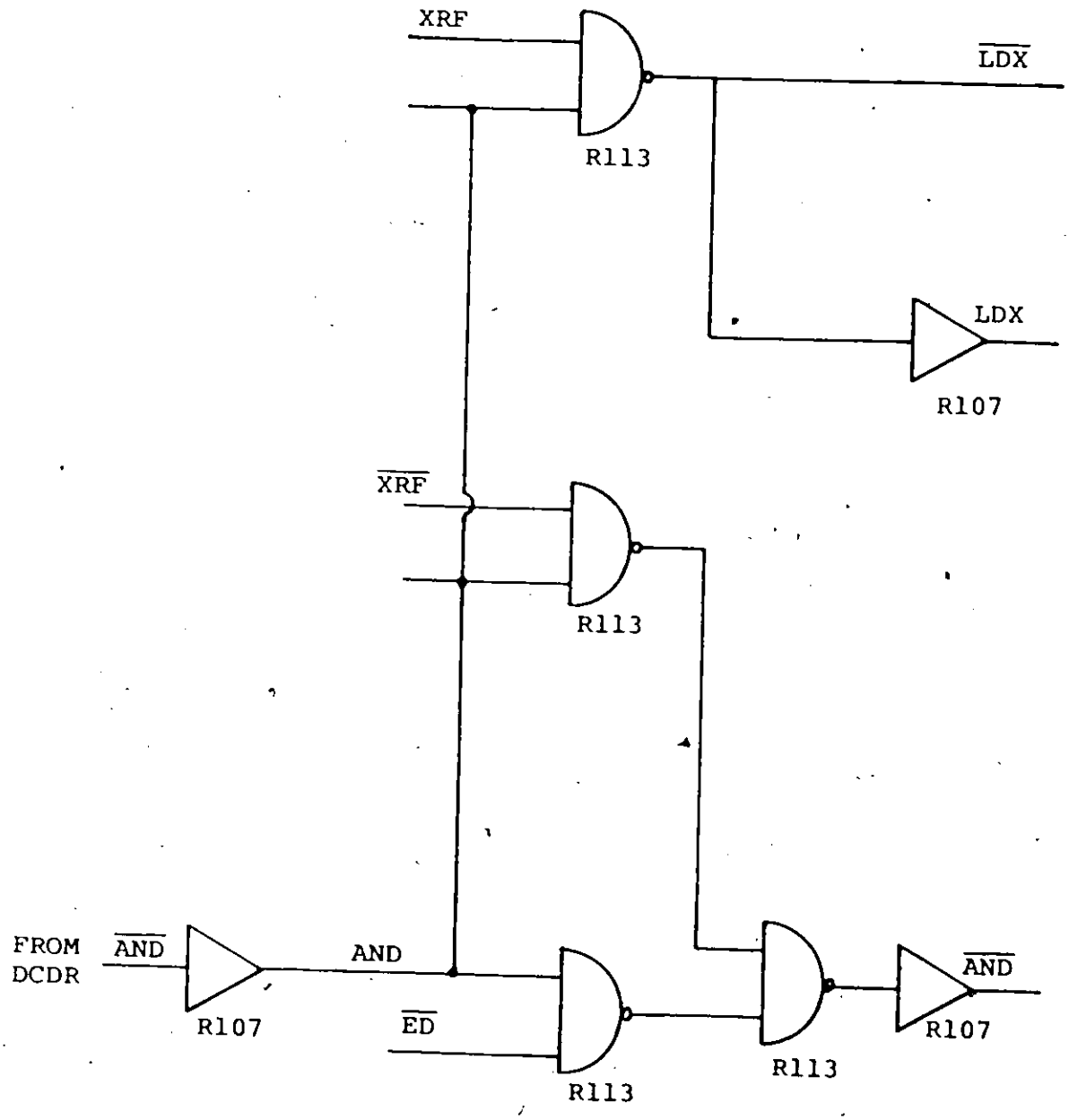


FIGURE 9: DECODING CIRCUIT FOR LDX

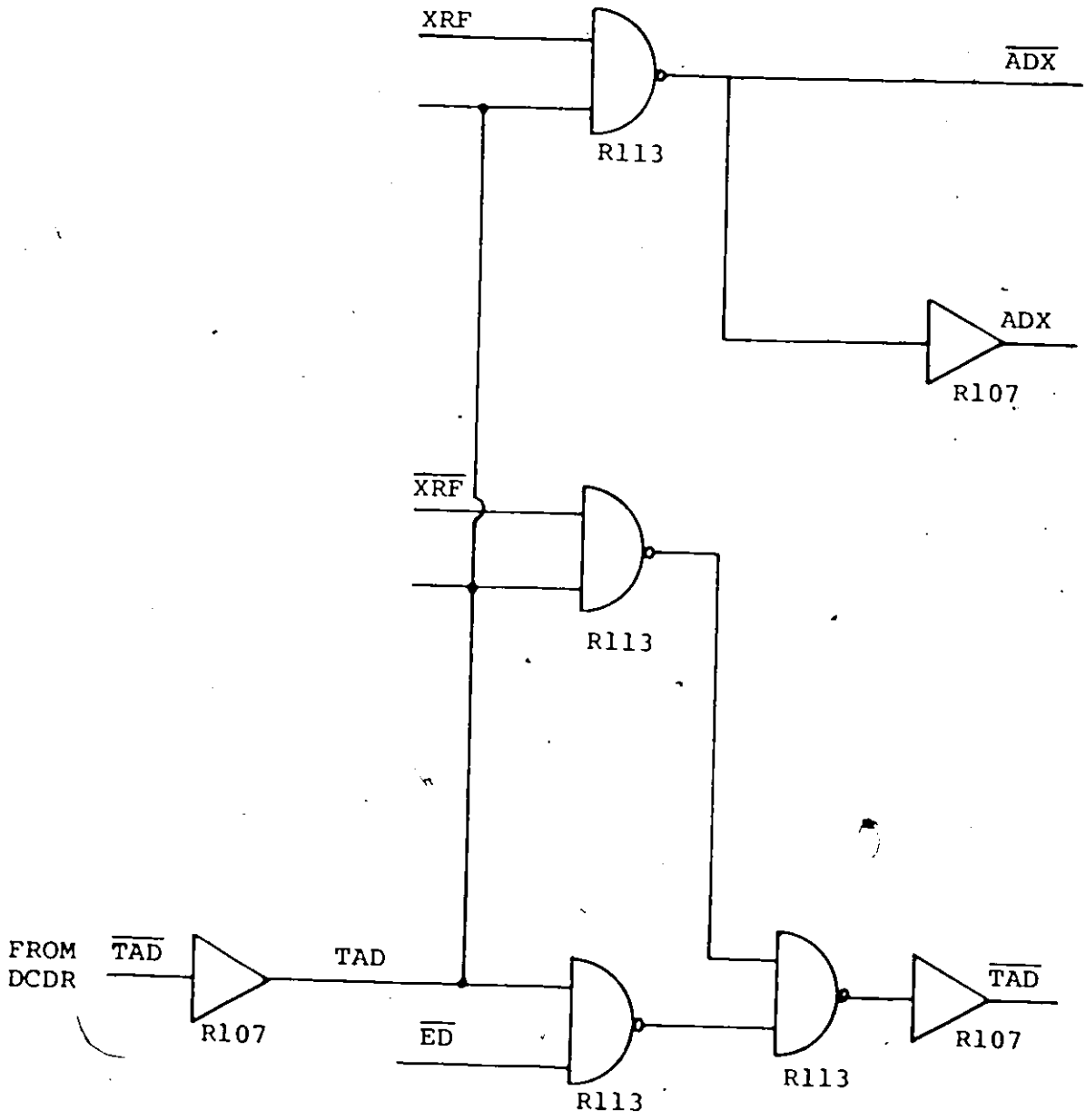


FIGURE 10: DECODING CIRCUIT FOR ADX

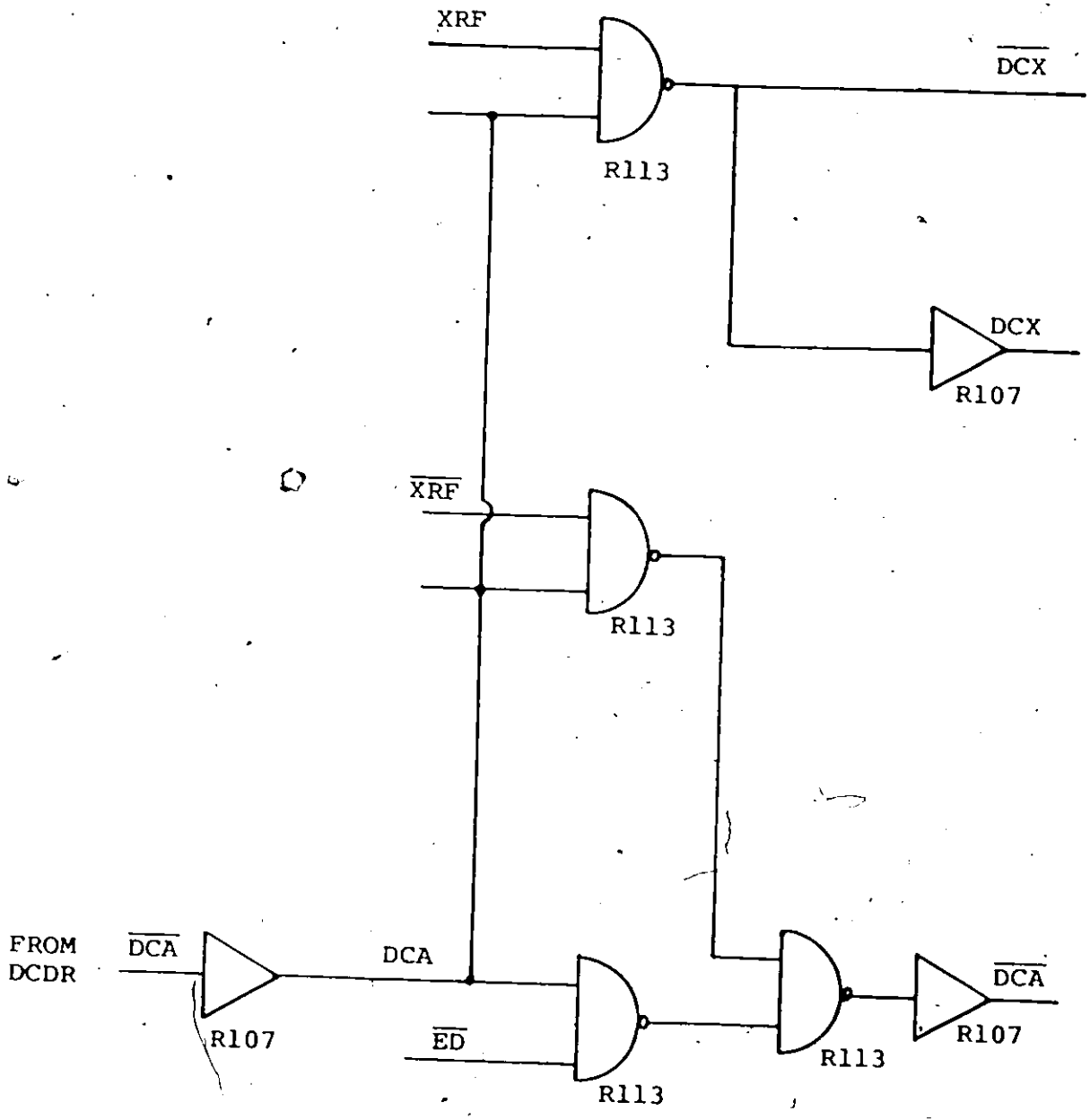
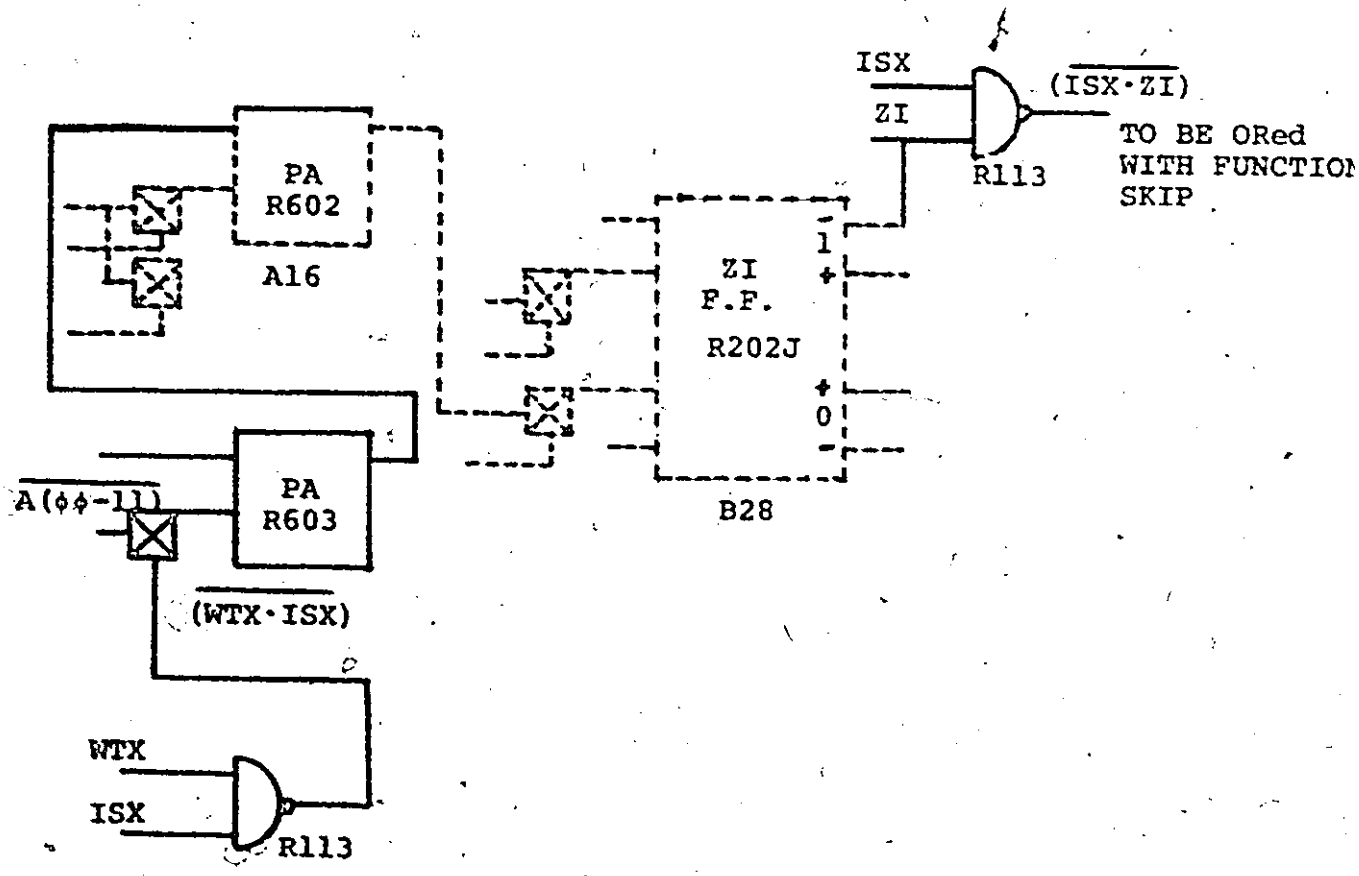


FIGURE 11: DECODING CIRCUIT FOR DCX



MODIFICATION TO ZI CIRCUIT

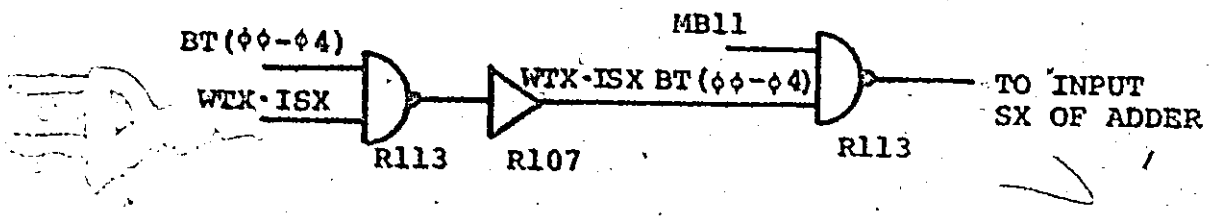


FIGURE 12

Similarly, figures 9, 10 and 11 show the decoding network for the remaining XR instructions - LDX, ADX and DCX respectively.

Associated with ISX, is the modification of the zero-indicator flip-flop (ZI), which is set by the final pulse in the Fetch time. It is cleared if the sum is ever true in the Execute time (WTX) of an ISZ or ISX or the operate skip group. This is shown in figure 12.

Unlike the ISZ instruction which allows the increment by only 1, ISX instruction is more flexible. The increment is specified by the immediate address of the instruction up to a maximum of 31_{10}. This is implemented as shown in the lower part of figure 12. The logic net allows the first five bits of MB, which represent the immediate address, to go into the SX input of the serial adder. For ISX, the input Sy of the adder is the contents of the index register.

3.5 MEMORY BUFFER REGISTER (MB)

Memory buffer register, as its name implies, is used as a buffer between the processor and the memory. Along with other shift inputs to MB$\phi\phi$ has been added another input from index register for depositing its contents in memory during the Execute time (WTX) and is shown in figure 13. NAND gate R111 which can have more than two inputs when an external diode is added to its pin S, was used for this logic implementation.

