1-1-1964

The logic design of a digital control computer.

Richard E. Shiner
University of Windsor

THE LOGIC DESIGN
OF A
DIGITAL CONTROL COMPUTER

BY

RICHARD E. SHINER

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
University of Windsor

Windsor, Ontario
1964

Approved by

P.A.V. Thomas

Dr. P. A. V. Thomas
Associate Professor

Dr. S. N. Kalra
Associate Professor

John N. Deck

Dr. J. N. Deck
Associate Professor

ABSTRACT

This thesis presents the logic design of a digital control computer presently being constructed at the University of Windsor.

The computer incorporates a pluggable program board, a magnetostrictive delay line memory, several working registers, and an arithmetic unit capable of addition, subtraction, multiplication, and division. The computer has the ability to modify data locations and to execute program jumps. Input facilities accept data from a keyboard and also from an external rotating shaft position digitizer. Output facilities are a set of lamps which indicate the contents of an accumulator register and also an output order which can be used to control an external system.

This thesis includes a complete set of logic diagrams of the computer, which utilizes NAND logic.

ACKNOWLEDGEMENTS

The author wishes to express sincere appreciation to Dr. P. A. V. Thomas, who supervised this work, for his helpful guidance and advice.

Acknowledgement is also due the National Research Council of Canada for financial assistance provided for this project.

TABLE OF CONTENTS

I. INTRODUCTION

The Electrical Engineering Department at the University of Windsor began, in June 1963, the study of a digital control system[1.] which requires an on-line digital computer. The exact nature of the problems and programs to be solved by the digital computer were not yet known. It was assumed that, as the research on the control system proceeded, new techniques of controlling would be found. It was decided, therefore, to develop a DIGITAL CONTROL COMPUTER and to incorporate into this computer the desired flexibility. This flexibility would include an extensive list of computer instructions, a suitable means of data storage, and special input and output facilities. This thesis presents the logic design of the digital control computer.

II. SYSTEM DESIGN

A concept of the design of the computing system is given.

2.1 The Number Format

The computer works in the binary number system using the two digits, ZERO and ONE. These two digits are represented in the machine by two voltage levels, 0 volts and -6 volts, respectively. The machine data word length is 12 bits long. The signed-2's complement representation is used for negative numbers. The sign of the binary number is denoted by the most significant digit called the sign bit. The 0 and 1 of the sign bit are chosen to represent, respectively, the positive and negative signs. These are taken to mean 0 and -1. The other digits are called the number digits and they have weights of 2^{-1}, 2^{-2}, 2^{-3},...2^{-11}. Thus, for example

$$0.01100000000 = 3/8$$

$$1.10100000000 = -3/8$$

Notice that a decimal place is used to separate the sign bit from the number bits. This decimal place does not, however, appear in the machine. The last number may be thought of as $-1 + 5/8 = -3/8$.

This number system limits the number range to fractions. Furthermore, since one of the external input devices is limited to 10 digits (excluding sign digit), the following range of numbers is adopted:

$$0.00000000000 \text{ to } 0.01111111111 \quad (\text{i.e. } 0 \leq x < 1/2)$$

and

$$1.10000000000 \text{ to } 1.11111111111 \quad (\text{i.e. } -1/2 \leq x < 0).$$

This system will give an accuracy of one part in 1024. In practice, all numbers will be reduced to fractions by a suitable scaling factor.

2.2 The Instruction Format

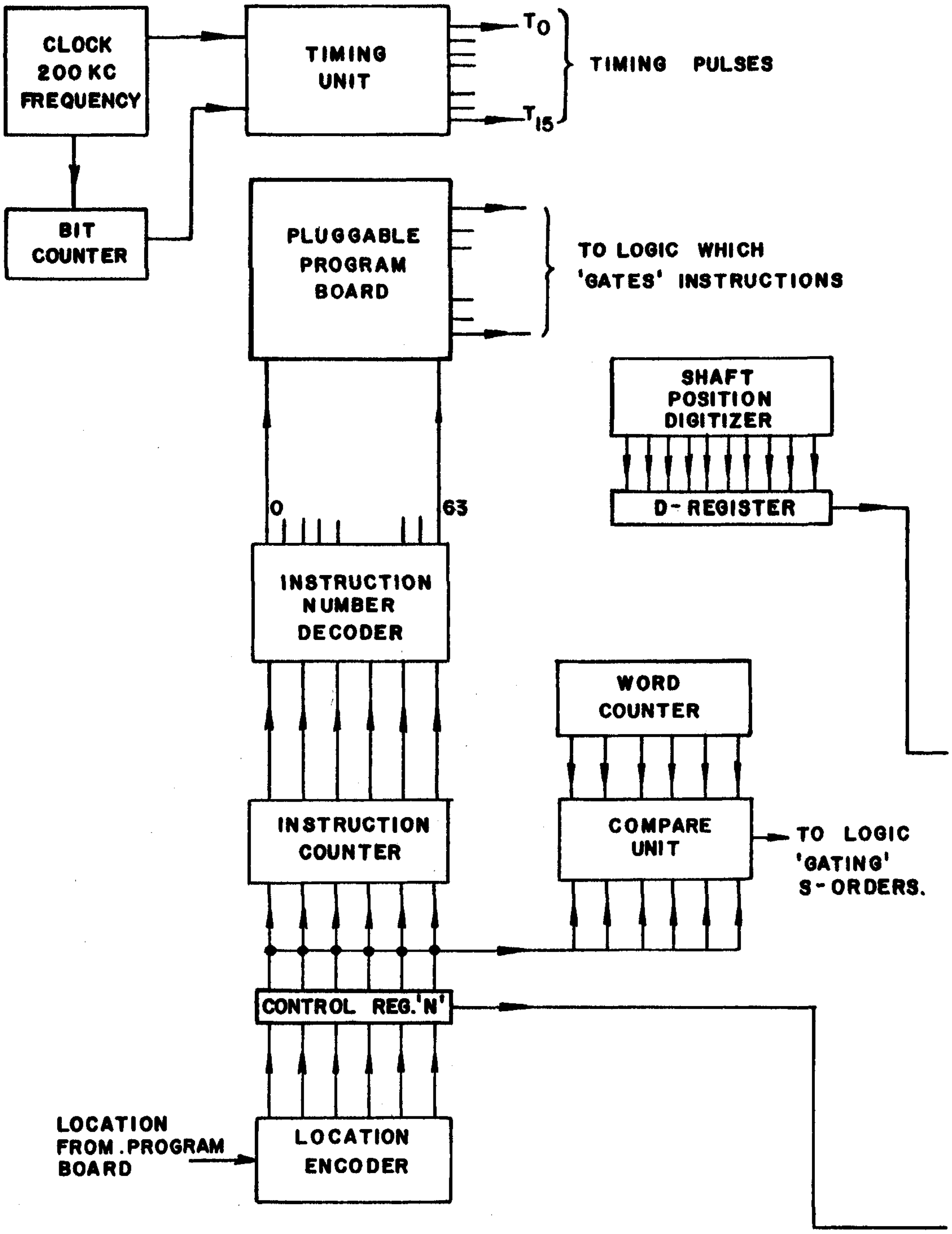
Instructions to be executed by the computer are divided into four parts. The first part specifies the particular order being called for, the second part gives the location number of the stored data, the third part specifies any required register, and the fourth part states whether modification of the data location number takes place.

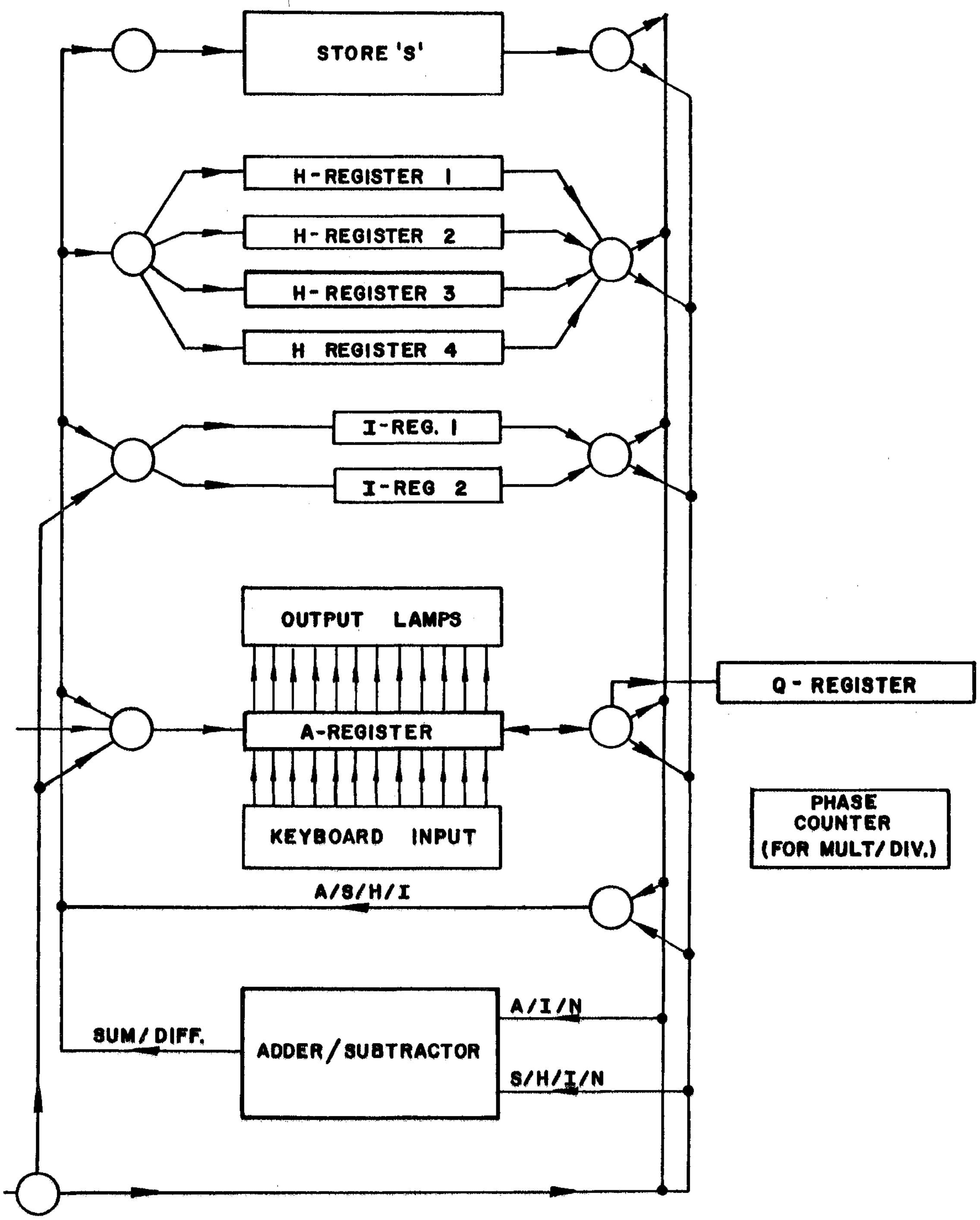
A complete description of how instructions are entered into the computer is given in section 2.11 below.

2.3 Computer Organization

A block diagram of the entire computer is presented in fig. 1. This diagram shows the organization of the computer. It presents the various registers and counters and indicates the general flow of information.

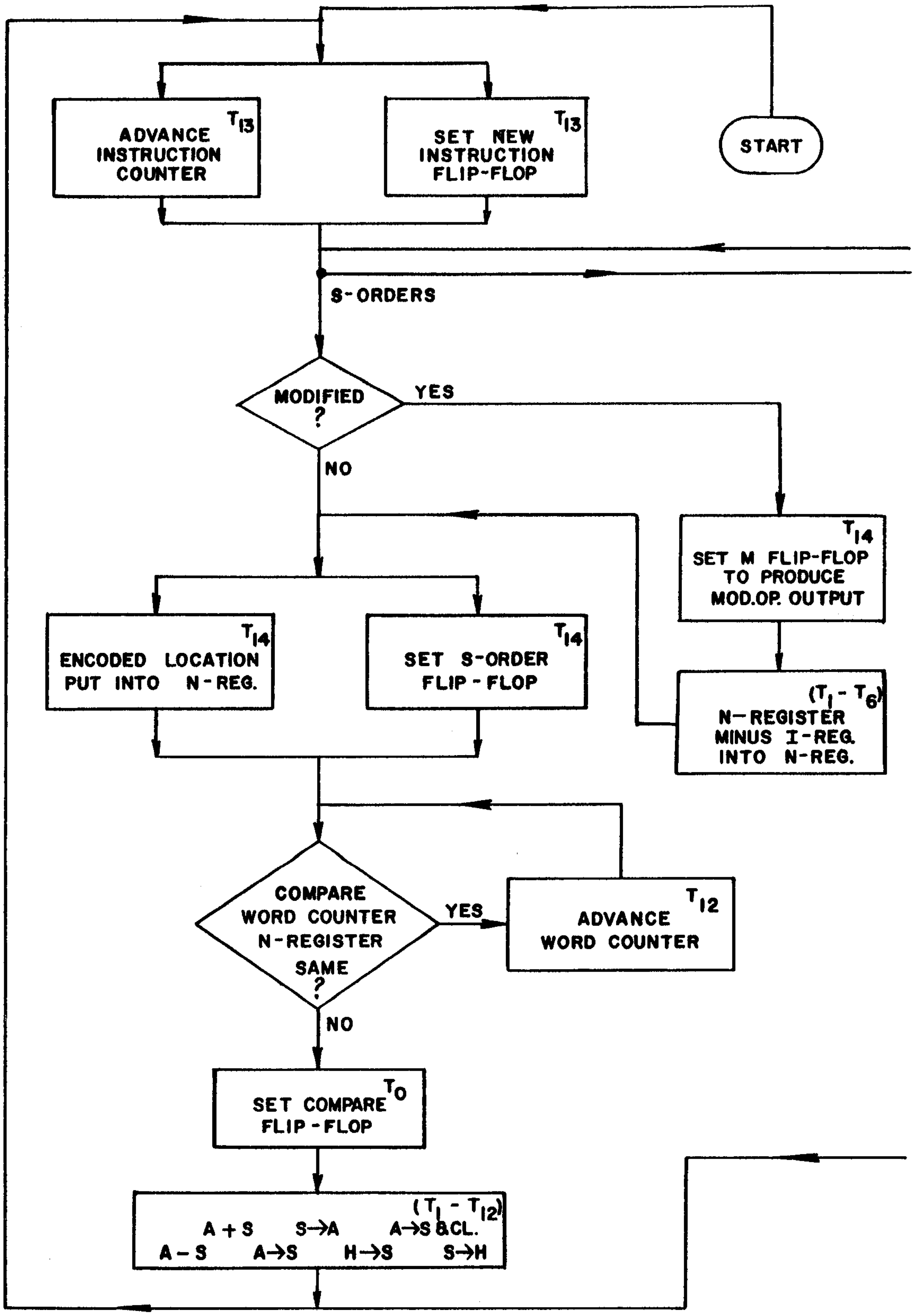
A time diagram for the computer is presented in fig.2. This diagram shows the sequence of operations for each type of instruction.

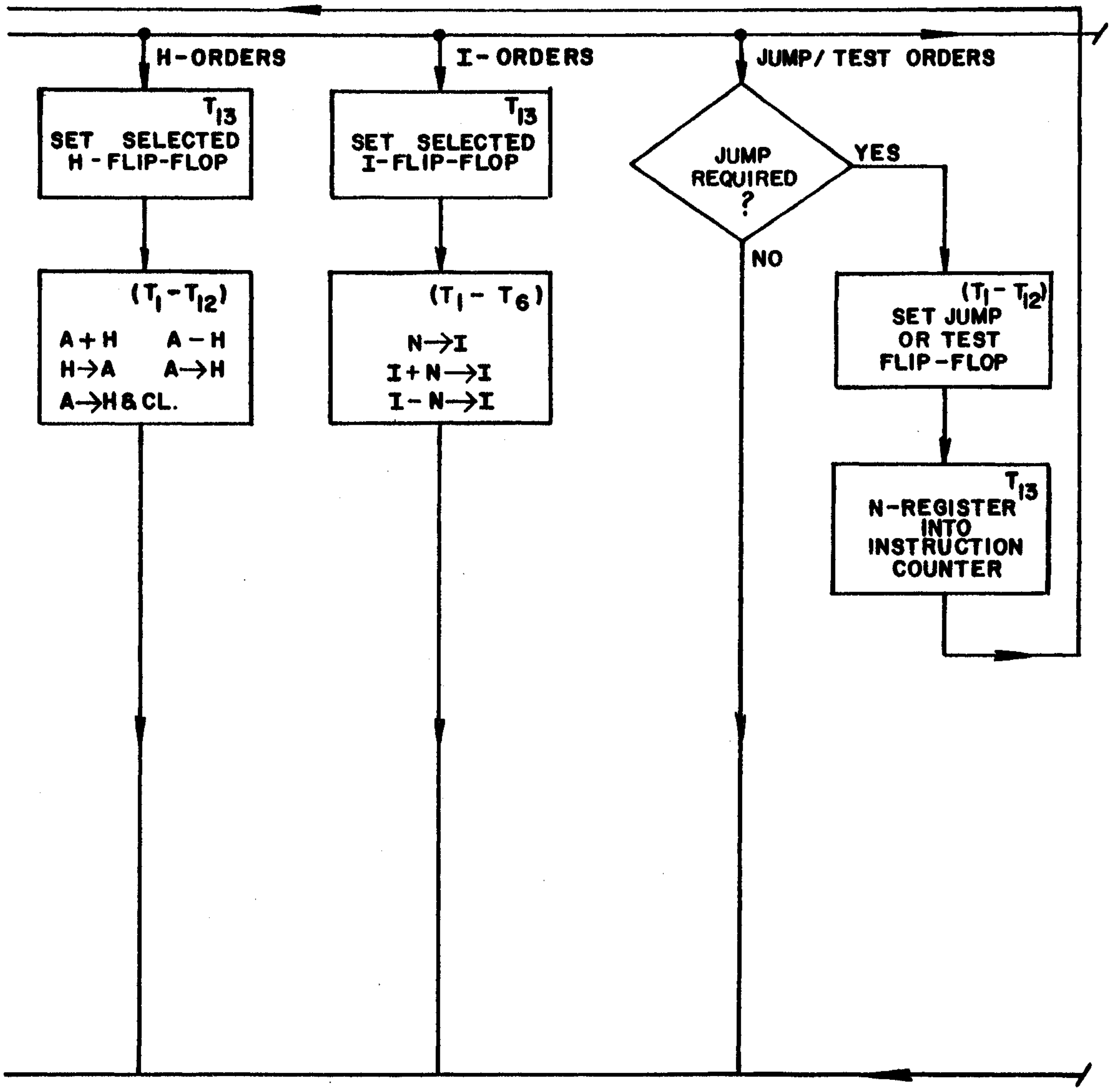

CLOCK 200 KC FREQUENCY
TIMING UNIT
T_0
T_{15}
TIMING PULSES
BIT COUNTER
PLUGGABLE PROGRAM BOARD
TO LOGIC WHICH 'GATES' INSTRUCTIONS
SHAFT POSITION DIGITIZER
D-REGISTER
0
63
INSTRUCTION NUMBER DECODER
WORD COUNTER
INSTRUCTION COUNTER
COMPARE UNIT
TO LOGIC 'GATING' S-ORDERS.
CONTROL REG.'N'
LOCATION FROM.PROGRAM BOARD
LOCATION ENCODER



BLOCK DIAGRAM OF COMPUTER

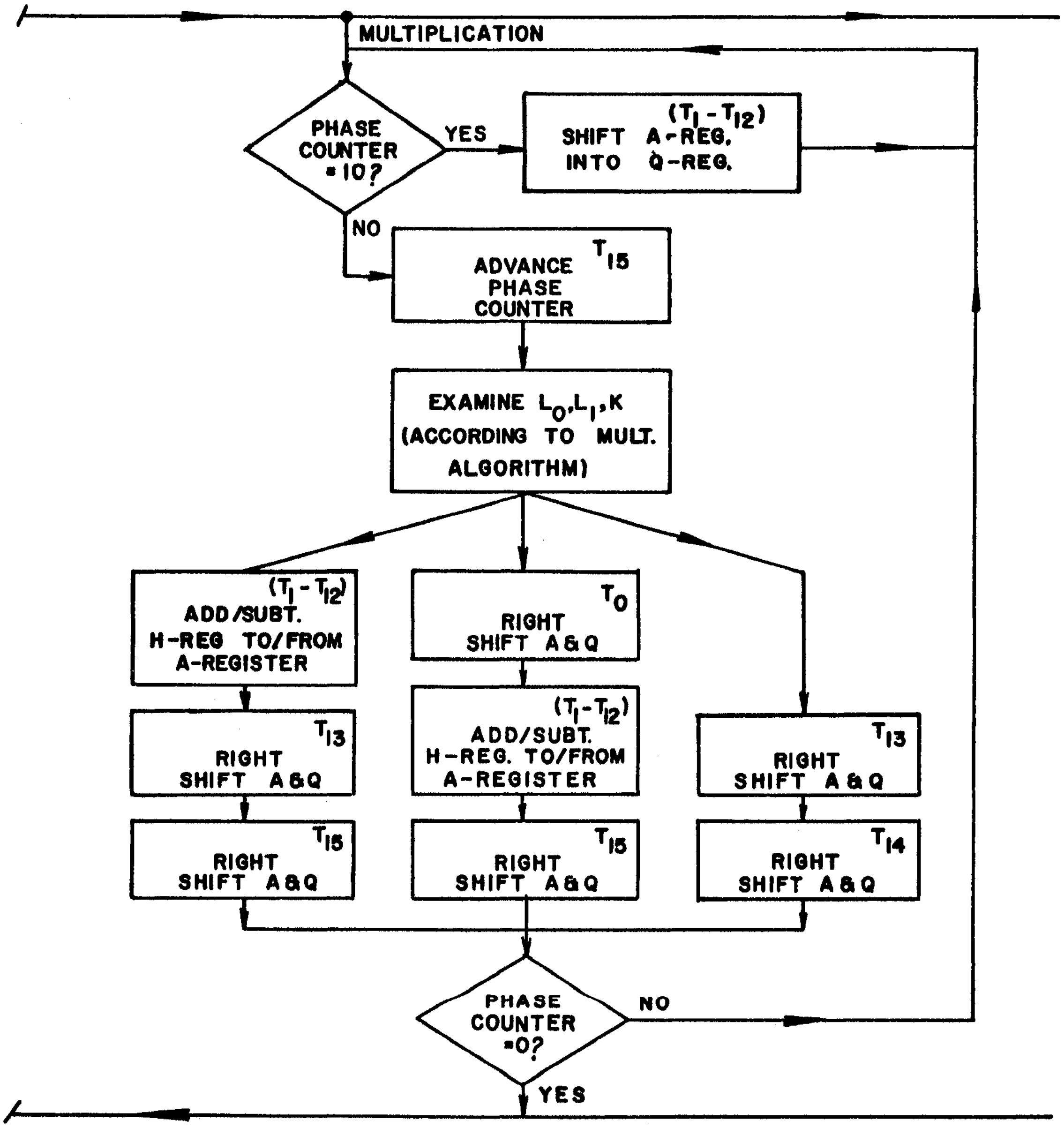
fig. I.

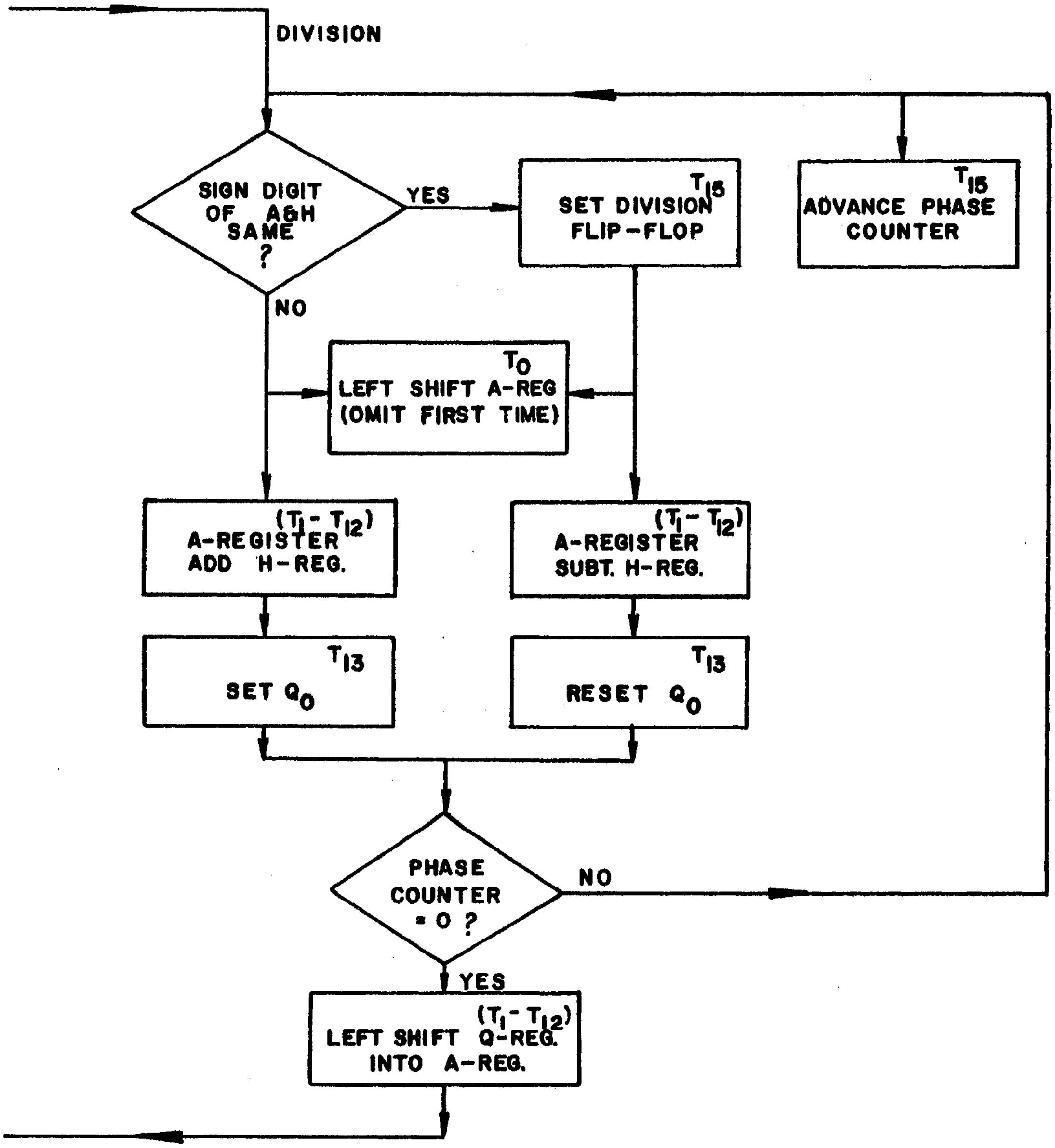

ADVANCE INSTRUCTION COUNTER
T_{13}
SET NEW INSTRUCTION FLIP-FLOP
T_{13}
START
S-ORDERS
MODIFIED ?
YES
NO
SET M FLIP-FLOP TO PRODUCE MOD.OP. OUTPUT
T_{14}
ENCODED LOCATION PUT INTO N-REG.
T_{14}
SET S-ORDER FLIP-FLOP
T_{14}
N-REGISTER MINUS I-REG. INTO N-REG.
(T_1 - T_6)
COMPARE WORD COUNTER N-REGISTER SAME ?
YES
ADVANCE WORD COUNTER
T_{12}
NO
SET COMPARE FLIP-FLOP
T_0
(T_1 - T_{12})
A + S
S→A
A→S &CL.
A - S
A→S
H→S
S→H



TIME DIAGRAM

fig. 2.


