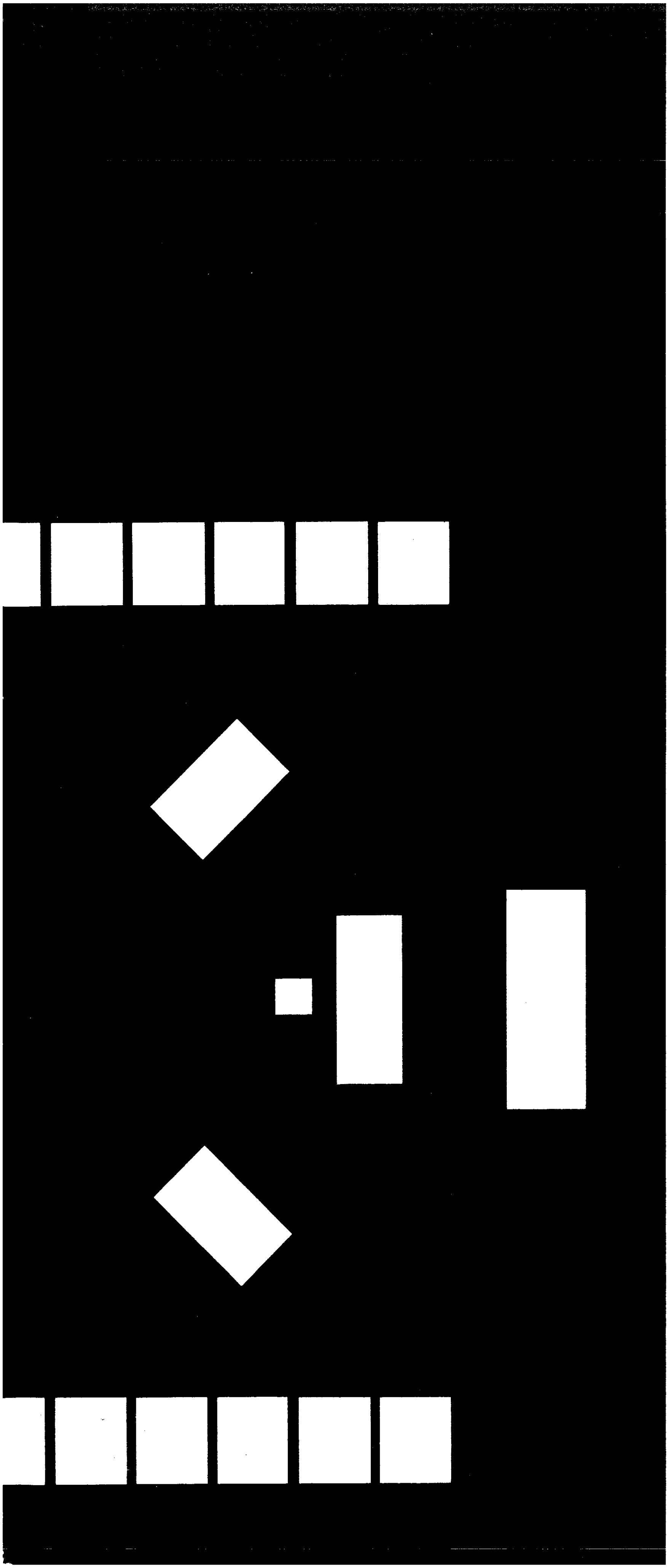


AN INTRODUCTION TO

DIGITAL COMPUTERS

Volume II

CONTROL DATA INSTITUTES

CONTROL DATA
CORPORATION

An Introduction to DIGITAL COMPUTERS

volume 2

FOR TRAINING PURPOSES ONLY

This manual was compiled and written by members of the instructional staff of

CONTROL DATA INSTITUTE
CONTROL DATA CORPORATION

CDI 60241700
January 1967

Former publication number: 091266A.

Foreword

Now that you have completed Volume I, you should understand number systems and how a computer is programmed in machine language, assembler language, and compiler language. You know what to expect the computer to do, even though you may not know how it is done.

The overall objective of Volume II is to teach you how the computer operates internally. This objective will be accomplished if you understand each subject as it is discussed and how that subject forms an integral part of the computer concept.

The order of subjects will be Boolean Algebra, Logic Circuitry, and the four basic sections of any digital computer (Arithmetic, Memory, Control, and Input/Output).

General Table of Contents

Contents

CHAPTER V BOOLEAN ALGEBRA

CHAPTER VI INTRODUCTION TO LOGIC CIRCUITRY

CHAPTER VII ARITHMETIC SECTION OF A COMPUTER

CHAPTER VIII COMPUTER STORAGE

CHAPTER IX CONTROL SECTION

CHAPTER X INPUT-OUTPUT SECTION

chapter v

Boolean Algebra

CHAPTER V

BOOLEAN ALGEBRA

INTRODUCTION

A digital computer is comprised of thousands of logic circuits, each capable of indicating yes or no, true or false, or logical "1" or logical "0". The output of a given circuit may be a blip on an oscilloscope to the Customer Engineer, a light on the console to the computer operator, or an astounding spectacle to a casual observer. A logical "1" from a given circuit represents an equation or combination of terms which can be analyzed by using Boolean Algebra much the same as an ordinary equation is analyzed by using ordinary algebra.

Boolean Algebra is a unique logical science and is, therefore, governed by a set of parametric laws, or statements.

The objectives of this Chapter are to teach you the fundamental concepts of Boolean Algebra, the laws that circumscribe its boundaries, and the manner in which logical circuits can be analyzed to determine output equations. This chapter will also teach you how to use the Veitch Diagram to simplify a Boolean expression but only after you understand t he laws which justify the diagram.

A HISTORY OF BOOLEAN ALGEBRA

In 1854 an English mathematician, George Boole, composed the treatise, An Investigation of the Laws of Thought on Which are Found the Mathematical Theories of Logic and Probabilities. This treatise was to perform a mathematical analysis of logic. His objective is perhaps best introduced by the following quotation from his first chapter:

> The design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed; to give expression to them in the symbolical language of a calculus, and upon this foundation to establish the science of Logic and to construct its method; to make that method itself the basis of a general method for the application of the mathematic doctrine of probabilities; and, finally, to collect from the various elements of truth brought to view in the course of these inquires some probable intimations concerning the nature and constitution of the human mind.[1]

[1] Boole, George, An Investigation of the Laws of Thought, (Dover Publications, N. Y., 1951). A reprint of the original book, published in 1854.

As a result of his investigation and the construction of a logical algebra, subsequent mathematicians and logicians were led into several new fields of mathematics. From the philosophy to the maturing mechanics developed an algebra which is now used in the design of digital computer logic circuitry. This algebra is called Boolean Algebra.

In 1938 a research assistant, Claude E. Shannon, in the department of electrical engineering at Massachusetts Institute of Technology, suggested that Boolean Algebra be used to solve problems in relay switching circuit design. This method was suggested in his M.S. degree thesis entitled, "A Symbolic Analysis of Relay and Switching Circuits". This paper presented a method for representing any circuit consisting of combinations of switches and relays by a set of mathematical expressions. The calculus used was shown to be exactly analogous to the calculus of propositions used in the field of symbolic logic. The basic techniques described by Shannon were adopted almost universally for the design and analysis of switching circuits. Because of the analogous relationship between the action of relay, vacuum tube, and transistor circuits, the same techniques were developed for the design of the switching circuits used in modern high speed computers.

There are several advantages in having a mathematical technique for the description of switching circuits. For one thing, it is far more convenient to calculate with expressions used to represent switching circuits than it is to use schematic or even logic diagrams. Just as an ordinary algebraic expression may be simplified by means of the basic theorems, the expression describing a given switching circuit network may also be reduced or simplified. This enables the designer of the logic to simplify the circuitry used, achieving economy of construction and reliability of operation. Boolean Algebra also provides an economical and straightforward way of describing the circuitry used in computers. In all, a knowledge of Boolean Algebra is indispensable in the computer field.

There are some basic differences between Boolean Algebra and ordinary algebra, the letter symbols may take a large number of values whereas in Boolean Algebra, they may assume only one of two possible values (consistant with the binary number system). Because of the binary nature of the variables involved, Boolean Algebra is much simpler than ordinary algebra. In ordinary algebra, the values have a numerical significance; in Boolean Algebra, only a logical significance.

In order to further illustrate the value and principles of Boolean Algebra, the following self-teaching device, in the form of a story about baseball has been included. It shows the flexibility of Boolean Algebra, used in this case to explain the conditions surrounding a baseball game. It shows the brevity that can be gained for a given situation, as exemplified by the final equation developed for this baseball analogy.

The proper progression through this section is to read the first unshaded area, answer the questions, check your answers by referring to the shaded area on the following page as a review and then read the unshaded area on that page. This process is then to be repeated until the entire self-teaching portion has been completed. The last page is entirely shaded and serves as a review of the entire subject.

We'll go to the baseball game if the following factors are considered!

FACTORS you must consider before going to a baseball game are:

Do you have \$10.00 you could spend?

If you have \$10.00 to spend let's call it "A" to prevent having to rewrite this longer statement over and over and over.

A = \$10.00 (This could also be called the PRESENCE of \$10.00)

If you do not have \$10.00, then you do not have "A".

$\boxed{\text{Not "A" is written } \overline{A}}$ not A = $\overline{A}$ = $\overline{\$10.00}$

$\overline{A}$ = $\overline{\$10.00}$ (This could also be called the ABSENCE of \$10.00)

Another factor to be considered: Is it Saturday? Let "B" stand for Saturday. Thus:

B = Saturday

$\overline{B}$ = Not Saturday

We can then state: "We will go to the baseball game if we have \$10.00 AND it's Saturday!"

As long as we are shortening everything, let a dot ⊡ represent AND. How would the statement: "We will go to the baseball game if we have \$10.00 AND it's Saturday!" be written symbolically?

ANSWER: ______________________________

A and B A · B	Mean the same thing	We're going to the baseball game if we have $10.00 AND it's Saturday.

As for multiplication in regular algebra, the · is sometimes omitted and the relationship is simply inferred. Thus A and B, A·B and AB all represent the same condition. I'm not much for baseball, but if I have $10.00 AND it's Saturday AND the game is with the Yankees, I'll go!

How would this additional factor be represented in the entire equation if we again abbreviate and have "C" stand for Yankees?

ANSWER: ____

ANSWER:

ABC - We're going to go: We have $10.00 AND it's Saturday AND the game is with the Yankees.

Review:

$$\left.\begin{array}{l} \text{A and B} \\ A \cdot B \\ AB \end{array}\right\} \quad \text{All mean the same thing} \quad \left\{\begin{array}{l} \text{We are going to the baseball game if} \\ \\ \text{we have \$10.00 AND it's Saturday.} \end{array}\right.$$

What if it's raining? Letting "D" stand for raining, we can say:

"ABC$\overline{D}$; if all are true, or present, we'll go".

Notice the $\overline{D}$ in the expression. We said that "D" stood for raining; thus, we'll go if it's NOT raining or $\overline{D}$.

ABC$\overline{D}$ means: "I'll go to the baseball game if I have $10.00 AND it's Saturday AND the game is with the Yankees AND it's not raining.

What would the expression be if I said I would go if I had $10.00, it was Saturday, the game was with the Yankees, and it was not raining OR I would go if I had $10.00, it was Saturday, the game was with the Yankees, and it was raining and I had a raincoat.

Let "E" = raincoat and the plus sign $\boxplus$ represent OR

Write the new expression here____

ANSWER:

$ABC\overline{D}$ = I'll go. (I have $10.00 AND it's Saturday AND Yankees AND it's not raining.

$ABCDE$ = I'll go. (I have $10.00 AND it's Saturday AND Yankees AND raining AND I have a raincoat:)

OR

$ABC\overline{D} + ABCDE$ = I'll go as long as either the top statement (called the first TERM) or the bottom statement (called the second TERM) is true:

Review:

$$\left.\begin{array}{l} \text{A and B} \\ A\cdot B \\ AB \end{array}\right\} \quad \text{All mean the same thing} \quad \left\{\begin{array}{l} \text{We're going to the baseball game if} \\ \\ \text{we have \$10.00 AND it's Saturday} \end{array}\right.$$

ABC - We're going to go: We have $10.00 AND it's Saturday AND the game is with the Yankees.

Everything else has been abbreviated, let's abbreviate the presence or absence of A condition.

If something is true or present call it a "1"

If something is false or absent call it a "0"

$ABC\overline{D}$ = 1 and 1 and 1 and 1 = 1 (do not add, just consider 1 and 1 = 1)

Note that the 1's and 0's are used to indicate the state of the individual factors or items as well as the validity of the entire statement.

REVIEW how B and D were defined if you have difficulty with the last statement.

Following our baseball story and the use of 1's and 0's, evaluate the expression $\overline{A}BC\overline{D}$ which means:

I do <u>not have $10.00</u> and the game is with the <u>Yankees</u> on a <u>sunny Saturday</u>.

ANSWER: ____

ANSWER:

$AB C\overline{D} = 0$ If all or any conditions are absent we do not go to the game.

In this case A = 0, B = 1, D = 0. Therefore, the entire expression is 0 since $0 \cdot 1 \cdot 1 \cdot 1 = 0$ because not all the factor were true.

Review:

A and B
$A \cdot B$
AB

All mean the same thing

We're going to the baseball game if we have $10.00 AND it's Saturday.

ABC - We're going to go: We have $10.00 AND it's Saturday AND the game is with the Yankees.

$ABC\overline{D}$ = I'll go. (I have $10.00 AND it's Saturday AND Yankees AND it's not raining)

ABCDE = I'll go. (I have $10.00 AND it's Saturday AND Yankees AND raining AND I have a raincoat.

OR

$ABC\overline{D} + ABCDE$ = I'll go as long as either the first term OR the second term is true.

$ABC\overline{D} + ABCDE = 1$ If all conditions are present, then we go to the game.

If an entire statement is equal to "1", at least one term is true or satisfied (equal to 1).

If an entire statement is equal to "0", all terms are false or unsatisfied (equal to "0").

How would the statement $ABC\overline{D} + ABCDE$ have to be amended in order to include going to any world series game under any condition (use G to represent this condition).

ANSWER:____

ANSWER:

$ABC\overline{D} + ABCDE + G = 1$

Review:

A and B
A • B
AB

All mean the same thing

We're going to the baseball game if we have \$10.00 AND it's Saturday.

ABC - We're going to go: We have \$10.00 AND it's Saturday AND the game is with the Yankees.

$ABC\overline{D}$ = I'll go. (I have \$10.00 AND it's Saturday AND Yankees AND it's not raining)

$ABCDE$ = I'll go. (I have \$10.00 AND it's Saturday AND Yankees AND raining AND I have a raincoat.)

OR

$ABC\overline{D} + ABCDE$ = I'll go as long as either the first term OR the second term is true.

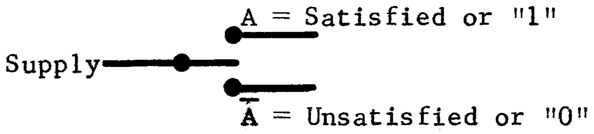
$ABC\overline{D} + ABCDE = 1$ If all factors within a term are present, then we go to the game. If at least one factor in each term is absent, we do not go to the game. $ABC\overline{D} = 0$

$ABC\overline{D} + ABCDE + G = 1$ (I have \$10.00 AND it's Saturday AND Yankees AND it's not raining) OR (I have \$10.00 AND it's Saturday AND Yankees AND raining AND I have a raincoat.) OR (It's the world series and I'm going anyway)

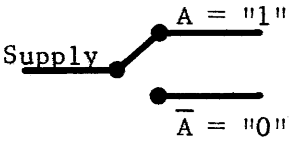
This problem has been represented through the use of LOGICAL ALGEBRA. This means of notation has greatly simplified the expression of the presence or the absence of the factors involved. This is the primary application of logical algebra in computers: To provide SIMPLIFICATION OF NOTATION.

Now try the following solution to the same problem presented in a similar format, but using electrical analogies.

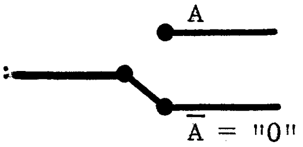
To relate the previous example to electronic circuits we can use the symbol for a switch that may be either closed and satisfied (1) or open and unsatisfied (0).



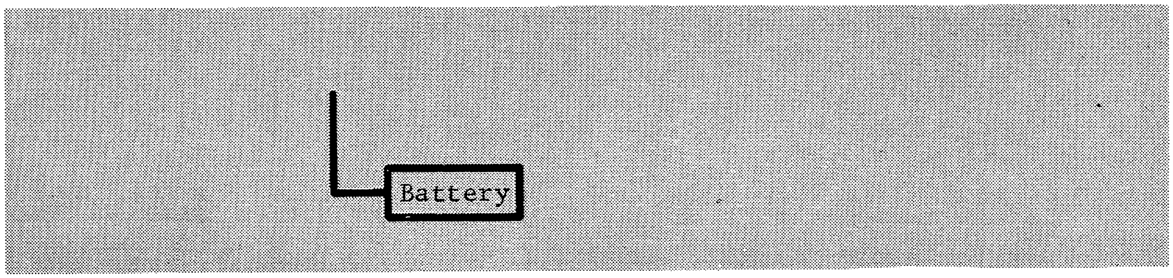
If "A" is satisfied (true), then the circuit exists as:



If "A" is not satisfied $(\bar{A})$, then the circuit exists as:

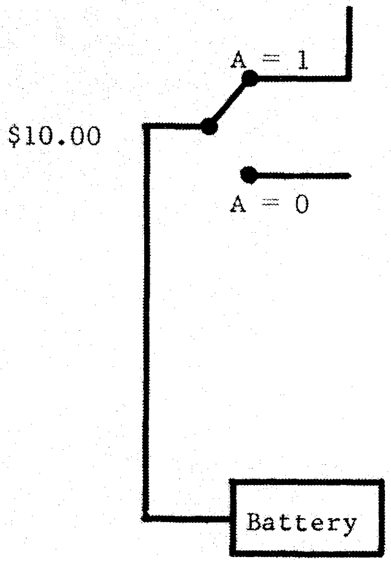


We made a statement that we would go to the ball game if we had $10.00 This was represented by A. How do you suppose this would be shown on a circuit? Indicate your answer by using the switch symbol and continuing the drawing up and to the left margin.



We have $10.00 (A). We also stated that if it was Saturday, (B), AND if the game was with the Yankees, (C), we would go to the ball game.

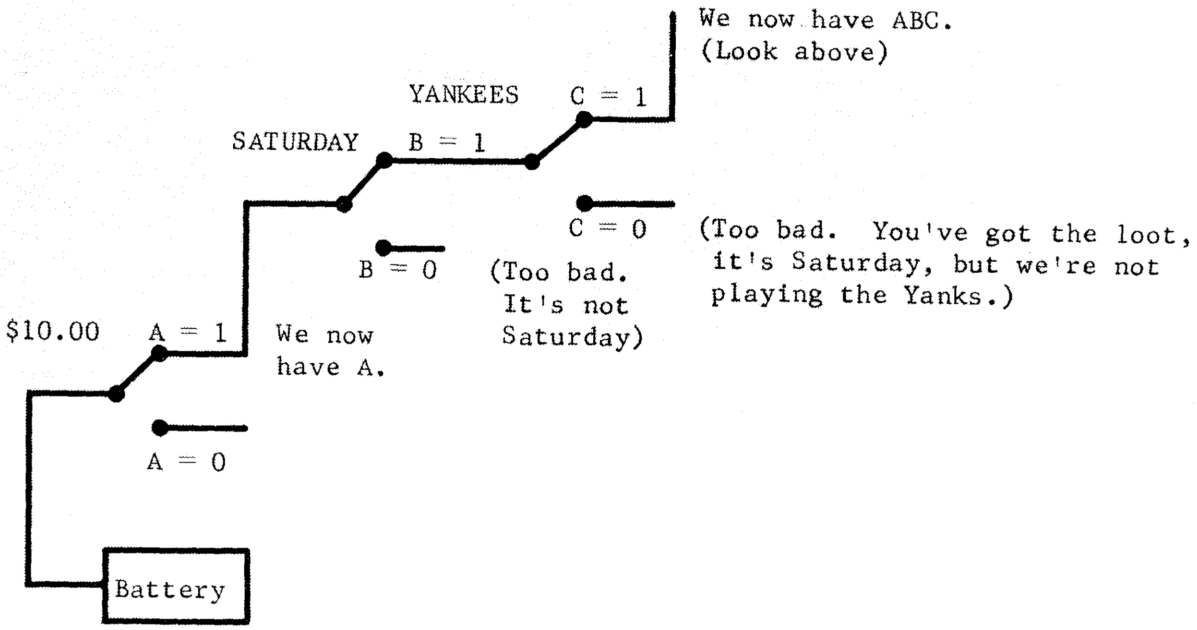
Consider how these additional factors would be added to the circuit below and, after you believe you know how, add them in the circuit.



We now have A. (see above)

(Dead end: Obviously you're not going to the ball game. You do not have $10.00.)

We also considered another factor: raining (D). We stated we would go if it was not raining OR we would go if it was raining AND we had a raincoat. Consider just the raining options. How would they be represented? If you think you know, add this to your drawing.



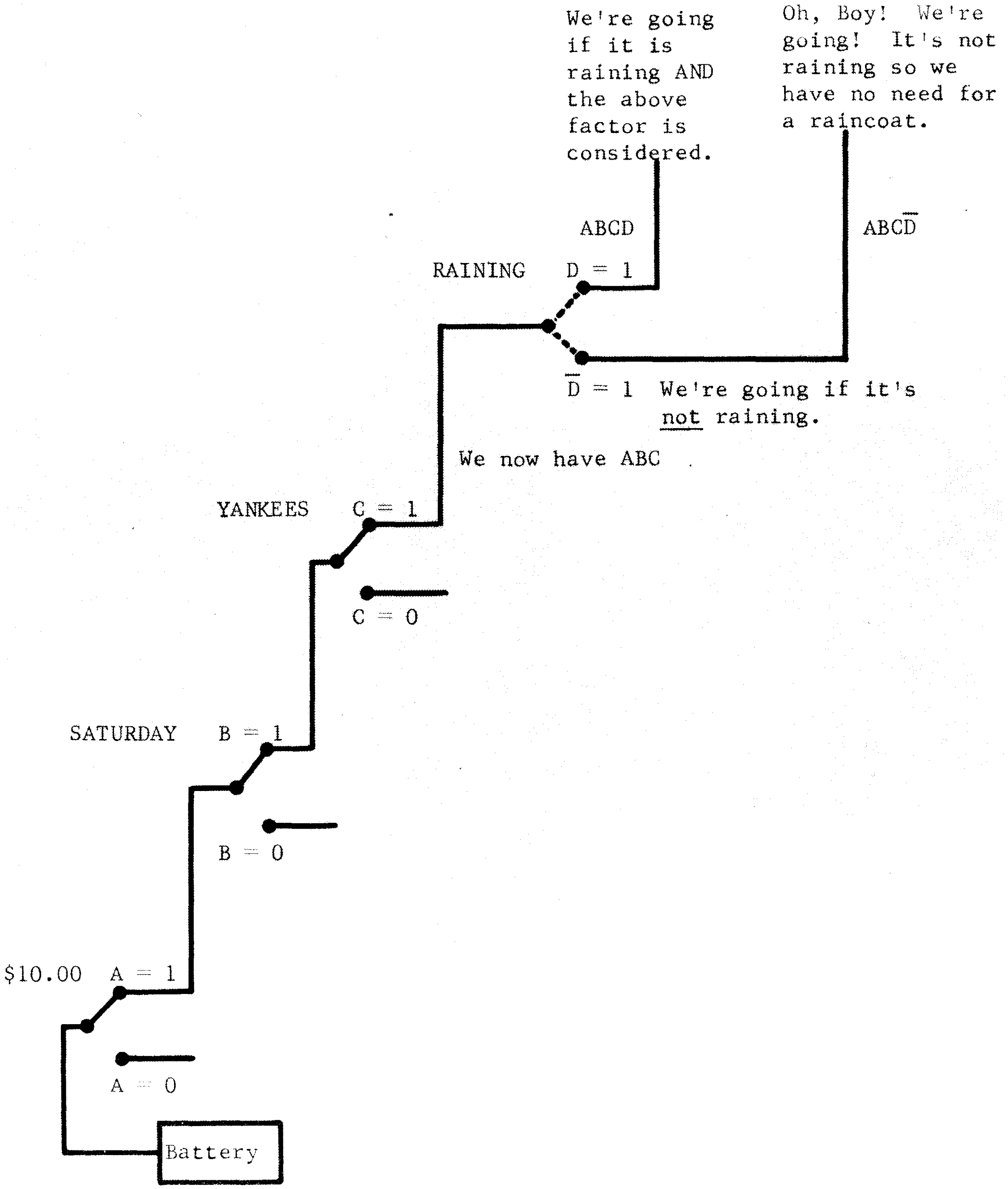
We now have ABC. (Look above)

(Too bad. You've got the loot, it's Saturday, but we're not playing the Yanks.)

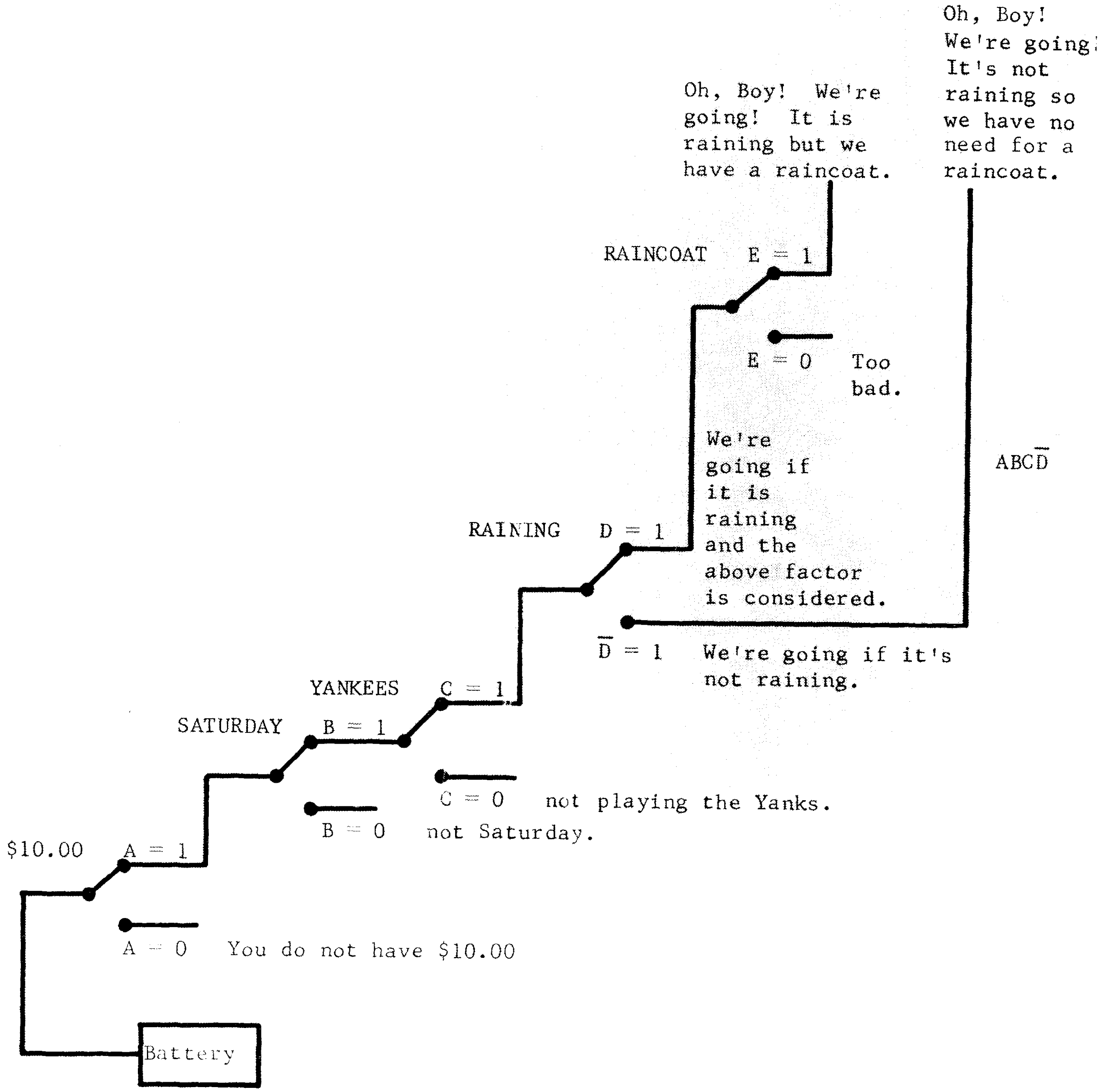
(Too bad. It's not Saturday)

We now have A.

If it is not raining, we are going to the game because now we know we have $10.00, it's Saturday, the game is with the Yankees AND it's not raining. (Warm up the car; here we come!) If it was raining, the possibility still exists of being able to go if we have a raincoat (E). How would this be shown in the circuit?

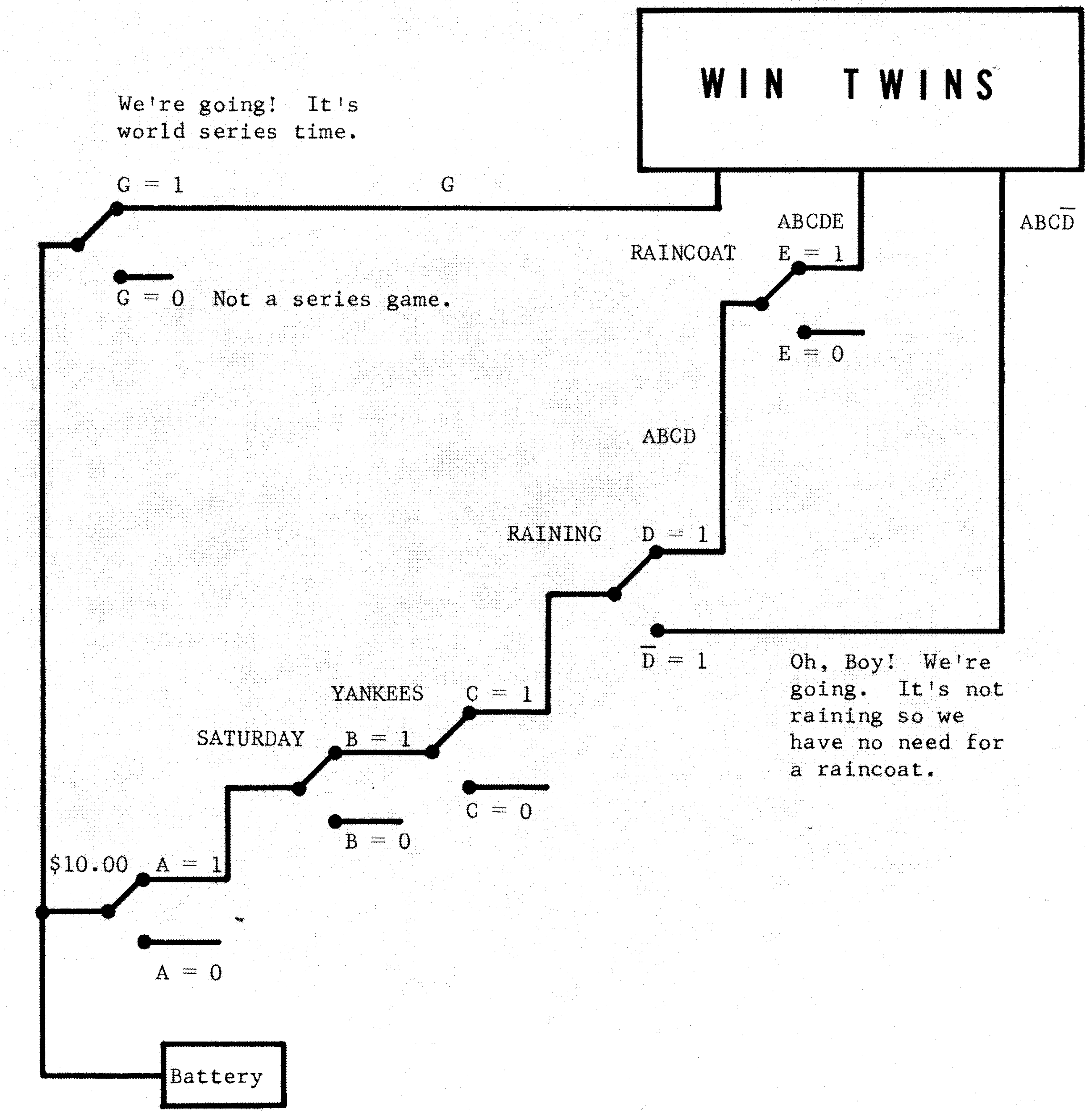

We're going if it is raining AND the above factor is considered.
Oh, Boy! We're going! It's not raining so we have no need for a raincoat.
ABCD
$AB C\overline{D}$
RAINING
D = 1
$\overline{D}$ = 1
We're going if it's not raining.
We now have ABC
YANKEES
C = 1
C = 0
SATURDAY
B = 1
B = 0
$10.00
A = 1
A = 0
Battery

There was one other consideration by which we could attend a ball game that overshadowed all the others. What was it? Right, the world series. Add that possibility to the diagram.



Oh, Boy! We're going! It's not raining so we have no need for a raincoat.
Oh, Boy! We're going! It is raining but we have a raincoat.
RAINCOAT E = 1
E = 0 Too bad.
ABC$\overline{D}$
We're going if it is raining and the above factor is considered.
RAINING D = 1
$\overline{D}$ = 1 We're going if it's not raining.
YANKEES C = 1
SATURDAY B = 1
C = 0 not playing the Yanks.
B = 0 not Saturday.
$10.00 A = 1
A = 0 You do not have $10.00
Battery

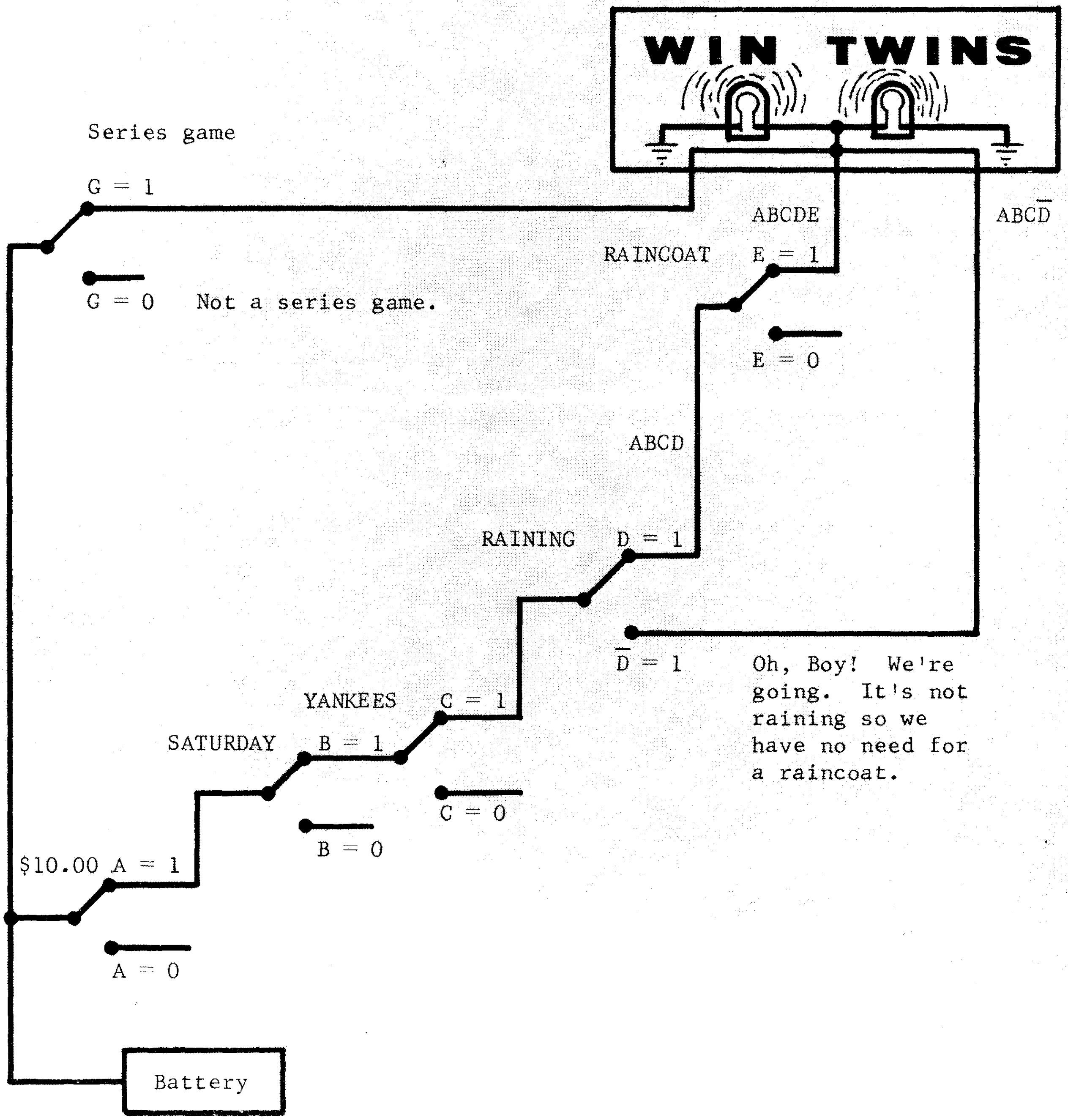
Complete your diagram by connecting the three available paths to light the "Win Twins" sign.



Congratulations! You have lighted the sign!

$$ABC\bar{D} + ABCDE + G = 1$$

$$ABCD\bar{E} + ABCDE + G = \text{"WIN TWINS"}$$



Oh, Boy! We're going. It's not raining so we have no need for a raincoat.

CONCEPTS OF BOOLEAN ALGEBRA

The language used by the computer engineer, designer, maintenance man and instructor to describe the internal characteristics of the computer is called Boolean Algebra. It contains representative symbols used in much the same way that letters are used in our alphabet. Also included are signs for grouping and punctuation used in a manner similar to commas, colons and periods in the English language.

This language is termed an algebra because some of its rules concerning rearrangement and simplification appear to follow rules of arithmetic algebra. Boolean Algebra is, however, a language of logic. Its rules, characteristics, purposes and uses are independent of arithmetic algebra. The analogy between the two types ends with their similar formats.

Boolean Algebra, like other organized processes, is based on assumptions and rules. Its central assumption is easy to grasp: Boolean variables of things must assume one of two distinct states or conditions. Its methods of operation and manipulation are based on reasoning. The student would do well not to simply memorize the rules, but to strive for a fundamental understanding of the concepts.

The following pages introduce the reader to methods and formats of Boolean Algebra. In the explanations of boolean expressions, an analogy to electrical switches is made. In addition to clarifying the meaning of the expression itself, the use of the switches aids in orienting the reader to the methods employed in the actual circuitry. The boolean expressions are the equivalents of the electrical circuits which illustrate them.

NOTE

> During this entire discussion of Boolean Algebra the letter o (oh) and the letter I (eye) are not used to represent variables. This is to prevent confusion with the commonly used values "1" (one) and "0" (zero) which are necessary in understanding the material.

When noting boolean expressions, two basic signs are used which serve to show the relationship that exists between individual factors and between terms. These two signs are the AND and the OR, shown symbolically by a centered dot (·) and a plus (+), respectively. The following topics will consider each of these in turn.

THE AND FUNCTION

Determining whether a given operation should or should not be carried out is usually the result of evaluating many facts. For example, a decision to go to a ball game could be influenced by the weather and the teams playing as well as many additional factors.

If it were decided that the ball game would be attended if the weather were favorable and the teams involved were the Yankees and Twins, then the decision to go would be a result of both these factors. Thus, it would be stated that: "If the weather was good AND it was a Yankees-Twins game, attendance could be expected".

Assigning literal values to these unknowns,

A = Favorable weather

B = Twins-Yankees game

C = Attendance

the statement, A and B = C, would summarize these conditions. In short form notation this would be written $A \cdot B = C$ or $AB = C$ where the sign ($\cdot$) is understood in the same fashion as it is in algebra.

Close evaluation of this statement should, however, reveal that it expresses only the relationship which exists between the factors A, and B, regarding attendance C and does not indicate the truth of either factor or of the statement. In order to appraise the truth of this statement one must know the state or condition of the variables.

It could also be stated that if A AND B each equal "1", then C would equal "1". If either A or B equals "0" then C would equal "0". The definition of the AND function, then, is: A logical relationship of factors such that the function is true only when all factors are true.

THE OR FUNCTION

The AND function, we have seen, is the logical relationship of variables, called factors. On the other hand, the OR function, is the logical relationship of terms, where a term may be composed of any number of factors. By definition, the OR function is: A logical relationship of terms such that the function is true if any term is true.

Again relating to baseball, it could be stated: If I have a ticket OR the game is not already a sellout, I will go.

Where A = I have a ticket

B = The game is not a sellout, and

C = I will go,

the expression could be written A + B = C. In this case if A = "1" or B = "1" or if both equal ones then C = "1". Only for the case where A = 0 and B = 0 does C = 0.

As in the explanation of the AND function, this expression only indicates the relationship which exists between terms; therefore the state of A and B must be known before the state of C can be determined.

THE AND GATE

Study the block diagram shown in Figure 5-1. Note that there are inputs to the AND circuit from three separate circuits. (The AND circuit is represented on a logic diagram by a small, open circle.)

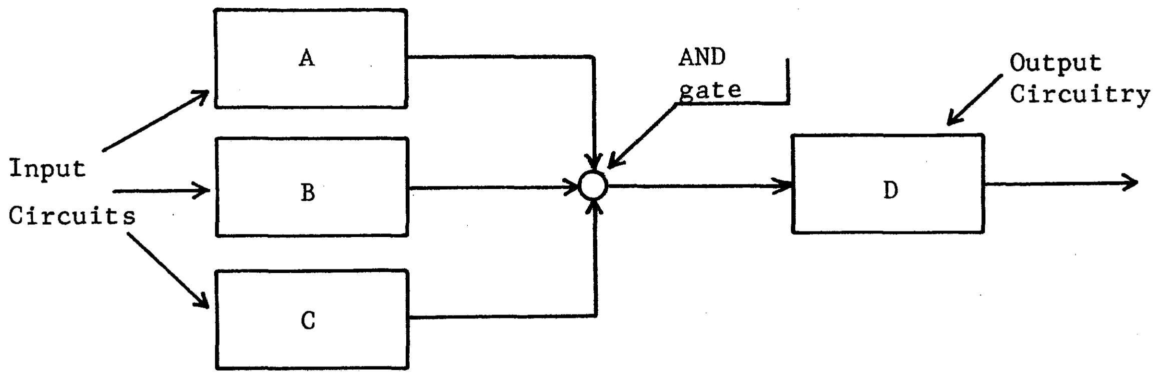


Figure 5-1. Block Diagram of an AND Gate.

If any one (or more) of three inputs (A, B or C) is "0", the output of the AND gate will be "0". Only if A = "1", B = "1", and C = "1", will the input to D be "1".

The simple electrical circuit analysis (Figure 5-2) denotes that the switch contacts A AND B AND C must be closed to light the lamp. A AND B AND C may be expressed as A•B•C, ABC. or (A) (B) (C) but the brief form (ABC) is normally used.

The lamp would be comparable to the actual AND gate (open circle) shown in Figure 5-1.

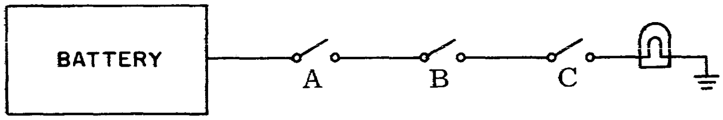


Figure 5-2. Switches Depicting AND Relationship.

AND gates are logical elements of the computer that perform a logical function by accepting inputs and giving new logical outputs. This is accomplished electrically by the use of components called diodes.

The AND gate requires all of its inputs to be logical ones before its output becomes a "1". Any other combination of inputs will result in a logical "0" output. Study the diagram (Figure 5-3) that follows:

Assume that

-5.8V = logical "1"

-1.1V = logical "0"

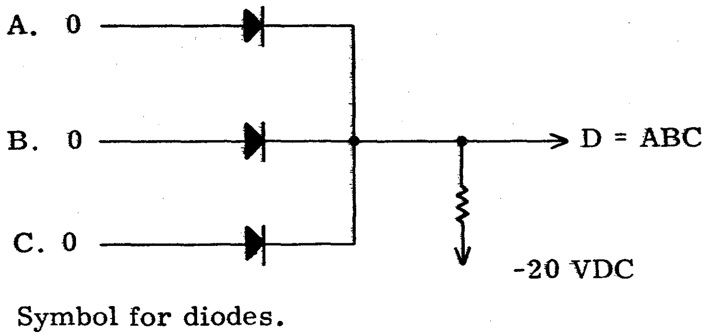


Figure 5-3. An AND Gate.

Three input signals called A, B, and C are applied to the AND gate. If these are present (ones) simultaneously, there will be a logical "1" out. For example:

ABC = 1; therefore, D = 1.

This expression is read, A <u>and</u> B <u>and</u> C all equal a logical "1"; therefore, D = logical "1".

If any one of the inputs is a logical "0" (A, for instance), the equation becomes ABC = D = "0" and the output of the AND gate is a logical zero. An AND gate need not be composed of three factors, as any number may be used. Circuit limitations, however, usually set the number of "legs" to approximately 4 or 5.

Electrically, the AND gate may be approached as follows:

With logical zeros at the input points (-1.1 volts), the diodes are forward biased and conduct. Since the impedence of the diodes in this state is negligible, the voltage at the output is -1.1 volts. If one of the inputs, A for instance, is caused to go to a logical "1" (5.8 volts), the associated diode ceases conduction as it is back-biased by the input signal. The diodes associated with inputs B and C, however, are still receiving logical zero (-1.1 volts) inputs and are allowed to conduct. This holds the output to -1.1 volts. (A more complete description of the actual circuitry is found in Chapter VI.)

If all three inputs receive logical one (-5.8 volts) inputs, the diodes cease conduction and the output level goes to -5.8V volts (logical "1"). In order to receive a logical one output from the AND gate, all three input signals must be logical ones (-5.8 volts each).

THE OR GATE

The OR (Figure 5-4) produces a "1" output when at least one of its inputs is a "1". The output of an OR gate is "0" only when all of its inputs are "0". An output circuit with an ORed input is shown in Figure 5-4. (The separate arrows shown entering D represent OR inputs).

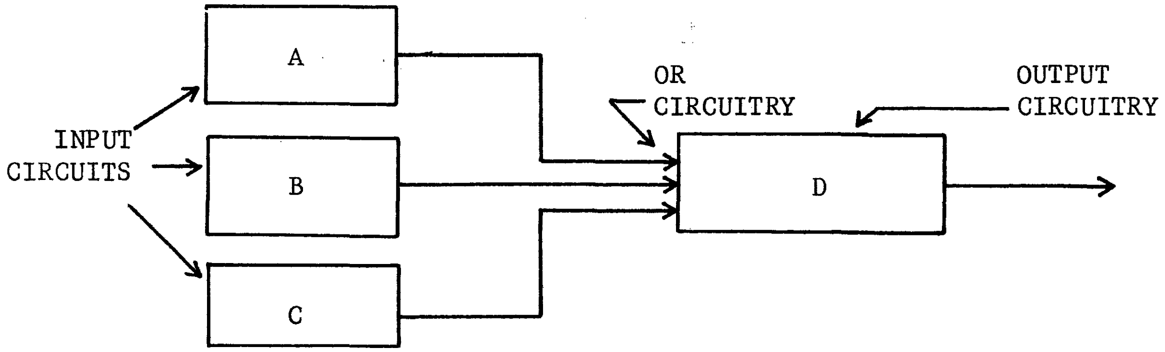


Figure 5-4. Block Diagram of an OR Gate

An OR relationship is available to the boolean variables A and B. Figure 5-5 is an electrical circuit analysis. It is important to note in the analysis that the functions of the boolean variables are electrically parallel such that if switch A closes or if B closes, or if both close, the lamp lights.

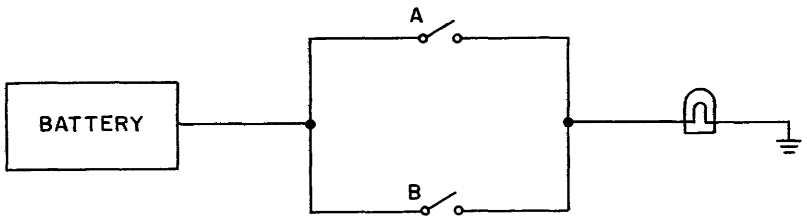


Figure 5-5. Switches Depicting the Relationship A + B.

This relationship is defined as the OR relationship. It means either one, OR the other, OR both switches are closed. The words "or both" denote the condition of the inclusive OR; that is, it includes the case where both are ones. Its counterpart, the exclusive OR, is described later.

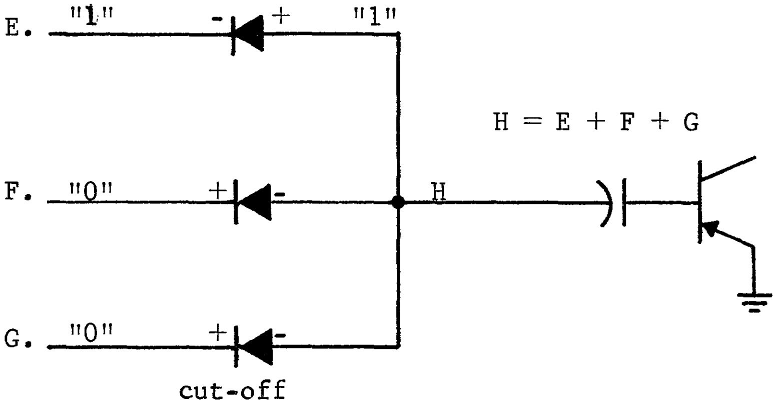


Figure 5-6. An OR Gate.

Three input signals, called E, F and G are applied to an OR gate. One or more of these signals being a logical "1" makes the output H = "1".

Therefore, H = E + F + G. This expression is read, the output H equals E or F or G.

Electrically the output H is a "1" when a -5.8 volts is present on the cathode of an input diode. This is caused by the forward bias applied to the diode by the input signal. If all of the inputs are logical zeros (-1.1 volts), none of the diodes conduct and the output remains at -1.1 volts (logical zero). A more complete description of the actual circuitry can be found in Chapter VI.

COMBINATION GATES

AND and OR gates may be connected to make the final result a function of many inputs. Figure 5-7 is an example of a combination network. The proof of the input expression is left to the student.

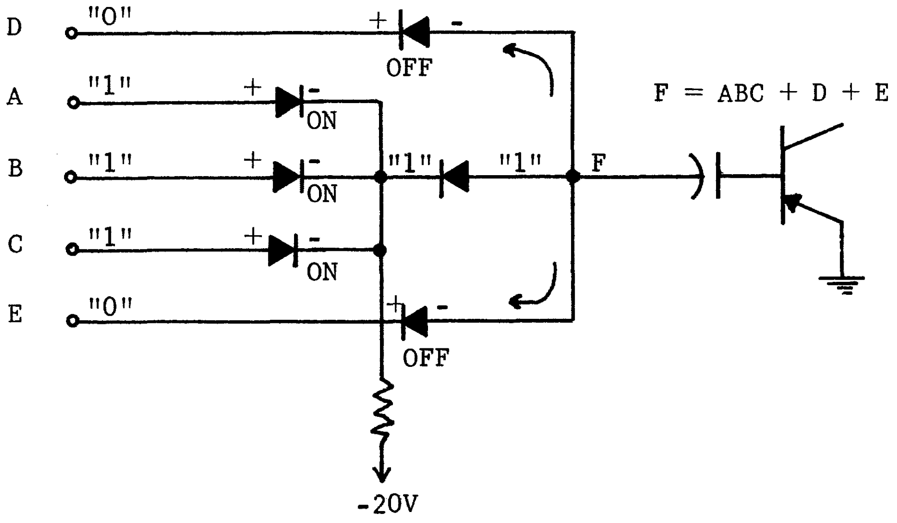


Figure 5-7. Combination Network.

EXCLUSIVE OR

Another logical function is commonly used and plays a very important role in understanding computers. This function is the exclusive OR (symbol ⊻). The exclusive OR (Figure 5-8) is never implied and must be specifically indicated by joining the terms with the correct sign.

It should be noted that the exclusive OR differs from the inclusive OR only in so far as it excludes the case 1+1=1, yielding instead 1+1=0 (see Table 5-1).

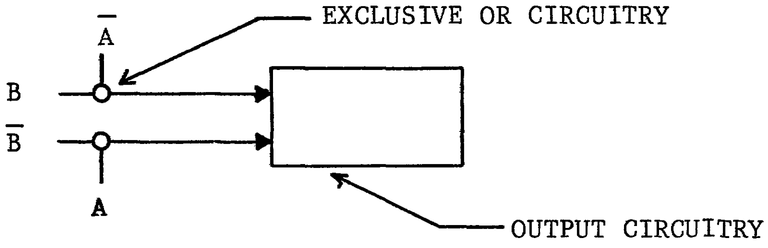


Figure 5-8. Exclusive OR

The switch analogy for the exclusive OR is illustrated in Figure 5-9.

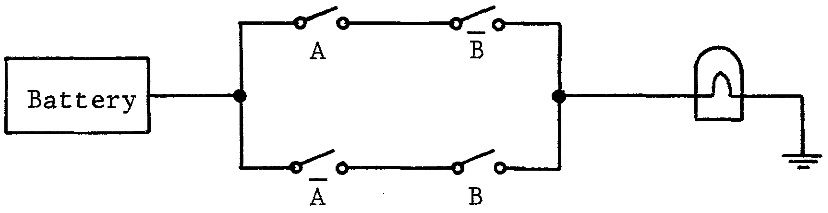


Figure 5-9. Switches Depicting the Exclusive OR.

It can be seen that $A\overline{B}$ or $\overline{A}B$ would light the lamp but that AB or $\overline{AB}$ would break both paths between the lamp and the battery. The boolean expression for the circuit, therefore, would be $A\overline{B} + \overline{A}B$. Boolean expressions are always written for a logical one.

The actual circuit for the exclusive OR is shown by Figure 5-10. The exclusive OR is a combination network similar to Figure 5-7 whereas the inclusive OR is similar to the circuit illustrated in Figure 5-6. Compare Figure 5-10 and 5-11.

Remember that the term A could represent a logical"1"but $\overline{A}$ could also represent a logical 1. If $\overline{A}$ ="1"then, of course, A ="0".

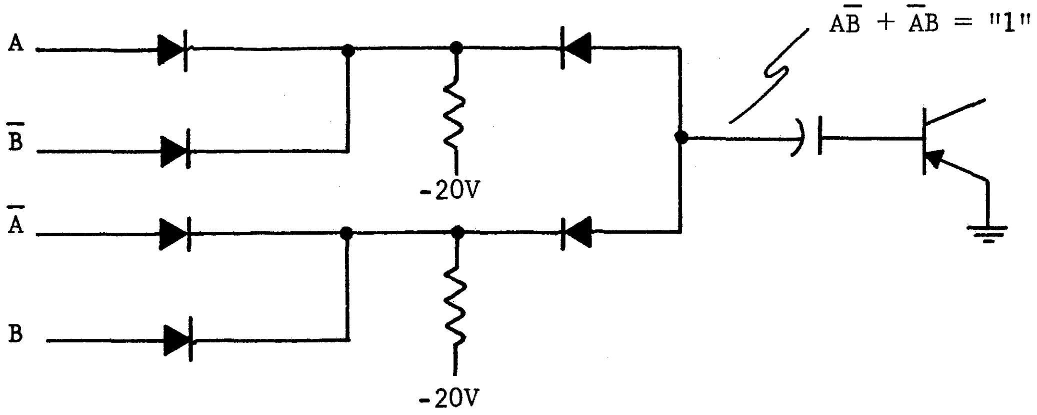


Figure 5-10. Exclusive OR.

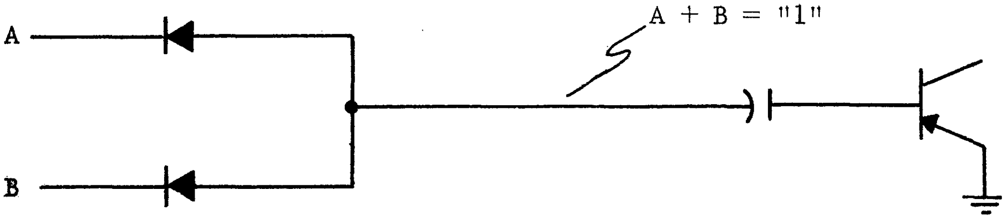


Figure 5-11. Inclusive OR.

Table 5-1 summarizes the relationships which have been discussed. Before going on, be sure you understand these relationships.

TABLE 5-1. LOGICAL COMBINATIONS

		AND	OR	Excl. OR
p	q	$p \cdot q$	$p+q$	$p \veebar q$
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

FUNDAMENTAL RULES OF BOOLEAN ALGEBRA

Boolean Algebra has a set of fundamental rules which allow expressions to be rearranged and simplified. It is frequently necessary to rearrange or simplify these expressions in order to remove redundancies and make the real relationships easier to understand.

Following is a series of rules which serve as guide lines for the manipulation of Boolean expressions. It is desirable for the student to grasp the underlying principles in each case rather than to commit the list to memory. Associated problems are included to aid in the application of the appropriate rule. In Boolean Algebra, as in regular algebra, the key to understanding is practice.

The first 5 statements covered are obvious and serve more as definitions than as rules. These 5, however, are referred to during the discussion of later statements.

STATEMENT 1

$A = 0$ if $A \neq 1$

$A = 1$ if $A \neq 0$

This is a doctrine stating that all things may be expressed by two distinct rules or principles, indicating that once a system is described by a set of relationships, the dual system automatically exists which is complementary or opposite. This principle is applied in Boolean Algebra to show the symmetry (balance) between the symbols "+" and "•", "1" and "0", and "A" and "$\bar{A}$".

This could also be stated:

If $A = 1$ then $\bar{A} = 0$

and if $A = 0$ then $\bar{A} = 1$

STATEMENT 2

$0 \neq 1$ and $1 \neq 0$

This statement is to clarify the relationship between a logical "1" and a logical "0". It is meant to show that the zeros and ones used are representative of states or conditions (false and true respectively).

STATEMENT 3

$0 \cdot 0 = 0$

$0 + 0 = 0$

This relationship verifies the fact that, regardless of the manner in which a series of zeros are related, the result is zero. This can be represented by a permanent open (a factor which can never be true) connected in either series or in parallel with another permanent open, resulting in an open. The following diagrams, Figures 5-12 and 5-13, utilize switches to represent these statements.

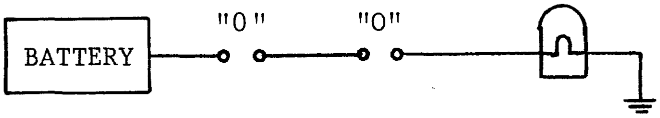


Figure 5-12. Switches Representing 0 AND 0.

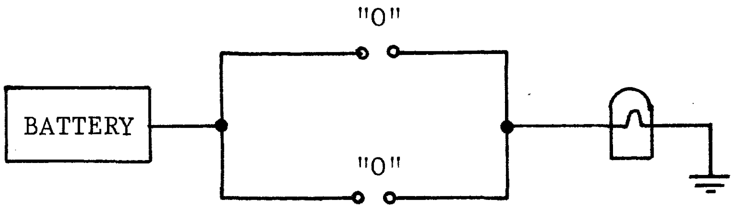


Figure 5-13. Switches Representing 0 OR 0.

STATEMENT 4

$1 \cdot 1 = 1$

$1 + 1 = 1$

In this case it should be seen that any combination of all ones results in just a single one. In the first case ($1 \cdot 1 = 1$), the analogy can be drawn to a simple straight piece of wire (permanent short) which contains a series of always present logical elements. Assuming that no physical change takes place in the wire, each of its logical components (an infinite number) is equal to a value of 1 (always true or present). Thus a series of 1's ANDed together produces an end result of a single 1 (in this example, the entire wire), as represented by Figure 5-14A.

To understand the second half of this statement, $1 + 1 = 1$, imagine two pieces of wire similar to the one just described connected in parallel (see Figure 5-14B) with each path being present. It can be seen that either one or the other is all that is necessary and that the other path is redundant and can be removed.

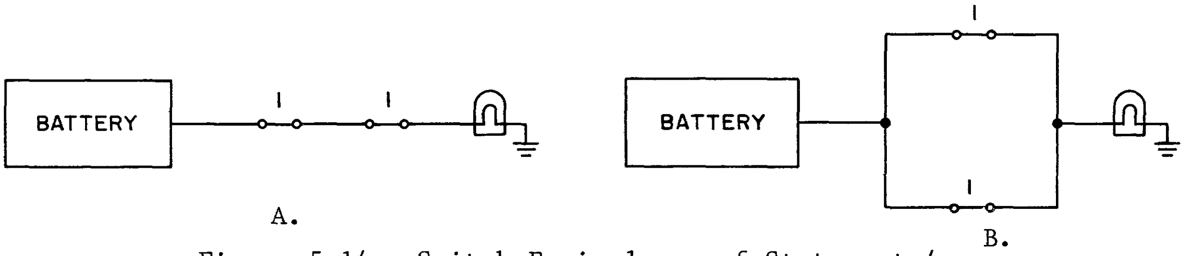


Figure 5-14. Switch Equivalence of Statement 4.

STATEMENT 5

$0 \cdot 1 = 0$

$0 + 1 = 1$

This statement is actually a combination of statements 3 and 4. The first half can be explained in the same manner as statement 3. Any time one element of a series of elements which are ANDed together is equal to zero, the entire expression must equal zero. The second part can be stated: Any time a group of ORed terms includes a 1, the entire expression is equal to 1 since at least one of the terms is always present; this fulfills the requirements for satisfying an ORed expression. See Figure 5-15.

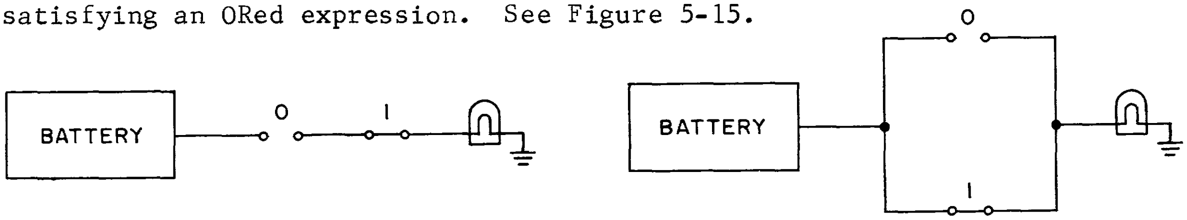


Figure 5-15. Switch Equivalence of Statement 5.

STATEMENT 6

$A + A = A$

If a Boolean expression exists in this format, statement 6 justifies its reduction to "A". Since, in any given expression, the value A can represent only one condition, it can be seen that more than one A in this case is redundant. Drawing an analogy back to the ball game and equating A with having $10.00, the statement A + A would read "I will go to the ball game if I have $10.00 or if I have $10.00." The redundancy here is obvious.

This statement can be used to simplify any expression containing identical ORed terms. As an example, $ABC + ABC = ABC$. Notice that each ORed term is identical. If they are not completely identical this rule does not apply.

The following table shows proper and improper application of this rule.

TABLE 5-2. PROPER AND IMPROPER USE OF STATEMENT 6.

Proper:	
(1)	$AB + AB = AB$
(2)	$\bar{A}BC + \bar{A}BC = \bar{A}BC$
(3)	$(XY) + (XY) = XY$
Improper:	
(1)	$\bar{A}B + AB = ?$
(2)	$ABC + AB = ?$
(3)	$(XYZ) + (ABC) = ?$

An electrical representation of this statement can be seen in Figure 5-16.