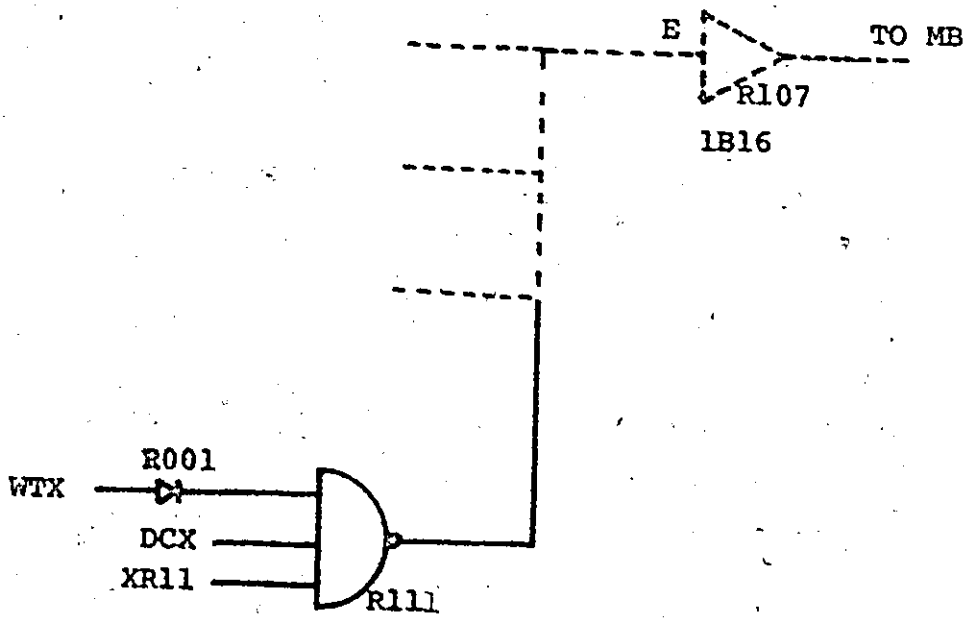


FIGURE 13: MODIFICATIONS TO MEMORY BUFFER REGISTER

3.6 MEMORY ADDRESS REGISTER (MA)

The 12-bits of memory address register select one of the 4096 locations in memory during a memory cycle. Modifications have been made mainly to the PAGE logic associated with MA. Paging provides a way to access the whole memory with only 7 bits assigned for address field in the instruction format. At the top of the figure 14 is shown the page zero flip-flop (PGZ), which is set by the first pulse in fetch time if bit 4 of the instruction is 0 in the normal mode or if bit 6 is 0 in the educational mode. In the normal form, the shift input nets supply the 7-bit address from MB for the first seven fetch pulses, but re-circulate the original contents of MA($\phi\phi$-ϕ4) in the rest of the word-time unless PGZ has been set, in which case the input is inhibited and MA will address page 0. On the other hand, in educational mode, the shift input nets supply 5-bit address from MB for the first five fetch pulses, but recirculate the original contents of MA$\phi\phi$-ϕ6 in the remaining word-time unless PGZ has been set, in which case just like normal mode the input is inhibited and MA will address page 0. MA receives 12-bit address from MB in the Defer time and recirculates itself when its contents are being transferred to PC in a jump. In the End time, except in JMP, it receives an address for the next instruction retrieval from PC, either directly or incremented by one through the adder.