MULTIPLICATION
PHASE COUNTER =10?
YES
NO
(T1 - T12) SHIFT A-REG. INTO Q-REG.
T15 ADVANCE PHASE COUNTER
EXAMINE L0,L1,K (ACCORDING TO MULT. ALGORITHM)
(T1 - T12) ADD/SUBT. H-REG TO/FROM A-REGISTER
T0 RIGHT SHIFT A&Q
T13 RIGHT SHIFT A&Q
(T1-T12) ADD/SUBT. H-REG. TO/FROM A-REGISTER
T13 RIGHT SHIFT A&Q
T15 RIGHT SHIFT A&Q
T15 RIGHT SHIFT A&Q
T14 RIGHT SHIFT A&Q
PHASE COUNTER =0?
NO
YES


DIVISION
SIGN DIGIT OF A&H SAME ?
YES
NO
T_{15} SET DIVISION FLIP-FLOP
T_{15} ADVANCE PHASE COUNTER
T_0 LEFT SHIFT A-REG (OMIT FIRST TIME)
$(T_1 - T_{12})$ A-REGISTER ADD H-REG.
$(T_1 - T_{12})$ A-REGISTER SUBT. H-REG.
T_{13} SET Q_0
T_{13} RESET Q_0
PHASE COUNTER = 0 ?
NO
YES
$(T_1 - T_{12})$ LEFT SHIFT Q-REG. INTO A-REG.

TIME DIAGRAM (CON'T.)

fig. 2.

2.4 Delay Line Store

The computer uses a recirculating, magnetostrictive delay line as the main data memory, STORE S. Magnetostrictive delay lines are passive electromechanical devices capable of delaying electrical signals. They reduce the velocity of an electrical wave by converting it to a sound wave and then reconverting it to an electrical signal. The delay line pack includes gating logic for entering information into the memory and a flip-flop to provide both assertion and negation outputs.

STORE S stores 61 data words of 12 bits, each of which are separated by 4 spacer bits to make up a computer word of 16 bits. The data words appear in serial mode at the output of the delay line with the least significant bit coming first and are recirculated to form a continuous, sequential access memory.

The computer operates at a clock frequency of 200 kilocycles per second. Bits appear at the output of the delay line at the rate of 200 thousand per second, or 5 microseconds apart. The delay line has a delay of 4880 microseconds and stores 976 bits of information, i.e. 61 words of 16 bits each.

The words in the STORE S are assigned location numbers L0 through L60. Program steps involving S specify the required location number. The computer can modify the location number during execution of the program, if desired.

2.5 Registers

The computer has an accumulator consisting of two, 12 bit, right and left shifting registers called the A-REGISTER and the Q-REGISTER. The A-REGISTER is the main input/output register for communication with the peripheral equipment. It also holds the augend or minuend for addition and subtraction operations and the multiplier and dividend for multiplication and division operations. The A-REGISTER also holds the operand in conditional jump operations. The Q-REGISTER is required for multiplication and division.

The computer is provided with four, 12 bit, HOLD REGISTERS or H-REGISTERS which serve as an immediate access working store. The H-REGISTERS partly overcome the disadvantage that the delay line store is not immediate access. Any one of the four H-REGISTERS can hold the addend, subtrahend, multiplicand or divisor for the addition, subtraction, multiplication and division instructions, respectively. Transfer instructions move data between the A-REGISTER and the H-REGISTER, and between the S-STORE and the H-REGISTERS.

Also provided are two, 6 bit, INDEX REGISTERS or I-REGISTERS which are required for modification of S-STORE location numbers.

2.6 Arithmetic Unit

The arithmetic unit is capable of addition, subtraction, multiplication, and division. It has a serial ADDER/ SUBTRACTOR unit. The addition and subtraction time is one machine word length (not counting the access time required for operations involving the S-STORE). Multiplication and division are performed by repeated use of the ADDER/SUBTRACTOR. Multiplication time is seven machine word lengths (560 microseconds). Division time is twelve machine word lengths (960 microseconds).

The arithmetic unit can shift the number in the A-REGISTER one bit to the left or one bit to the right. These operations require only one machine word length and are equivalent to multiplying or dividing the number in the A-REGISTER by 2. This provides a fast way of doubling or halving a number.

2.7 Adder/Subtractor Unit

The following symbols are defined.

X = augend or minuend

Y = addend or subtrhend

Z = sum or difference

W_i = input carry or input borrow

C_o = output carry

B_o = output borrow

The truth table for a full adder/subtractor is is given in fig. 3.

X	Y	W_i	Z	C_o	B_o
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	1

fig. 3. ADDER/SUBTRACTOR TRUTH TABLE

This truth table is implemented by the ADDER/SUBTRACTOR logic.

2.8 Multiplication Unit

The computer performs multiplication by the serial, repeated addition and subtraction method. An algorithm for multiplication in the signed-2's complement form, developed by A. D. Booth,[2.] depends on the examination of the two least significant digits of the multiplier register. This is followed by either an addition or subtraction, or neither, of the multiplicand to the partial product after which a right shift of the partial product and the multiplier takes place. This is repeated until multiplication is complete.

This computer uses a multiple digit multiplication method[3] which is an extension of the above method and which reduces by almost one-half the time required for multipliction.

The operation is as follows. For the first phase, the multiplier in the A-REGISTER is shifted into the Q-REGISTER. For the following six phases, three digits of the multiplier (the two least significant bits of the Q-REGISTER, and the bit held in the KEEP FLIP-FLOP) are examined according to the algorithm in fig. 4.

Multiplier Digits		Keep Digit	Algorithm
L_1	L_0	K	
0	0	0	Neither add nor subt. Shift right twice.
0	1	0	Add. Then shift right twice.
1	0	0	Shift right. Subt. Shift right again.
1	1	0	Subtract. Then shift right twice.
0	0	1	Add. Then shift right twice.
0	1	1	Shift right. Add. Shift right again
1	0	1	Subtract. Then shift right twice.
1	1	1	Neither add nor subt. Shift right twice.

fig. 4. MULTIPLICATION ALGORITHM

To implement this algorithm, a PHASE COUNTER is used to count the phases (each phase is one machine word in length), and a SHIFT MEMORY FLIP-FLOP is used to count the shifts.

2.9 Division Unit

The computer performs division by the non-restoring method.[2.] The divisor is either added to, or subtracted from, the partial remainder, depending on the sign of the divisor and that of the partial remainder. If these two signs agree, a subtraction is performed and the quotient digit is 1. If the signs do not agree, an addition is performed and the quotient digit is zero. In either case, a new partial product is next formed by the proper left shift. This process continues until the complete quotient is built up in the Q-REGISTER. The final answer is obtained by left shifting the Q-REGISTER into the A-REGISTER during phase 12.

2.10 Control Unit

The control unit directs the internal operation of the computer causing operations to occur in proper sequence and at the right time. A master CLOCK, operating at a frequency of 200 kilocycles per second, supplies a continuous train of 2 microsecond pulses, T. A four stage BIT COUNTER counts the pulses, T, in order to provide a binary count of zero to fifteen. A TIMING UNIT provides separate timing signals T_0 through T_{15}. It also provides two pulse trains called $(T_1$-$T_{12})$, and $(T_1 - T_6)$. This is clarified by fig. 5. These timing signals correspond to the consecutive bits of the 16 bit machine word and supply the basic timing for the Computer.

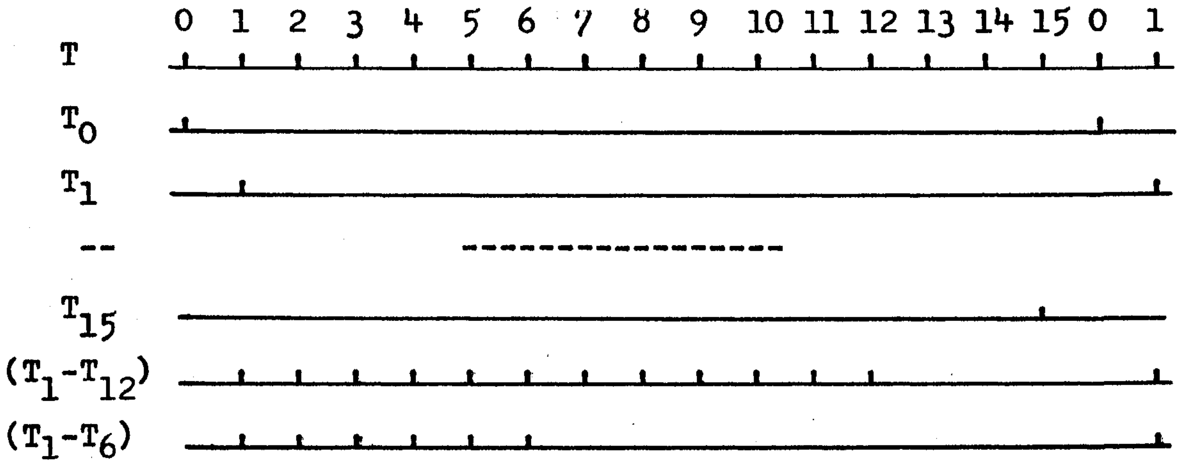


fig. 5. TIMING SIGNALS

The control unit has a six bit INSTRUCTION COUNTER. This counter is initially set to zero by the ZERO COUNTER button. Then the START button is pressed, the counter is increased by 1. This begins the program whose first step number is one. The INSTRUCTION COUNTER counts the program steps as they are performed and increases consecutively to each new program step when the previous step is completed. The INSTRUCTION COUNTER begins counting at 1 and counts up to 63, after which it returns to zero and repeats counting. This gives a total of 64 consecutive program steps (including step zero). A HALT order will stop the counter.

In conjunction with the INSTRUCTION COUNTER is an INSTRUCTION NUMBER DECODER. There are 64 output lines from the decoder, labelled DO through D63. These output lines correspond to the particular program step being executed. For example, if program step 5 is being executed, the INSTRUCTION COUNTER contains 000101 (5 in binary notation), and line D5 is energized. The output lines DO through D63 are connected to the PROGRAM BOARD described later.

The control unit also has a CONTROL REGISTER or N-REGISTER. The CONTROL REGISTER has several purposes. It is used during the execution of orders involving the S-STORE. For these orders, the data location number, LO through L60, is placed in binary form into the N-REGISTER. This binary number comes from the LOCATION ENCODER which in turn has received the location number from the PROGRAM BOARD. The WORD COUNTER is a six bit binary counter. The number in the WORD COUNTER corresponds to the location number of the data word which will appear next (not the present word) at the output of the delay line S-STORE. By comparing the binary number in the CONTROL REGISTER and in the WORD COUNTER, a COMPARE UNIT determines when the required S-STORE location becomes available for use. The COMPARE UNIT will at this time gate the S-order.

A further use of the CONTROL REGISTER is in JUMP/ TEST orders. For these orders the CONTROL REGISTER contains the number of the next program step to be executed if a program jump is to be performed. As before, this number comes from the LOCATION ENCODER which receives the program step number from the PROGRAM BOARD. If the jump is to be performed, the number in the CONTROL REGISTER is transferred to the INSTRUCTION COUNTER so that the INSTRUCTION COUNTER now holds the next program step number. If the jump is not to be performed, the INSTRUCTION COUNTER is increased by 1 so that the computer performs the next consecutive program step.

A final use of the CONTROL REGISTER is to hold the number N, in conjunction with I-REGISTER orders and for modification of data location numbers. Here N is considered to be a six bit binary integer which is entered into the CONTROL REGISTER from the PROGRAM BOARD after encoding. N can be right-shifted out of the CONTROL REGISTER as required.

2.11 Program Board

The program to be executed by the computer enters the machine on a PROGRAM BOARD. The PROGRAM BOARD is a panel containing 32 rows by 40 columns of holes or "hubs". When the PROGRAM BOARD is inserted into the machine it provides the operator access to internal circuitry. By inserting wires into the panel holes according to a definite pattern, the operator may design any desired program.

The program board is shown in fig. 7. It is divided into four sections, namely: ORDERS, PROGRAM STEPS, REGISTERS, and DATA LOCATIONS. The section called PROGRAM STEPS contains 64 groups of four hubs each. These in turn are labelled O, R, L, M for Order, Register, Location and Modifier, respectively. Each group of four hubs is numbered and is connected through four diodes to the corresponding output of the INSTRUCTION NUMBER DECODER. The INSTRUCTION

NUMBER DECODER outputs are energized (0 volts) in sequence starting at number D0 and proceeding to number D63 so that each group of hubs will also be energized in like sequence. The energized hubs are at 0 volts, whereas, all others remain at -6 volts.

The hubs in the section called ORDERS are connected as bus bars of eight hubs each. These bus bars are labelled H→A, A→H, A→H & Cl., etc. corresponding to the names of all the available orders. The hubs in the section called REGISTERS are connected as bus bars of 32 hubs each and are labelled H1, H2, H3, H4, I1, I2, M1, and M2. Finally, the hubs in the section called DATA LOCATIONS are connected as bus bars of eight hubs each. Each bus bar in this section is assigned one location number from L0 through L60. All bus bars are connected to internal circuitry.

To set a PROGRAM STEP, the operator connects the corresponding hub O to the particular bus bar in the section called ORDERS which calls for the required order. If the order involves an H-REGISTER or I-REGISTER, the hub R is connected to the particular H1, H2, H3, H4, I1, or I2 required. If the order involves the data S-STORE, and hence requires a data location, then L is connected to the appropriate hub L0 through L60. If the order involves the N-REGISTER, the section called DATA LOCATIONS is now used to assign a value to N. For example, if N is to have the binary value 001110 (which is equivalent 14), the hub L is connected to L14. Finally, if the order involves a data S-STORE location number which is to be modified, the hub M is connected to M1 or M2, depending upon whether the modifier register is I1, or I2, respectively.

In the section called PROGRAM STEPS, the diodes are used for isolation of each step. For example, consider the two program steps shown in fig. 6. It is clear that if the diodes were replaced by short circuits, then D5, and hence $\overline{A+H}$, would be taken to 0 volts. The diodes prevent this.

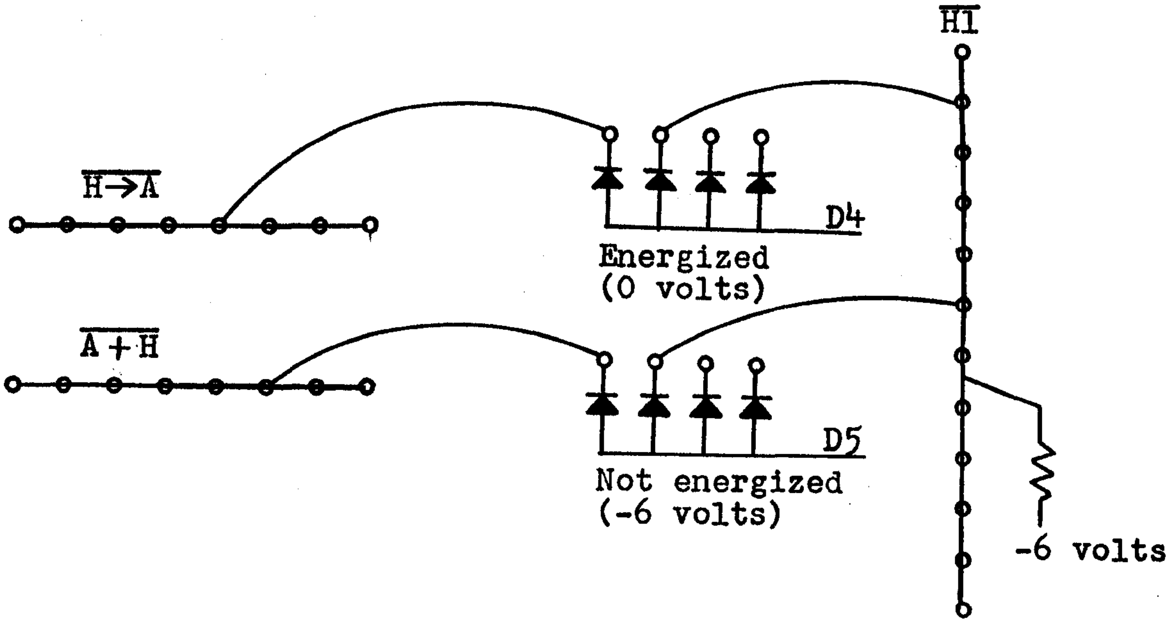
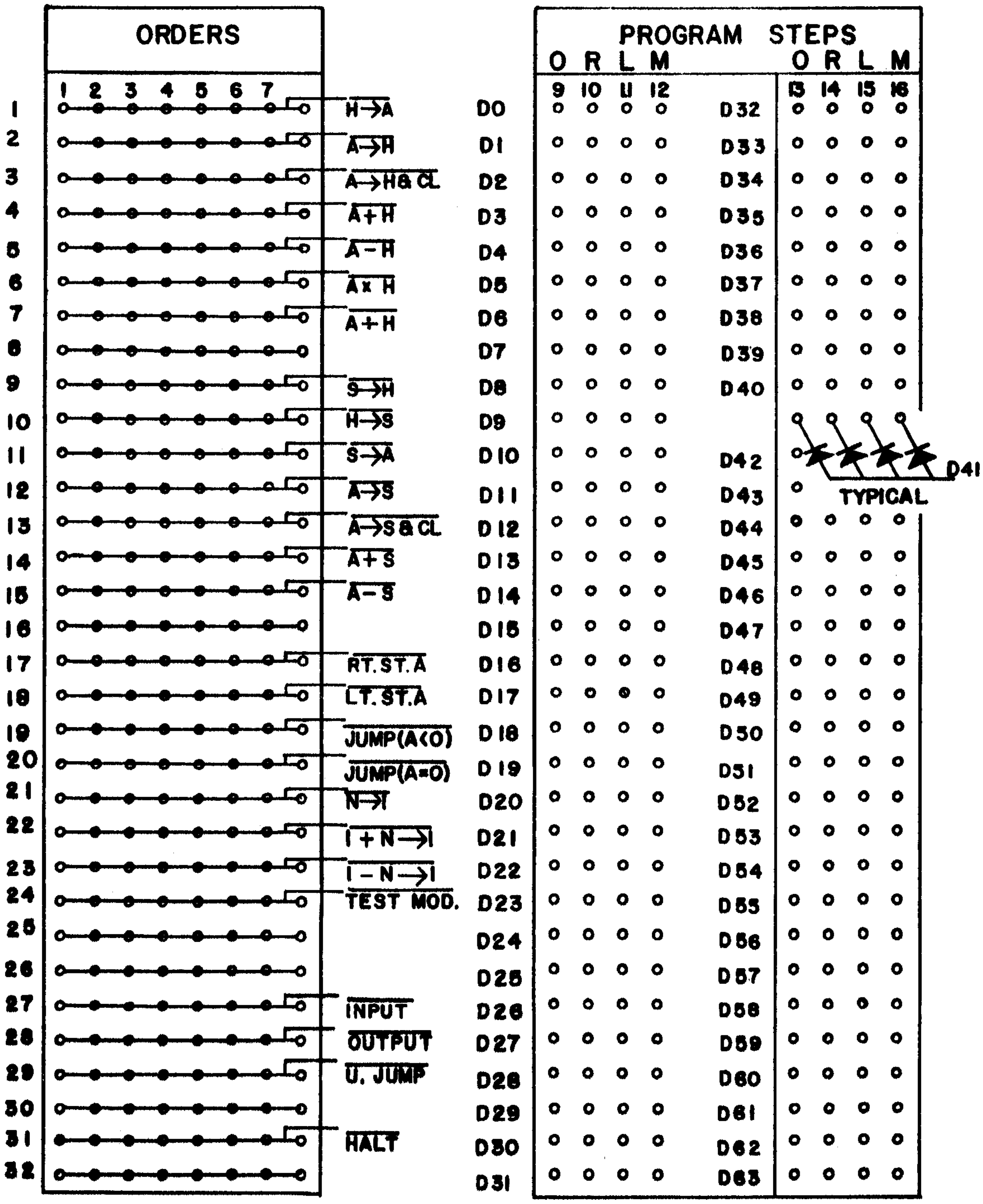
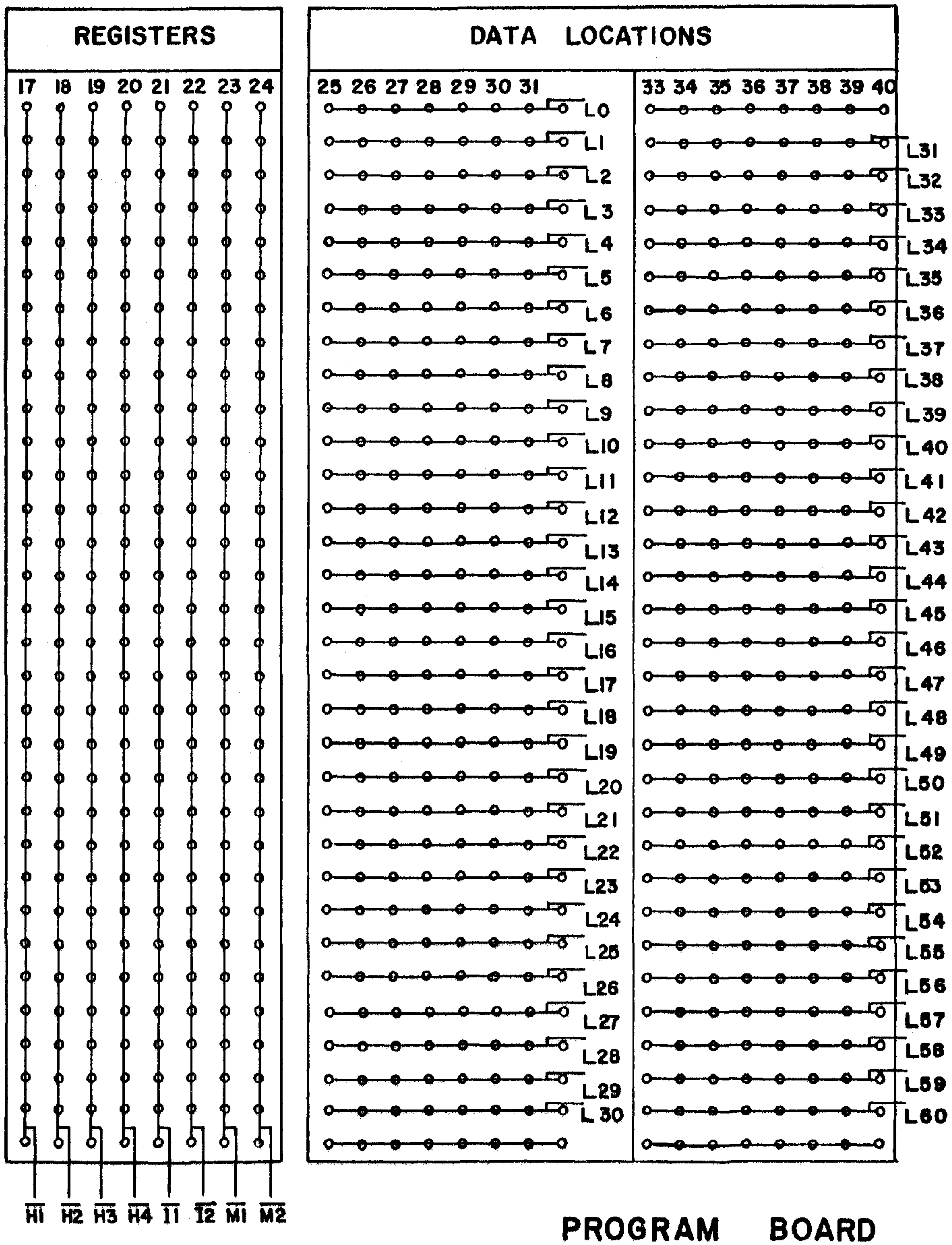


fig. 6. DIODE ISOLATION

This section has shown how to program the orders onto the PROGRAM BOARD. In all, there are 26 different orders, which are given in the Appendix.