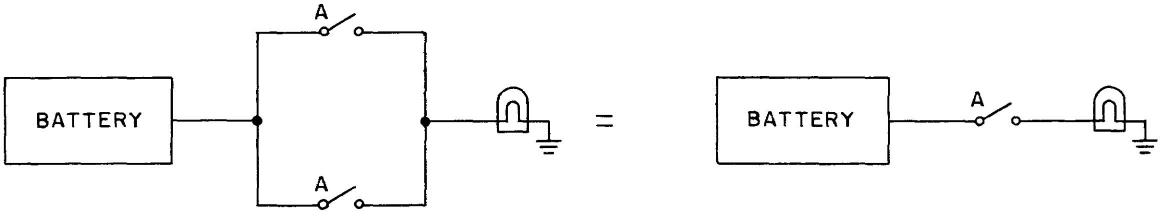


Figure 5-16. Switch Equivalence of Statement 6.

Work the following problems by indicating if the answer given is true or false. In the area to the right of the problem explain briefly why the rule does not apply to the problems checked false.

T F 1) $ABC + ABC = ABC$

T F 2) $AC + A\overline{C} = AC$

T F 3) $MNS + MNS + MNS = MNS$

T F 4) $LPR + LPR + \overline{L}PR = \overline{L}PR$

The correct answers to these problems can be found at the end of this chapter. Write, in your own words, a summary of statement 6.

STATEMENT 7

$A \cdot A = A$

Statement 7 is concerned with two identical factors being ANDed together. Remember that in any given expression, the factor A (in this example) can be used to represent only one variable. It is certain that if the value A is true in one case, it is also true in the other. Therefore, it is necessary that A be included only once as a factor in an AND function. Note that this rule does not apply if the repeated factor is combined with other terms. The following table shows proper and improper application.

TABLE 4-3. PROPER AND IMPROPER USE OF STATEMENT 7

Proper	
(1)	$(ABC) \cdot (ABC) = ABC$
(2)	$(\overline{A}\ \overline{B}\ \overline{C})\cdot(\overline{A}\ \overline{B}\ \overline{C})=\overline{A}\ \overline{B}\ \overline{C}$
(3)	$(AB) \cdot (AB) \cdot (AB) = AB$
Improper	
(1)	$(AB) \cdot (ABC) = ?$
(2)	$(\overline{A}\ \overline{B}\ \overline{C}) \cdot (ABC) = ?$
(3)	$(AB) \cdot (\overline{A}\ \overline{B}) \cdot xy = ?$

Why are each of the improper statements incorrect?

(1) ____

(2) ____

(3) ____

Using switches, the rule can be demonstrated as in Figure 5-17.

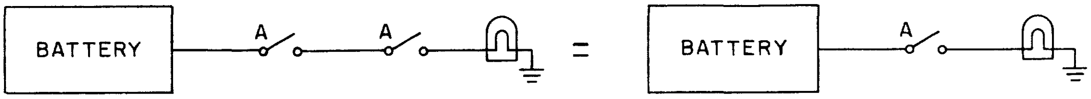


Figure 5-17. Switch Equivalence of Statement 7.

Work the following problems by indicating if the answer given is true or false. For all answers checked as false, explain the reason in the area to the right of the problem.

T F 5) $(XY)\ (XY)\ (XY) = XY$

T F 6) $(AB)\ (AB)\ (A\overline{B}) = AB$

T F 7) $(AB\overline{C})\ (\overline{C}BA)\ (B\overline{C}A) = AB\overline{C}$

T F 8) $A \cdot A \cdot \overline{A} = A$

Correct answers to the problems may be found at the end of this chapter. Write, in your own words, an explanation of this rule.

STATEMENT 8

$A + 1 = 1$

This statement is an expansion of statement 4 (second half, 1 + 1 = 1) and can be explained by substituting an A for one of the logical 1's in statement 4. Thus statement 4 becomes statement 8.

Statement 4 = 1 + 1 = 1 ;

substitute A = A + 1 = 1 = statement 8.

The same reasoning holds true for both statement 4 and 8 since they each involve a term parallel with a permanently closed (non-variable) path. Note that the terms are ORed with each other. This is a prime consideration in both statements. Table 4-4 shows proper and improper use of statement 8.

TABLE 5-4. PROPER AND IMPROPER USE OF STATEMENT 8.

Proper:		
(1)	$ABC + 1 = 1$	(in this case, ABC is one term
(2)	$\overline{A} + 1 = 1$	which is ORed with a logical one).
(3)	$A + B + C + 1 = 1$	
Improper:		
(1)	$ABC\ .\ 1 \neq 1$	the one is not ORed with other
(2)	$ABC\ .\ 1 + \overline{A}BC \neq 1$	terms, therefore, rule 8 does not apply.

The electrical circuit equivalent to statement 8 is shown in Figure 5-18.

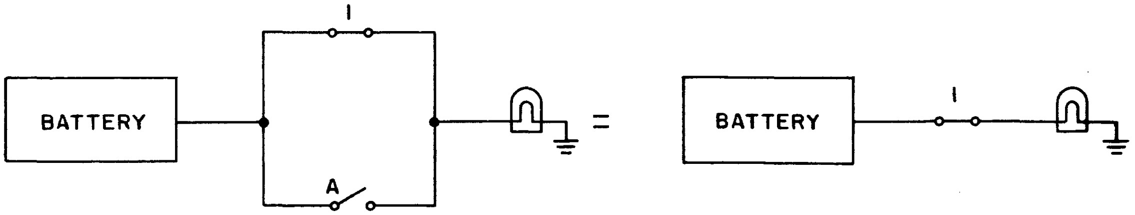


Figure 5-18. Switch Equivalent, Statement 8.

Work the following problems by checking each answer either true or false. If the answer is false, explain the reason in the area to the right of the problem

Answers to these problems can be found at the end of this chapter.

T F 9) $A + ABC + DE + 1 = 1$

T F 10) $B + 1 + C = 1$

T F 11) $ABC + DEF + 1 + xyz = 1$

T F 12) $ABC1 + DEF = 1$

In your own words write a summary of this rule.

STATEMENT 9

$A \cdot 0 = 0$

This statement is an expansion of statement 3 ($0 \cdot 0 = 0$) where the variable A has replaced one zero.

Statement 3 $= 0 \cdot 0 = 0$

Substitute $A = A \cdot 0 = 0$ statement 9

Table 4-5 shows the proper and improper use of statement 9.

TABLE 5-5. PROPER AND IMPROPER USE OF STATEMENT 9.

Proper	
(1)	$ABC \cdot 0 = 0$
(2)	$(A)\ (B)\ (C) \cdot 0 = 0$
(3)	$\bar{A} \cdot 0 = 0$
Improper	
(1)	$ABC + 0 \neq 0$
(2)	$A + B + C \cdot 0 \neq 0$
(3)	$\bar{A}\ \bar{B}\ C + 0 \neq 0$

Why are each of the improper uses incorrect?

(1) ____

(2) ____

(3) ____

The electrical circuit using switches to illustrate statement 9 is shown in Figure 5-19.

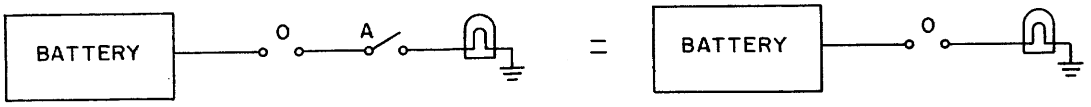


Figure 5-19. Switch Equivalence of Statement 9.

Work the following problems by indicating if the answer given is true or false. In the area to the right of the problem, explain the reason why each answer checked false is incorrect.

T F 13) $wxyz \cdot 0 = 0$

T F 14) $(\bar{A})\ (\bar{B})\ (\bar{C})\ (0) = 0$

T F 15) $(A + B + C) \cdot (0) = 0$

T F 16) $(A + A) \cdot (0) = 0$

You know where to find the answers. In your own words write a summary of this rule.

STATEMENT 10

$A + 0 = A$

This statement is a variation of statement 6 which states that $A + A = A$. If, in statement 6, one A were removed (or if it wasn't present initially), the statement would be written $A + 0 = A$. A common-sense explanation of this rule states that $A = A$ if it is not paralleled by any other factors. The electrical circuit diagram in Figure 5-20 makes this clear.

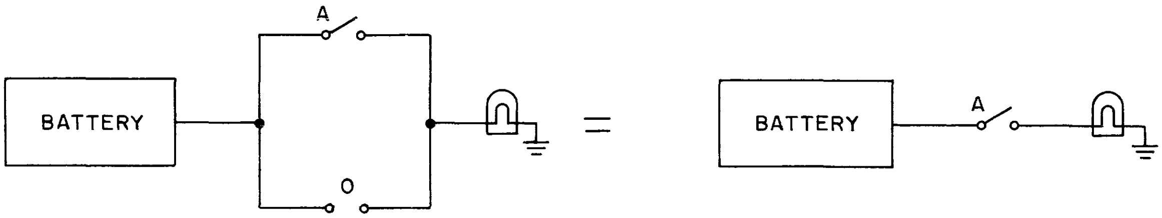


Figure 5-20. Switch Equivalence of Statement 10.

Note that in the right hand side of the diagram, A can be considered to be ORed with an infinite number of permanent opens (or zeros).

The proper and improper uses of statement 10 are shown in Table 5-6.

TABLE 5-6. PROPER AND IMPROPER USE OF STATEMENT 10.

Proper		
(1)	$ABC + 0 = ABC$	
(2)	$\bar{A} + B + C + 0 = \bar{A} + B + C$	
(3)	$MN + PQ + RS + 0 = MN + PQ + RS$	
Improper		
(1)	$(ABC)\ (0) = ?$	zero is not
(2)	$(\bar{A})\ (B)\ (C)\ (0) = ?$	ORed, so
(3)	$(MN)\ (0) + (PQ)\ (0) = ?$	rule does not apply

Work the following problems by indicating true or false with the answers as given. In the area to the right of the problem explain briefly why each false choice is incorrect. Also list the proper rule that validates each true answer.

T F 17) xyz + 0 = xyz

T F 18) ABCO = ABC

T F 19) AB + RS + xy + 0 = AB

T F 20) (zm) (0) (RS) = (ZM) (RS)

In your own words write an explanation of rule 10.

STATEMENT 11

$A \cdot 1 = A$

This statement is a variation of statement 7 which states $A \cdot A = A$. In statement 11, the value 1 has been substituted for one of the A's. A common-sense explanation shows that any single A is connected in series with an infinite number of constant logical ones. These would be the segments of the wire that make up the circuitry. This rule justifies the simplification of any expression containing ANDed terms, some of which are equal to 1. These 1's are redundant terms and can be removed without altering the effect of the resulting expression (or circuitry). The electrical circuit equivalent for statement 11 is shown in Figure 5-21.

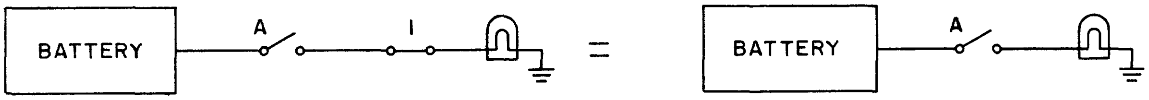


Figure 5-21. Switch Equivalence of Statement 11.

Assume it is desired to introduce a 1 into an expression. Would statement 11 allow the insertion of a 1 or a value which is equal to 1 into an expression?

Why or why not?

The proper and improper application of statement 11 is shown in Table 5-7.

TABLE 5-7. PROPER AND IMPROPER USE OF STATEMENT 11.

Proper		
(1)	$ABCD \cdot 1 = ABCD$	
(2)	$(A)\ (B)\ (C)\ (1) = ABC$	
(3)	$(A)\ (\overline{B})\ (\overline{C}) = (A)\ (\overline{B})\ (\overline{C})\ (1)$	
Improper		
(1)	$A + B + C + 1 = 1$	not an application of rule 11 but rather of rule 8.
(2)	$A = A + 1$	not an application of rule 11 but rather of rule 8.
(3)	$ABCD + 1 = 1$	not an application of rule 11 but rather of rule 8.

Work the following problems by indicating if the answers given are true or false. In the area to the right of the problem explain why problems marked false are incorrect. If you mark the statement true, list the statement number that validates it (if other than statement 11).

T F 21) $ABC = (A)\ (1)\ (B)\ (C)$

T F 22) $LMN \cdot 1 = (LMN)\ (1)\ (1)$

T F 23) (AB) (CD) (1) = (AB) (CD)

T F 24) (AB) (xy) (CD) = (1) (AB) (1) (xy) (1) (CD)

In your own words, write a summary of this rule.

STATEMENT 12

$A + \overline{A} = 1$

The main characteristic to recognize when applying this statement is that it deals with a term ORed with its complement (opposite). In addition, statement 12 is a direct result of applying two previously-covered rules. These two are:

Statement 1: if $X = 0$ then $\overline{X} = 1$

Statement 5: $0 + 1 = 1$

By evaluating statement 12 in view of statement 1 it can be seen that under any set of conditions either A or $\overline{A}$ must be equal to a "1". It can also be seen that when A is equal to 1, the condition of $\overline{A}$ must be equal to zero and vice-verse. If it is assumed that $A = 1$, then $\overline{A} = 0$ (written $1 + 0$); or if it is assumed that $A = 0$, then $\overline{A} = 1$ (written $0 + 1$). Each of these are equal to "1" by statement 5. By substitution, $A + \overline{A} = 1$, where $(A + \overline{A})$ bears the relationship of $(1 + 0)$ or $0 + 1)$.

The statement can be applied to any condition where a term is ORed with its complement, regardless of the number of factors within each term. The requirement, however, is that the terms must be truly complements of each other. The proper and improper applications of statement 12 are given in the Table 5-8.

TABLE 5-8. PROPER AND IMPROPER APPLICATIONS OF STATEMENT 12.

Proper	
(1)	$m + \overline{m} = 1$
(2)	$AB + \overline{AB} = 1$
(3)	$\overline{xy} + xy = 1$
(4)	$(\bar{c}d) + (\overline{\bar{c}d}) = 1$
Improper	
(1)	$m \cdot \overline{m} \neq 1$
(2)	$AB + \bar{A}B \neq 1$
(3)	$AB + \overline{xy} \neq 1$
(4)	$ABC + \overline{AB}C \neq 1$
(5)	$\bar{A}\bar{B}C + \overline{ABC} \neq 1$

Why are the improper statements in the table incorrect?

Work the following problems by indicating if the answers given are true or false. In the space to the right of the problem explain why each false answer is incorrect.

T F 25) $A\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,\bar{C} = 1$

T F 26) $(\overline{ABC + DEF}) + (ABC + DEF) = 1$

T F 27) $AB + \overline{AB} = 0$

T F 28) $AB + \overline{CD} = 1$

In your own words, write a summary of this rule.

STATEMENT 13

$A \cdot \overline{A} = 0$

This statement can best be explained by referring to two previously-covered rules, statement 1 and statement 5. Statement 1 states: if $X = 0$ then $\overline{X} = 1$. Statement 5 states: $1 \cdot 0 = 0$. Statement 1 shows that complementary terms can never be equal to each other. Therefore, the values for $(A \cdot \overline{A})$ must assume either of the following formats $(1 \cdot 0)$ or $0 \cdot 1)$. In either case the problem can be further resolved by invoking statement 5, since these formats fit its characteristics. By referring to statement 5 the entire expression is seen to equal zero. The following is a step-by-step outline of the above process.

Statement 13	$A . \overline{A} = 0$
Substitution	$(0.1) = 0$ or $(1 \cdot 0) = 0$
Statement 5	$(0) + (0) = 0$
Statement 3	$0 = 0$

The main characteristic to look for when applying statement 13 is a factor to be ANDed with its complement. The number of factors included is immaterial so long as factors are complementary and are ANDed together.

Table 5-9 shows the proper and improper application of statement 13.

TABLE 5-9. PROPER AND IMPROPER USE OF STATEMENT 13.

Proper	
(1)	$A \cdot \bar{A} = 0$
(2)	$\overline{AB} \cdot AB = 0$
(3)	$\bar{A}B \cdot \overline{\bar{A}B} = 0$
Improper	
(1)	$A + \bar{A} \neq 0$
(2)	$\overline{AB} \cdot \bar{A}\ \bar{B} \neq 0$
(3)	$ABC \cdot \overline{AB} \neq 0$

Why are the improper statements incorrect?

Solve the following problems by indicating if the answers are true or false. In the area to the right of the problems marked false, explain why they are incorrect.

T	F	(29)	$AB \cdot \overline{AB} = 0$
T	F	(30)	$(xAB) \cdot (\overline{ABx}) = 0$
T	F	(31)	$(\overline{x+B}) \cdot (B+x) = 0$
T	F	(32)	$(ABC) \cdot (\bar{A}\ \bar{B}\ \bar{C}) = 0$

In your own words, write a brief summary of this rule.

STATEMENT 14

$\overline{\overline{A}} = A$

This statement is designed for cases which include double negatives. This statement may read "not $\overline{A}$ = A". An analogy can be drawn to the use of double negatives in English. A person corrected for saying "I don't have no money" would be told that the double negative is the same as no negative at all; it is the same as saying "I do have money". Another example would be to turn a playing card, which is lying on the table, over once for each negation. The result would be the card returning to the same state in which it started.

Both of these analogies are true for statement 14. If $\overline{A}$ is not present then A must be present. Thus $\overline{\overline{A}} = A$.

This statement applies to a term of any length and consisting of any internal method of grouping so long as both negation bars cover the entire term. Table 5-10 exemplifies the proper and improper usage of statement 14.

TABLE 5-10. PROPER AND IMPROPER USE OF STATEMENT 14.

Proper	
(1)	$\overline{\overline{AB}} = AB$
(2)	$M\overline{N} = \overline{\overline{M\overline{N}}}$ (notice that the negation over N has not been disturbed).
(3)	$\overline{\overline{AB + c}} = AB + \overline{c}$
Improper	
(1)	$\overline{\overline{A}B} \neq AB$
(2)	$\overline{\overline{A + B}C} \neq A + BC$
(3)	$\overline{\overline{M\overline{N}}} = MN$

Work the following problems by indicating if the answers given are true or false. In the area to the right of each problem marked false, explain why the given answer is incorrect.

T F 33) $(\overline{\overline{A} + \overline{B} + \overline{C}}) = A + B\ C$

T F 34) $\overline{\overline{A + B}} + C = A + B + C$

T F 35) $\overline{\overline{A}} + \overline{\overline{B}} = A + B$

T F 36) $\overline{\overline{\overline{A}} + \overline{\overline{B}}} = \overline{A + B}$

Check your answers. In your own words, write a summary of this rule.

Before continuing to the remaining statements, work the following problems as a review.

Use only the rules covered thus far and reduce each expression as far as possible. Some problems may require the application of more than one rule to reduce it to its simplest form. Check your answers.

REVIEW QUESTIONS

37) $0 \cdot 0 =$

38) $1 + 1 =$

39) $1 \cdot 1 =$

40) $1 + 0 =$

41) $0 + 0 =$

42) $1 \cdot 0 =$

43) $A \cdot 0 =$

44) $1 \cdot A =$

45) $0 + 1 =$

46) $0 \cdot 1 =$

47) $A + 1 =$

48) $0 + A =$

49) $(A + B + CD) \cdot 0 =$

50) $AB + \overline{AB} =$

51) $A + \overline{A}A =$

52) $AB + \overline{BA} + AA =$

53) $\overline{\overline{A}}A + A + \overline{A} + A + 1 =$

54) $\overline{A} + \overline{B} + A + B + \overline{B} =$

55) $AB + AB + AB + AB + AB =$

56) $\overline{\overline{\overline{AB}}} + CD =$

57) $(C+L)\ (C \cdot 1) + (0 \cdot A)\ (1 + A) =$

If correct answers were not obtained for all of the review problems, the pages relating to the corresponding type problem should be reread and the problem retried.

STATEMENT 15

$A + AB = A$

This statement is applied to simplify a type of expression in which the terms are not altogether common. The major important characteristics in this format are: 1) The expression must contain ORed terms (in this case A ORed with AB) and 2) There must be a factor which is common to all terms involved (in this case the factor A appears in each of the ORed terms).

To explain this statement, each condition of A is assumed and a logical result is obtained.

Assume the value of A is 1 (A is true).

Original statement:	$A + AB = A$
Then by substitution (placing 1's for A's),	$1 + 1 \cdot B = 1$
By applying statement 11,	$1 + B = 1$
By applying statement 8,	$1 = 1$

Thus, when the condition for A is true the equation is true.

Assume the value of A is 0 (A is not true).

Original statement:	$A + AB = A$
By Substitution for A,	$0 + 0 \cdot B = 0$
Applying statement 9,	$0 + 0 = 0$
Applying statement 3,	$0 = 0$

Thus, when the condition for A is false the entire equation is false.

Summarizing these two conclusions, it can be seen that the resulting truth of this expression has no dependence on the variable B but is directly dependent on the state of variable A.

The result can be verified by evaluating the electrical equivalence of the expression as shown in Figure 5-22.

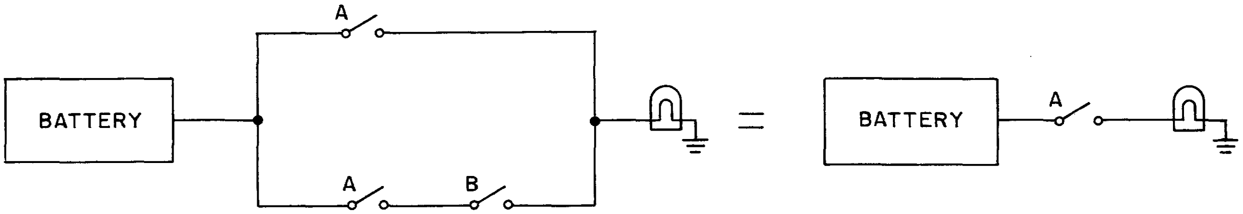


Figure 5-22. Switch Equivalence of A + AB = A

From this diagram, it can be seen that a path exists any time A is true (switches closed) and there are no paths completed when A is not true (switches labeled A open). Therefore, the output depends entirely upon A and the term AB is redundant.

In the format of statement 15, a single term may contain more than one variable as long as all the variables in one ORed term appear in the other.

Example

Original Statement:	$A + AB = A$
Substitute xyz for A:	$xyz + xyzB = xyz.$
Substitute lm for B:	$xyz + xyzlm = xyz.$

This final expression, $xyz + xyzlm = xyz$, fulfills the format requirements for statement 15.

Table 5-11 shows expressions simplified by statement 15. It also shows formats which are commonly mistaken as being properly reduced entirely with statement 15.

TABLE 5-11. PROPER AND IMPROPER FORMAT FOR STATEMENT 15

Correct format and application of statement 15.

(1) $\overline{C} + \overline{C}D = \overline{C}$

(2) $AB\overline{C} + AB\overline{C}DEF = AB\overline{C}$

(3) $(\overline{ABC}) + (\overline{ABC})\ (DEF) = (\overline{ABC})$

(4) $AB + ABC + DEF = AB + DEF$

Incorrect format - cannot be resolved by statement 15 alone.

(1) $AB + ACD$ (Both variables in the left hand term do not appear in the right hand term).

(2) $A\overline{B} + ABD$ (The variables in the left hand term are not in the same form as they are in the right. Although both A and B appear as variables, in the left term B is negated, in the right hand term it is not negated. This prevents invoking statement 15.)

Solve the following problems by indicating if the answer given is true or false. In the space to the right of the problems marked false explain why these problems are incorrect.

T F (58) $ABDEF + DE = DE$

T F (59) $\bar{A}\ \bar{B} + A\ \bar{B}\ \bar{C} = \bar{A}\bar{B}$

T F (60) $(AB) + C \quad + DE \quad (AB) + C \; = \; (AB) + C$

T F (61) $F\ 310\ F220 + F400\ F310\ F220 = F310\ F220$

STATEMENT 16

$A(B + C) = AB + AC$

This statement is most often used to manipulate already existing expressions. It does not actually eliminate any redundancies, but allows an expression to be written into a form for grouping with other terms. Quite often this grouping will allow reduction via another statement.

The electrical equivalence of this rule, Figure 5-23, shows that in order for the bulb to light, switch A must be closed (A present or true) along with either switch B or switch C.

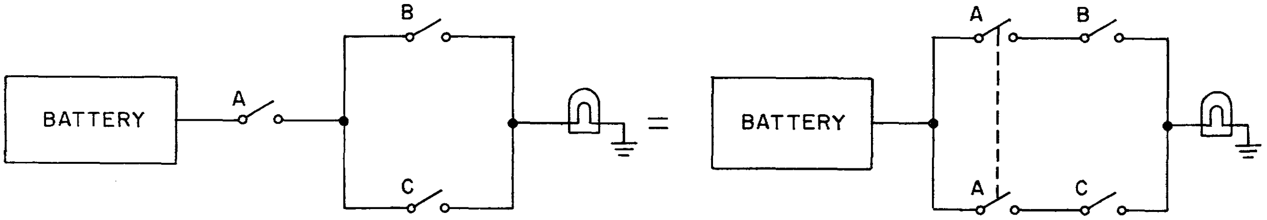


Figure 5-23. Switch Equivalence of Statement 16.

The switches labeled A in the right-hand diagram are joined since they must be operated together. This is dictated by the fact that in a boolean expression any given letter can stand for only one variable; thus, all terms labeled A, for example, must be in the same state (all true or all false) at a given time.

To see the significance of statement 16, follow this example through to its simplest form.

Example 1

Initial expression to be simplified: $A(B + C) + AB = ?$

Apply statement 16 to left-hand term: $AB + AC + AB =$

Apply statement 6: $AB + AC$

Apply statement 16 in reverse: $A(B + C)$

The last step in the example could have been omitted, if desired, since either format is acceptable.

This brief, simple example points out the use of statement 16 in that, what was not an apparent application of statement 6 became obvious after the original expression was expanded. Furthermore, in its original grouping, the variables within the term A(B + C) could not have been used separately to simplify other areas of a longer expression. Consider the following example.

Example 2

Expression to be simplified: $A\ (BC + DE) + B + \overline{ADE}$

Statement 16: $ABC + ADE + B + \overline{ADE}$

Statement 15: $ADE + B + \overline{ADE}$

Statement 12: $1 + B$

Statement 8: 1

Here the final form of the original expression is 1. In its original format, without expanding the term A(BC + DE), it is not apparent that reduction would result. However, upon expanding this single term, it is obvious that the expression was not in its simplest form.

Work the following problems indicating if the answer given is true or false. To the right of the problems marked as false, explain why each is not a correct statement.

T F (62) $\overline{A}(A + B) = \overline{A}\ A + \overline{A}\ B$

T F (63) $AB(D + E) = ABD + E$

T F (64) $AB(BC + BD) = ABBC + ABBD$

T F (65) $\overline{A}\ (B + \overline{A}BC) = \overline{A}\ B + \overline{A}\ \overline{A}\ B\ C$

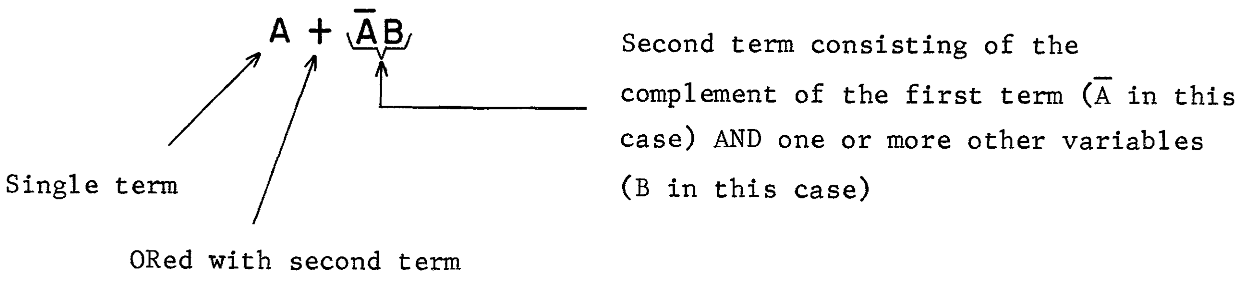
T F 66) $ABCD + ADEF = AD\ (BC + EF)$

By circling the number, indicate which of the problems just completed can be further simplified. Check to see if you are correct. For practice, perform these simplifications applying all the rules covered to this point.

STATEMENT 17

$A + \bar{A}\ B = A + B$

This statement allows the removal of a redundant term through use of statement 1. Investigation of the format should reveal that it consists of a single term (A in this case) ORed with its complement which is ANDed with one or more other variables. This is illustrated below.


A + ĀB
Single term
ORed with second term
Second term consisting of the complement of the first term (Ā in this case) AND one or more other variables (B in this case)

Other examples of the correct format are:

$AB + (\overline{AB})\ C = AB + C$

$\bar{A} + AB = \bar{A} + B$

$\overline{\bar{A}\ C} + \bar{A}CD = \overline{\bar{A}C} + D$

The student should be sure that the correct format is understood before proceeding. As practice, indicate which of the following problems are in the correct format for applying statement 17. Check your answers.

(1) $\bar{A}B + A$

(2) $AB + ABC$

(3) $ABC + \overline{AB}DE$

The explanation of this rule can most easily be seen by reviewing the electrical circuit equivalent given in Figure 5-24.

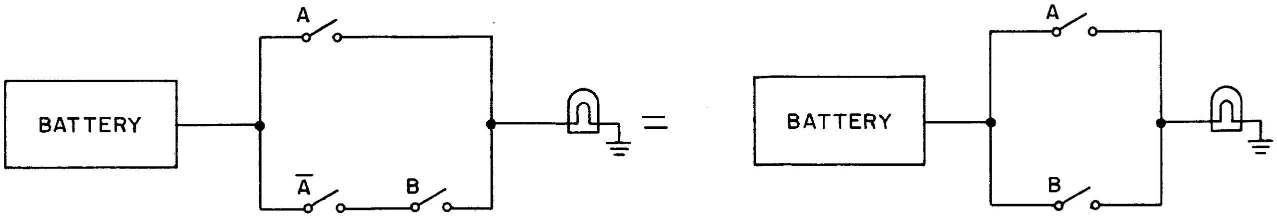


Figure 5-24. Switch Equivalence of Statement 17.

From the circuits, it can be seen that closing switch A (making condition A true) lights the lamp in either case. If the top path were open (switch $\underline{A}$ open or A $\underline{=}$ 0) then, by statement 1, the presence of (switch $\bar{A}$ closed OR $\bar{A}$ = 1) is ensured. Therefore, in absence of the top path, the only variable whose state has_to be determined is B. It is unnecessary to restate (include) the switch $\bar{A}$ since it has already been established that it closes if switch A is open.

This rule, in addition to removing redundant terms, also assists in making clear a long expression which does not appear as if it can be readily simplified. The following example illustrates this point.

Example

Original expression: $A + \bar{A}B + BCD$

Applying statement 17: $A + B + BCD$

Applying statement 15: $A + B$

In the example, eliminating the redundant $\bar{A}$ (using statement 17) made it obvious that the entire term BCD was redundant by statement 15. Thus, the application of statement 17, in addition to removing a redundant term, has made further reduction more obvious.

Also notice in the example that statement 17 applies even though there is an added term BCD. This can be done as long as any additional terms are ORed.

Work the following problems by indicating if the answers given are true or false. Explain why each of the answers marked false does not fit the format for statement 17.

T F 67) $x + A\bar{x} = x + A$

T F 68) $Ax + \overline{Ax}\ y = Ax + y$

T F 69) $AB + AB\bar{x} = AB + \bar{x}$

T F 70) $AB + xy + \overline{AB}\ CD = AB + CD + xy$

STATEMENT 18

$\overline{A + B} = \bar{A}\ \bar{B}$; $\overline{AB} = \bar{A} + \bar{B}$

This statement is made up of two distinct parts. For identification purposes they are referred to as follows:

Part 1: $\overline{A + B} = \bar{A}\ \bar{B}$

Part 2: $\overline{AB} = \bar{A} + \bar{B}$

The need for this rule arises from the desire to break down entirely negated expressions into individual terms. The purpose of this is to facilitate additional simplification by allowing these new terms to be combined with other terms in the entire expression through the use of previous rules. As an illustration refer to the following example.

Example 1

1) $B + \bar{D} + E + (\overline{BCD})$

2) $B + \bar{D} + E + \bar{B} + \bar{C} + \bar{D}$

3) $B + \bar{D} + E + \bar{B} + \bar{C}$

4) $1 + \bar{D} + E + \bar{C}$

5) 1

In example 1, the original expression (1) shows no apparent rearrangements; however, since the terms under the negation bar are similar to those outside the negation, breaking down of this term may be expected to yield additional simplification. The term under the bar is read "not the quantity BCD" and the factors within this term cannot be used individually until the negation bar is broken. By applying statement 18, part 2, expression (2) is obtained. Here the bar has been broken and the individual factors can be grouped with other terms. Expression (3) is arrived at by applying statement 6 to eliminate the redundant $\bar{D}$. In expression (4), the values $B + \bar{B}$ are reduced to 1 by statement 12; and, in the final expression (5), the result equals 1 by statement 8.

The development of statement 18 was accomplished by a man named De Morgan and in his honor, statement 18 is sometimes called De Morgan's Theorem. This theorem is explained by breaking each part into two sections. The first includes a logical explanation; the second discusses the mechanical operations necessary to perform the conversions.

Part 1

Inspection of the first part of statement 18 $(\overline{A + B} = \bar{A}\ \bar{B})$ shows that it is read "not the quantity A OR B." If "the quantity" is not present it means that both A and B are not present. This is concluded from the definition of OR, which reads "the quantity is true if A OR B is true". Since the quantity is not true, implied by the negation bar over the entire quantity, neither of the variables is true. Thus $(\overline{A + B}) = \bar{A}\ \bar{B}$. Notice that the final expression is the AND function of each of the variables and that each of the variables has been negated. It is important to note that $(\overline{AB})$ and $(\bar{A}\ \bar{B})$ are not equal to each other.

To mechanically perform the operation, the following statement is used: To break the negation bar which combines several ORed terms, change each OR sign to the AND sign and negate the individual variables, for example:

$$(\overline{A + B + C}) =$$

$$\bar{A} \cdot \bar{B} \cdot \bar{C}$$

Negate individual terms

change OR's to AND's

Additional Examples of Statement 18, Part 1:

1) $(\overline{A + B + C + D}) = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$

2) $(\overline{A + B + C}) + D = (\bar{A} \cdot \bar{B} \cdot \bar{C}) + D$

Notice that the term D is not affected as it is not included under the negation.

3) $(\overline{\bar{A} + B + C}) = \bar{\bar{A}} \cdot \bar{B} \cdot \bar{C} \cdot = A \cdot \bar{B} \cdot \bar{C}$

In this case following the rule and negating each factor causes variable A to be negated twice. Statement 14 allows this to be simplified to A.

4) $(\overline{A + B + \bar{C}})\ (W + X + Y) = (\bar{A} \cdot \bar{B} \cdot C)\ (W + X + Y)$

As in example 3, the second parenthetical factor is not changed since the negation does not extend over this factor.

5) $(\overline{A + B + C}) + (\overline{D + E + F}) = (\overline{A} \cdot \overline{B} \cdot \overline{C}) + (\overline{D} \cdot \overline{E} \cdot \overline{F})$

Here each term must have the rule applied separately as the bar is drawn separately over each term.

6) $(\overline{A + B + C) + (D + E + F}) = (\overline{A} \cdot \overline{B} \cdot \overline{C}) \cdot (\overline{D} \cdot \overline{E} \cdot \overline{F})$

In this case the bar includes both terms, so the rule is applied to the entire expression.

Work the following problems and check your answers.

71) $(\overline{X + Y})\ (A + B) =$

72) $(\overline{A + B})\ (\overline{L + M + N}) =$

73) $(\overline{A + B) + (M + P + R}) =$

74) $(\overline{AB + C}) =$

75) $(\overline{AB + CD + E}) =$

If correct answers are not obtained for these problems, reread the area on part 1 of statement 18 before continuing.

Statement 18, part 2, takes into account the converse of part 1; that is, it covers a group of ANDed factors which are joined together by a negation bar.

Part 2

$(\overline{AB}) = \overline{A} + \overline{B}$

In this case "not the quantity AB" implies that AB = 0. If this is true, then, by the rules of the AND functions, one of the following three conditions of A and B must have been true:

1) $A = 0$ or $B = 0$ $\quad 0 \cdot 0 = 0$

2) $A = 0$ or $B = 1$ $\quad 0 \cdot 1 = 0$

3) $A = 1$ or $B - 0$ $\quad 1 \cdot 0 = 0$

These three conditions can be summarized by the statement: Either A or B must equal zero. Written logically this would appear as A + B. This is the result given for applying statement 18, part 2, to (AB). In summary this could be written as:

$$\overline{(AB)} = \bar{A} + \bar{B}$$

The mechanics for applying part two are similar to those for part one. For part two, the rule states: Break the bar by changing the AND sign to the OR sign and negate each term. An example is shown below.

$$\overline{xyz} =$$

$$\bar{x} \underset{\nwarrow \text{Change AND's to OR's}}{+} \bar{y} + \bar{z} \longleftarrow \text{Negate individual terms}$$

Additional Examples of Rule 18, Part 2:

1) $\overline{(ABCD)} = \bar{A} + \bar{B} + \bar{C} + \bar{D}$

2) $\overline{(ABC)}D = (\bar{A} + \bar{B} + \bar{C})\ D$

Notice that the variable D is not affected as it is not included under the negation. Also, it must retain its relationship of being ANDed with the entire quantity.

3) $\overline{(AB\overline{C})}\quad (WXY) = (\bar{A} + \bar{B} + C)\quad (WXY)$

Notice (WXY) is not affected since the negation bar does not extend over it.

5) $\overline{(ABC)}\quad \overline{(DEF)} = (\bar{A} + \bar{B} + \bar{C})\quad (\bar{D} + \bar{E} + \bar{F})$

Here each parenthetical factor must be handled spearately as the bar does not extend over both, but is drawn over each factor separately. Note that the AND relationship between the two quantities is retained.

6) $\overline{(ABC)\quad (DEF)} = \bar{A} + \bar{B} + \bar{C} + \bar{D} + \bar{E} + \bar{F}$

As the bar extends over both parenthetical factors, all the AND signs are changed to OR signs and each resultant term is negated.

Work the following problems and check your answers.

76) $(x + y)\quad \overline{(A \cdot B)} =$

77) $(\overline{AB}) \quad (\overline{LMN}) =$

78) $\overline{AB} + \overline{CD} + \overline{EF} =$

A general rule which permits the simplification of any expression which is negated is expressed as follows:

> To find the complement of a boolean expression, change all + signs to •, all • signs to +, and replace each variable by its complement.

Work the following problems dealing with De Morgan's Theorem. Check your work.

80) $\overline{A \cdot B \cdot C} =$

81) $\overline{A + BC} =$

82) $(\overline{\overline{A + B}) + (\overline{C + D}})=$

83) $\overline{AB + AC + BCD} =$

84) $(\overline{A + BC) (A + C}) =$

85) $\overline{A + B} =$

86) $\overline{A + B + C} =$

87) $(\overline{\overline{A + B}) (C + D}) =$

Quite often the rules you have just learned are expressed as a set of laws. Let's examine those laws briefly to ensure that they agree with the rules you have learned.

1. Laws of Intersection

   a. A factor combined with a logical 1 in an AND function is equal to itself. e.q., $A \cdot 1 = A$

   b. A factor combined with a logical 0 in an AND function is equal to zero. e.q., $A \cdot 0 = 0$

2. Laws of Union

   a. A term combined with a logical 1 in an OR function is equal to one. e.q., $A + 1 = 1$

   b. A term combined with a logical 0 in an OR function is equal itself. e.q., $A + 0 = A$

3. Laws of Complements

   a. A factor combined with its complement in an AND function is equal to zero. e.q., $A \cdot \bar{A} = 0$

   b. A term combined with its complement in an OR function is equal to one. e.q., $A + \bar{A} = 1$

4. Idempotent Laws

   a. A factor combined with itself in an AND function is equal to the factor by itself. e.q., $A \cdot A = A$

   b. A term combined with itself in an OR function is equal to the term by itself. e.q., $A + A = A$

5. Law of double negatives

   The negation of a negated term or group of terms results in the original term. e.q., $\bar{\bar{A}} = A$; $\overline{\overline{A(B + C)}} = A(B + C)$

6. Commutative Law

   If two or more variables are combined in either an AND or an OR function, the result is the same regardless of the order in which the variables appear.
   e.q., $A \cdot B = B \cdot A$; $A + B = B + A$

7. Laws of Absorption

   Any expression involving <u>both</u> AND and OR function can be simplified if:

   a. Factors common to all terms are used
      e.q., $A(A + B) = A$
      $A + A \cdot B = A$

   b. A factor and its complement are used:
      e.q., $A(\bar{A} \pm B) = AB$
      $A + \bar{A} \cdot B = A + B$

8. Distributive Law

   If two or more variables are combined using both AND and OR functions, the final result is equal to the combination of all variables in both functions.
   e.q., $A(B + C) = AB + AC$
   $A + (B \cdot C) = (A + B)(A + C)$

9. Associative Law

   If two or more variables are combined in an AND or an OR function, the result is the same regardless of how the variables are grouped. e.q., $A(BC) = A \cdot B \cdot C$

$A + (B + C) = A + B + C$

10. De Morgan's Law

Any negated boolean expression can be simplified by changing all AND functions to OR functions and all OR functions to AND functions, and replacing each term with its negated value.

e.q., $\overline{A \cdot B} = \overline{A} + \overline{B}$

$\overline{A + B} = \overline{A} \cdot \overline{B}$

Note: This law is sometimes stated simply as "You may break the line _if_ you change the sign".

Simplify the following expressions in accordance with the laws just covered. The answers are at the end of this chapter.

1. Intersection

88 $B \, . \, 1 =$

89 $A \, . \, B \, . \, 1 =$

90 $A + B \, . \, 1 =$

91 $F \, . \, 0 =$

92 $(A + B) \, . \, 0 =$

93 $1 \, . \, (A + B + C) =$

94 $AB \, . \, 0 =$

95 $(A + B + CD) \, . \, 0 =$

2. Union

96 $A + 1 =$

97 $A \, . \, B + 1 =$

98 $A + B + C + 1 =$

99 $A\,C + B\,C + A\,B + 1 =$

100 $A + 0 =$

101 $A\,B\,C + 0 =$

102 $A + B + C + 0 =$

103 $A + B \, . \, C \, . \, D \, . \, 0 =$

3. Complements

104 $A \, . \, \overline{A} =$

105 $A\,B\,\overline{B}\,C =$

106 $C \, . \, A\,B\,\overline{C} =$

108 $\overline{A} + A =$

109 $(\overline{B} \, . \, 1) + B =$

110 $(D + \overline{A}\,A\,B)\,(C\,E\,\overline{E} + \overline{D}) =$

107 $(A + 0)\ (C + D\,\overline{D}) =$

111 $A\,B + \overline{A\,B} =$

4. Idempotent

112 $A \cdot A =$

113 $A\,A\,B\,B =$

114 $R\,\overline{A\,A} + R\,A\,A =$

115 $(A + \overline{B}\,B\,C)\quad (D + 1)A =$

116 $A + A =$

117 $\overline{A} \cdot \overline{A} + \overline{A} =$

118 $A\,B\,\overline{B} + C + C =$

119 $A\,B\,\overline{A} + (B\,\overline{B}\,C + C) + C(\,D + 1) =$

5. Double Negatives

120 $\overline{\overline{A}} =$

121 $\overline{\overline{A}}\,\overline{\overline{B}} =$

122 $\overline{\overline{A\,B}} =$

123 $\overline{\overline{A\,B}\,C} =$

124 $A\,(\overline{B + \overline{C}}) =$

125 $\overline{\overline{A}\,B\,C} + (\overline{A} + \overline{B\,C}) =$

126 $\overline{\overline{\overline{\overline{\overline{\overline{A} \cdot B} \cdot C} \cdot D} \cdot E} \cdot F} =$

127 $\overline{A\,\overline{B} + \overline{C\,D} + \overline{A\,B}} =$

6. Commutative

128 $A\,B =$

129 $A(B + C) =$

130 $AB(C + DE) =$

131 $\overline{A} + BC + BC\overline{D} + EF + G =$

132 $A + B =$

133 $AB + CD + EF =$

134 $A(A + B + C + 1)\,\overline{A}\,D =$

135 $\overline{A}\left[B + \overline{C}D + (D + \overline{D})\right]\overline{A}\,B\,C =$

7. Absorption

136 $A(A + B) =$

137 $A(A + \overline{A}\,B) =$

138 $C(C + D + E + F) =$

139 $A\,B + C\,D + A\,B\,C\,D =$

140 $A + A\,B =$

141 $\overline{A}\,B\,(C + D) + B =$

142 $D + A\,B\,C\,D =$

143 $(A + \overline{B} + \overline{C})\ (A\,B\,C) =$

8. Distributive

144 $A\,(\,B + C) =$

145 $A\,C + C\,D =$

148 $A\,A + A\,B =$

149 $A + B\,C =$

146 $ABC + ABD + ABEF =$

147 $C\ (\overline{C} + \overline{D}) =$

150 $A + \overline{A}\ B$

151 $A + (\overline{A}\ B\ C\ D) =$

9. Associative

152 $A\ (B\ C) =$

153 $(AB)\ C + (AB)\ C =$

154 $(A)\ (B)\ (C) =$

155 $A \cdot (B) \cdot (CD) =$

156 $A + (B + C + D) =$

157 $(A+B) + (C + D) =$

158 $(\overline{B} + B) + (\overline{D} + D) =$

159 $(A\overline{A}) + (B\overline{B}) + (C\overline{C}) =$

10. DeMorgan's Law

160 $1 \neq 0\ ,\ 0 \neq 1\ , \therefore\ \overline{A\overline{A\overline{A}}} =$

161 $\overline{A \cdot B} =$

162 $\overline{A\ B\ C} =$

163 $\overline{A\ B + C\ D} =$

164 $\overline{AB + BC + CD} =$

165 $\overline{(\overline{A+B+C})\ (\overline{D+EF})} =$

166 $\overline{A + B + C} =$

167 $(\overline{A+B})\ (\overline{C + D}) =$

VEITCH DIAGRAMS

Now that we have proven the statements or laws that govern operations on boolean expressions, let's examine an easier method of simplifing boolean equations. This method employs the use of the Veitch Diagram.

Perhaps there were times on preceding exercises that you were undecided whether or not you had the equation in its simplest form. The Veitch diagram automatically provides the simplest expression of a given equation.

Each variable of an equation has two states. The Veitch diagram is a matrix that allows expression of all possible combinations ($2^2 = 4$), which are:

AB = 00
01
10
11

Therefore, the diagram for the expression AB would have four squares as illustrated by Figure 5-25.

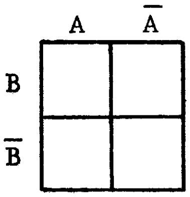


Figure 5-25. Veitch Diagram of Two Variables

The boolean equation AB + ABC would require a three-variable Veitch diagram as shown in Figure 5-26. Although the first term contains only two factors (2^2 combinations), the second term contains three (2^3 or eight combinations) factors

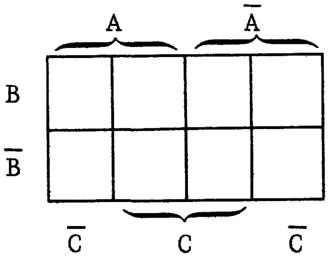


Figure 5-26. Veitch Diagram of Three Variables.

What type of diagram would the equation AC + ABD require? If your answer is a three-variable diagram, you are wrong. True, the second term contains only three but there are four variables in the entire expression which requires a four-variable Veitch diagram (Figure 5-27).

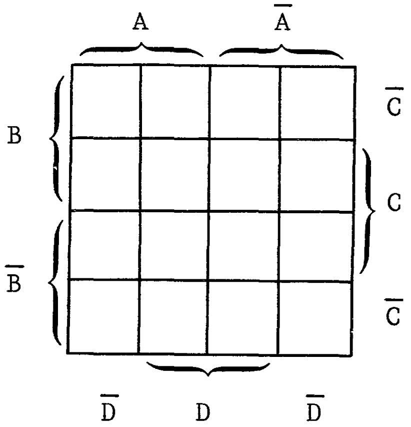


Figure 5-27. Veitch Diagram of Four Variables.

A five-variable equation would have 2^5 or 32 combinations and would be a diagram containing 32 squares (Figure 5-28). Compare the diagrams of Figure 5-27 and 5-28 and note the differences.

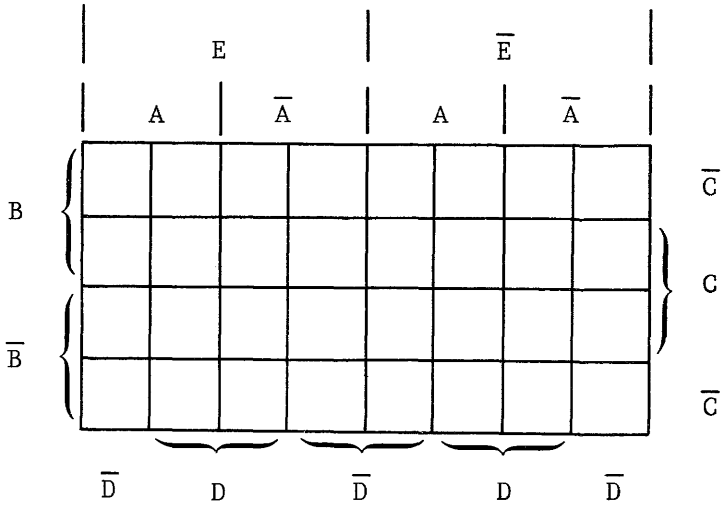


Figure 5-28. Veitch Diagram of Four Variables.

The six-variable boolean expression would require the size of it's diagram to be doubled over that of Figure 5-28, as illustrated in Figure 5-29. You probably already know why -- because 2^6 equals 64 and a six-variable expression could have 64 possible combinations.

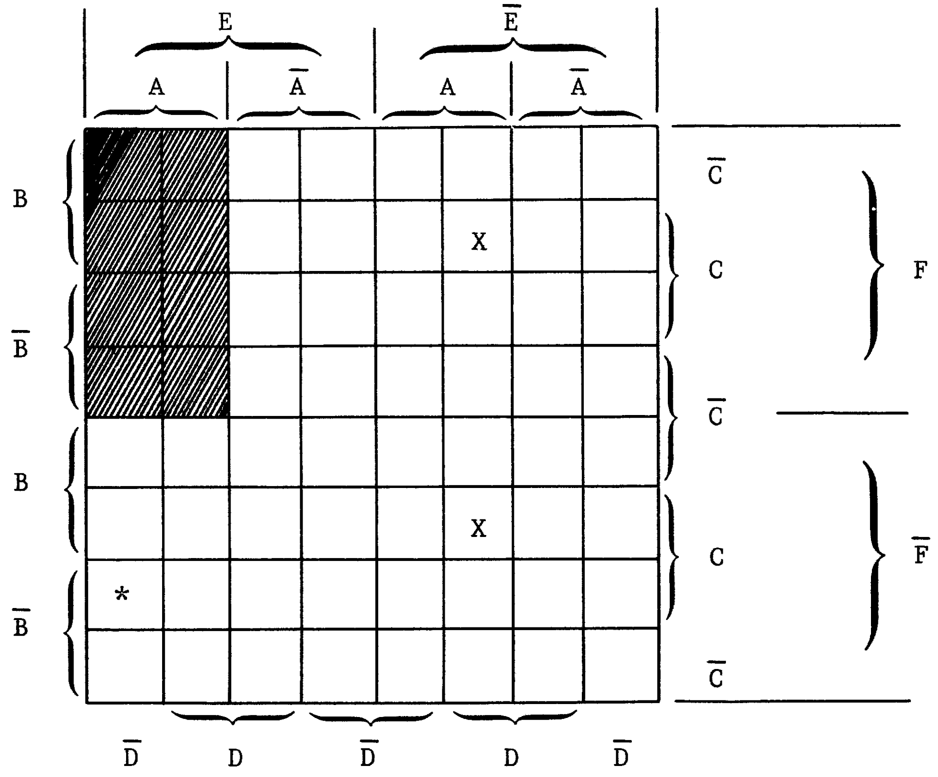


Figure 5-29. Veitch Diagram of Six Variables.

The terms surrounding the diagram may be arranged in other configurations as long as any six-variable expression can be defined. For example, the term $A \bar{B} C \bar{D} E \bar{F}$ is defined by the square containing the asterisk (Figure 5-29).

The term $A B C D \bar{E}$ of a six variable expression does not define the condition of F. Therefore, the two squares that contain the "X" both define $A B C D \bar{E}$.

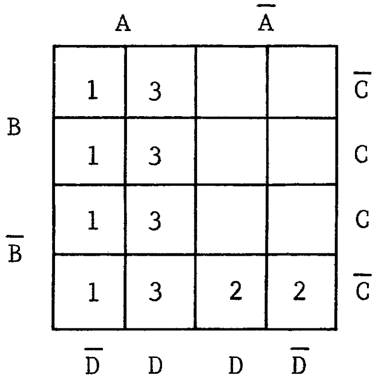
The shaded portion of the diagram defines the term AEF. Eight squares are required to define this term because B, C, and D are not defined.

PLOTTING THE EQUATION

Now that you understand how to construct a Veitch diagram for a given number of variables, plot the expression $A \bar{D} + \bar{A} \bar{B} \bar{C} + A D$ on the diagram in Figure 5-30.

NOTE:

"Not bars" may not include more than one term. $(\overline{AB} = A + \overline{B})$.



	A		$\overline{A}$		
B	1	3			$\overline{C}$
	1	3			C
$\overline{B}$	1	3			C
	1	3	2	2	$\overline{C}$
	$\overline{D}$	D	D	$\overline{D}$	

Figure 5-30. Veitch Diagram of Four Variables.

1. Place a 1 in each square that defines the first term of the expression. $(\overline{AD})$

2. Place a 2 in each square that defines the second term $(\overline{A}\ \overline{B}\ \overline{C})$.

3. Place a 3 in each square that defines the third term (AD).

This process would continue for each term until the entire expression has been plotted.

NOTE:

The numbers in the squares have no significance except to denote which term justified filling the square. After you become acquainted with the procedure, all numbers may be substituted with an X, if desired. More than one number in a given square indicates redundancy in the original equation, but otherwise has no significance.

SIMPLIFICATION

The simplification of any plotted expression is conditioned by the following rules.

1. If a number appears in adjacent squares or at opposite ends of a row or column, one of the variables may be dropped.

2. If any complete row or column, any block of four squares, or the four corner squares contain a number, two of the variables may be dropped.

3. If any two adjacent rows or columns, the top and bottom rows, or the two outside columns are completely filled with a number, three of the variables may be dropped.

4. All involved squares (containing a number) must be considered in the final equation. A square may be used more than once if necessary.

Figure 5-31 is a reproduction of the diagram on which the expression $A\bar{D}$ + $\bar{A}\ \bar{B}\ \bar{C}$ + A D was plotted (Figure 5-31). In accordance with the preceding rules, the equation could be simplified as follows:

	A		$\bar{A}$		
B	X	X			$\bar{C}$
	X	X			C
$\bar{B}$	X	X			C
	X	X	X	X	$\bar{C}$
	$\bar{D}$	D	D	$\bar{D}$	

Rows - Horizontal

Columns - Vertical

Figure 5-31. Simplification of Expression $A\ \bar{D} + \bar{A}\ \bar{B}\ \bar{C}\ + A\ D$

1. The two left columns would translate as A (Rule 3) because that variable is common to all eight squares.

2. The bottom row would translate as $\bar{B}\ \bar{C}$ (Rule 2) because the entire row is common to those two variables.

Therefore, because all marked squares have been considered at least once, the simplified equation for $A\ \bar{D} + \bar{A}\ \bar{B}\ \bar{C} + A\ D$ would be $A + \bar{B}\ \bar{C}$.

Let's simplify another boolean expression using the three-variable Veitch Diagram. Apply the same rules used to simplify the four-variable expression.

Plot and simplify the equation $A\,\overline{A}\,B + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}$ using the diagram in Figure 5-32.

	A		$\overline{A}$	
B				
$\overline{B}$	2		3	3
	$\overline{C}$	C	C	$\overline{C}$

Figure 5-32. Simplification of Expression $A\,\overline{A}\,B + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}$.

1. The first term cannot be plotted because there are no squares common to $A\,\overline{A}\,B$. The law of complements also proves this fact (i.e., $A \cdot \overline{A} = 0$).

2. The second term would justify the square that contains the 2.

3. The third term would justify the squares that contain a 3.

The Veitch Diagram may be visualized with the two ends connected in the form of a cylinder. This would then make the two bottom corner squares adjacent and one term could be dropped (simplification Rule 1). The top and bottom may also be connected to form a cylinder.

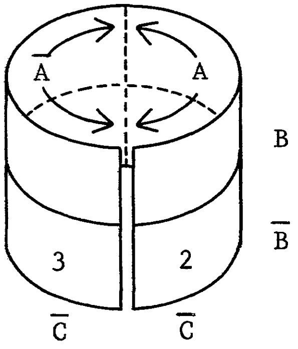


Figure 5-33. Veitch Diagram as a Cylinder.

These two adjacent squares would be defined as $\overline{B}\,\overline{C}$.

The two adjacent squares in Figure 5-32 would be defined as $\bar{A}\ \bar{B}$. Therefore, the simplified equivalence of $A\ \bar{A}\ B + A\ \bar{B}\ \bar{C} + \bar{A}\ \bar{B}$ would be $\bar{A}\ \bar{B} + \bar{B}\ \bar{C}$. The simplified expression is not readily apparent without the aid of the Veitch Diagram.

The preceding example could be expresses as illustrated in Figure 5-34. Notice that the variables are rearranged on the diagram (Compare Figures 5-32 and 5-34).

	B		$\bar{B}$	
A				2
$\bar{A}$			3	3
	$\bar{C}$	C	C	$\bar{C}$

Figure 5-34. Three Variable Veitch Diagram (rearranged)

Plot the same equation on the new diagram $(A\ \bar{A}\ B + A\ \bar{B}\ \bar{C} + \bar{A}\ \bar{B})$.

1) Again, the first term cannot be plotted.

2) The second term justifies the square containing a 2.

3) The third term justifies these squares containing a 3.

The column translates as $\bar{B}\ \bar{C}$ and the two horizontally adjacent squares translate as $\bar{A}\ \bar{B}$. Again, the equation simplifies to $\bar{A}\ \bar{B} + \bar{B}\ \bar{C}$.

Simplify the boolean expression $A\bar{B} + ABC + A\bar{B}CD + \bar{C}\ \bar{D}$ using the diagram in Figure 5-35.

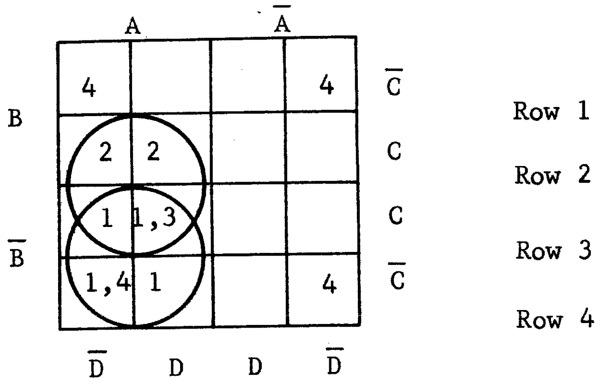


Figure 5-35. Simplification of Expression $A\bar{B} + ABC + A\bar{B}CD + \bar{C}\ \bar{D}$.

1. The block of four squares (Rows 2 and 3) translate as AC (Rule 2).

2. The block of four squares (Rows 3 and 4) translate as $A\overline{B}$ (Rule 2).

3. The four corners translate as $\overline{C}\ \overline{D}$ (Rule 2).

Sketch a diagram similar to Figure 5-35 but interchange the C and a $\overline{C}$ factors; also interchange the D and $\overline{D}$ factors. The four squares that contain a 4 in Figure 5-35 would now be a block of four. This justifies "the four corners" portion of simplification Rule 2.

The simplified expression of $A\overline{B} + ABC + A\overline{B}CD + \overline{C}\ \overline{D}$ is $AC + A\overline{B} + \overline{C}\ \overline{D}$. Switch analysis of the equation will prove the solution.

Plot the boolean expression $A\overline{D} + \overline{A}B + \overline{A}\ \overline{B} + A\overline{B}D + ABD$ on the diagram in Figure 5-36. What is the simplified equivalence of the expression? Justify your answer and then check the solution at the end of the chapter.

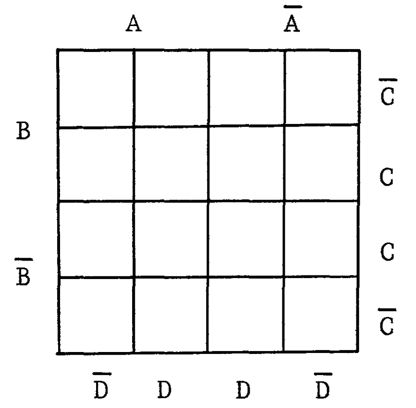


Figure 5-36. Veitch Diagram for Expression $A\overline{D} + \overline{A}B + \overline{A}\ \overline{B} + A\overline{B}D + ABD$.

Try_one more!_ If you correctly simplify the expression $A + \overline{A}B + \overline{A}\ \overline{B}\ \overline{C} + ABC\overline{C} + ABD + \overline{D}$, you understand the Veitch Diagram and it's use. Congratulations! If not, you win the BOOBEAN prize. Plot the equation in Figure 5-37 and simplify. Check your solution with the one at the end of the chapter.

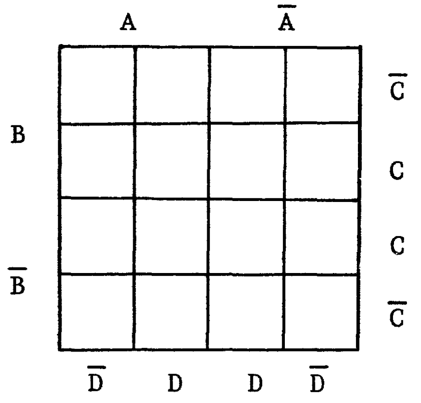


Figure 5-37. Veitch Diagram for Expression $A + \bar{A}B + \bar{A}\ \bar{B}\ \bar{C} + ABC\bar{C} + ABD + \bar{D}$.

The Veitch Diagram may also be used to find the complement of an equation. This is accomplished by plotting the equation, as before, but then consider the blank squares instead at the numbered squares. What would be the complement of the equations plotted in Figures 5-35, 5-36, and 5-37? Check your solution with those at the end of the chapter.

You will learn in subsequent chapter that the main building block of a CDC computer is the inverter, an electronic circuit that complements between input and output. A logical "1" is complemented to a logical "0" but the "1" represents a boolean expression which must also be complemented.

Another way, in addition to the Veitch Diagram, to complement an equation is to "not" the entire equation and then apply De Morgan's Law (Statement 18).

For example, the complement of the equation $A + B + \bar{C} + D$ would be expressed as $\overline{A + B + \bar{C} + D}$, which is simplified to $\bar{A}\ \bar{B}\ C\ D$. Compare your two solutions to Figure 5-37.

Complement the following equations using either the Veitch Diagram or De Morgan's Law. Compare your answers with these at the end of the chapter.