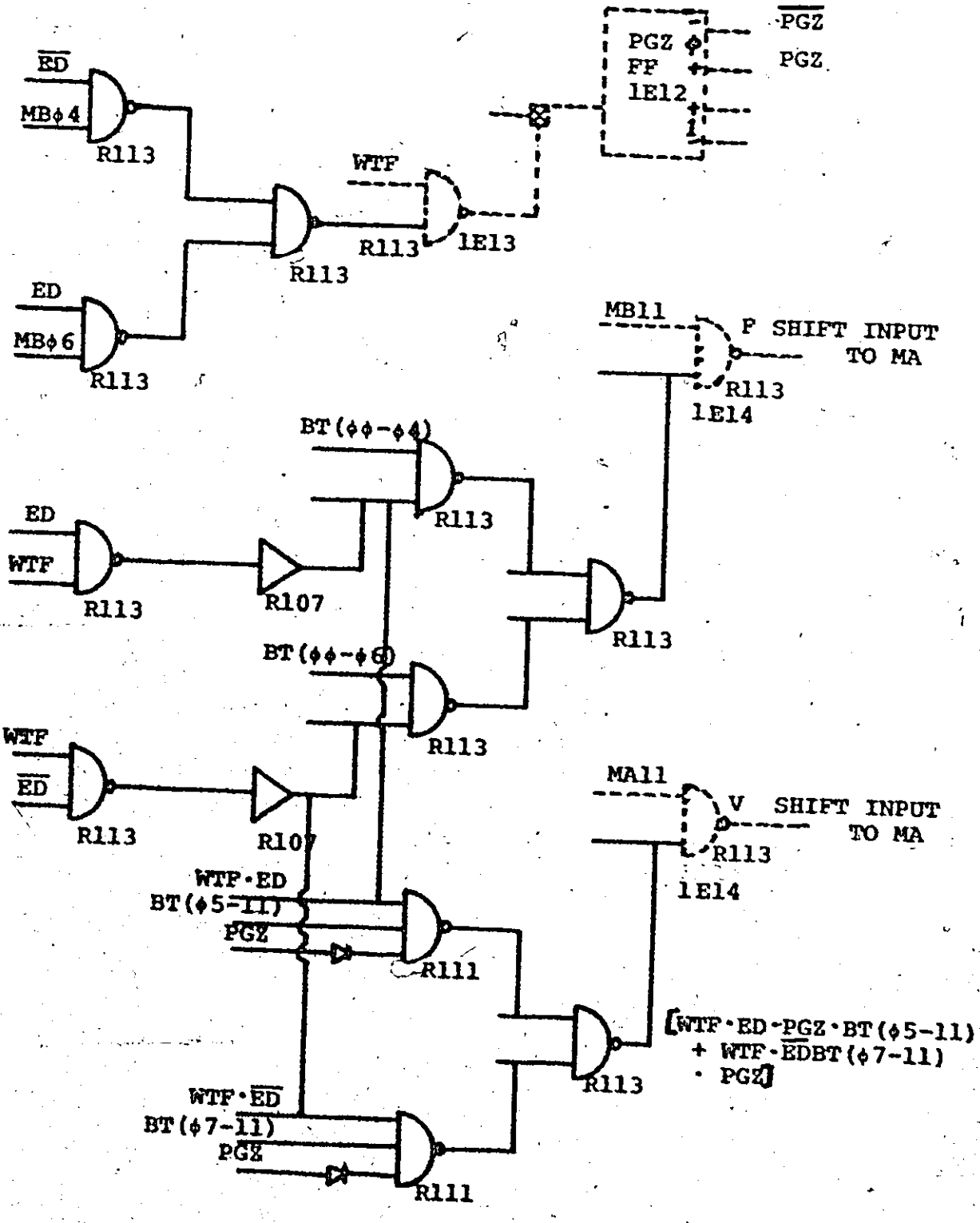


FIGURE 14: MODIFICATIONS TO MA

3.7 TIMING CIRCUITS

Following are the various small changes made to the timing circuits to produce the necessary controlling conditions for different processor word-times. These conditions are required to implement the timing diagram of figure 3 for the modified version.

3.7(a) Bit Timing

Bit-time levels are used to generate the time pulses and control the logic that transfers the address to MA after each instruction is fetched. The generation of bit-time levels from the T-states of time ring counter is shown on the top of figure 15. Originally only BT(ϕϕ-ϕ6) and BT(ϕ7-11) were required to control the page logic for addressing the whole memory. But in the educational mode, two extra bit-time levels - BT(ϕϕ-ϕ4) and BT(ϕ5-11) are required for the reduced page size. BT(ϕ5-11) is the direct output of the flip-flop T_4. This and other details of bit-timing are shown in the PDP-8/S drawing No. 11 attached in the appendix. BT(ϕϕ-ϕ4) is produced by the logic shown at the bottom of figure 15.

3.7(b) Inhibit Auto-Index

As shown in the PDP-8/S word-timing drawing in the appendix, the logic net at the left of AUTO-INDEX flip-flop (AI) grounded the enabling level for the AI set gate if the left half of MA was clear, bit 5 through 7 of MB were clear and MBϕ8 was 1. The appearance of this

T_0	T_1	T_2	T_3	T_4	T_5	T_6	
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	
1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	
1	1	ϕ	ϕ	ϕ	ϕ	ϕ	
1	1	1	ϕ	ϕ	ϕ	ϕ	
1	1	1	1	ϕ	ϕ	ϕ	
1	1	1	1	1	ϕ	ϕ	
1	1	1	1	1	1	ϕ	BT(ϕϕ-11)
1	1	1	1	1	1	1	
ϕ	1	1	1	1	1	1	
ϕ	ϕ	1	1	1	1	1	
ϕ	ϕ	ϕ	1	1	1	1	
ϕ	ϕ	ϕ	ϕ	1	1	1	
ϕ	ϕ	ϕ	ϕ	ϕ	1	1	BT12
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	1	BT13

$BT(\phi 5\text{-}11) = T_4$ $\quad BT(\phi\phi\text{-}\phi 4) = -T_4 \cdot -T_6$

$BT\phi\phi = -T\phi \cdot -T6$

$BT(\phi\phi\text{-}\phi 1) = -T1 \cdot -T6$

$BT(\phi\phi\text{-}\phi 6) = -T6$

$BT(\phi 7\text{-}11) = T4 \cdot T6$

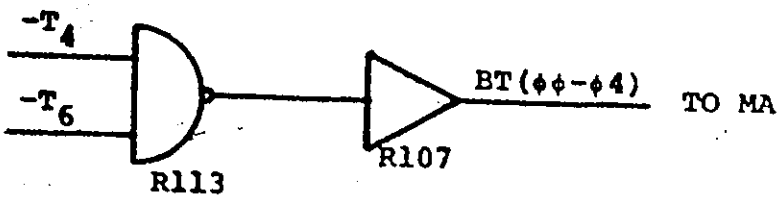


FIGURE 15: MODIFICATIONS TO THE BIT-TIMING

configuration in the Fetch time of an instruction that called for an indirect addressing indicated that one of the auto-indexing locations, 10_8-17_8, was being addressed. But for educational mode, the auto-indexing was inhibited because it has been replaced by indexing. To achieve this, an extra input ED has been added to the grounding logic as shown in figure 16.

3.7(c) Inhibit WTF to WTX

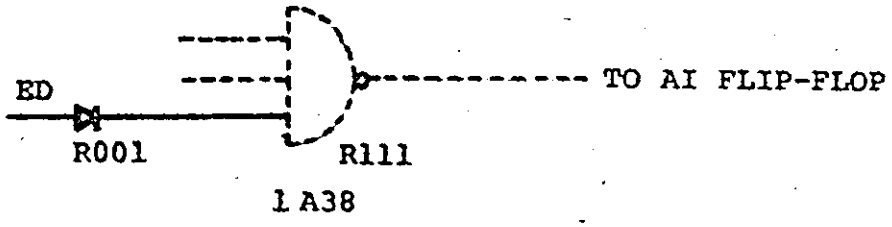
When an instruction calls for indirect addressing or indexing, at the end of the Fetch time, the processor should not enter execute time WTX. This is achieved, by adding the logic shown at the bottom of figure 16 to the existing controlling logic to the left of the WTX flip-flop.

3.7(d) Inhibit WTE

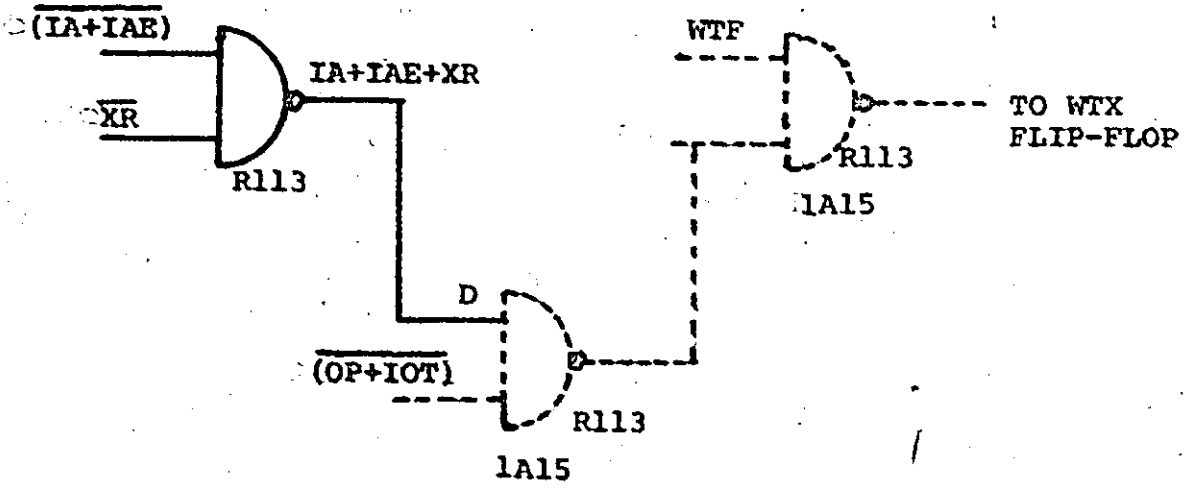
A13 pulse clears any word-time flip-flop that is set, and sets only the ones appropriate for the next word-time. At the end of the Fetch time, we do not want A13 to set WTE flip-flop when indexing is specified in an instruction. This is done by the logic net shown at the top of figure 17.

3.7(e) Inhibit WRITE in Memory

As shown in timing diagram of figure 3 for educational version, whenever indexing is desired alone or with indirect addressing, we want to inhibit the writing of the effective address in the memory. This has been

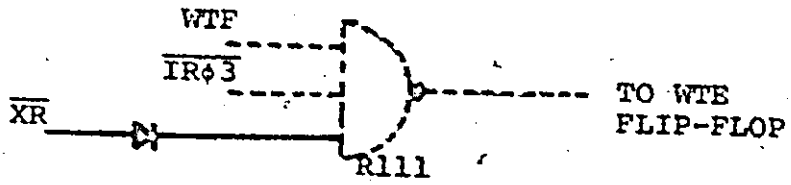


INHIBIT AUTO-INDEX

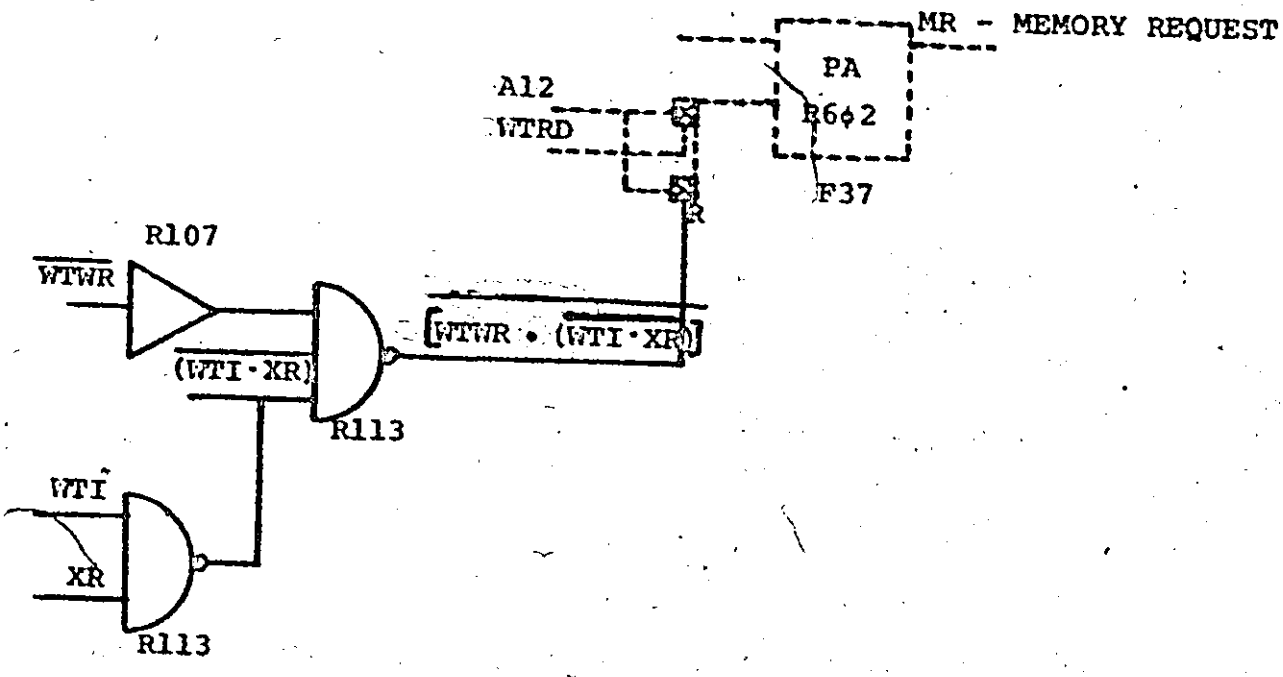


INHIBIT WTF TO WTX

FIGURE 16



INHIBIT WTF TO WTE



INHIBIT WRITE

FIGURE 17

implemented as shown in the logic net of figure 17. Whenever XR is present for the Indexing time WTI the memory request (MR) is inhibited.