O - ORDER
R - REGISTER
L - LOCATION
M - MODIFIER


REGISTERS
17 18 19 20 21 22 23 24
H1 H2 H3 H4 I1 I2 M1 M2
DATA LOCATIONS
25 26 27 28 29 30 31
33 34 35 36 37 38 39 40
L0 L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19 L20 L21 L22 L23 L24 L25 L26 L27 L28 L29 L30
L31 L32 L33 L34 L35 L36 L37 L38 L39 L40 L41 L42 L43 L44 L45 L46 L47 L48 L49 L50 L51 L52 L53 L54 L55 L56 L57 L58 L59 L60

PROGRAM BOARD

2.12 Modification and Program Looping

Often a program will require a program loop, that is, a set of sequential program steps are to be repeated a number of times before the program proceeds to the next instruction. Also, the data location of some, or all, of the S-orders in the loop are to be incremented by an integral value, N, each time around the loop. This process of modifying the data location is called MODIFICATION.

Modification and program looping can best be shown by the sample program given in fig. 8.

Program Steps	Instructions	Comments
1	Preceeding orders	
--		
10		
11	N→I	(N = 10)
12	S→A (M) Loop	Typical S-order (Modified)
13	Other orders	
--		
22		
23	I - N→I	(N = 1)
24	TEST MOD.	(N = 12)
25	Following orders	
--		
63		

fig. 8. SAMPLE OF MODIFICATION AND LOOPING

Suppose that, in this program, the loop is to be repeated 10 times, with the initial location to be 30 and the final location to be 39. Program step 11, i.e. N→I, is used to preset the I-REGISTER to 10. The steps 12 to 24 constitute the program loop. It is assumed that step 12 requires modification. Step 12 is programed on the PROGRAM BOARD with L connected to L40 (not L30). This is explained as follows.

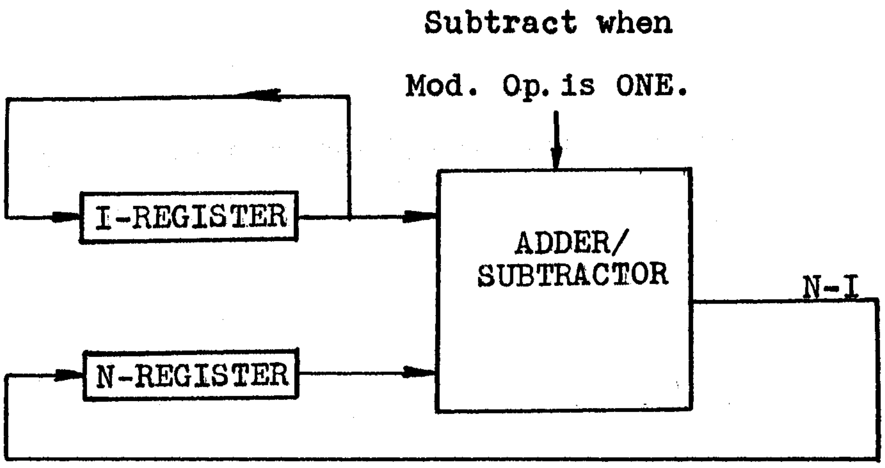


fig. 9. MODIFICATION PROCESS

Modified instructions are executed in two stages. The location is modified during the first stage and the instruction is carried out during the second stage. During the first stage the signal MODIFICATION OPERATION (Mod. Op.) is generated. This signal causes the difference, N-I, to be placed into the N-REGISTER as shown in fig. 9. During the second stage, the effective value of the data location is now N-I. In the sample problem, the effective location for the first time around the loop is 40 - 10 = 30.

Refering still to the sample problem, program steps 23 and 24 cause the program to loop. Step 23, i.e. I - N→I, changes the number in the I-REGISTER. In the sample problem, N equals 1, so that at the completion of the first loop the number in the I-REGISTER is 10 - 1 = 9. The I - N→I order also controls the TEST FLIP-FLOP. If I - N is zero, the TEST FLIP-FLOP is set to ONE, otherwise to ZERO. Step 24 is the TEST MODIFIER order. It depends on the previous setting of the TEST FLIP-FLOP. If the TEST FLIP-FLOP is ZERO, this order causes the computer to jump to the given program step (back to step 12 in the sample program), and hence to loop. If the TEST FLIP-FLOP is ONE, this order causes the computer to go to the next consecutive program step (step 25 in the sample program), and hence leave the loop.

2.13 Input/Output

The computer has two means of data input. The first is a KEYBOARD with one CLEAR ACCUMULATOR button and 12 INPUT buttons for entering ONE'S into the twelve bit positions of the A-REGISTER. To enter data into the S-STORE from the KEYBOARD, the subroutine shown in fig. 10. may be used.

Program Steps	Instructions	Comments
0		Press ZERO COUNTER and START.
1	N→I	N = Number of data words
2	HALT	Enter data into A-REGISTER through KEYBOARD. Press START.
3	A→S	
4	I - N→I	N = increment in data location (usually 1).
5	TEST MOD.	This causes a loop back to step 2.
6	HALT	If desired to go directly on with program, this HALT is omitted.
7		
--		Main program.
63		

fig. 10 INPUT SUBROUTINE

The second means of input is a SHAFT POSITION DIGITIZER. This unit continually monitors the position of a rotating shaft in the external system and converts the shaft position into a 10 bit binary number which is stored in a 10 bit register called the D-REGISTER. This register is actually a part of the DIGITIZER. The INPUT order transfers the data in the D-REGISTER to the 10 least significant bits of the A-REGISTER.

The computer has two means of output. The first is a set of 12 lamps connected to the outputs of the twelve stages of the A-REGISTER to indicate the contents of the

A-REGISTER. Usually a HALT order is used to make the A-REGISTER static. The second means of output is the OUTPUT order. The OUTPUT order sets one of two OUTPUT FLIP-FLOPS to ONE. These flip-flops can be used to gate control signals in any external on-line system.

III. IMPLEMENTATION

A discussion of how the computer logic is implemented by the use of NAND logic follows.

3.1 The Logic Elements

The physical circuitry which implements the logic for the computer has been purchased commercially from Computer Control Company, Inc. The circuitry is in the form of digital logic modules. Each module contains a number of basic logic elements.

The fundamental logic element used is the NAND (NOT-AND) gate. The NAND function is as follows:

$$Y = \overline{X_1 \cdot X_2 \cdot X_3 \cdots X_n}$$

where Y is the output, $X_1, X_2, X_3, \ldots X_n$ are the inputs, the dot (•) is the function AND, and the bar over the terms means negation. The truth table for the NAND function is given in fig. 11.

X_1	X_2	$\overline{X_1 \cdot X_2}$
1	1	0
1	0	1
0	1	1
0	0	1

fig. 11. NAND TRUTH TABLE

The NAND gate is a universal element; it can be used to synthesize AND-OR-NOT logic which would otherwise require three logic elements. A method of converting the familiar AND-OR logic to NAND logic is required.

A device which performs the NOT function must produce as an output the negation, or opposite value of its input. Thus, a ONE (-6 volts) input signal is changed to a ZERO (0 volts) output signal; and a ZERO input signal is changed to a ONE output signal. A single NAND circuit performs this function. A logic element used this way is referred to as an INVERTER.

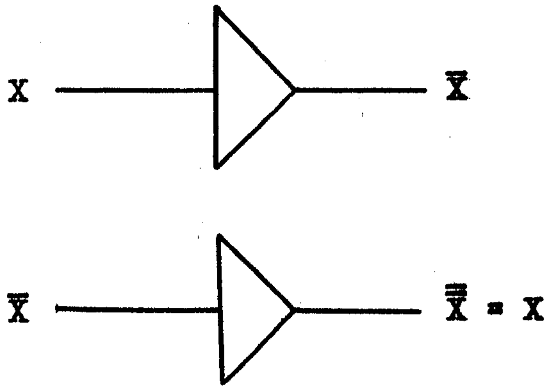


fig. 12. NOT FUNCTION

A device which performs the OR function must have a ONE output when any of its inputs is ONE; and ZERO output when all of its inputs are ZERO. To accomplish the OR function with NAND circuits, each signal is first inverted and then applied to the NAND circuit. Thus, a NAND circuit performs the OR function if its inputs are the negation of the signals that are to be ORed together. This follows from De Morgan's theorem

$$X_1 + X_2 = \overline{\overline{X_1} \cdot \overline{X_2}}$$

where + is the function OR.

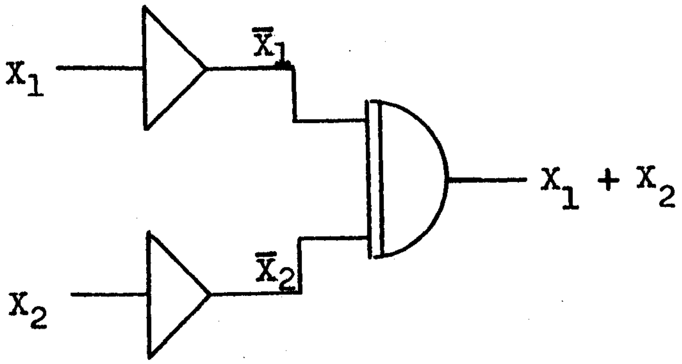


fig. 13. OR FUNCTION

A device which performs the AND FUNCTION must have a ONE output when all of its inputs are ONE, and a ZERO output when any of its inputs are ZERO. By definition, the NAND circuit produces the negation of the AND function. A second NAND can be used to remove this negation and thus form the pure AND function.

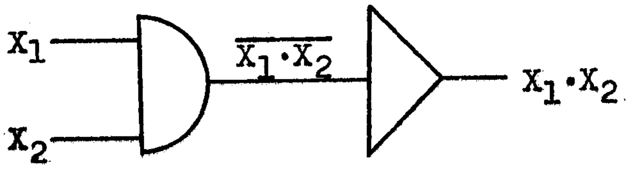


fig. 14. AND FUNCTION

Figures 12, 13, and 14 show the NOT, OR, and AND functions as they appear in the logic diagrams that follow. The functions are drawn this way so as to be readily recognized. Note, however, that all logic elements are NAND gates.

There are a set of rules that permit direct substitution of NAND gates for both AND and OR gates.

1. A NAND gate (used as an inverter) must be inserted between any AND gate that drives another AND gate and between any OR gate that drives another OR gate.
2. All inputs to OR gates that are not from within the logic network must be inverted. (If the negation is not available as an input, a NAND gate can be used to perform the inversion.)
3. The last gate should be an OR gate. If not, a NAND gate must be added to obtain the assertions of the function being implemented.

3.2 The Physical Structure

The physical computer structure consists of an internal power supply and four racks. The four racks are designed to hold the logic modules. These are labelled from top to bottom as A, B, C, and D, respectively. Each rack holds 28 logic modules, numbered 1 through 28. Each module has 35 pins, numbered 1 through 35. Each module contains from 1 to 8 logic elements and these are assigned letters, A, B, C, etc.

Therefore, an element can be fully identified as shown in fig. 15. This is a NAND gate situated in the 15th module position of RACK A. B refers to the letter assigned to this particular element within this module. 13 and 17 are the pin inputs and 23 is the pin output.

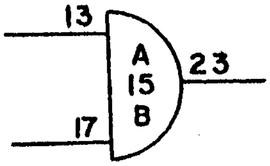


fig. 15. NAND LOGIC NOTATION

Logic elements other than NAND Gates are also assigned a neumonic code as follows:

FF	Basic flip-flop
FA	Gated flip-flop
UF	Universal flip-flop
SR	Shift register
BC	Binary counter
PA	Inverting power amplifier
PN	Non-inverting power amplifier
OD	Octal/decimal decoder
MC	Master clock

Diode Clusters may be used to expand the number of inputs to a NAND gate. These are identified as shown in fig. 16. If only one or two diodes are required (rather than a large cluster), these diodes may be wired externally. They are identified by giving the pin number, followed by a D.

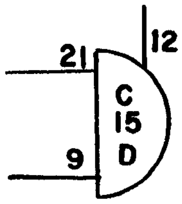


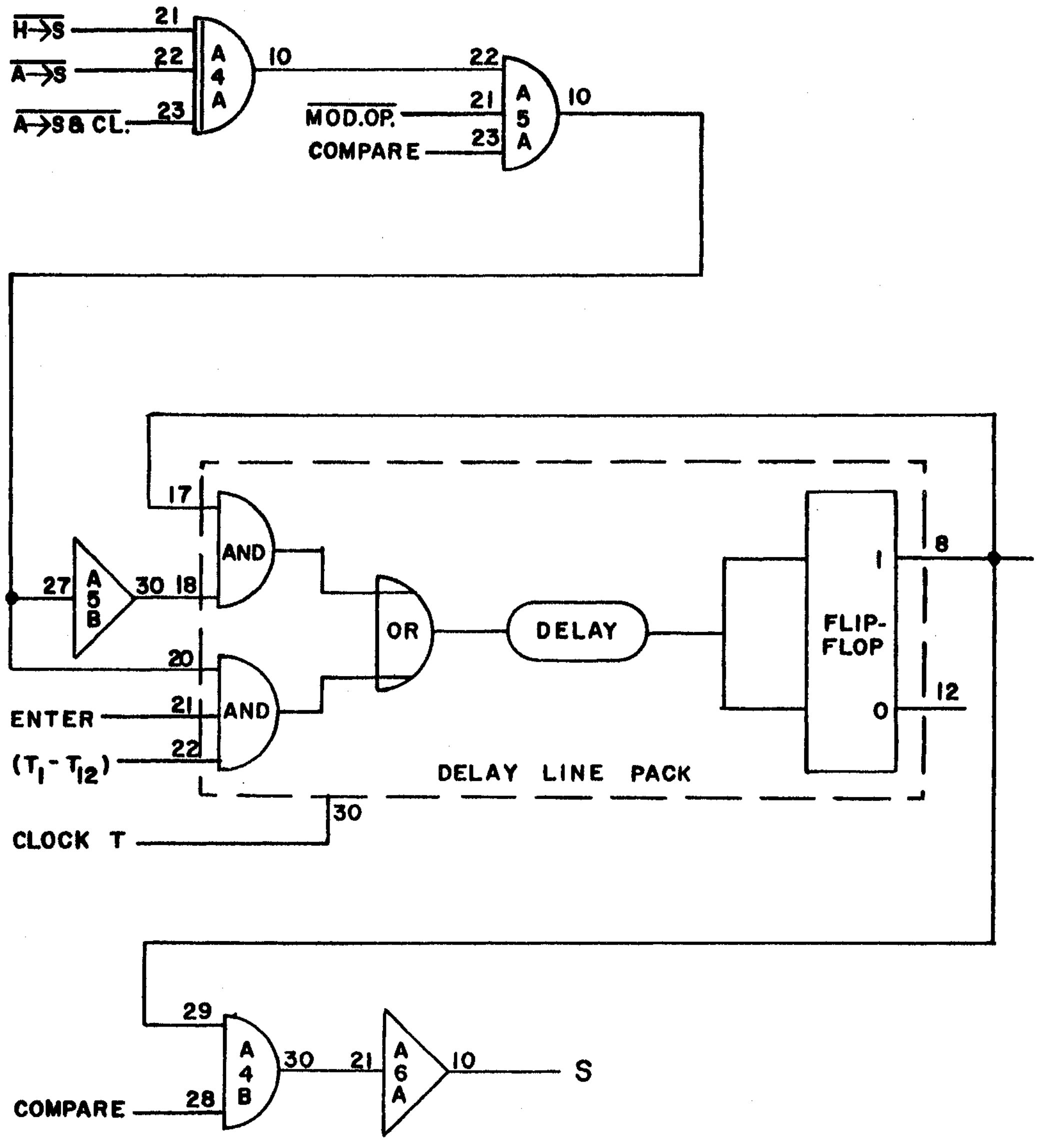
fig. 16. DIODE CLUSTER

The Reference Manual[4.] for the logic modules gives complete information concerning each logic circuit.

IV. LOGIC DIAGRAMS

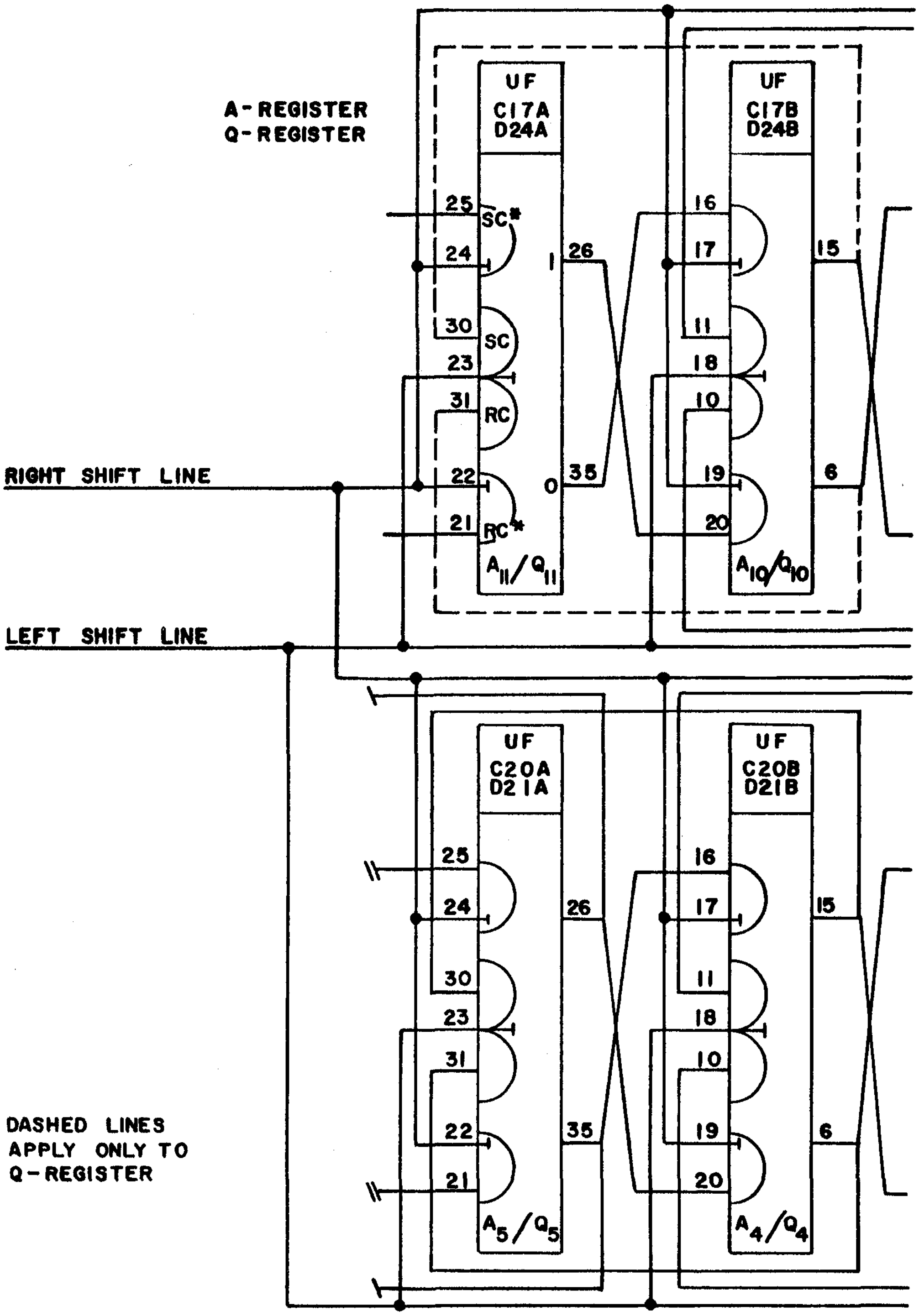
At this point a complete set of logic diagrams is given. These logic diagrams are tabulated below.

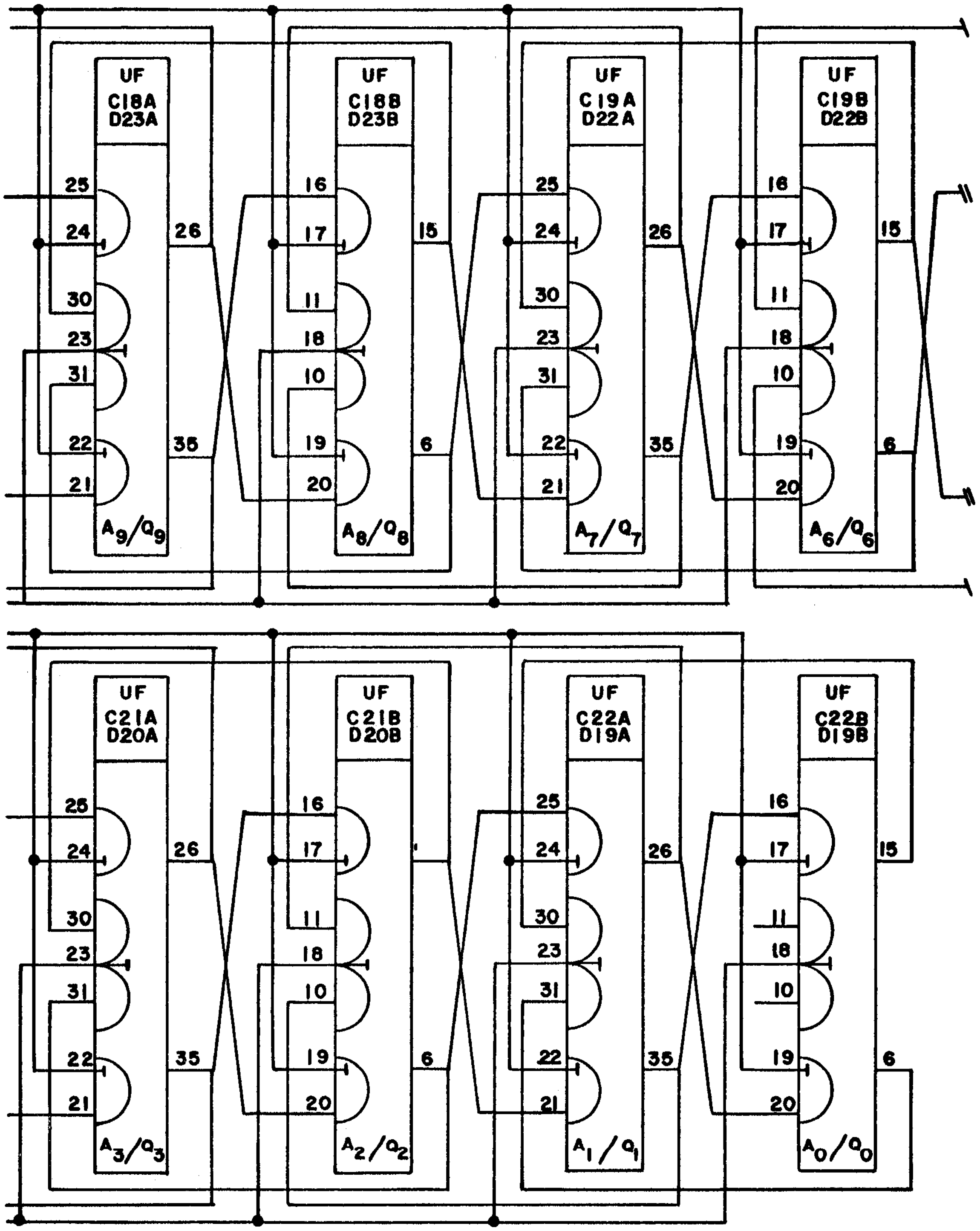
LOGIC DIAGRAMS OF COMPUTER



DELAY LINE STORE - S

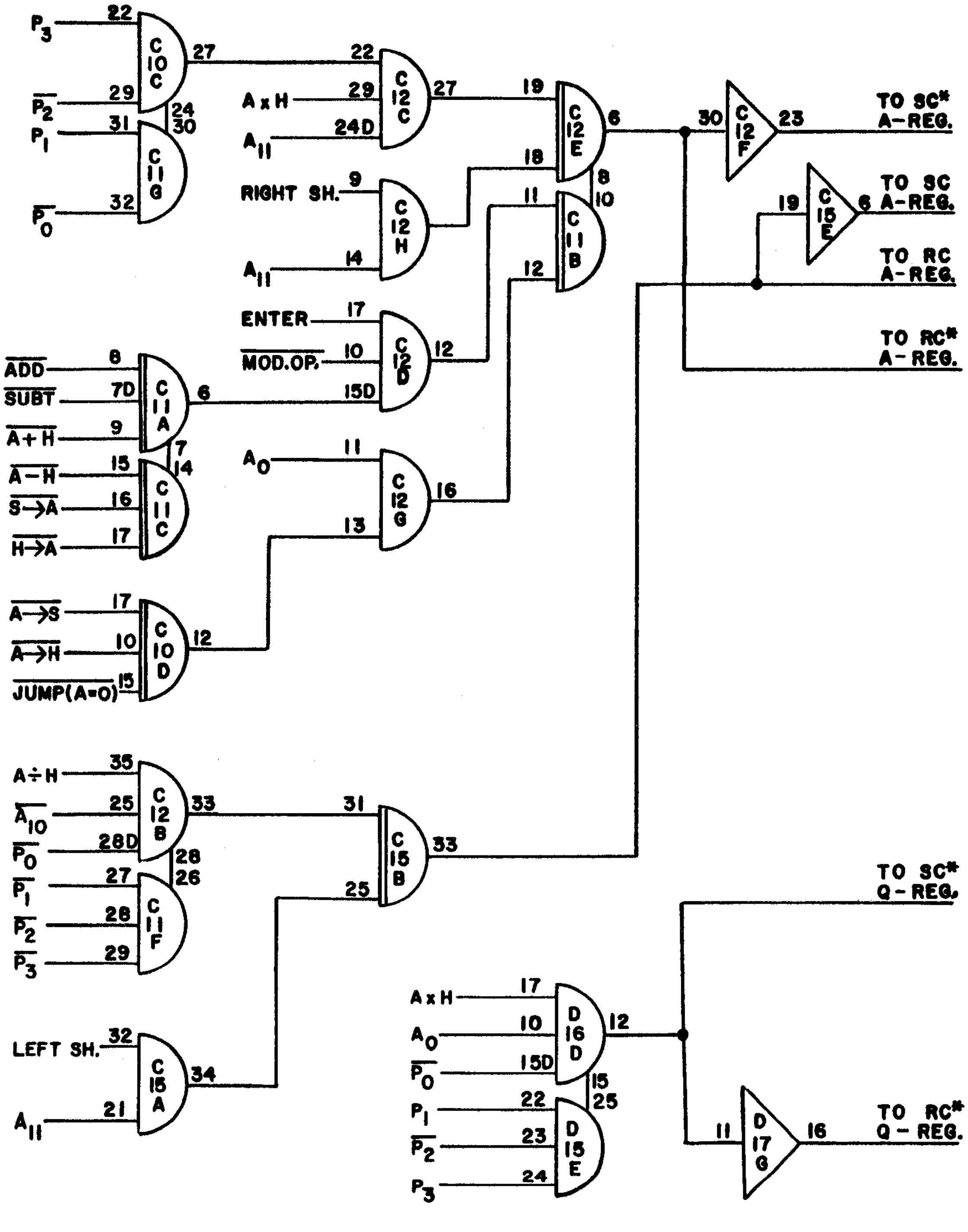
fig. 17.