168. $A + \bar{B}\ C$

169. $\overline{A\ \bar{B}\ C\ \bar{D}\ E\ \bar{F}}$

170. $\overline{A + \overline{B + \overline{C + \overline{D}}}}$

171. $\overline{GH} + KZ + \overline{NO}$

172. $P\ \bar{R} + \bar{R}\ \bar{S} + \bar{T}\ \bar{R}$

173. $A + B + C + \overline{AB} + \overline{AC} + \overline{BC}$

174. $\overline{\overline{\bar{V} + \bar{E} + \bar{R}}} + \bar{Y}$

175. $\overline{\overline{\overline{(GO)} + \overline{(OD)}}}$

ANSWER TO REVIEW QUESTIONS

37	0	Rule 3
38	1	Rule 4
39	1	Rule 4
40	1	Rule 5
41	0	Rule 3
42	0	Rule 5
43	0	Rule 9
44	A	Rule 11
45	1	Rule 5
46	0	Rule 5
47	1	Rule 8
48	A	Rule 10
49	0	Rule 9
50	1	Rule 12

51 $A + \overline{A}\ A + A + 0$ (Rule 13) $= A$ (Rule 10)

52 $AB + \overline{BA} + AA = AB + \overline{BA} + A$(Rule 7) $= 1 + A$ (Rule 12) $= 1$ (Rule 8)

53 $\overline{A}A + A + \overline{A} + A + 1 = 1$ (Rule 8)

54 $\overline{A} + \overline{B} + A + B + \overline{B} = \overline{A} + \overline{B} + A + 1$ (Rule 12) $= 1$ (Rule 8)

55 AB (Rule 6)

56 $\overline{AB} + CD$ (Rule 14)

57 (C+1) (C.1) + (0.A) (1+A) = (1 (Rule 8)) (C (Rule 11)) + (0 (Rule 9))
(1) (Rule 8)) = (1)(C)+(0)(1) = C(Rule 11) + 0(Rule 5) = C(Rule 10)

Statement 15

58 T

59 F not substantiated by Statement 15

60 T

61 T

Statement 16

62 T but if simplified $= 0 + \overline{A}\,B + \overline{A}\,B$

63 F $ABD + ABE$

64 T but if simplified $= ABC + ABD = AB\,(C+D)$

65 T but if simplified $= \overline{A}\,B$ (Rule 15)

66 T

Statement 17

67 F $= A$

68 F $= Y$

69 F $= AB$

70 F $= xy + CD$

Statement 18

71 $(\overline{X}\,\overline{Y})\,(A+B) = A\,\overline{X}\,\overline{Y} + B\,\overline{X}\,\overline{Y}$

72 $(\overline{A}\,\overline{B})\,(\overline{L}\,\overline{M}\,\overline{N}) = \overline{A}\,\overline{B}\,\overline{L}\,\overline{M}\,\overline{N}$

73 $(\overline{A}\,\overline{B})(\overline{M}\,\overline{P}\,\overline{R}) = \overline{A}\,\overline{B}\,\overline{M}\,\overline{P}\,\overline{R} = \overline{B}\,\overline{R}\,\overline{A}\,\overline{M}\,\overline{P} = \overline{P}\,\overline{R}\,\overline{A}\,\overline{M}\,\overline{B} =$ Forget it

74 $(\overline{AB})\,\overline{C}$ or $(\overline{A} + \overline{B})\overline{C}$ or $\overline{A}\,\overline{C} + \overline{B}\,\overline{C}$

75 $(\overline{AB})\,(\overline{CD})\,\overline{E}$

76 $(x+y)(\overline{A}+\overline{B}) = \overline{A}X + \overline{A}Y + \overline{B}X + \overline{B}Y$

77 $(\overline{A}+\overline{B})(\overline{L}+\overline{M}+\overline{N}) = \overline{A}\,\overline{L} + \overline{A}\,\overline{M} + \overline{A}\,\overline{N} + \overline{B}\,\overline{L} + \overline{B}\,\overline{M} + \overline{B}\,\overline{N}$

78 $(\overline{A}+\overline{B}) + (\overline{M}+\overline{P}+\overline{R}+) = \overline{A} + \overline{B} + \overline{M} + \overline{P} + \overline{R}$

79 $\overline{A} + \overline{B} + \overline{C} + \overline{D} + \overline{E} + \overline{F}$

Answers to Boolean problems simplified by using DeMorgan's Theorem.

80) $\overline{A} + \overline{B} + \overline{C}$

81) $\overline{A}\ (\overline{B} + \overline{C}) = \overline{A}\ \overline{B} + \overline{A}\ \overline{C}$

82) $(A+B) \cdot (C+D) = AC + AD + BC + BD$

83) $(\overline{AB})\ (\overline{AC})\ (\overline{BCD}) = (\overline{A}+\overline{B})\ (\overline{A}+\overline{C})\ (\overline{B}+\overline{C}+\overline{D}) = \overline{A}\ (\overline{B}+\overline{C}+\overline{D}) + \overline{B}\ \overline{C}$

84) $\overline{A}\ (\overline{B} + \overline{C})$

85) $\overline{A}\ \overline{B}$

86) $\overline{A}\ \overline{B}\ \overline{C}$

87) $(A+B) + \overline{C}\ \overline{D} = A + B + \overline{C}\ \overline{D}$

ANSWERS TO BOOLEAN LAWS

88	B
89	AB
90	A+B
91	0
92	0
93	A+B+C
94	0
95	0
96	1
97	1
98	1
99	1
100	A
101	ABC
102	A+B+C
103	A
104	0
105	0
106	0
107	AC
108	1
109	1
110	0
111	1
112	A
113	AB
114	$R\overline{A} + RA = R$
115	A
116	A
117	$\overline{A}$
118	C
119	C
120	A
121	AB
122	AB
123	$AB+\overline{C}$
124	$A\ \overline{B}\ C$
125	$AB + \overline{C} + \overline{AB} + \overline{C} = 1$
126	$A + \overline{B}C + \overline{D}E + \overline{F}$
127	ABCD
128	BA
129	(B+C) A = AB + AC = CA + AB
130	(C+DE) AB = (DE+C) BA
131	A+BC+EF+G = A+G+FE+CB
132	B+A
133	EF+DC+BA = FE+DC+BA

134 $A \bar{A} D = D \bar{A} A = A D \bar{A} = 0$

135 $\bar{A} B C = C \bar{A} B = B \bar{A} C$

136 A

137 $A + B$

138 C

139 $AB + CD$

140 A

141 B

142 D

143 ABC

144 $AB+AC$

145 $C (A+D)$

146 $AB (C+ D + EF)$

147 $C \bar{C} + C \bar{D} = C \bar{D}$

148 $A (A+B) = A$

149 $(A+B) (A+C)$

150 $(A+\bar{A}) (A+B) = A+B$

151 $(A+\bar{A})(A+B)(A+C)(A+D) = A (B+C+D)$

152 $ABC = A.B.C = (A) (B) (C)$

153 ABC

154 ABC

155 $ABCD$

156 $A+B+C+D$

157 $A+B+C+D$

158 $\bar{B}+B+\bar{D}+D = 1$

159 $A\bar{A} + B\bar{B} + C\bar{C} = 0$

160 $\bar{A}+A\bar{A} = \bar{A}+0 = \bar{A}$

161 $\bar{A} + \bar{B}$

162 $\bar{A} + \bar{B} + \bar{C}$

163 $(\bar{A}+\bar{B}) (\bar{C}+\bar{D})$

164 $(\bar{A}+\bar{B}) (\bar{B}+\bar{C}) (\bar{C}+\bar{D})$

165 $A+BC+D+EF$

166 $\bar{A} \bar{B} \bar{C}$

167 $\bar{A} \bar{B} \bar{C} \bar{D}$

Solution to Figure 5-36.

$$\begin{array}{c|cccc|c}
 & \multicolumn{2}{c}{A} & \multicolumn{2}{c}{\bar{A}} & \\
\hline
B & 1 & 5 & 2 & 2 & \bar{C} \\
B & 1 & 5 & 2 & 2 & C \\
\bar{B} & 1 & 4 & 3 & 3 & C \\
\bar{B} & 1 & 4 & 3 & 3 & \bar{C} \\
\hline
 & \bar{D} & D & D & \bar{D} &
\end{array}$$

The expression justifies a number (or X) in every square of the diagram as indicated above. This indicates that the expression is equal to 1. Simplification of the expression by using statements will prove it equal to 1.

Solution to Figure 5-37

$$\begin{array}{c|cccc|c}
 & \multicolumn{2}{c}{A} & \multicolumn{2}{c}{\bar{A}} & \\
\hline
B & 1,6 & 1,5 & 2 & 2,6 & \bar{C} \\
B & 1,6 & 1,5 & 2 & 2,6 & C \\
\bar{B} & 1,6 & 1 & & 6 & C \\
\bar{B} & 1,6 & 1 & 3 & 3,6 & C \\
\hline
 & \bar{D} & D & D & \bar{D} &
\end{array}$$

$$\underset{1}{A} + \underset{2}{\bar{A}\,B} + \underset{3}{\bar{A}\,\bar{B}\,\bar{C}} + \underset{4}{A\,B\,\bar{C}\,\bar{C}} + \underset{5}{A\,B\,D} + \underset{6}{\bar{D}}$$

1) The two left columns translate as A (Rule 3)

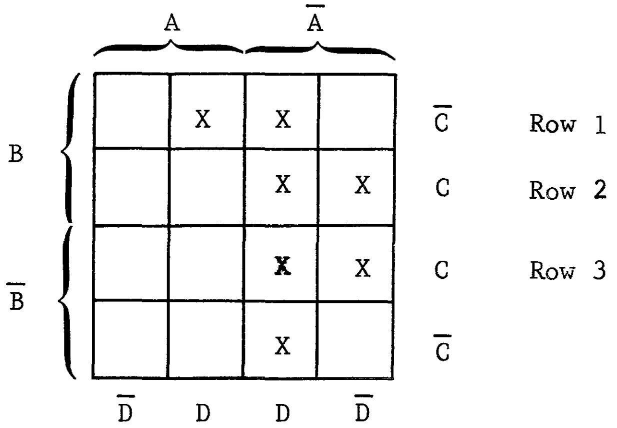
2) The two outside columns translate as $\bar{D}$ (Rule 3)

3) The top two rows translate as B (Rule 3)

4) The top and bottom rows translate as $\overline{C}$ (Rule 3). Remember the cylinder?

Therefore, the expression simplifies to $A + B + \overline{C} + \overline{D}$.

Solutions to Complements of Expressions



Solution to Figure 5-35.

1. The block of four squares translate as $\overline{A}$ C (Rows 2 and 3).

2. Column 3 translates as $\overline{A}$ D.

3. Two adjacent squares in Row 1 translate a B $\overline{C}$ D.

Therefore, the complement of the expression used in Figure 5-35 is equal to $\overline{A}C + \overline{A}D + B\overline{C}D$.

Solution to complement of Figure 5-36.

Because there are no blank squares, the solution to Figure 5-36 complemented is "0". Remember, the complement of "1" is "0".

Solution to complement of Figure 5-37.

Pretty easy! The only blank square is $\overline{A}$ $\overline{B}$ C D, which is the simplified expression.

Answer to boolean exercises (complements)

168 $\overline{A}\ (B + \overline{C})$

169 $A\ \overline{B}\ C\ \overline{D}\ E\ \overline{F}$

170 $(A+\overline{B})\ (C+\overline{D})$

171 $G\ H\ N\ O\ (\overline{K} + \overline{Z})$

172 $R + \overline{P}\ \ S\ \ T$

173 0

174 VERY

175 GOOD

Which is your grade?

SUMMARY

You should now understand the concepts of Boolean Algebra and how to simplify boolean expressions either by using the statements or the Veitch Diagram. Simplified notation has been emphasized throughout this chapter. The importance of this notation should be obvious when one considers the baseball game analogy presented early in the chapter.

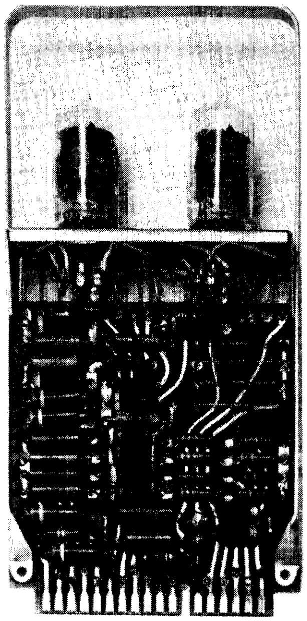
A design engineer uses simplified notation constantly to check and improve existing circuitry. It is possible to completely design a computer with simplified notation. The need for simplifying the statements should be obvious; it eliminates redundant circuits.

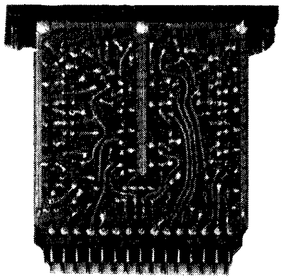
A maintenance technician needs to know the simplified notation because he must translate logical circuits to troubleshoot a malfunction. A circuit takes on a definite meaning after the output is translated. This distinguishes that circuit from several thousand others that also output logical ones and zeros.

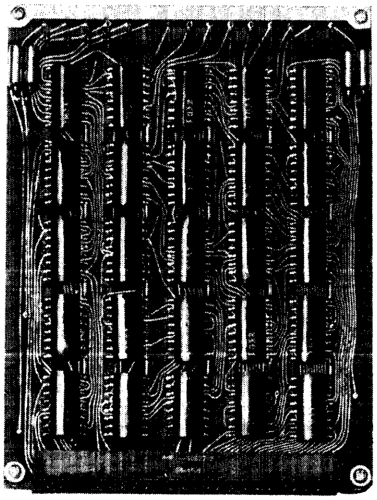
The following five chapters cover logic and the four sections of a digital computer. You will find that a well-grounded knowledge of Boolean Algebra will greatly enhance the understanding of those chapters.

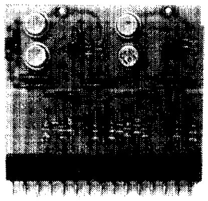
chapter VI

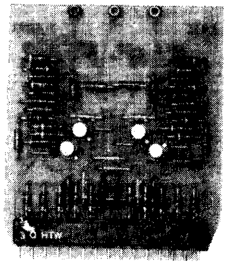
Introduction to Logic Circuitry

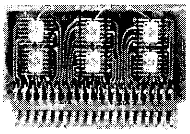












CHAPTER VI

INTRODUCTION TO LOGIC CIRCUITRY

INTRODUCTION

Logic may be defined as "a Science that deals with the formal principles of reasoning". You have often heard mentioned "The decision was logical" or "the logical thing to do....", meaning that action or thoughts were governed by certain reasoning processes.

Computer operations can be controlled by certain events or results. Therefore, the computer must be able to make decisions. A prevalent, but false,opinion exists that electronic computers can "think" or "reason". However, computers can make logical decisions based on available facts. The yes or no answer, never maybe, is compatable to the binary system you learned back in Chapter III.

The electronic circuitry of a computer capable of making the logical decisions is understandably referred to as logic circuitry. Because nearly all of the circuitry is in some way involved with the final result, the major portion of any electronic computer is comprised of logic circuitry.

The objectives of this chapter are to:

1) Teach you the logic symbol and associated function of each of the basic logic circuits.

2) Acquaint you with the actual circuit operation of several families of computer logic.

3) Teach you how to interconnect logic circuits to form AND gates and OR gates.

4) Apply the Boolean Algebra learned in the preceding chapter to logic circuits and to analyze input and output equations.

Whenever an electronic digital computer is designed, the responsibility is often divided into three areas. One group is responsible for the overall system design, another for the logical design, and the third group would be responsible for the actual electronic circuit design.

Figure 6-1 illustrates these three groups and their interrelationships.

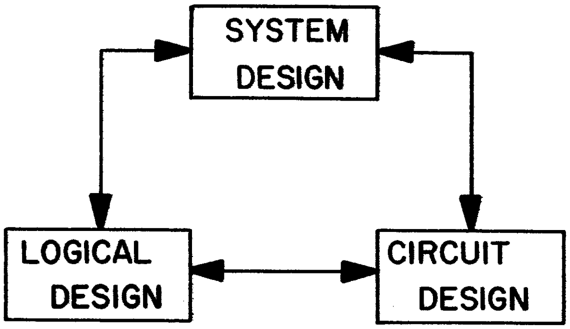


Figure 6-1

COMPUTER DESIGN

SYSTEM DESIGN

In the early design stages of a specific computer, the system design group makes a detailed study of competitive machines. There are two factors, speed and power, which are closely related in the final computer. Power describes the complexity of the instruction set and the general capability of the machine to work with large numbers. If the machine is powerful but slow, it is able to work with complex problems but is not able to complete the job as quickly as its competitors. On the other hand, if the computer is fast but not powerful, a limit in programming capability keeps the proposed design at a disadvantage. The balance between these characteristics is illustrated by the formula $S/P = C$, where S is speed, P is power, and C is relative capability. All of these variables are relative to a given competitive computer.

As an example, suppose the system design group is to specify a machine that is to be competitive with the PH-1. This computer is built by an imaginary company referred to as the Phantasm Computer Corporation. The PH-1 has a speed that is assigned relative index 1. The power of the proposed new design is 2. This means that the new computer requires twice as many instructions to complete a given problem. The speed of the proposed design is four times that of the Ph-1.

$$\frac{S}{P} = C$$

$$\frac{4}{2} = 2$$

The conclusion is that the new design should be able to process a given problem twice as fast as the PH-1. The system design group is responsible for coordinating the two variables of the formula.

Speed and power may vary as long as the final result is consistent with the original specification. Before final specifications are established, a third variable is added to the basic formula. This variable represents the comparative price of the machines and is defined as E. To illustrate the operation of this variable, assume the machine being designed costs ten per cent more than the PH-1. (E = 1.1)

$$\frac{S}{P.E} = C$$

Substituting the relative values:

$$\frac{4}{(1.1) \quad (2)} = 1.8$$

The formula states that the proposed computer is 10% more expensive than the PH-1, four times faster and requires twice as many instructions to solve a typical problem. The system design group can now assign project areas to the logical design group and the electronic (circuit) design group.

LOGICAL DESIGN

The logical design group is responsible for the power of the computer. The design of larger arithmetic sections and more comprehensive instructions are usually its objective. A close relationship exists between the logical design group and the circuit design group. Sometimes the only way to accomplish a given function is with a specially designed circuit. Before a final design is accomplished, many adjustments may be made in the price, speed and power specifications.

ELECTRONIC DESIGN

The circuit design group is concerned with physical as well as electronic aspects of computer design. It is desirable to achieve a final design that is economical, easy to repair, compact and suitable to the logical application. The purpose of the remainder of this chapter is to discuss the suitability of circuits for logical application.

In the early stages of design, general characteristics are outlined. Two important factors are speed of memory and added propogation time. Of all the basic units of the computer, these two are most often used and, therefore, form the most critical basic elements. As an example, if a multiply instruction is performed, the adder is used repeatedly until the answer is completed. A computer also averages more than one memory cycle for each instruction that is completed. Chapter VIII discusses memory design considerations in detail.

The add circuit provides a good basis for the design of a computer system. It has many active elements with a good many inputs and outputs. If a working arithmetic unit can be designed, the remainder of the computer can very likely be built.

CHOICE OF THE PROPER CIRCUIT

The choice of different circuit configurations is often limited by design criteria. Overall system design dictates how fast the circuits must react or switch and, consequently, also restricts the freedom of the design engineer. Faster circuits require better components and more sophisticated techniques.

One factor determining the speed of a circuit is the voltage range over which it must conduct; another is whether the transistors are operating at saturation and cutoff or somewhere in between. A saturated transistor is one that is conducting as much as possible and is being forced in that condition. More time is required to remove the forced condition and switch the transistor to cutoff than if it is conducting <u>near</u> saturation. A correlation may be made between a saturated transistor and a base runner caught "leaning the wrong way". It requires time for the base runner to "lean the right way" before he actually starts to move.

Figure 6-2 illustrates an emitter amplifier that would be classified as saturated logic.

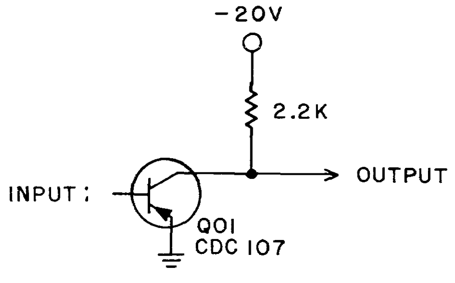


Figure 6-2. Emitter Amplifier (Saturated Logic)

Saturated Logic

Remember that the system design group has specified a certain add time. The circuit of Figure 6-2 requires a finite amount of time to change from a "1" output to a "0" output. If saturated logic is used, the transister will be cut off in the "1" state and saturated in the "0" state. The advantage of this operation is that the circuit is very simple. A disadvantage is that the circuit is slowed by this technique. As an example, suppose the transistor is biased in the off condition by a reverse bias of +.2 volts and in the on condition by a bias beyond saturation (Figure 6-3).

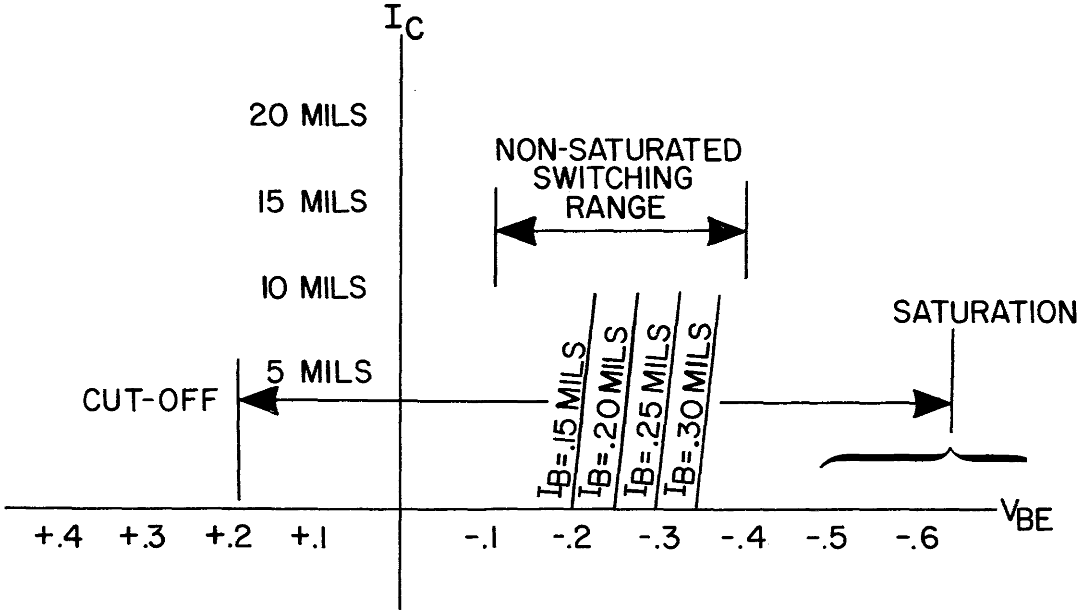


Figure 6-3. Biased Transistor.

Nonsaturated Logic

If a method can be used to control the transistor more accurately, the circuit switching time will be considerably reduced. Suppose a voltage feedback were added to the original circuit in the manner shown in figure 6-4.

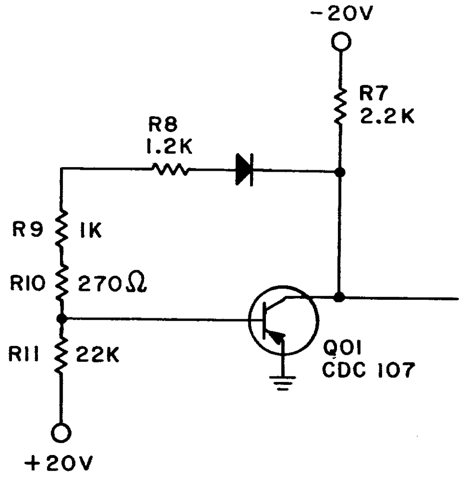


Figure 6-4. Addition of Voltage Feedback.

Observe the values of the resistors. From the base to -20 volts there are 270 ohms, 1 K, 1.2 K, a forward biased diode, (for the discussion, a negligible resistance in the forward bias condition) and a 2.2 K resistor. This makes a total of 4.67 KΩ of resistance. From the base of Q01 to ground there are 22 KΩ of resistance. The voltage divider establishes a negative potential on the base of Q01 which forward biases the transistor into conduction. Since a small amount of base voltage can cause a large amount of base current and collector currect, assume the base to be at approximately 200 millivolts. Using ohms law to determine the current through R11, we have:

$$I = \frac{E}{R} = \frac{20.2}{22 \times 10^{3}} = .918 \times 10^{-3}$$

It can be assumed that approximately .9 ma. of current flows through R11. The current flow available through R11 forms a basis for estimating the current in Q01.

If R7 and Q01 operate independent of the feedback network, the current and voltage acting on R7 follows the characteristic shown in figure 6-5.

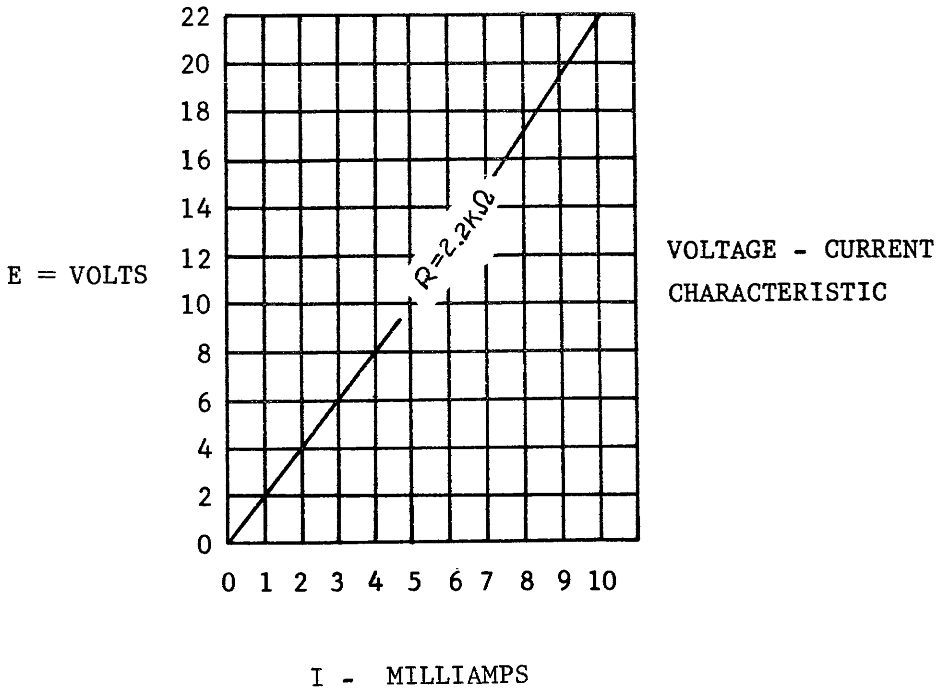


Figure 6-5. Voltage - Current Characteristics

The current through R7 varies from a maximum of 9.18 ma., at 20 volts, to a minimum of 0 ma., at 0 volts.

Since the base current under these conditions from 0 to .303 milliamps, the sum of the currents of R11 and the base current varies from .918 to 1.2 milliamps. An analysis of the 2.4KΩ , represented by R10, shows that Q01 Collector voltage varies from 3 to 4 volts.

Before the discussion is continued, recall the following points. The inverter circuit is a single inverter in a common-emitter configuration. The current in the feedback network is controlled mainly by the +20 volt supply acting through R11. In the normal course of operation, the natural output at the collector of Q01 is approximately 3 volts, with a tendency to increase to four volts. The series string of resistors is acting like a voltage divider with a slight negative at the base of Q01, and 3 volts at the collector of Q01.

Returning to the discussion of collector current, notice that there are approximately 17 volts across R7, which cause a current of 7.7 milliamps (Figure 6-5). Since slightly more than one milliamp flows in R8, R9, and R10, one collector current of Q01 in about 6.7 ma., with a base current of .22 ma. The effect of this action is felt if an attempt is made to change the collector voltage. Suppose the collector voltage begins to rise. The current through R8, R9, and R10 tends to increase. The transistor tends to conduct more, adding to the current through R7.

The feedback is basically a voltage control feedback.

FAMILIES OF LOGIC CIRCUITS

Four families of logic circuits are employed in the operation of Control Data computers. They are,chronologically, the 1604 family, the 3000 family, the 6000 family, and intergrated chip logic. Because the basic building block of a CDC computer is the inverter circuit, further analysis of logic circuits will "lean" toward the basic inverter.

1604 FAMILY OF CIRCUITS

The first CDC computer, the 1604, and several others of comparable speed were designed and manufactured using 1604 logic. Operation is synchronized by a 2.5 mc clock oscillator providing pulse widths of .2 microseconds. The switching of time of an average inverter is 30-50 nanoseconds, depending somewhat upon circuit application.

Physical Characteristics

All 1604 logic circuits are printed on a fiberglass board 2 and 7/16 inches wide by 2 and 1/8 inches high. A fifteen pin jack is mounted along the bottom to accomodate all connections to the circuit, including power. Figure 6-6 illustrates a typical inverter card with power connections to pin 13, 14, and 15 on the left. The remaining 12 pins are for logic inputs, outputs or unused.

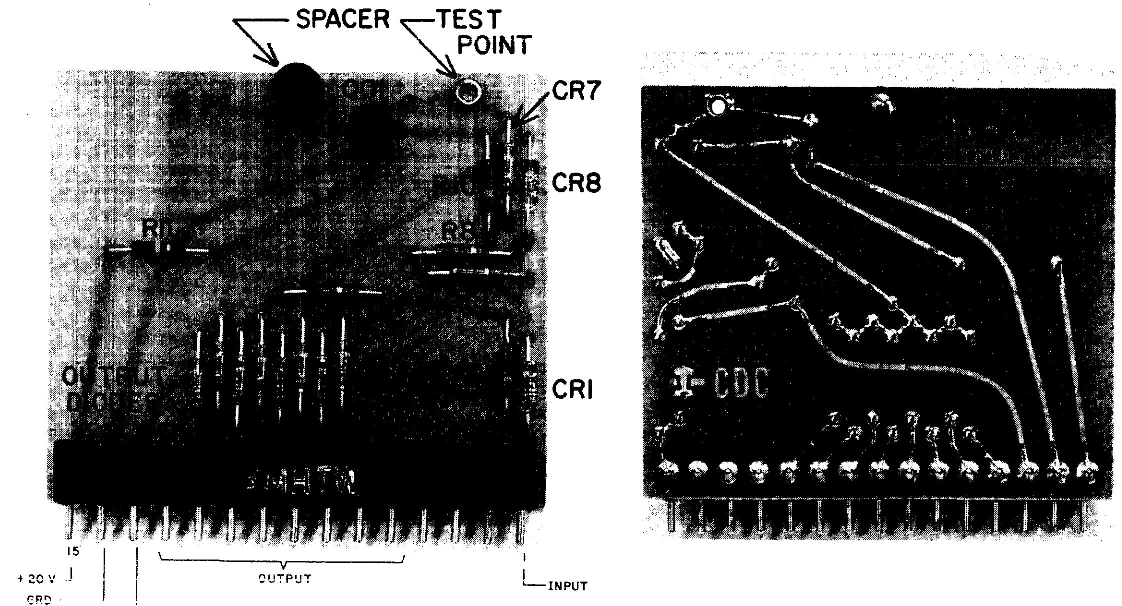


Figure 6-6. 1604 Circuit Card

Circuit Operation

Perhaps you recognize that portion of Figure 6-7 with heavy lines as the circuit explained in the nonsaturable logic section of this chapter. The remaining components in the circuit are to provide for inputs and outputs. The input circuit consists of R1 and CR1.

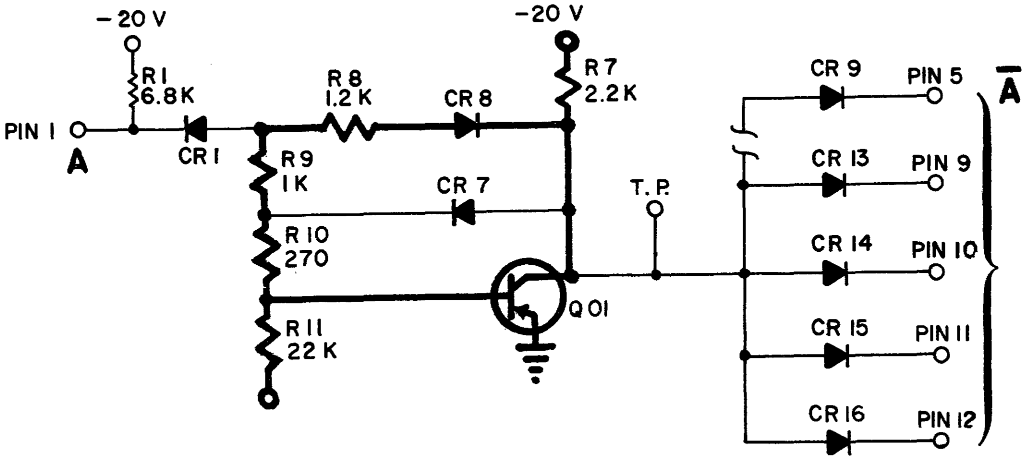


Figure 6-7. The basic Inverter (11A Card) Circuit

Because the card has a single input and performs the NOT function, the boolean output equation would be expressed as the input equation but with the bar over it.

When -20V is connected to R1, the first effect is the voltage across R1, which has an instantaneous value of 18.5 volts. This tends to develop approximately 2.72 milliamps, which is felt through R9, R10, and R11. The circuit stabilizes with 2.5 milliamps flowing through R1, CR1, and R9. At the junction of R9, the current flow divides with approximately 1.5 milliamps through CR7 and the remainder (≈ 1 milliamp) through R10 and R11. The voltage developed at the output in this case is about .5 volts. The voltage at the junction between CR1 and R9 is now approximately -3 volts.

CR7 allows a feedback from Q01 for a 0.5 volt output. If the collector changes in a positive direction, the positive is felt at the base of Q01 which tends to turn off the transistor. This forces the voltage to return to the original 0.5 volt level.

The 8 diodes on the output of the inverter are the maximum number allowed. The circuit as shown in Figure 6-7 is a standard 11A card.

The established voltage potentials for 1604 logic circuits are -3vdc for a logical "1" and -0.5 for a logical zero. Future references to 1604 logic signals will assume the same potentials unless otherwise stated. Keep in mind that the inverter produces a 180° electrical phase shift. A logical "1" input results in a logical "0" output and vice versa.

Logical Input

The resistor-diode combination just discussed is a logical "1" source. In order to obtain control, an inverter output diode is connected to the junction between R1 and CR1.

Figure 6-8 illustrates the connection of two single inverters and the logical symbol that represents the circuit. Since thousands of inverters are used in most Control Data computers, an alphanumeric designator names each inverter. This in the form of a letter prefix and a three-digit suffix.

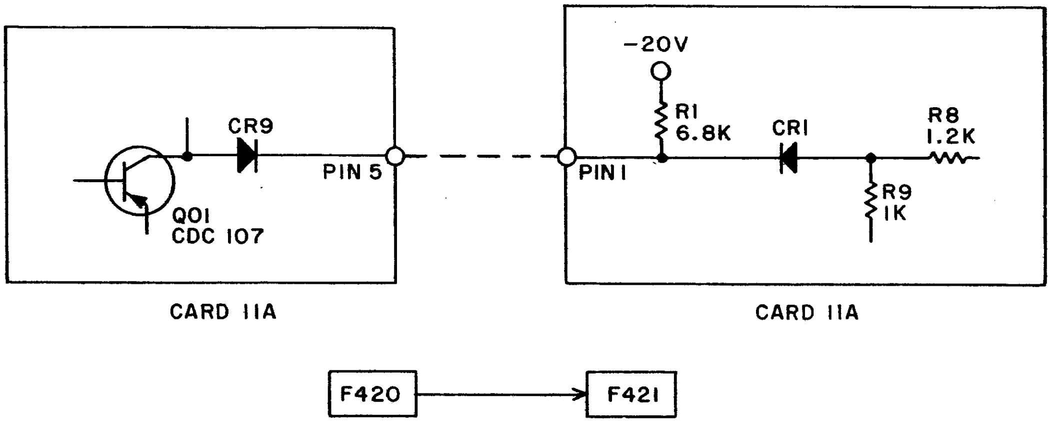


Figure 6-8. Connection of Two Single Inverters

And Circuits

When an AND gate is desired, more inverters are connected to the same input (figure 6-9). If F360, F420, or F422 have a zero output, a path is completed for current flow from -20 volts, through R1, CR9, and Q01 to ground. Since .5 volts is felt at pin 1 of F421, CR1 is reversed biased. F421 assumes a zero input and a one output. The only way to achieve a forward bias on CR1 if F421 is to have -3 volts out of F360, F420 and F422. Since R1 forms a current path for the AND diodes (CR9) it is commonly referred to as an AND load.

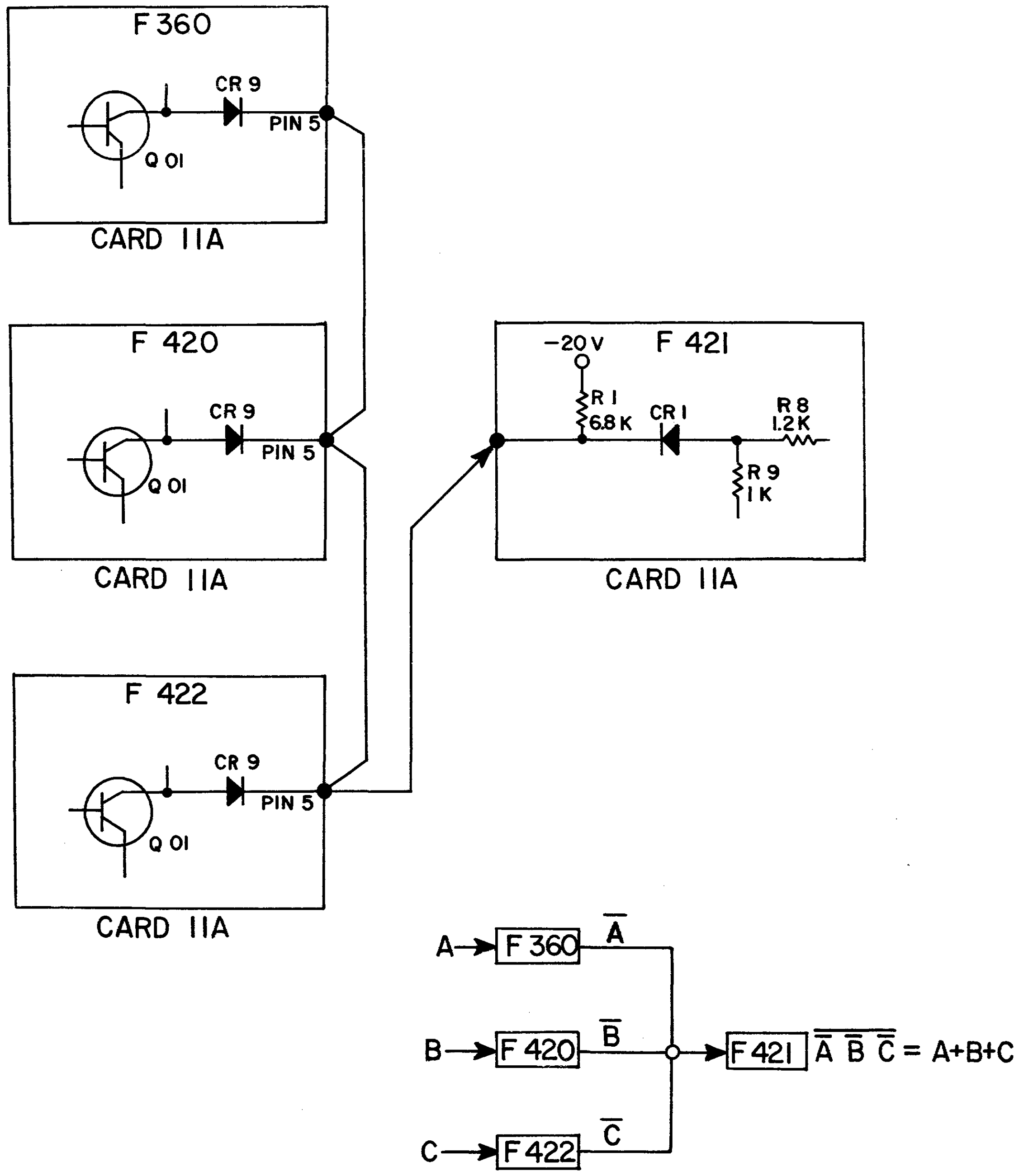


Figure 6-9. AND Gate Connections.

OR Circuits

When an OR circuit is desired, the basic inverter is used, with the addition of more input circuits. Each additional input circuit is identical to R1 and CR1. The maximum number of input circuits allowed on an inverter is 6. When a two-input card is produced, it is designated Card Type 12. For three inputs, 13; four inputs, 14; and so on to six inputs, 16.

On figure 6-10, the 11A card that represents F421 has been replaced by a 12A card.

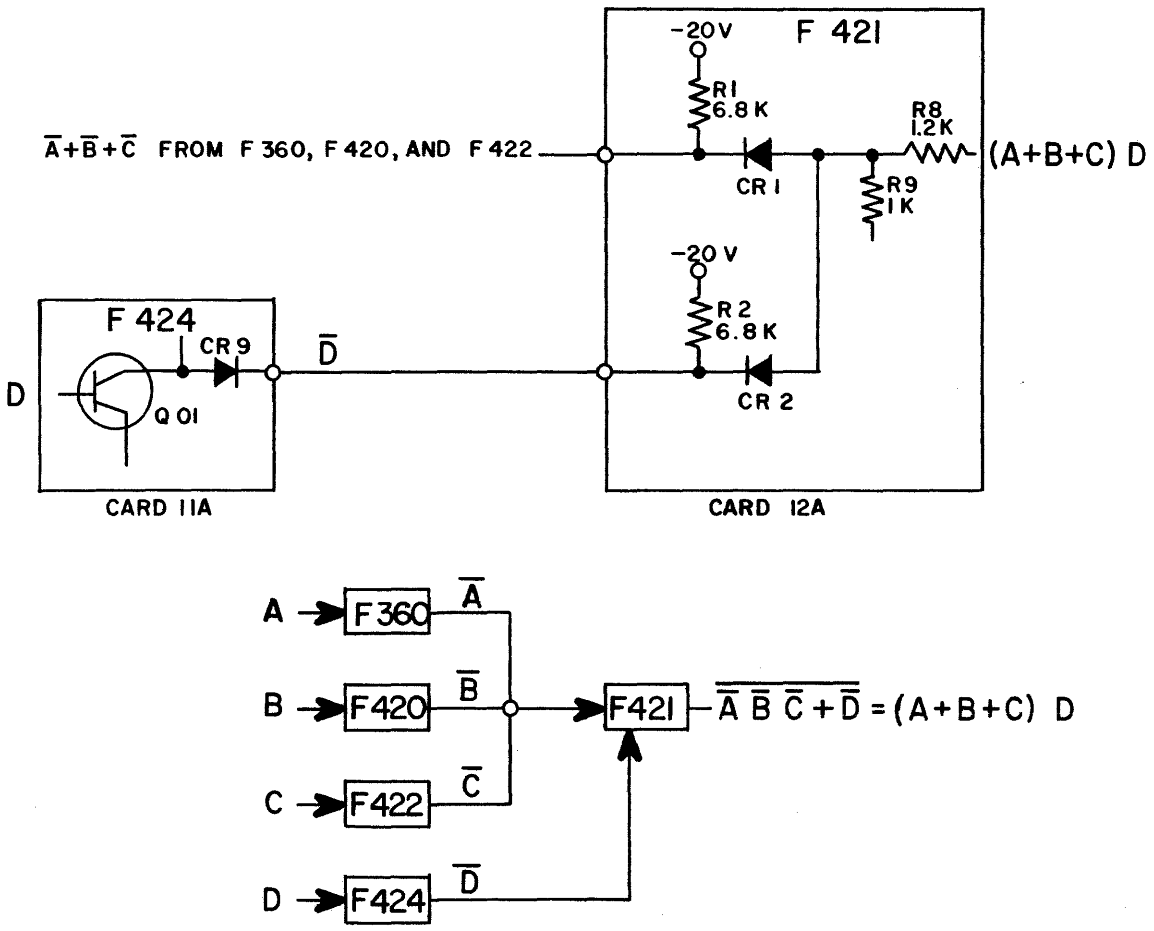


Figure 6-10. An OR Circuit

You have undoubtedly detected the system utilized to number the cards (an 11 card has one input, a 12 card has two inputs, etc.). Figure 6-11 illustrates the correlation between card types inputs (I) and outputs (O). The subscript distinguishes the circuits on a given card.

Type	Title	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	Inverter	O				O	O	O	O	O	O	O	O			
12	Inverter	I	I			O	O	O	O	O	O	O	O			
13	Inverter	I	I	I		O	O	O	O	O	O	O	O			
14	Inverter	I	I	I	I	O	O	O	O	O	O	O	O			
15	Inverter	I	I	I	I	I	O	O	O	O	O	O	O			
16	Inverter	I	I	I	I	I	I	O	O	O	O	O	O			
20	Quadruple Inverter	I_A	O_A	O_A	I_B	O_B	O_B	I_C	O_C	O_C	I_D	O_D	O_D			
21	Double Inverter	I_A	O_A	O_A	O_A	O_A	O_A	I_B	O_B	O_B	O_B	O_B	O_B			
22	Double Inverter	I_A	I_A	O_A	O_A	O_A	O_A	I_B	I_B	O_B	O_B	O_B	O_B			
23	Double Inverter	I_A	I_A	I_A	O_A	O_A	O_A	I_B	I_B	I_B	O_B	O_B	O_B			
24	Double Inverter	I_A	I_A	I_A	I_A	O_A	O_A	I_B	I_B	I_B	I_B	O_B	O_B			
28	Triple Inverter	I_A	I_A	O_A	O_A	I_B	I_B	O_B	O_B	I_C	I_C	O_C	O_C			
29	Triple Inverter	I_A	O_A	O_A	O_A	I_B	O_B	O_B	O_B	I_C	O_C	O_C	O_C			

Figure 6-11

There are some deviations from the standard numbering system illustrated by the type 20, 28, and 29 cards in the preceding figure. However, all card types beginning with a lX or 2X designation are inverters; lX usually indicating single invertors, 2X usually indicating double inverters.

Control Data Computers using 1604 logic circuits are the 160A , the 924A, and the 1604A, B, and C. Some peripheral equipments utilize 1604 logic even though installed with a system of different logic.

Ground Rules

The following ground rules for usage of the basic 1604 inverter circuits are intended as guidelines in obtaining optimum performance. They are not intended to be excessively restrictive since it is often found that a circuit will operate satisfactorily in a configuration which may deviate somewhat from one or more of the ground rules. Decisions as to when a ground rule may be violated must be based upon various electronic and timing considerations and are the responsibility of the designer.

1) A maximum of eight outputs may be taken from a single inverter.

2) An Inverter will drive a maximum of six simultaneously-gated AND loads.

3) The total number of inputs and outputs of a single inverter must not exceed twelve.

4) The number of OR inputs to an inverter is limited to a maximum of six.

5) For high speed operation, the number of AND connections which can be made to a single OR input should be limited to four. If timing is not critical, the maximum number of AND connections can be increased to six.

6) All **unused input** pins must be grounded.

7) The minimum switching time for a mesa transistor inverter driving one load is approximately 30 nanoseconds. This will increase to about 75 nanoseconds as additional loads are added.

8) The minimum switching time for a drift transistor inverter driving one load is approximately 50 nanoseconds. This will increase to about 100 nanoseconds as additional loads are added.

The advantages of the 1604 type logic circuit are that it is economical because of the few parts necessary to build a basic inverter, easy to repair because all components are in plain view, compact and suitable to the logical application.

There are a few limitations to this circuit. There are only twelve pins available for logical connections. If two inverters share the same card, there are only six pins for each inverter. The nominal limits for inverter switch time is 30 to 75 nanoseconds.

3600 FAMILY OF CIRCUIT

Continued development in our technical world necessitates technological advances in the computer industry. If successive generations of computers are to become more powerful, they must be able to perform at much greater speeds than ever before.

One way to increase the speed of a computer is to increase the master clock frequency. This apparent solution sounds relatively simple. Let's increase the master clock frequency to 8.0 mc from the 2.5 mc frequency utilized with 1604 1 ogic. The pulse duration is now .0625 ns (62.5 nanoseconds) instead of the .2 ns previously used; but the same computer should operate more than three times faster than before. However, our rose-colored glasses are defective. A 1604 inverter circuit required 30-50 (sometimes more) nanoseconds to switch. That is nearly the entire pulse duration (phase time) at the 8.0 mc rate.

It's become obvious that something must be done. The revised apparent solution is to design new logic that will switch faster. So... back to the drawing board.

Design of the 3600 family of circuits provided two definite advantages for computer design:

(1) greater speed

(2) increased availability of circuit connections.

In the 1604 circuit, when a combination of inputs and outputs are considered, the maximum connections are 12, as a single output pin and single diode are required for each connection. If another method of connection were available, all circuits would be capable of maximum outputs without regard to input. Greater basic speed is necessary because the 3600 family of computers is designed to meet the needs of users who have a large volume of work load.

Circuit Speed

The first and most obvious modification of a circuit in order to increase its speed is the use of a faster transistor. However, as a result of this increase, circuit reactances have a greater effect. The circuit shown in figure 6-12 is basically similar to a 1604 type basic inverter.

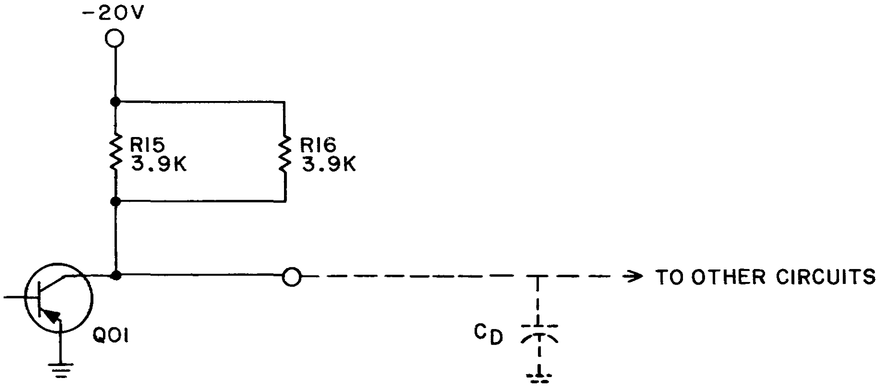


Figure 6-12. 3600 Circuit

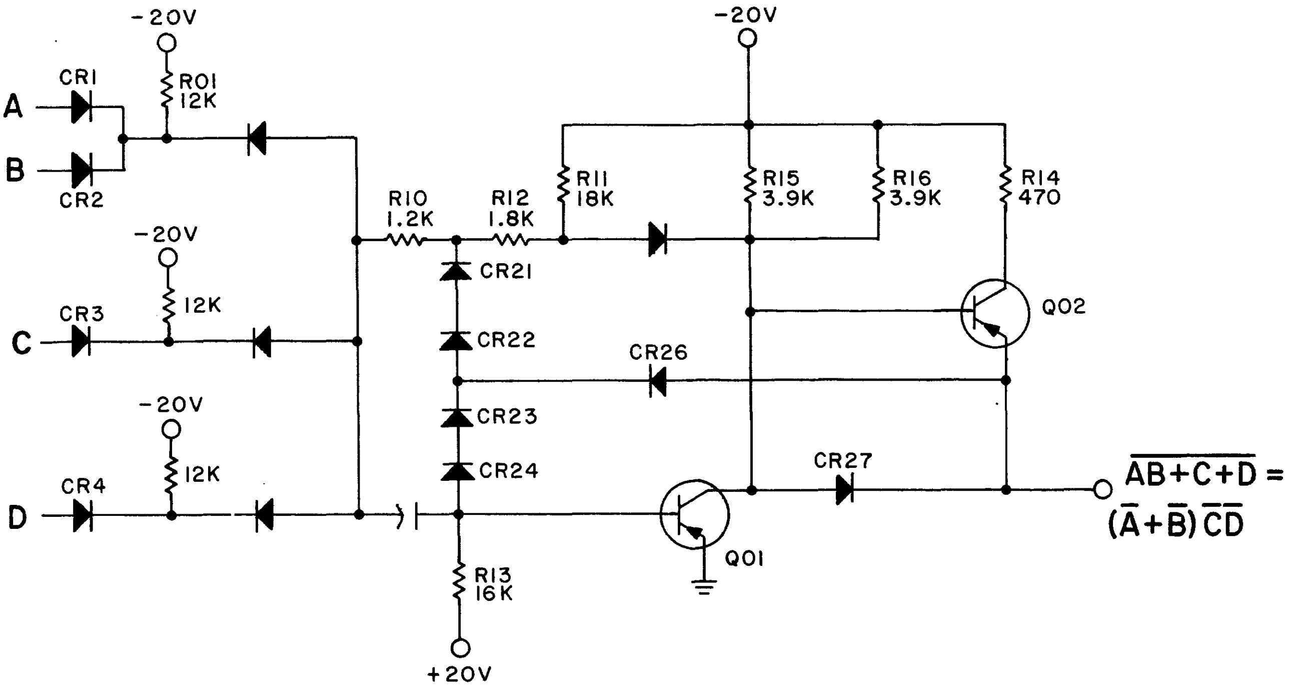


Figure 6-13. The 3600 Inverter Circuit

Q01, in this circuit, is equivalent to Q01 in the 1604 circuit (Figure 6-7). R15 and $16 are equivalent to the collector load R7 on the 1604 circuit. A close examination of CR27 and Q02 and Q02 shows that circuit operation is essentially the same as for the 1604 inverter when Q01 is in a heavy conduction state. The collector of Q01 is minimum, and the AND load conducts through CR27. Because of a small voltage drop across CR27, Q02 is cut off. Assume now that Q01 is beginning to cut off. If the distributive capacitance of circuits connected to this output is sufficient, the voltage at the output pin tends to remain the same. Under these conditions, CR27 cuts off because of backward bias and Q02 turns on. Since Q02 is capable of conducting very heavily, the distributive capacitance assumes a more negative voltage in a very short time. Notice that CR27 is not a logical element, but is a switch when going from one logical level to another.

Logic Diodes

In order to use all output capability from an inverter, the AND diodes are moved to the input of the circuit. As an example, Figure 6-13 is shown as a three-way OR circuit which is being fed by at least a one-way AND gate. The reason for the one-way AND is to maintain isolation. By having a variety of input configurations, a great many logical operations can be performed. The maximum number of connections in a 3600 inverter is eleven inputs and eight outputs.

Voltage Feedback Network

The resistors that were used in the 1604 type circuit have been, in part, replaced by some special silicon diodes. Forward voltage drop is on the order of .67 volts if a 2 milliamp current level is maintained. At this level of current, signal input has very little effect on voltage drop. This means that a load variation can be felt much more quickly at the base of Q01, and a better regulation of output is accomplished.

Logical One Output

Suppose Q01 is in a low level of conduction, (all inputs are close to ground). The base of Q01 is slightly negative with respect to ground. Since there is 20 volts across R13, approximately 1.25 milliamps is flowing. Assuming a few tenths of a milliamp base current, the voltage across each diode (CR21, 22, 23,24) should be slightly mo re than .6 volts. This indicates a voltage at the junction between CR21 and R12 of at least 2.5 volts, with a tendency toward 3 volts. Continuing through R12, this resistor should drop at least 2.5 volts. Therefore, the output of the inverter should be at least 5 volts. In practice, this output level is approximately 5.8 volts and is defined as a logical one.

Logical Zero Output

Assume one of the inputs goes in a negative direction. This causes a forward bias across the OR diode and this gate element adds to the current through the four series diodes. Since more current is available through the base of Q01, it goes into a heavy conduction state. CR27 turns on and CR 26 becomes forward biased when the output voltage reaches approximately 1.1 volts. The feedback through CR26 regulates the output at this level which is defined as a logical zero.

Logical Connections

Since only one output is needed in the 3600 inverter circuit, it is convenient to group two inverters on a single card when only a few inputs are used. Suppose the logical circuit of figure 6-8 were duplicated with 3600 circuits (figure 6-14).

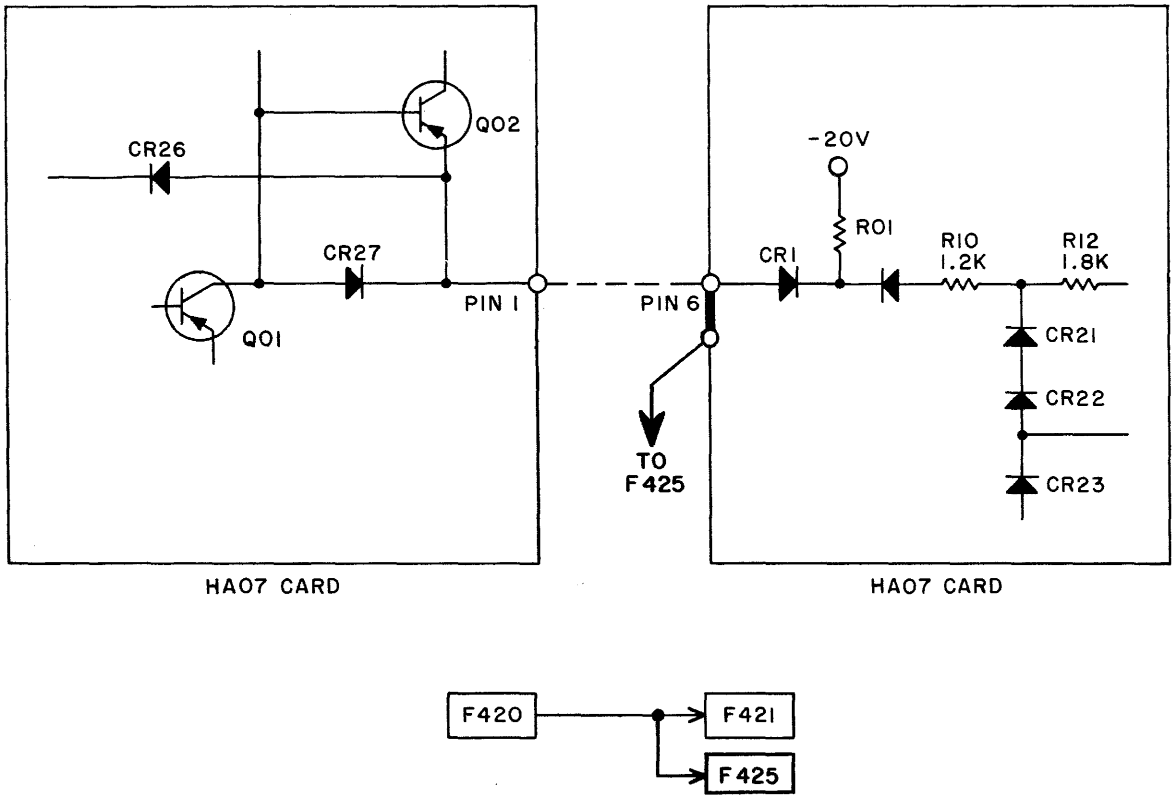


Figure 6-14. 3600 Circuits

If F420 drives more than one inverter, the outputs are taken from pin one of F420 or pin six of F421.

Figure 6-15 duplicates the logic represented by Figure 6-10. In the present case, 3600 circuits are used. Notice in particular the different method of connecting an AND gate.

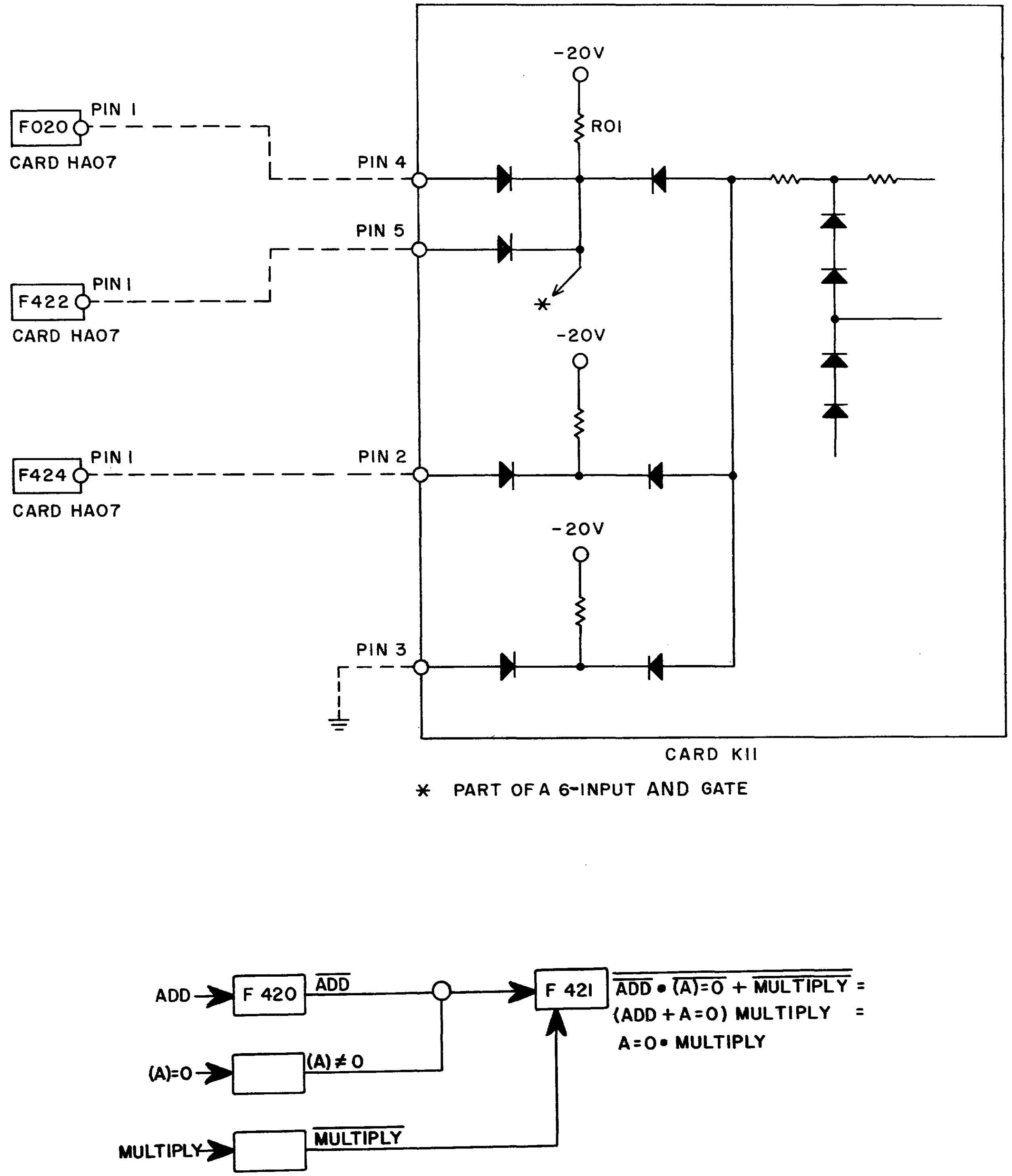
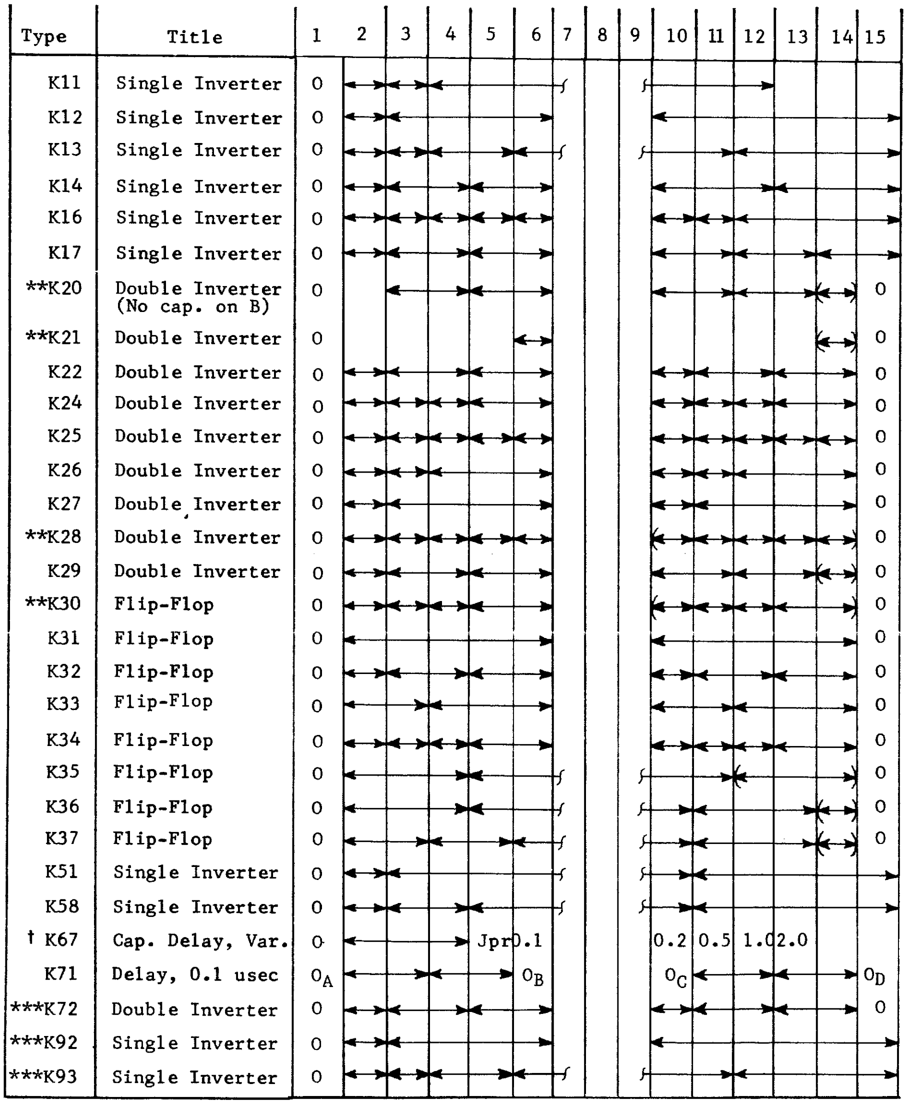


Figure 6-15. 3600 Logical Connections.