3.7(f) WTF to WTI for Indexing

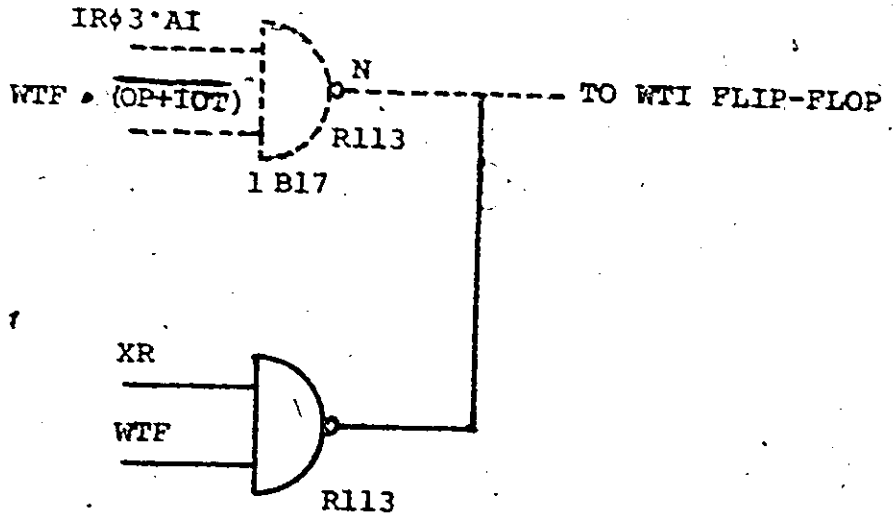
For an instruction in which indexing is specified, at the end of the Fetch time the processor must enter the Index time to modify the address. This is achieved by adding the logic net shown in figure 18 to the existing net that determines when the processor should enter the Index time.

3.7(g) Change WTF to WTD for Indirect-Addressing

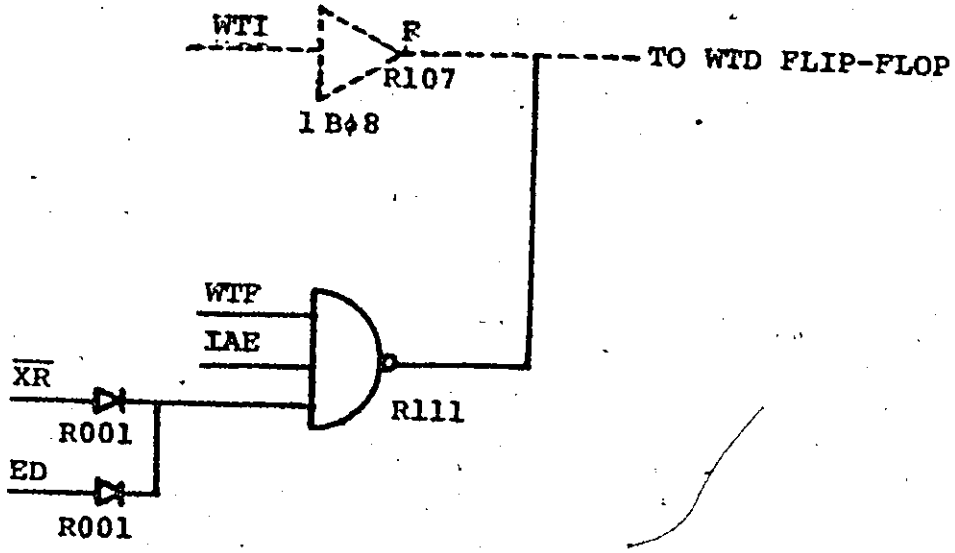
For an indirectly addressed instruction, the processor should enter the Defer time (WTD) to move the new address to MA and request a memory cycle to retrieve the operand, at the end of the Fetch time. In the educational version, this situation should exist when indirect addressing is desired but no indexing. This has been implemented as shown in figure 18. To the existing net which decides when the processor should enter the Defer time has been added the net shown.

3.7(h) Modification of Function WTRD

Function WTRD, controls the processor when during any word-time data is to be read from the memory. For the index register instructions LDX and ADX, at the end of the Fetch time or the Defer time, the operand is to be read from memory. This is achieved by ORing LDX and ADX



(a) CHANGE WTF TO WTI FOR INDEXING



(b) CHANGE WTF TO WTD FOR INDIRECT ADDRESSING

FIGURE 18

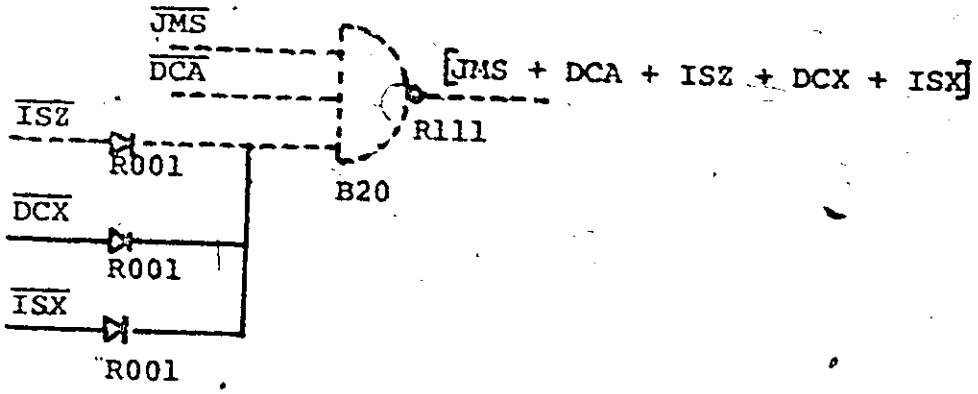
as shown in figure 19 to the function (AND + TAD + ISZ), thereby modifying WTRD to:

$$WTRD = WTF\ [-(OP + IOT) \cdot IR\phi 3 + AND + TAD + ISZ + LDX + ADX] + WTD\ (AND + TAD + ISZ + LDX + ADX) + WTE + EX$$

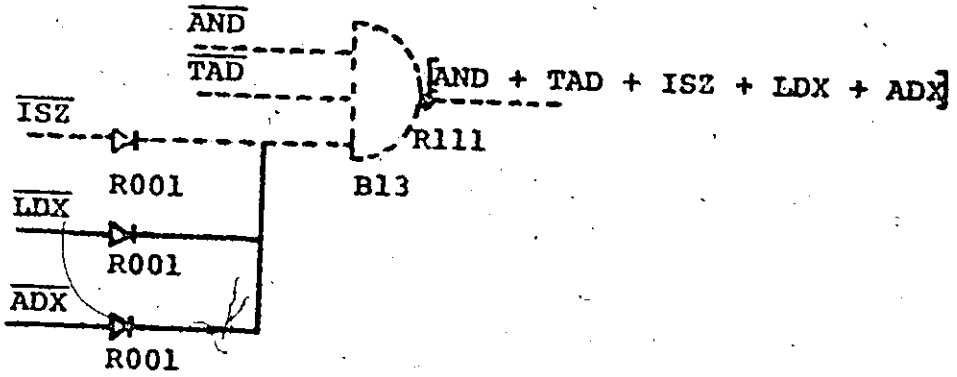
3.7(i) Modification of Function WTWR

Similarly, WTWR controls the processor whenever for any word-time, data is to be deposited in the memory. For the added index register instruction, this needs to be done for DCX and ISX instructions during WTX. This is implemented by ORing DCX and ISX with ISZ + DCA + JMS as shown in figure 19. As a result, WTWR is modified to:

$$WTWR = WTI + WTX\ (ISZ + DCA + JMS + DCX + ISX) + WTB + DC$$



MODIFICATION TO WTWR



MODIFICATION TO WTRD

FIGURE 19

Chapter IV

SYSTEM PERFORMANCE TEST

A number of simple programs were run, both in the educational and the normal modes, to test the modified version. A typical example of the test program is shown in figure 20. Here numbers stored in sequential locations from 25_8-34_8 are added, and after various manipulations showing the different features of the modified PDP-8/S, the correct result was displayed in the accumulator.

LOCN.	SYMB.	OCTAL
1	CLA	76ϕϕ
2	LDX 21	ϕ421
3	TAD X 35	1135
4	ISX 1	24ϕ1
5	JMP 3	5ϕϕ3
6	LDX 22	ϕ422
7	DCX 77	3477
1ϕ	LDX 2ϕ	ϕ42ϕ
11	ADX 21	1421
12	DCA I X 77	3377
13	TAD I 24	1224
14	HLT	74ϕ2
2ϕ	ϕ	
21	777ϕ	
22	33	
23	ϕ	
24	23	
25	1	
26	2	
27	3	
3ϕ	4	
31	5	
32	6	
33	7	
34	1ϕ	
35	35	