A-REGISTER
Q-REGISTER
UF
C17A
D24A
UF
C17B
D24B
25
SC*
24
30
SC
23
31
RC
22
21
RC*
1
26
0
35
A11/Q11
16
17
11
18
10
19
20
15
6
A10/Q10
RIGHT SHIFT LINE
LEFT SHIFT LINE
UF
C20A
D21A
UF
C20B
D21B
25
24
30
23
31
22
21
26
35
A5/Q5
16
17
11
18
10
19
20
15
6
A4/Q4
DASHED LINES
APPLY ONLY TO
Q-REGISTER



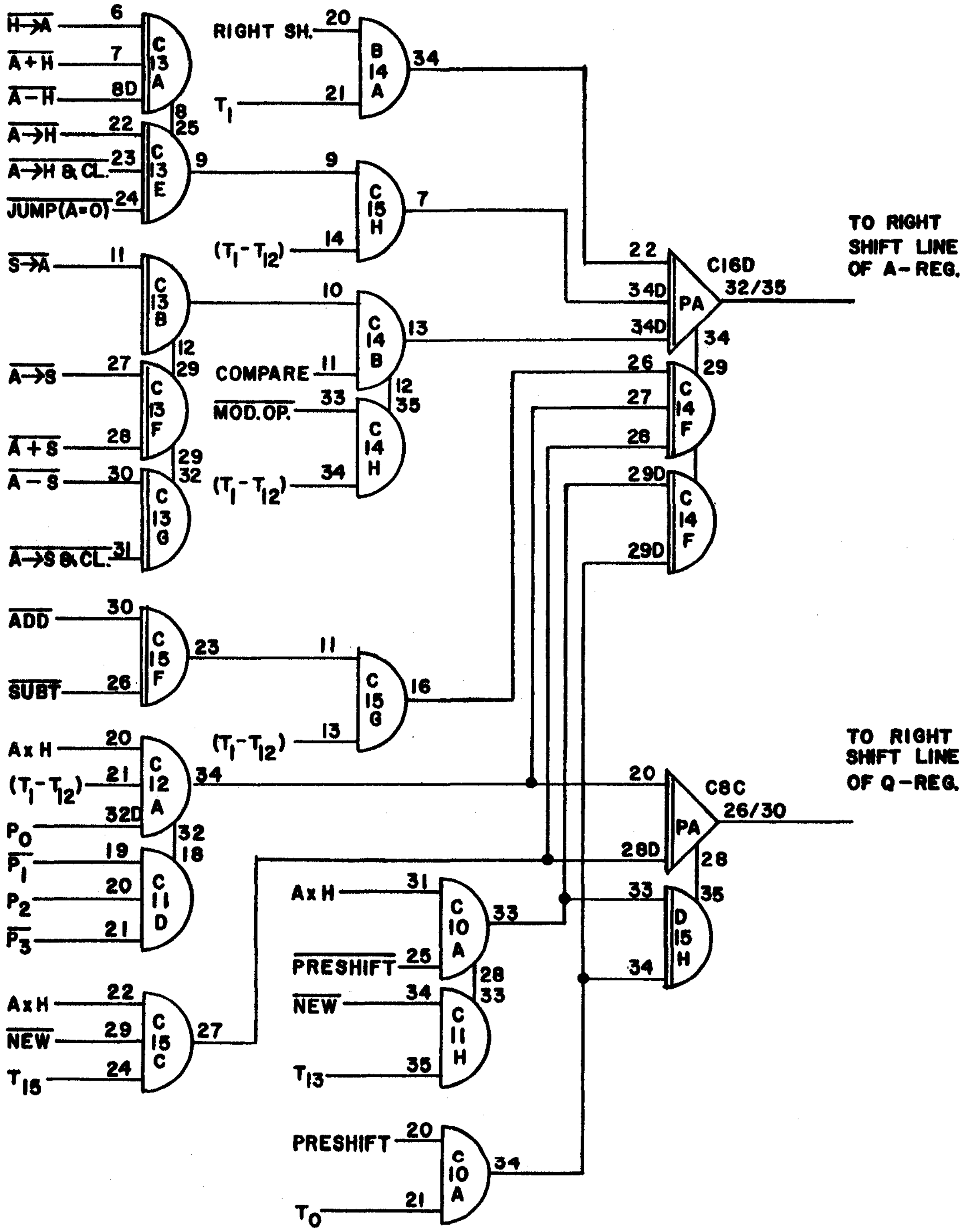
ACCUMULATOR

fig.18.



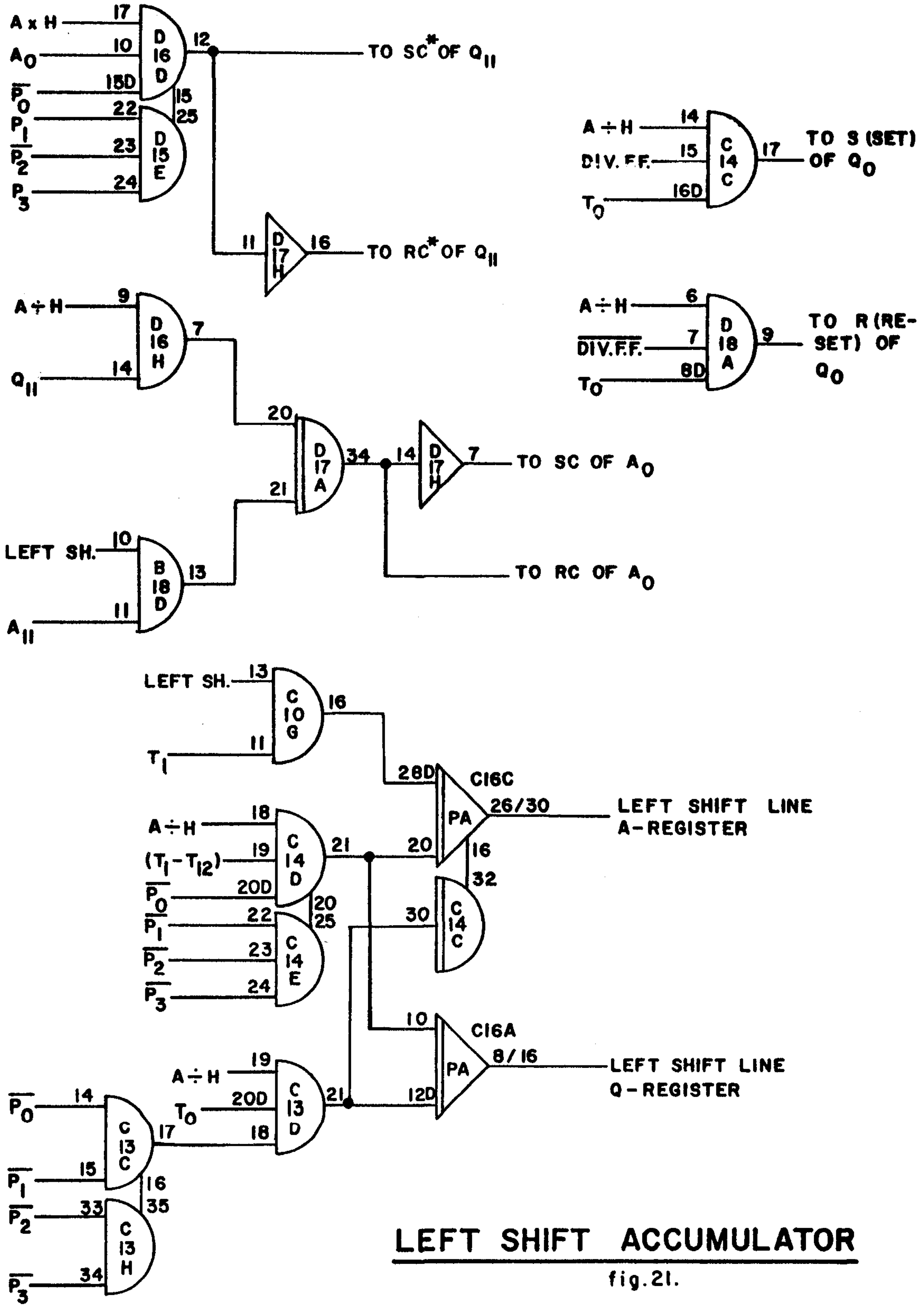
GATING OF ACCUMULATOR

fig. 19.



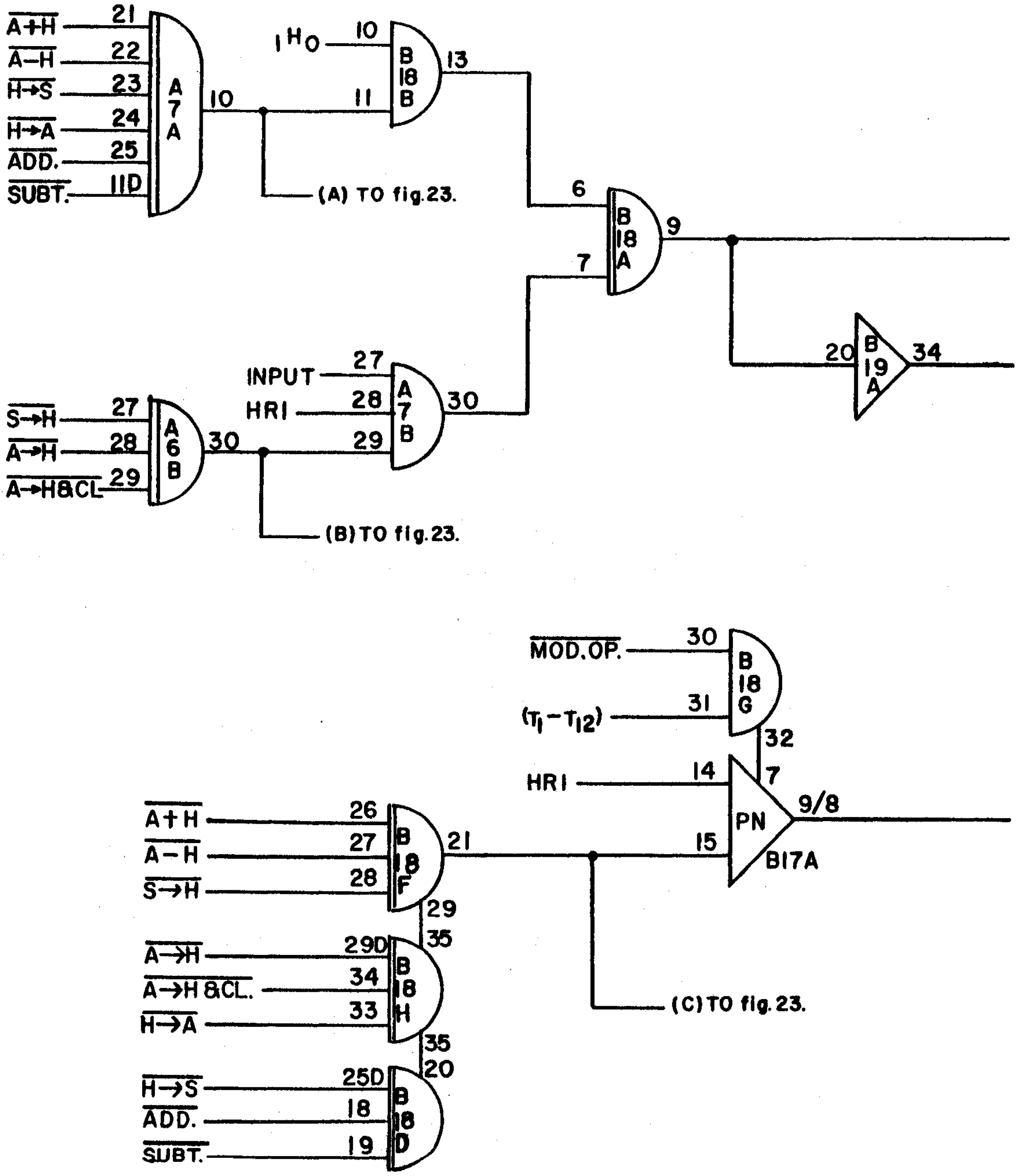
RIGHT SHIFT ACCUMULATOR

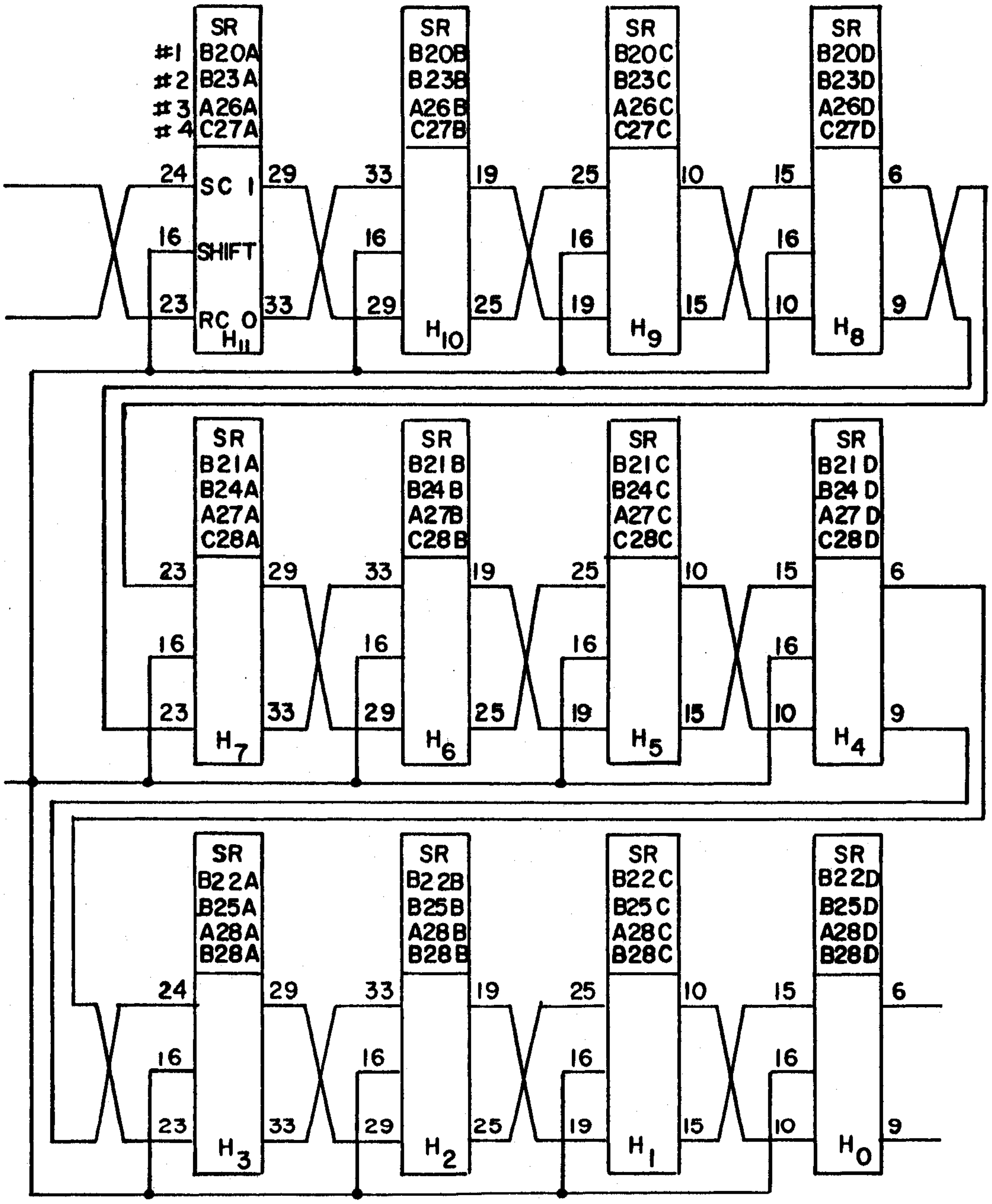
fig.20.



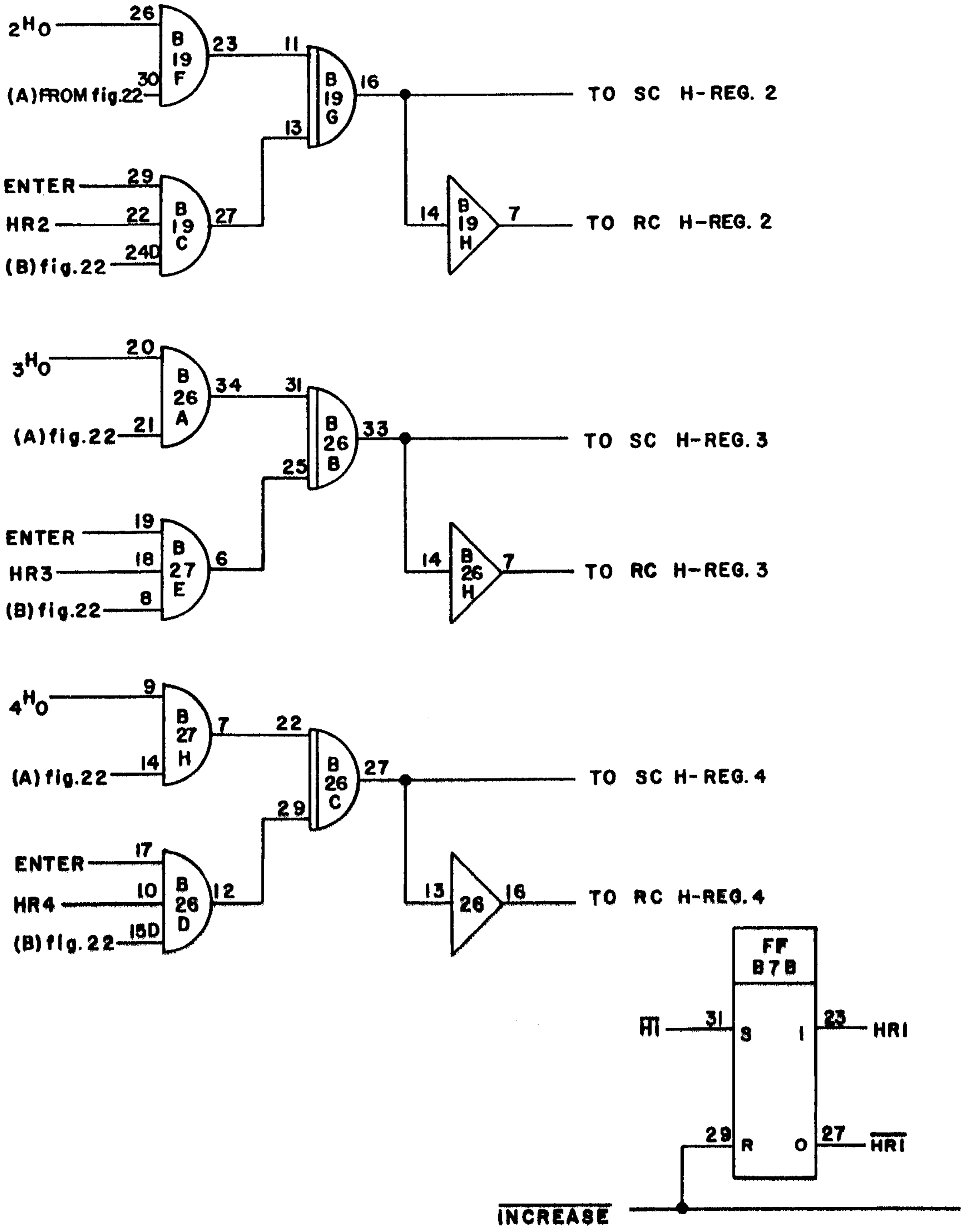
LEFT SHIFT ACCUMULATOR

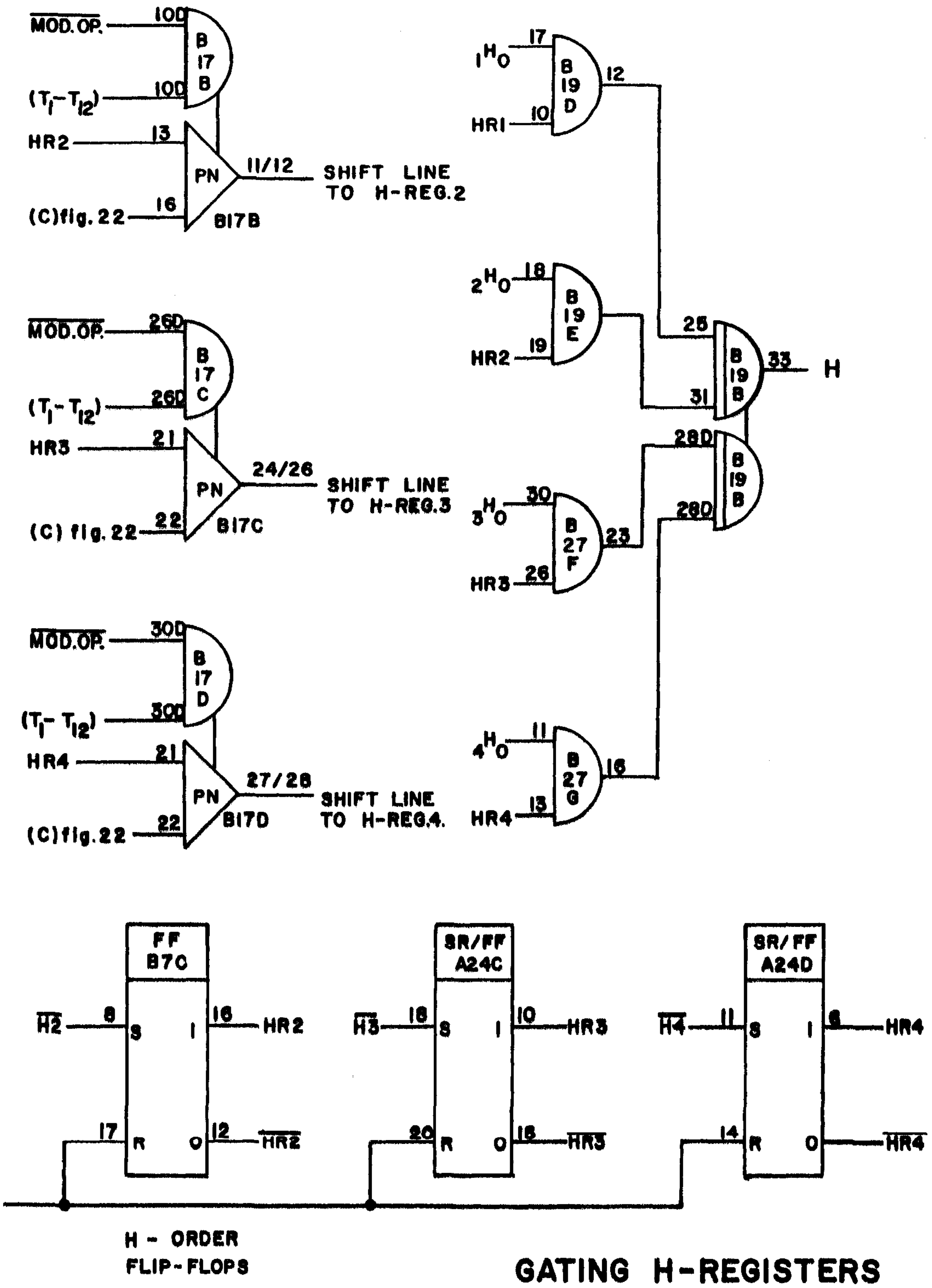
fig.21.


$\overline{A+H}$ 21
$\overline{A-H}$ 22
$\overline{H\to S}$ 23
$\overline{H\to A}$ 24
$\overline{ADD.}$ 25
$\overline{SUBT.}$ 11D
A 7 A
10
$1H_O$ 10
11
B 18 B
13
(A) TO fig.23.
6
7
B 18 A
9
20
B 19 A
34
INPUT 27
HRI 28
29
A 7 B
30
$\overline{S\to H}$ 27
$\overline{A\to H}$ 28
$\overline{A\to H\&CL}$ 29
A 6 B
30
(B) TO fig.23.
$\overline{MOD.OP.}$ 30
(T_1-T_{12}) 31
B 18 G
32
7
HRI 14
15
PN
B17A
9/8
$\overline{A+H}$ 26
$\overline{A-H}$ 27
$\overline{S\to H}$ 28
B 18 F
21
29
35
$\overline{A\to H}$ 29D
$\overline{A\to H\&CL.}$ 34
$\overline{H\to A}$ 33
B 18 H
35
20
(C) TO fig.23.
$\overline{H\to S}$ 25D
$\overline{ADD.}$ 18
$\overline{SUBT.}$ 19
B 18 D



H-REGISTERS
fig.22.