The K11 card is designed to accept up to three ORed inputs. One of the input legs is a six-input AND gate. In this illustration, it is assumed that the other four inputs are not needed and can be left open. Pin three has no inputs. If a ground had not been used in this case, pin three would force a constant "one" into the inverter and the input inverters would lose control. Depending on the number of connections, the switching time for this circuit is between 10 and 20 nanoseconds.

Figure 6-16 illustrates the input connections to some common 3600 logic cards. Notice the type K28 double inverter. The parenthesis encompassing pins 10-14 indicate the B half of the card. The type K21 card has a single input for each of the two inverters on the card.

Figure 6-16. 3600 Logic Card Input Connections



Type	Title	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
K11	Single Inverter	0														
K12	Single Inverter	0														
K13	Single Inverter	0														
K14	Single Inverter	0														
K16	Single Inverter	0														
K17	Single Inverter	0														
**K20	Double Inverter (No cap. on B)	0														0
**K21	Double Inverter	0														0
K22	Double Inverter	0														0
K24	Double Inverter	0														0
K25	Double Inverter	0														0
K26	Double Inverter	0														0
K27	Double Inverter	0														0
**K28	Double Inverter	0														0
K29	Double Inverter	0														0
**K30	Flip-Flop	0														0
K31	Flip-Flop	0														0
K32	Flip-Flop	0														0
K33	Flip-Flop	0														0
K34	Flip-Flop	0														0
K35	Flip-Flop	0														0
K36	Flip-Flop	0														0
K37	Flip-Flop	0														0
K51	Single Inverter	0														
K58	Single Inverter	0														
† K67	Cap. Delay, Var.	0				Jpr	0.1				0.2	0.5	1.0	2.0		
K71	Delay, 0.1 usec	0_A					0_B				0_C					0_D
***K72	Double Inverter	0														0
***K92	Single Inverter	0														
***K93	Single Inverter	0														

† Delays given in microseconds.

* Unless otherwise noted: pin 7 equals + 20v; pin 8 equals ground; pin 9 equals - 20v.

** No speed-up capacitor.

*** 10 pf speed up capacitor

Physical Characteristics

All 3600 Logic Circuits are printed on a fiberglass board 2 and 11/16 inches wide by 3 inches high with the 15-pin plug mounted on the bottom. Power pins are the three center pins 7, 8, and 9 instead of 13-15 that was characteristic of the 1604 circuit. Compare 6-6 and 6-17.

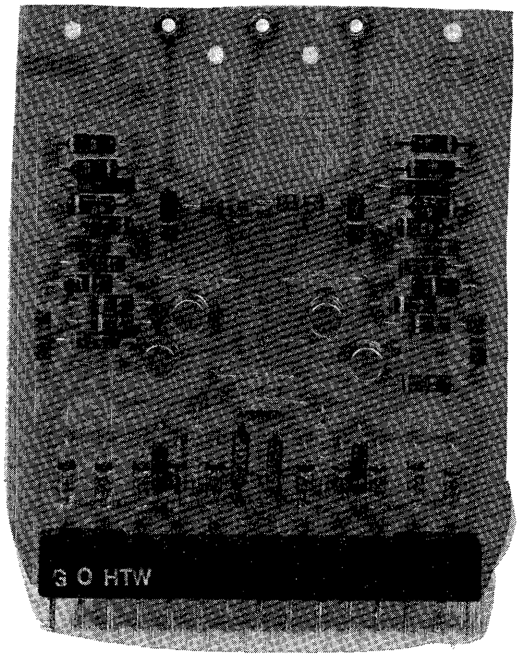


Figure 6-17. Physical Characteristics of 3600 Circuit (Double inverter)

The 3600 circuit board is physically larger than the 1604 circuit and the bottom corners are beveled (See figure 6-17).

Ground Rules (Effective November 21, 1963)

The following ground rules for usage of the basic 3600 inverter circuit are the result of tests performed by the Special Projects Department, Government Systems Division of Control Data Corporation. Inquiries concerning these ground rules should be addressed to the above department.

The ground rules are intended as guidelines in obtaining optimum circuit performance. They should not be considered as being excessively restrictive since it is often found that a circuit will operate satisfactorily in a less than optimum configuration which may deviate somewhat from one or more of the ground rules. Decisions as to when a ground rule may be violated must be based upon various electronic considerations, and are the responsibility of the designer.

Loading

Any single input to a recipient inverter, regardless of whether it is a part of a 1, 2, 3, 4, 5, or 6 input AND or an OR input, is considered to be one load.

1) An inverter may simultaneously drive eight AND loads, eight OR loads, or any combination up to eight loads total.

2) A flip-flop or a control delay may drive only seven loads, because it is required to provide its own feedback which constitutes one load.

Unused Inputs

3) In case an entire AND input group is unused, at least one of the inputs must be grounded.

4) All unused OR inputs must be grounded, if using type C or CA cards.

5) All unused single-way AND inputs must be grounded.

The 3600 logic circuit paved the way for the design of faster computers. Better quality components and sophistication of circuit design permitted an increase of master clock frequency to 8.0 mc.

Control Data Computers utilizing 3600 logic circuits are the 160G and all 3000 series computers except the 3500.

6600 FAMILY OF CIRCUITS

Although the 3600 logic circuits permitted computers to operate three times faster than those using 1604 logic, the increase was only temporarily sufficient. Industrial and Scientific demands for still faster computers became the "mother of invention" for the 6600 Family of Circuits.

What approach should be taken to design a new family of circuits that would have an appreciable speed advantage over the 3600 circuits? Adding more components? Faster clock ? Better transistors?

Each time another component is added to a circuit, inherent inductance and capacitance is also added which causes the circuit to soon reach the point of diminishing returns. Additional components do not justify the slight increase in circuit quality.

If adding components tends to slow the circuit, could circuit speed be improved by removing some of the passive components?

Why not use premium transistors, lower voltage levels, and again use the saturable logic explained at the beginning of this chapter? This approach was used for circuit design of the 6600 computer. The new concept allowed for greater packing density and a much faster circuit. Switching time was decreased to 5 nanoseconds as composed to 20 nanoseconds for 3600 logic and 50 nanoseconds for 1604 logic.

Physical Characteristics

The 6600 logic circuit consists of two fiberglass boards connected together on a 30-pin plug. The components are between the two boards connected in "cordwood" fashion (see lower left corner of Figure 6-18)

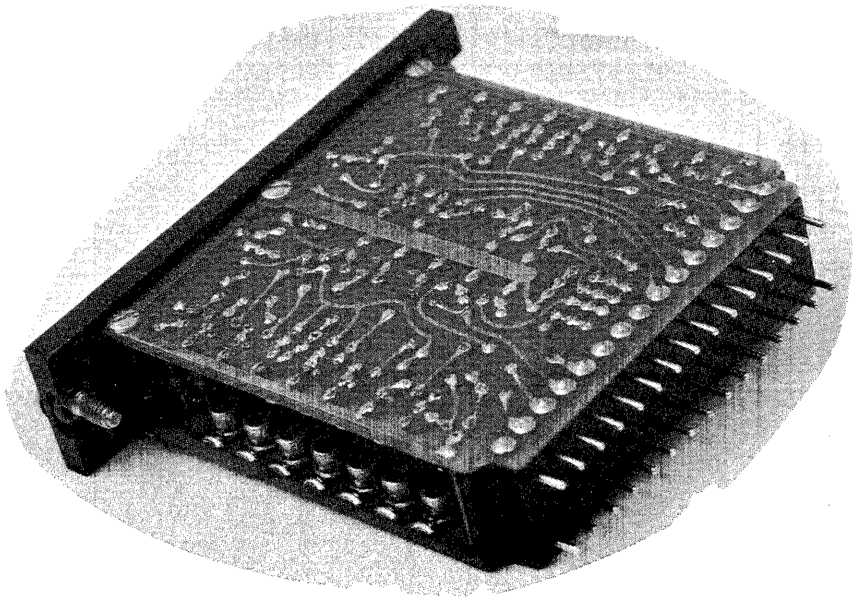


Figure 6-18. 6600 Module

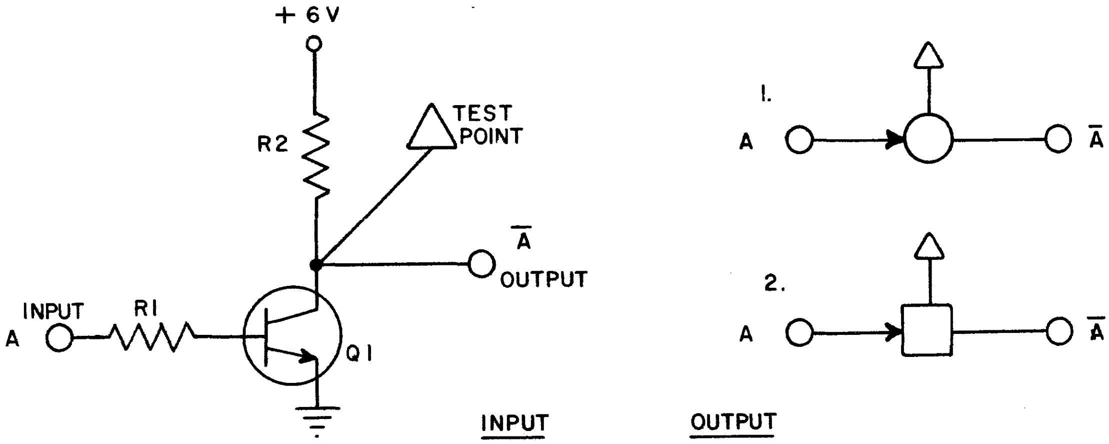
This new compact design prompted the change from "card" to "module". A single module may contain 64 transistors, each of which could form an inverter circuit. Interconnection between modules are twisted-pair with one of the conductors grounded at both ends. Coaxial test points on the exposed edge of the module provide oscilloscope viewing points.

Logical Operation

The basic circuit is a single NPN silicon transistor connected in a ground-emitter amplifier circuit (figure 6-19). The transistor operates from a +6 vdc supply. The inherent 180° phase shift of the transistor provides the basic inversion (or NOT) function. The two signal levels and their logical significance are as follows:

+0.2v	"1"
+1.2v	"0"

A +1.2v input turns on the transistor and drives it to saturation. The output signal is then the drop across the transistor, +0.2v. When a +0.2v input cuts off the transistor, the output rises toward +6v but is held to +1.2v by the voltage-dividing effect of the collector resistor and the input resistance of the driven load.



INPUT	OUTPUT
+0.2 V "1"	+ 1.2V "0"
+1.2V "0"	+0.2V "1"

Figure 6-19. Basic Inverter

The threshold voltage on the base of the transistor is approximately +0.8v (virtually no conduction below +0.7v) which permits the circuit to reject noise signals up to about +0.5v. Circuit parameters for cutoff and saturation are listed below.

	V_B	V_C	I_B (min)	I_C	Average Switching Time
Cutoff	0.2	1.2	0	0	5 nsec
Saturation	0.8	0.2	1 ma	10 ma	

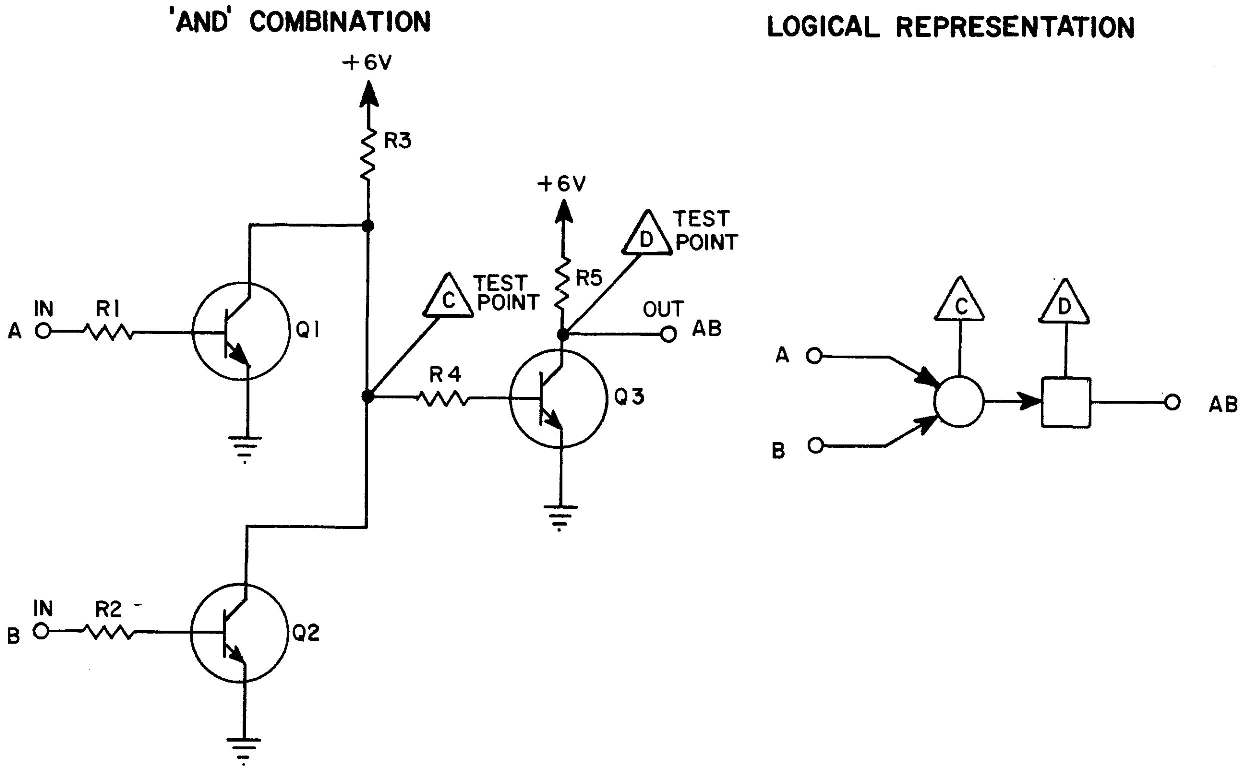
Input Combinations

Different portions of the computer require various combinations of incoming signals. The circuit configuration dictates whether all signals must be present, or only one of several is required, or whether all signals must be absent to achieve the desired result. These configurations are referred to as AND, OR, and NOR respectively.

The AND Circuit

The circuit to combine (AND) two signals A and B to provide an output is illustrated by figure 6-20. When either input A or B is a logical "0" (+1.2v), Q1 or Q2 is conducting at saturation and the potential at test Point C is +0.2v (the drop across the transistor). This, in turn, cuts off Q3 and the output becomes a logical "0" (+1.2v). When both A and B are a logical "1", both Q1 and Q2 are cut off and the potential at test Point C begins to rise toward +6v. When it reaches +.8V, Q3 conducts to saturation.

Base, or leakage, current from Q3 flows through R4 and R3 which form a voltage divider. The base-to-emitter drop of Q3 plus the drop across R4 establish the level at point C at +1.2v. With Q3 conducting, the output is +0.2v. Hence the output is +0.2v only when both inputs, A and B, are +0.2v.



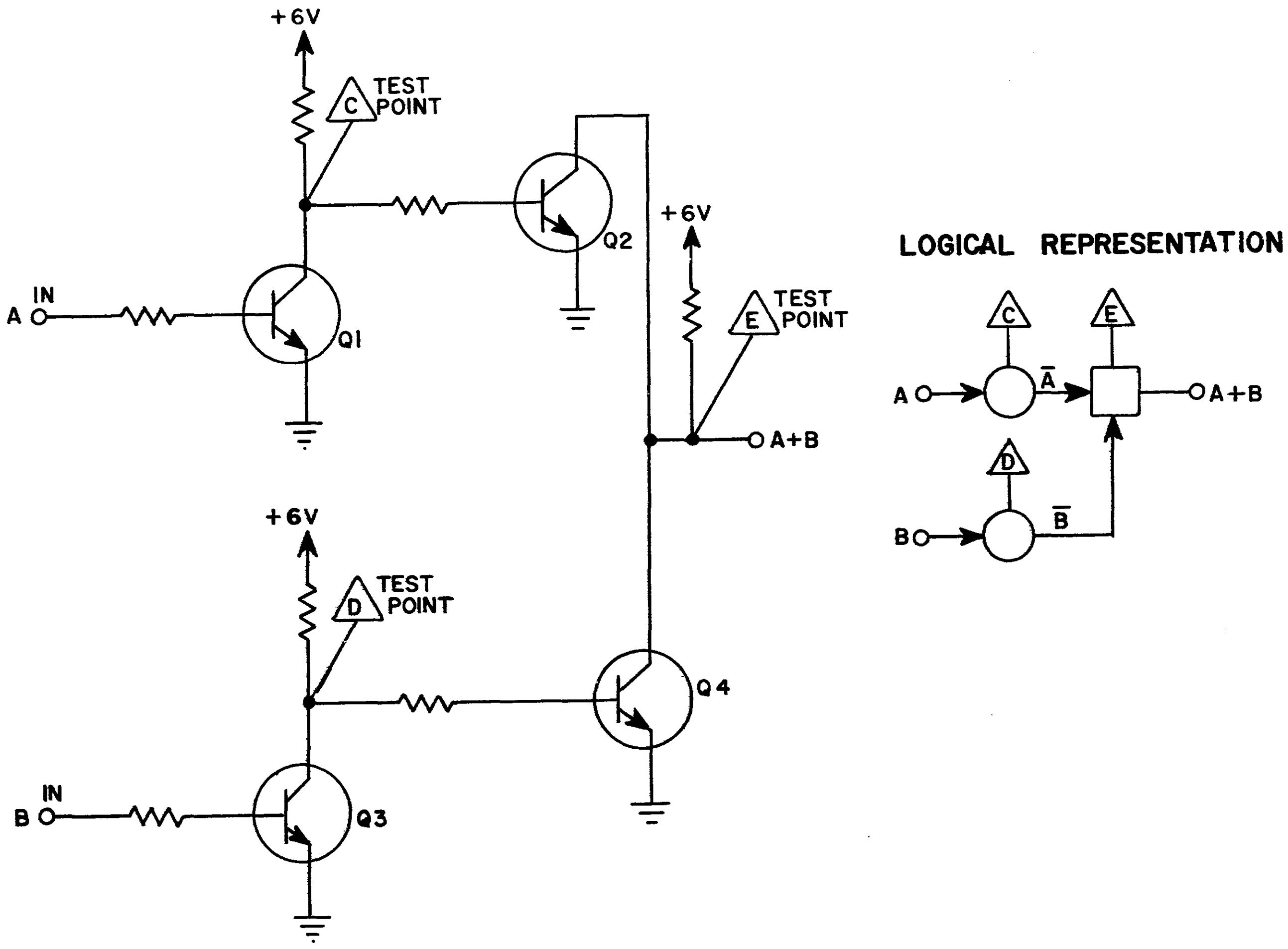
INPUT A	INPUT B	OUTPUT
+0.2V "1"	+0.2V "1"	+0.2V "1"
+0.2 V "1"	+1.2V "0"	+1.2V "0"
+1.2V "0"	+0.2V "1"	+1.2V "0"
+1.2V "0"	+1.2V "0"	+1.2 V "0"

Figure 6-20. AND Circuit

The OR Circuit

The circuit in Figure 6-21 shows the OR configuration for two signals, A <u>or</u> B. A logical "1" input at either A <u>or</u> B results in a logical "1" output.

'OR' COMBINATION



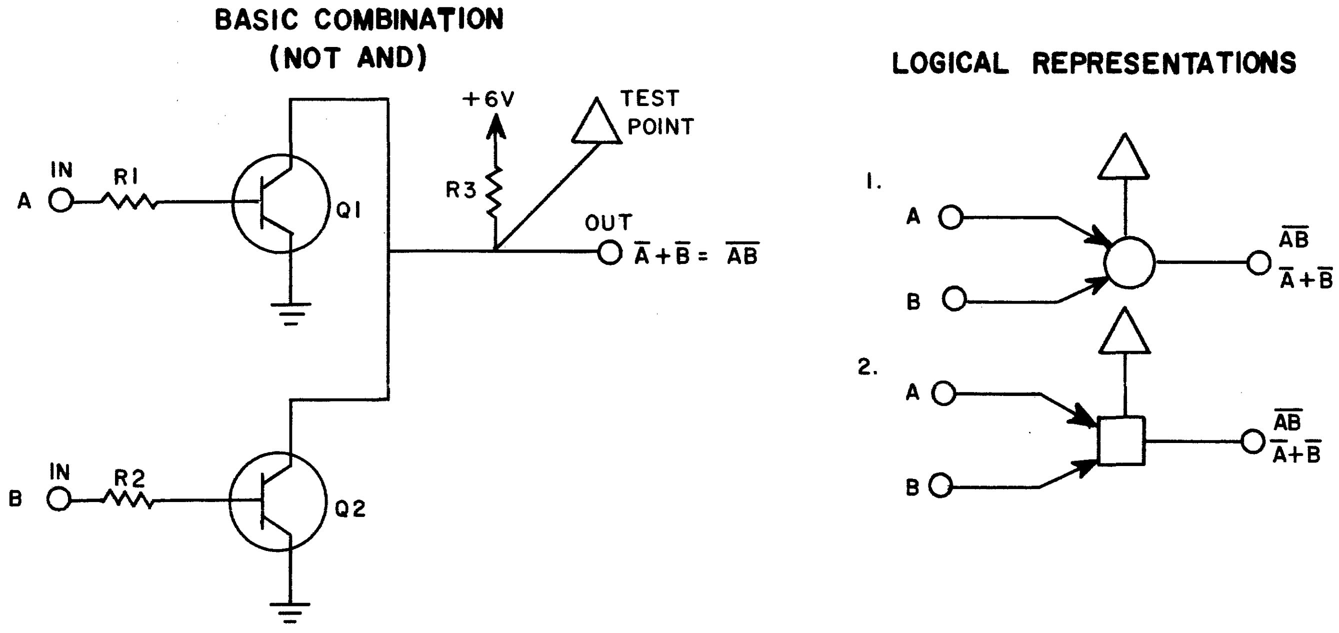
INPUT A	INPUT B	OUTPUT
+0.2 V "1"	+0.2 V "1"	+0.2 V "1"
+0.2 V "1"	+1.2 V "0"	+0.2 V "1"
+1.2 V "0"	+0.2 V "1"	+0.2 V "1"
+1.2 V "0"	+1.2 V "0"	+1.2 V "0"

Figure 6-21. OR circuit

A logical "1" at A causes Q1 to cut off. Its collector goes toward +6v and Q2 conducts to saturation. The output is the drop across the transistor (+0.2v) and constitutes a logical "1". The arrow represents the transistor and consequently the inverting component. Therefore, the inputs to the box Figure 6-21 (collector resistor) are the outputs of Q2 and Q4 and represents A or B. The collector resistor does not invert and the boolean output of the circuit is A+B as shown.

The NAND circuit

Both circuits previously discussed inverted the signals twice. Therefore, the overall circuit effectively had no inversion. The NAND circuit has a single transistor per input which means that the output will be the inversion of the input. This means that if both inputs are at +0.2v (logical 1), the output will be +1.2v (logical 0). Any other combination of inputs will result in a logical "1" out. Both transistors share a common collector resistor as in the AND circuit. Six transistors may feed the circuit and all would share the common collector resistor. Refer to Figure 6-22 and analyze the circuit.



INPUT A	INPUT B	OUTPUT
+0.2 V "1"	+0.2 V "1"	+1.2V "0"
+0.2V "1"	+1.2 V "0"	+0.2V "1"
+1.2V "0"	+0.2 V "1"	+0.2V "1"
+1.2V "0"	+1.2 V "0"	+0.2V "1"

Figure 6-22

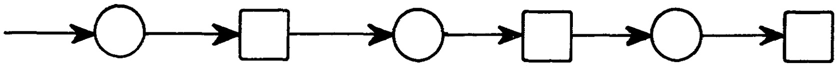
Logic Diagram Symbols

The symbols used to represent circuits on logic diagrams are as follows:

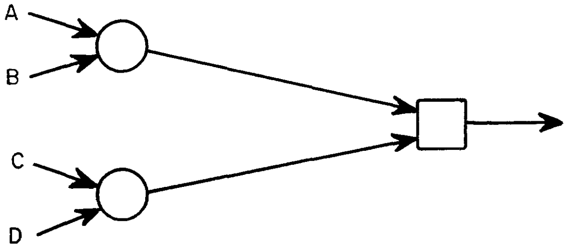
Symbol	Logical Function	Literal Significance
⟶	inversion	transistor (including its base resistor)
□	usually OR combination (one or more "O's")	collector load resistor
○	usually AND combination (all "1's")	collector load resistor

The use of two symbols, circle and square, to represent the combining of signals is a convenient way to indicate alternating logical levels. This alternation results from the inversion accompanying each successive combination. Logic is performed by strings or successions of steps: invert, combine, invert, combine, etc. Thus, every other combining step is the reverse of its predecessor.

A diagram of such a logical string of steps appears as:



Where possible, diagrams have circles and squares appearing in alternation and in such a way as to indicate whether the effect of a combination is an AND or an OR. Consider the logical circuit shown below:



What combinations of incoming signals would produce a logical "1" output? The next chapter will show you how to analyse a circuit and derive the logical output equation. If your answer to the output was AB or · CD, you are correct.

The development of 6600 logic opened new avenues for computer design. The 5 nanosecond switching time enabled the reduction of pulse durations to 25 nanoseconds. However, the twisted-pair interconnecting wiring has an inherent delay of 1.3 nanoseconds per foot. Wire lengths must be considered to insure that ANDed inputs arrive simultaneously.

You should now be able to make comparisons among the three families of conventional logic circuitry. Analyze the following table to validate your conclusions.

Family	Type	Switching Time (approximate)	Pulse Duration	Logic "1"	Levels "0"
1604	non-saturable	50 ns	200 ns	-3V	-0.5V
3600	non-saturable	20 ns	62.5 ns	-5.8V	-1.1V
6600	saturable	5 ns	25 ns	+0.2V	+1.2V

Table 6-1

Figure 6-23 illustrates the physical comparison of the three families of circuits. The cards/modules are shown inserted into the plugs which would be permanently attached to the computer main frame. Can you identify the different types from the photograph?

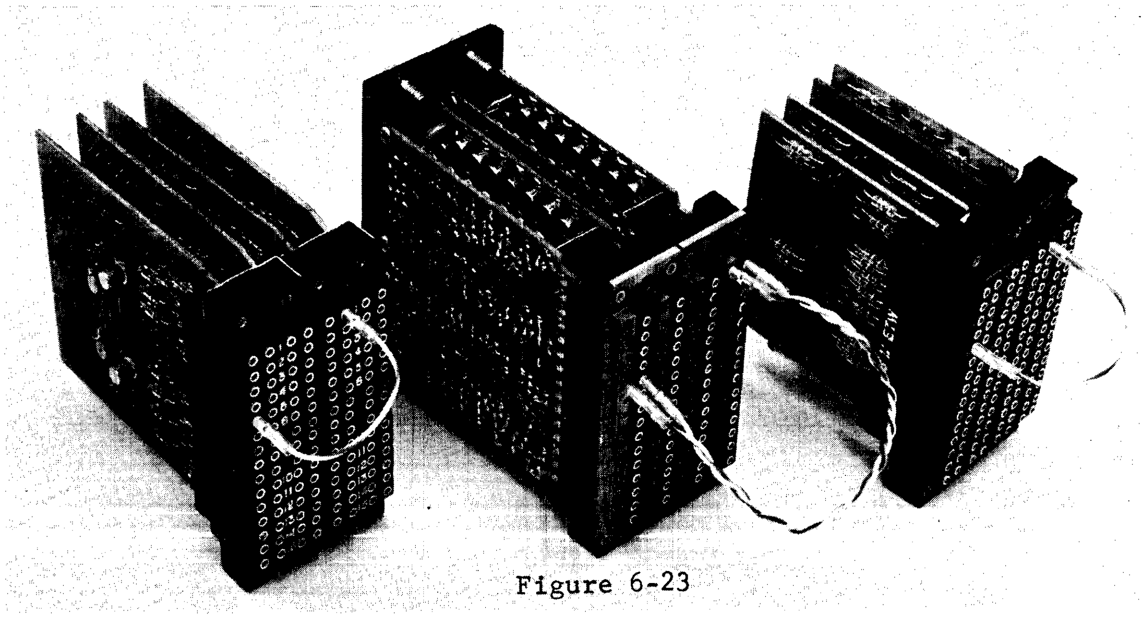


Figure 6-23

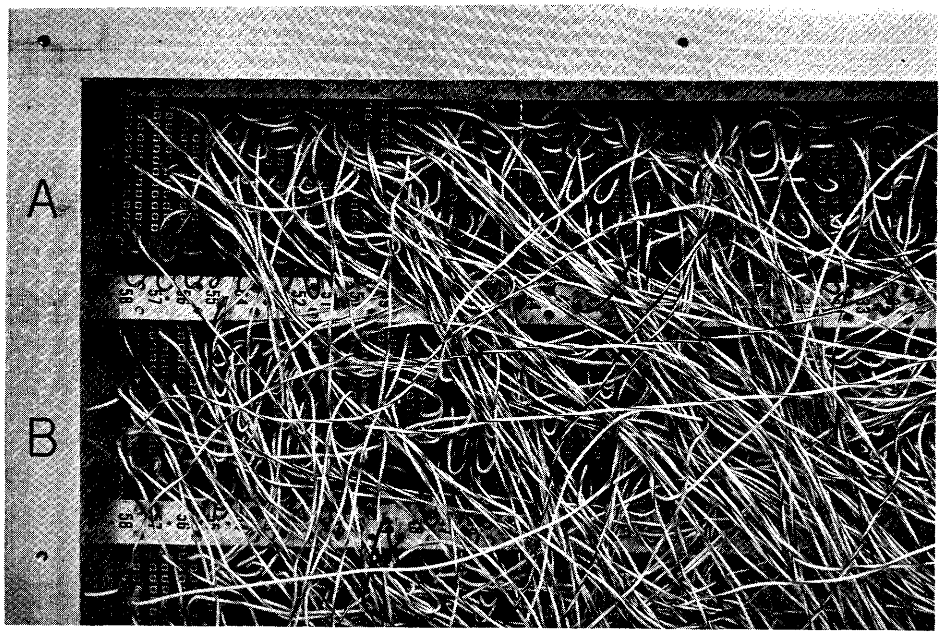


Figure 6-24

Figure 6-24 shows the "wire side" of a computer main frame. Notice the power inputs to each plug. What type of circuitry is being used?

The double row of pins at location A58 connect to a pluggable card on the other side of the chassis (see Figure 6-23). The two taper-pin jacks for each pin number allow for series connections (see Figure 6-14).

INTEGRATED CHIP LOGIC

Recent developments in the semiconductor industry have produced an entirely new product line. The concept is to micro-miniature semiconductor components and to crowd dozens of them onto a small chip. Transistors lose their familiar shape and become simple junctions on the chip.

This concept permits powerful computers to occupy much less space than with conventional logic. A medium-sized computer constructed from chip logic could easily be held in one hand.

The photograph in Figure 6-25 illustrates the comparison of typical integrated circuits to the other three families of CDC logic circuitry. The numbers indicate how many of a particular card type would be required to construct a 50-bit register.

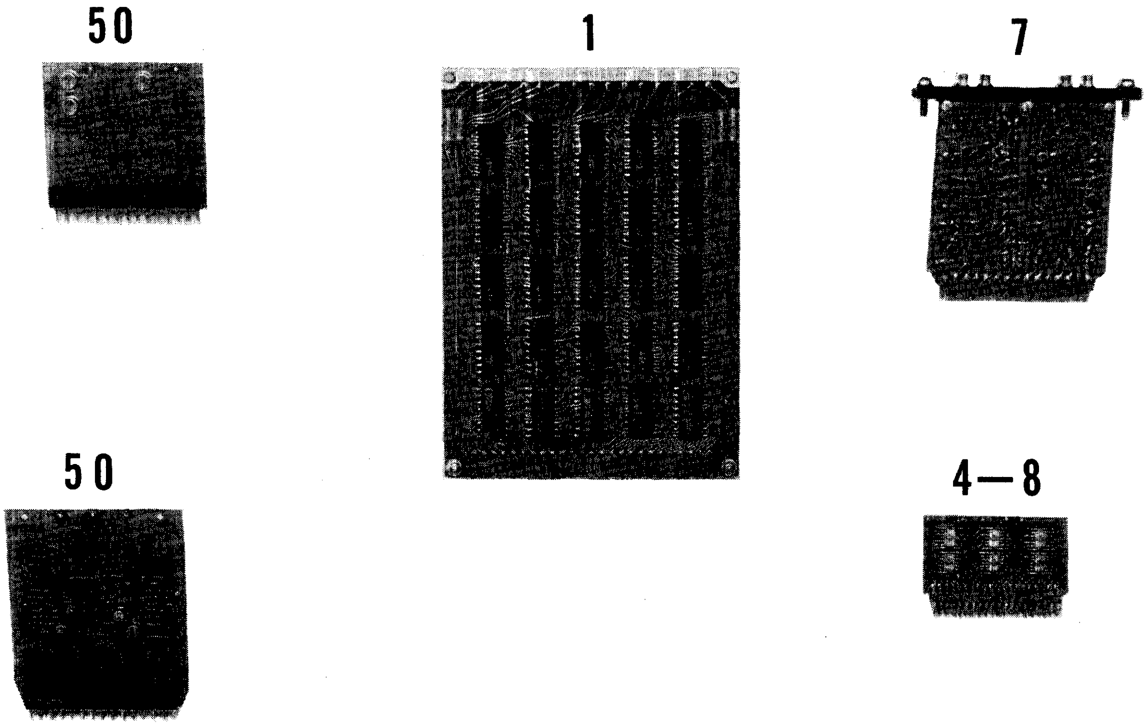


Figure 6-25. Physical comparison of logic families

Physical Construction

The first type of integrated logic to be discussed will be the 14-lead flat pack. Each flat pack contains two to four inverters, depending upon circuit configuration. Figure 6-26 illustrates a pluggable 40-pin module containing six flat packs.

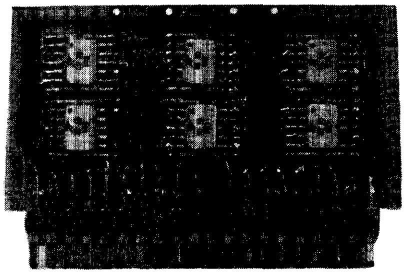


Figure 6-26. 14-lead flat pack

Logic Operation

The illustration in Figure 6-27 show the logical symbols for typical circuits. The triangle denotes an AND gate and the circle is indicative of an inverter.

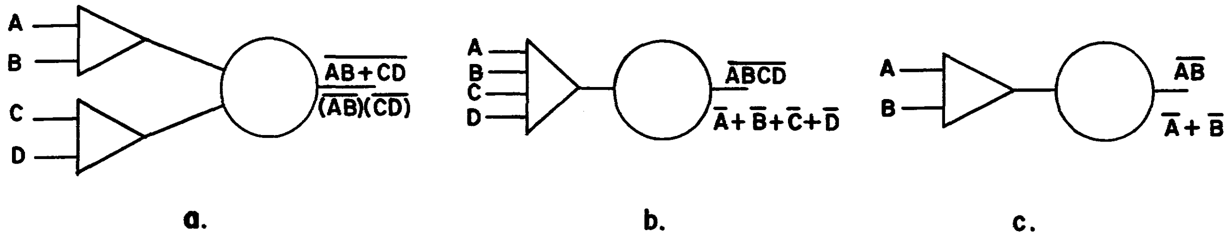


Figure 6-27. Logical Representations

The first circuit (a.) will produce a logical "0" output if either A and B or C and D are logical "1's". It will, therefore, output a logical "1" as long as either A or B is a logical "0" and either C or D is also a logical "0" $(\bar{A} + \bar{B})$ $(\bar{C} + \bar{D})$.

Circuit b. will output a logical "0" only if A, B, C, and D are all logical "1's". Any input being a logical "0" would render a logical "1" output $(\bar{A} + \bar{B} + \bar{C} + \bar{D})$.

Circuit c. will output a "0" if both A and B are present; a logical "1" if either or both inputs are a logical "0" $(\bar{A} + \bar{B})$.

The equivalent circuit for the logic symbol illustrated by Figure 6-27 (a) is shown in Figure 6-28. Note the boolean output expression.

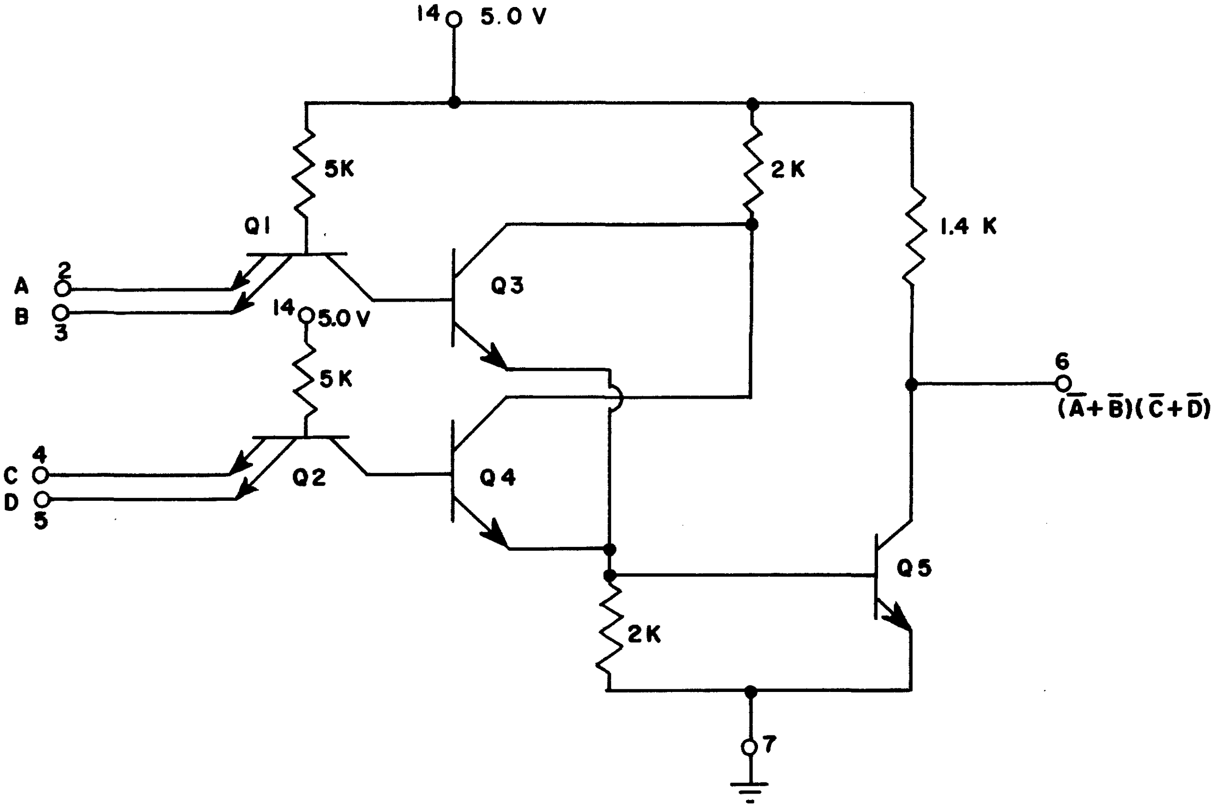
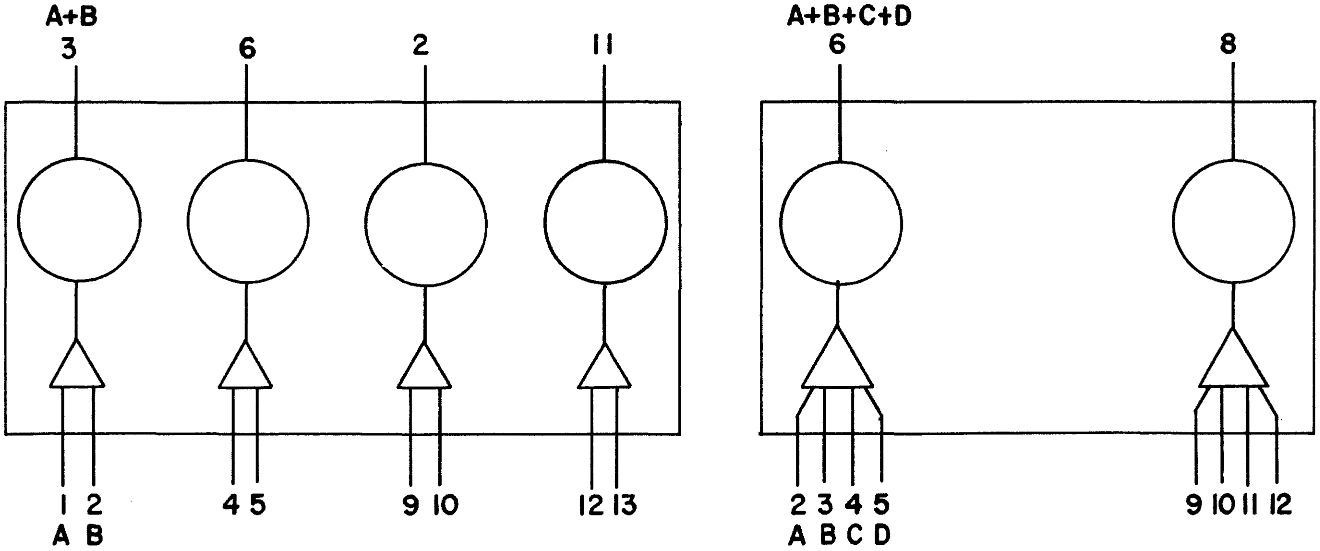


Figure 6-28. Integrated circuit with AND OR inputs

Another identical circuit is also contained within the same flat pack. Pin assignments for standard configurations are illustrated in Figure 6-29.



4 AND inverters

2 AND inverters

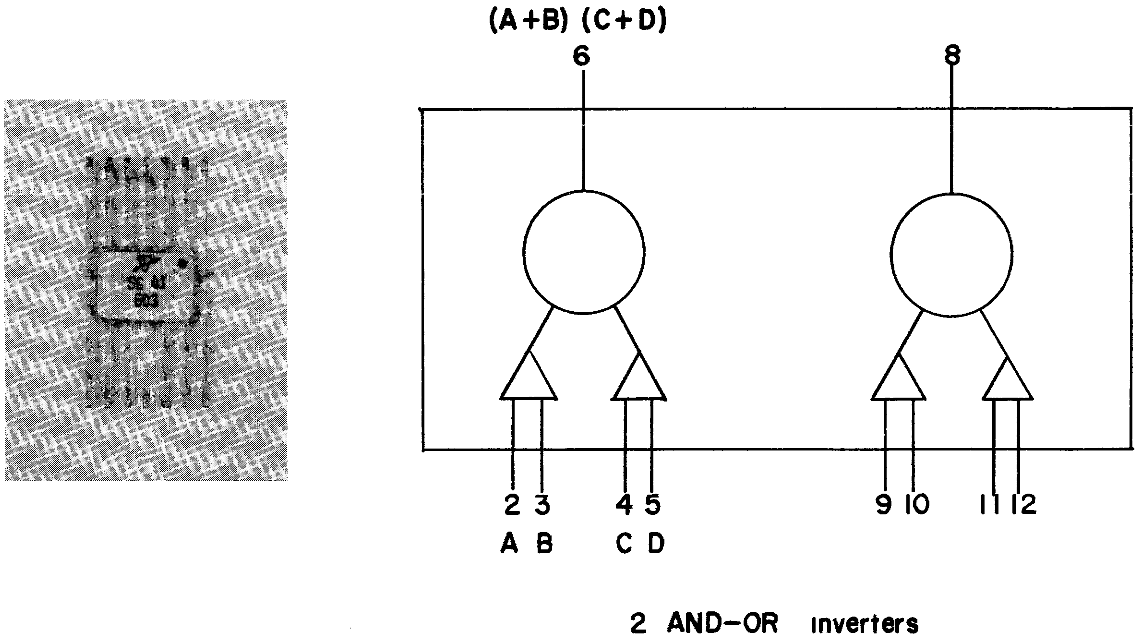


Figure 6-29. 14-lead flat pack pin assignments

Each circuit consists of three stages: an input stage, an emitter follower and an output stage. Logical levels are approximately +3.5 v for a logical "1" and +0.24v for a logical "0".

Circuit Operation (Refer to Figure 6-28)

When both inputs to one of the multiple-emitter transistors, Q1 or Q2, are at the logical "1" potential, the transistor is biased in the "off" condition. The base-collector junction forms a forward biased diode which applies a portion of +5v to turn on the appropriate establishes a positive potential on the base of Q5 causing it to conduct. The output is then a logical "0" or about +0.24v (Drop across Q5 with grounded emitter).

If either input is a logical "0", the base-emitter relationship causes the input stage to conduct. This causes the base of the emitter follower to go negative which cuts it off, in turn cutting off the output stage, Q5. The output of the circuit is then a logical "1" or about 3.5v.

Another type of integrated logic is the inebrid DTL (diode transistor logic) module shown in Figure 6-30.

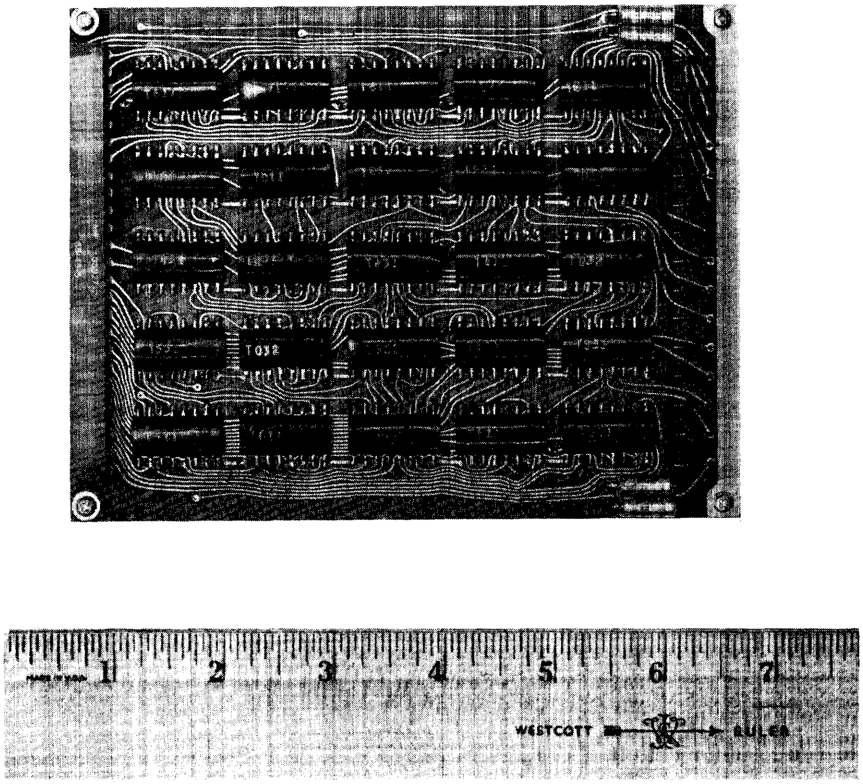


Figure 6-30. CDC intebrid "Fifty-pack"

The module measures 4½ by 6 inches and contains fifty epoxy encapsulated circuits, twenty-five on each side. The logic symbols and equivalent circuits closely approximate those of 1604 logic, except that the logic operates at saturation.

The logic supply voltage is +6v and the operating levels are .9v for a logical "0" and 2.2v for a logical one. The one hundred pin connecter at the left end of the module provides input and output connections for the circuits.
The other end of the module is accessible when installed in the computer and has thirty test points for maintenance purposes.

The number on each circuit identifies its type. This facilitates replacement and also indicates which diagram to examine for a logic analysis.

Figure 6-31 illustrates a single integrated circuit and the seven stages of construction.

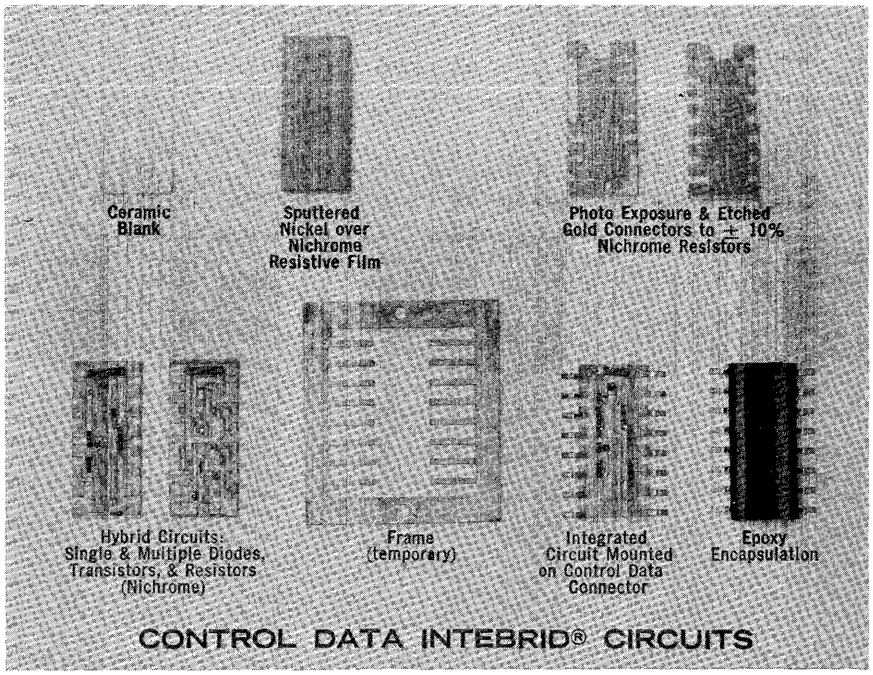

Ceramic Blank
Sputtered Nickel over Nichrome Resistive Film
Photo Exposure & Etched Gold Connectors to ± 10% Nichrome Resistors
Hybrid Circuits: Single & Multiple Diodes, Transistors, & Resistors (Nichrome)
Frame (temporary)
Integrated Circuit Mounted on Control Data Connector
Epoxy Encapsulation
CONTROL DATA INTEBRID® CIRCUITS

Figure 6-31. Construction of an integrated circuit

Figure 6-32 illustrates the circuit enclosed on a typical 16-lead interbrid chip. Notice the simularity to the 1604 type circuit shown in Figure 6-7. The absence of the biasing network is indicative of saturated logic.

The chief advantages of integrated circuits over the conventional types are:

1. Lower manufacturing costs.
2. Lower power and cooling requirements.
3. Less physical space required.

Some disadvantages are:

1. More difficult to replace components.
2. Switching times relatively long.
   a. 15 - 60 nanoseconds for a 14-lead flat pack circuit.
   b. 10 - 25 nanoseconds for a DTL intebrid circuit.

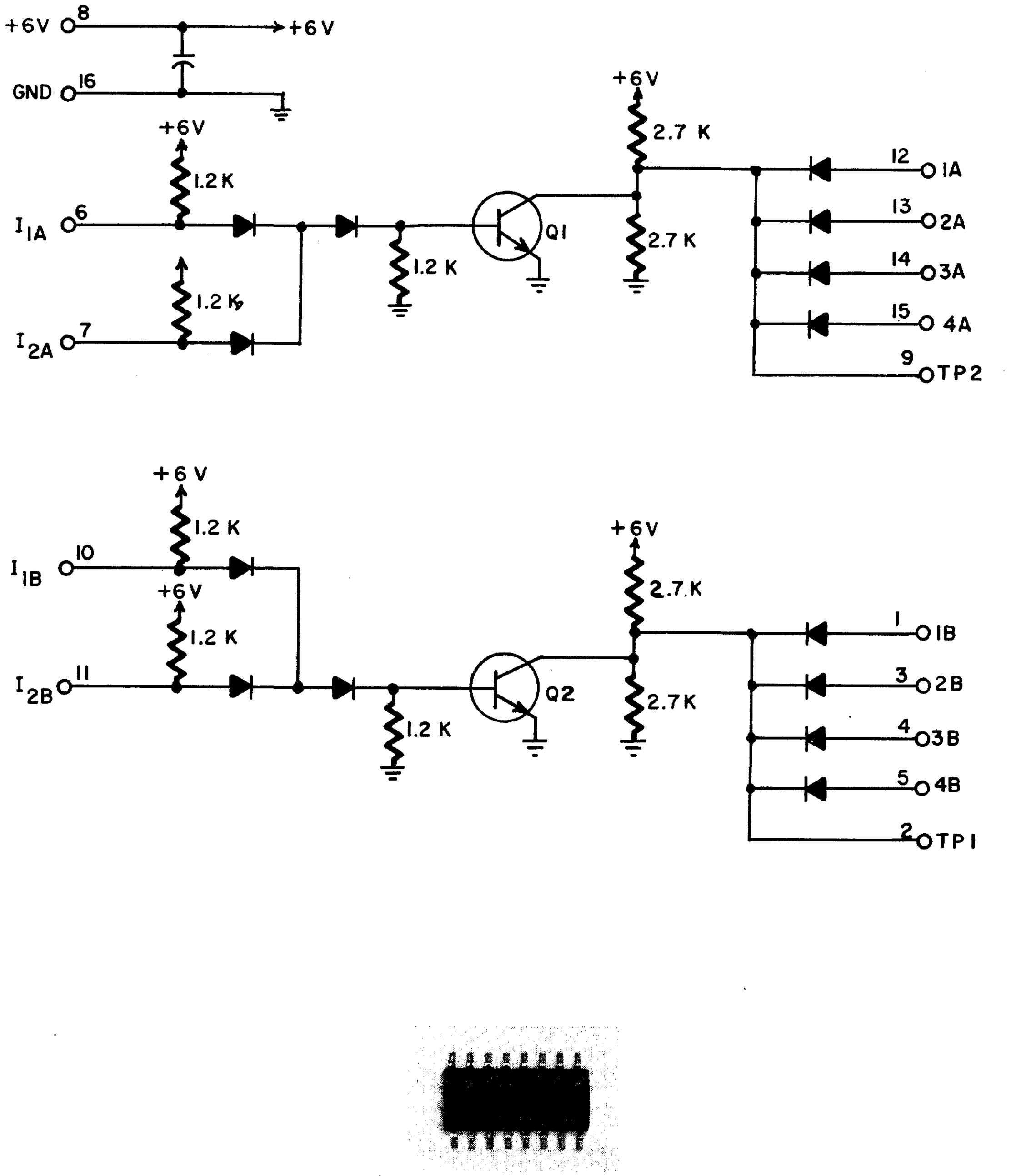


Figure 6-32. DTL intebrid circuit (Type D24)

The incorporation of integrated circuits into computer design should produce a more reliable computer at lower cost to the customer. Technological advances in the development of integrated circuits will someday lead to the pocket-sized computer for the family doctor and the nuclear physicist.

BUILDING BLOCKS OF LOGIC CIRCUITRY

It was previously stated that the basic building block of a CDC computer is the inverter circuit. Several types of logic circuitry have been analyzed and you should now understand why an inverter has a given output when various input combinations are applied. The type of logic being used and voltage levels are now insignificant in relation to understanding the operation of a digital computer.

From this point forward, computer circuits will be expressed as logical symbols such as the rectangular box to represent an inverter. You know what happens inside the box and what to expect as an output.

The different families of logic circuits are slightly different symbols to represent some of the circuit configurations. Therefore, it will prevent unnecessary confusion if one set of symbols is adopted and future discussions are limited to the one selected. The remainder of the text will therefore be referenced to the 3600 family of circuits.

By using the standard inverter circuit, it is possible to "build" other circuits necessary for computer operation. The majority of computer circuitry is comprised of the inverter, the flip-flop and the control delay. The flip-flop is merely two interconnected inverters and the control delay is made up of three interconnected inverters with clocked inputs. Figure 6-33 illustrates the logical symbols for each type of circuit.

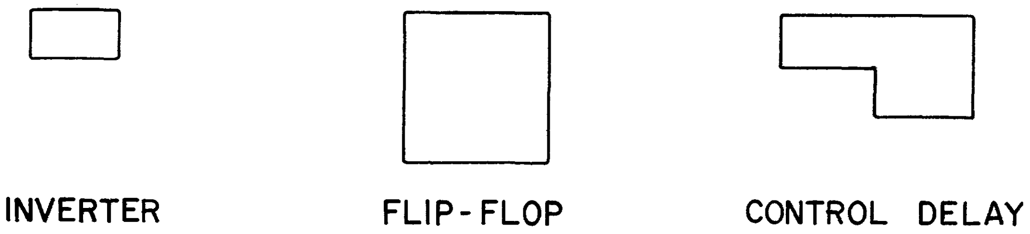


Figure 6-33

Each logic symbol represents a different logic circuit and will be identified by an alpha character followed by three digits.

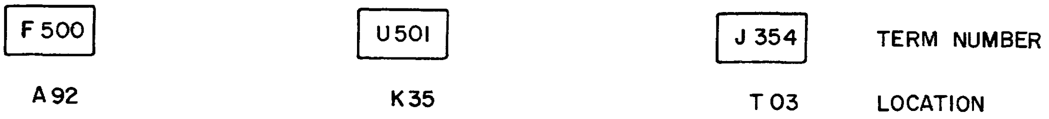
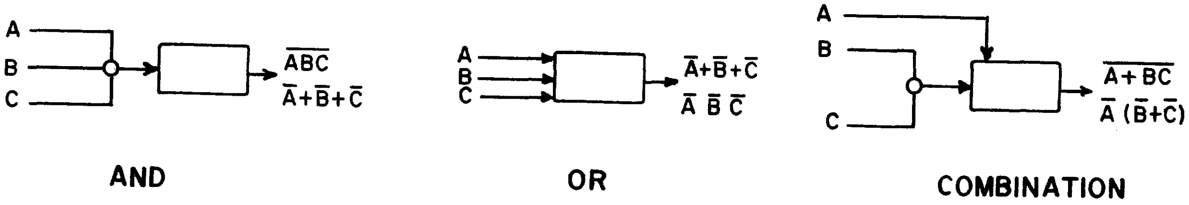


Figure 6-34

Another number will be adjacent to the symbol and identifies the physical location of that circuit in the computer main frame. A flip-flop may have two location numbers because it is sometimes on two different cards. A control delay will always require two cards and consequently will also have two location numbers.

INVERTER CIRCUIT

The inverter circuit effects a 180° phase shift between input and output. Inputs are either ANDed or ORed or may be a combination of both. (See Figure 6-35)



A
B
C
$\overline{ABC}$
$\bar{A}+\bar{B}+\bar{C}$
AND
A
B
C
$\overline{A+B+C}$
$\bar{A}\,\bar{B}\,\bar{C}$
OR
A
B
C
$\overline{A+BC}$
$\bar{A}(\bar{B}+\bar{C})$
COMBINATION

Figure 6-35

If the inputs are ANDed, all must be logical ones to produce a logical "0" output. Any other combination of inputs (any input a logical "0") will produce a logical "1" output. Figure 6-36 illustrates some of the possible combinations.

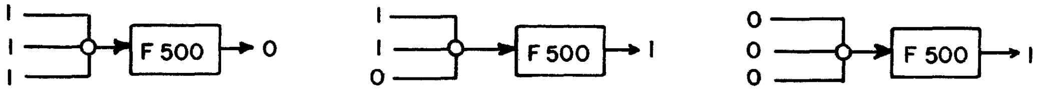


Figure 6-36. ANDed Inputs

If the inputs are ORed, any logical "1" input will force a logical "0" output. Figure 6-37 illustrates the OR circuit.

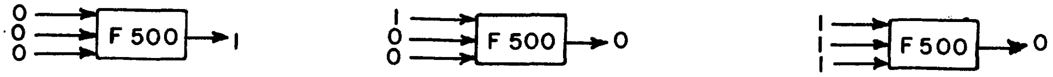


Figure 6-37. ORed Inputs

If the inverter has a combination of AND and OR inputs, the circuit will produce a "0" output if the OR term is a logical "1" or if the AND gate is made. The circuit will only produce a logical "1" output if the OR term is a logical "0" and at least one leg of the AND gate is a zero. Figure 6-38 illustrates the OR circuit with an ANDed input.

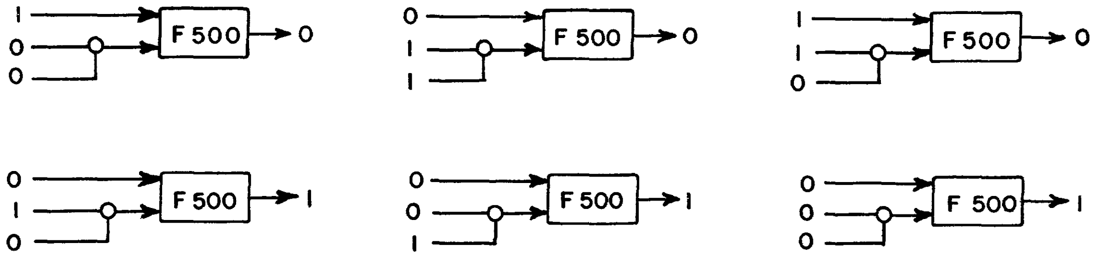


Figure 6-38. Inverter with OR and ANDed Inputs

The design engineer decides what combinations of inputs should feed each circuit and the circuits are then permanently wired to that configuration.

The small circle exterior to the logic symbol represents the AND gate but remember that the AND gate is actually on the inverter card (refer back to Figure 6-15).

Now that you understand the logical operation of the inverter, let's connect two of them together to form a flip-flop.

FLIP - FLOP

All temporary storage of information in the computer is accomplished by flip-flops. A flip-flop is composed of two single inverter circuits interconnected as shown in Figure 6-39. Each rectangle represents a single inverter. One inverter constitutes the set side of the flip-flop; the other, the clear side (Figure 6-39).

The flip-flop is switched to the "1"(set) state by a set input that is a "1". Conversely, it is switched to the "0" (cleared) state by a clear input that is a "1".

The storage ability means that the flip-flop remains in the state indicative of the last "1" input received. If a "1" pulse is present at the set input, the output of A000 will be a "0", which, in turn, feeds A001 giving an output of "1". The output from A001 feeds A000 which causes the output of A000 to remain a "0" until a "1" is present on the clear input. After the "1" on the set input line returns to "0", the flip flop remains in the "1" (set) state.