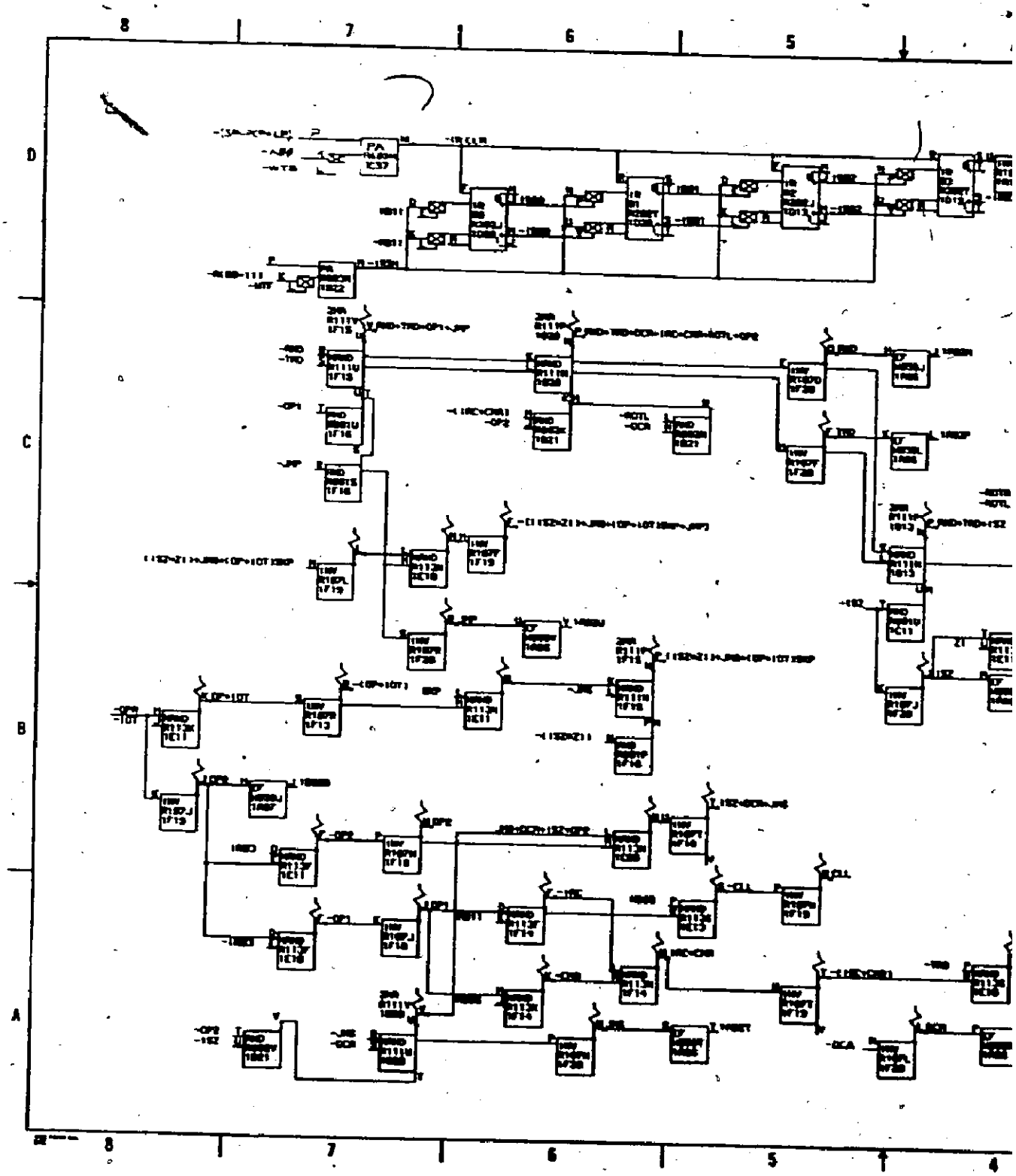
FIGURE 20: SAMPLE TEST PROGRAM

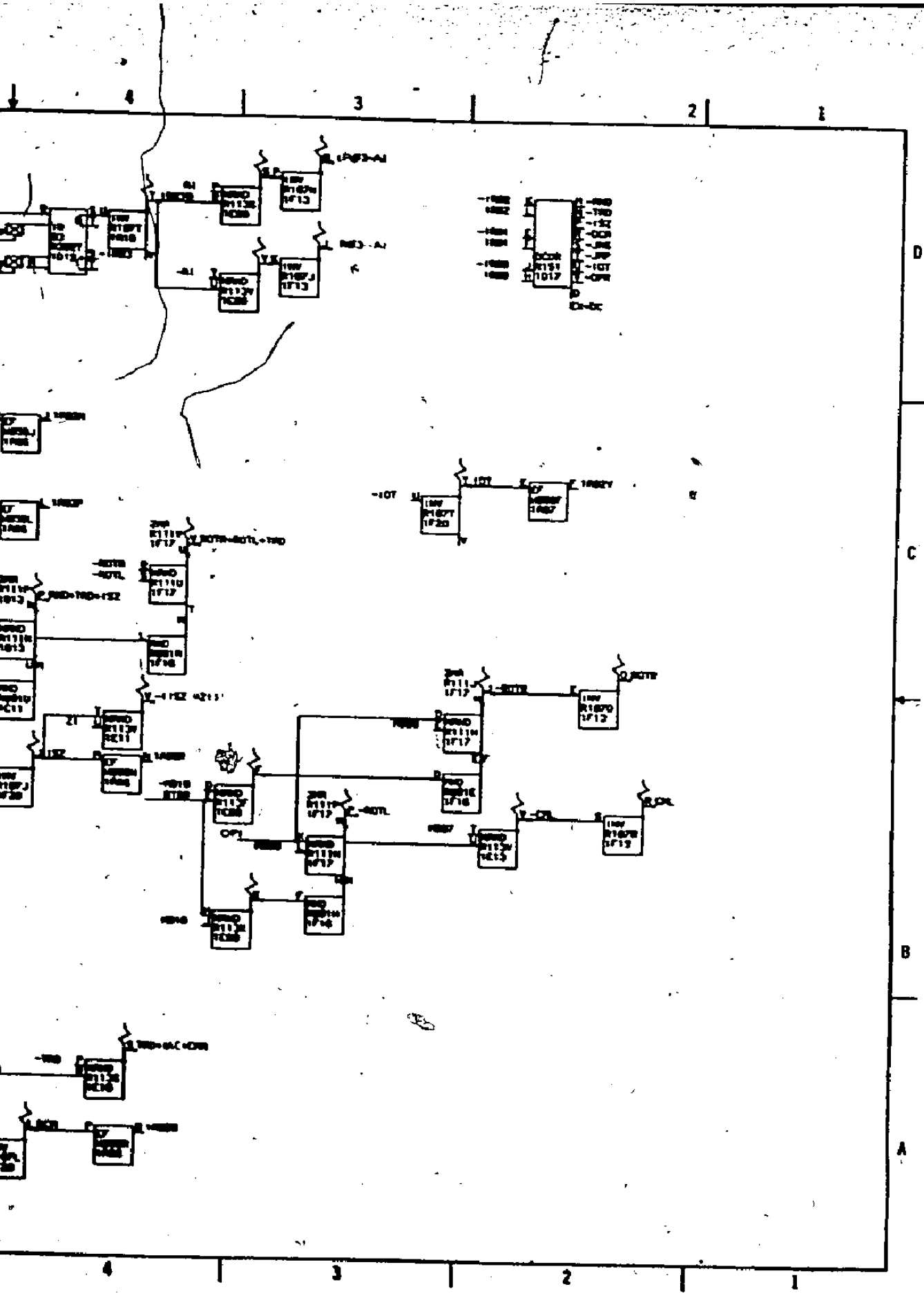
Chapter V

CONCLUSIONS

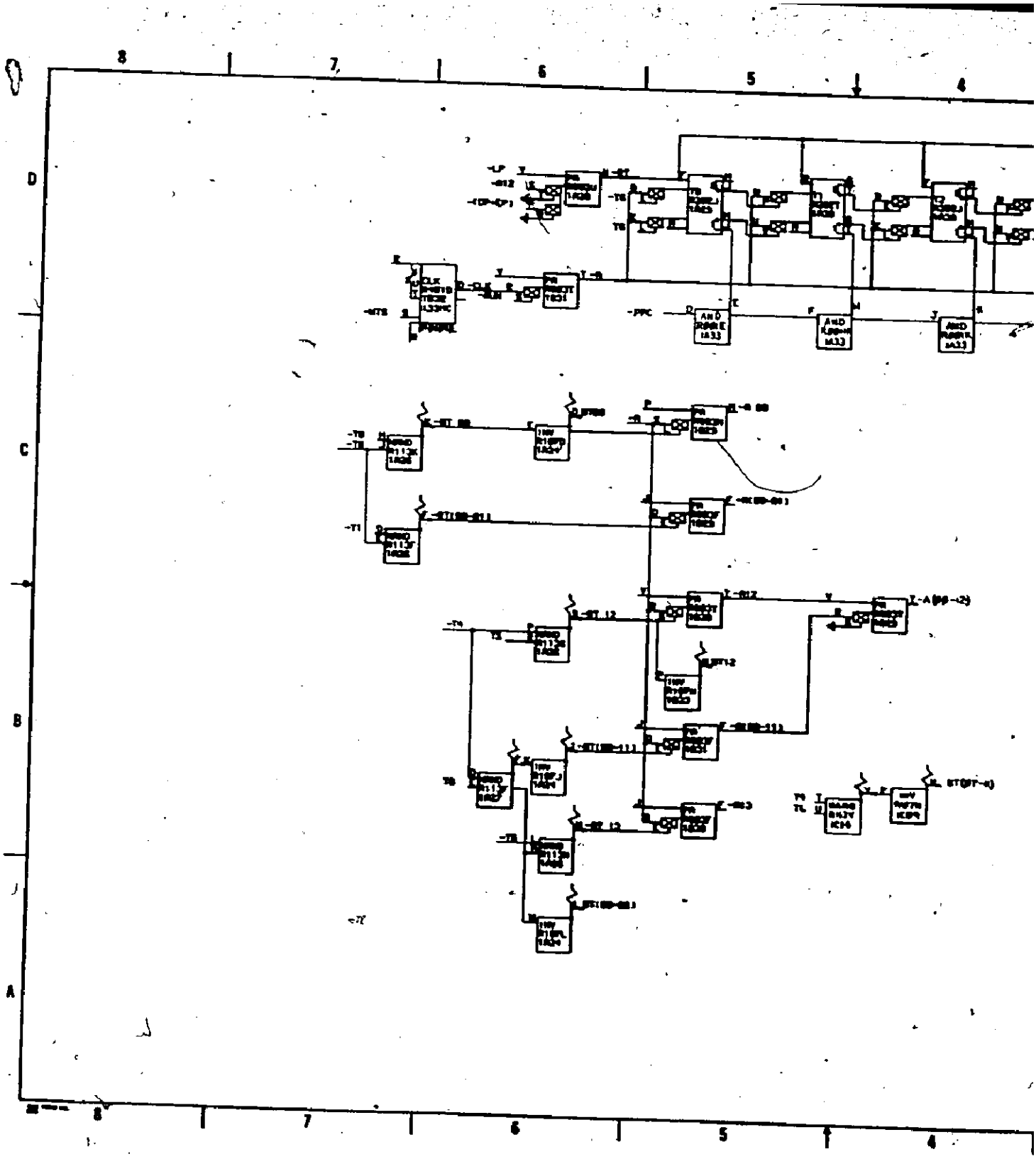
The educational computer was successfully designed and implemented as correct results were obtained for different test programs, both for the educational and the normal modes. The index register added is of full length of 12-bits instead of five bits (the address portion of the instruction) because it was convenient from a design point of view. This full length index register can serve as a general-purpose register for temporary storage because it has all the desired facilities of a register, i.e., loading, clearing and shifting.