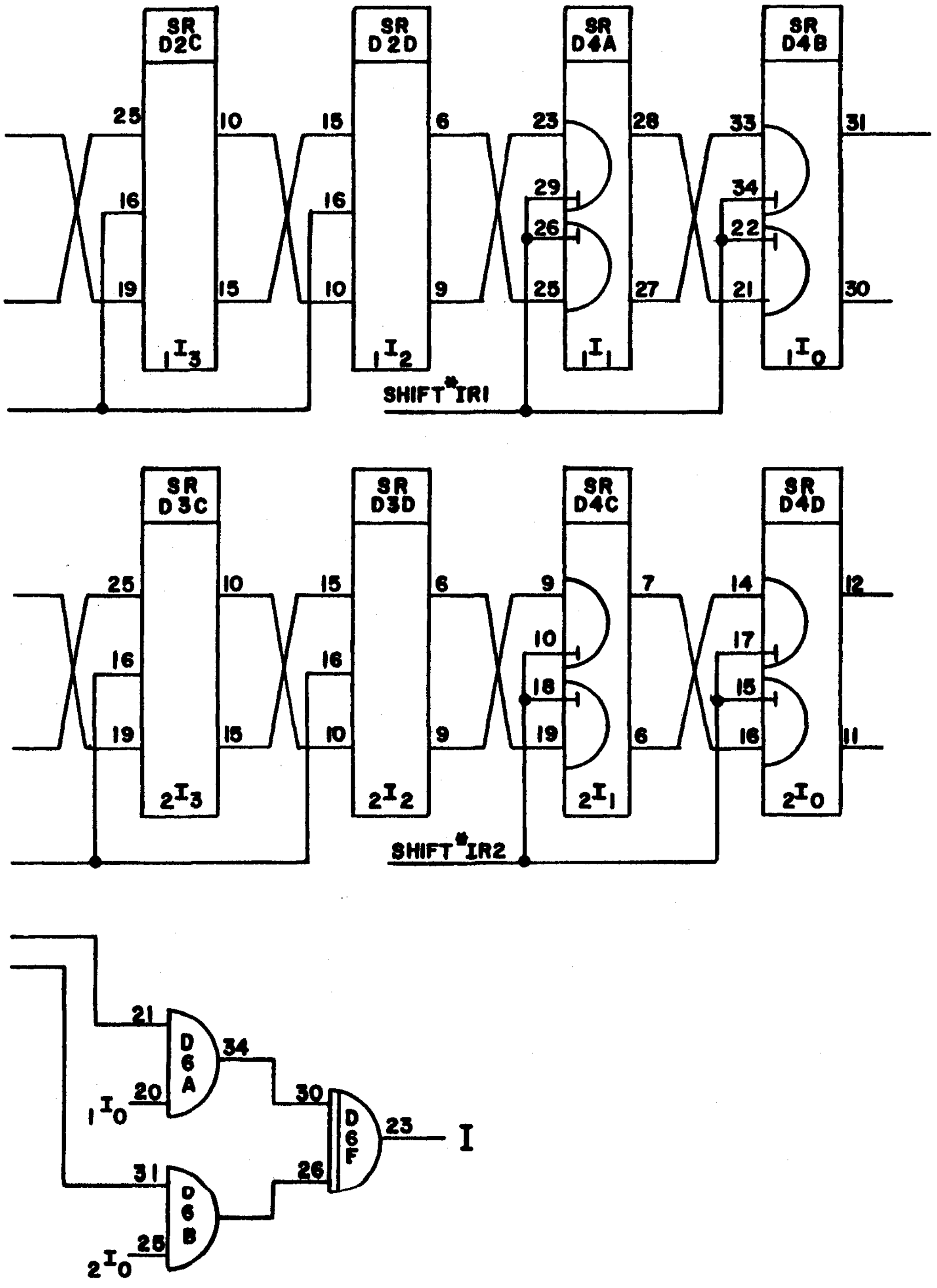

2^H0
26
B 19 F
30
(A)FROM fig.22
23
11
B 19 G
13
16
TO SC H-REG. 2
ENTER
29
HR2
22
B 19 C
(B) fig.22
24D
27
14
B 19 H
7
TO RC H-REG. 2
3^H0
20
B 26 A
(A) fig.22
21
34
31
B 26 B
25
33
TO SC H-REG. 3
ENTER
19
HR3
18
B 27 E
(B) fig.22
8
6
14
B 26 H
7
TO RC H-REG. 3
4^H0
9
B 27 H
(A) fig.22
14
7
22
B 26 C
29
27
TO SC H-REG. 4
ENTER
17
HR4
10
B 26 D
(B) fig.22
15D
12
13
26
16
TO RC H-REG. 4
FF
B7B
HI
31
S
1
23
HR1
29
R
0
27
$\overline{HR1}$
$\overline{INCREASE}$



H - ORDER FLIP-FLOPS

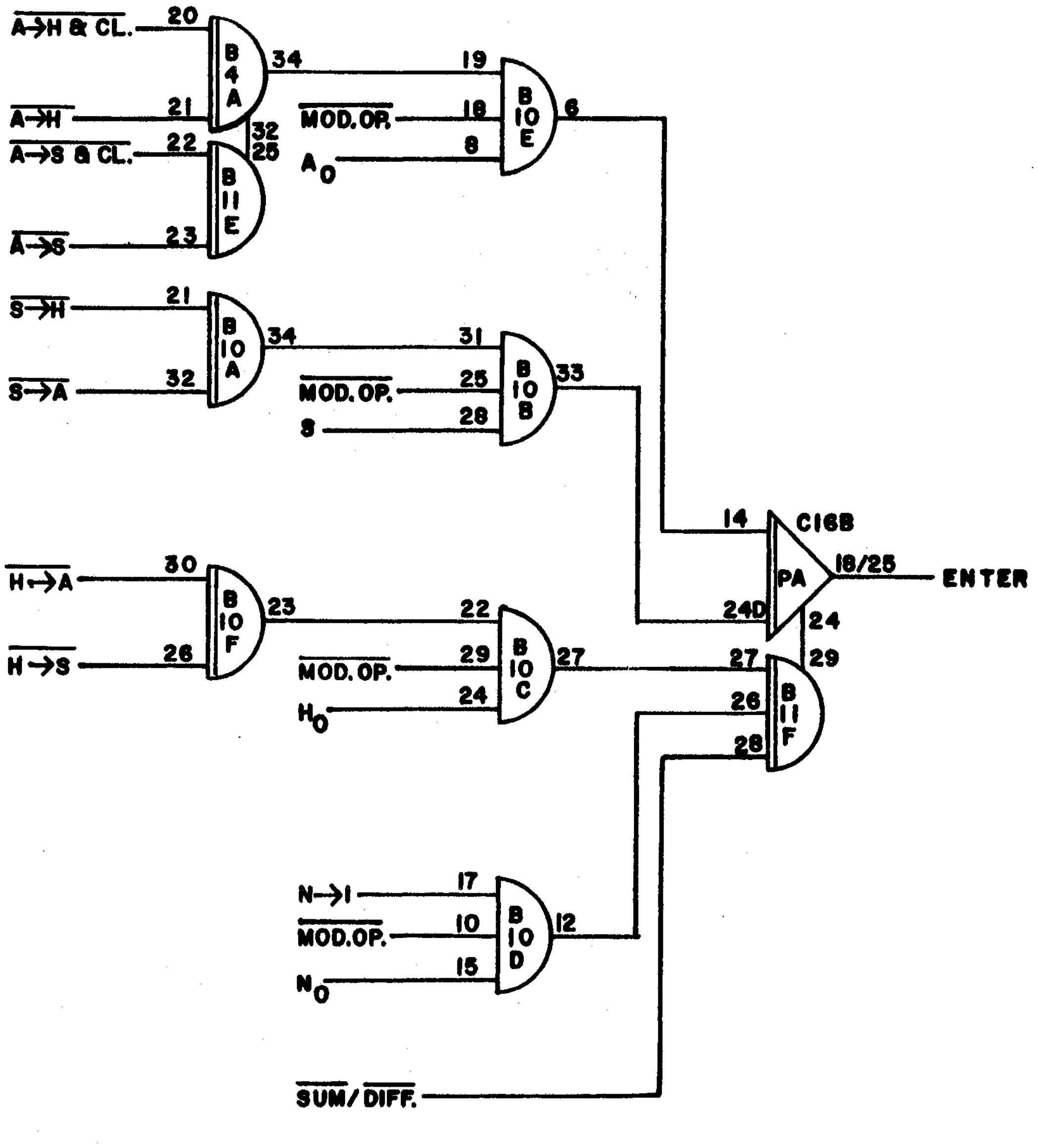
GATING H-REGISTERS

fig. 23.



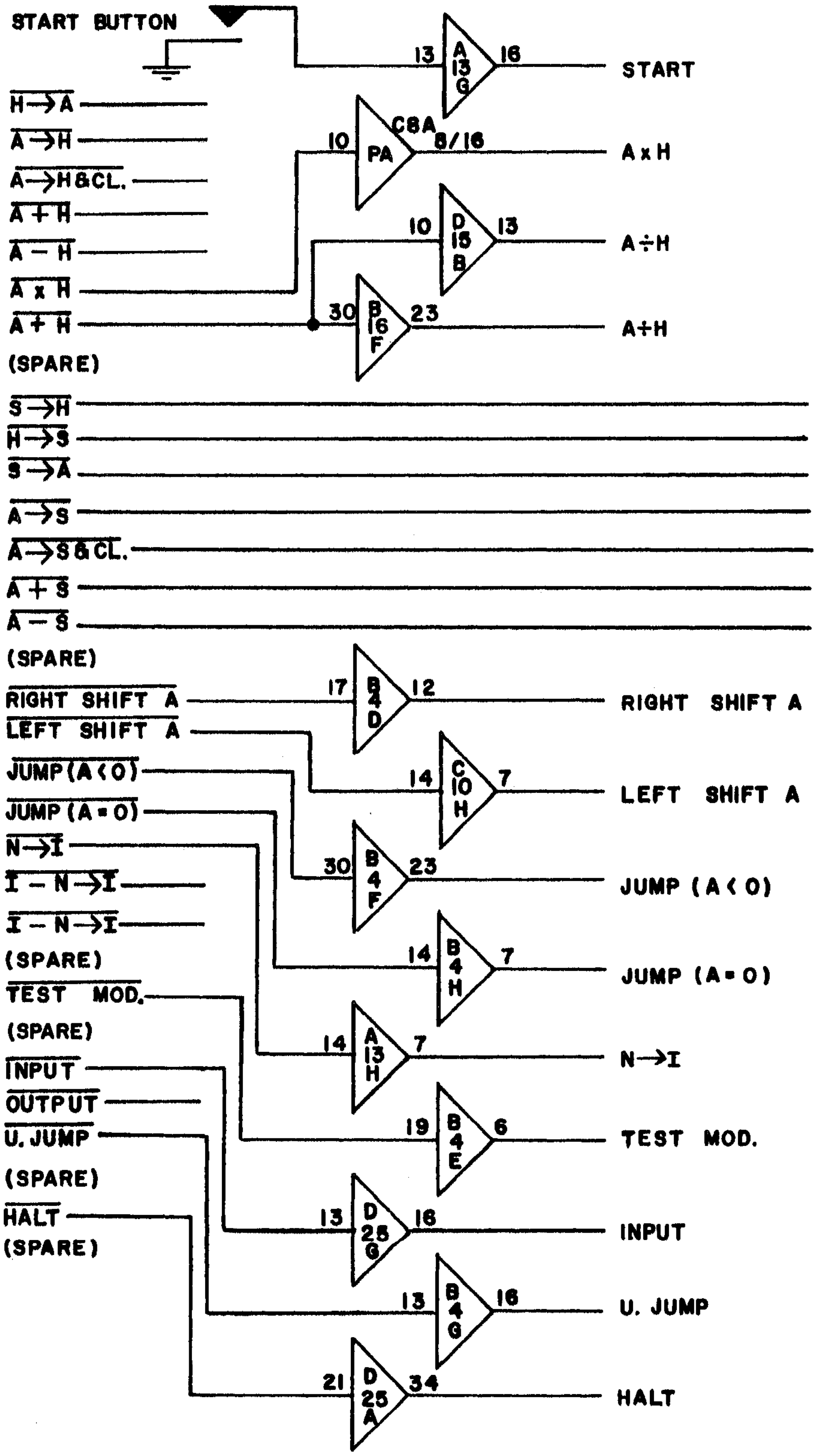
I-REGISTERS

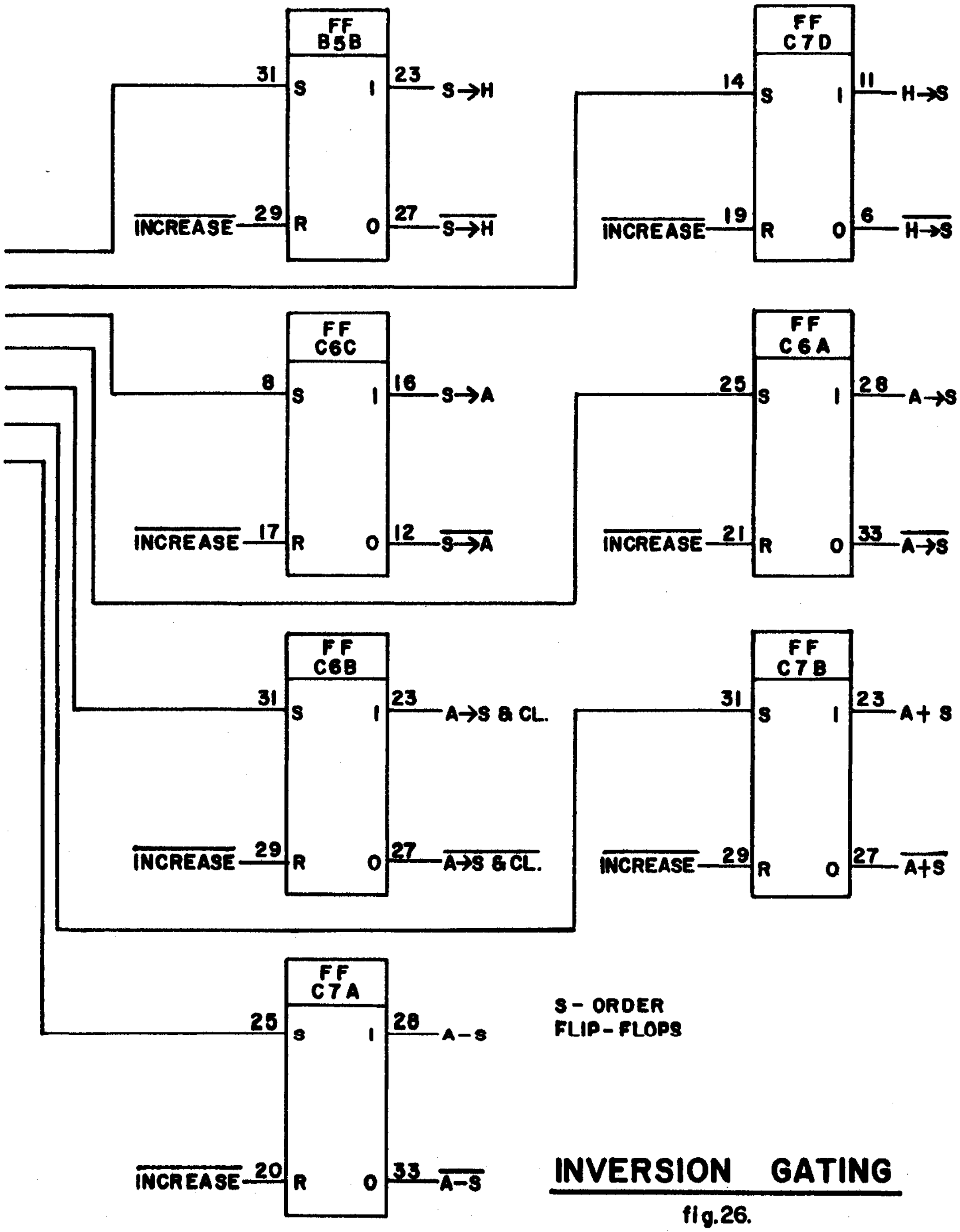
fig.24



'ENTER' BUS

fig.25.

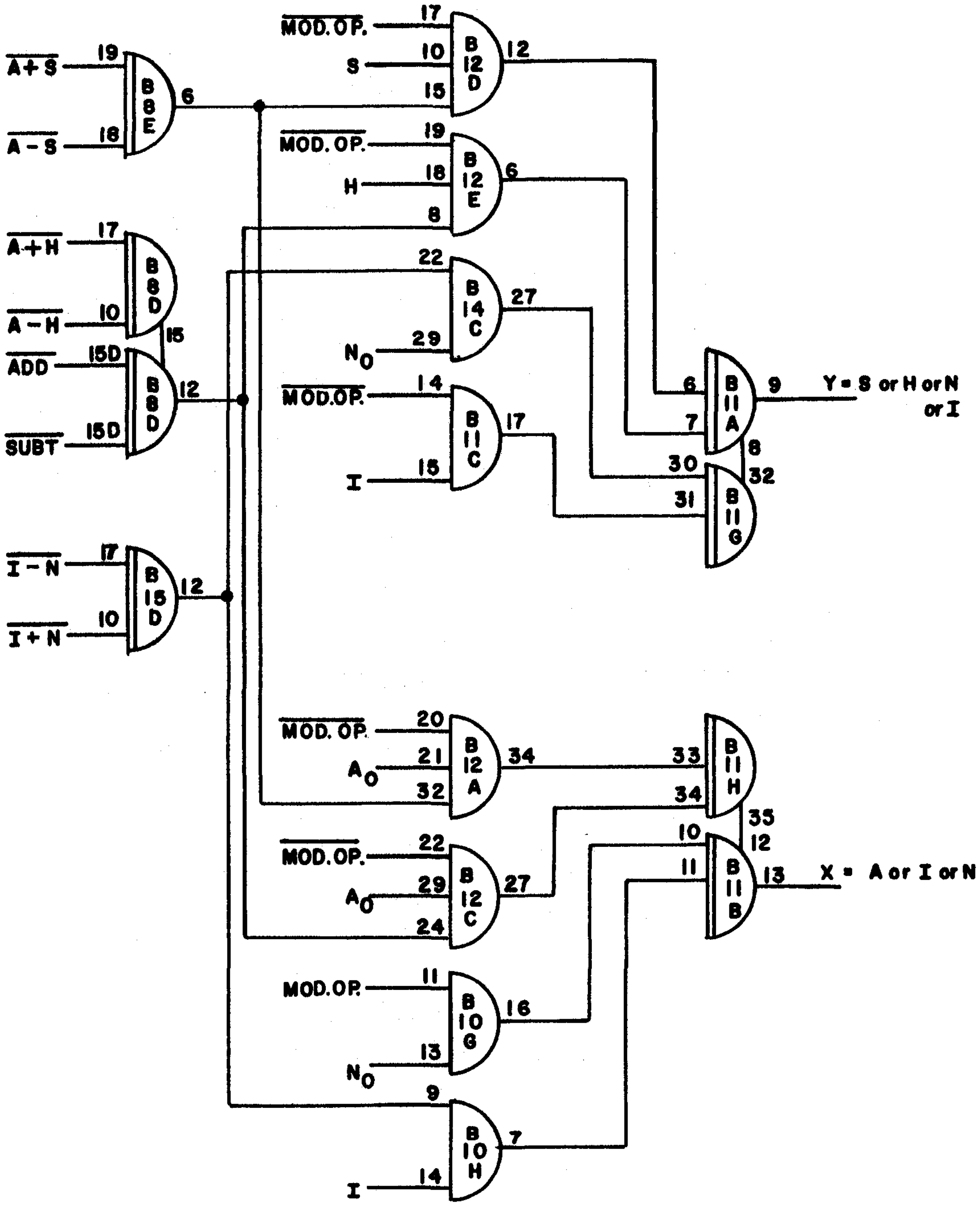

START BUTTON
$\overline{H\rightarrow A}$
$\overline{A\rightarrow H}$
$\overline{A\rightarrow H\&CL.}$
$\overline{A+H}$
$\overline{A-H}$
$\overline{A\times H}$
$\overline{A\div H}$
(SPARE)
$\overline{S\rightarrow H}$
$\overline{H\rightarrow S}$
$\overline{S\rightarrow A}$
$\overline{A\rightarrow S}$
$\overline{A\rightarrow S\&CL.}$
$\overline{A+S}$
$\overline{A-S}$
(SPARE)
RIGHT SHIFT A
LEFT SHIFT A
JUMP(A<0)
JUMP(A=0)
$\overline{N\rightarrow I}$
$\overline{I-N\rightarrow I}$
$\overline{I-N\rightarrow I}$
(SPARE)
TEST MOD.
(SPARE)
INPUT
OUTPUT
U. JUMP
(SPARE)
HALT
(SPARE)
13 A 13 G 16
START
10 PA C8A 8/16
A x H
10 D 15 B 13
A÷H
30 B 16 F 23
A+H
17 B 4 D 12
RIGHT SHIFT A
14 C 10 H 7
LEFT SHIFT A
30 B 4 F 23
JUMP (A<0)
14 B 4 H 7
JUMP (A=0)
14 A 13 H 7
N→I
19 B 4 E 6
TEST MOD.
13 D 25 G 16
INPUT
13 B 4 G 16
U. JUMP
21 D 25 A 34
HALT


FF B5B
31 S
1 23 S→H
$\overline{\text{INCREASE}}$ 29 R
0 27 $\overline{\text{S→H}}$
FF C7D
14 S
1 11 H→S
$\overline{\text{INCREASE}}$ 19 R
0 6 $\overline{\text{H→S}}$
FF C6C
8 S
1 16 S→A
$\overline{\text{INCREASE}}$ 17 R
0 12 $\overline{\text{S→A}}$
FF C6A
25 S
1 28 A→S
$\overline{\text{INCREASE}}$ 21 R
0 33 $\overline{\text{A→S}}$
FF C6B
31 S
1 23 A→S & CL.
$\overline{\text{INCREASE}}$ 29 R
0 27 $\overline{\text{A→S \& CL.}}$
FF C7B
31 S
1 23 A+S
$\overline{\text{INCREASE}}$ 29 R
0 27 $\overline{\text{A+S}}$
FF C7A
25 S
1 28 A−S
$\overline{\text{INCREASE}}$ 20 R
0 33 $\overline{\text{A−S}}$

S - ORDER
FLIP - FLOPS

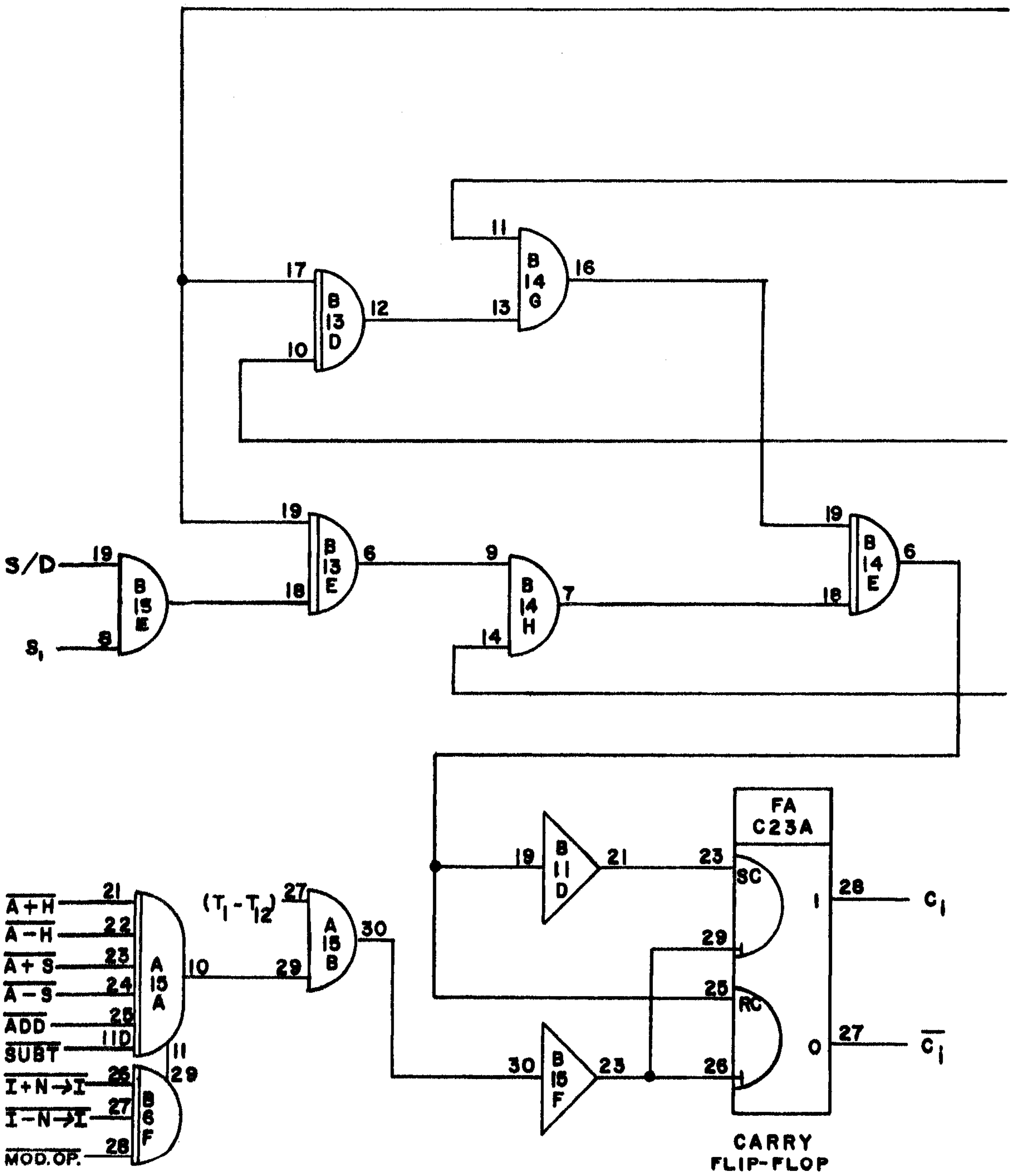
INVERSION GATING

fig.26.