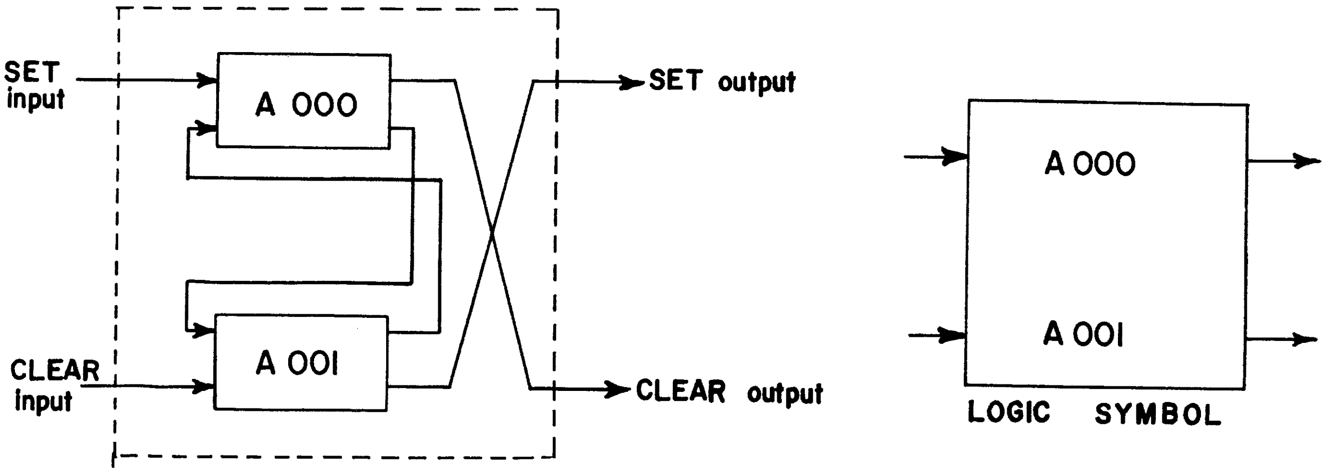


Figure 6-39. Interconnected Inverter Circuits

The operation is similar when a "1" appears on the clear input line. A zero impressed on either input has no effect.

When the flip-flop is set, A001 has a "1" output and A000 has a "0" output. Conversely, when the flip-flop is cleared, A001 has a "0" output, and A000 has a "1" output.

You probably remember, when you were studying programming back in Chapter IV, that the term "register" was frequently used. A register is nothing more than a group of flip-flops, each capable of temporarily storing a logical "1" or a logical "0". A register is not that vent on top of the computer when the hot air comes out. Perhaps you remember that instructions were read from memory into the F register when they were translated. Because the instruction is a 24-bit word, the F register is a group of 24 flip-flops, each independent of all others. Figure 6-40 illustrates the F register containing an ADD instruction with indirect addressing from memory location 12345_8.

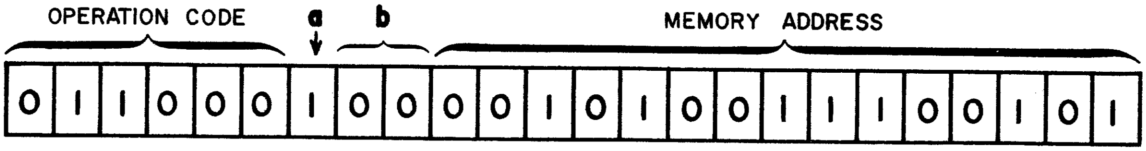


Figure 6-40. F register

Each box in the preceding figure represents an individual flip-flop.

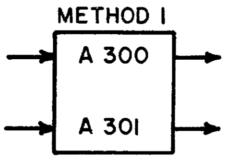
If the box contains a "1", the flip-flop is set or in the "one" state. The flip-flops in the F register feed a translation network of inverters which interpret the instruction in the register and determine which operation is to be performed.

Flip-Flop Numbering

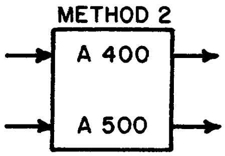
Various systems for numbering flip-flops are used and it is advised that the student be familiar with as many as possible. In most instances a common system is used for each type computer. However, the numbering may differ between registers within the same system.

The following are the most common methods of numbering.

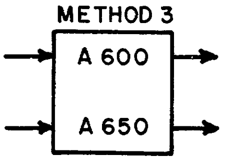
1) Set input to A300, set output from A301
Clear input to A301, clear output from A300



2) Set input to A400, set output from A500
Clear input to A500, clear output from A400



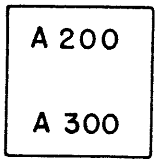
3) Set input to A600, set output from A650
Clear input to A650, clear output from A600



The larger term number in each of the 3 cases denotes the set output; the smaller number denotes the clear output.

Additional investigation should reveal a difference between methods 1 and 2. In method 1, the last digit of the inverter number is increased or decreased by one; in the second, the first digit is varied by one. The odd last digit used in the first numbering method represents the set output. In the second method the odd first digit represents the set output.

The third method adds a bias of 50 to differentiate the set input from the clear input side. The letter and remaining two digits are used to indicate the stage position of the register held by the flip-flop. Figure 6-41 illustrates this.



The 2 and the 3 designate the set and clear side inputs. The 00 in both cases denote that this flip-flop represents the 2^0 power bit in the register. The literal designator A locates the flip-flop in the A register.

To become familiar with the flip-flop and inverter, determine the outputs of the circuits in Figures 6-41 to 6-43. Make a list of the inverter terms showing the appropriate digit that must be in the F register for that inverter term to have a logical one output.

The flip-flops to the left on each page represents the upper nine bits (three bits on each page) of the F register. The inverter terms fed from these outputs are control terms used by the computer to designate that the F register is holding a particular function code and "a" and "b" designators.

For example, the upper octal digit of F must be a 4 in order that F274 = 1. This is usually written F274 = 4X. The X signifies that the F274 term does not translate the second octal digit.

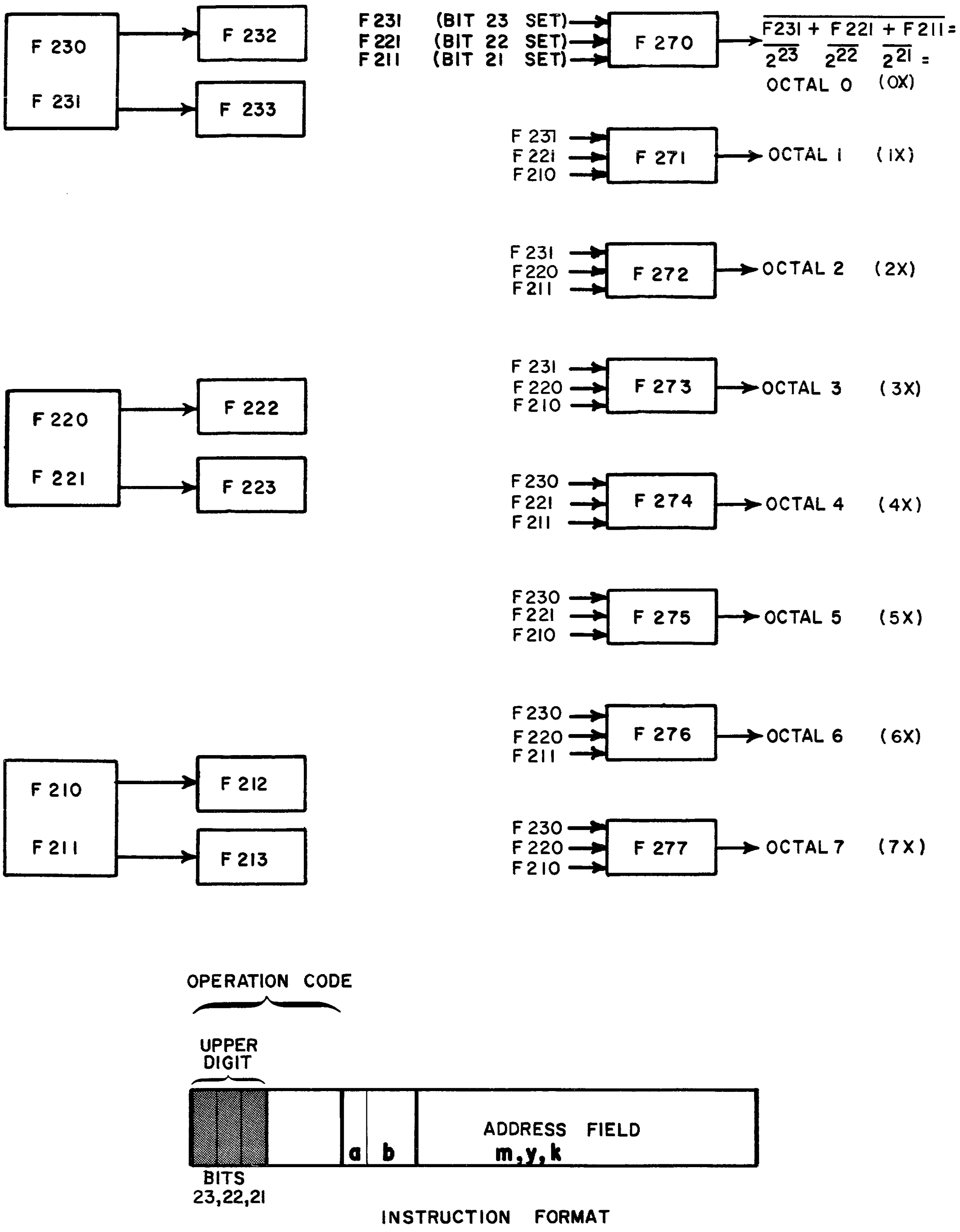


Figure 6-41. Function Translation (Operation code upper digit)

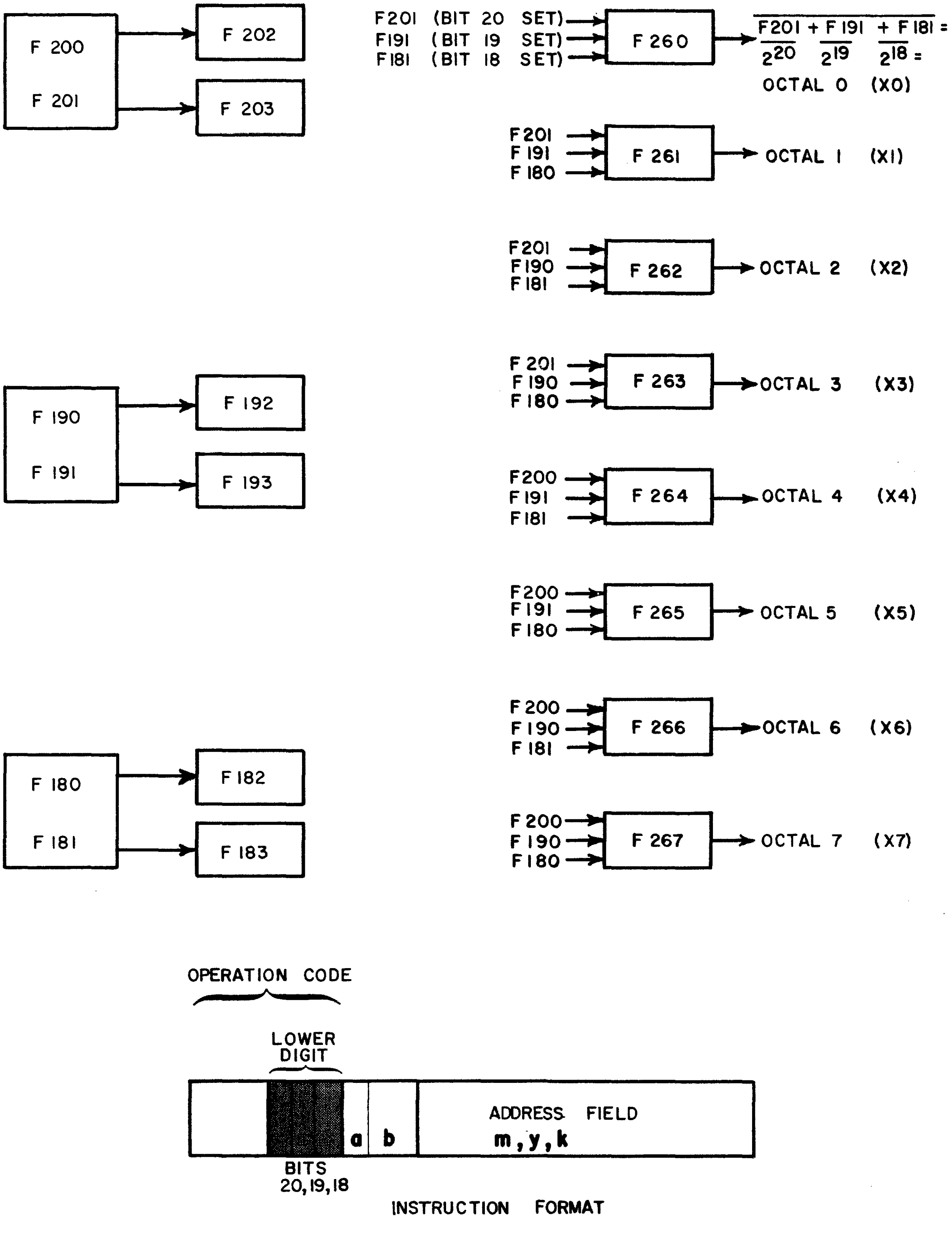


Figure 6-42. Function Translation (Operation code lower digit)

The outputs of the F26X Terms in Figure 6-42 have been translated for you. If all three flip-flops are cleared, as they would be with a 30 in the F register, the set outputs would all be zeros and the clear outputs would all be ones, as illustrated. Remember that the set outputs actually come from inverters F201, F191, and F181. These three terms feed F260 as ORed inputs. All zeros into an inverter allow a logical one output. The only time F260 would output a one would be if the lower octal digit of the operation code was an octal zero (20,30,40).

Label the inputs to the remaining F26X terms. Only one of them should output a one at any given time. Hint! The inputs to F266 should be F200, F190, and F181 or perhaps F202, F192, and F183. Did you notice any significance to the numbering sequence of the inverters?

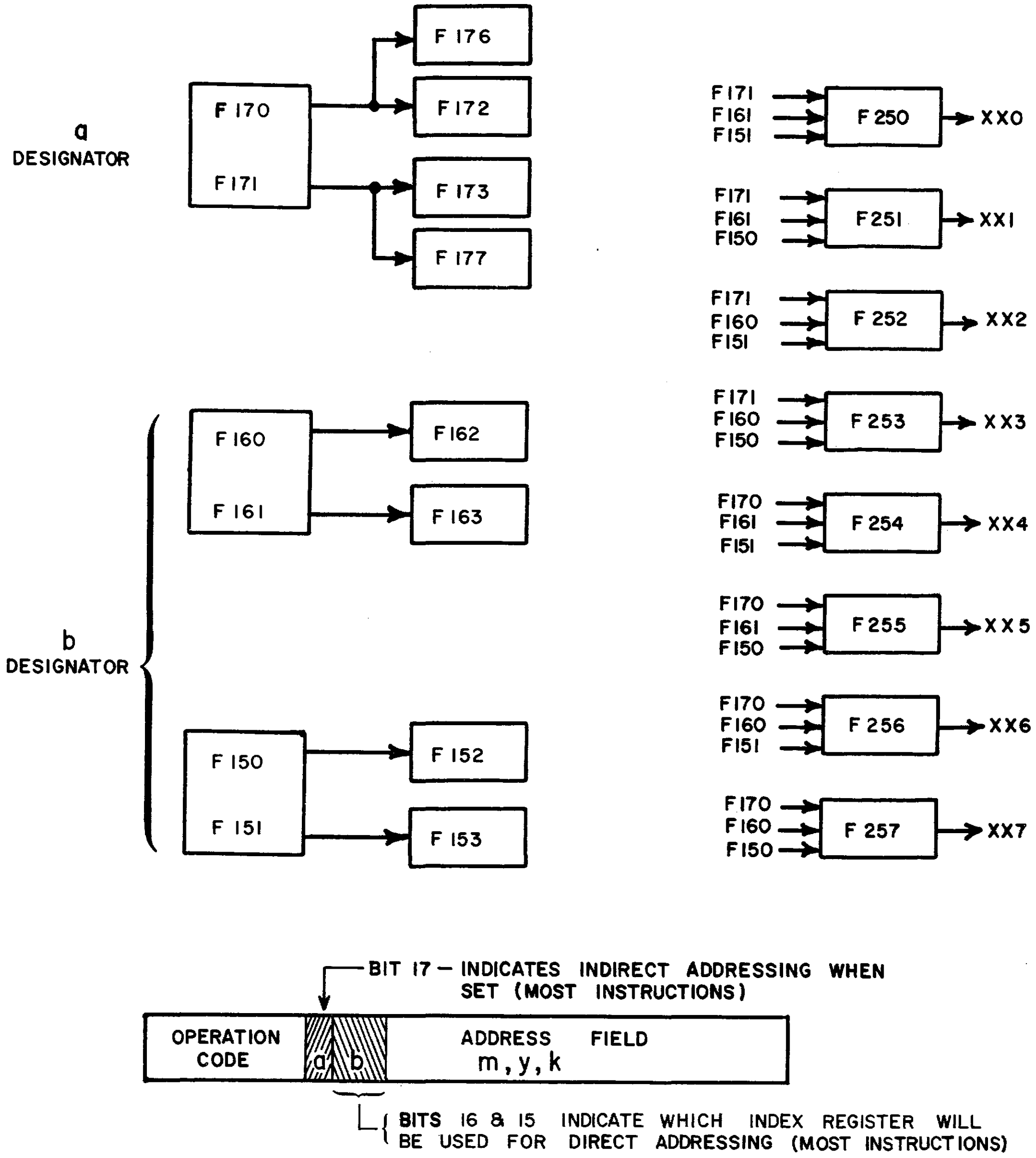


Figure 6-43. Function Translation (a and b designators)

Bit 17 usually indicates whether or not indirect addressing is to be performed. The two additional inverters, F176 and F177, are required because many operations are conditioned by that bit. Remember, each inverter can feed a maximum of eight circuits.

THE CONTROL DELAY

The control delay is a combination of three interconnected inverters with additional clock inputs from the master clock circuit. Its function is to delay a logical "1" input for a controlled amount of time.

The computer is able to execute instructions by performing various operations in a logical sequence. Interconnected control delays form a timing chain to which a logical sequence of events may be synchronized. The delay time, using 3600 logic, is 62.5 nanoseconds. All control delays in the computer are synchronized by the same master clock to insure constant delay times. Less expensive RC delays could be used but variations in delay times would present timing problems.

Inputs to control delays are normally present for one phase time and are usually synchronized by the output of a preceding control delay. Figure 6-44 illustrates a typical configuration. Notice that term numbers alternate. (EVEN, ODD, EVEN)

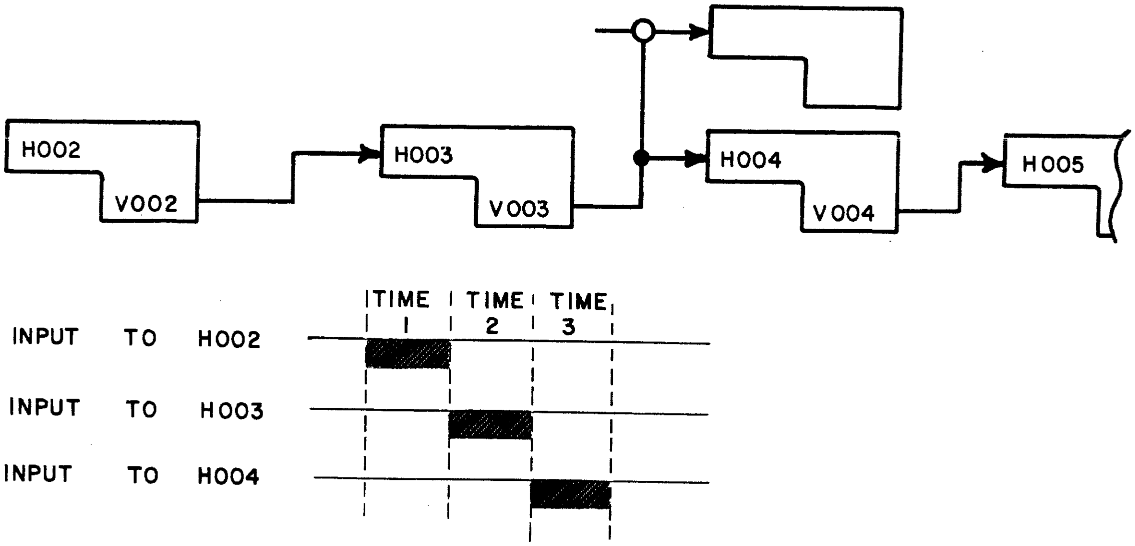


Figure 6-44

Definition of a Clock Phase Time

The master clock oscillator drives a parallel string of clock amplifier cards, type C01. If the output is taken directly from the C01 card, it is referred to as a raw clock. If the C01 card feeds a standard inverter, the inverter is referred to as a clock slave. Due to the inverter switching time, slave outputs will delay the raw clock by 20 nanoseconds.

The alternating pulses from the raw clock are referred to as ODD and EVEN. A raw clock (ODD) will have a logical zero output at an odd time and a logical one output at an even time. The same relationship holds true for the raw clock (EVEN) (See Figure 6-45)

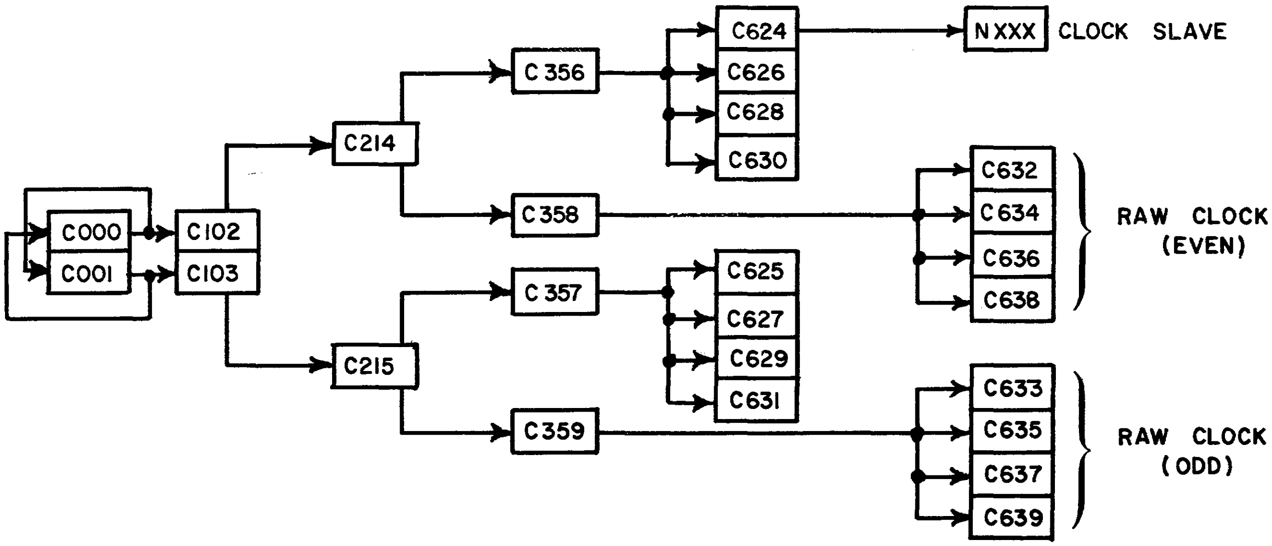


Figure 6-45. Master Clock Pyramid

Most CXXX terms in the computer are raw clock outputs (communication register terms are exceptions), whereas a clock slave term usually is an NXXX term.

Description

The control delay consists of a flip-flop (H000) havings its feedback ANDed with a raw clock signal, and one or more inverters (V000) (N000). Each inverter has two ORed inputs; one from the A section of the flip-flop and the other from a raw clock (See Figure 6-46).

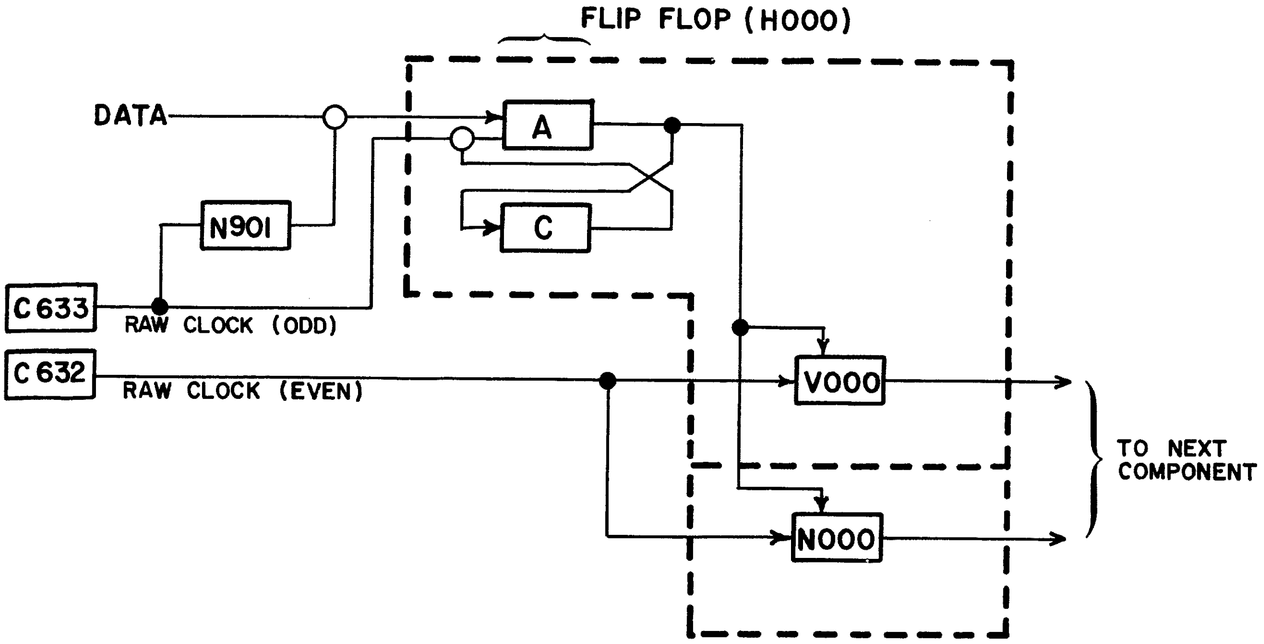


Figure 6-46. Even Control Delay

Because the proceding control delay is EVEN(H000, V000̲) the data input must be at an odd time and the output will occur during the following time (EVEN). The overall circuit does not invert the input, but provides a 62.5 nanosecond delay between input and output.

The clock inputs to a control delay are not shown on logic diagrams. All even control delays will have the same inputs as Figure 6-46. The two raw clock terms would be reversed for an even time.

<u>General Operation of the Control Delay</u> (Refer to Figure 6-46)

The basic operation of the control delay is as follows:

1) Set the "flip-flop" for two phase times.

   a. Set by data pulse during first phase time

   b. Latched up by odd clock during second phase time

2) Enable the output. While the "flip-flop" is set, the side output will be a logical "0". During the second

phase time when the even clock is also a logical "0", inverters V000 and N000 will produce logical ones out.

3) Clear the flip-flop. During the third phase time, the data pulse will be absent breaking the associated AND gate and the ODD clock will be a logical "0", breaking its associated AND gate. Inverter A will output a logical "1" which effectively produces a clear input to the "flip-flop". The logical "1" output from inverter A prevents any further outputs from the control dèlay.

Circuit Operation

Figures 6-46 and 6-47 will aid in understanding the actual circuit operation of the control delay. Figure 6-47 illustrates the quiescent condition and then the series of events as the data pulse is applied

Prior to the arrival of the data pulse, both inputs to inverter A are logical zeros and it outputs a logical "1". This forces logical terms out of inverter C and V000, N000.

The data pulse arrives and, 20 nanoseconds later, N901 outputs a logical "1" making the AND gate and providing the SET input to the flip-flop. Inverter A outputs a logical "0" allowing inverter C to output a logical "1". V000 and N000 cannot output logical ones because the even clock is a "1" during this phase time (ODD).

During the next phase time, the flip-flop latches up by making the "odd clock and inverter C" AND gate. This keeps inverters A and C in the same condition as during the previous phase time. V000 and N000 now output a logical "1" because both the even clock and inverter A outputs are logical zeros.

The flip-flop clears during the next phase time (ODD) because the absence of an input data pulse breaks that AND gate and the odd clock is now a logical "0", breaking the feedback AND gate. Inverter A outputs a logical "1" forcing zeros out of V000, N000, and inverter C.

Another data input at some later time would re-initiate the entire sequence of events. If the data pulse is a continuous signal, the control delay would output pulses at every even time.

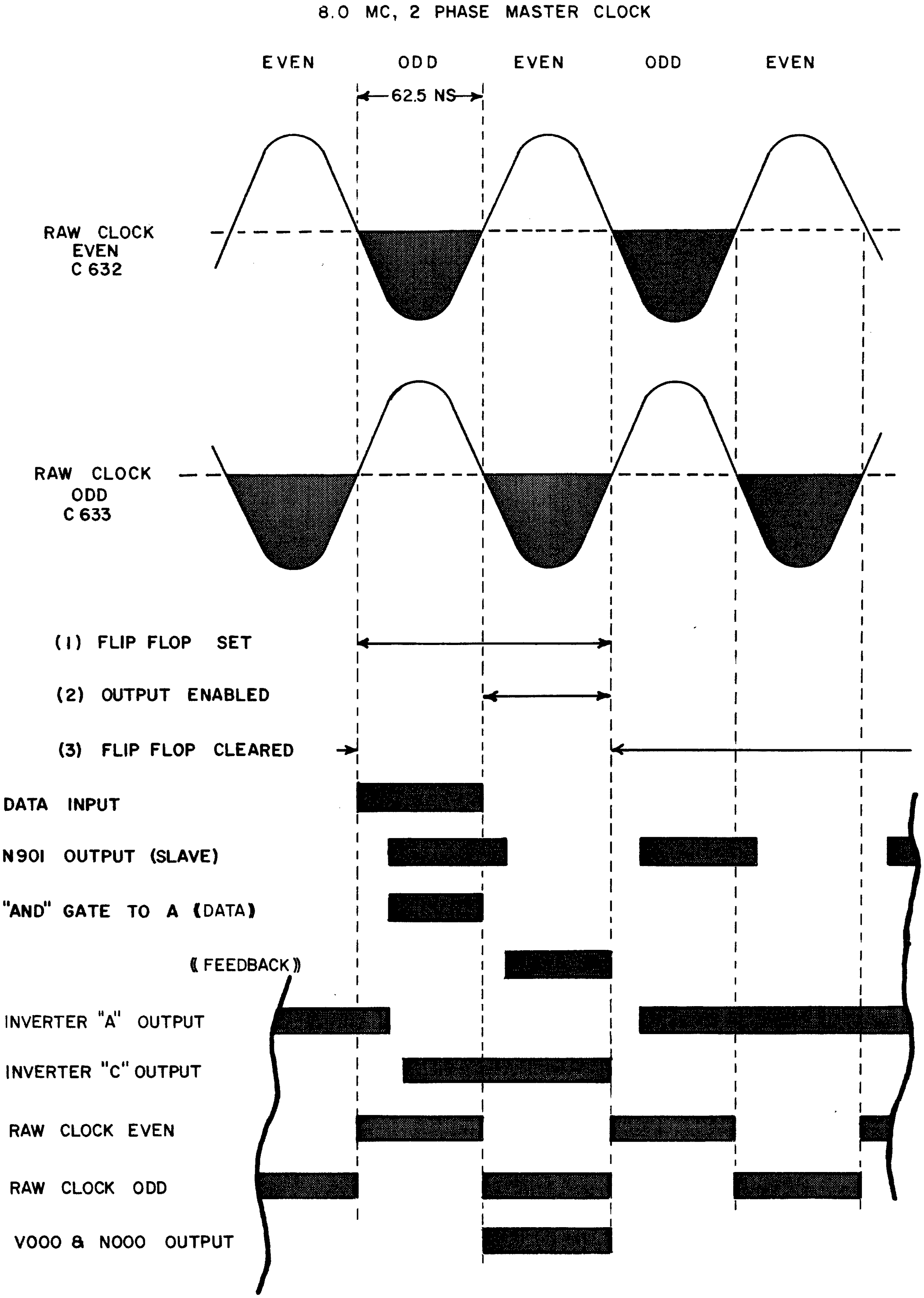


Figure 6-47 Control Delay Timing

Rules for Interconnecting the Control Delay

1) Two clocked symbols may not be gated together.

2) Every input term of an H---equation must contain one and only one clocked symbol.

3) The phase of the inputs to an H--- must be opposite to the phase of the outputs.

4) The last term in an H--- equation must be a C--- (clock) symbol.

5) The input terms of a V--- or N--- equation must be of the same phase as the outputs.

6) The last term of a V--- or N--- equation must be a C---(clock) symbol.

7) No V--- or N--- symbol may appear as an input to a V--- or N--- equation.

8) At least one input term to a V--- or N--- equation must contain an H--- symbol.

9) An H--- symbol must not be combined with a V--- or N--- in an equation term without special consideration of the timing involved.

REGISTERS AND TYPES OF TRANSMISSION

You learned earlier in the chapter that a register is a group of flip-flops used to temporarily store a computer word. Each flip-flop is associated with one bit of the word.

The F register holds the current instruction after it is read from memory, the P register holds the address of the current instruction and the A register contains an operand. Each register serves a different function in the overall operation of the computer, yet each is made up of individual flip-flops which are identical in operation.

The register is the only means the computer has to remember a word read from a storage location. Until the word is returned to permanent storage, it must be continuously held in at least one register. Remember that inverter outputs are continually controlled by the inputs, whereas a flip-flop will remain set until a clear "pulse" is applied. Therefore, the flip-flop is often referred to as a bistable memory device.

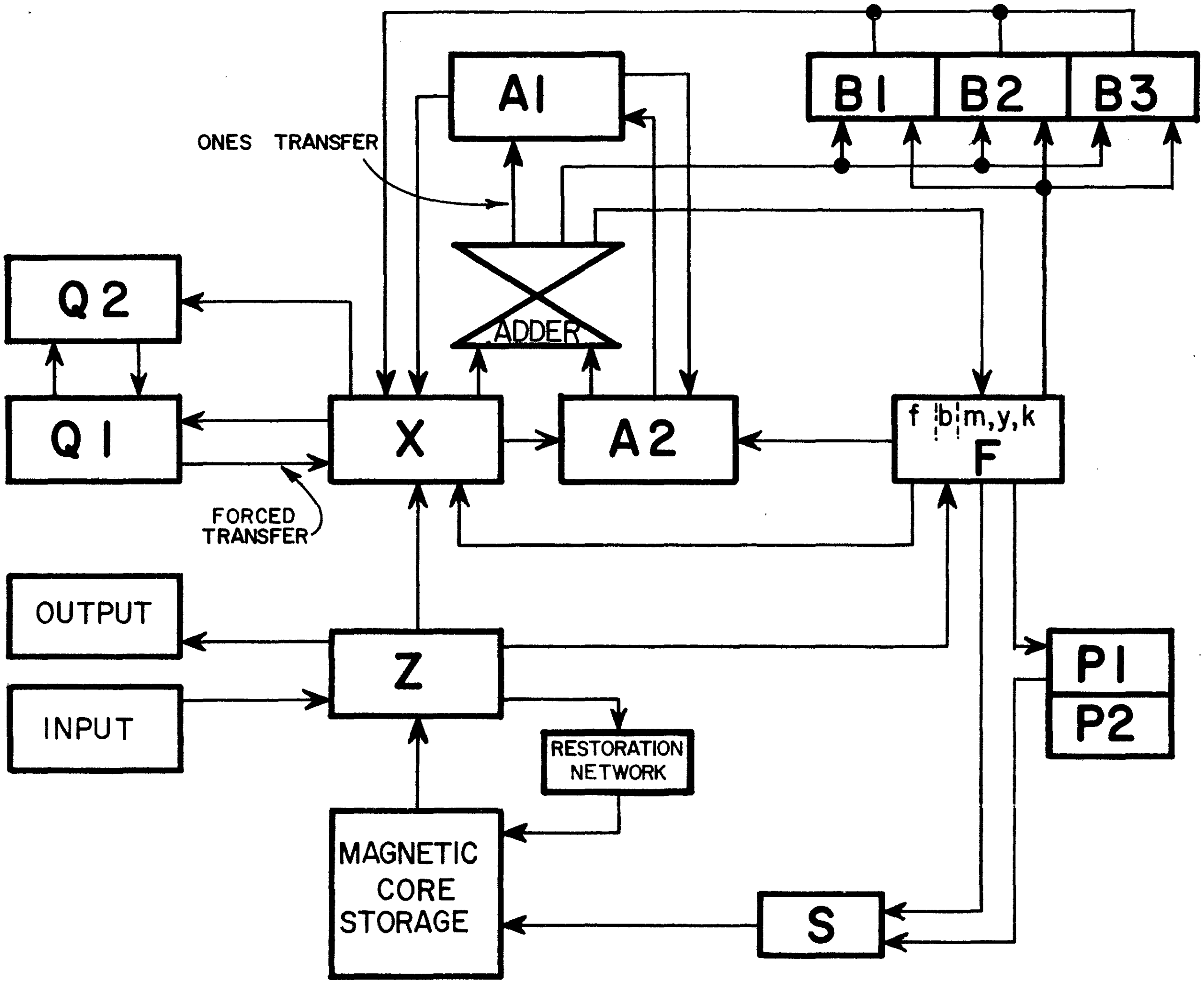


Figure 6-48 Basic Computer Block Diagram

TYPES OF TRANSMISSION

Data inputs to register flip-flops may come from a variety of sources. The source may be magnetic core storage, another register or some peripheral equipment communicating with the computer. The type of circuit actually feeding the register usually determines the type of transmission.

Ones Transfer

Some inputs to the Al register are from the adder (standard inverters, Figure 6-48). An inverter has a single output and cannot feed both set and clear sides of the flip-flop. In this situation, the entire register is normally cleared (all flip-flops to the "0" state) and then the appropriate flip-flops are "set" with a ones transfer during the next phase time. Figure 6-49 illustrates a ones transfer between the adder and the lower three stages of the Al register.

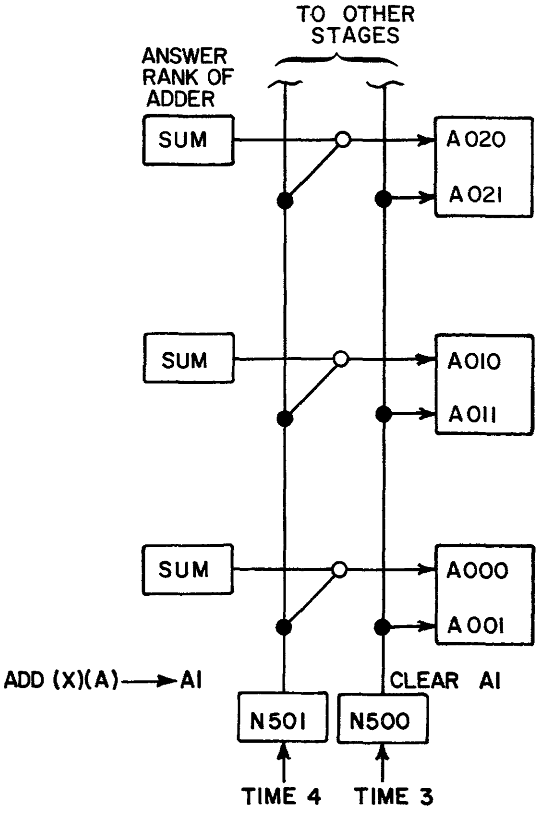


Figure 6-49. Ones Transfer

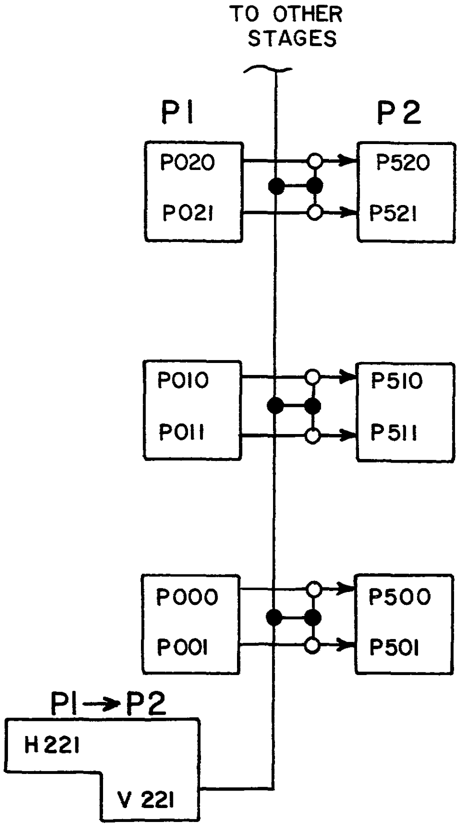


Figure 6-50. Forced Transfer

Forced Transfer

A forced transfer between registers is accomplished faster than the ones transfer due to the elimination of the clear pulse. Here the clear signal is not necessary because both sides of the transmitting register are being gated into the receiving register. This type of transmission between registers is called forced transmission, because the receiving register is forced to the same state as the transmitting register, (Figures 6-48 and 6-50).

Ones Complement Transfer

In Chapter III learned how to form the ones complement of a binary number. Figures 6-51 illustrates how the computer could form the ones complement of a number.

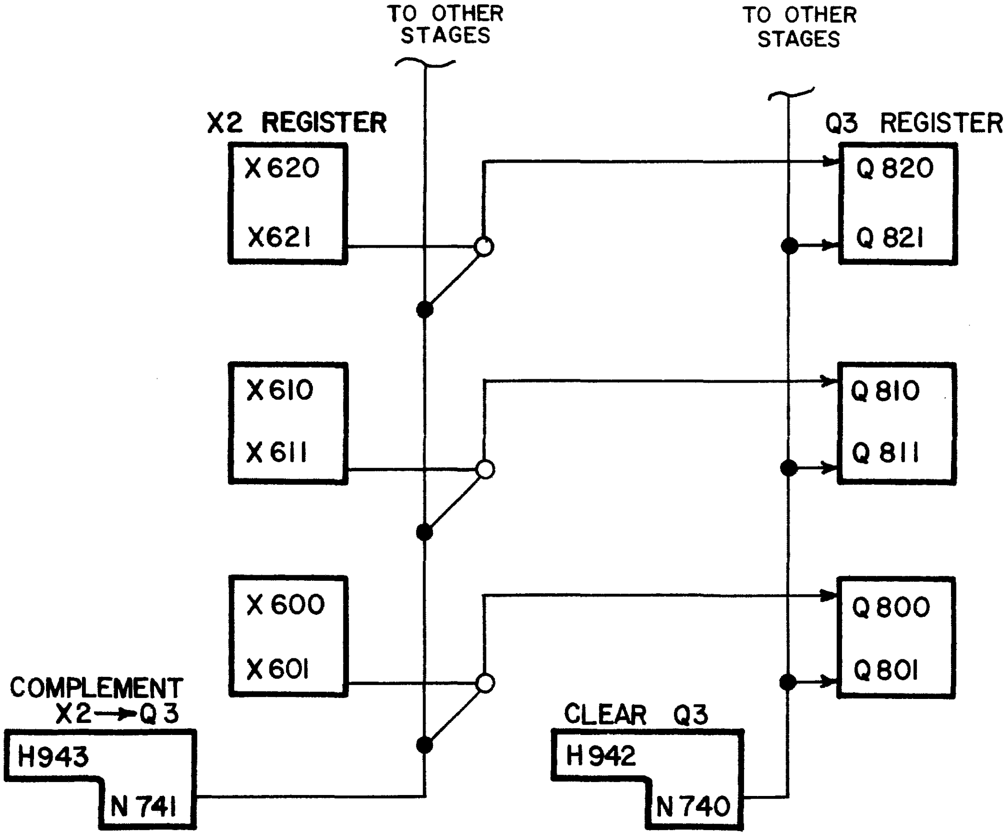


Figure 6-51. Ones complement transfer

The output of N740 would clear the Q3 register and then the transfer would be made by the enable pulse out of N741 during the next phase time X2 and Q3 are not shown on Figure 6-48).

Zeros Transfer

Another type of transfer is sometimes used. All flip-flops in the register are set and then the appropriate ones are cleared out with a zeros transfer. (See Figure 6-52)

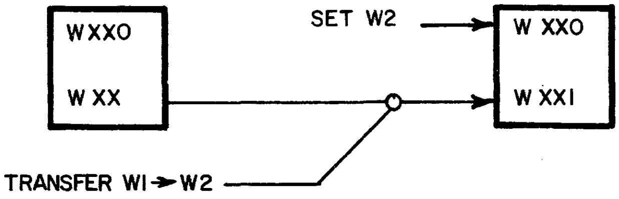


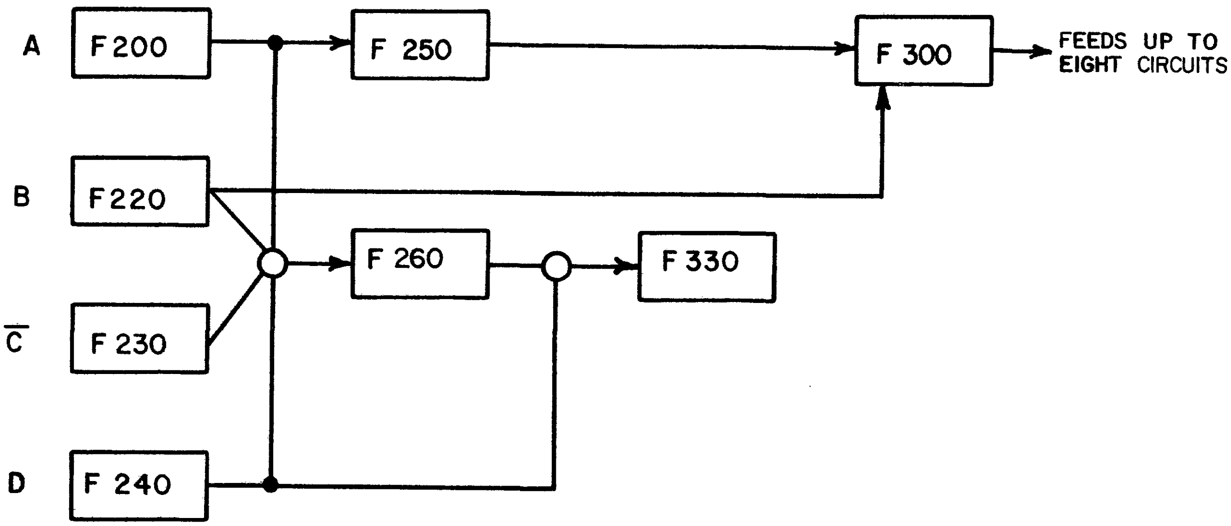
Figure 6-52. Zeros Transfer

The end result is the same as the ones transfer.

Complete the following exercises and check your answers with those at the end of the chapter.

1. What are the basic differences between the different families of computer logic circuitry?
2. What are the voltage levels of the two logical outputs of a 3600 inverter circuit? A 3600 flip-flop?
3. What three factors are considered in the design of a digital computer?
4. Draw two different circuits, labeling inputs and outputs, that would yield $\overline{B}C + \overline{A}$ as the boolean output equation. Which uses the least amount of circuitry?

5. Illustrate how the following circuit would actually be wired using 1604 logic circuitry. Use as few cards as possible and label your diagram with card types and pin numbers. Reference Figure 6-11 for card configurations.



6. Using the same diagram as exercise #5, illustrate the actual wiring connections you would make if 3600 logic were being used instead of 1604 logic. Use as few cards as possible and indicate pin numbers. Figure 6-16 illustrates the input connections for some common 3600 logic circuits.

7. What would be the output of a flip-flop if logical ones were applied to both set and clear inputs? What state would the flip-flop assume if if both inputs are removed?

8. What is the content of the computer register when power is first turned on?

9. What would happen to the output of a control delay if the clock input pin to the V term is open?

10. What is the difference between saturated and non saturated logic? What are the advantages of each?

OPERATION OF REGISTERS

If a series of flip-flops are used to store a word, this series of flip-flops is called a register. Registers can be singleranked (used to store a word) or doubleranked (used to store a word and also update (add one) to the contents). The double-ranked registers are called counters because they can be used to count by ones. The operation of the Program Address counter,or P register, is shown on the following pages.

THE PROGRAM ADDRESS COUNTER OR P REGISTER

The Program Address counter (P register) provides program continuity by sequentially generating the storage addresses from which the individual instructions of the program are obtained.

The P register holds the address of each instruction. After the appropriate instructions are executed, the count in P is advanced by one to the address of the next instruction. It is possible for the P register to set to a new count by an appropriate instruction. The initial setting of the P register is entered manually from the console. Thereafter, the P register is under machine control.

Technically, the P register is a twos complement, open-ended, additive counter with a modulus of 2^{15}.

The P register is in two ranks. These are kept equal by a pulse which transfers P^2 to P^1 via a forced transfer at every even clock time. The commands which enable control of the P register are:

a) Clear P^1, which clears all states of P^1 to zero. Master Clear also clears P^2 via a $P^1 \rightarrow P^2$ transfer.

b) $P^2 \rightarrow P^1$

c) Advance P, which increases the contents of P^2 by 1.

The P register is broken down into 4 groups as shown in Figure 5-53. The lower three groups consist of four binary bits and the uppermost group consists of the remaining three.

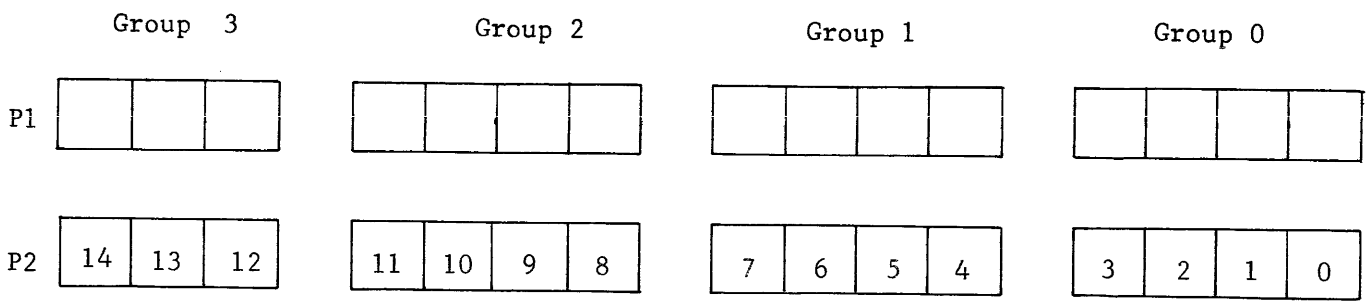


Figure 6-53 P register groups (15 stages -- 2 ranks)

To correctly advance the P register requires:

1) That, initially, Ranks 1 and 2 are equal.

2) A receipt of an advance P pulse.

3) A receipt of a $P^2 \rightarrow P^1$ pulse to allow a forced transfer from $P^2 \rightarrow P^1$.

Advancement of the P register is broken into two basic parts, stage and group advancement, each governed by unique rules conditioned on receipt of an advance P pulse.

STAGE ADVANCEMENT RULES

Basically, the rules for advancing stages within a group are as follows:

1) Always complement the lowest bit (2^0) of P^1 into P^2.

2) If bit 2^0 of P^1 is set, also complement bit 2^1 of P^1 into P^2.

3) If bits 2^0 and 2^1 of P^1 are both set, complement bit 2^2 of P^1 into P^2.

4) If bits 2^0, 2^1 and 2^2 are all set, complement bit 2^3 of P^1 into P^2.

The following illustrations show how the four stages of group 0 are advanced in accordance with the rules above.

1) Complement the lowest bit.

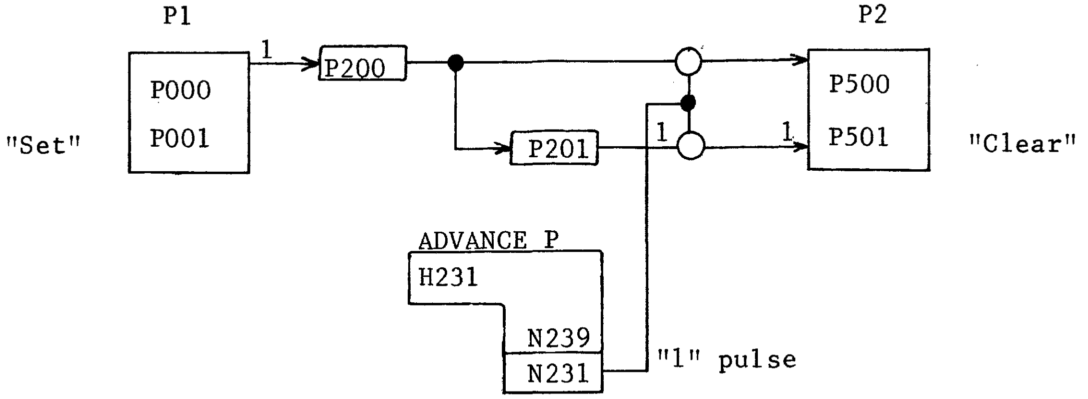


Figure 6-54 Illustration of rule 1

P200 is an inverter off the "set" side output of P000/001, which makes it the same as the "clear" side of the flip-flop. P201, fed by P200, is again the same as the "set" side output. This method is used to acquire more outputs.

2) If the lowest bit is set, complement the next.

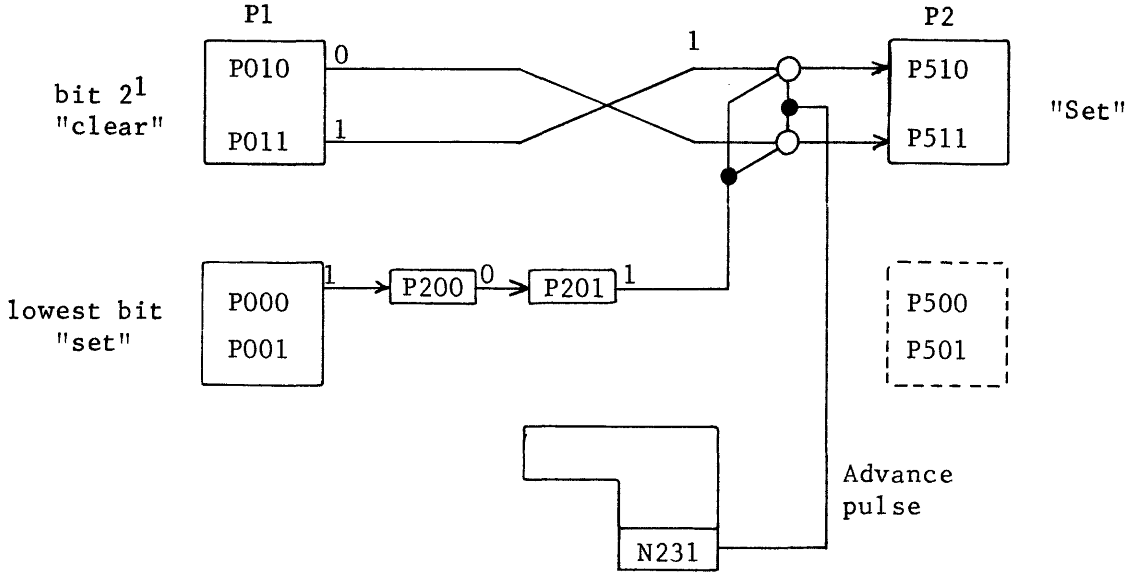


Figure 6-55 Illustration of rule 2

As you can see in Figure 6-55, if the lowest bit of P^1 had been cleared, P201 would output a logical zero and neither AND gate into P510/511 could have been made. In that case the flip-flop does not change from its previous state.

3) If the two losest bits are set, complement bit 2^2...
   Refer to Figures 6-54 and 6-55 for illustration of rule 3.

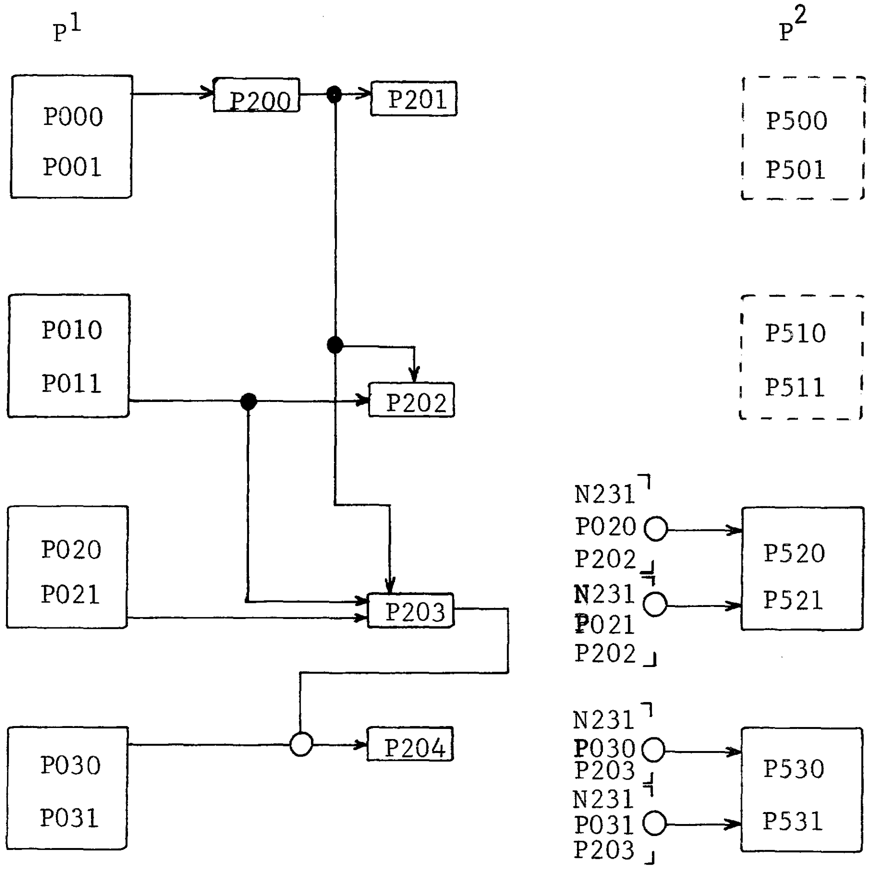


Figure 6-56 Group 0

Before we try to make the AND gates into P520/521 and P530/531, let's examine the P20X inverters to see when they would have ones out. Remember, an inverter with ORed inputs must have all logical zeros in to produce a logical "1" output. We have already agreed that P200 will have a logical zero out if the flip-flop P000/001 is set. Because this term feeds both P202 and P203, it's obvious that the lowest bit (P000/001) must be set before those inverters could output a one.

When would P202 output a one? Only if both the two lower bits are set.
When would P203 output a logical "1"? All inputs would be zero only if all three lower bits were set.

Let's make a table to tabulate our findings of when the dismissed terms would have ones out. We'll talk about P204 a little later.

P200	lowest bit (2^0) cleared
P201	lowest bit (2^0) set
P202	bits 2^0 and 2^1 both set
P203	bits 2^0, 2^1, and 2^2 all set

Now look at the inputs to P520/521. Which term is different on the two AND gates? Right, P020 and P021 are from the same bit position of P^1. Let's redraw the flip flop and label the inputs. (Figure 6-57)

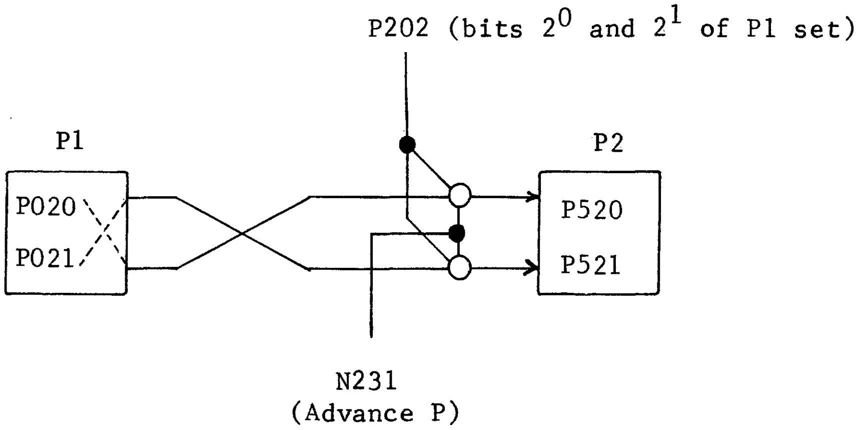


Figure 6-57 Rule 3

Remember that the set side output of flip-flop P020/021 is actually fed by inverter P021. If that doesn't seem correct to you, go back to Figure 6-39 and refresh your memory.

Can you verify rule 3 from Figure 6-57? If the lower two bits of P^1 are both set (P202), complement bit 2^2 of P^1 into P^2 (forced complement transfer gated by "advance P" pulse).

You should now be able to validate rule 4 by examining Figures 6-56 and 6-58.

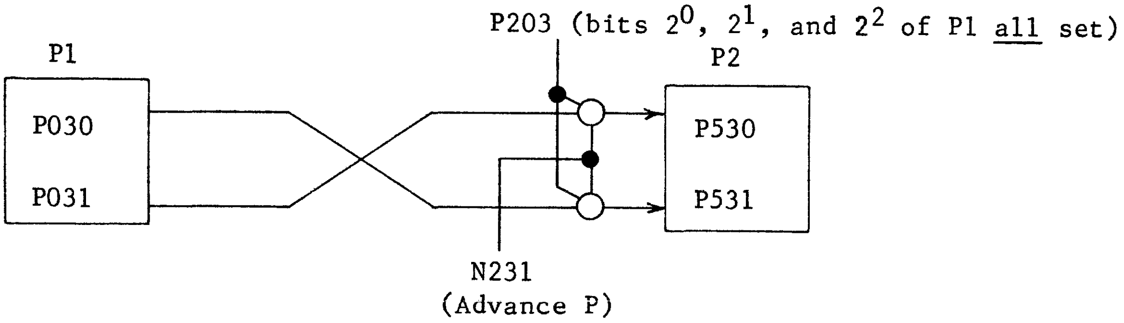


Figure 6-58 Rule 4

Now let's see if those rules work. Assume a count of 0101 (5_8) in the lower 4 bits and see what happens when the Advance P pulse is applied.

1) lowest bit goes "cleared" according to rule 1.

2) bit 2^1 goes "set" as per rule 2.

3) bit 2^0 remains a "1" because neither gate into P520/521 can be made.

4) bit 2^3 remains a "0" because neither gate into P530/531 can be made.

The lowest four bits are now 0110. Well, Lo and Behold! The P register contained a 5 and, after advancement, it now contains a 6. Will miracles never cease?

Go ahead and try a few more if you're still suspicious.

Convinced? Good! Now let <u>us</u> advance and try the group advancement procedures.

Group Advancement Rules:

1) Group 0 is advanced unconditionally in accordance with the stage advance rules upon receipt of the advance pulse.

2) Group 1 receives an advance pulse <u>only</u> if all bits of Group 0 are set (logical ones). Stage advance rules apply within the group if the advance pulse is received.

3) Group 2 receives an advance pulse <u>only</u> if all bits in both Group 0 and Group 1 are set.

4) Group 3 receives an advance pulse <u>only</u> if all lower twelve bits (Groups 0, 1, and 2)are set.

Let's examine the Advance P pulse for Group 1 to see when it will be present. The stages within Group 1 will be advanced exactly as we advanced Group 0 except that the bit positions are different. The problem, before we can complement bit 2^4 of P^1 into P^2, is to get the advance pulse.