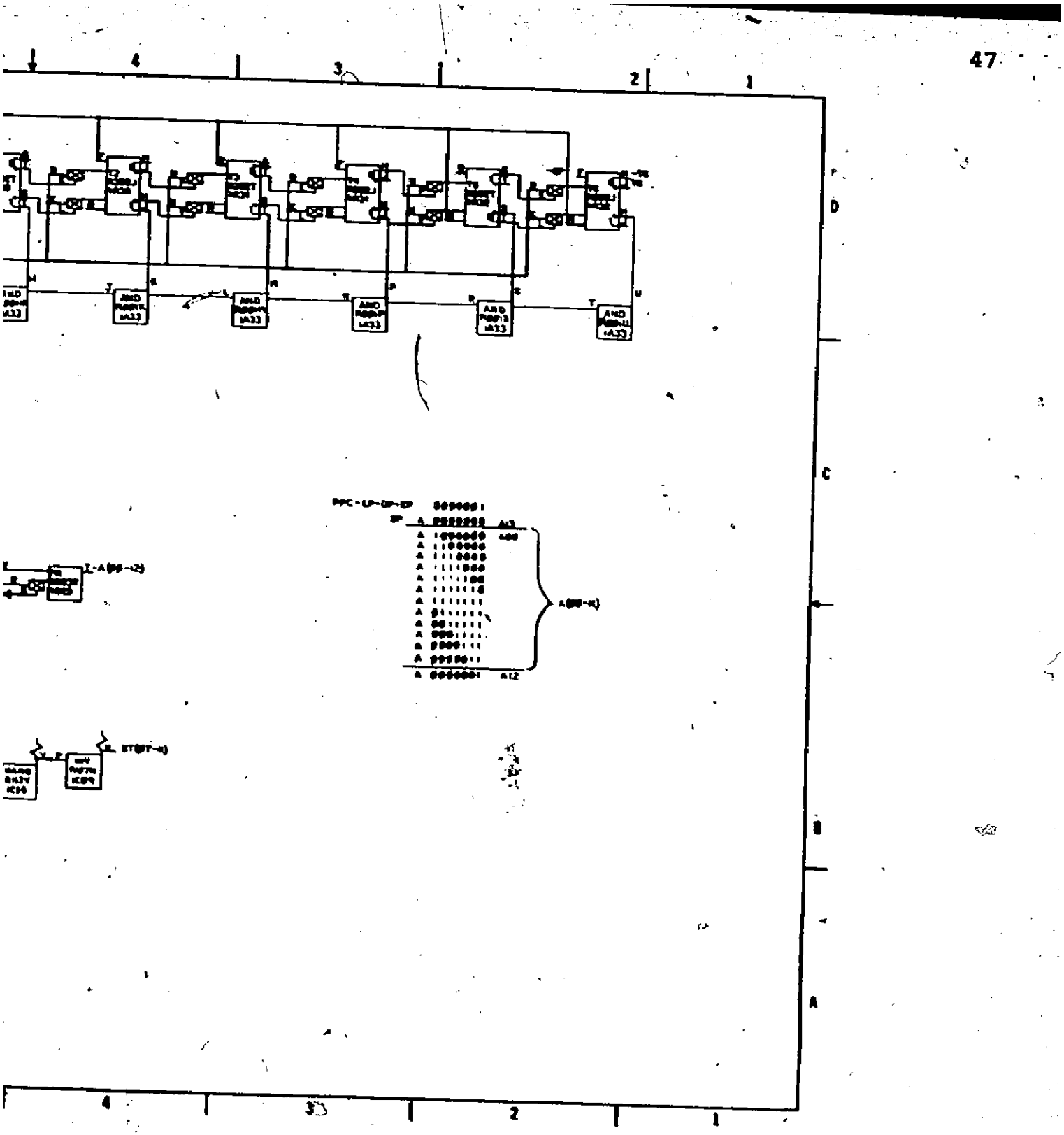
APPENDIX


8
7
6
5
4
D
C
B
A

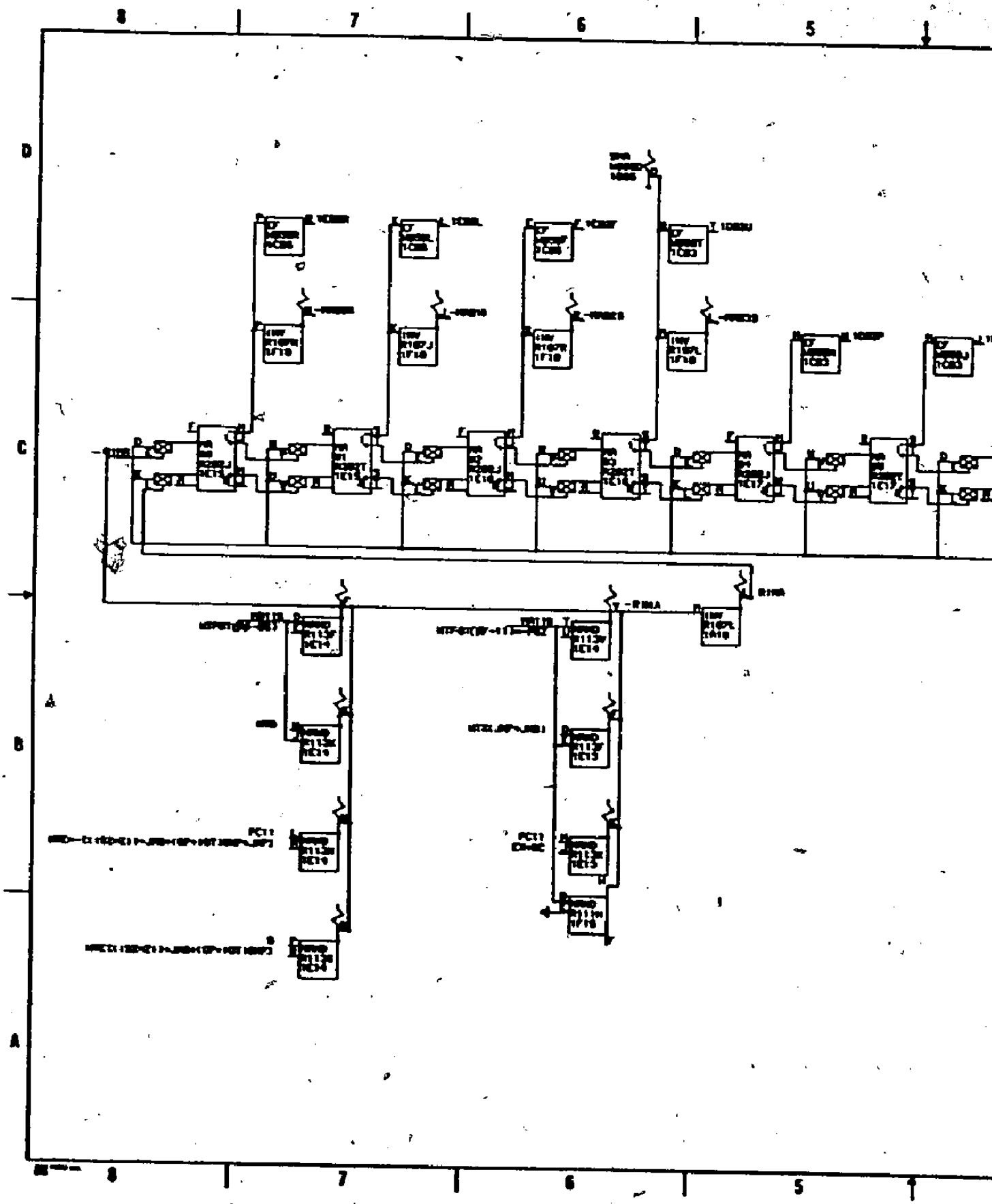


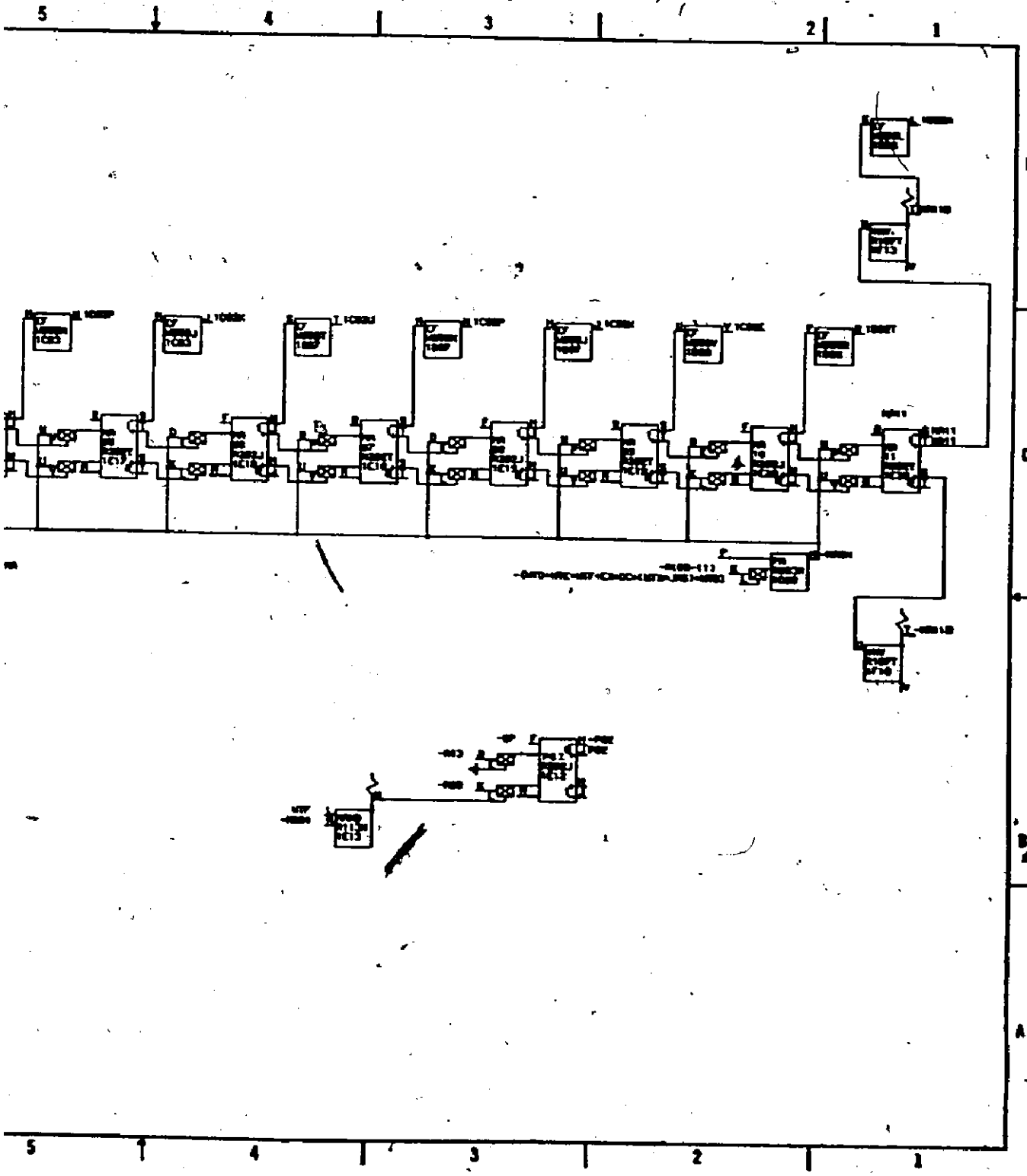
Instruction Register D-BS-85-0-16


8
7
6
5
4
D
C
B
A

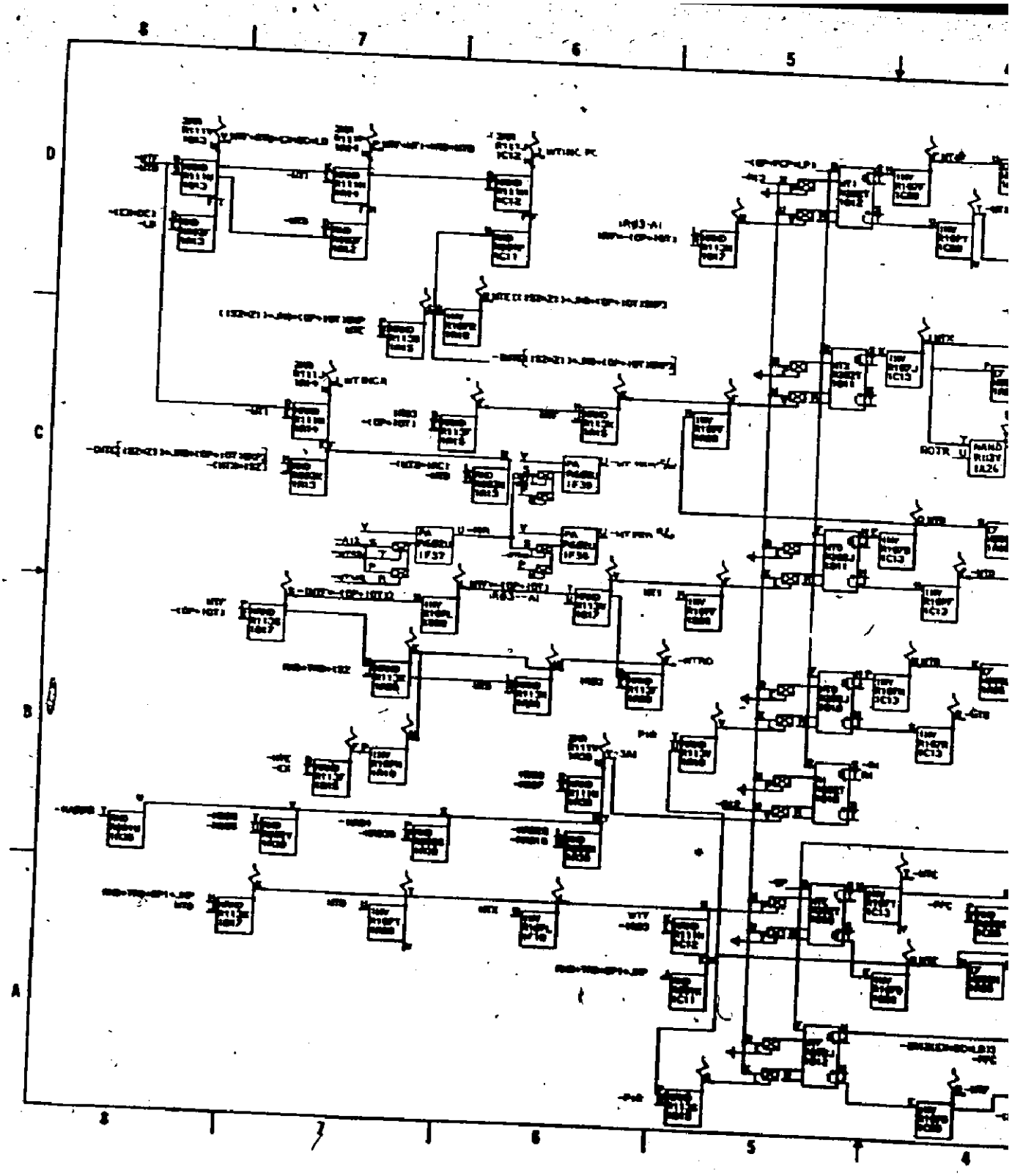


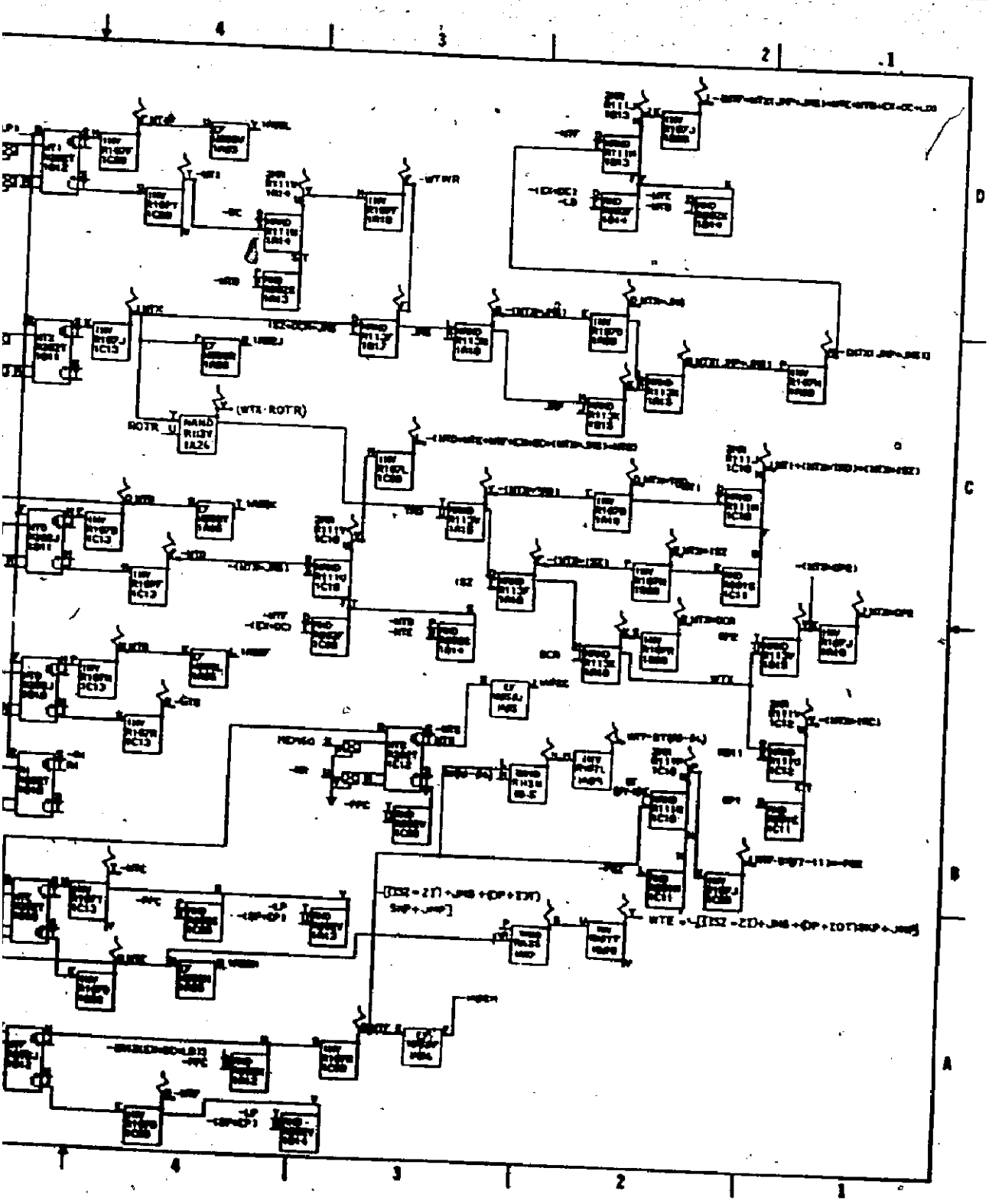
Bit Timing D-BS-BS-0-11



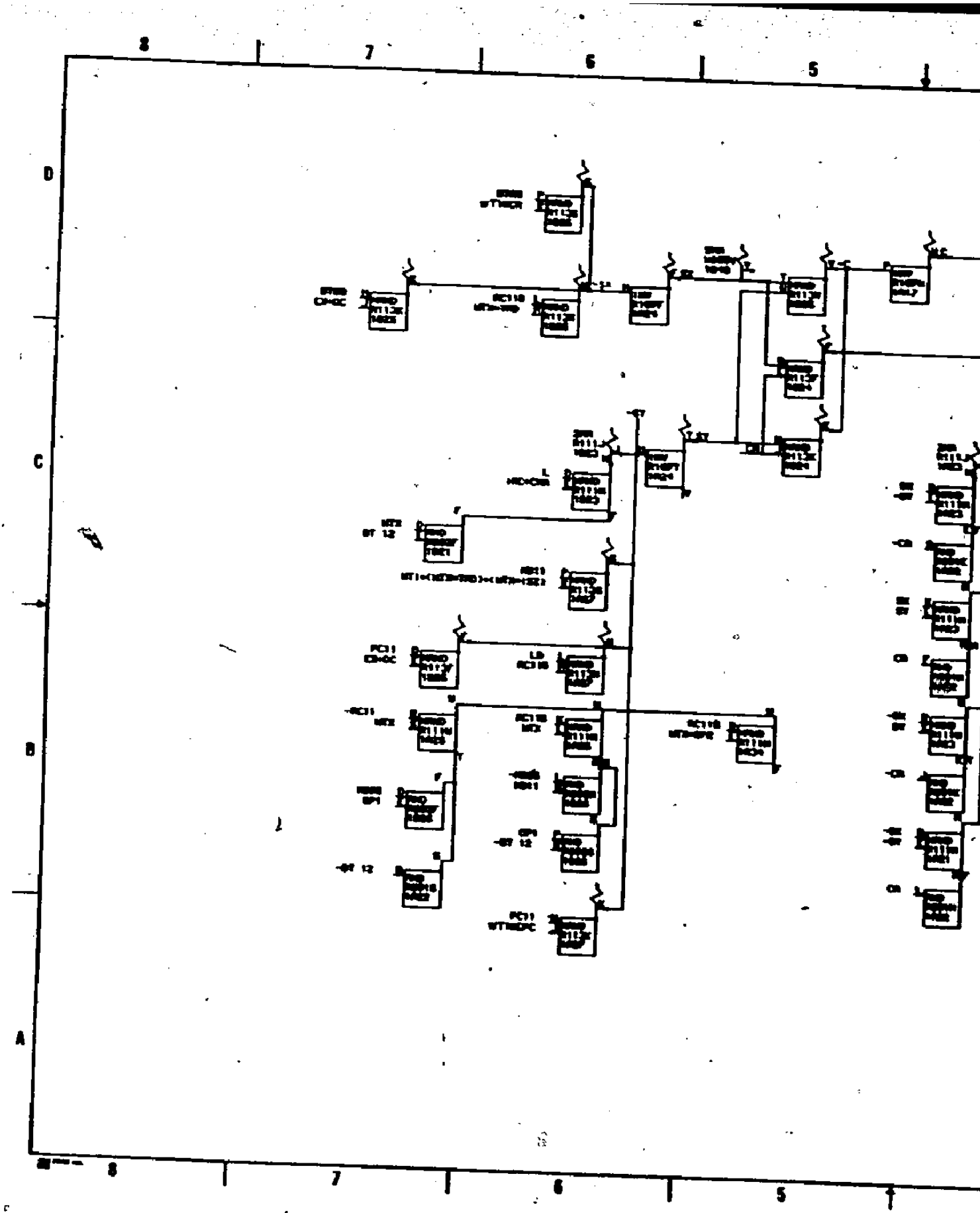


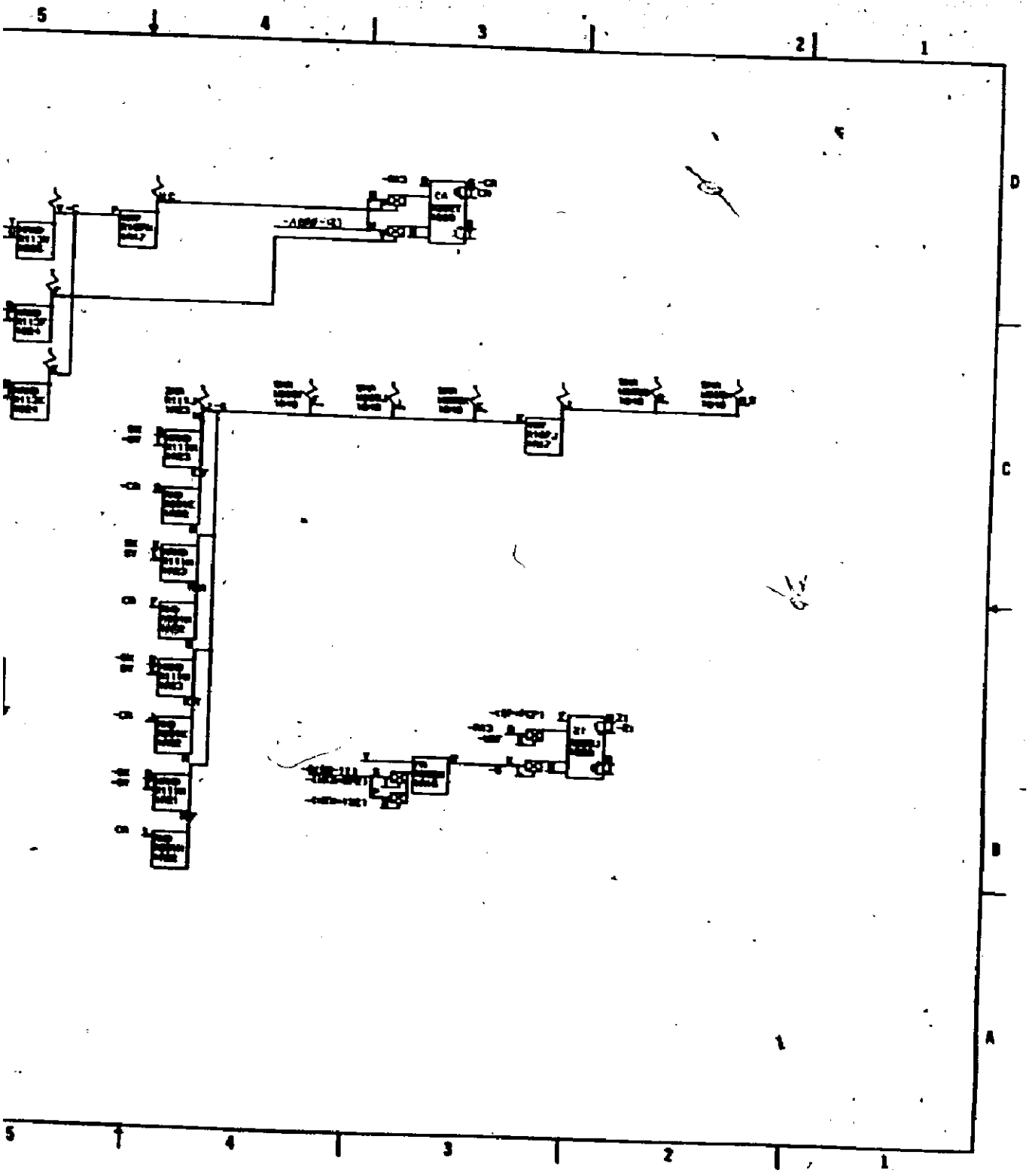
Memory Address D-8S-8S-0-12

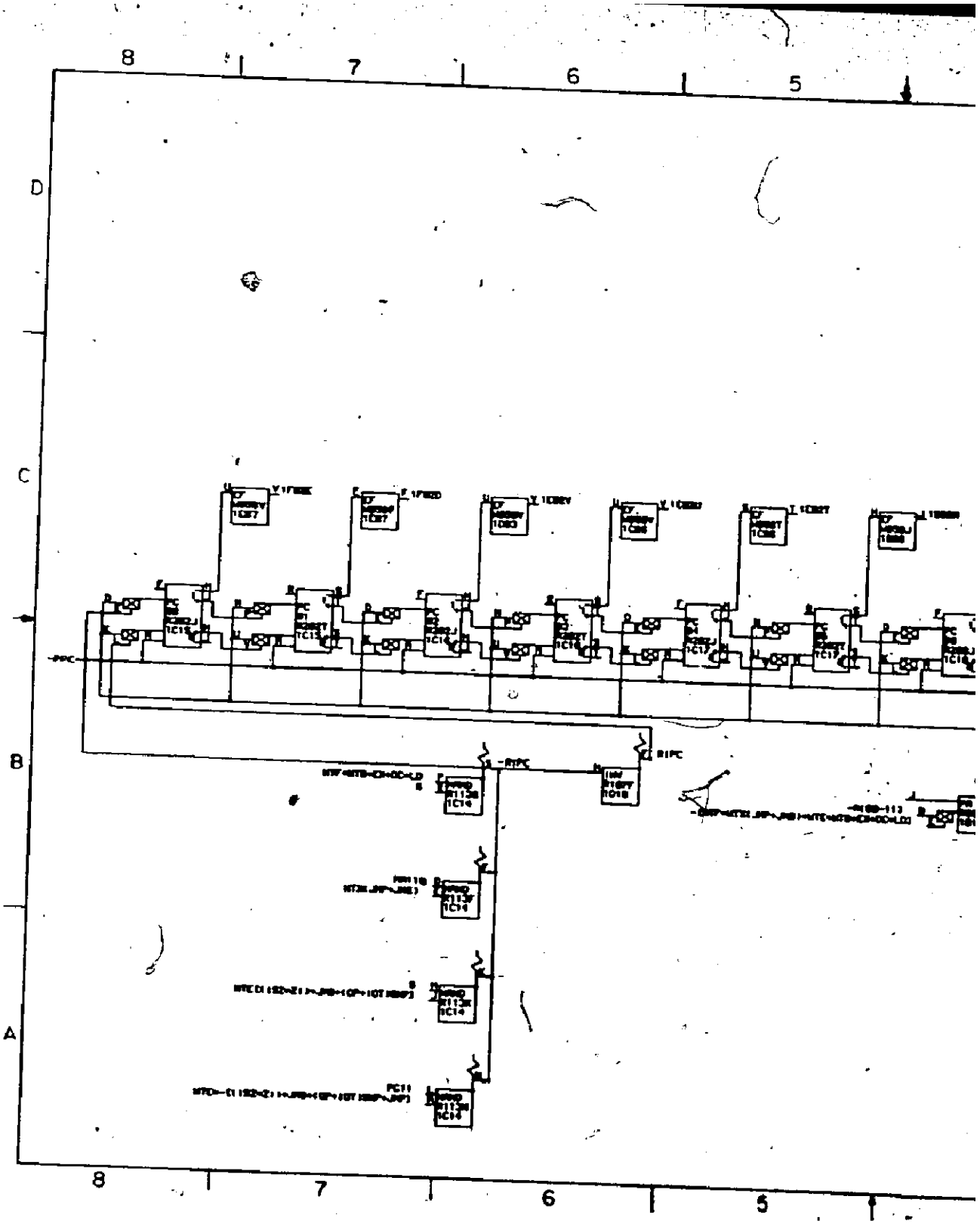


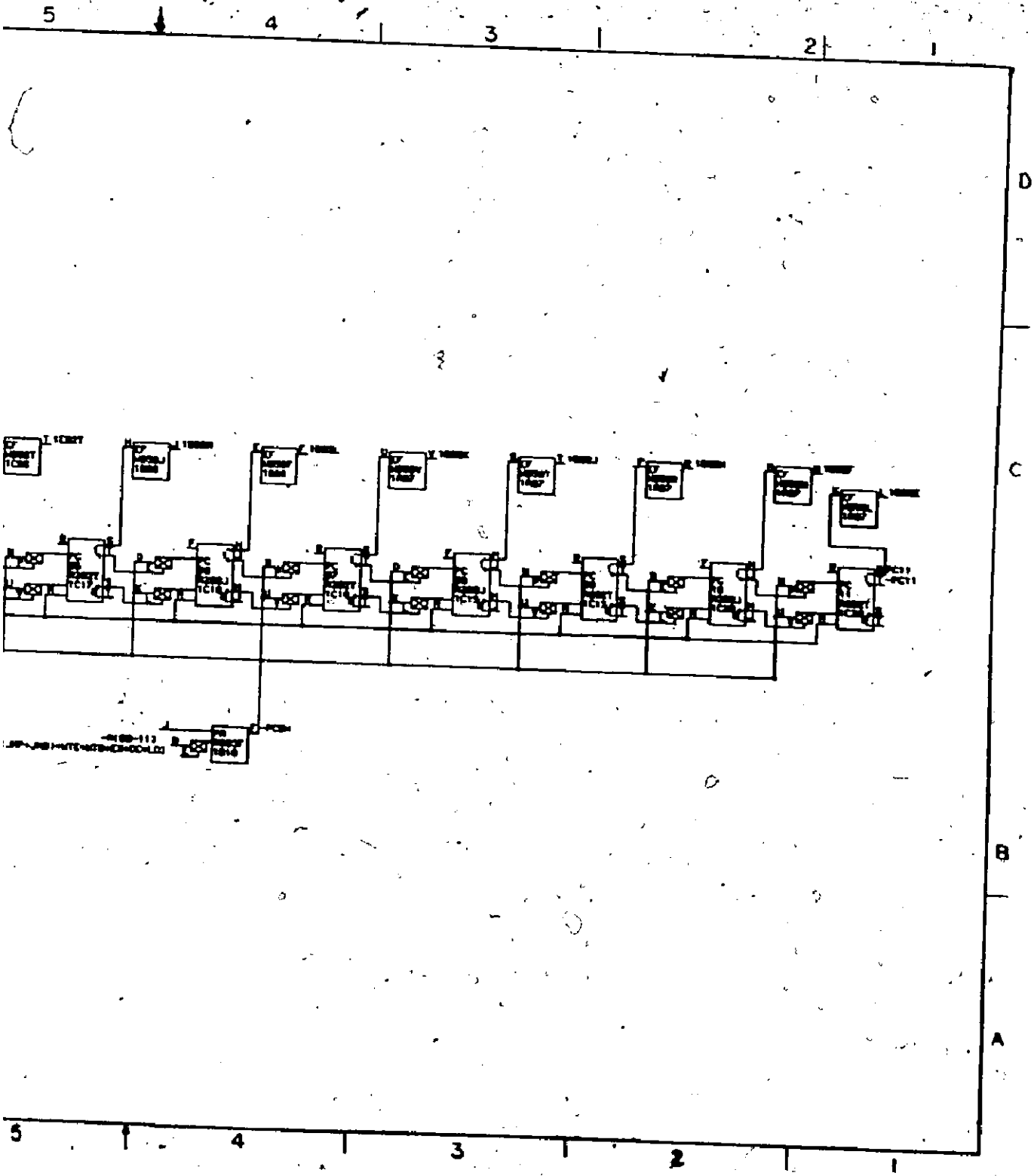


Word Timing D-BS-BS-0-13