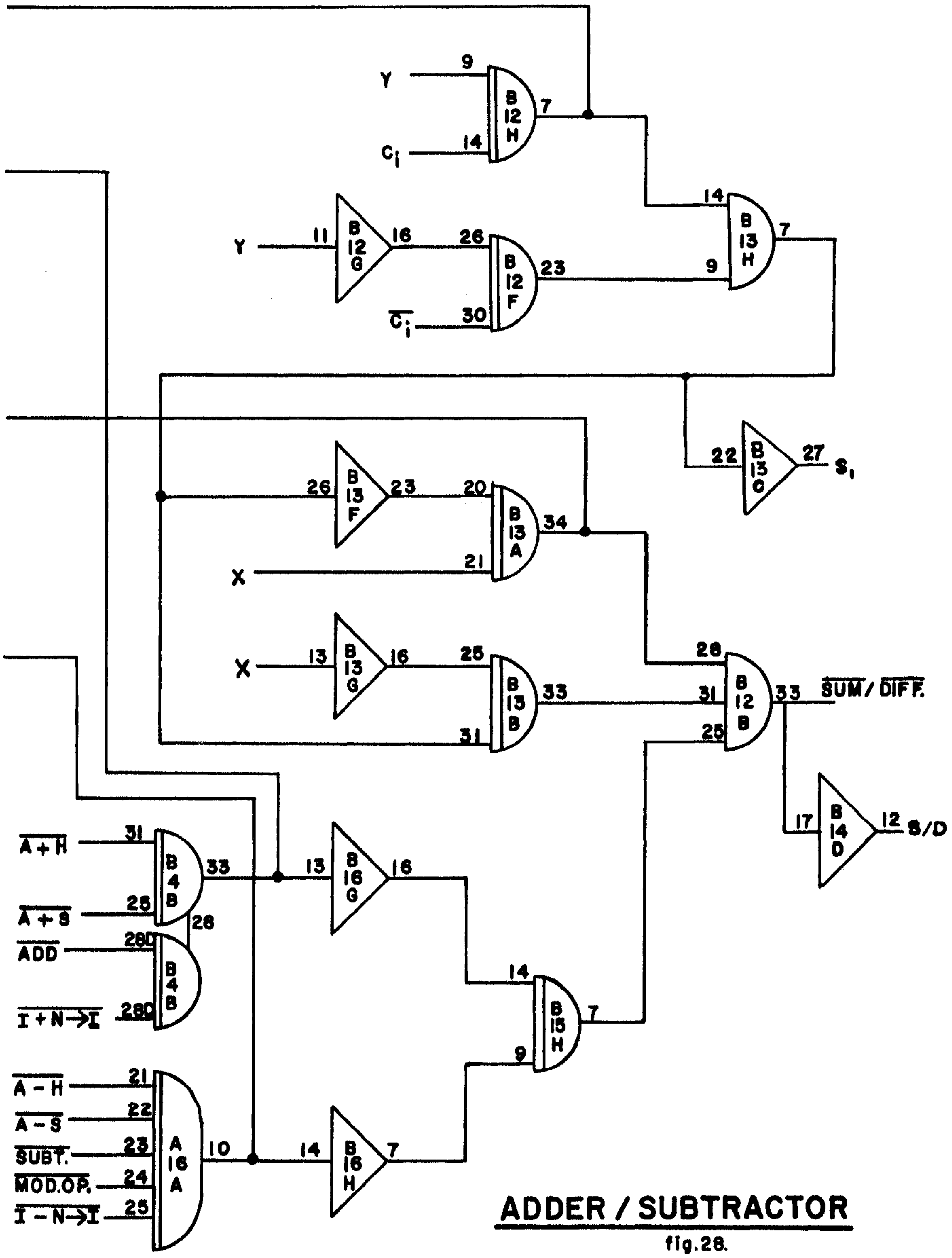


INPUTS TO ADDER/SUBTRACTOR

fig.27.

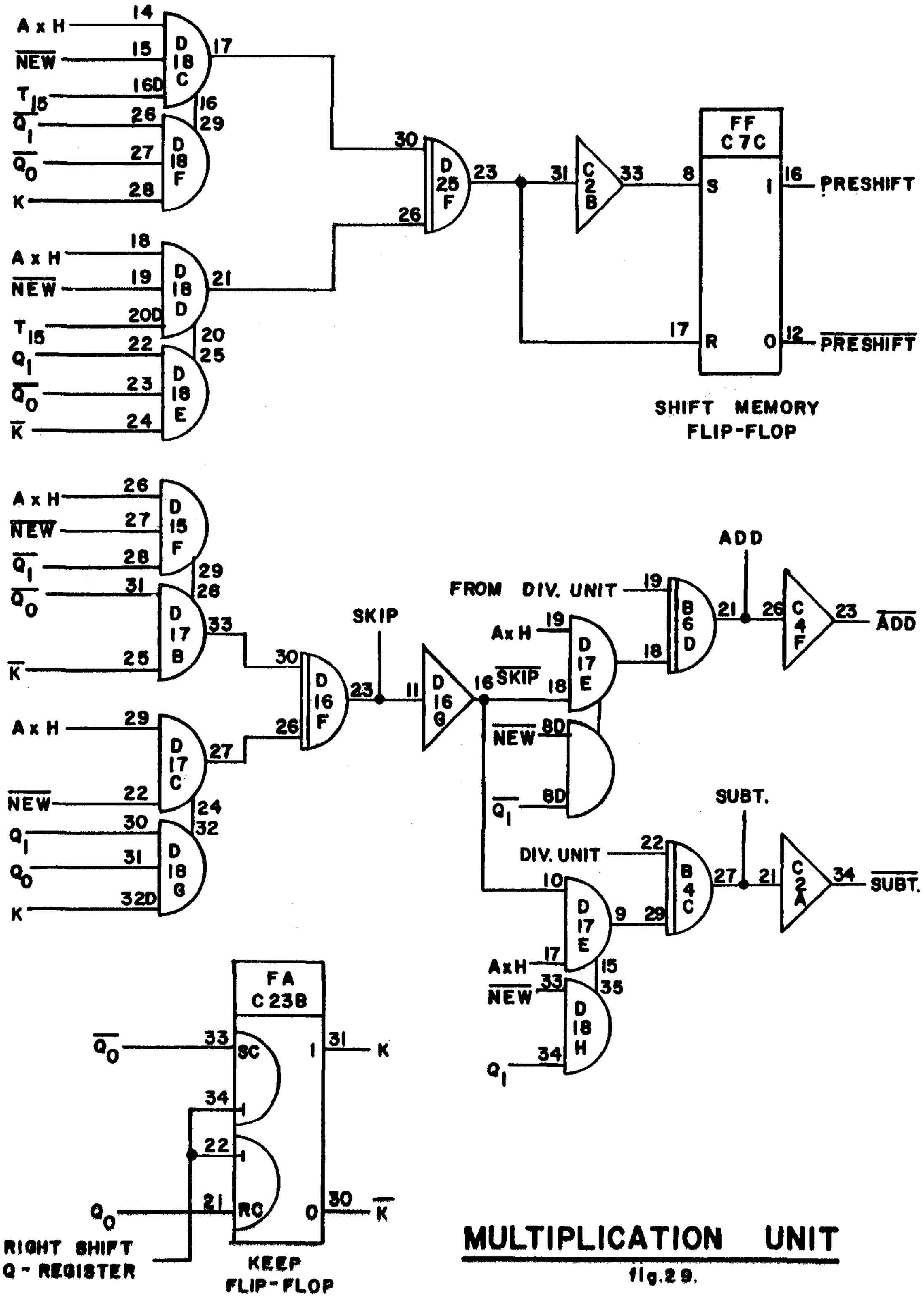


17
10
B 13 D
12
11
13
B 14 G
16
S/D
19
S_1
8
B 15 E
19
18
B 13 E
6
9
14
B 14 H
7
19
18
B 14 E
6
$\overline{A+H}$
21
$\overline{A-H}$
22
$\overline{A+S}$
23
$\overline{A-S}$
24
$\overline{ADD}$
25
$\overline{SUBT}$
11D
A 15 A
10
11
29
$\overline{I+N \to I}$
26
$\overline{I-N \to I}$
27
$\overline{MOD.OP.}$
28
B 6 F
$(T_1 - T_{12})$
27
29
A 15 B
30
19
B 11 D
21
30
B 15 F
23
FA
C23A
23
SC
29
25
RC
26
1
28
C_1
0
27
$\overline{C_1}$
CARRY
FLIP-FLOP



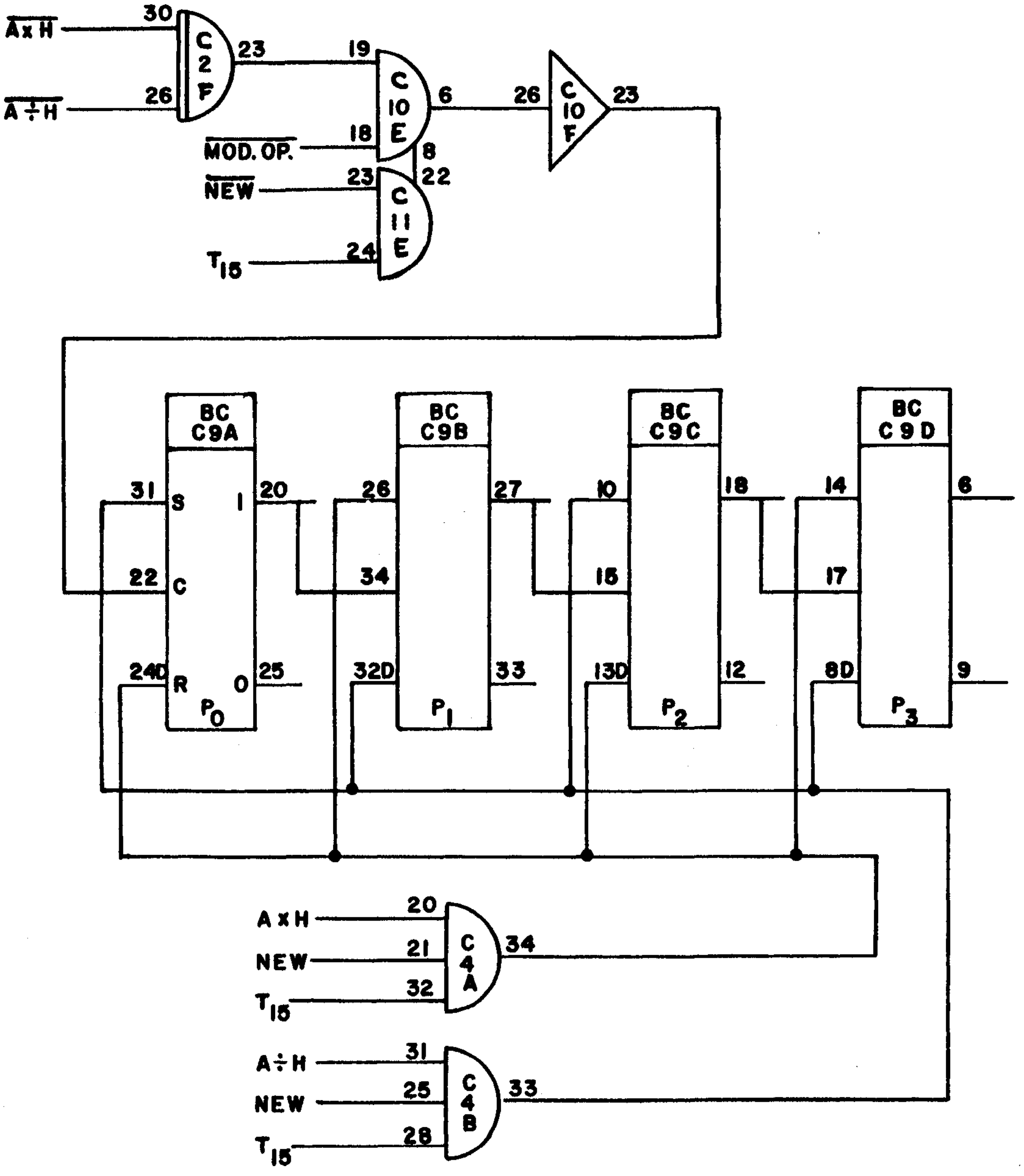
ADDER / SUBTRACTOR

fig.28.



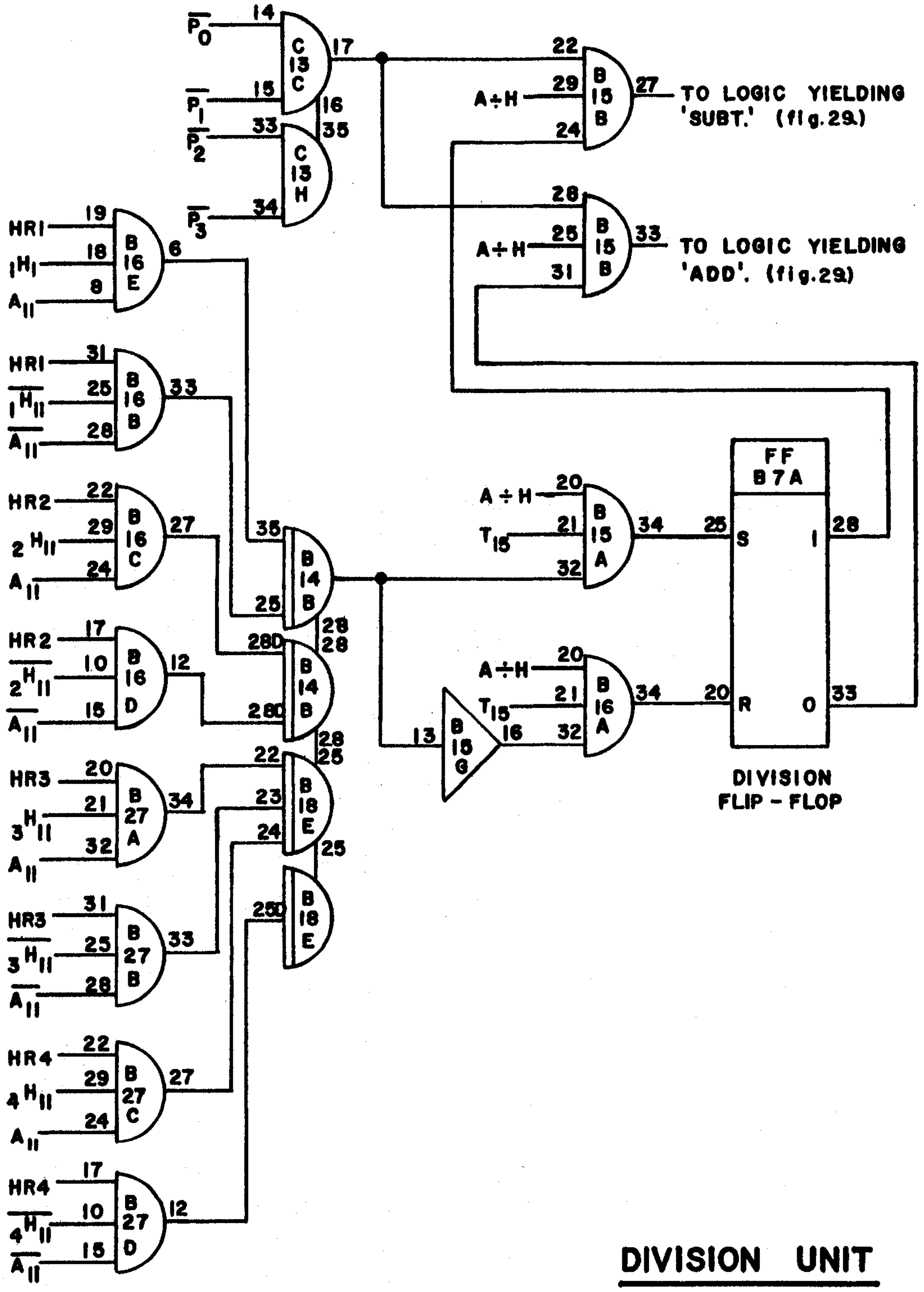
MULTIPLICATION UNIT

fig.29.



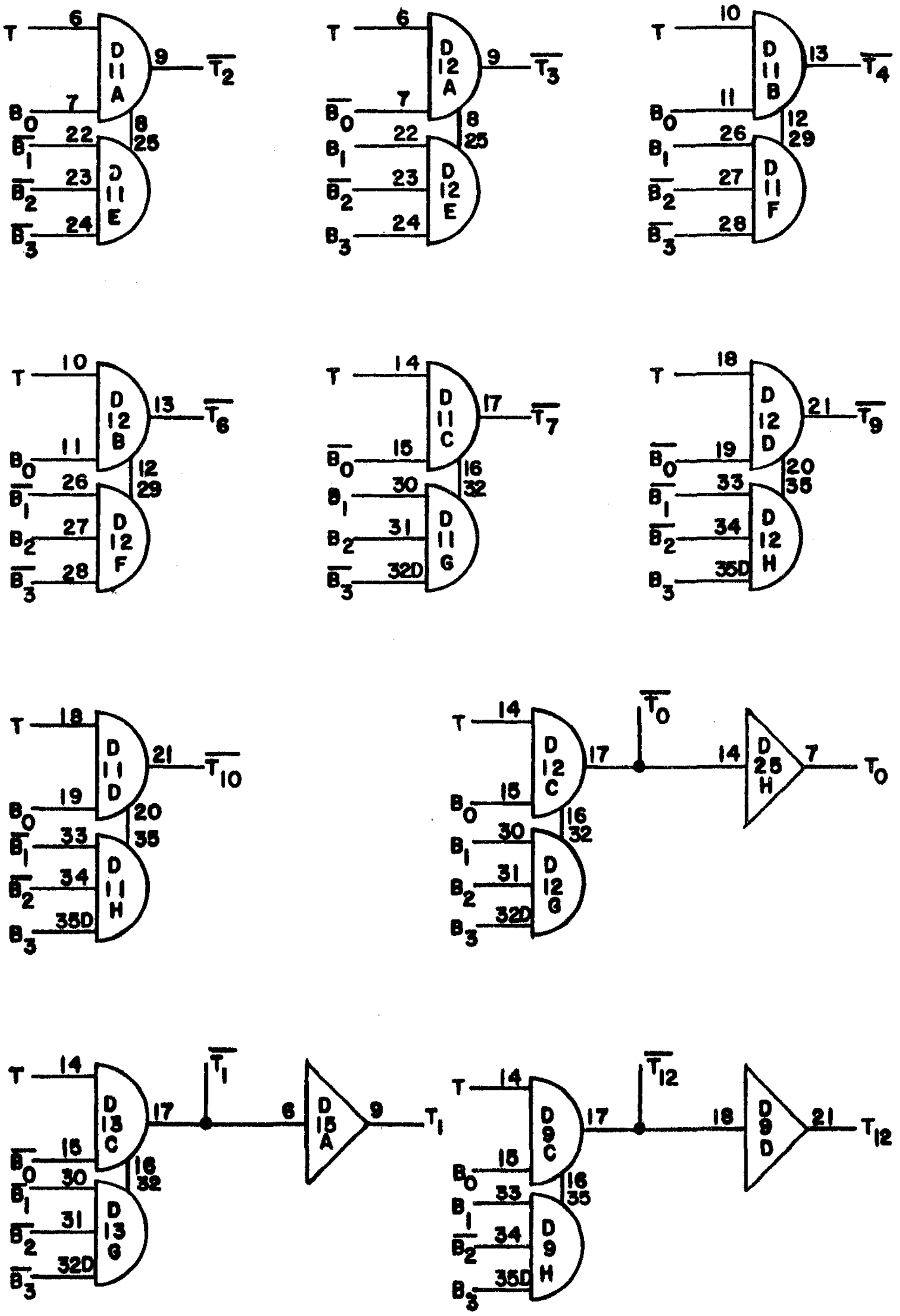
PHASE COUNTER

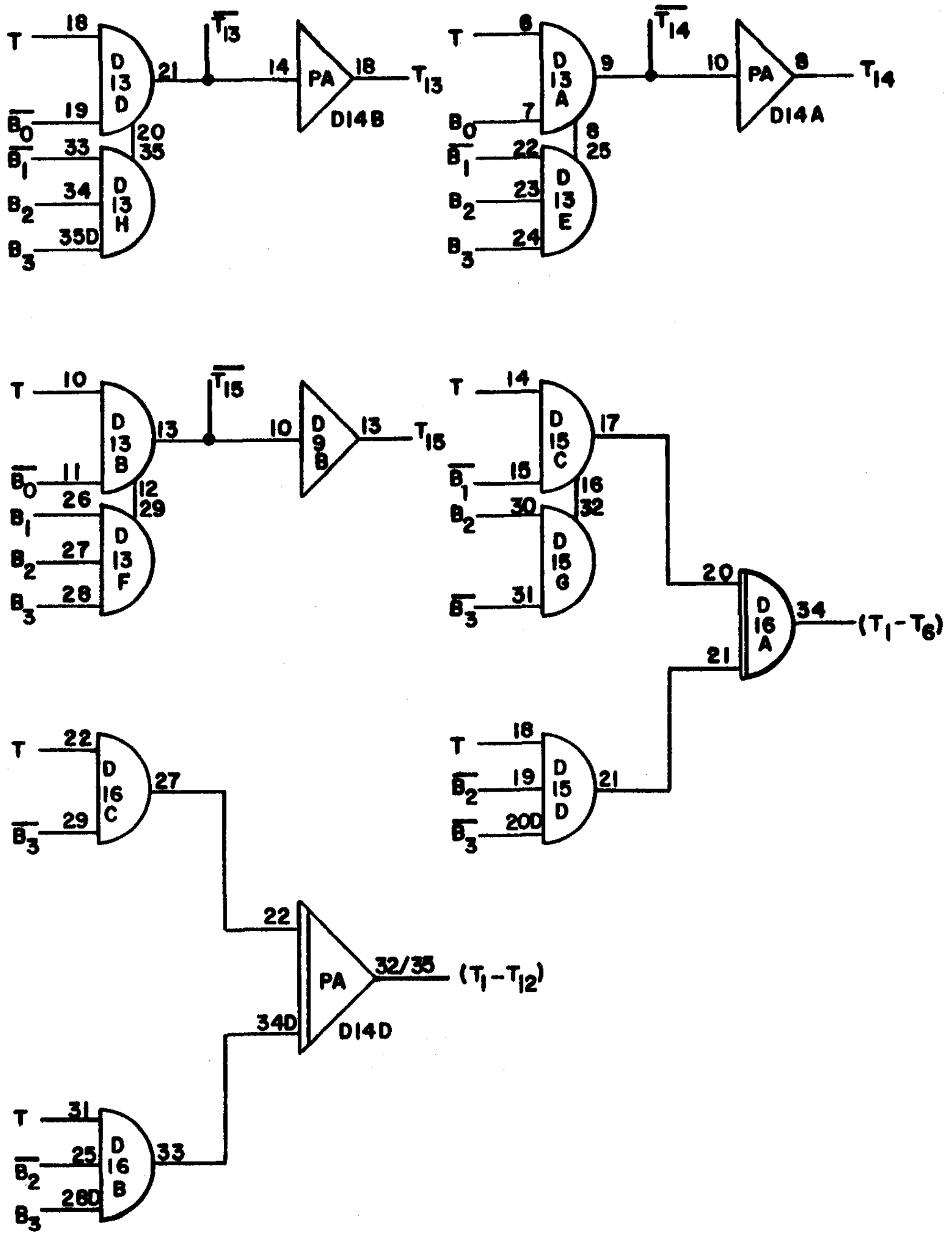
fig.30.