Back on Figure 6-56, there was an inverter (P204) that we failed to mention. We were looking for a logical "1" out of the other inverters, but we need a logical "0" out of P204. To get a zero out, the AND gate must be made. When would P204 output a zero? (Figure 6-57)

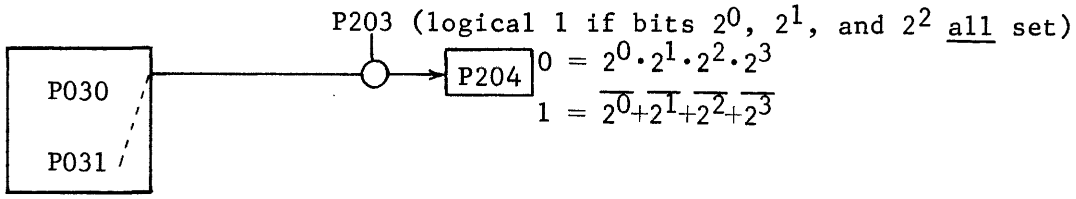

P203 (logical 1 if bits 2^0, 2^1, and 2^2 all set)
P030
P031
P204
$0 = 2^0 \cdot 2^1 \cdot 2^2 \cdot 2^3$
$1 = \overline{2^0}+\overline{2^1}+\overline{2^2}+\overline{2^3}$

Figure 6-59 Group 1 Advance Enable

P031 will be a logical "1" if bit 2^3 is set and P203 will output a logical "1" only if the three lower bits are set. Therefore, the AND gate will be made and P204 will output a logical zero only if the lower four bits (Group 0) are all set.

Examination of Figure 6-60 will illustrate why we were interested in a logical "0" out of P204 instead of a logical "1".

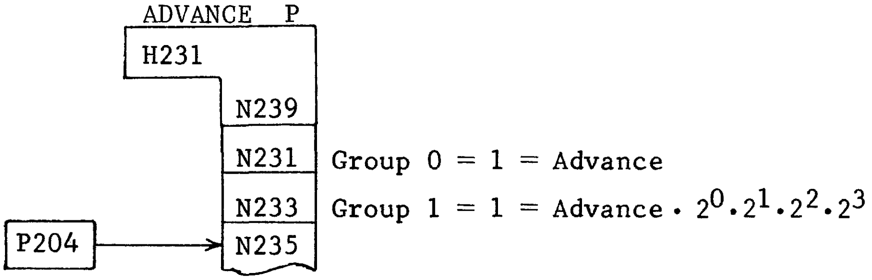

ADVANCE P
H231
N239
N231
N233
N235
P204
Group 0 = 1 = Advance
Group 1 = 1 = Advance $\cdot$ $2^0 \cdot 2^1 \cdot 2^2 \cdot 2^3$

Figure 6-60 Group 1 Advance Pulse

Although N233 is part of a central delay, it is still a standard inverter which needs all logical 0's in, to output a logical "1". If the output of P204 is a logical one (lower four bits not all set), it would prevent N233 from outputting a logical one even though N239 and N231 do.

Figure 6-61 illustrates the Advance P pulse for all groups and the following equalization pulse.

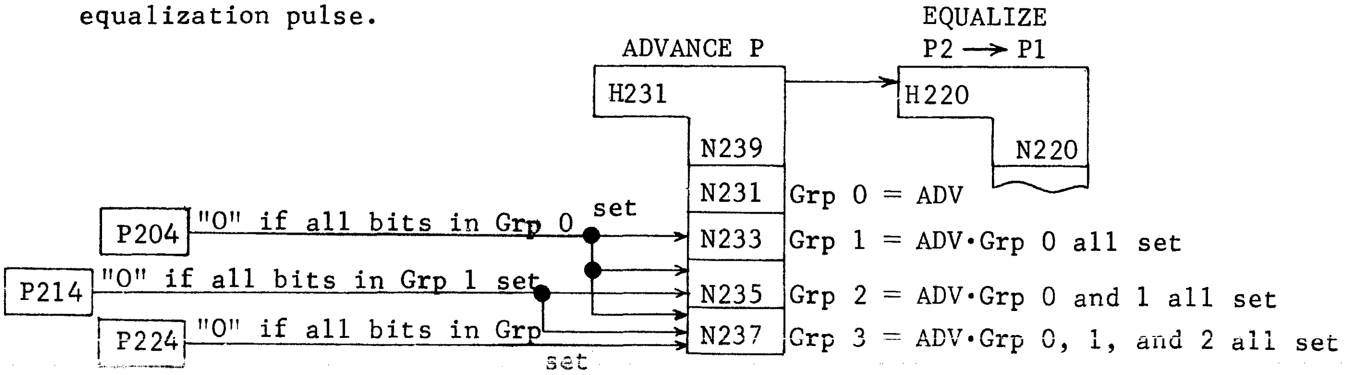

ADVANCE P
EQUALIZE
P2 → P1
H231
H220
N239
N220
N231
N233
N235
N237
Grp 0 = ADV
Grp 1 = ADV·Grp 0 all set
Grp 2 = ADV·Grp 0 and 1 all set
Grp 3 = ADV·Grp 0, 1, and 2 all set
P204
"0" if all bits in Grp 0 set
P214
"0" if all bits in Grp 1 set
P224
"0" if all bits in Grp set

Figure 6-61 P Register Advance Enables

Remember these few facts in reference to a double-rank counter:

1) All affected bits will be advanced instantaneously upon receipt of the advance pulse.

2) If a bit position is advanced, it will be the complement of its counterpart in the P^1 register (advance transfers are forced complement).

3) Unaffected bits retain the same state as before the advance pulse.

4) The equalize $P^2 \longrightarrow P^1$ pulse enables a forced transfer of all bit of the advanced P^2 register back into P^1 in preparation for the next advance pulse.

5) The contents of the P^1 register determines what count will be "advanced" into P^2.

The following examples indicate the initial contents of the equalized P^1 and P^2 registers. What will be the contents of P^2 after the advance pulse.

1. 12345 ____________________
2. 70707 ____________________
3. 00777 ____________________
4. 77770 ____________________
5. 77767 ____________________
6. 00000 ____________________
7. 77776 ____________________
8. If the P register were a ones complement counter, which answer(s) would be affected? How?
9. What would be the result if the P register were looking for cleared bits instead of set bits? Apply the same rules, but check for cleared flip-flops. What would be the new answers to problems 1 through 8? Check your answers against these at the end of the chapter.

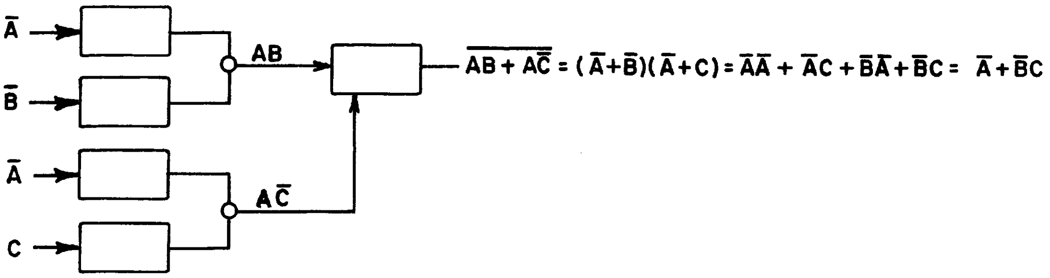
ANSWERS TO LOGIC EXERCISES

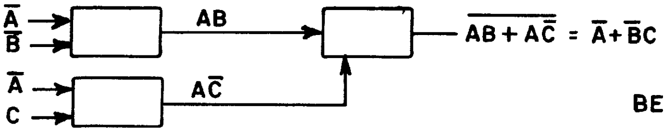
1. Switching time, logical voltage levels, physical size.

2. The standard logic voltage levels for all 3600 type circuits are -5.8v for a logical 1 and -1.1v for a logical 0.

3. System design, logical design, circuit design.

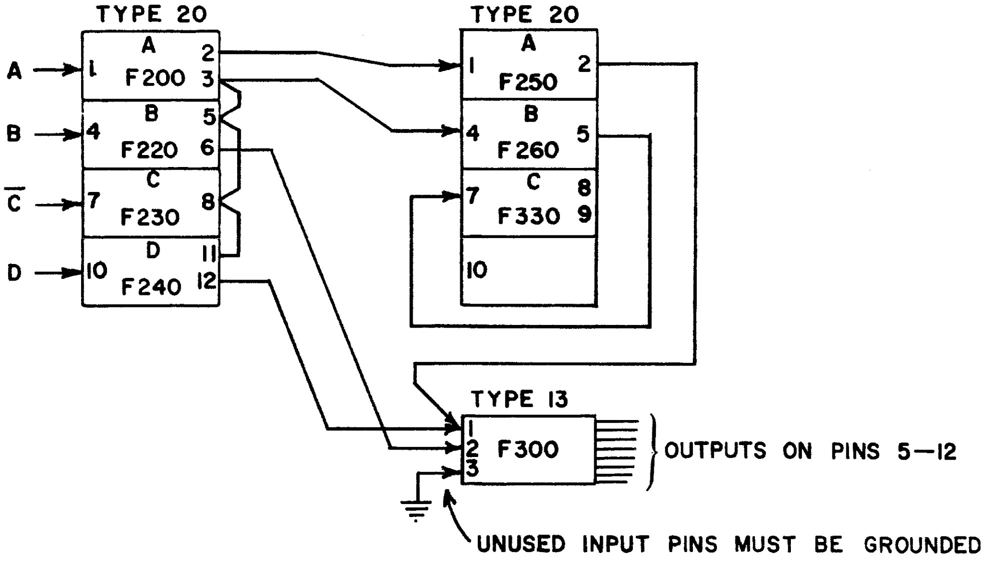
4.


$\bar{A}$
$\bar{B}$
$\bar{A}$
C
AB
$A\bar{C}$
$\overline{AB+A\bar{C}} = (\bar{A}+\bar{B})(\bar{A}+C) = \bar{A}\bar{A}+\bar{A}C+\bar{B}\bar{A}+\bar{B}C = \bar{A}+\bar{B}C$



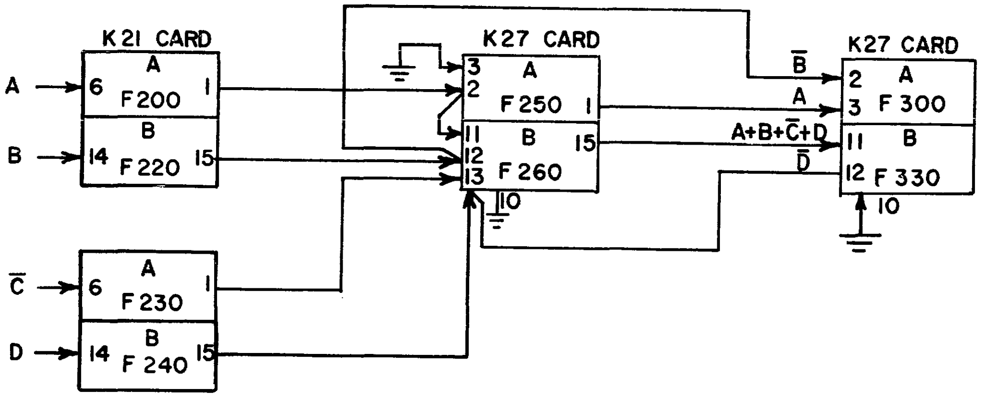
BEST (FEWEST CARDS)

5.



1604 Logic Wiring Connections

6.



3600 logic circuit wiring connections

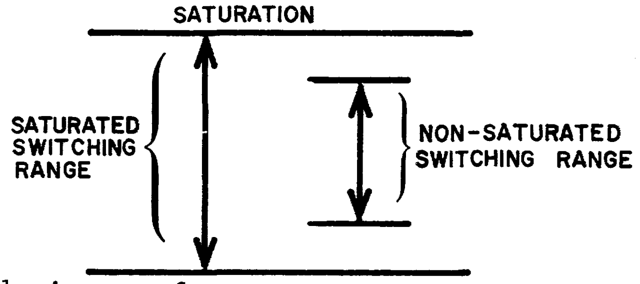
All open input pins produce a logical "1" input. Therefore, at least one pin must be grannded if a complete AND gate is unused. However, a single input to a four-way AND gate would control that AND gate if the other three pins are open (logical ones). For example, F300 above,is on a K27 card. Pin 3 has the input and pins 4, 5, and 6 are left open.

7. Both inputs would be a logical zero. If the set input is removed first, the flip flop would go cleared; set if the clear input is removed first. If both are removed simultaneously, the state would be unpredictable (depends upon which inverter switches faster).

8. Unpredictable. However, most computers have a Power On Master Clear which clears most registers and control flip-flops.

9. The control delay would output a constant zero because the open input pin is a logical one.

10. Saturated logic switches all the way from cut-off to saturation but non-saturated logic is always conducting between above cut-off and below saturation due to the bias network.



Saturated logic uses fewer components but, unless premium transistors are used, usually requires more time to switch.

Non-saturated logic switches faster because it switches over a smaller percentage of the overall range. However, the bias feedback components increase the cost of the circuit.

ANSWERS TO P REGISTER EXERCISES

1. 12346

2. 70710

3. 01000

4. 77771

5. 77770

6. 00001

7. 77777

8. 7, 00000

9. decrementing network instead of incrementing

    1) 12344

    2) 70706

    3) 00776

    4) 77767

    5) 77766

    6) 77777

    7) 77775

    8) 6, 77776

SUMMARY

This chapter has discussed two aspects of logical design. The first section of the chapter illustrates how an electronic design can be applied to computers in consideration of basic speed and economics. An important point here is that the circuit must be competitive and profitable. The second section discussed logical design of networks. Letters and numbers were used to show logical significance.

This chapter is only a brief outline of a selected logic. Remember that a company that has one hundred design engineers will very likely be able to develop one hundred different methods of designing and building logic. If you remember this point and are receptive to new ideas, it will be easy to understand new and different logical methods.

This chapter has explained the function of several types of inverter circuits. Whether the inverter is of the 1604 family, the 3600 family, or the integrated logic family, it can still be used to form the other two building blocks of a CDC computer. Those three building blocks, with a variety of input combinations, provide the design engineer with the latitude necessary to design the majority of computer logic circuits. Other special circuits are also required, but will be discussed as the need arises.

Two inverters, with feedback, make a flip-flop and the flip-flops are used to form the registers. Three inverters, properly connected and with clock inputs, make a control delay while a string of control delays can be used to construct a timing chain. With those capabilities, we can proceed and build a computer. The next chapter discusses the Arithmetic Section. You will discover that the entire adder is merely a group of interconnected inverters that generate the sum of two operands.

chapter VII

Arithmetic Section

Arithmetic

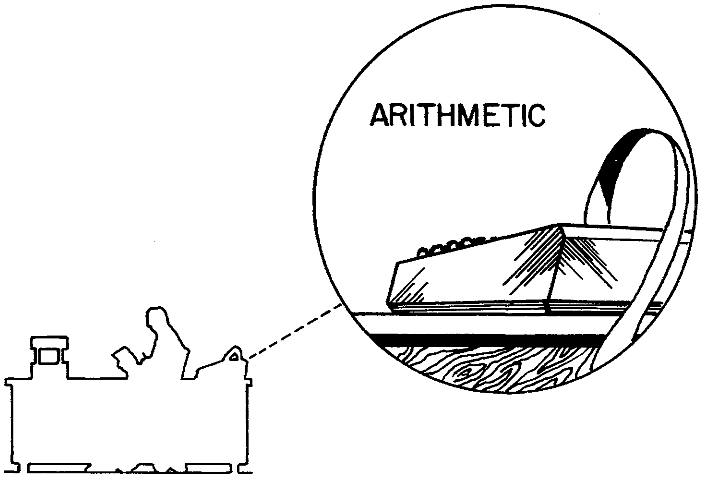
CHAPTER VII

ARITHMETIC SECTION OF A COMPUTER

INTRODUCTION

You learned in earlier chapters that a digital computer consists of five main sections -- each necessary for its operation. This chapter explains the function of one of them, the Arithmetic Section. The other sections of a computer are equally important; but, because you can already see the need for computations, the Arithmetic Section will be discussed first.

The comparison is again made between a computer and a man, perhaps you, sitting at a desk.



The adding machine performs the actual arithmetic operations for the man, synonymous with the arithmetic section for the computer.

The man could do his calculations by recording the numbers (operands) and then performing the desired calculations in his head. This method would soon become laborious, would require more time, and would be subject to human error. Although the increased speed and accuracy that the adding machine affords is obvious, there are some other items you may have overlooked. When the man pushes a numbered key on the adding machine, he is, in fact, entering his information in a serial fashion, i.e., one digit follows another. When he pushes the add key, the adding machine adds the number to the other numbers that are in the mechanical wheels and cogs of the machine all at once, in parallel fashion.

Computers can operate in either fashion, serial (a digit at a time) or parallel (many digits at once). Some computers have the ability to be programmed to select the type of arithmetic operation that is desired.

A serial computer requires fewer circuits but a parallel computer permits an appreciable speed increase. Because increased computer applications demand faster computers, the parallel computer is the predominant type in use today.

The Arithmetic Section of the computer performs several functions in addition to the four basic arithmetic operations. Operands may be checked for positive or negative sign, overflow conditions are monitored, comparisons may be made between two operands, and arithmetic registers may be examined for contents equal to zero.

The four basic arithmetic operations may all be accomplished by adding only, or subtracting only. The objective of this chapter is to familiarize you with the theory and operation of a computer Arithmetic Section. Figure 7-1 illustrates the arithmetic section of the basic computer.

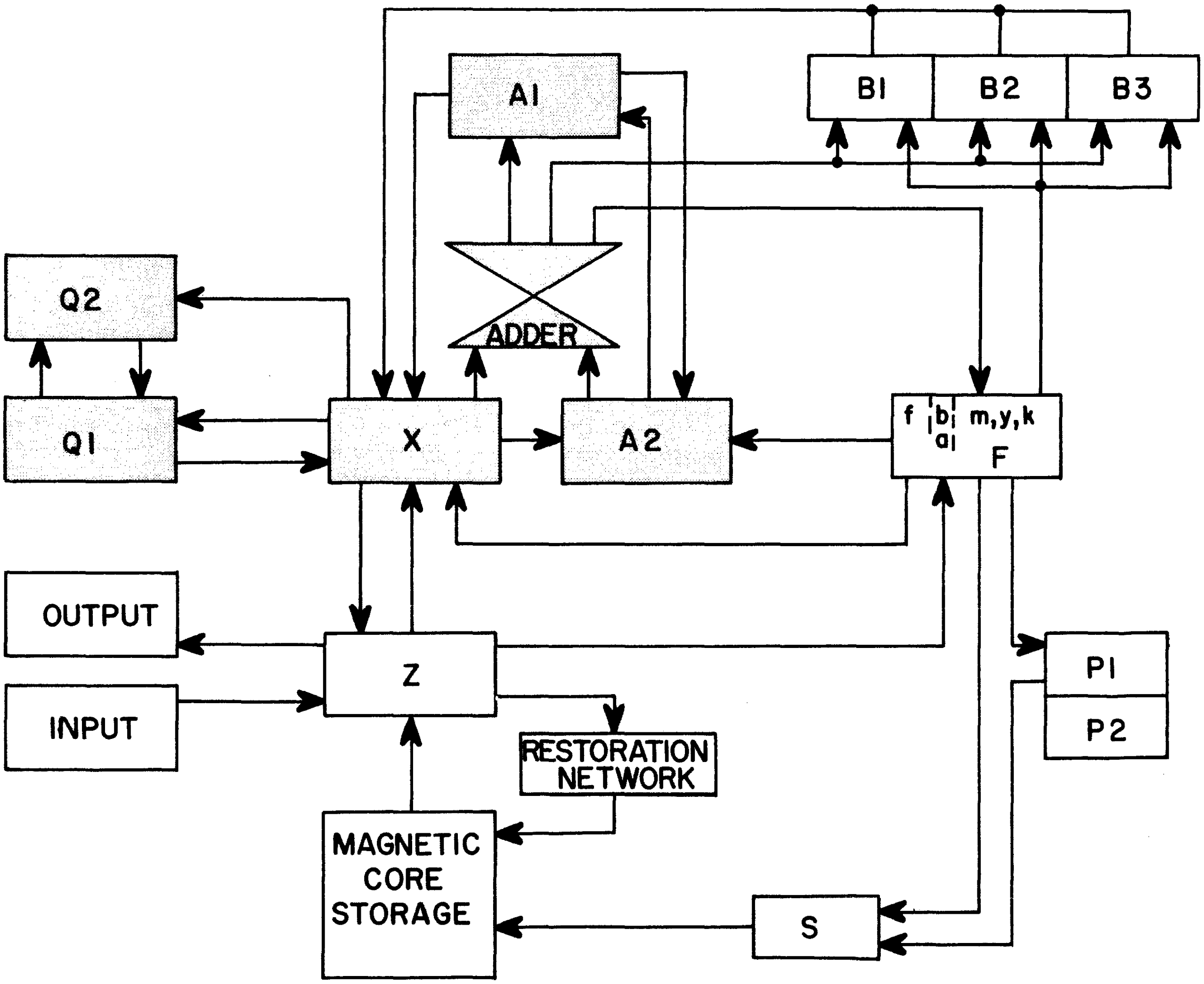


Figure 7-1. Basic Computer Block Diagram

REVIEW OF LOGIC

The circuitry used in the Arithmetic Section has already been introduced in Chapter VI. The three basic circuits are the inverter, the flip-flop and the control delay. Of these three, the inverter and the flip-flop are used extensively in the design of the Arithmetic Section. Either of them may have AND and OR inputs to produce the desired output. The design engineer determines what outputs are required and then designs the circuit to meet those requirements.

Figure 7-2 illustrates the inverter and flip-flop with AND and OR inputs.

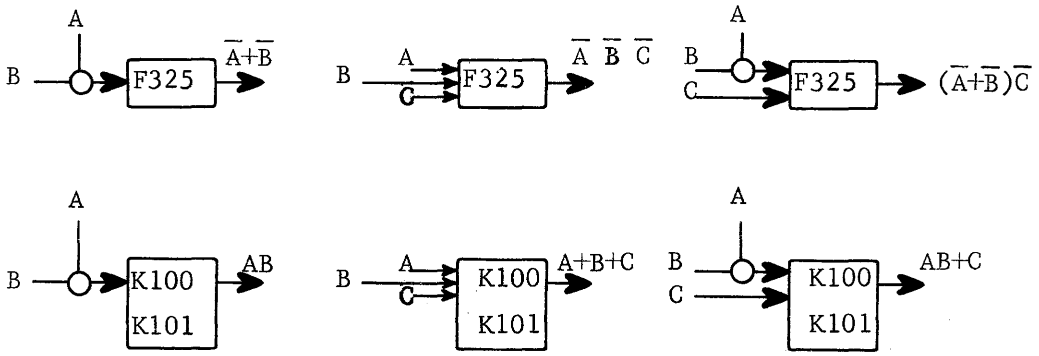

A
B
F325
$\overline{A}+\overline{B}$
B
A
C
F325
$\overline{A}\ \overline{B}\ \overline{C}$
A
B
C
F325
$(\overline{A}+\overline{B})\overline{C}$
A
B
K100
K101
AB
B
A
C
K100
K101
A+B+C
A
B
C
K100
K101
AB+C

Figure 7-2. Logic Review

The Boolean output equation for each circuit defines the inputs required to produce a logical "1" out of the circuit. Because of the complementing effect of the inverter, the input equation must be negated and simplified to provide the output equation.

The flip-flop is a storage device and does not invert between input and output. Therefore, the set side output equation is the same as the set side input equation.

FUNCTIONS OF ARITHMETIC SECTION

SIGN SENSING

One of the instructions you learned in Chapter IV was the AZJ or A Jump. The j designator (0-3) determines what conditions are being monitored to determine whether or not the jump will be made. The AZJ,GE instruction compares the (A) with positive zero and, if equal to or greater than positive zero, the jump is made. If the sign bit of the A Register is a logical "1", it indicates a negative operand which would be less than zero (no jump). However, if the operand is positive zero or some positive value, the sign bit of the A Register would be a logical "0" (jump).

The uppermost bit (sign bit) of the A Register would then determine whether the jump would be made when executing either an AZJ,GE or AZJ,LT instruction. Figure 7-3 illustrates the sensing network.

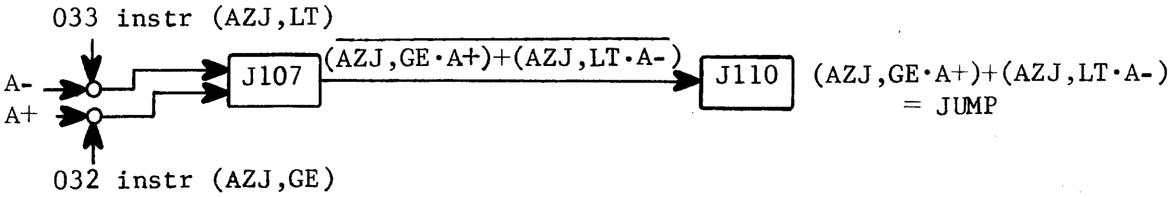

033 instr (AZJ,LT)
A-
A+
J107
$\overline{(AZJ,GE \cdot A+)+(AZJ,LT \cdot A-)}$
J110
(AZJ,GE·A+)+(AZJ,LT·A-) = JUMP
032 instr (AZJ,GE)

Figure 7-3. "A" Register Sensing Network

An AND gate into J107 will be made only if either an AZJ,GE instruction is being executed AND the A Register sign is positive OR an AZJ,LT instruction is being executed and the sign of A is negative.

In either case, J110 will output a logical "1" and will indicate that jump conditions have been met.

REGISTER SENSING

The AZJ,EQ and AZJ,NE instructions compare the entire contents of the A Register with zero. For these two instructions, positive zero is considered equal to negative zero. Therefore, before jump conditions can be met for the AZJ,EQ instruction, the A Register must contain either 0000000_8 or 77777777_8 (positive or negative zero). Figure 7-4 illustrates how both + and - zero are considered to be equal to zero.

The output of L077 (Figure 7-4) indicates that the contents of the A1 Register are equal to either positive or negative zero. This term feeds into the same circuit used for bit sensing.

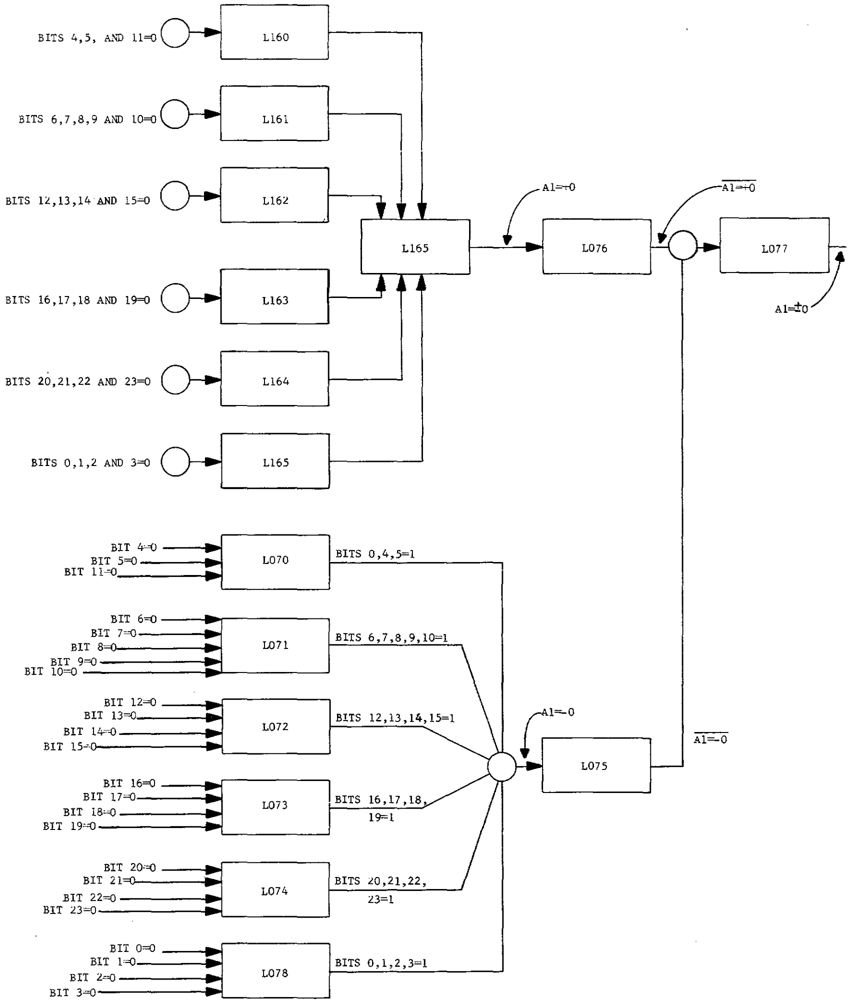


Figure 7-4. A = 0 Sensing Network

Figure 7-5 indicates the addition to the "Jump conditions satisfied" circuit.

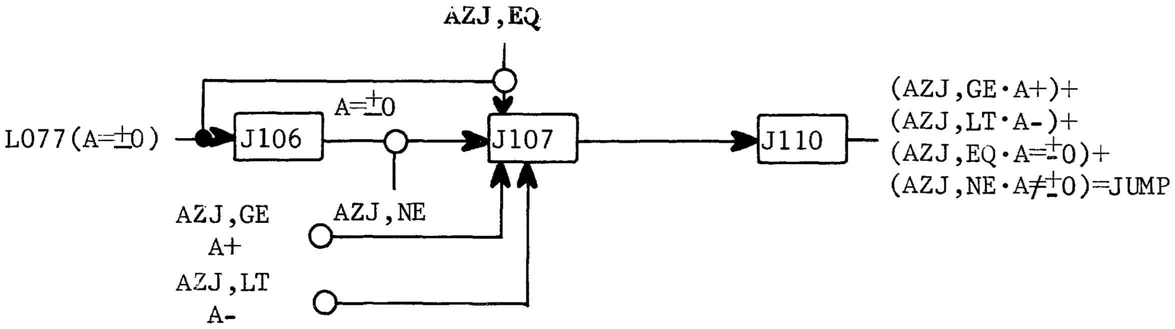


Figure 7-5. A Register Sensing Network

The output of J110, as shown, indicates that one of the AZJ instructions is being executed and that the sensed condition is present. J110 would enable the jump to occur by feeding an AND gate in the Control Section.

Answer the following questions and then check your answers with those at the end of the chapter.

1. What is the input equation for J107?

2. What is the simplified output equation for J107?

3. Generate the output equation for J110 by placing a NOT bar across the input equation and simplifying. Does your equation agree with the one in figure 7-5?

4. Why is the input equation to J107 equal to the output equation for J110?

Figures 7-3, 7-4, and 7-5 illustrate how the computer senses jump conditions when executing an AZJ instruction. Other registers contents are also sensed, but the methods of sensing remain essentially the same.

OVERFLOW

The conditions that constitute overflow were explained in Chapter III. If two operands of like signs are added and sum is of the opposite sign, overflow has occurred and the answer is incorrect. If two operands of unlike signs are subtracted and difference is the same sign as that of the subtrahend, overflow has occurred and the answer is incorrect.

ADD		SUBTRACT	
+	-	+	-
+	-	-	+
-	+	-	+

OVERFLOW CONDITIONS

Overflow does not indicate that the computer made a mistake. The programmer made the mistake of using operands whose sum or difference exceeds the modulus of the machine. (What happens when the filling station attendant tries to ADD that last three cents worth of gasoline to your tank when it's already full? OVERFLOW!) The computer informs the programmer of his mistake when an overflow condition occurs.

Overflow is an undesirable side effect common to any device that uses modulus -1 arithmetic (complement notation).

Figure 7-6 illustrates how the computer checks for overflow each time an ADA is performed. The same circuit is also used for the SBA instruction but those inputs are not shown.

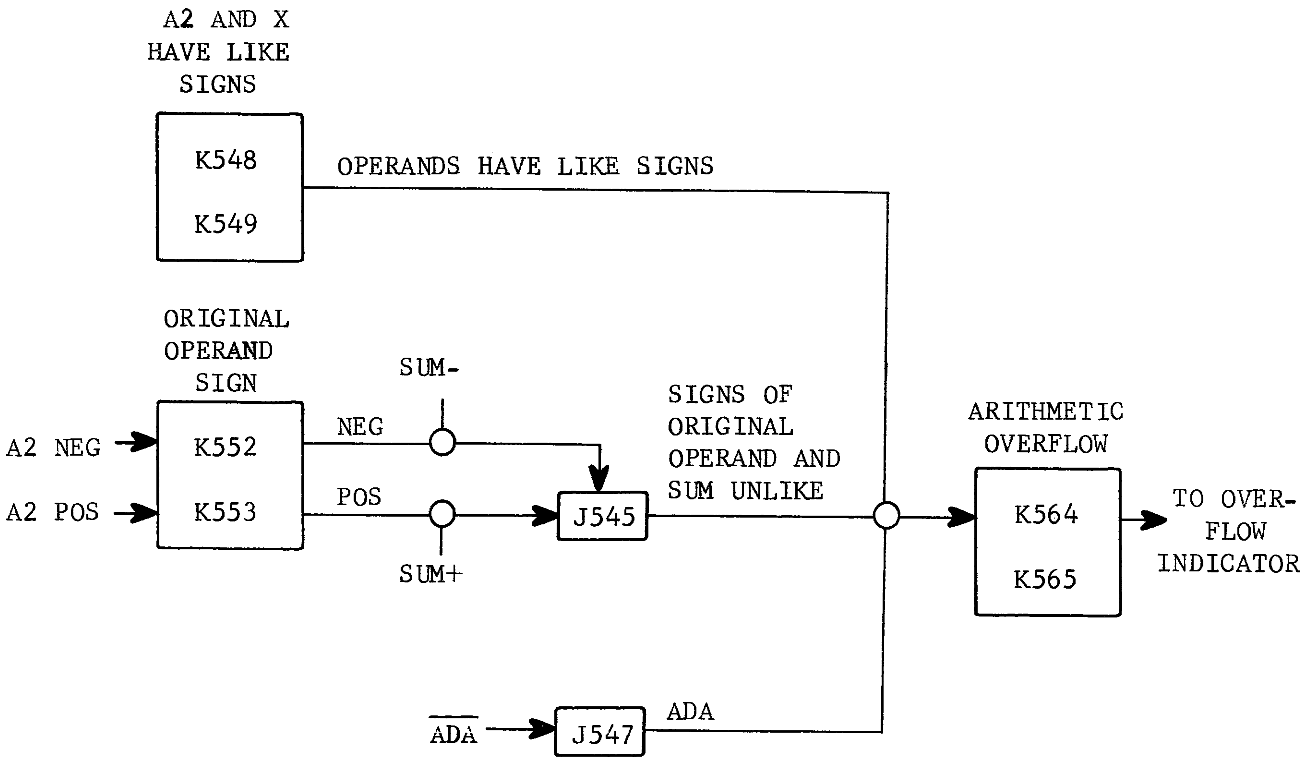


Figure 7-6. Arithmetic Overflow

The two operands are added by placing one in the X1 Register and the other in the A2 Register, both of which feed the Adder. The sum is fed from the Adder to the A1 Register and back to A2. A sign change in A2 produces a logical "1" out of J545 (Figure 7-6).

The Overflow FF will be set if there is a sign change in the A2 Register and the two original operands had like signs and an ADA instruction is being performed. Normal program operation will be interrupted before the next instruction is executed because of the incorrect answer.

Notice that register contents were not considered, only the signs of the operands and the sign of the sum.

Questions:

5. Does overflow indicate that end around carry is necessary?
6. Could the answer be correct even though overflow has occurred? Explain.
7. Why is it not possible to get overflow while executing a multiply instruction?
8. What is the relationship between overflow and end around carry?

ARITHMETIC OPERATIONS

Arithmetic operations of a computer may be accomplished in either serial or parallel mode, and may also be accomplished either by adding only or by subtracting only. If all operations are performed by subtraction, the computer is said to have a subtractive adder; if by addition, the computer is said to have an additive adder.

A theoretical computer, the "Binary Bit Buster" has a subtractive adder. Let's see how it adds two operands by subtracting.

First, by conventional methods, add the following:

$$\begin{array}{lll} & 000\ 010\ 100\ 110_2 = & 0246_8 \\ \text{ADD} & 001\ 011\ 101\ 111_2 = & 1357_8 \\ \hline & 001\ 110\ 010\ 101_2 & 1625_8 \end{array}$$

Now form the ones complement of the addend and subtract (the same way the "Binary Bit Buster" adds).

$001\ 011\ 101\ 111_2$ complemented equals $110\ 100\ 010\ 000_2$

$$\begin{array}{lrl} & 000\ 010\ 100\ 110_2 & \\ \text{SUBTRACT} & 110\ 100\ 010\ 000_2 & \longleftarrow \text{original addend complemented} \\ \hline & 001\ 110\ 010\ 110 & \\ & 1 & \longleftarrow \text{End around Borrow} \\ \hline & 001\ 110\ 010\ 101 & = 1625_8 \end{array}$$

The sum of the two operands (added) was 1625_8, but the same answer was also achieved by complementing and subtracting -- the same method used by the "B B B".

A descendent of "B B B" has an additive adder which also performs all four basic arithmetic operations. Let's see how "B B B Junior" performs these operations. First, by conventional methods:

$$\begin{array}{lrcl} \text{SUBTRACT} & 000\ 010\ 100\ 110_2 & = & 0246_8 \\ - & 001\ 011\ 101\ 111_2 & = & 1357_8 \\ \hline & 110\ 110\ 110\ 111 & & 6667 \\ & 1 & & 1 \longleftarrow \text{End around borrow} \\ \hline & 110\ 110\ 110\ 110 & & 6666_8 \end{array}$$

Now subtract by complementing the subtrahend and adding.

$$\begin{array}{lrl} \text{ADD} & 000\ 010\ 100\ 110 & \\ + & \underline{110\ 100\ 010\ 000} & \longleftarrow \text{subtrahend complemented} \\ & 110\ 110\ 110\ 110_2 & = 6666_8 \end{array}$$

Just lucky, or does it really work? If the end around borrow confuses you, do an Unconditional Jump back to Chapter III and review.

Multiplication is performed by "B B B Junior" by doing a series of adds. For example, binary multiplication is normally performed as follows (let's use two positive six bit operands):

$$\begin{array}{lrcr} & 011\ 101 & = & 35_8 \\ \text{X} & 010\ 101 & = & \underline{25_8} \\ \hline & 011\ 101 & & 221 \\ & 01\ 110\ 1 & & \underline{72} \\ & 0\ 111\ 01 & & 1141_8 \\ \hline & 1\ 001\ 100\ 001 & & \end{array}$$

The conventional method is to form all of the partial products and then ADD them together. However, the computer can only ADD two numbers at a given time. It achieves the same result in the following manner:

$$\begin{array}{lrl} & 011\ 101 = 35_8 & \\ & 010\ 101 = 25_8 & \\ \hline & 011\ 101 & \text{1st partial product (multiplicand)} \\ & 01\ 110\ 1 & \text{multiplicand left shifted under multiplier bit} \\ \text{ADD} & \underline{\qquad\qquad} & \\ & 10\ 010\ 001 & \text{2nd partial product} \\ & 0\ 111\ 01 & \text{multiplicand left shifted under next} \\ \text{ADD} & \underline{\qquad\qquad} & \text{multiplier bit} \\ & 1\ 001\ 100\ 001 & \text{final product} = 1141_8 \end{array}$$

The product is the same in both examples, only the sequence of operations was changed.

The computer always multiplies using positive operands. Both are checked and, if negative, are complemented to magnitude format. If both are negative or both positive, the product will be positive. If the original signs are unlike, the product will be complemented to make the answer correct (negative).

It seems logical that if multiplication can be performed by a series of add operations, division is possible by performing a series of subtract operations.

You should now be convinced that all four arithmetic operations can be performed in either an additive or subtractive device. Let's assume that the computer being studied has an additive adder.

Two operands, $+27_8$ and -15_8, are being added in a modulus 2^6-1 device; they would be expressed in binary as

$$\begin{array}{lrl} & 010\ 100 & = +24 \\ & \underline{110\ 010} & = -15 \\ & 000\ 110 & \\ & \underline{1} & \longleftarrow \text{End around carry} \\ \text{and the sum would be} & 000\ 111 & = +07 \end{array}$$

How many stages generated a carry? How many stages had carries coming in from the next lower stage? What bit combination generated the carry? What bit combination stopped or satisfied the carry propagation? Which combination could pass the carry to the next bit and not satisfy it?

One carry was generated in the second stage from the left (bit 2^4) but it was propagated to the upper stage (2^5) and end around to the lower stage (2^0). A "1" and "1" combination generated the carry which was satisfied by the "0" and "0" combination (bit 2^0). The upper stage was a "1" and "0" combination which passed the generated carry on to the next stage (2^0 or EAC).

Add the operands in the following examples. Label each stage that generates a carry with a G, each stage that could satisfy a carry with an S, and the passes (all others) with a P. Draw a small arrow over each stage that has a carry coming into it. Check your solutions with those at the end of the chapter. Indicate if overflow occurs.

All examples are modulus 2^6-1.

$$\begin{array}{rr} 9) & 000\ 001 \\ & +\ \underline{111\ 111} \end{array} \qquad \begin{array}{rr} 10) & 101\ 010 \\ & +\ \underline{011\ 011} \end{array}$$

$$\begin{array}{rr} 11) & 000\ 000 \\ & +\ \underline{000\ 000} \end{array} \qquad \begin{array}{rr} 12) & 111\ 111 \\ & +\ \underline{111\ 111} \end{array}$$

$$\begin{array}{rr} 13) & 111\ 110 \\ & +\ \underline{011\ 110} \end{array} \qquad \begin{array}{rr} 14) & 101\ 010 \\ & +\ \underline{101\ 010} \end{array}$$

Now subtract the operands in the following examples by complementing and adding. Use the same method as before and check for overflow.

$$\begin{array}{lr} 15) & 000\ 100 \\ & -\ 111\ 110 \\ \hline \end{array} \qquad \begin{array}{lr} 16) & 101\ 010 \\ & -\ 011\ 011 \\ \hline \end{array}$$

$$\begin{array}{lr} 17) & 100\ 111 \\ & -\ 111\ 000 \\ \hline \end{array} \qquad \begin{array}{lr} 18) & 011\ 101 \\ & -\ 011\ 101 \\ \hline \end{array}$$

$$\begin{array}{lr} 19) & 111\ 000 \\ & -\ 110\ 111 \\ \hline \end{array} \qquad \begin{array}{lr} 20) & 000\ 100 \\ & -\ 110\ 000 \\ \hline \end{array}$$

You are now able to add and to determine what bit combinations of the original operands are classified as generates, satisfies, and passes. Let's examine the results of the foregoing problems and learn another way to form the sum.

The operands used in problem #9 were:

$$\begin{array}{rcl} 000\ 001 & = & +01 \\ 111\ 111 & = & -00 \\ \hline \end{array}$$

and the respective stages were classified as:

P P P P P G.

Each stage had a carry propagated into it because there was no stage that could stop or satisfy the carry.

Let's do a stage-by-stage (bit-by-bit) half add of the two operands. A half add is similar to a full add except that all carries are disregarded.

$$\begin{array}{r} \overset{\swarrow}{0}\overset{\swarrow}{0}\overset{\swarrow}{0}\ \overset{\swarrow}{0}\overset{\swarrow}{0}\overset{\swarrow}{1} \\ 111\ 111 \\ \hline 111\ 110 \end{array}$$

Now complement each answer bit that had a carry coming into it (arrow).

Because every stage had a carry input, each bit is complemented and the answer is 000 001 = $+01_8$. Wasn't that the correct answer for problem #9?

You gotta be kidding! Bet it won't work the next time!

Well, let's see if it will. The operands for problem #10 were:

$$\begin{array}{rcl} \overset{\swarrow}{P}\overset{\swarrow}{P}G\ \overset{\swarrow}{S}\overset{\swarrow}{G}\overset{\swarrow}{P} & & \\ 101\ 010 & = & -25 \\ +\ 011\ 011 & = & +33 \\ \hline \end{array}$$

The partial sum (result of half add) is 110 001. Complement all stages that have an incoming carry (all stages except 2^3). The final answer is:

$$\begin{array}{ll}
\overset{\downarrow}{1}\overset{\downarrow}{1}0\ \overset{\downarrow}{0}\overset{\downarrow}{0}\overset{\downarrow}{1} & \text{partial sum} \\
 & \text{complement bits with incoming carries (arrow)} \\
\cline{1-1}
000\ 110 & \text{final sum}
\end{array}$$

That answer also agrees with our previous solution. Problem #14 had overflow and consequently the wrong answer. Let's see if our new method also gives the same wrong answer.

$$\left.\begin{array}{rl}
101\ 010 & = -25 \\
101\ 010 & = -25 \\
\hline
010\ 100 & \\
1 & \\
\hline
010\ 101 & = +25
\end{array}\right\} \quad = \quad
\begin{array}{rl}
\overset{\curvearrowright}{G\overset{\downarrow}{S}G\ \overset{\downarrow}{S}GS} & \\
101\ 010 & \\
101\ 010 & \\
\hline
000\ 000 & \text{partial sum} \\
1\ \ \ 1\ 1 & \text{complement carry inputs} \\
\hline
010\ 101 &
\end{array}$$

By golly, the answer may be incorrect (overflow) but at least the method still works...and works...and works...every time.

Perhaps you also noticed that the partial add answer bit is a "1" if that stage was a pass and a "0" if that stage was either a generate or a satisfy (pass).

The boolean expression for ones out of the adder would therefore be PASS AND $\overline{\text{CARRY INPUT}}$ or, $\overline{\text{PASS}}$ AND CARRY INPUT. If neither of the two conditions are met, that stage of the adder would output a zero.

Figure 7-7 illustrates how our adder would look in simplified form.

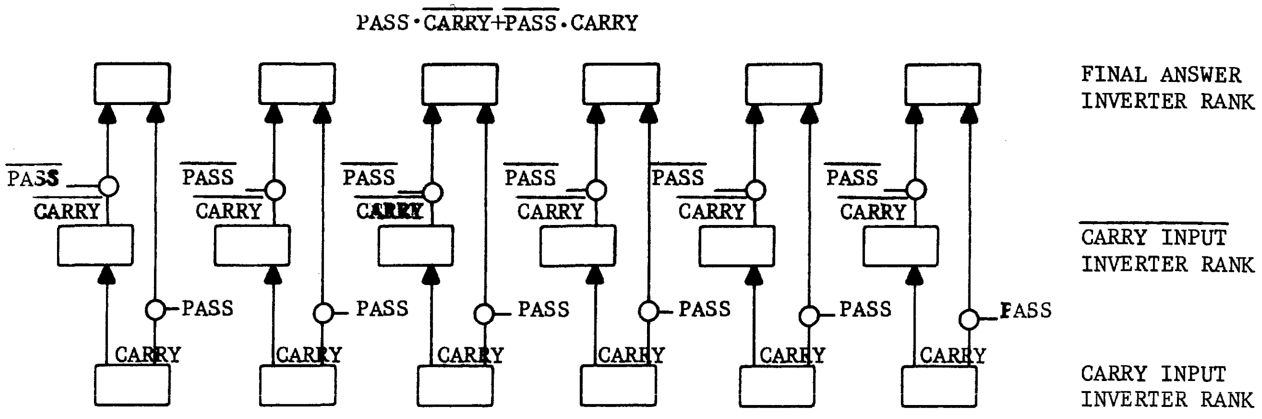

PASS·$\overline{\text{CARRY}}$+$\overline{\text{PASS}}$·CARRY
FINAL ANSWER INVERTER RANK
$\overline{\text{PASS}}$
$\overline{\text{CARRY}}$
$\overline{\text{CARRY INPUT}}$ INVERTER RANK
PASS
CARRY
CARRY INPUT INVERTER RANK

Figure 7-7. Simplified Adder

There are now only two problems left to resolve:

1) form the partial sum of the two operands (PASS).
2) determine which stages have carry inputs (CARRY).

The partial sum is formed by applying the operands to an inverter rank in an Exclusive OR combination. The Exclusive OR was discussed and illustrated in Chapter V (Figures 5-8,9, and 10) but, as a matter of review, is again illustrated in Figure 7-8.

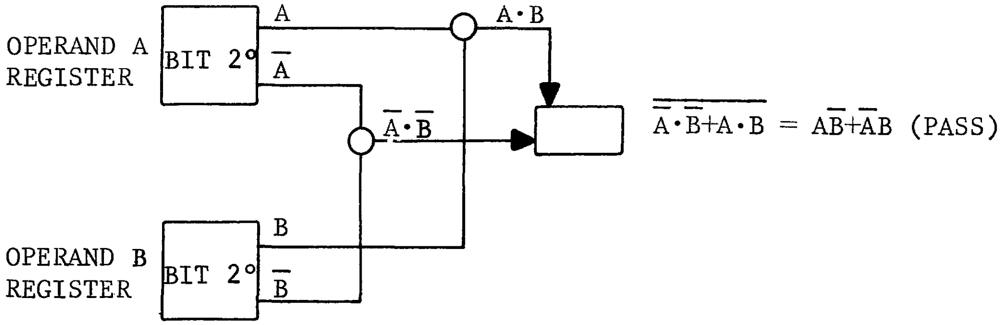

OPERAND A REGISTER
BIT 2^0
A
$\overline{A}$
$A \cdot B$
$\overline{A} \cdot \overline{B}$
$\overline{\overline{A} \cdot \overline{B} + A \cdot B} = A\overline{B}+\overline{A}B$ (PASS)
OPERAND B REGISTER
BIT 2^0
B
$\overline{B}$

Figure 7-8. Exclusive OR

The inputs to the inverter are different from those in Chapter V because the inverter "complements" the input equation. If a register is used for the partial add (half add), the inputs would be $A\overline{B}+\overline{A}B$ -- yielding the same equation as the output of the inverter rank.

The problem of performing the half add was relatively easy to solve. Now let's look at the remaining problem: How do we determine which stages have carry inputs? The only conceptual difference between a serial adder and a parallel adder is the method used to determine incoming carries. If the lowest bit is added first and then each successive bit added in sequence, considering carries, the method is serial. We add serially with pencil and paper because the human brain is serial. However, if each of six men considered only two half-added bits and an incoming carry, the sum could be formed and the method would be parallel.

SERIAL ADDER

The serial adder forms the sum of two operands as a sequence of events. Starting with the lowest bit-power and working upward, each bit of the answer is formed. Carry inputs are considered and interim answer bits are formed. If no end around carry exists, the interim answer is the final answer. If end around carry does exist, another complete pass is made and the final answer is formed. The following example illustrates the necessity of two passes if EAC exists.

```
ADD  1 0 1 0 1 0     CARRY from upper stage only (EAC)
     1 1 0 1 0 1
   ─────────────
  ┌─0 1 1 1 1 1      interim answer
  └──────────▶ 1     (end around carry now produces a carry
   ─────────────
     1 0 0 0 0 0     input to every stage.)
```

No more than two complete passes through the adder are ever necessary and one pass is sufficient if no end around carry exists. Figure 7-9 illustrates the complete logic necessary to form an answer bit and to generate a carry for the add operation.

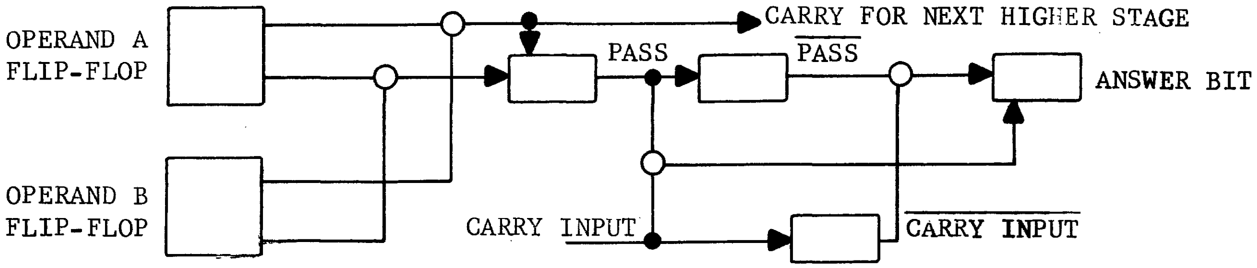


Figure 7-9. Serial Adder with Carry Generation

The same circuit (Figure 7-9) would be used for all bit positions. Each answer bit is gated to its respective position in an answer register. The two operands from the next higher bit position are then gated into the operand flip-flops and the second answer bit and carry bit are formed. The process repeats for all remaining bits to be added. If EAC exists, each bit position must again be examined to form the final answer bit and possibly generate a carry. The second pass is essentially adding the interim answer and carry inputs. If any stage generated a carry on the first pass, the interim answer bit would be a zero and could not generate a carry on the second pass.
For example:

```
 1 1 1 0
 0 0 1 1
─────────
 0 0 0 1    1st pass (interim answer)
       1
─────────
 0 0 1 0    2nd pass (final answer)
```

A carry is generated by forming the logical product (AND function) of the two operands. Figure 7-10 compares the logical product with those you have previously learned.

For the logical product, both operands must be ones (generate) before the carry is generated (Figure 7-9). A carry could also be generated by any stage if that stage was a pass AND it had a carry input.

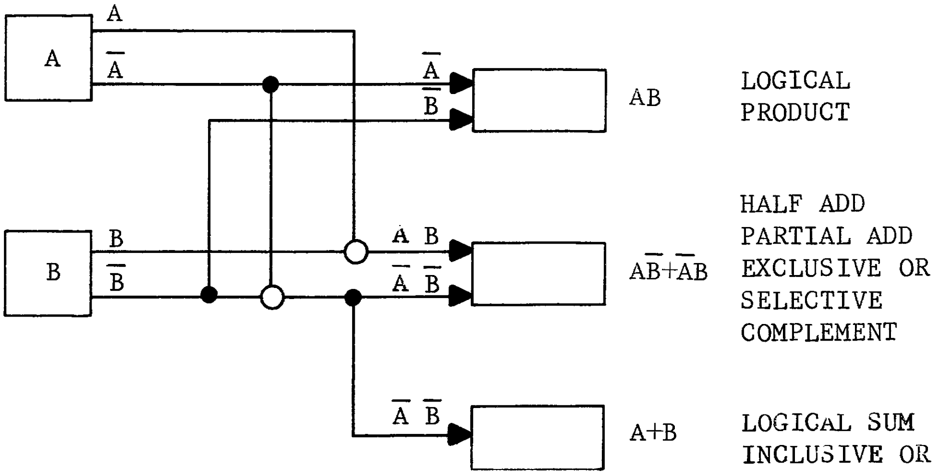


Figure 7-10. Logical Combinations

Design a carry generation network to include that possibility in the following space. Check your design with solution #21 at the end of the chapter. The boolean equation should be Generate + Pass • carry input.

Problem #21. Carry Generation Network:

Very little circuitry is involved in the serial adder. However, each bit must be formed in sequence and each stage of the operands must be examined twice if the answer produced an end-around-carry. In that case, twelve discrete time intervals would be required to form the answer in a 6-bit adder. For example:

Time 1 form interim answer and generate carry if necessary (bit 2^0)

Time 2 form interim answer, considering carry input, and generate carry if necessary (bit 2^1)

↓

Time 6 form interim answer, considering carry input, and generate EAC if upper stage is a generate OR a pass with a carry input (bit 2^5)

If EAC does not exist, the interim answer is the final answer.

If EAC does exist, times 7-12 are required.

Time 7 complement interim answer bit (2^0) and generate a carry if interim answer bit was a logical one.

Time 8 complement interim answer bit if there is a carry input and generate a carry if interim answer bit was a logical one.

↓

Time 12 complement interim answer bit (2^5) if there is a carry input.

The disadvantage of the serial adder, although not much logic circuitry is required, is the time needed to form the final answer. Each bit combination would require the following sequence:

1. Gate the operands into the adder flip-flops.

2. Form the interim or final sum and generate a carry if the stage is a generate.

3. Gate answer bit to answer register, carry to Holding Flip-flop for next stage.

4. Clear adder flip-flops before gating in next stage.

If EAC exists

5. Gate interim answer bit and carry to adder flip-flops.

6. Form final answer bit and generate carry if necessary.

7. Repeat steps 3 and 4.

A 24-bit adder would probably require approximately 6 to 12 microseconds to form the sum of two operands. The speed of a digital computer could be appreciably increased if the addition could be performed in parallel. Let's examine a parallel adder to see if the speed increase justifies the added (ones complement) cost of the additional circuitry.

PARALLEL ADDER

Parallel addition -- impossible you say. Remember the six men discussed earlier, each forming an answer bit simultaneously with the others. Each man only needs to know the two operand bits and whether or not a carry input exists. With this information alone, each man can form his final answer bit. For example,

```
ADD  0 1 1 0 1 0                            1 0 0 1 0 1<-- input carries
     1 1 0 0 1 0                            0 1 1 0 1 0
                                            1 1 0 0 1 0
    ____________        OR PARTIAL ADD
                                           ____________
     0 0 1 1 0 0                            0 0 1 1 0 1
               1
    ____________

     0 0 1 1 0 1
```

Notice that if a column contains an odd number of ones (1 or 3), the answer bit is a one. Any other combination (even number of ones or all zeros) would yield a zero answer bit.

The partial adding of the two answer bits presents no problem (we've done that before) but how do we determine if a stage has a carry input before the lower stages have been added. Interested in finding out? If so, continue.

Figure 7-11 illustrates the parallel adder in block diagram form. It is a portion of the same computer block diagram that has been and will continue to be used throughout the manual. Although you now know how a serial adder functions, our computer and the majority of modern digital computers in operation today have a parallel adder.

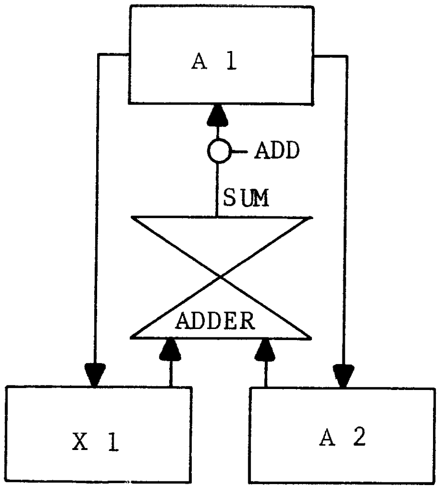


Figure 7-11. Parallel Adder

The operands are held in the X1 and A2 Registers while the answer is being formed. This is because the adder is comprised entirely of inverters and the inputs must be held static until a sum is formed. The output of the adder is the sum and feeds the A1 Register, the F Register, or one of the B Registers.

The adder is continually forming the sum of (X) and (A2). The add pulse (Figure 7-11) merely allows the answer to be gated from the adder to the A1 Register.

The symbol for the adder is in the form of an hourglass because the actual circuit configuration alludes to that symbol (see Figure 7-12). Keep in mind what functions the adder must actually perform.

1. determine which stages have carry inputs.

2. half add the two operand bits and the carry bit.

Now that you understand what the adder must accomplish to form the sum, let's analyze it and see how the adder performs its task. The modulus of the arithmetic registers determines the number of stages necessary in the parallel adder (2^{24} or 2^{24}-1 requires a 24-bit adder). The adder to be analyzed (Figure 7-12) contains only six stages. Those six stages could be expanded to 24, 48, or ? and still function exactly the same as the 6-bit adder. The adder would be expanded horizontally to accommodate larger operands but additional ranks would not be required.

Each rectangular box represents an inverter which is usually identified by a letter followed by three digits. However, the adder in Figure 7-12 is labeled with mnemonic codes, each code representing the conditions under which that inverter will output a logical one. The terms feeding the inverters are also mnemonic except those from the operand registers (A2 and X1).

Each bit position of the adder constitutes a stage (2^0-2^5) and three stages constitute a group. Group 0 (stages 2^0-2^2)is on the right side of the diagram and Group 1 (stages 2^3-2^5) is on the left.

If a mnemonic code begins with an S, it defines a condition relative to a single Stage. The number in the code defines the bit-power of that stage and the remainder of the code defines a condition.

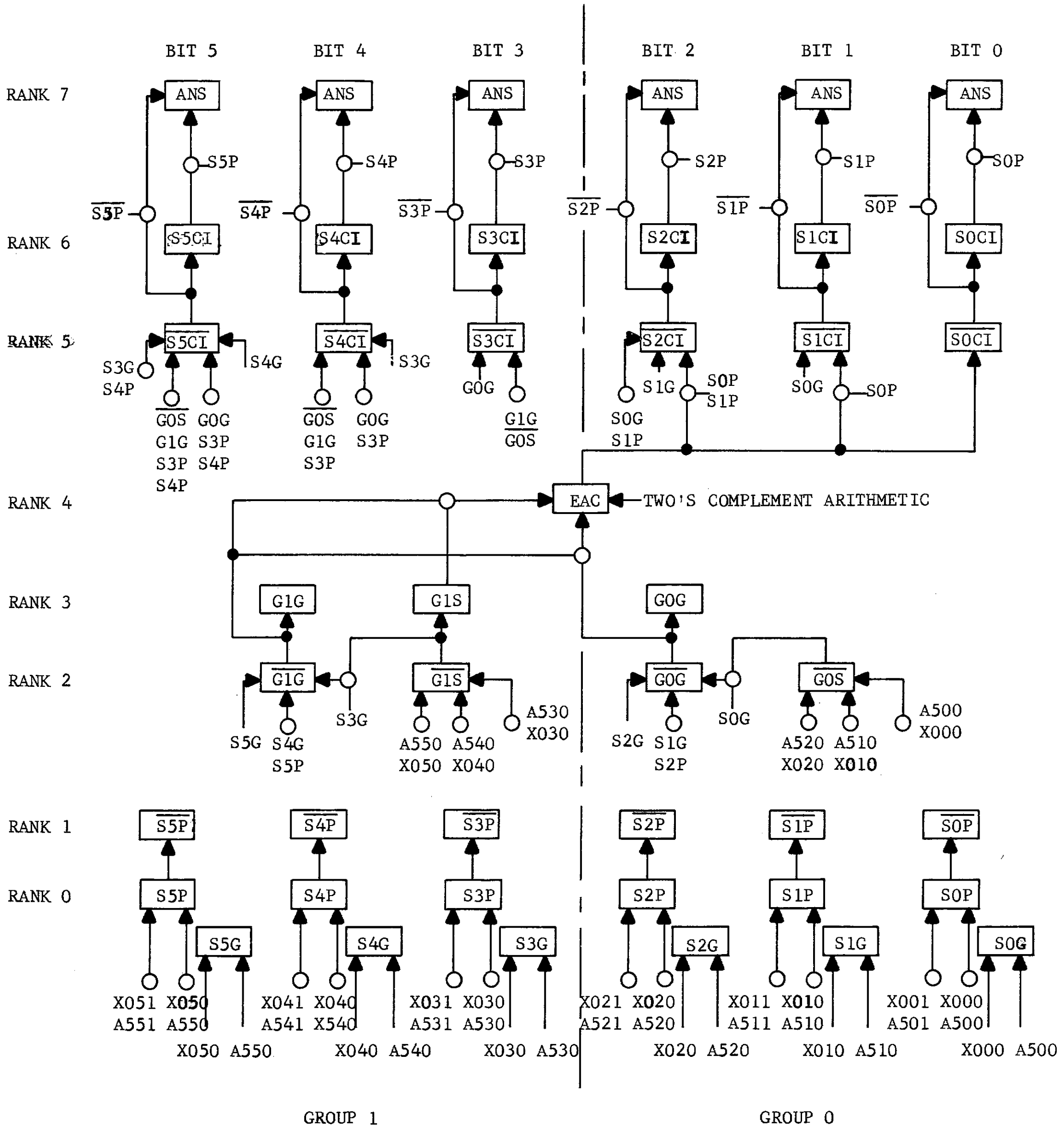


Figure 7-12. 6-Bit Additive Adder

For example:

1) SOG indicates that the inverter will output a logical one if stage 2^0 is a generate (one-one combination).

2) $\overline{\text{S3P}}$ outputs a logical one if stage 2^3 is not a pass ($\overline{\text{PASS}}$).

3) SICI outputs a one if stage 2^1 has a carry input (CI).

If the code begins with a G, it defines a condition relative to that Group. The number and the remaining letter define the group (0 or 1) and the condition. For example:

1) $\overline{\text{GOS}}$ indicates that there are no satisfies in group 0 (stage 2^0 $\overline{\text{satisfy}}$ and stage 2^1 $\overline{\text{satisfy}}$ and stage 2^2 $\overline{\text{satisfy}}$).