A-15




5
4
3
2
1
D
C
B
A


8
7
6
5
D
C
B
A



Program Counter D-85-85-0-19

REFERENCES

1. Bartee, Thomas C. DIGITAL COMPUTER FUNDAMENTALS; McGraw-Hill, Toronto, Ontario (1960).

2. Flores, Ivan. DIGITAL COMPUTER LOGIC AND FUNDAMENTALS; Prentice-Hall, Toronto, Ontario (1962).

3. Staff of Digital Equipment Corporation; PDP-8/S MAINTAINENCE MANUAL, Digital Equipment Corporation, Maynard, Massachusetts (1962).

4. Staff of Digital Equipment Corporation; LOGIC HANDBOOK; Digital Equipment Corporation, Maynard, Massachusetts (1967).

5. Staff of Digital Equipment Corporation; INTRODUCTION TO PROGRAMMING FOR PDP-8 SERIES; Digital Equipment Corporation, Maynard, Massachusetts (1970).

6. Thomas, P.A.V. and H.S. Sodhi. PDP-8/S WITH INDEX REGISTER; Proceedings of the 7th Canadian DECUS Symposium, Ottawa, Ontario (March 1974).

VITA AUCTORIS

1947 Born on July 6th in Narwana, Haryana, India.

1969 Graduated from Punjabi University, Patiala, Punjab, India.

1974 Candidate for the degree of M.A.Sc. in Electrical Engineering at the University of Windsor.