DIVISION UNIT

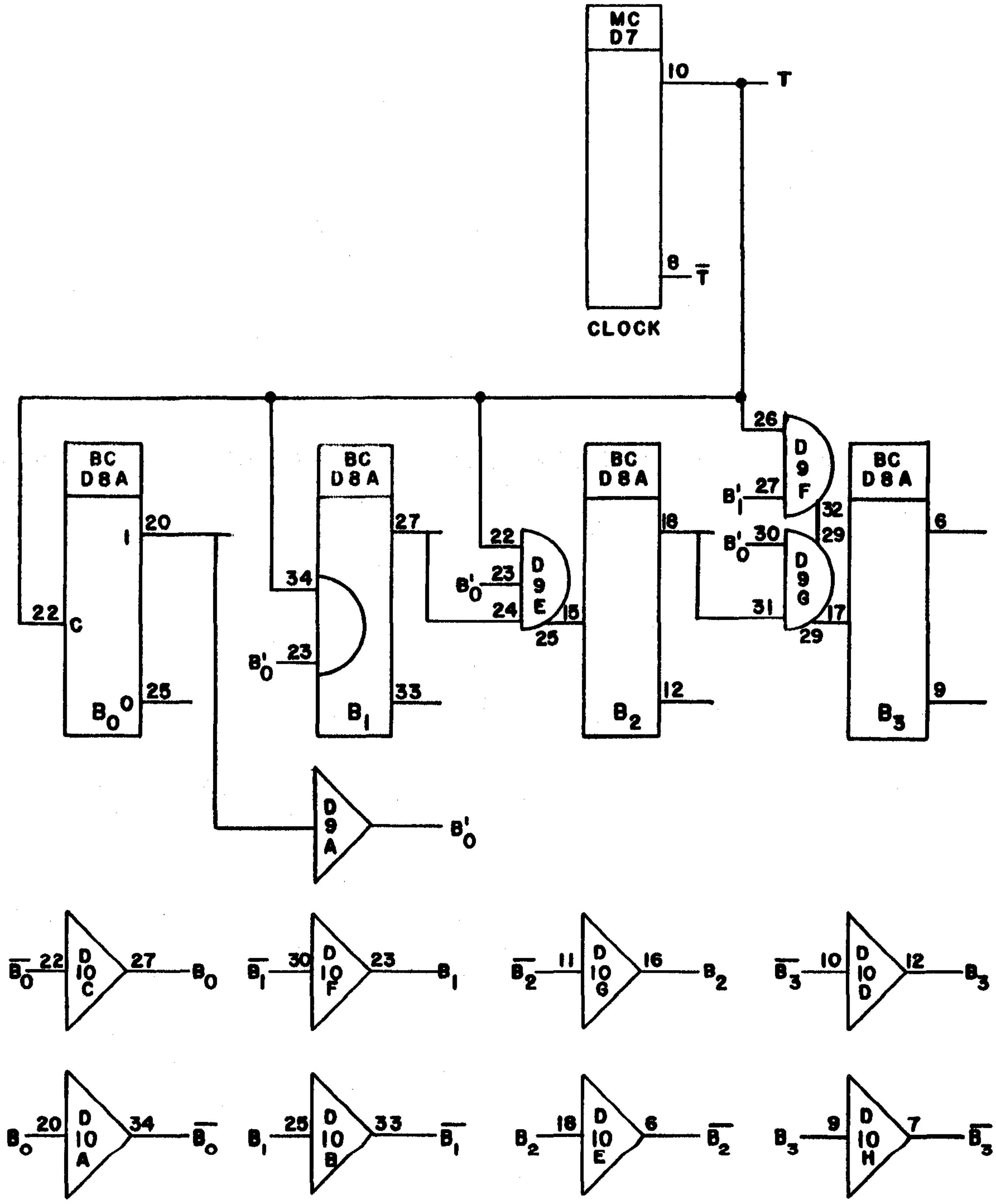
fig.31


T
6
D 11 A
9
$\overline{T_2}$
B_0
7
$\overline{B_1}$
22
8
25
D 11 E
$\overline{B_2}$
23
$\overline{B_3}$
24
T
6
D 12 A
9
$\overline{T_3}$
$\overline{B_0}$
7
8
25
B_1
22
D 12 E
$\overline{B_2}$
23
B_3
24
T
10
D 11 B
13
$\overline{T_4}$
B_0
11
12
29
B_1
26
D 11 F
$\overline{B_2}$
27
$\overline{B_3}$
28
T
10
D 12 B
13
$\overline{T_6}$
B_0
11
12
29
$\overline{B_1}$
26
D 12 F
B_2
27
$\overline{B_3}$
28
T
14
D 11 C
17
$\overline{T_7}$
$\overline{B_0}$
15
16
32
B_1
30
D 11 G
B_2
31
$\overline{B_3}$
32D
T
18
D 12 D
21
$\overline{T_9}$
$\overline{B_0}$
19
20
35
$\overline{B_1}$
33
D 12 H
$\overline{B_2}$
34
B_3
35D
T
18
D 11 D
21
$\overline{T_{10}}$
B_0
19
20
35
$\overline{B_1}$
33
D 11 H
$\overline{B_2}$
34
B_3
35D
T
14
D 12 C
17
$\overline{T_0}$
14
D 25 H
7
T_0
B_0
15
16
32
B_1
30
D 12 G
B_2
31
B_3
32D
T
14
D 13 C
17
$\overline{T_1}$
6
D 15 A
9
T_1
$\overline{B_0}$
15
16
32
$\overline{B_1}$
30
$\overline{B_2}$
31
D 13 G
$\overline{B_3}$
32D
T
14
D 9 C
17
$\overline{T_{12}}$
18
D 9 D
21
T_{12}
B_0
15
16
35
B_1
33
$\overline{B_2}$
34
D 9 H
B_3
35D



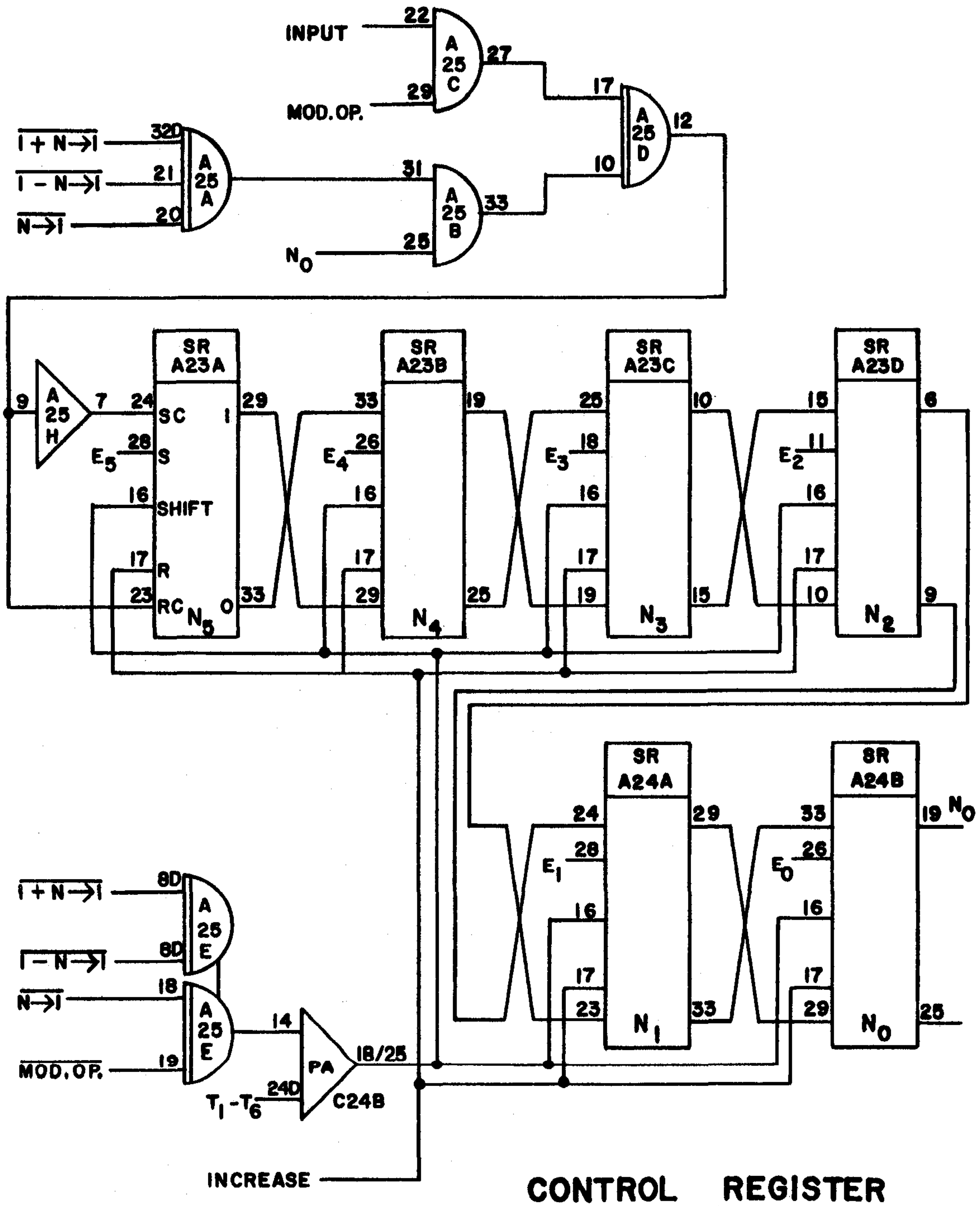
TIMING UNIT

fig.32.



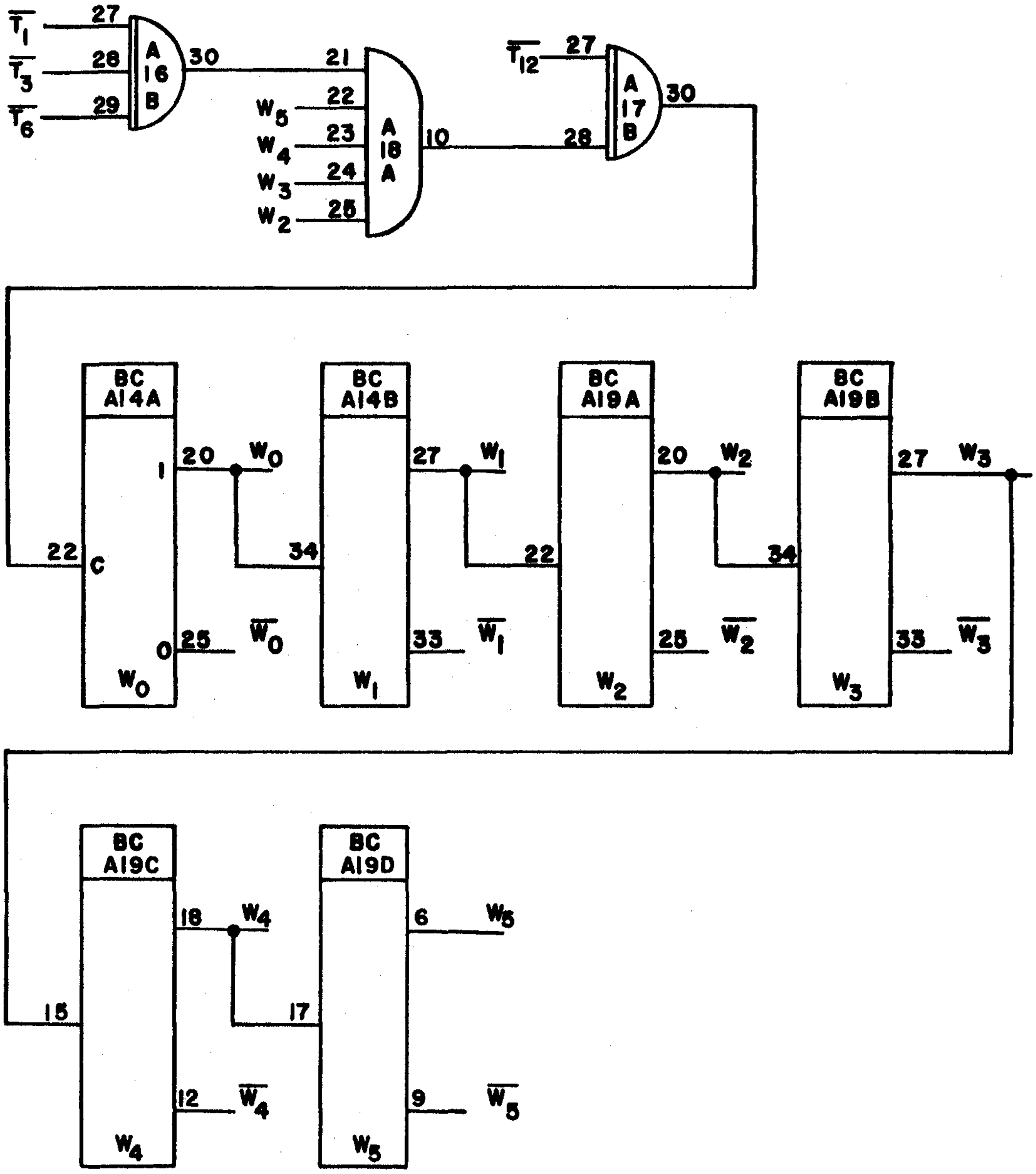
BIT COUNTER

fig.33.



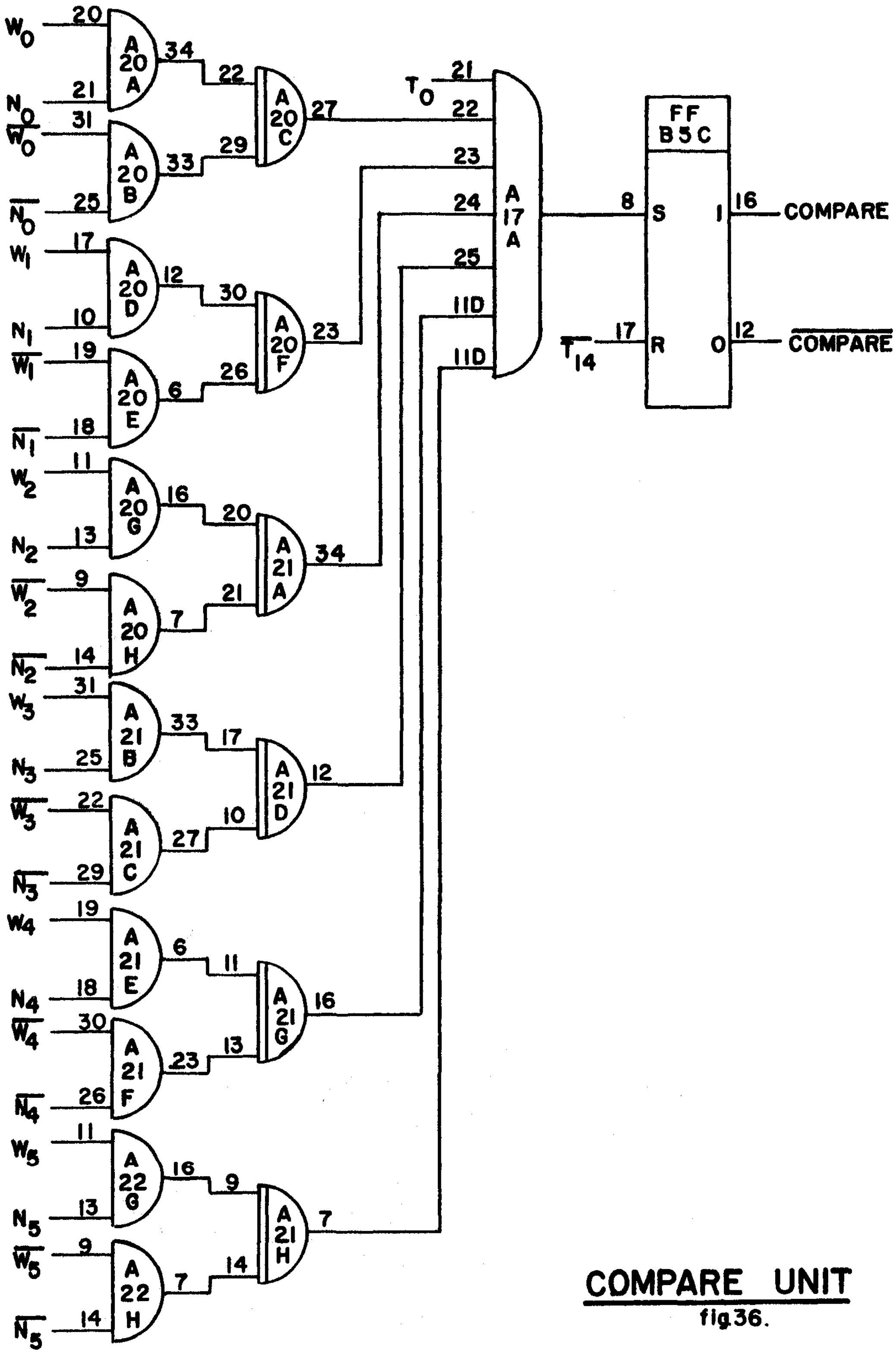
CONTROL REGISTER

fig.34.

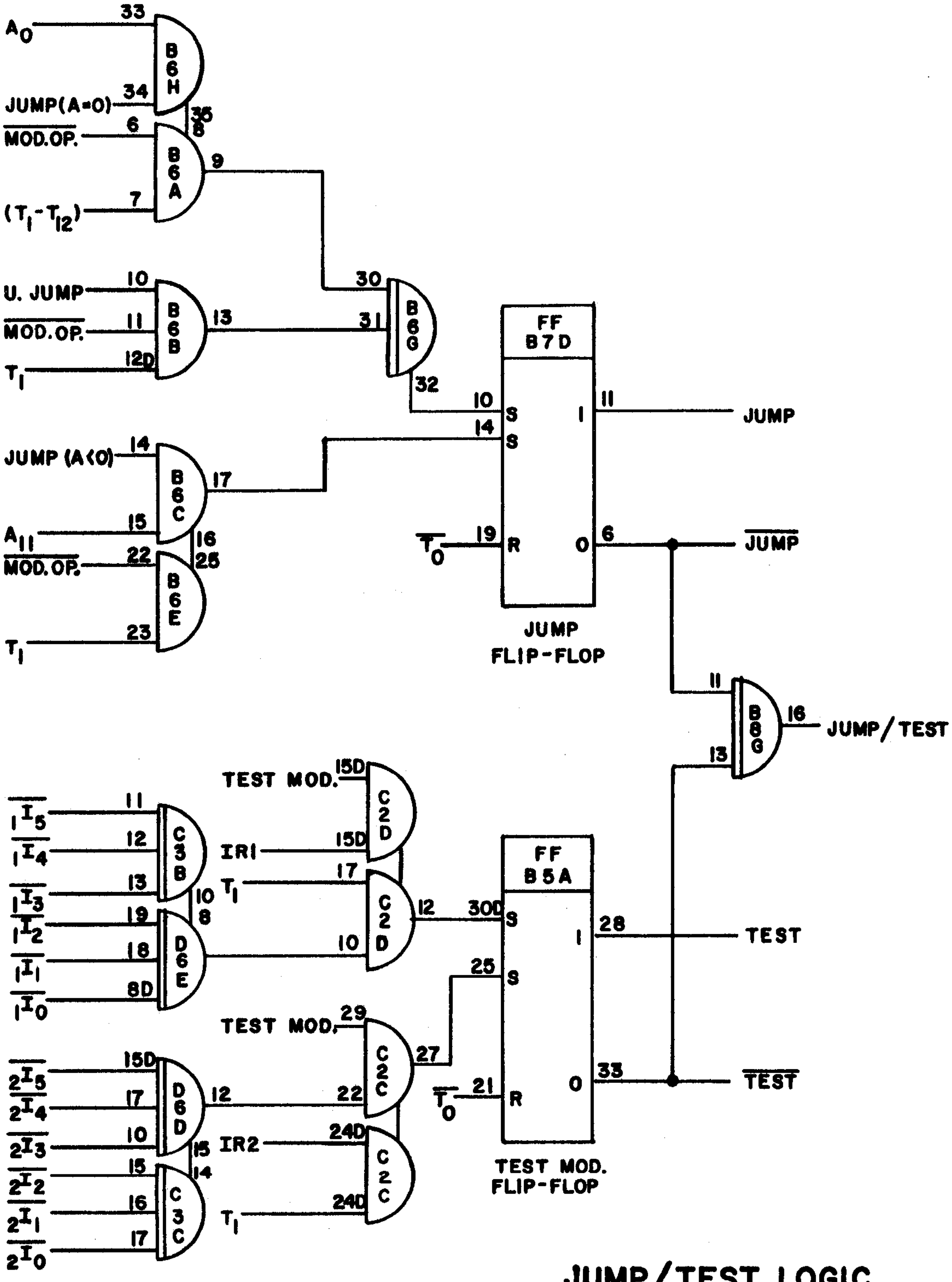


WORD COUNTER

fig.35.

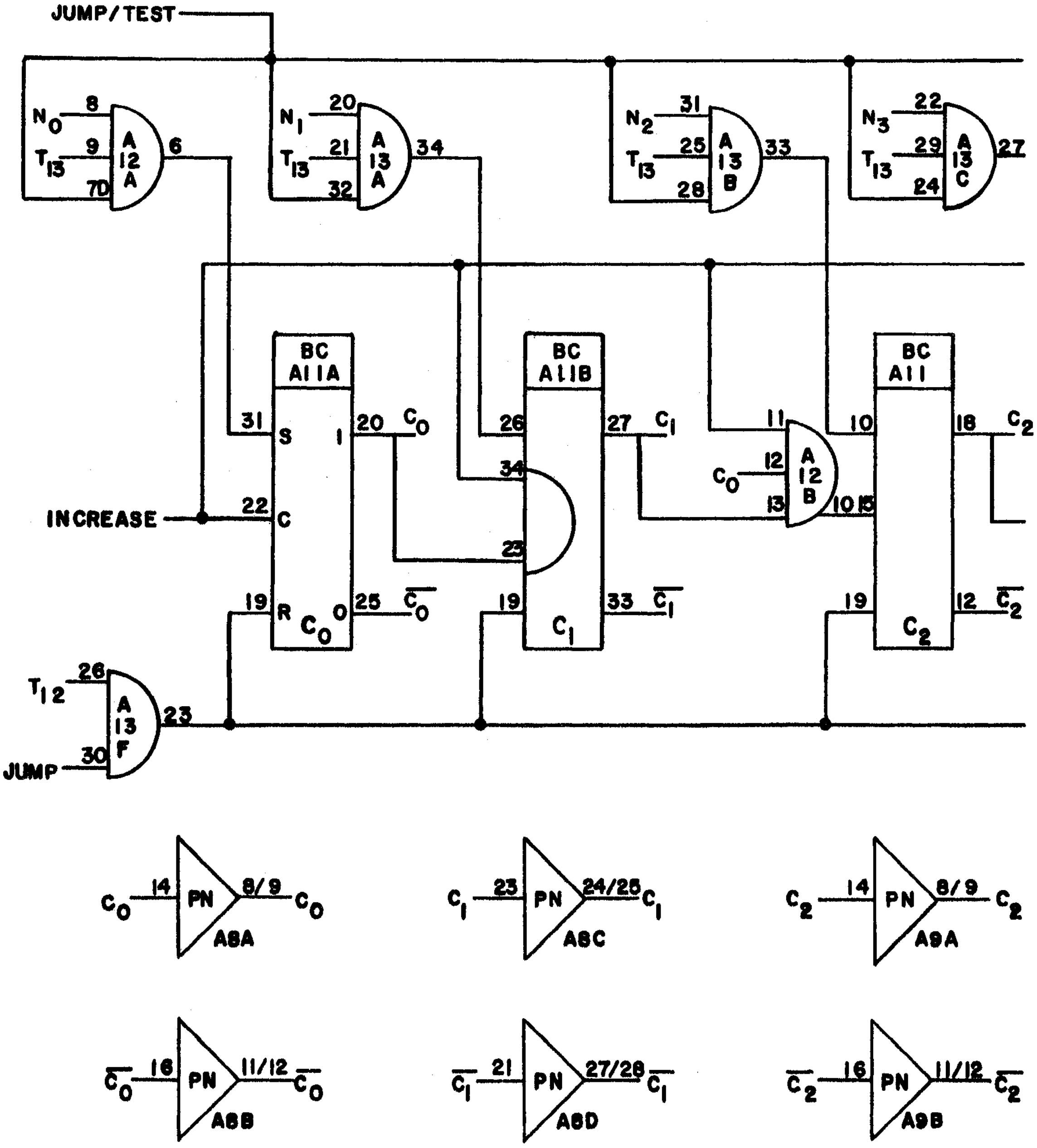


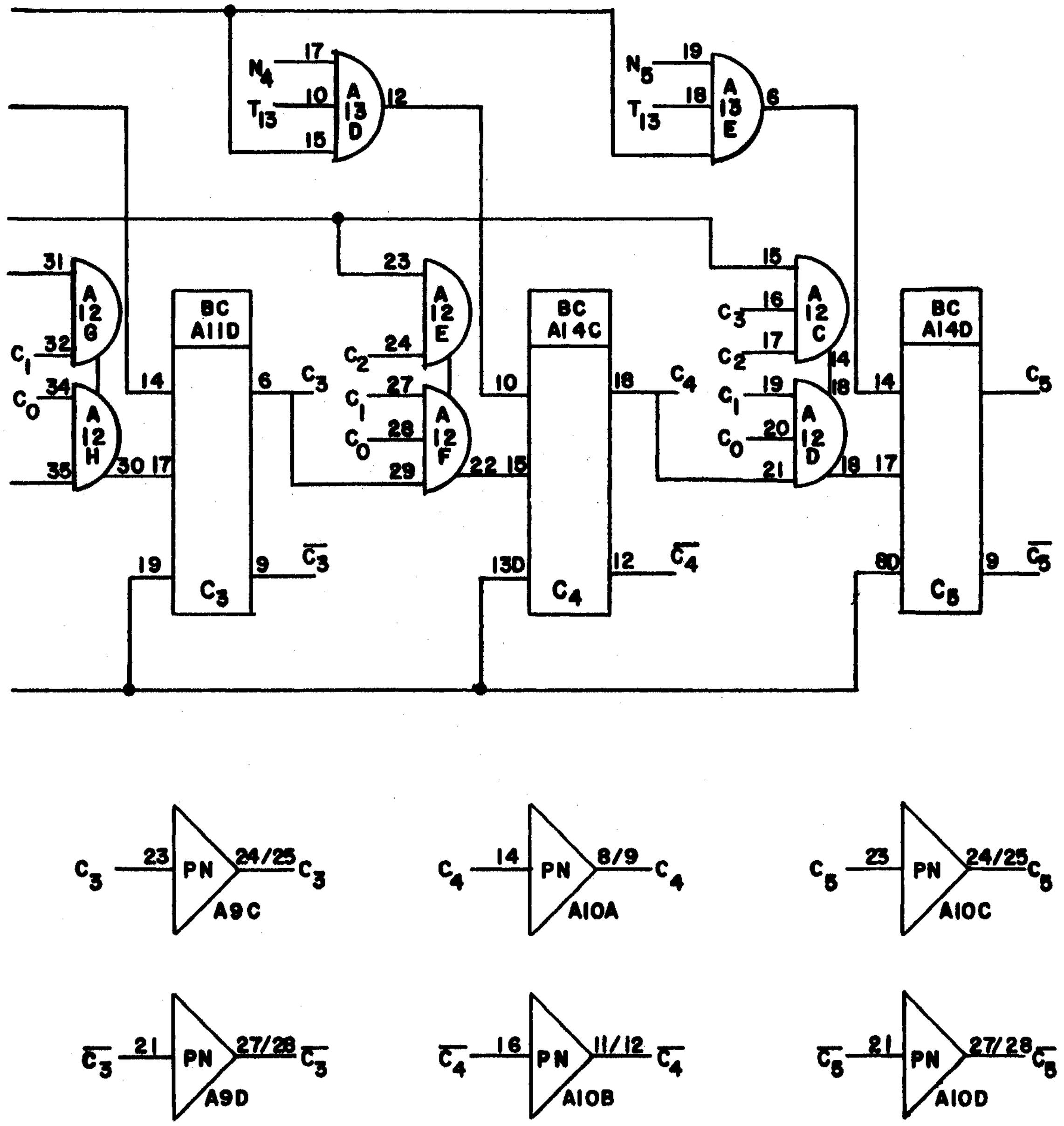
COMPARE UNIT
fig.36.