2) GOG indicates that there was a generate somewhere in Group 0 not satisfied by group 0 (a carry leaving the group due to a generate within the group).

The grouping of stages together is necessary because the logic cards (inverters) may have a maximum of 11 input terms (pin limitations). Because a group term defines three stages collectively, it could then replace a three-way AND gate and reduce the number of inputs.

Adder Operation

Rank 0. The function of rank 0 is to determine which stages are generates and which are passes. If a given stage is not a pass or a generate, it must be a satisfy. The lower half of Rank 0 determines which stages are generates. This is accomplished by applying the clear side outputs of the operand flip-flops to the inverter. A Rank 0 inverter will output a one only if both flip-flops are set. This constitutes a stage generate (Figure 7-13). If either or both flip-flops are cleared, the SOG inverter would output a logical 0 ($\overline{\text{generate}}$).

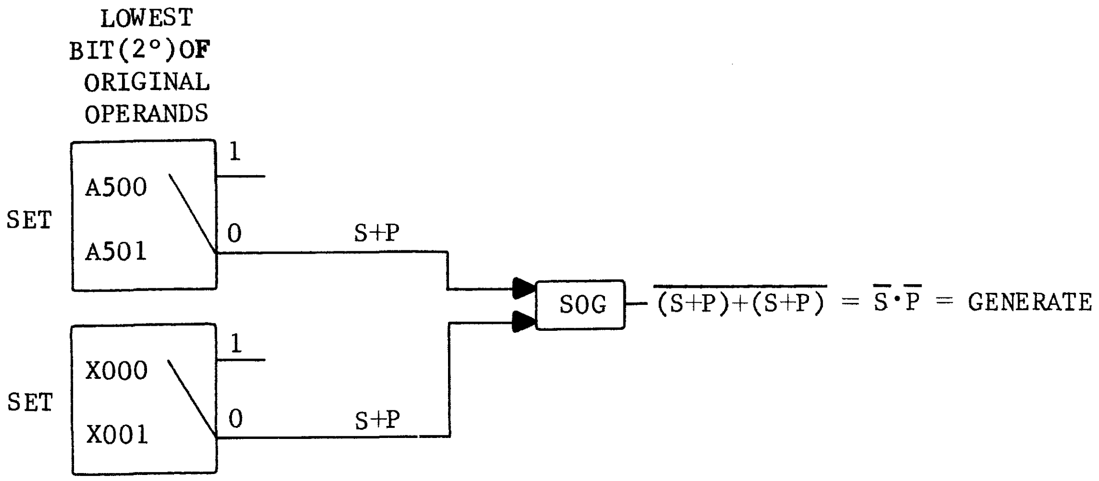

LOWEST BIT(2°)OF ORIGINAL OPERANDS
SET
A500
A501
1
0
S+P
SET
X000
X001
1
0
S+P
SOG
$\overline{(S+P)+(S+P)} = \overline{S}\cdot\overline{P}$ = GENERATE

Figure 7-13. Stage Generate Inverters

The upper half of Rank 0 defines these stages that are passes. A stage is considered to be a pass if the two operand bits are unlike (0 and 1 or 1 and 0). These two combinations are described as an EXCLUSIVE OR because the one-one combination of the Inclusive OR is excluded. The actual circuit illustrated in Figure 7-14.

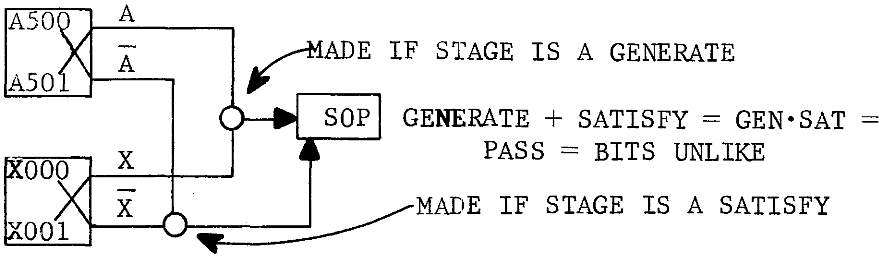

A500
A501
A
$\overline{A}$
X000
X001
X
$\overline{X}$
SOP
MADE IF STAGE IS A GENERATE
GENERATE + SATISFY = GEN•SAT =
PASS = BITS UNLIKE
MADE IF STAGE IS A SATISFY

Figure 7-14. Stage "Pass" inverters

Rank 1. Rank 1 is the complement of the upper half of Rank 0 and defines stages that are not passes ($\overline{PASS}$). You learned earlier that it is possible to form the sum of two operands if each stage is defined as PASS or $\overline{PASS}$ and whether or not each stage has a carry input. Rank 0 and Rank 1 have defined the stages as PASS or $\overline{PASS}$. The sole function of ranks 2 through 5 is to determine the other factor--those stages that have carry inputs.

Rank 2 and Rank 3. Ranks 2 and 3 are group terms, each collectively defining the status of three stages. The GOS term indicates that group 0 does not have a generate (carry) leaving the group that was generated within the group.

22. What would be the outputs of $\overline{GOG}$, $\overline{G1G}$, and $\overline{GOS}$, $\overline{G1S}$ under each of the following conditions? Check your answers with those at the end of the chapter.

	Group 1	Group 0	$\overline{GOG}$	$\overline{G1G}$	$\overline{GOS}$	$\overline{G1S}$
a)	GGS	PGS	___	___	___	___
b)	PPG	GPS	___	___	___	___
c)	PGS	GGG	___	___	___	___
d)	GGS	GSS	___	___	___	___
e)	SGG	GSS	___	___	___	___
f)	PGS	PPP	___	___	___	___
g)	SSS	GPG	___	___	___	___
h)	PPP	GGG	___	___	___	___
i)	GSP	SSS	___	___	___	___

23. What would the six stages have to be if all four terms of Rank 2 have a one output?

Figure 7-15 illustrates how the $\overline{GOS}$ and $\overline{GOG}$ conditions are determined.

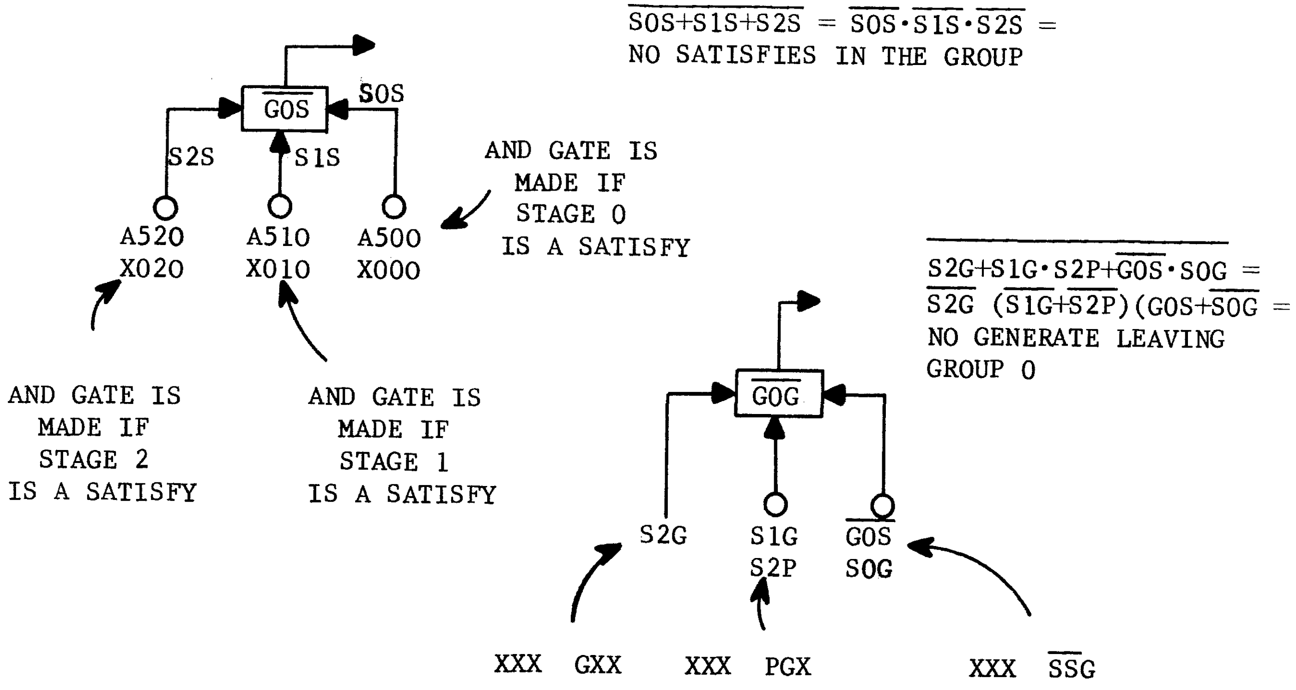

$\overline{S0S+S1S+S2S} = \overline{S0S}\cdot\overline{S1S}\cdot\overline{S2S} =$
NO SATISFIES IN THE GROUP
$\overline{GOS}$
S0S
S2S
S1S
AND GATE IS MADE IF STAGE 0 IS A SATISFY
A520 X020
A510 X010
A500 X000
AND GATE IS MADE IF STAGE 2 IS A SATISFY
AND GATE IS MADE IF STAGE 1 IS A SATISFY
$\overline{S2G+S1G\cdot S2P+\overline{GOS}\cdot S0G} =$
$\overline{S2G}\ (\overline{S1G}+\overline{S2P})(GOS+\overline{S0G}) =$
NO GENERATE LEAVING GROUP 0
$\overline{GOG}$
S2G
S1G S2P
$\overline{GOS}$ S0G
XXX GXX
XXX PGX
XXX $\overline{SSG}$

Figure 7-15

The $\overline{GOG}$ term has three inputs, each of which indicates a condition that would cause a generate to leave the group. In none of these conditions exists, each input would be a zero and the GOG term would output a one, indicating $\overline{GOG}$.

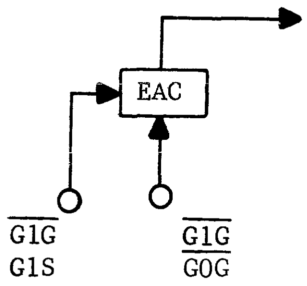
Rank 3 is the complement of rank 2. The GOS term indicates that group 0 does have a satisfy in one of the three stages. The GOG term indicates that the group has a generate and it is leaving the group. The output equation of GOG is the same as the input equation to $\overline{GOG}$ (S2G+S1G·S2P+S0G·$\overline{GOS}$); the output of $\overline{GOS}$ is the same as the input to $\overline{GOS}$ (S0S+S1S+S2S).

These terms will be used to determine whether or not EAC (end around carry) exists.

Rank 4. Rank 4 consists of a single term (EAC) that examines all stages collectively to determine if EAC exists. The group terms feed EAC to reduce the number of inputs. Figure 7-16 illustrates how the EAC inverter determines if EAC exists by employing the use of the group terms. There are only two conditions that could cause EAC. If group 1 has a generate leaving, it must be carried end around. If group 0 has a generate leaving and group 1 does not have a satisfy, the generate would be propagated through group 1 and consequently end around.

1. GXX XXX   2. $\overline{S}\ \overline{S}\ S$ G X X

The Boolean output equation for EAC should reflect these two possible conditions.



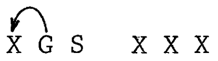
$\overline{\overline{G1G}\cdot G1S+\overline{G1G}\cdot\overline{G0G}} =$
$(G1G+\overline{G1S})\ (G1G+G0G) =$
$G1G+\overline{G1S}\cdot G0G = EAC$

Figure 7-16. End around carry

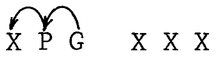
Isn't that amazing? It works!

Assume for a moment that our adder has four groups and 12 stages. What additional inputs would be required to indicate all EAC conditions? Draw your circuit in the following space and then check your solution with #24 at the end of the chapter.

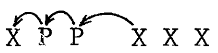
Rank 5. A logical one out of rank 5 indicates that a stage has no carry input; a logical zero indicates that the stage does have a carry input. The lowest bit term, $\overline{SOCI}$, is fed by EAC. If EAC does exist, $\overline{SOCI}$, will output a zero indicating the carry input to that stage. Under what conditions could stage 5 ($\overline{S5CI}$) have a carry input?

1) 

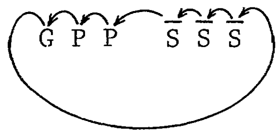
(Stage four a generate.)

2) 

(Stage three a generate and stage four a pass.)

3) 

(Group 0 generate and stages three and four passes.)

4) 

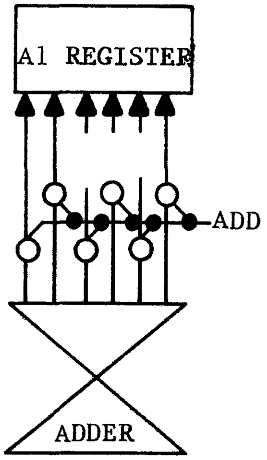
(Group 1 generate, Group 0 satisfy, stages three and four both passes.)

These four conditions happen to be the inputs to $\overline{\text{S5CI}}$. If none of those conditions exist, the stage does not have a carry input and S5CI will output a logical one.

Rank 6. The output of rank 6 is the complement of the output of rank 5. For any two operands, either $\overline{\text{SOCI}}$ (rank 5) or SOCI (rank 6) will output a logical one. The other term must therefore output a logical zero. These two terms are ANDed with the PASS-$\overline{\text{PASS}}$ terms from ranks 0 and 1 to form the complement of the answer.

Rank 7. Rank 7 will output the sum of the two original operands in X1 and A2. The two gates into rank 7 are SOCI·PASS or $\overline{\text{SOCI}}\cdot\overline{\text{PASS}}$ which would render the output equation (SOCI·PASS) + (SOCI·$\overline{\text{PASS}}$). These are the two conditions under which we decided the answer bit should be a one. If one of these two conditions exist, both of the AND gates into Rank 7 will be broken and will allow the ANS inverter to output a logical "1".

The output of the adder (sum) feeds the A1 Register which can hold the sum. The sum is gated to the A1 register only if the ADD pulse is generated.



SUM to A1 register

Figure 7-17. Adder output

What would be the result if the complement of an operand is loaded into either X1 or A2 and the answer is gated into the A1 register? Remember how it was possible to subtract by complementing one of the operands and adding?

Partial Add

Some computer instructions require the adder to Half Add (partial add) two operands. What modifications would have to be made to the adder to allow it to do a partial add? Remember, the partial add is a stage by stage add with carrys disregarded. Figure 7-18 illustrates one way to force the adder to partial add.

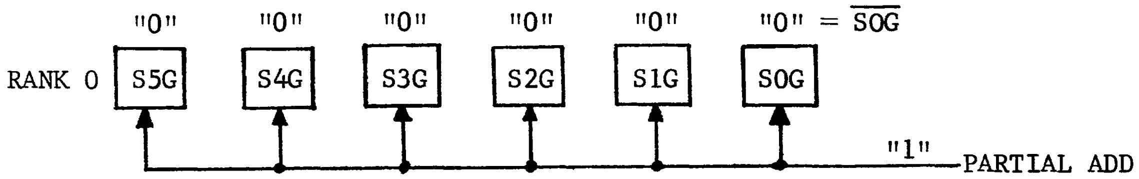


Figure 7-18. Partial Add

The partial add input would force each stage generate inverter to ouput a logical zero. If a "1" translates as SOG, a "0" would indicate $\overline{\text{SOG}}$, simulating the effect that no stages are generates. If no generates exist, there can not be any carry inputs and all inverters in Rank 5 (SOCI-S5CI) will output a logical one. Rank 5 is then ANDed with stage $\overline{\text{PASS}}$ terms (bits alike) to form the partial sum.

Another method of partial adding is illustrated in Figure 7-19. This method would be faster than the previous method because time need not be allowed for the adder to determine which stages have input carries. Only the logic for stage 0 is shown but the other stages would have identical inputs.

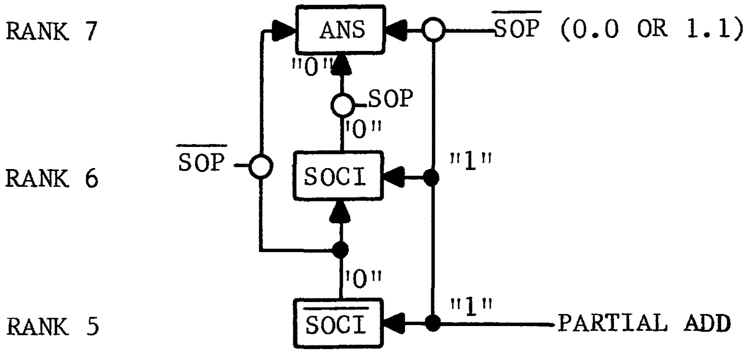


Figure 7-19. Partial Add

The partial add input to $\overline{\text{SOCI}}$ and SOCI would force the output of both to a zero breaking both normal AND gates into the ANS term. However, the partial add input would enable a third AND gate into the answer inverter. If the stage is not a pass (bits alike), the answer inverter will output a zero. If the stage is a pass, none of the gates into the answer inverter are made and it will output a one (0-1 or 1-0+ answer bit of "1").

Logical Product

The logical product of two operands results in a one bit only if both operand bits are ones.

0 X 1 = 0

0 X 0 = 0

1 X 0 = 0

1 X 1 = 1

The adder could also be modified to output the logical product as illustrated in Figure 7-20. Again, only the circuitry for stage 0 is shown.

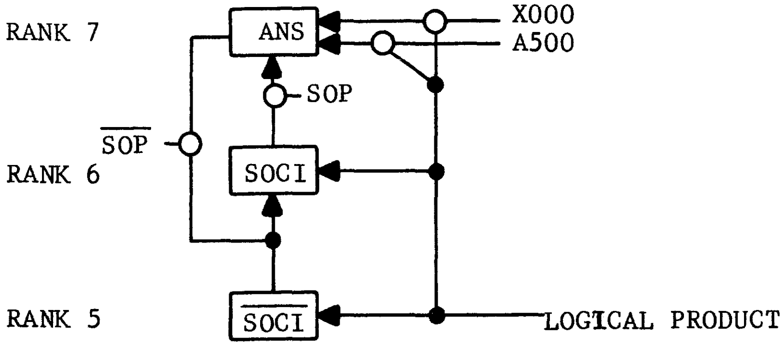


Figure 7-20. Logical Product

Again, both the $\overline{\text{SOCI}}$ and the SOCI terms are disabled (zeros out) by the logical product input. The ANSWER inverter will output a logical one only if both operands have a one in that bit position. If either operand flip-flop (X000/X001 or A500/A501) is cleared (PASS or SATISFY), one of the AND gates will be made and the ANSWER inverter will output a logical zero.

Twos Complement Arithmetic

The same adder that is normally used to form the sum of two operands (ones complement arithmetic) is also used to modify addresses. One common computer memory size consists of $32{,}768_{10}$ words, accessed by addresses 00000_8 through 77777_8. The two signed operands 00000_8 and 77777_8 both represent the quantity zero but addresses 00000_8 and 77777_8 each represent a different memory location. If ones complement additive arithmetic is used for address modification, memory location 00000_8 could not be accessed by modification. For example ADD in ones complement notation

$$\begin{array}{r} 77776 \\ \underline{00001} \\ 77777_8 \end{array} \qquad\qquad \begin{array}{rr} & 77777 \\ & \underline{00001} \\ & 00000 \\ \text{EAC} & \underline{1} \\ & 00001 \end{array}$$

Address 00000_8 would be missed because EAC compensated for the extra operand in the modulus -1 device. What measures could be taken to prevent skipping over address 00000_8?

Address modification must be done in twos complement notation if all addresses are to be accessable. Because EAC readjusts the answer when passing through negative zero, why not prevent EAC and its compensation? Let's see what would happen.

```
77776          77777
00001          00001
-----          -----
77777          00000
                        disable EAC
               -----
               00000
```

By preventing EAC from readjusting the answer, addresses 00000_8 and 77777_8 can both be accessed. Some computers do all arithmetic in ones complement notation and live with the fact that one memory location cannot be accessed through modification. If twos complement arithmetic is desired, it can be accomplished by forcing a logical "1" into the EAC inverter (Figure 7-12) and disabling the output so EAC can never exist.

If address modification is performed in ones complement notation in the arithmetic adder, provisions must be made to transform the 15-bit address and modifier to adder length (twenty-four bits).

The 15-bit address and modifier are added in the 24-bit adder by extending the upper bit (2^{14}) of the address and modifier throughout the remaining nine bits of the A2 and X registers. Figure 7-21 illustrates how this is accomplished.

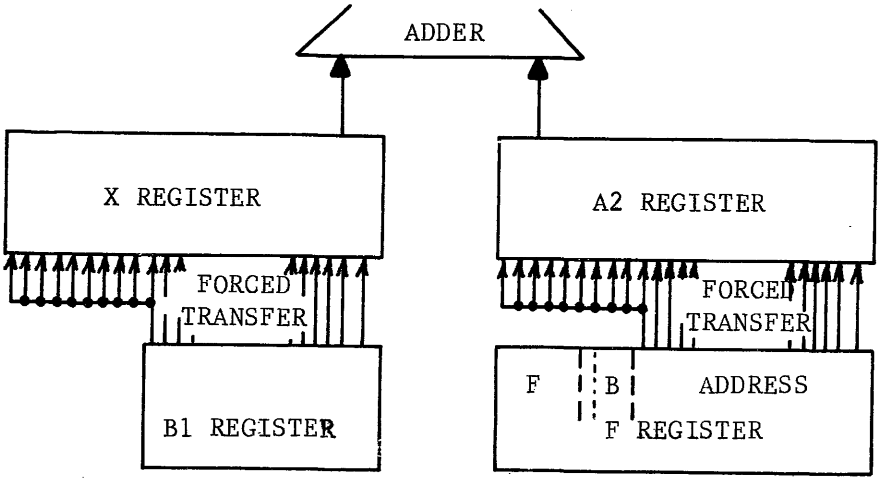


Figure 7-21. Sign extension

Only the lower fifteen bits of the answer (modified address) are gated back to the F register, but sign extension was necessary to derive the correct ones complement answer (permit EAC if existent). Would sign extension be required for twos complement address arithmetic? Why, or why not?

Answer the review questions and then check your answers with those at the end of the chapter.

25) How could the adder be modified to produce the complement of the sum as an output?

26) Two operands are being added by the 6-bit adder in Figure 7-12. Due to a malfunction, the SOCI inverter outputs a continuous logical one. The adder would produce ___?___ as the sum of 53_8 and 22_8.

27) Without chasing ones and zeros, complete the following diagram indicating the logical output of every adder inverter (Figure 7-12) if 13_8 and 12_8 are being added.

RANK 7 ____ ____ ____ ____ ____ ____

RANK 6 ____ ____ ____ ____ ____ ____

RANK 5 ____ ____ ____ ____ ____ ____

RANK 4 ____

RANK 3 ____ ____ ____ ____

RANK 2 ____ ____ ____ ____

RANK 1 ____ ____ ____ ____ ____ ____

____ ____ ____ ____ ____ ____

RANK 0 ____ ____ ____ ____ ____ ____

28) How are operands subtracted in an additive adder?

29) List the terms that refer to a partial add operation.

30) Add the following operands in both sevens and eights complement arithmetic.

Why is the result sometimes the same?

a) 77000
01000

b) 52525
25253

c) 00010
00077

31) What is the advantage of having twos complement capabilities in a ones complement adder?

32) The addition of a number and its complement always results in a negative zero answer. However, there is no arithmetic quantity that can be defined as negative zero. Why?

This chapter has discussed a computer Arithmetic Section, its function and operation. The Arithmetic Section consists of the A1, A2, Q1, Q2, and X registers and the adder. Register contents may be sensed for equality to zero or the sign bit may be sensed to determine the sign of an operand. The A2 and X registers feed the adder and the answer is gated to A1, F (address portion), or one of the index registers. The Q registers are auxiliary arithmetic registers and are used in conjunction with the A registers during multiply and divide operations.

You learned the theory and operation of both the serial and parallel adder, and the advantages of each. The adder always forms the <u>sum</u> of two operands but may also produce the effective <u>difference</u> if one of the operands is complemented before being fed to the adder.

Emphasis was placed on the parallel adder, commonplace in modern digital computers. The actual adder operation was explained and the input and output boolean equation for each term was derived.

The adder (Figure 7-12) is capable of forming the sum of either signed (ones complement) or unsigned (twos complement) operands. Fifteen-bit operands can be added by entering them into the A2 and X registers with sign extension. The output of the adder is monitored for overflow, caused by exceeding the modulus of the device when performing signed (modulus -1) arithmetic. You also learned how an adder forms the partial sum (Exclusive OR) and the logical product of two operands.

If you understand each of the topics discussed in this chapter, you should be able to understand the Arithmetic Section of any computer. The circuits may appear unfamiliar in different computers, but the Arithmetic Section has essentially the same function as that of any other digital computer.

ANSWERS TO CHAPTER VII QUESTIONS

1. $(\text{AZJ,EQ}\cdot A = +0) + (\text{AZJ,NE}\cdot A \neq +0) + (\text{AZJ,GE}\cdot A+) + (\text{AZJ,LT}\cdot A-)$

2. $(\overline{\text{AZJ,EQ}} + \overline{A + +0})\ (\overline{\text{AZJ,NE}} + \overline{A \neq +0})\ (\overline{\text{AZJ,GE}} + \overline{A\ \text{pos}})\ (\overline{\text{AZJ,LT}} + \overline{A\ \text{neg}}) =$

   $(\overline{\text{AZJ,EQ}}) + A \neq +0)\ (\overline{\text{AZJ,NE}} + A = +0)\ (\overline{\text{AZJ,GE}} + A\ \text{neg})\ (\overline{\text{AZJ,LT}} + A\ \text{pos}) =$
   "jump conditions not met"

3. $\overline{(\overline{\text{AZJ,EQ}} + A \neq +0)\ (\overline{\text{AZJ,NE}} + A = +0)\ (\overline{\text{AZJ,GE}} + A\ \text{neg})\ (\overline{\text{AZJ,LT}} + A\ \text{pos})} =$

   $(\text{AZJ,EQ}\cdot\overline{A \neq +0}) + (\text{AZJ,NE}\cdot\overline{A = +0}) + (\text{AZJ,GE}\cdot\overline{A\ \text{neg}}) + (\text{AZJ,LT}\cdot\overline{A\ \text{pos}}) =$

   $(\text{AZJ,EQ}\cdot A = +0) + (\text{AZJ,NE}\cdot A \neq +0) + (\text{AZJ,GE}\cdot A\ \text{pos}) + (\text{AZJ,LT}\cdot A\ \text{neg}) =$
   JUMP

4. A J107 $\overline{A}$ J110 $\overline{\overline{A}} = A$

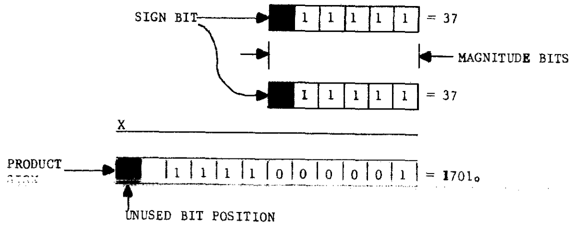
5. No. The sign bit could change although no carry is propagated "end around."

   Example:

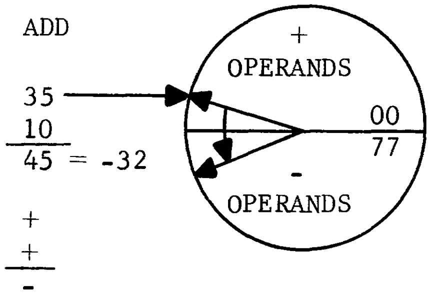
$$\begin{array}{lcl} 010\ 111 & = & +\ 27_8 \\ \underline{010\ 000} & = & +\ 20_8 \\ \\ 100\ 111 & = & 47_8 = -30 = \text{overflow but } \overline{\text{EAC}} \end{array}$$

6. No! In the preceeding example, the answer is 47 which is equal to -30 when using complement notation. The sum of 27 and 20 is 47 but 47 could not be expressed in a six bit <u>signed</u> register.

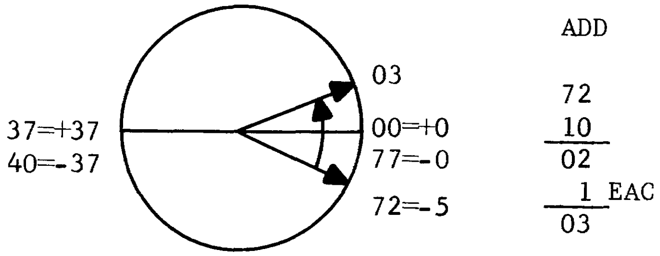
7. If you multiply two 6-bit operands, a 12-bit product would result. Each operand consisted of one sign bit and five magnitude bits. Therefore, the two operands would produce a 10-bit product in a 12-bit register. This results in one extra bit position not being used because the product has only one sign bit. Any carry generated by <u>any</u> two operands could be propogated only to the tenth bit (2^9) and could not affect the sign bit. For example, the largest positive 5-bit operand is $011\ 111_2$ (37_8). $37 \times 37 = 1701_8$.



8. There is none. Overflow indicates that the results of either an ADD or SUBTRACT have exceeded the modulus of the device whereas end around carry indicates a compensation for passing over negative zero.

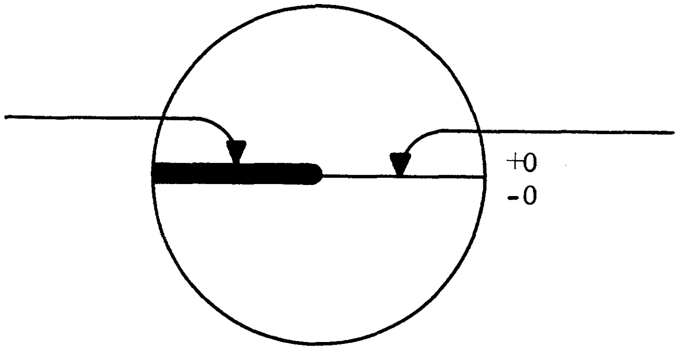


OVERFLOW



END AROUND CARRY

PASSING OVER THIS LINE EXCEEDS THE MODULUS OF THE DEVICE AND RESULTS IN OVERFLOW (INCORRECT ANSWER)



PASSING OVER THIS LINE RESULTS IN EAC COMPENSATING FOR NEGATIVE ZERO (CORRECT ANSWER)

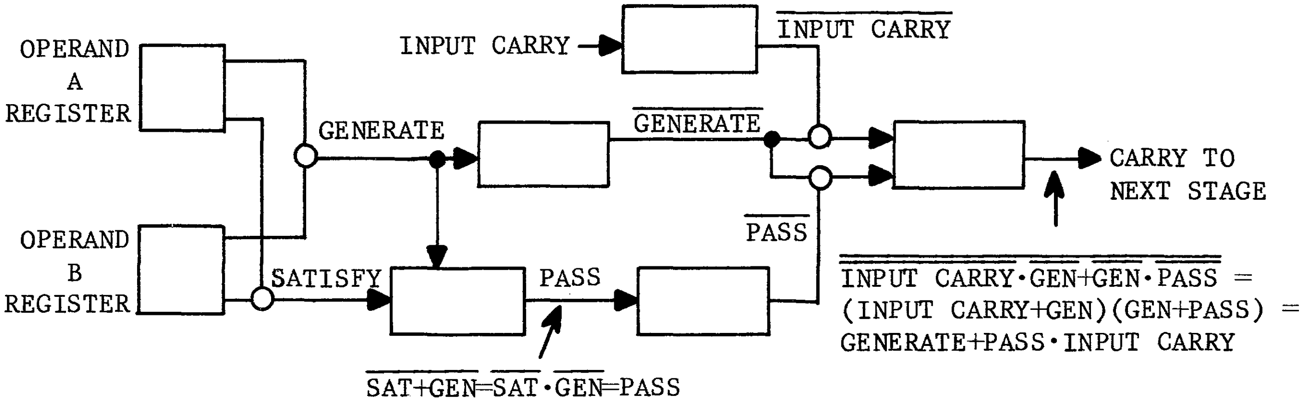
```
9.   P P P   P P G                10.  P P G   S G P
     0 0 0   0 0 1   = +01             1 0 1   0 1 0   = -25

     1 1 1   1 1 1   = -00             0 1 1   0 1 1   = +33
     -------------                     -------------
     0 0 0   0 0 0                     0 0 0   1 0 1

                 1                                 1
     -------------                     -------------
     0 0 0   0 0 1   = +01             0 0 0  1 1 0    = +06
```

```
11.     S S S    S S S                   12.   G G G    G G G
        0 0 0    0 0 0   = +0                  1 1 1    1 1 1   = -00

        0 0 0    0 0 0   = +0                  1 1 1    1 1 1   = -00
        -------------                          -------------

        0 0 0    0 0 0   = +0                  1 1 1    1 1 0

                                                            1
                                               -------------

                                               1 1 1    1 1 1   = -00


13.     P G G    G G S                   14.   G S G    S G S
        1 1 1    1 1 0   = - 01                1 0 1    0 1 0   = -25

        0 1 1    1 1 0   = + 36                1 0 1    0 1 0   = -25
        -------------                          -------------

        0 1 1    1 0 0                         0 1 0    1 0 0

                     1                                      1
        -------------                          -------------    -----

        0 1 1    1 0 1 = + 35                  0 1 0    1 0 1   = +25

                                                    OVERFLOW

                                               S S S    P S P
15.     0 0 0    1 0 0 = +04                   0 0 0    1 0 0 = +04

   -    1 1 1    1 1 0 = -01 complemented=+    0 0 0    0 0 1 = +01
   ------------------    ---                   -------------------
                         +05                   0 0 0    1 0 1 = +05

                                               G S P    P P S
16.     1 0 1    0 1 0 = -25                   1 0 1    0 1 0 =-25

      - 0 1 1    0 1 1 = +33  complemented= +1 0 0    1 0 0 =-33
        --------------   ---                 ---------------   ---
                         +17                   0 0 1    1 1 0
                                                            1
                                               ---------  ---   -----
                          OVERFLOW             0 0 1    1 1 1 = 60 = 17


                                               P S S    G G G
17.     1 0 0    1 1 1 = -30                   1 0 0    1 1 1 = -30

     -  1 1 1    0 0 0 = -07 complemented= +   0 0 0    1 1 1 = +07
     ----------------    ---               -----------------------
                         -21                   1 0 1    1 1 0 = -21


                                               P P P    P P P
18.     0 1 1    1 0 1 = +35                   0 1 1    1 0 1 = +35

     -  0 1 1    1 0 1 = +35 complemented- +   1 0 0    0 1 0 = -35
     ----------------    ---                   -------------------
                         +00                   1 1 1    1 1 1 = -00
```

$$\begin{array}{llcll}
 & & & P\;P\;G & S\;S\;S \\
19. & 1\;1\;1\quad 0\;0\;0 = -07 & & 1\;1\;1 & 0\;0\;0 = -07 \\
 & -\underline{1\;1\;0\quad 1\;1\;1} = \underline{-10} & \text{complemented} = + & \underline{0\;0\;1} & \underline{0\;0\;0} = \underline{+10} \\
 & \qquad\qquad\qquad\quad +01 & & 0\;0\;0 & 0\;0\;0 \\
 & & & & 1 \\
 & & & \overline{0\;0\;0} & \overline{0\;0\;1} = \overline{+01}
\end{array}$$

$$\begin{array}{llcll}
 & & & S\;S\;P & G\;P\;P \\
20. & 0\;0\;0\quad 1\;0\;0 = +04 & & 0\;0\;0 & 1\;0\;0 = +04 \\
 & -\underline{1\;1\;0\quad 0\;0\;0} = \underline{-17} & \text{complemented} = + & \underline{0\;0\;1} & \underline{1\;1\;1} = \underline{+17} \\
 & \qquad\qquad\qquad\quad +23 & & 0\;1\;0 & 0\;1\;1 = +23
\end{array}$$

21. Carry Generation network


OPERAND
A
REGISTER
OPERAND
B
REGISTER
INPUT CARRY
$\overline{INPUT\ CARRY}$
GENERATE
$\overline{GENERATE}$
SATISFY
PASS
$\overline{PASS}$
CARRY TO
NEXT STAGE
$\overline{SAT+GEN}=\overline{SAT}\cdot\overline{GEN}=PASS$
$\overline{\overline{INPUT\ CARRY}\cdot\overline{GEN}+\overline{GEN}\cdot\overline{PASS}}$ =
(INPUT CARRY+GEN)(GEN+PASS) =
GENERATE+PASS·INPUT CARRY

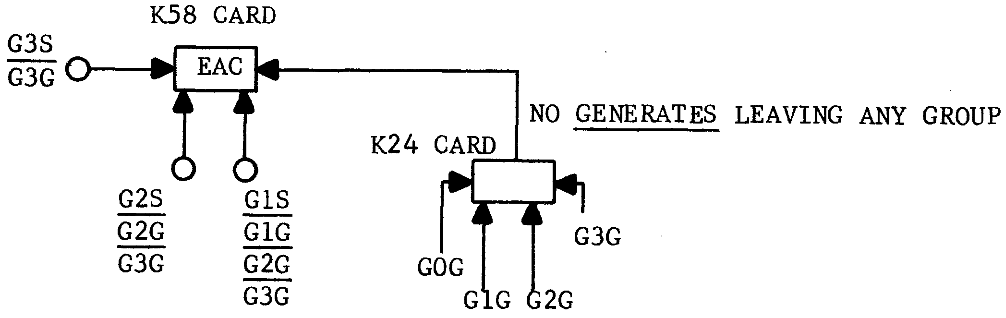
22.

	Group 1	Group 0	$\overline{GOG}$	$\overline{GIG}$	$\overline{GOS}$	$\overline{GIS}$
a)	GGS	PGS	0	0	0	0
b)	PPG	GPS	0	0	0	1
c)	PGS	GGG	0	0	1	0
d)	GGS	GSS	0	0	1	1
e)	SGG	GSS	0	1	0	0
f)	PGS	PPP	0	1	0	1
g)	SSS	GPG	0	1	1	0
h)	PPP	GGG	0	1	1	1
i)	GSP	SSS	1	0	0	0

nice binary count, isn't it?

23. All six stages would have to be PASSES because if there are no satisfies in either group, a generate anywhere in a group would cause $\overline{GOG}$ and $\overline{GIG}$ to output a logical zero.

24.


K58 CARD
EAC
$G3S\ \overline{G3G}$
$G2S\ \overline{G2G}\ \overline{G3G}$
$G1S\ \overline{G1G}\ \overline{G2G}\ \overline{G3G}$
NO GENERATES LEAVING ANY GROUP
K24 CARD
GOG
G1G
G2G
G3G

You probably had another four-way AND gate instead of the extra inverter, but remember the limitation of eleven input pins per card. See Figure 6-16 for pin connections. Incidentally, if you designed your circuit by starting with the desired boolean output equation ($G3G + G2G \cdot \overline{G3S} + G1G \cdot \overline{G2S} + GOG \cdot \overline{GIS} \cdot \overline{G2S} \cdot \overline{G3S}$) AND worked back through the inverter for the inputs AND arrived at the correct solution . . . . CONGRATULATIONS! You really understand Boolean Algebra and its application.

25. By interchanging the SOP and $\overline{SOP}$ terms between ranks 6 and 7.

26. 74_8. Both SOCI and $\overline{SOCI}$ output ones with these two operands so one gate into the answer inverter must be made.

27. The two operands dictate these conditions

SSG SGP
001 010 = 12
001 011 = 13

Rank 7	0	1	0		1	0	1
Rank 6	0	1	0		1	0	0
Rank 5	1	0	1		0	1	1
Rank 4				0			
Rank 3		0	1			0	1
Rank 2		1	0			1	0
Rank 1	1	1	1		1	1	0
Rank 0	0	0	0		0	0	1
	0	0	1		0	1	0

28. By complementing one of them before it gets to the adder.

29. Half add
Partial add
Exclusive OR
Selective complement

30. a. 00001
00000

b. 00001
00000

c. 00107
00107

Twos complement arithmetic prevents an EAC in the adder. If a natural EAC is not produced when two operands are added, the ones complement answer and the twos complement answer would then be the same.

31. Allows unsigned and signed arithmetic (modulus or modulus -1 arithmetic) with the same adder.

32. There is no actual positive zero either. Zero is defined as "no magnitude" and, therefore, requires no sign. Either positive or negative zero (in complemented notation) is interpreted as zero (see Figure 7-4).

chapter VIII

Computer Storage

Memory

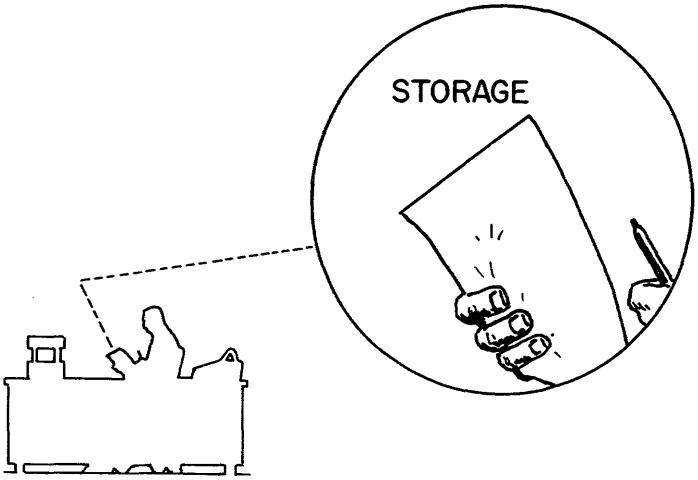
CHAPTER VIII

COMPUTER STORAGE

INTRODUCTION

In its more literal form, the word memory is defined as the power of remembering. A computer requires specially designed circuitry or devices that are capable of storing (remembering) information. The storage circuits or devices are called its memory because the devices allow for the recording and, later, the recall of information. The storage devices themselves hold the information; the storage section controls the recording and recalling of that information.

Again looking at the man at the desk, an analogy to the pencil and paper (storage) and the computer storage section can be made.



The man can write information on the paper and later read it back; a computer storage section also has the ability to write and read back information.

The man can write the information at any location on the paper, e.g., the fifth or the twenty-sixth line, but he must first locate the fifth or the twenty-sixth line. To read back the proper information he must again locate the appropriate line (location).

A computer can also do this. It can write into any of its memory locations and later read back the information from that location. It must be able to locate a specific location upon command.

The man at the desk can record a number on paper one digit at a time (serial) or, by using a rubber stamp, he can record all the digits at one time (parallel). A computer has the same ability: some record information serially, others are parallel in their operations, and still others use a combination of both methods.

Parallel recording is faster because an entire word is read or written in one operation.

When the man writes information on paper it remains there, irrespective of how many times the information is read. Some computer storage devices have this ability to retain information indefinitely. Other storage devices retain information only while the power is on. Some computer storage devices destroy the information as it is read. In such cases, the storage section must rewrite the information at the referenced location while it supplies the information to other sections of the computer.

One further analogy can be made. Students in grade school are taught to read a paper from the top of the page to the bottom, taking each line in sequence. The student is also taught to locate any page in a book by the page number. The actual reading of the paper is in a sequential mode (one line following the other). Finding a page number is more of a random process or mode. Computer memories can operate in either or both of these modes.

The objective of this chapter is to familiarize you with the different types of computer storage devices; the theory of sequential and random access memories; and, specifically, the theory and operation of the magnetic core, random access type of computer storage.

The more common types of computer storage devices are the drum, disk, and the magnetic core. Each type of device functions by recording information magnetically at a definite location with the capability of later referencing that same location to retrieve the information. The method used to read and record information determines whether the device has sequential access or random access to information. The information may be in the form of: a program, data, a combination of both.

SEQUENTIAL ACCESS TO INFORMATION

Reading the daily newspaper or a magazine is an example of sequential access to information. Because of the limitations of the human brain, a word (or a group of words) is read and stored in memory (the brain) before the next word is read. The entire book or paper can be memorized by concentrating on sequential bits of information.

Some computer devices also function sequentially. The transfer of information between the computer and input/output equipments is accomplished by the sequential reading or writing of computer words (24 binary bits per word).

One type of input equipment is the punched card reader. The programs you wrote in chapter IV could be recorded on a deck of punched cards, with information on each card read <u>sequentially</u> into the computer. The assembler or compiler translates the coded information received from the cards into the object program, storing the information into sequential storage locations. The operator sets the program address register (P) to the address of the first instruction and starts the computer. The computer assumes that the remaining instructions are in sequential order unless a Skip or Jump instruction is encountered.

RANDOM ACCESS TO INFORMATION

In the course of program execution, operand addresses are specified in the instructions. The operands may be in adjacent storage locations or widely separated. In some cases, instructions may repeatedly access data from two or three locations. Roots, powers, and other more complex functions are solved in this manner.

When a Jump instruction is encountered in the execution of a program, the program continues at some new location in memory. Although a new sequence of instructions is often found at the jump address, another jump may be encountered. This indicates that an instruction may be referenced in a random manner, but more importantly, this illustrates that random and sequential access operations can be mixed.

Another example of random access is the instant recall of information read from the magazine or newspaper. Although the information was read sequentially, the human brain can access bits of that information without reviewing the entire paper or book. You can remember isolated incidences from years back without reviewing day-by-day events, another example of random access to information.

Perhaps you wish to show a friend an article in a book. You simply remember the page number (address) of the article and turn to that page (random access). When the article is located, it is read line-by-line (sequential access). The process of locating the article and reading it employs the use of both methods of information access, sequential and random-access.

DESIGNS OF STORAGE DEVICES

The design of a computer storage device is based on the operational characteristics just discussed. The selection of a particular design is dependent upon the particular application and the desired result.

A sequential access memory scans all addresses in sequence until the desired address is reached. In many machines the scanning begins when the computer is turned on and ends when the computer is turned off.

A random access memory device can instantly reference data at any location. This totally elecronic device is therefore faster than its sequential access counterpart. The random access memory is arranged somewhat like a matrix. A storage location is referenced by selecting any two coordinates in the matrix, providing immediate access to any location.

Both random and sequential access memories are used in computers. Each type of device has advantages and disadvantages; selection of one or the other is based on the desired capability of the machine. Inherently, random access memories are faster in all respects than sequential access devices. On the other hand, a random memory is complex and expensive on a cost-per-bit basis. Sequential access memory is slower, but also is simple, easy to repair, and very inexpensive on a cost-per-bit basis.

SEQUENTIAL ACCESS DEVICES

DRUM MEMORY

The drum is one of the older types of storage devices. It can be compared to the cylinder phonographs that were used in the early days of recording. The drum is made of a metallic substance machined to close tolerances with a ferrous oxide coating. It is rotated in the close proximity of read, write, and erase heads which are added to the assembly.

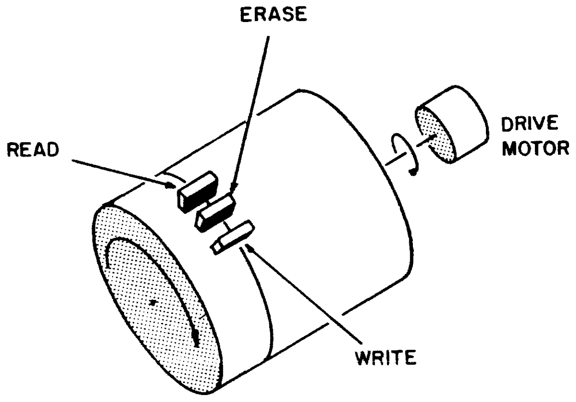


Figure 8-1. Drum Memory

The drum in Figure 8-1 illustrates the process by which data is written, stored, and then read at the read station. The time span between writing and reading depends on: (1) the size of the drum, which, as it gets larger, increases the distance the data must travel to the read or write head; and (2) the speed of the drum which, if increased, reduces the time between write and read operations. The erase head removes both data and noise from the drum so that new data may be added. The drum holds information, but an amplifier must be added so that stored information remains on the drum. The amplifier transfers information from the read head to the write head (Figure 8-2)/

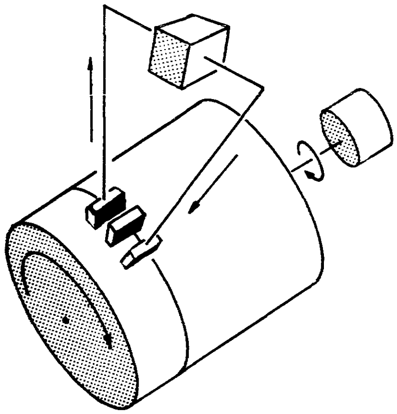


Figure 8-2. Restoration of Recorded Data.

The drum then holds information indefinitely. When new information is written, the read operation can be deleted and new information written in its place.

During one rotation, all of the information on the drum is present at the read amplifiers. It is necessary to have some method to determine when a desired data bit will appear on the output of the read amplifier. One method is to use two timing tracks on the drum. One of the tracks has only a single pulse which is called a revolution mark. This mark establishes an imaginary beginning for the drum. The other track is a clock channel. A periodic waveshape is permanently recorded on the clock channel which, if counted, indicates the position of the drum at any instant. For example, assume that the desired data on the drum is half way around the drum from the revolution mark. If the clock channel has 4096 pulses around the drum, the desired data can be read by counting the pulses after the revolution mark, and sampling the output of the read amplifiers when 2048 clock pulses have occurred. This type of drum appears in Figure 8-3. In the figure, the revolution mark has just passed; the data is being referenced.

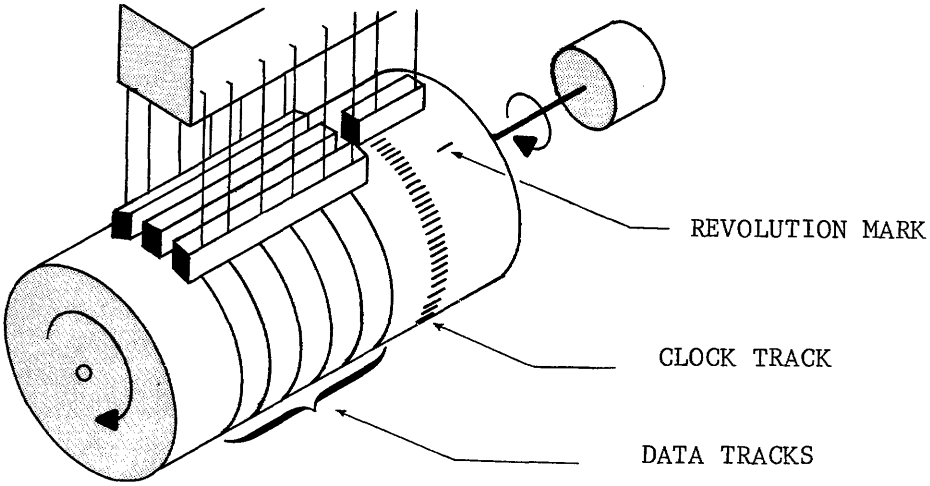


Figure 8-3. Timing of a Two-Track Drum

There are two ways to store data on a drum memory. A serial method is used when speed is not essential. One of the tracks is selected and a data or instruction word is written on the track one bit at a time until the entire word is stored at sequential bit positions of a single track. Several hundred to several thousand words can be stored on a given track. This method is commonly used when the drum is referenced on a single-word basis, with subsequent references to be taken from other portions of the drum.

A parallel method is commonly used when data is referenced in large blocks. In this mode there are enough tracks on the drum for each bit of a data word. If 12-bit words are used, each bit of the word is stored on a separate track. the advantages of drum memory can be seen if consideration is given to access time.

If words are stored or referenced in one-word blocks, the mean time needed to reference a word is the average of the maximum and minimum delays. If the speed of the drum is 50 revolutions per second, revolution time and,consequently, the maximum delay is 20 milliseconds. This occurs if the reference is made after the desired data word has just passed the read station.

If the word appears just as an attempt is made to reference it, the minimum time span occurs. The average delay is found by subtracting the minimum time (0 milliseconds) from the maximum time (20 milliseconds) and dividing the answer by 2. An average delay of 10 milliseconds is produced. This is the average reference time or average access time.

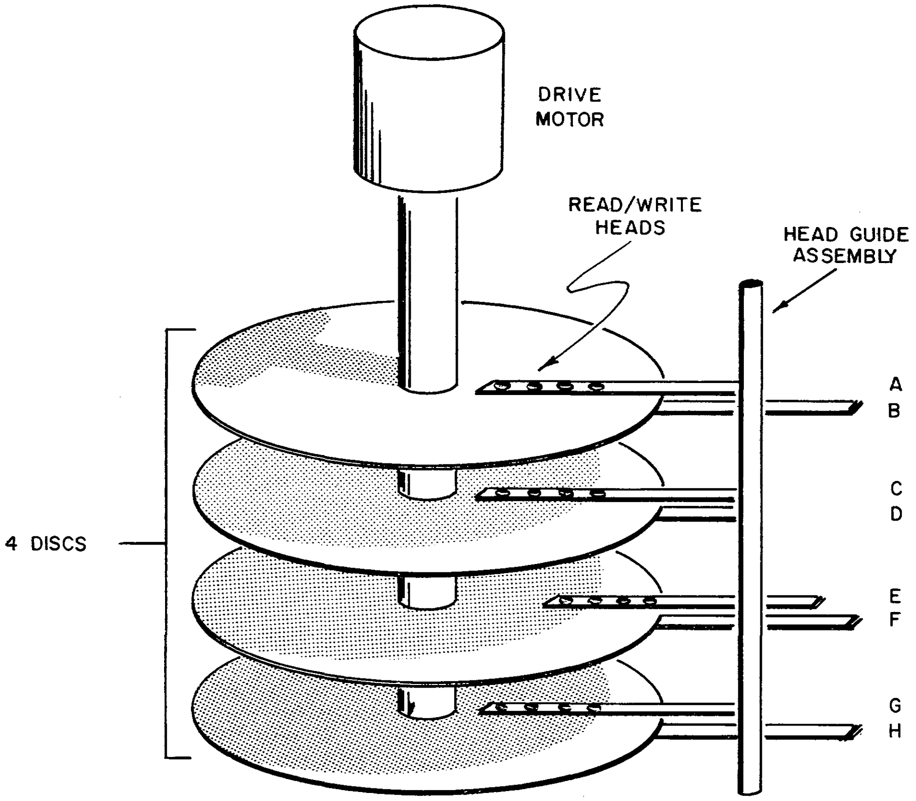
If the average transfer time for a block of 4096 words is considered, it is still necessary to wait for an average of 10 milliseconds for the beginning of the block. Once the transfer begins, however, the rate becomes approximately 4.88 usec per word. Adding the two quantities (4.88 usec per word and 10 milliseconds average time) to first access, the average transfer rate for each word in a block of 4096 words is approximately 7.32 usec.

There are both desirable and undesirable qualities in the drum storage device. The drum provides a good source of block storage, a relatively simple system, constant track length, constant bit spacing, and a constant transfer rate. However, one-word access time is excessively long. Another undesirable quality is that power failure may destroy the contents of memory.

DISK MEMORY

The disk memory, like the drum, is a rotary device. It is selected as a memory device if a greater amount of storage is needed than can be provided by the drum. The recording area of the disk is much greater than the drum, using comparable amounts of space. The drum required one set of read/write heads for each data track. Each disk surface may have 512 data tracks but only one set of read/write heads on a movable arm. The arm positions the head assembly over the desired track and information may then be read or new information recorded.

One type of Control Data disk file has four sets of read/write heads, positioned along a movable arm, for each disc surface. Each set of heads is used for 1/4 of the data tracks on a surface. The advantage provided by multiple heads is that each arm travels less distance to access data with a resultant decrease in time. However, the additional inertia of the increased mass tends to increase start and stop time. Figure 8-4 illustrates the multiple head disk.



The read/write heads are permanently mounted on a movable arm. Arm A for example, is as far toward the center as is possible. Arm B is retracted as far as is possible. Both surfaces of the disk are capable of recording information.

Figure 8-4. Disk Memory

A read/write head assembly may consist of a single head or three individual heads (read, write, and erase). The single head configuration is used for both reading and writing. No erase head is required because previous data is destroyed by writing over it. However, if the head is not exactly positioned over the track, some previous data may remain along one side of the track causing troublesome noise.

The read/write head assembly consisting of three separate heads is normally considered as advantageous over the single head. The erase head is wider than the other heads, erasing both the old data and any noise along the side of the track.

Information is recorded by magnetizing an oxide surface. A small amount of current is applied to the coil of an electromagnet, the write head. The current through the coil produces a magnetic field induced into the magnet. The gap in the magnet has a buildup of magnetic flux lines which magnetizes a tiny spot on the disk or drum. The polarization of the spot is determined by the information being written. However, all information is recorded in a binary representation which requires only a logical one or a logical zero. Figure 8-5 illustrates how information is recorded on a magnetic surface.

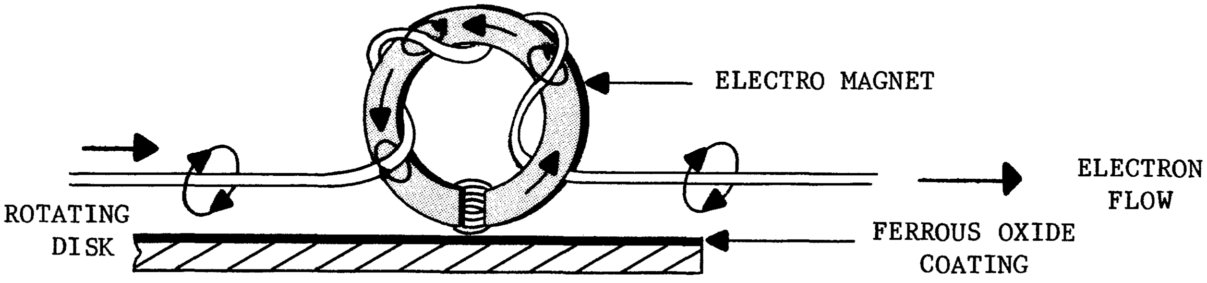


Figure 8-5. Write Head

If the same head is also used for reading, a magnetic field is induced into the magnet when a magnetized spot on the surface passes under the head. The magnetic field of the magnet induces into the coil a current flow in one of two directions. The current flow is detected and translated as one bit of information.

The three-head configuration is illustrated by Figure 8-6. Notice the width of the erase head and its effect on old data (garbage) as new information is recorded.

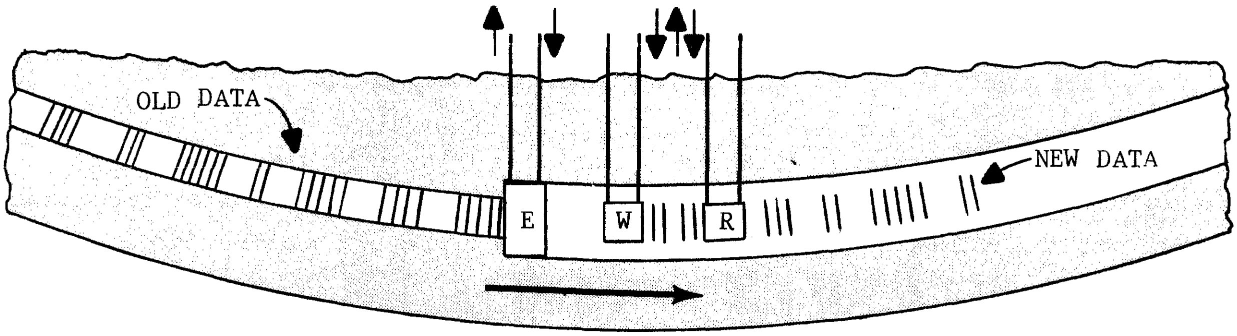


Figure 8-6. (Top view) Three-head Configuration, Writing.

Both the disk and the drum are acceptable memory devices, but, because of the mechanical movement involved, they are slow to access and are vulnerable to mechanical difficulties.

RANDOM ACCESS STORAGE DEVICES

A purely electronic memory device could access any storage location at random because mechanical movement is not required to position the information for reading. Mechanical problems would also be eliminated and access to information would be faster. One possibility would be flip-flops. However, the cost per bit for a flip-flop memory would be prohibitive and a power interruption would cause all stored information to be lost.

The storage section of a computer normally employs the use of a random access device, such a magnetic core or thin-film memory, because of increased access time. Since this is the most prevalent form of memory in present day computers, this chapter will place emphasis upon the magnetic core type of computer storage.

MAGNETIC CORE STORAGE

The storage section of the basic computer is illustrated by Figure 8-7. It consists of the memory stack, the Z register and restoration network, and the memory address register (S register).

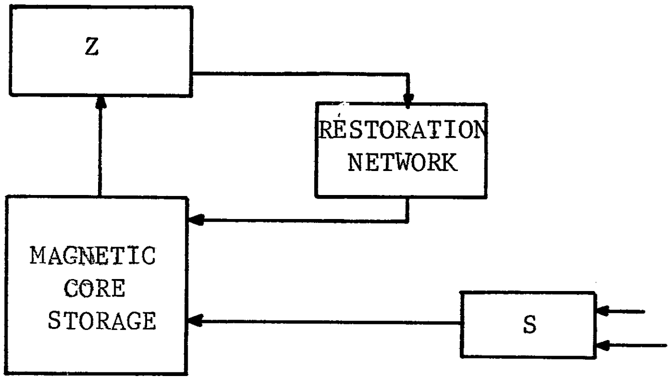


Figure 8-7. Computer Memory Section

Before a word can be read from storage, its address must be transferred to the S register. If an instruction is being read, the address is obtained from the P register. An operand address is obtained from the lower fifteen bits of the F register (Figure 8-8).

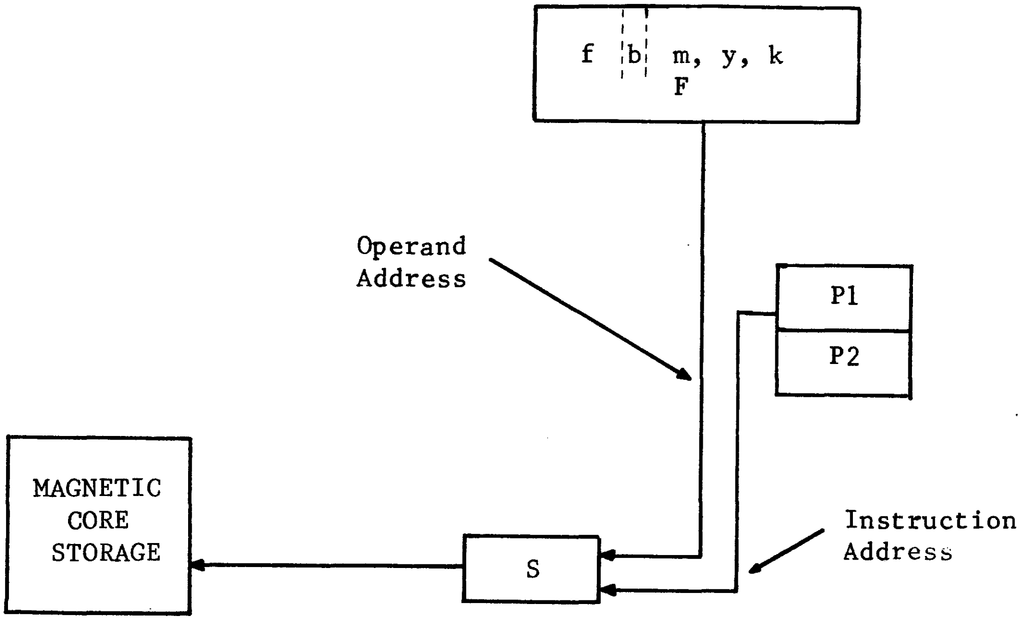


Figure 8-8. Determination of Address Source.

A common memory size is 32,768 locations of 24-bit words. Each of the words may be referenced by one of the octal addresses between 00000 and 77777 (77777_8= $32,767_{10}$ and address 00000 adds one more, for a total of $32,768_{10}$).

When the address of the desired "word" has been transferred to the S register, memory is initiated and the information contained in 24 tiny magnetic cores is read from storage to the Z register. If the word was read as an instruction, it is then gated to the F register; if an operand, it is gated to the X register. The contents of the referenced memory location are destroyed by the reading process and must be restored at that same location for future reference (Figure 8-9).