JUMP/TEST LOGIC

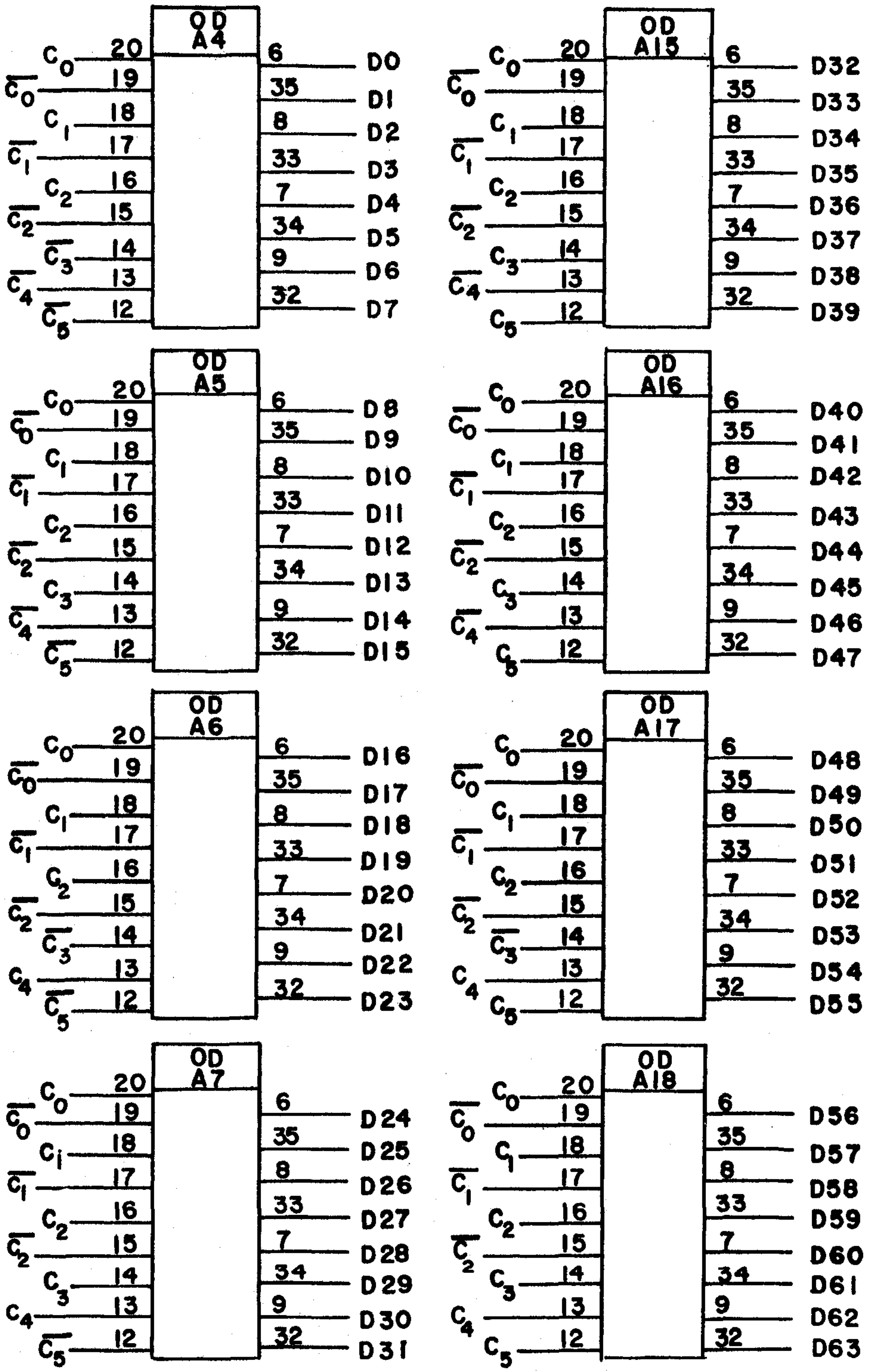
fig.37


JUMP/TEST
N0
T13
8
9
70
A 12 A
6
N1
T13
20
21
32
A 13 A
34
N2
T13
31
25
28
A 13 B
33
N3
T13
22
29
24
A 13 C
27
BC A11A
31
S
1
20
C0
INCREASE
22
C
19
R
C0
0
25
C̄0
BC A11B
26
34
23
27
C1
19
C1
33
C̄1
11
C0
12
13
A 12 B
10
BC A11
10
15
18
C2
19
C2
12
C̄2
T12
26
JUMP
30
A 13 F
23
C0
14
PN
8/9
C0
A8A
C1
23
PN
24/25
C1
A8C
C2
14
PN
8/9
C2
A9A
C̄0
16
PN
11/12
C̄0
A8B
C̄1
21
PN
27/28
C̄1
A8D
C̄2
16
PN
11/12
C̄2
A9B



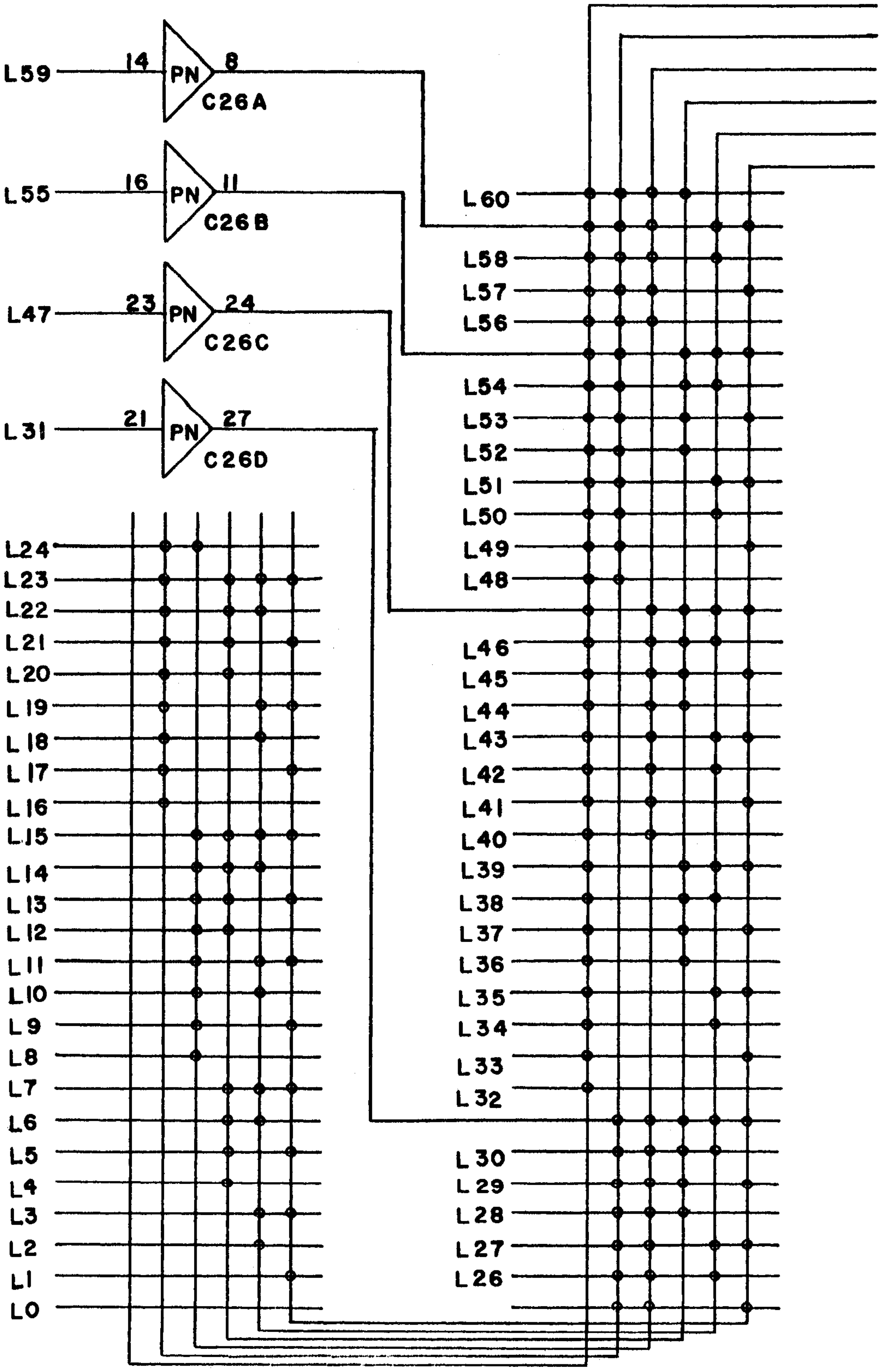
INSTRUCTION COUNTER

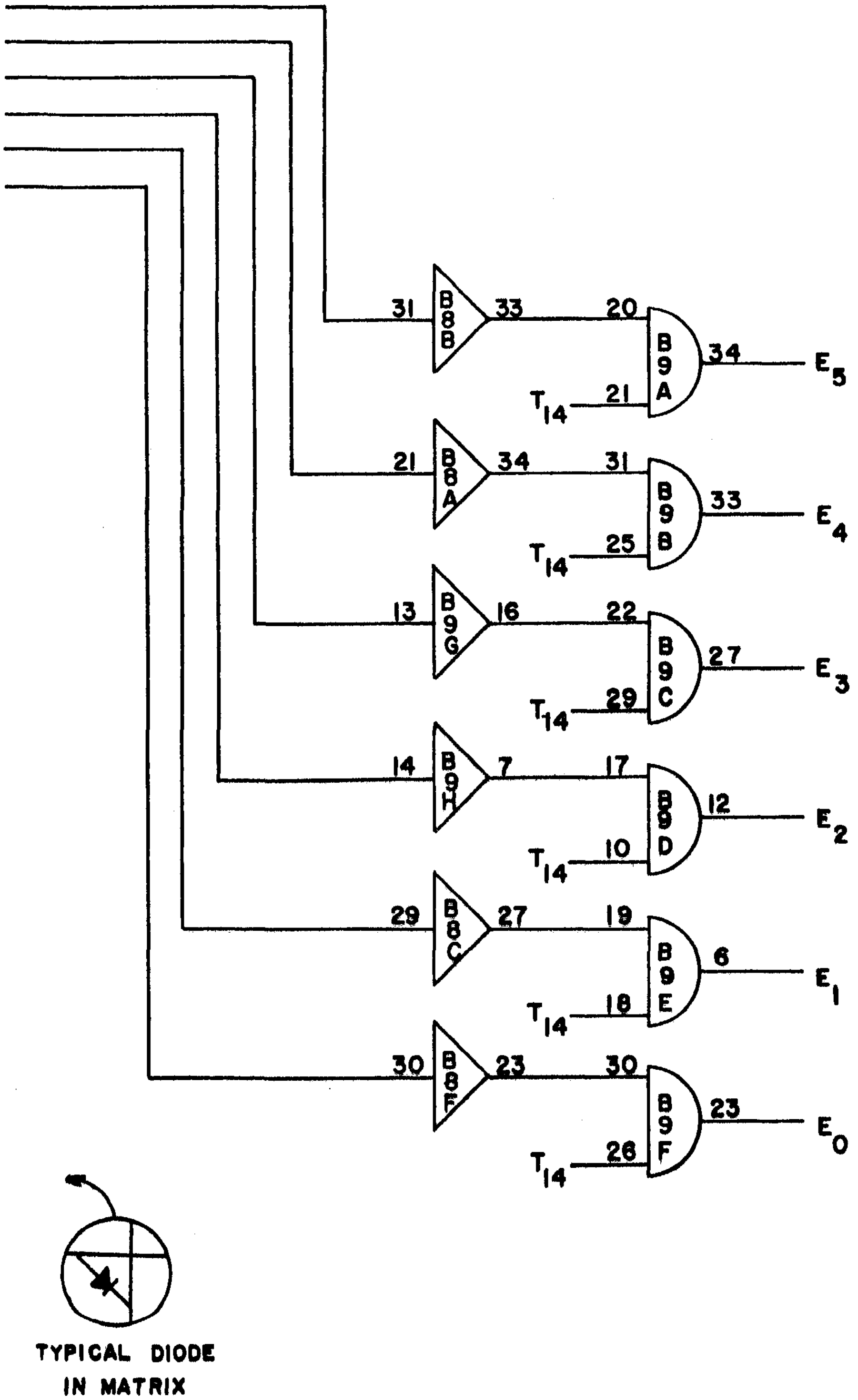
fig.38.



INSTRUCTION NUMBER DECODER

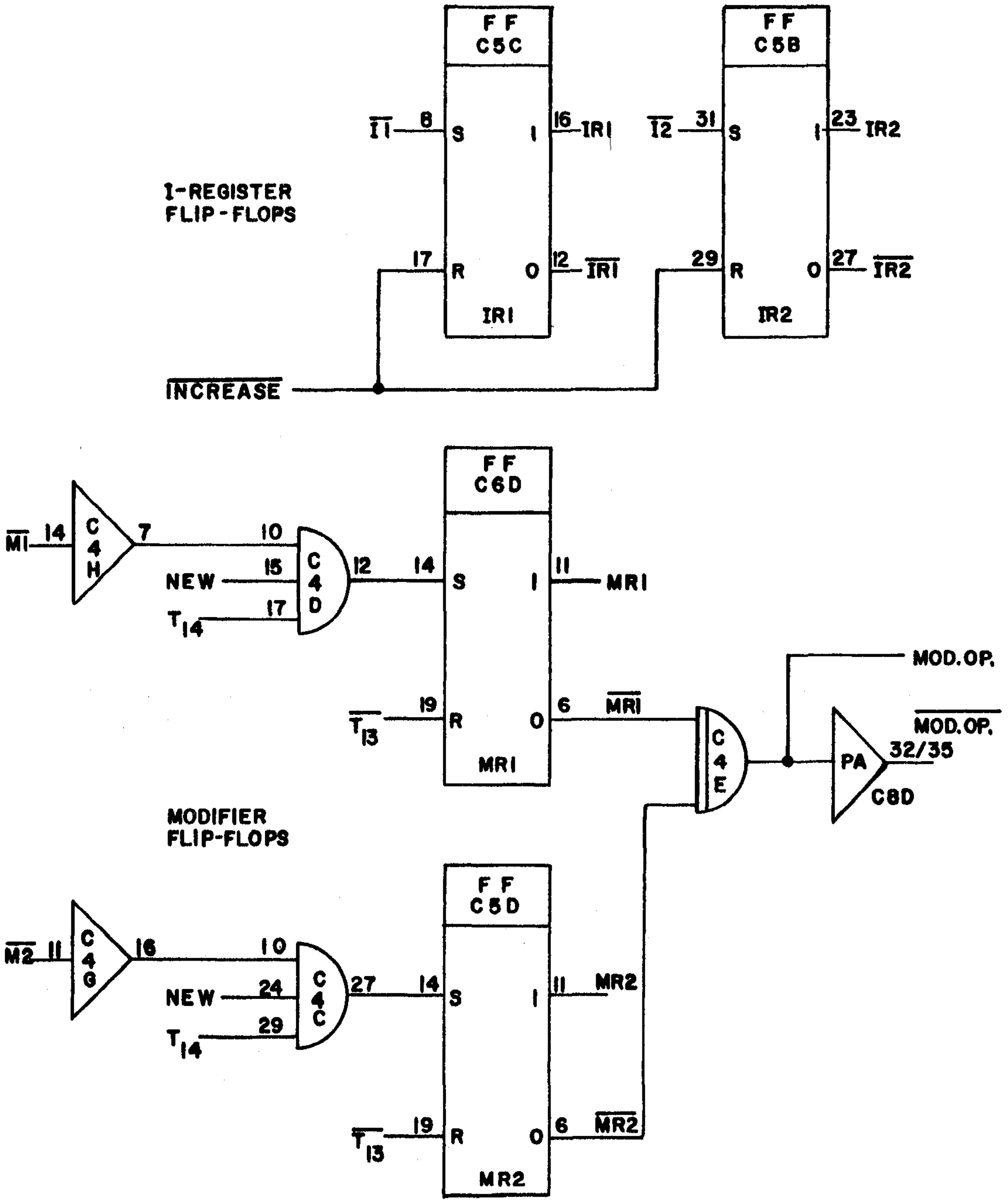
fig. 39.


L59
14
PN
8
C26A
L55
16
PN
11
C26B
L47
23
PN
24
C26C
L31
21
PN
27
C26D
L60
L58
L57
L56
L54
L53
L52
L51
L50
L49
L48
L46
L45
L44
L43
L42
L41
L40
L39
L38
L37
L36
L35
L34
L33
L32
L30
L29
L28
L27
L26
L24
L23
L22
L21
L20
L19
L18
L17
L16
L15
L14
L13
L12
L11
L10
L9
L8
L7
L6
L5
L4
L3
L2
L1
L0



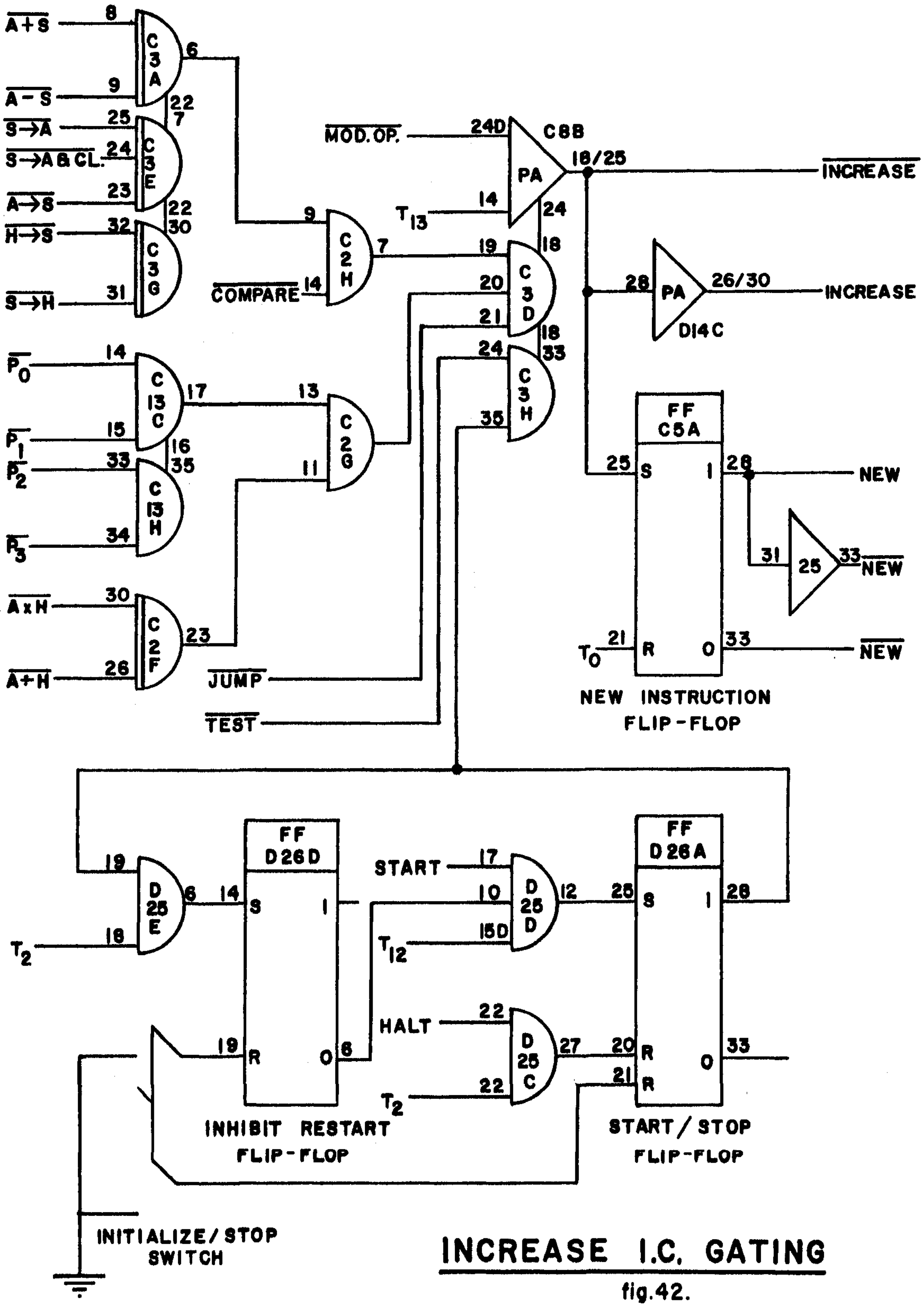
LOCATION ENCODER

fig.40.



MODIFICATION

fig.41



INCREASE I.C. GATING

fig.42.

V. CONCLUSIONS AND RECOMMENDATIONS

A complete digital control computer has been designed; a large part of which has now been constructed, and tested. It is felt that the computer achieves its goal of flexibility. The list of available orders can be extended to include other orders if required. With the present number of H-registers, and one delay line, suitable working and data storage is achieved. The number of H-registers and delay lines can easily be increased if necessary.

REFERENCES

1. RAYZAK, R. J.; A Digital Controller Strategy for Optimization and Adaptation of Control Systems Representable in the Phase Plane, M. A. Sc. Thesis, University of Windsor (1964).

2. CHU, Yaohan; Digital Computer Design Fundamentals, Mc Graw-Hill Book Company, Inc., Toronto (1962).

3. THOMAS, P. A. V.; The Design Philosophy of a Small Electronic Automatic Digital Computer, Ph. D. Thesis, University of Glasgow (1961).

4. STAFF OF COMPUTER CONTROL COMPANY, INC.; Instruction Manual for 200-KC S-PAC Digital Modules, Computer Control Company, Inc., Framingham, Massachusetts (1963).

APPENDIX

COMPUTER INSTRUCTION LIST

Note: Unless specified, an operand source remains unaltered.

1. H→A Transfer the contents of specified H-register (H1, H2, H3, H4) to A-register.
2. A→H Transfer contents of A-register into specified H-register.
3. A→H & CL.A Transfer contents of A-register to specified H-register and clear A-register to zero.
4. A + H→A Add number in the specified H-register to number in the A-register. Put sum in A-register.
5. A − H→A Subtract number in the specified H-register to number in the A-register. Put difference in A-register.
6. A x H→A Multiply the number in the A-register by the number in the specified H-register and place the product into the A-register.
7. A ÷ H→A Divide the number in the A-register by the number in the specified H-register and place the quotient into the A-register.
8. (spare)
9. S→H Transfer contents of given location (L0 through L60) in store S to specified H-register.

10. H→S — Transfer contents of specified H-register to given location in store S.

11. S→A — Transfer contents of given location in store S to specified H-register.

12. A→S — Transfer contents of A-register to given location in store S.

13. A→S & CL.A — Transfer contents of A-register to given location in store S and clear A-register to zero.

14. A + S→A — Add number in given location of store S to number in A-register and place sum in A-register.

15. A − S→A — Subtract number in given location of store S from number in A-register and place the difference in A-register.

16. (spare)

17. Right Shift A — Shift contents of A-register one bit to the right. Repeat the sign bit.

18. Left Shift A — Shift contents of A-register one bit to the left. Repeat the sign bit.

19. Jump (A <0) — If the number in A-register is less than zero, jump to the given program step (given by L0 through L60). If not, go to the next consecutive program step.

20. Jump (A = 0) — If the number in the A-register is equal to zero, jump to the given program step. If not, go to the next consecutive program step.

21. N→I — Transfer the binary equivalent of an integer, N (given by LO through L60), into specified I-register (I1 or I2).

22. I + N→I — Add the integer, N, to number in specified I-register and place the sum into the same register.

23. I - N→I — Subtract the integer, N, from the number in the specified I-register and place the difference into the same register. Also, if the result of (I - N) is zero, set the TEST FLIP-FLOP to ONE. (Note: this is for location modification.)

24. (spare)

25. Test Modifier — If TEST FLIP-FLOP is ZERO, jump to the given program step (given by LO through L60). If TEST FLIP-FLOP is ONE, go to the next consecutive program step.

26. (spare)

27. Input — Transfer data from D-register (digitizer register on the shaft position encoder) to A-register.

28. Output — Set OUTPUT FLIP-FLOP (1 or 2) to ONE. (Note: these flip-flops can be used to gate control signals in any external on-line system.)

29. Unconditional Jump — Jump to the given program step (given by LO through L60).

30. (spare)

31. Halt — Stop the execution of the program at the Halt program step.

32. (spare)

COMPUTER BUTTONS

CLEAR ACCUMULATOR — Set A-register to zero

STOP and INITIALIZE — Reset START-STOP logic. This button must preceed START button at all times.

ZERO COUNTER — Set Instruction counter to zero.

START — Increase Instruction Counter by 1 and allow counter to continue counting until a Halt order is reached.

ACCUMULATOR INPUTS (12 buttons A_{11} - A_0) — Input data to A-register from a keyboard. Each button sets corresponding stage of A-register to ONE.

VITA AUCTORIS

1940 Born on July 18, in Windsor, Ontario.

1954 Completed elementary education at St. Bernard Elementary School, Windsor, Ontario.

1959 Completed Grade XIII at Walkerville Collegiate Institute, Windsor, Ontario.

1963 Graduated from Assumption University, Windsor, Ontario, with degree of B.A.Sc. in Electrical Engineering.

1964 Candidate for degree of M.A.Sc. degree in Electrical Engineering at the University of Windsor.