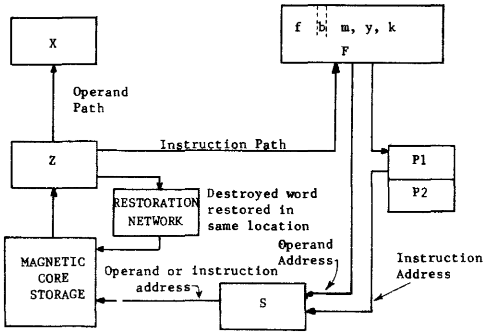


Figure 8-9. Reading a Memory Word.

The size of a core is expressed by two numbers separated by a slash to indicate its outside and inside diameter in thousandths of an inch. Some common core sizes are 80/50, 50/30, 32/20, and 20/12 (Figure 8-10).

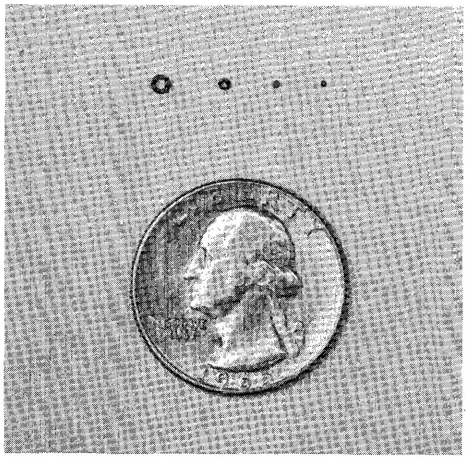


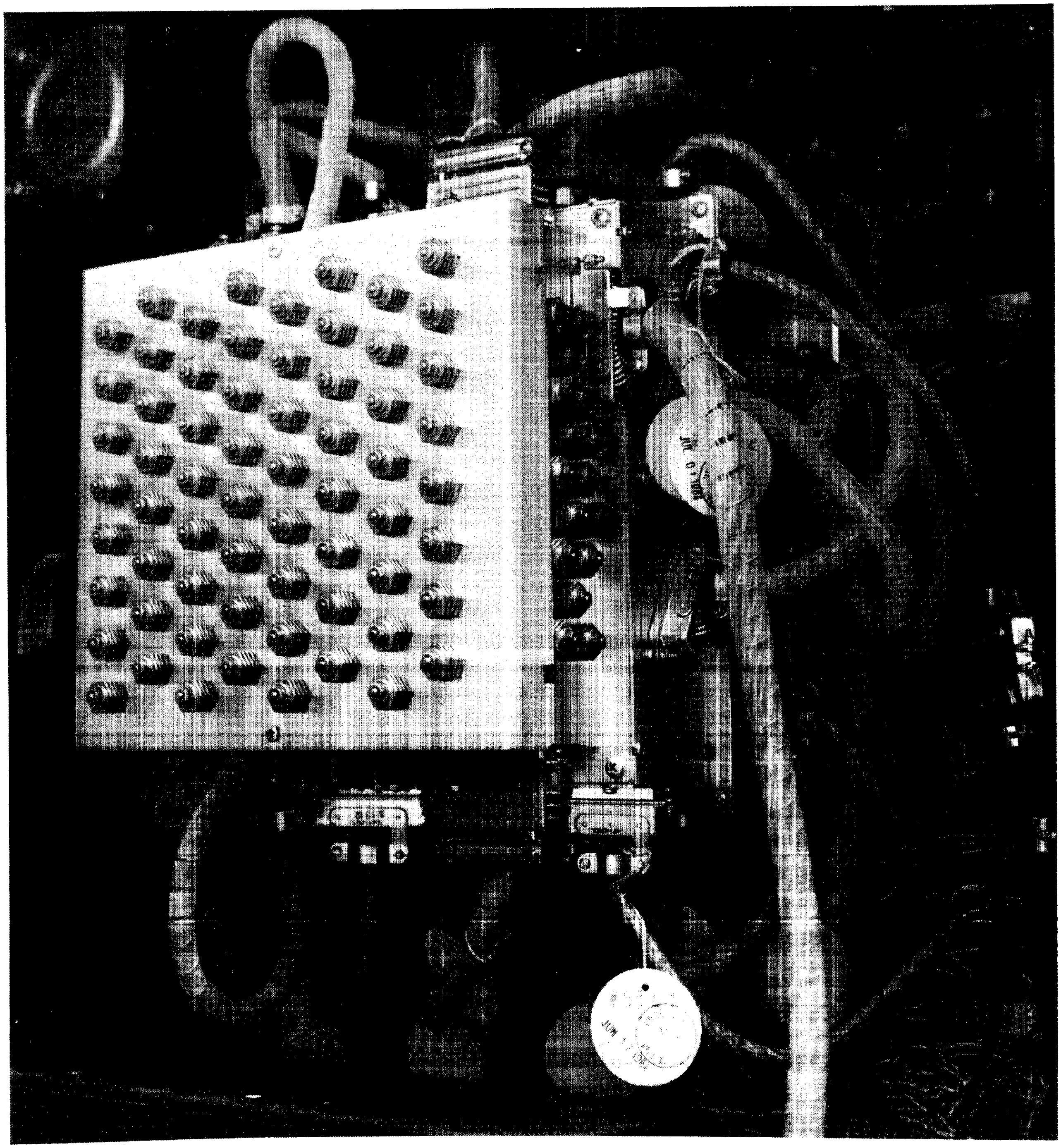
Figure 8-10. Magnetic Cores.

Experimentation is being conducted to perfect a tiny 5/3 core with the hole through the center only about 1½ times the diameter of a human hair. Decreasing the physical size of the core is advantageous because a smaller core has less mass and consequently switches faster.

Four insulated wires pass through each core and attach to a frame. Each frame contains 16, 384 cores evenly divided into four bit-planes of 4,096 cores each. The frame, with the cores and wires, is called a memory wafer (Figure 8-11).

Each memory wafer contains four bits of 4,096 different memory words, with one bit of each word in each of the four planes. Six identical wafers are sandwiched together to form a field that contains 4,096 24-bit words (Figure 8-12).

Two fields are then placed back-to-back to form an 8K memory stack (8,192 words). The cover sheet to this chapter shows an actual 8K memory stack. Four identical 8K stacks are required to form the standard 32K memory (32,768 words).



Phvsical Construction

The operation of the magnetic core storage memory understandably centers around the tiny magnetic core. It is made by pressing minute ferrite particles into the shape of a tiny doughnut and bonding them with epoxy. Each core is capable of storing one bit of one memory word. Consequently, a memory of 32,768 word (24 bits each) would require more than 0.78 million magnetic cores. Some larger computers have a magnetic core memory that requires as many as 13 million cores!

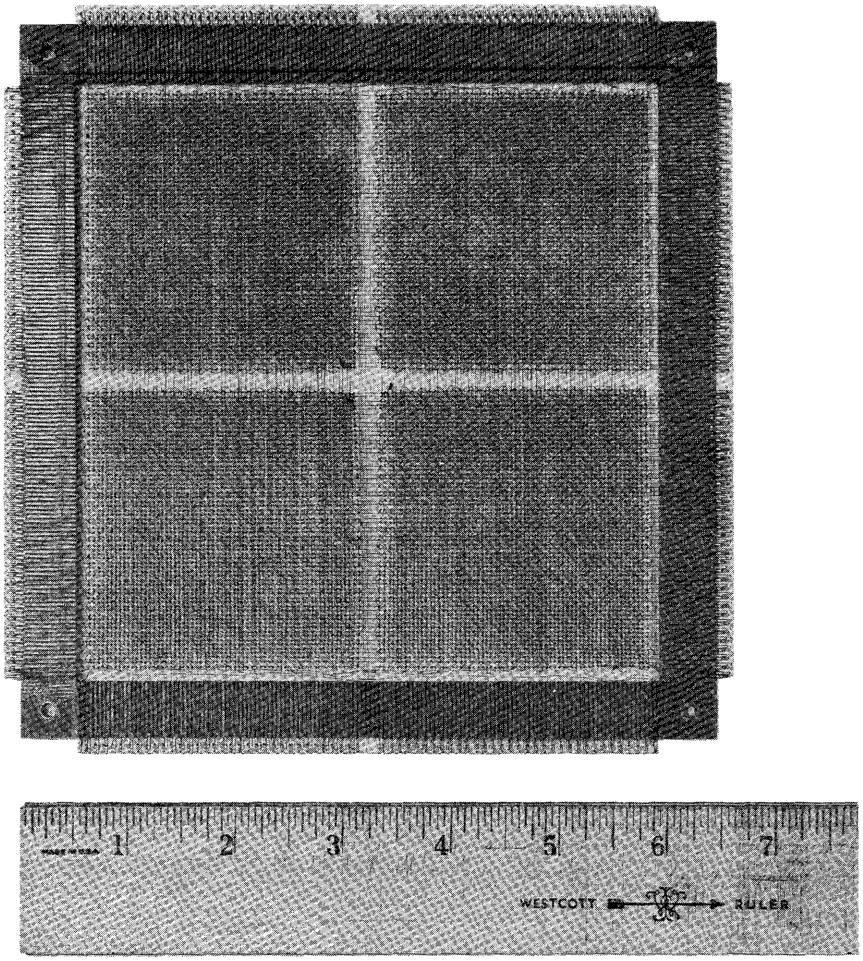


Figure 8-11. Memory wafer (four planes)

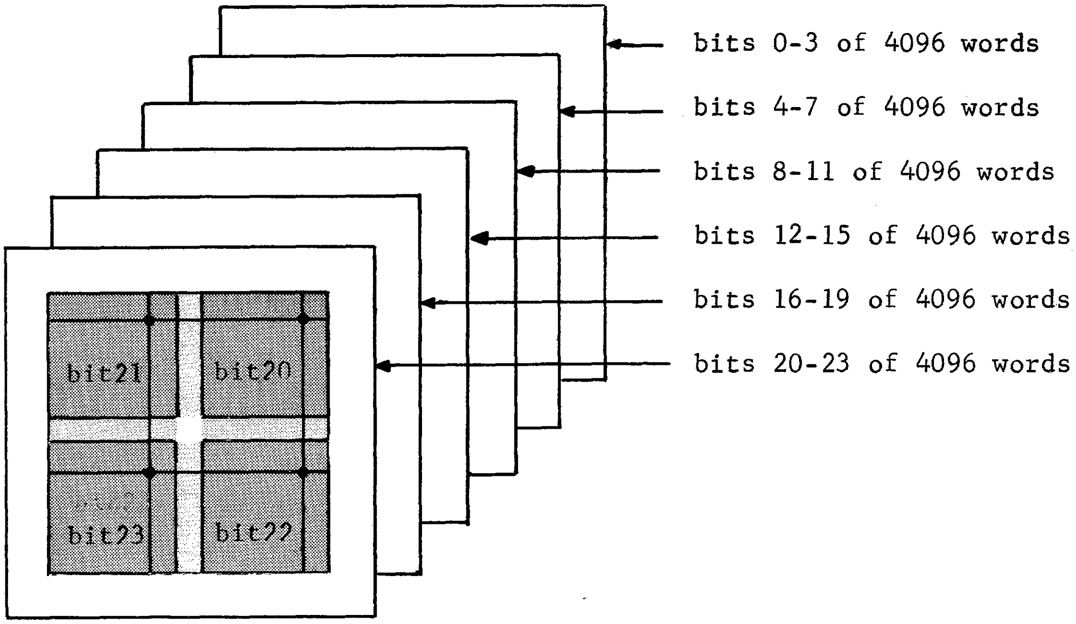


Figure 8-12. Field of 4096 words

Address Selection

Each of the 32,768 storage locations is referenced by a 15-bit address obtained from either the P or the F register. The address must select one of four memory stacks, one of the two fields in the selected stack, and one of 4096 locations in the selected field.

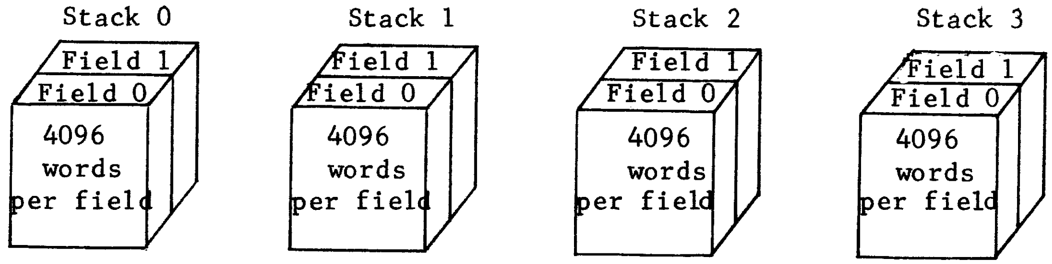


Figure 8-13. 32,768 word memory

The upper two binary bits of the address (2^{14} and 2^{13}) have four possible combinations and are used to select the stack containing the desired word. For example, address 51021_8 would reference a word contained in module 2 (upper two bits are 10).

00	stack 0	addresses 00000_8 to 17777_8
01	stack 1	addresses 20000_8 to 37777_8
10	stack 2	addresses 40000_8 to 57777_8
11	stack 3	addresses 60000_8 to 77777_8

The next bit of the address (2^{12}) is used to select one of the two fields in the selected stack. Using the same address (51021), the 2^{12} bit would be a logical "1" and, consequently, field 1 would be selected. Figure 8-14 illustrates how the upper three address bits are used to select one of the four stacks and one of the two fields within that stack.

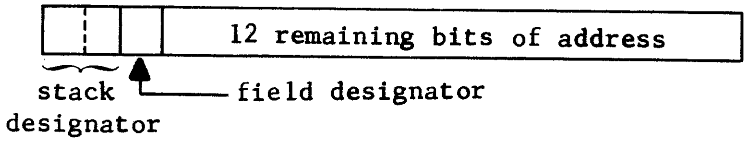


Figure 8-14. Stack and Field selection

It is apparent that all addresses beginning with an octal 5 would reference a word located in stack 2, field 1. Each of the other octal numbers would reference a different field (4 would also reference stack 2 but field 0).

The remaining problem is to select one of the 4096 words in the selected field with the remaining twelve address bits. To facilitate your understanding of the selection methods, examine the following mileage chart similar to one found in a United States road atlas.

The function of the mileage chart is to provide a reference of distances between cities. Total mileage or distance is found by tracing the name of one of the cities on the horizontal lines, and the name of the other city on the vertical column. The unique spot where the two lines intersect is the total mileage between the cities.

UNITED STATES MILEAGE CHART

594611	Atlanta, Ga.	Boston, Mass.	Cheyenne, Wyo.	Chicago, Ill.	Cincinnati, Ohio	Cleveland, Ohio	Dallas, Texas	Denver, Colo.	Des Moines, Iowa	Detroit, Mich.	Houston, Texas	Indianapolis, Ind.	Kansas City, Mo.	Los Angeles, Calif.	Louisville, Ky.	Memphis, Tenn.	Milwaukee, Wis.	Mpls.-St. Paul, Minn.	New Orleans, La.	New York, N. Y.	Omaha, Nebr.	Philadelphia, Pa.	Pittsburgh, Pa.	Portland, Oreg.	St. Louis, Mo.	Salt Lake City, Utah	San Francisco, Calif.	Seattle, Wash.	Toledo, Ohio	Tulsa, Okla.	Washington, D. C.	Wichita, Kans.
Akron, Ohio	680	669	1322	350	227	33	1193	1344	684	179	1287	291	783	2418	342	720	438	767	1063	474	824	396	101	2535	534	1786	2540	2468	122	956	335	978
Atlanta, Ga.		1084	1475	702	454	701	835	1440	899	724	842	510	805	2229	396	382	787	1092	518	876	1014	780	714	2697	554	1940	2543	2774	667	811	617	932
Austin, Texas	960	1986	994	1153	1139	1382	196	910	907	1361	163	1084	702	1402	1031	648	1226	1159	523	1780	859	1700	1419	2126	858	1315	1800	2208	1305	478	1563	575
Baltimore, Md.	681	403	1640	677	502	343	1441	1631	1007	507	1486	572	1061	2696	613	961	765	1104	1173	186	1144	97	222	2853	812	2104	2858	2795	450	1234	38	1256
Birmingham, Ala.	157	1209	1379	658	475	733	668	1312	837	745	678	487	722	2082	373	253	750	1051	365	992	930	907	751	2590	490	1812	2369	2667	688	683	753	804
Bismarck, N. Dak.	1533	1852	587	849	1156	1205	1175	689	672	1131	1419	1033	789	1659	1147	1256	771	431	1617	1676	584	1608	1316	1352	987	953	1646	1223	1095	973	1547	786
Boise, Idaho	2251	2729	762	1746	1959	2082	1612	840	1400	2008	1816	1849	1414	893	1941	1888	1763	1438	2108	2544	1262	2467	2184	445	1677	362	664	517	1972	1525	2412	1352
Boston, Mass.	1084		1962	994	855	651	1818	1984	1324	716	1888	925	1414	3049	966	1338	1078	1417	1575	217	1464	305	576	3194	1165	2426	3180	3108	760	1587	441	1609
Buffalo, N. Y.	893	459	1507	530	429	190	1389	1536	848	252	1495	485	970	2614	541	910	609	938	1270	366	983	355	220	2720	725	1971	2725	2653	305	1136	361	1165
Carson City, Nev.	2448	2966	1004	1983	2196	2319	1674	1059	1642	2245	1863	2086	1659	447	2178	2064	2000	1783	2175	2781	1504	2713	2421	610	1914	543	221	793	2209	1681	2649	1520
Charleston, S. Car.	301	946	1786	910	629	780	1120	1742	1195	867	1095	722	1090	2546	619	683	998	1327	731	740	1318	642	677	2997	836	2250	2860	3028	810	1128	505	1249
Cheyenne, Wyo.	1475	1962		976	1192	1315	867	101	638	1241	1112	1082	657	1169	1176	1123	996	805	1369	1777	500	1709	1417	1207	912	458	1220	1279	1205	770	1645	595
Chicago, Ill.	702	994	976		293	340	954	1025	339	268	1095	184	504	2128	298	538	89	408	930	814	474	746	451	2192	294	1443	2197	2118	233	703	672	699
Cincinnati, Ohio	454	855	1192	293		239	971	1169	578	255	1058	109	596	2226	111	481	382	701	816	639	698	571	284	2405	342	1656	2410	2412	198	751	492	791
Cleveland, Ohio	701	651	1315	340	239		1197	1344	668	164	1303	295	780	2422	351	720	429	748	1078	481	803	414	122	2528	535	1779	2533	2461	107	946	356	975
Columbus, Ohio	560	750	1248	310	106	141	1074	1234	625	185	1166	173	658	2299	218	596	399	718	928	544	755	466	182	2461	413	1712	2466	2428	128	824	392	853
Dallas, Texas	835	1818	867	954	971	1197		784	708	1176	242	899	503	1434	863	480	1027	960	501	1602	660	1534	1251	2053	659	1242	1769	2135	1119	279	1403	386
Denver, Colo.	1440	1984	101	1025	1169	1344	784		683	1287	1028	1059	606	1157	1130	1056	1042	851	1285	1768	546	1700	1417	1285	866	516	1274	1357	1249	686	1631	508
Des Moines, Iowa	899	1324	638	339	578	668	708	683		596	951	469	206	1821	572	605	356	252	1016	1144	135	1076	771	1851	335	1102	1856	1862	567	451	1015	401
Detroit, Mich.	724	716	1241	268	255	164	1176	1287	596		1290	277	748	2401	367	722	357	676	1091	628	731	576	280	2454	517	1705	2459	2387	57	928	515	943
Dubuque, Iowa	828	1171	823	181	470	521	888	869	184	449	1112	361	387	2006	475	596	172	256	1008	1005	319	927	632	2036	310	1287	2041	1950	414	632	866	582
Duluth, Minn.	1177	1479	958	480	783	832	1113	1004	403	726	1356	664	609	2077	778	972	405	150	1369	1303	515	1235	943	1806	675	1358	2104	1720	722	859	1174	803
El Paso, Texas	1465	2375	746	1444	1552	1748	621	654	1186	1727	756	1450	940	808	1474	1110	1545	1408	1117	2159	1066	2091	1808	1688	1210	877	1206	1770	1670	788	2022	750
Evansville, Ind.	414	1083	1081	292	225	463	763	1035	505	445	850	168	424	2053	125	277	381	683	669	877	621	799	509	2292	170	1543	2297	2367	388	579	717	619
Fort Wayne, Ind.	619	850	1121	156	151	193	1014	1144	471	161	1128	116	587	2239	218	561	245	564	942	661	606	593	300	2334	358	1585	2339	2275	104	767	532	782
Fort Worth, Texas	868	1851	844	981	1004	1224	31	760	735	1203	261	926	530	1374	896	513	1054	987	534	1645	687	1567	1284	2037	686	1219	1732	2109	1146	288	1436	385
Glacier Nat'l. Park	2246	2562	856	1564	1862	1910	1782	957	1426	1833	2028	1752	1483	1355	1865	1956	1491	1146	2291	2396	1278	2321	2028	666	1738	655	1222	576	1797	1648	2261	1489
Grand Canyon, Ariz.	1879	2706	839	1744	1862	2061	1063	783	1448	2020	1307	1766	1238	512	1782	1463	1811	1666	1569	2484	1327	2403	2117	1236	1496	390	836	1313	1978	1087	2331	1011
Harrisburg, Pa.	731	385	1608	645	470	311	1433	1599	975	475	1503	540	1029	2664	581	953	733	1062	1192	167	1112	101	189	2821	780	2072	2826	2763	418	1202	112	1224
Helena, Mont.	2100	2497	712	1459	1801	1850	1577	811	1251	1776	1839	1691	1308	1232	1804	1800	1430	1027	2078	2284	1120	2253	1961	686	1527	498	1137	610	1740	1440	2144	1256

Figure 8-15. Mileage Chart

The names of the cities are recognized by the appropriate letters of the alphabet aligned so they spell out the city being sought.

If the same principle of the mileage chart is utilized, the computer can determine a specific location by using the bits of the address to spell out the horizontal and vertical lines desired.

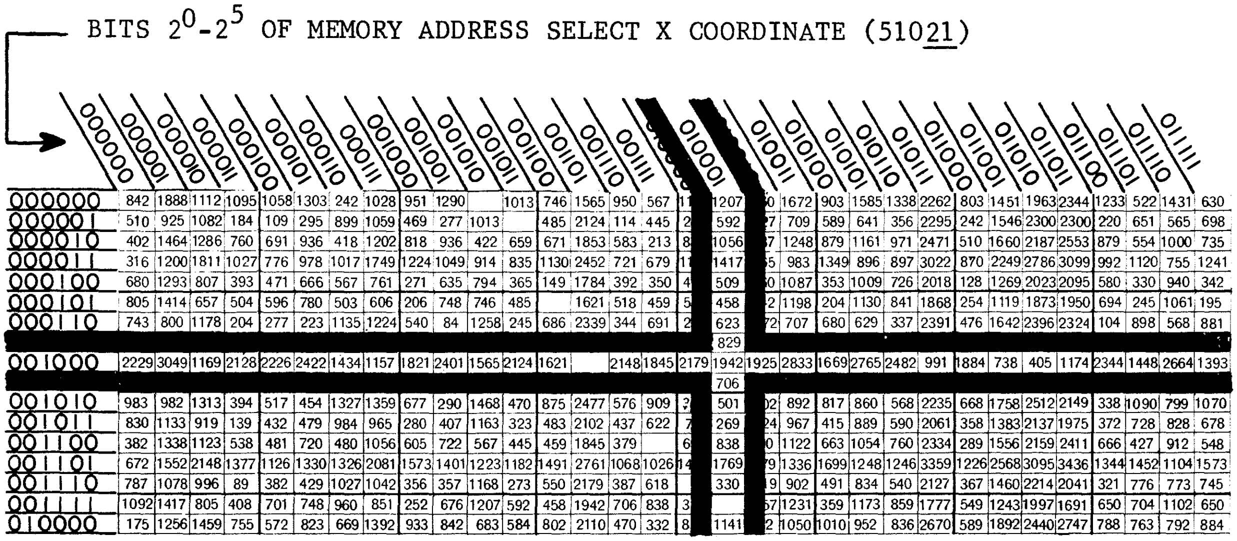

BITS 2^0-2^5 OF MEMORY ADDRESS SELECT X COORDINATE (51021)
bits 2^6-2^{11} of memory address select y coordinate (51021)
000000
000001
000010
000011
000100
000101
000110
001000
001010
001011
001100
001101
001110
001111
010000
1942

Figure 8-16. Memory Selection

By replacing the names of the cities with a 6-bit operand, the distances may still be found by locating the square coincident to two binary numbers.

The upper three bits of address 51021 were used to determine the stack and field that contains the desired word. The remaining twelve bits determine a location within the field. The lower six bits of the address (010 001) are used to determine the X coordinate and bits 2^6-2^{11} (001 000) determine the Y coordinate. The word being referenced is at the intersection of the two coordinates and, in Figure 8-16, is 1942. If all possible combinations of six bits are used, each could designate one of 64 X or 64 Y coordinates. The X and Y coordinates could intersect at 4096 different points and select one of the 4096 words in a field.

It was previously stated that each core has four wires passing through its center. Two of those wires are known as the X and Y drive lines which select that core. In order to read a 24-bit word from storage, there must be twenty-four points of intersection within the field. Figure 8-17 illustrates how the selected X and Y drive lines are connected in the field to provide the twenty-four necessary intersections and select the twenty-four cores.

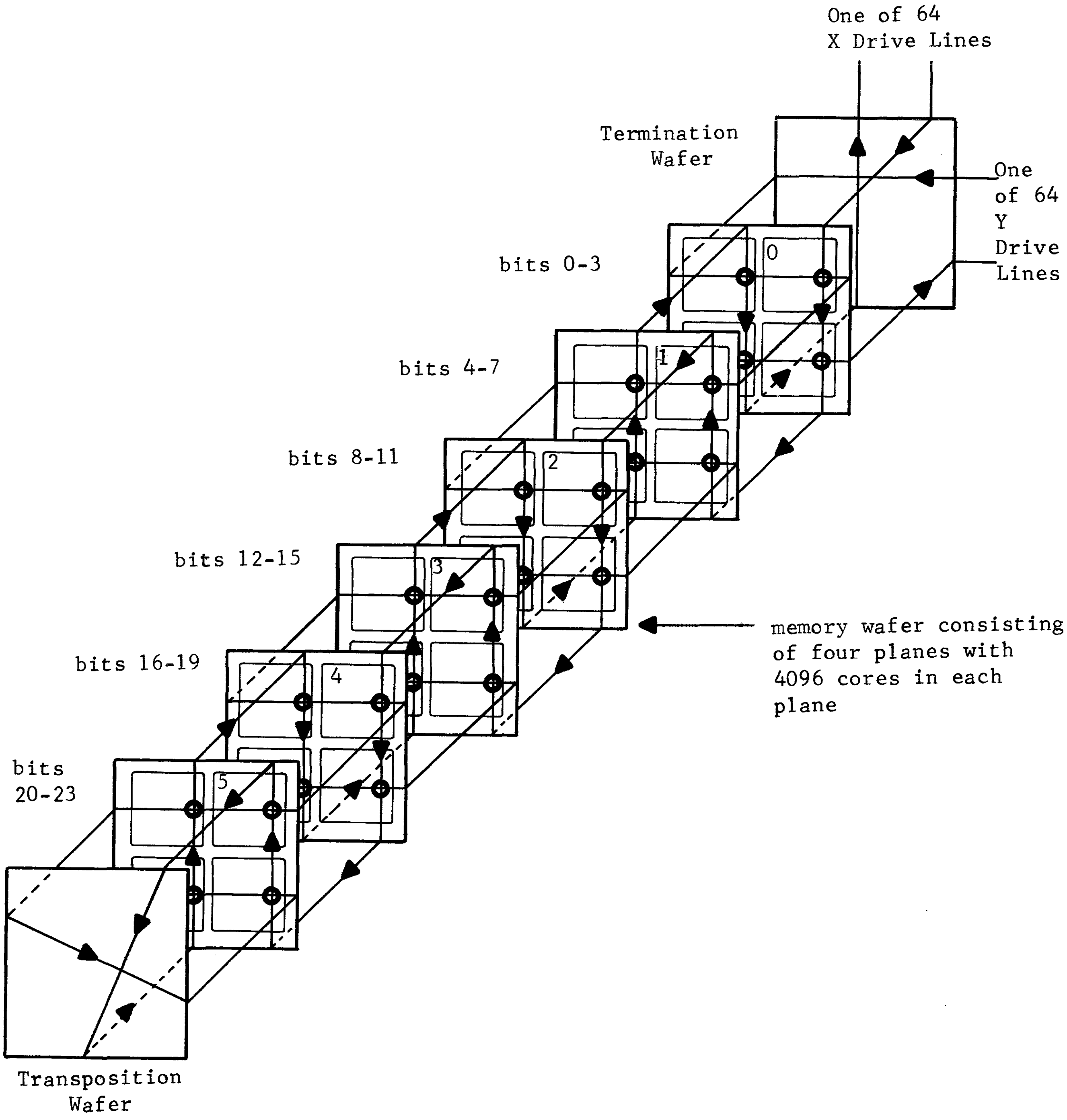


Figure 8-17. Selecting a word within a field

Although you now know how a 24-bit word is selected, the theory of magnetic core operation has not yet been discussed. Let's taste one of those little iron doughnuts (like mother used to make) and examine the recipe.

Magnetic Core Theory of Operation

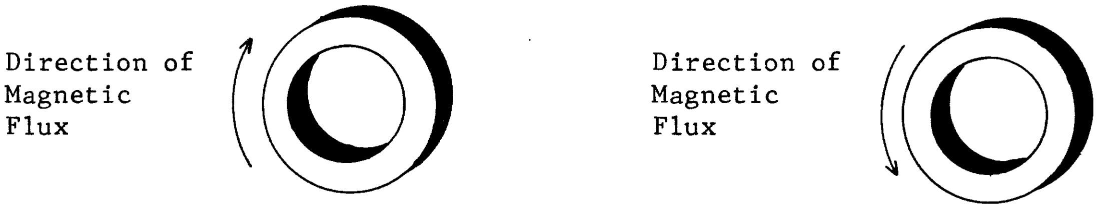
A core is capable of storing binary information as a result of three basic properties.

1) The core magnetizes when a suitable magnetizing force is applied.

2) The core remains magnetized after the magnetizing force is removed.

3) The core may be magnetized in either of two directions as determined by the direction of the magnetizing force.

These properties permit a magnetic core to be defined as a bistable device capable of storing a "1" or a "0", depending upon the direction of its residual magnetism.



Core in "1" State

Core In "0" State

Figure 8-18. Magnetized Cores

In order to apply a magnetizing force to a core it is necessary to cause current to flow in the drive lines that pass through it. The current through a drive line produces a magnetic field around the line and, if of sufficient magnitude, may switch the core to its opposite state.

The core in Figure 8-19a has a single drive line passing through its center. As electrons flow through the conductor (into the page), a magnetic field is produced in the direction indicated. If the direction of electron flow is reversed (out of the page), the magnetic field also reverses (Figure 8-19b).

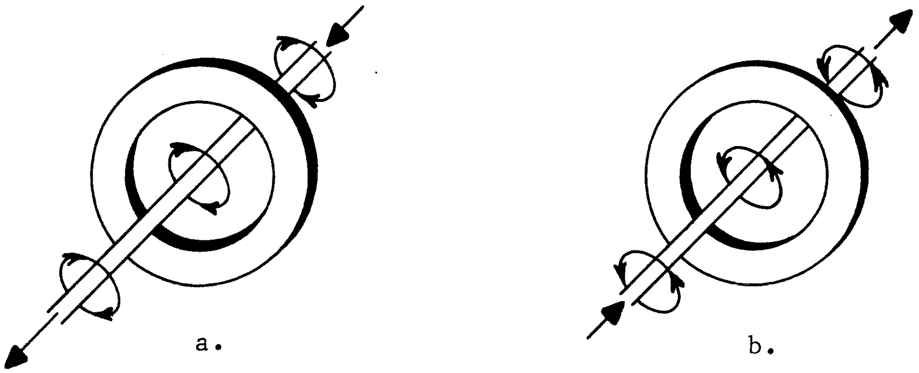


Figure 8-19. Magnetic Fields Produced by Current Flow

Enough current could be forced through the conductor to generate a magnetic field of sufficient strength to switch the core. However, each drive line passes through 128 cores (two planes) and all cores would be affected instead of the desired number of one per plane (Figure 8-17).

Instead of passing only one drive line through a core, two lines are used and pass through at right angles to each other. The core is switched only if both drive lines have current flow through them. Each plane would have only one core coincident to any two given drive lines (Figure 8-20).

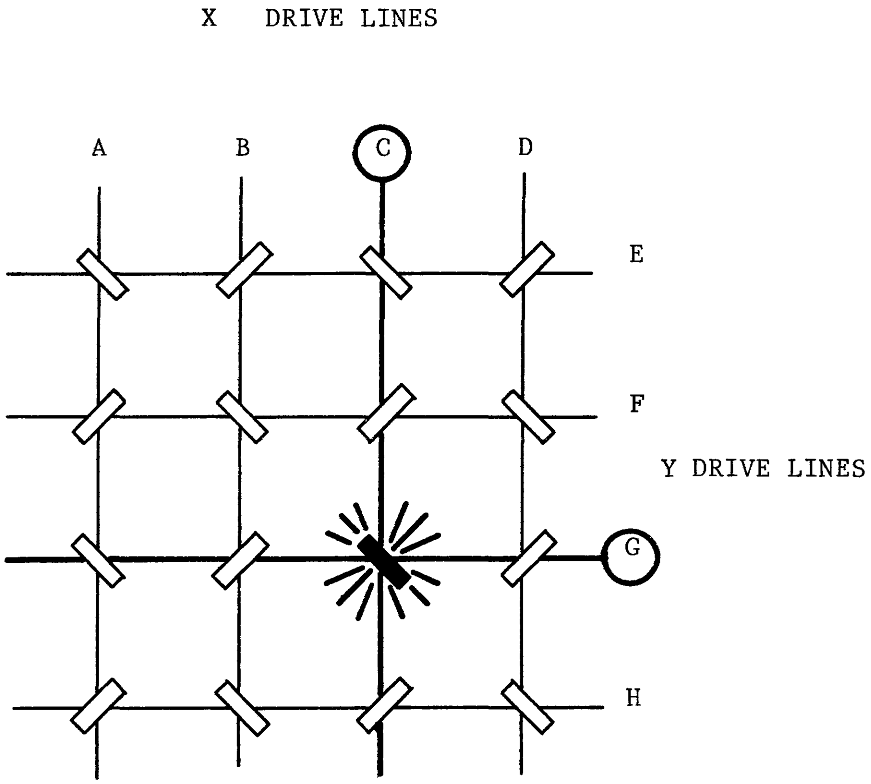


Figure 8-20. Magnetic Cores of a Bit Plane.

Address CG would cause current flow through drive lines C and G. Only one core has both drive lines passing through it, and would be the selected core in that plane. Notice that three other cores along drive line C and three along drive line G will be somewhat affected by the magnetic field of a single drive line.

Each current carrying drive lines produces slightly more than half the magnetic field intensity (flux) required to switch a core. Although half-selected cores are not affected, the core at the intersection of two driven lines may be switched by the two aiding fields.

The Hysteresis Curve

The manner in which a core magnetically responds to the application and removal of a magnetizing force is graphically represented by a hysteresis curve. The two quantities taken into consideration are:

1) An "H" value representing the field intensity caused by drive current. Since drive current may occur in either direction along the drive line(s), the resultant field intensity may be positive or negative. The magnitude and polarity of field intensity (magnetizing force) is measured along the horizontal axis of the hysteresis curve (Figure 8-21).

2) A "B" value representing the flux density of the magnetic field generated by the core itself. Since the core may be magnetized in either direction, flux density may be positive or negative. The magnitude and polarity of flux density (core magnetization) is measured along the vertical axis of the hysteresis curve (Figure 8-21).

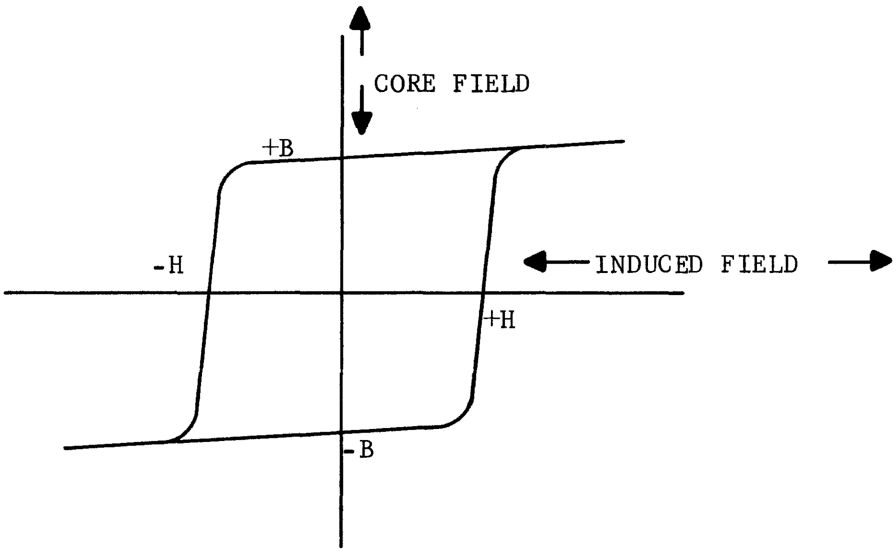


Figure 8-21. Hysteresis Curve for a Ferrite Core

A hysteresis curve is obtained by plotting flux density, B, as a function of field intensity, H. Time is not used in determining the curve and the speed with which cores react to drive currents must be discussed separately. The rectangular nature of the hysteresis curve for a ferrite core indicates the operating properties which make it a practical storage element.

Switching a Core

A magnetic core in the "1" state would have the residual magnetic properties illustrated in Figure 8-22 as BR. No current is flowing through the drive lines and the magnetism of the core is not being influenced. The core is actually a permanent magnet with positive residual magnetism.

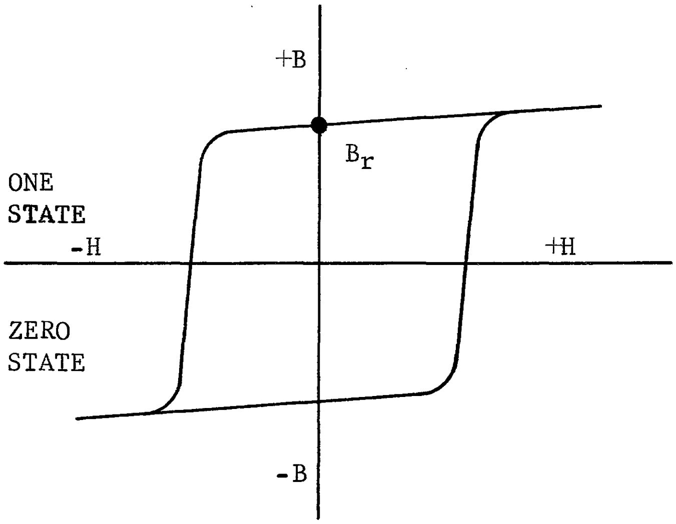


Figure 8-22. Residual Magnetism of a Core in the "1" State

When read current is applied to only one drive line through the core, it is said to be half-selected and the field intensity is expressed as -½ H. The magnetic field opposes the state of the core but is not of sufficient magnitude to switch the core (Figure 8-23). When the -½ H field intensity is removed, the flux density of the core returns to the Br level but at a slightly lower value than before. Repeated application and removal of -½ H field intensities does not significantly reduce the Br level of the core.

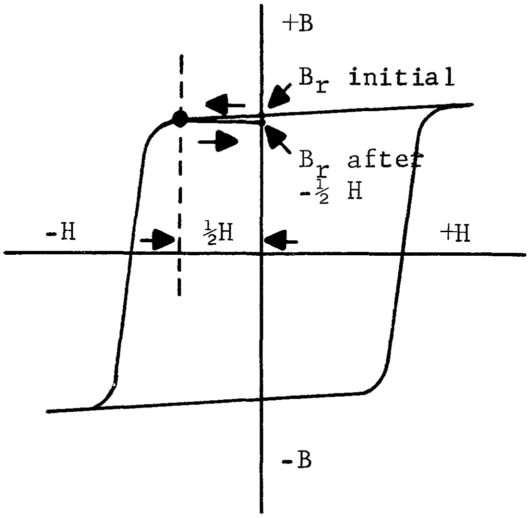


Figure 8-23. Response of a Core in the "1" State to a ½ H Field Intensity.

If a full read current (current in both drive lines) is employed, a field intensity of -H is generated. Flux density is altered as indicated on the hysteresis curve (Figure 8-24). Note that core magnetization passes the knee of the loop as positive flux density is reduced to zero and negative flux density rises to a saturated level $-B_s$. Immediately after the knee of the loop is passed, small increases in field intensity cause large increases in flux activity. When the -H field intensity is removed, flux density returns to a residual level of $-B_r$. Once again the core may be considered a permanent magnet but its negative residual magnetism is characteristic of the "0" state. Full read current unconditionally drives cores to the "0" state.

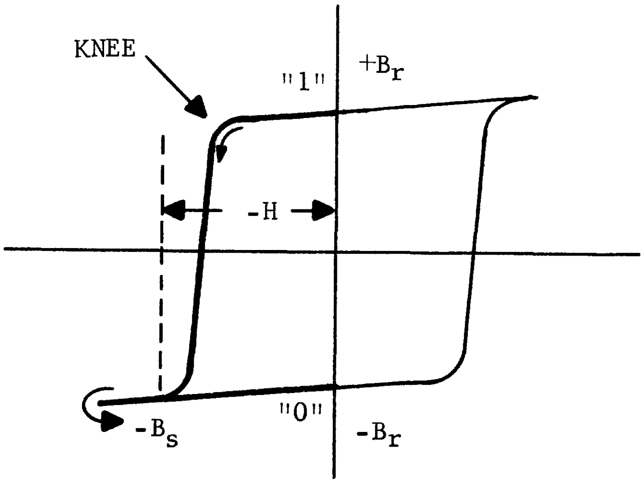


Figure 8-24. Core Response to -H Field Intensity

If full read current is applied to a core already in the zero state, the magnetic field produced by the drive lines is in the same direction as that of the core and the core does not switch states (Figure 8-25). When the read current is removed, the core returns to the $-B_r$ level.

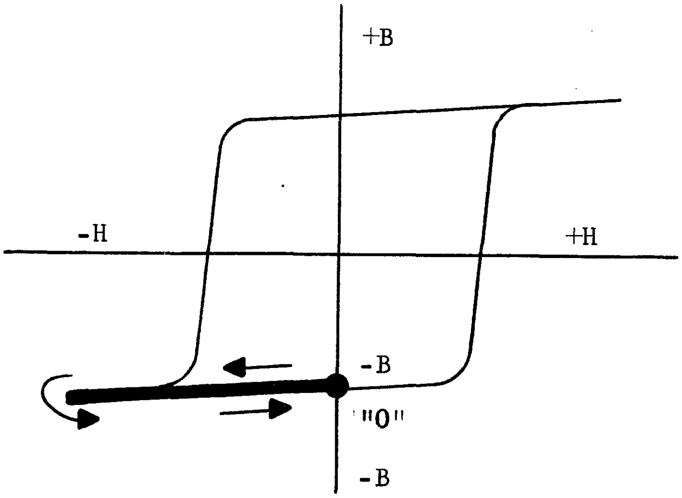


Figure 8-25. Response of a core in the "0" State to -H field intensity

Reading a Stored Word

The entire concept of magnetic core storage centers around the fact that a core produces a discernable magnetic field as it switches. Read current always switches all "one-state" cores of the selected word to the zero state whereas cores already in the zero state are not switched. Because a core is a closed magnet, its magnetic field is virtually non-existent when the core is in the remanant one or zero state. However, as the core is switched, flux density surrounding it rapidly increases (Figure 8-26).

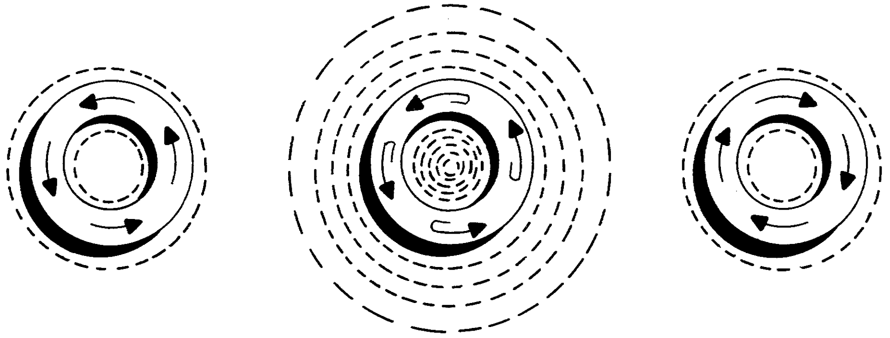


Figure 8-26. Flux Density

Flux activity at the core during read time is indicative of whether the core represents a "1" or a "0". The flux activity is sensed by a third wire through the core called the sense line (Figure 8-27).

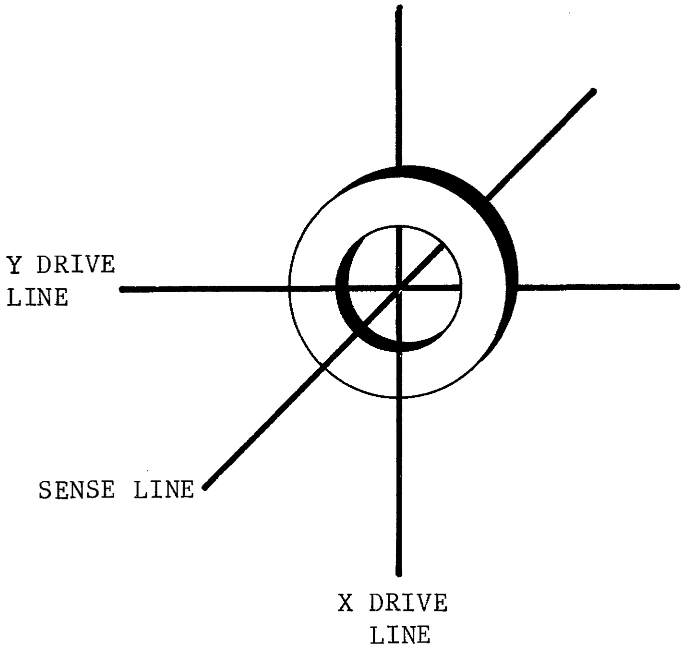


Figure 8-27. Core with Drive and Sense Lines

If a core switches from a "1" to a "0" during read time, the resulting magnetic field induces a voltage of a few millivolts into the sense line. A core already in the "0" state at read time produces very little flux activity and the induced voltage into the sense line is much smaller in duration and amplitude (Figure 8-28).

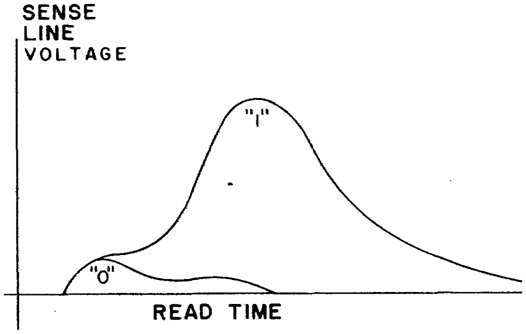


Figure 8-28. Sense Line Voltage During Read Time for Both a "1" and a "0" Core.

Each of the four planes of each wafer has a separate sense line threaded diagonally through all 4096 cores. Because only one core on the plane is being read, an induced voltage into the sense line indicates that the referenced core originally contained a "1". If the sense line were not threaded diagonally, it would be parallel to one of the drive lines and the magnetic field produced by the drive line would be induced into it. The sense line is threaded in such a manner that the flux activity of the 127 half-selected cores in the plane is effectively cancelled to zero. Figure 8-29 illustrates how the sense line is threaded through a plane.

Both ends of the sense line are connected to a differential amplifier called a sense amp. Twenty-four sense amplifiers are associated with each field and detect the state of the twenty-four bits of the selected word. The output of the sense amplifier is a logical "1" as a core switches from a "1" to a "0" and outputs a logical "0" at all other times.

The outputs of the sense amplifiers are gated directly to the previously-cleared Z register via a ones transfer. After the word has been read, all cores at that memory address are in the zero state, but the word is duplicated in the Z register (Figure 8-30).

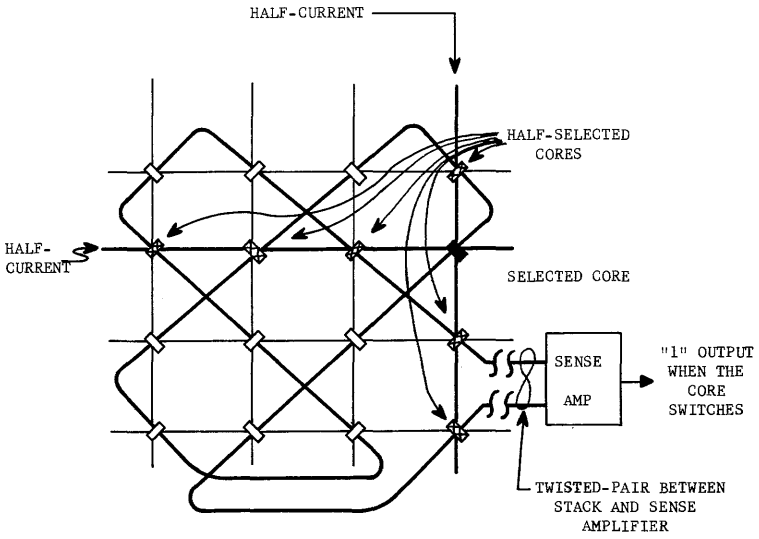

HALF-CURRENT
HALF-SELECTED CORES
HALF-CURRENT
SELECTED CORE
SENSE AMP
"1" OUTPUT WHEN THE CORE SWITCHES
TWISTED-PAIR BETWEEN STACK AND SENSE AMPLIFIER

Figure 8-29. Sense Line Wiring Through a Plane.

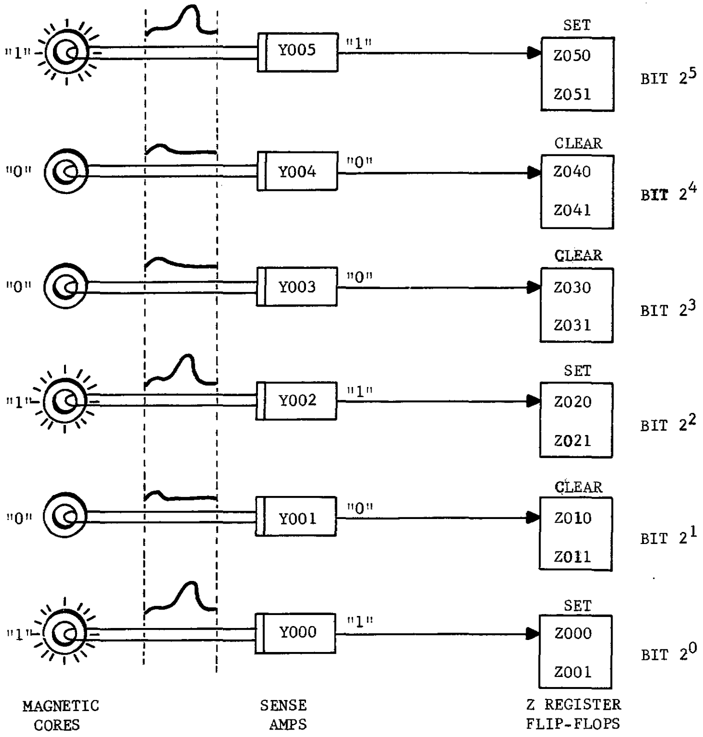


Figure 8-30. Transfer of Lower Six Bits of a Memory Word to the Z Register.

Now that the word is in the Z register, it may be transferred either to the X register, as an operand, or to the F register, as an instruction.

Because the stored word was destroyed during the read cycle, it must now be restored back in memory at the same address. This is accomplished by reproducing the contents of the Z register back in memory through the restoration network (Figure 8-31).

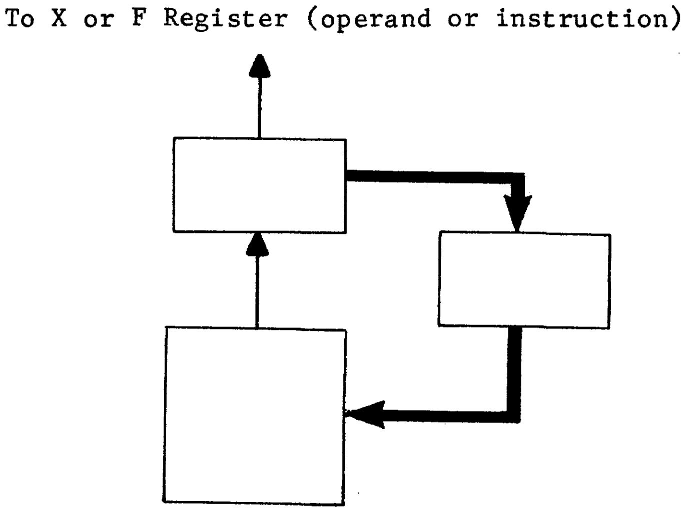


Figure 8-31. Restoration of a Storage Word

Restoring the Word in Memory

During the reading of the memory word, all cores at the selected location were switched to the zero state by the read drive current. A reversal of drive current (write current) would switch all cores to the one state. The only problem encountered during restoration is preventing those cores originally in the zero state from being switched to a one when the drive current is reversed. This is accomplished with the fourth wire through each core, called the inhibit line (Figure 8-32).

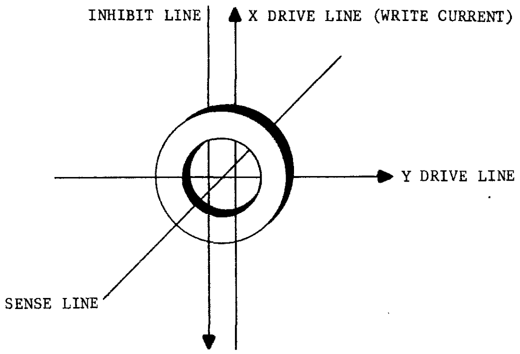


Figure 8-32. Magnetic Core with Write, Sense, and Inhibit Lines

The inhibit line always passes through the core parallel to either the X or Y drive line. If the core was originally in the zero state, a current generator causes current to flow through the inhibit line in the opposite direction to current flow through the parallel drive line. Because the currents through the two parallel lines are equal in amplitude, their opposing magnetic fields cancel each other (Figure 8-33). The result is that a total field intensity of only $+\frac{1}{2}$H is produced, insufficient to switch the core to a one.

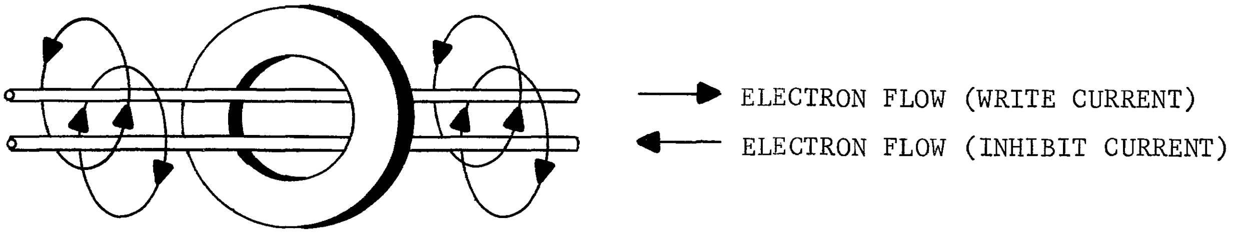

ELECTRON FLOW (WRITE CURRENT)
ELECTRON FLOW (INHIBIT CURRENT)

Figure 8-33. Cancelling Effect of Inhibit and Half-Write Currents

The inhibited core is now essentially the same as any other half-selected core along either drive line. The inhibit current is produced by an inhibit generator controlled by the information in the Z register. If a Z register flip-flop is set after the reading of a memory word, it indicates that its associated core was in the one state and should return to that same state. Cleared flip-flops in the Z register are indicative of a zero-state cores and the core must be prevented from being switched to a one by write current.

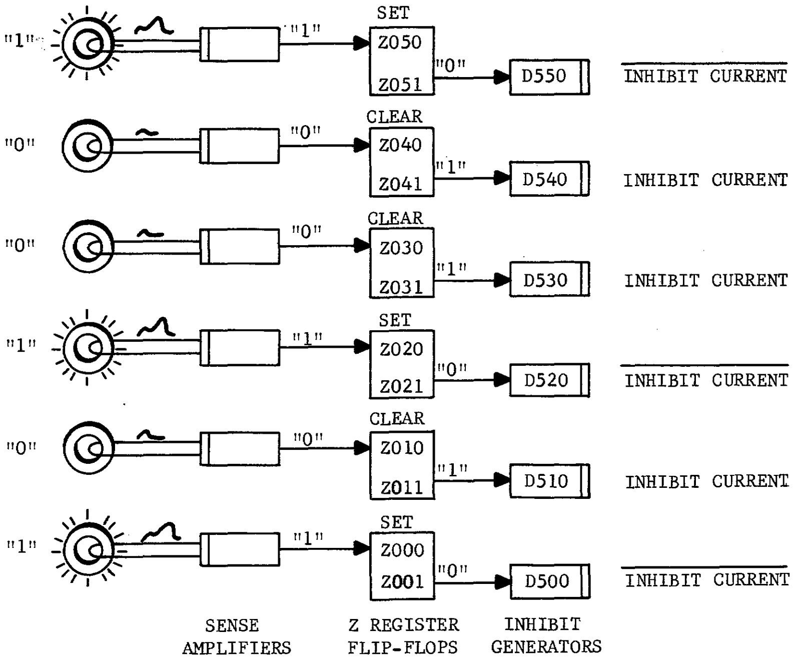


Figure 8-34. Restoration of a Memory Word with Inhibit Currents

Figure 8-34 illustrates how the inhibit generators are connected to the Z register. Each plane has its own inhibit line and associated generator. A logical one into the inhibit generator produces inhibit current and the associated core remains in the zero state.

It should now be apparent that the process of reading and restoring a word in memory must be accomplished in a definite logical sequence and that the timing of perations is quite critical. When a word is desired from storage, a memory cycle is initiated that synchronizes operations and generates the necessary timing pulses.

Memory Cycle

A complete memory cycle is initiated each time a word is read from or written into memory. The control section of the computer generates a Memory Request, which collapses a charged delay line to generate the memory-associated timing pulses. The memory cycle is divided into two main events; the read cycle when the word at the selected address is read and the write cycle when the word is restored back in the same location. Figure 8-35 illustrates the two major portions of a memory cycle and the reversal of current through the drive lines.

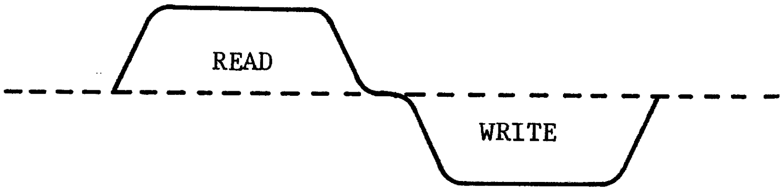


Figure 8-35. Computer Memory Cycle

A Memory Request signal sets the Pulse Generator flip-flop, applying a logical "1" to the delay line driver causing its output to drop from -10 volts to approximately ground potential. As the delay line collapses and the voltage decays along its length, a sequence of timed pulses is tapped off to initiate the memory orientated operations (Figure 8-36).

The set-side output of the Pulse Generator flip-flop also has two other functions:

1) Clears the Z register prior to the receipt of the memory word.

2) Transfers the address of the desired word to the S register.

The Z register must be cleared because the outputs of the sense amplifiers will be gated to it via a ones transfer. A ones transfer is always made into a previously cleared register.

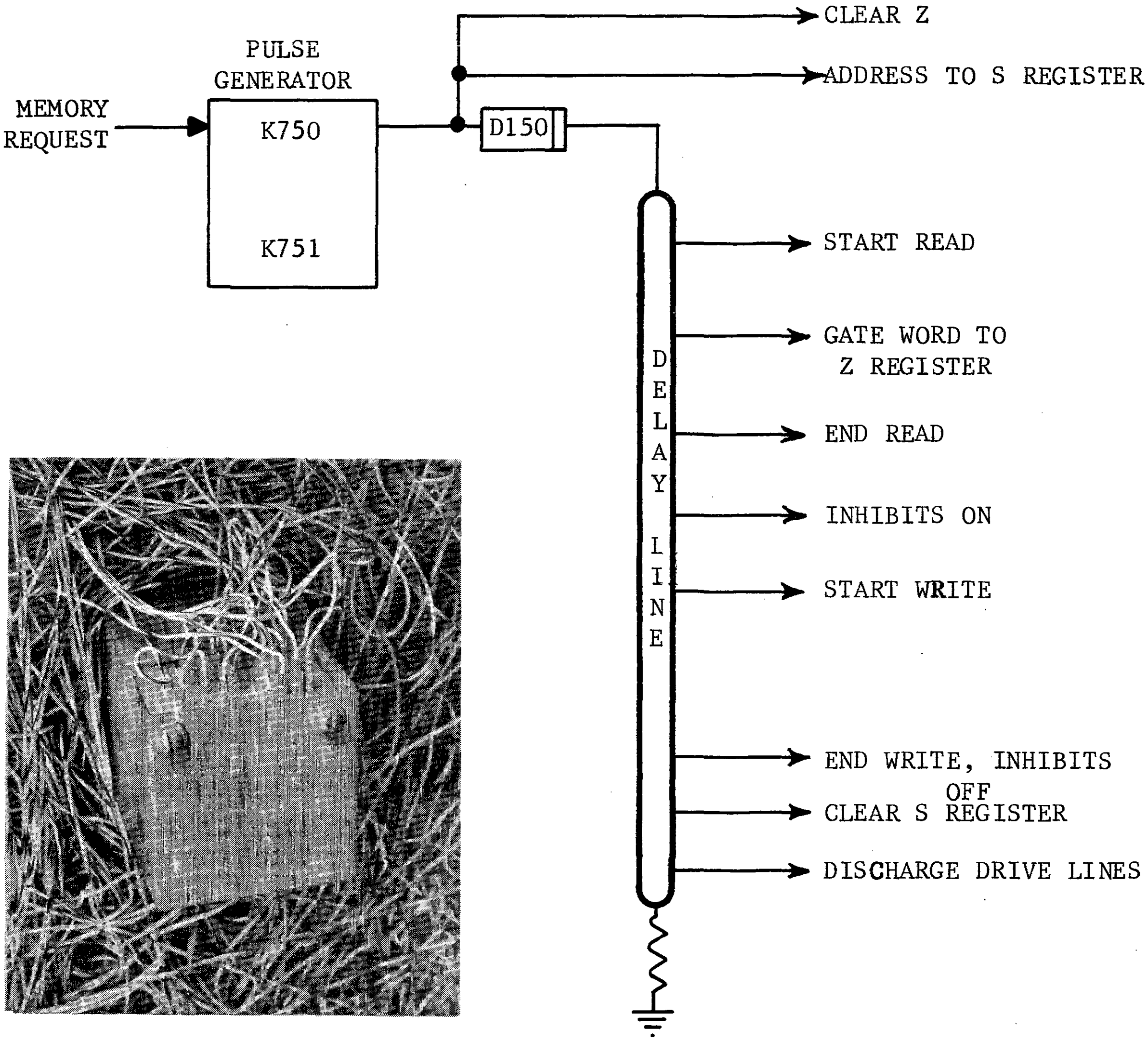


Figure 8-36. Memory Delay Line

If the memory reference is to read an instruction, the address is obtained from the P register. If an operand is to be read, the address is obtained from the lower 15 bits of the F register (address field). In either case, the address must be in the S register before memory is initiated. Figure 8-37 illustrates the function of the Pulse Generator flip-flop.

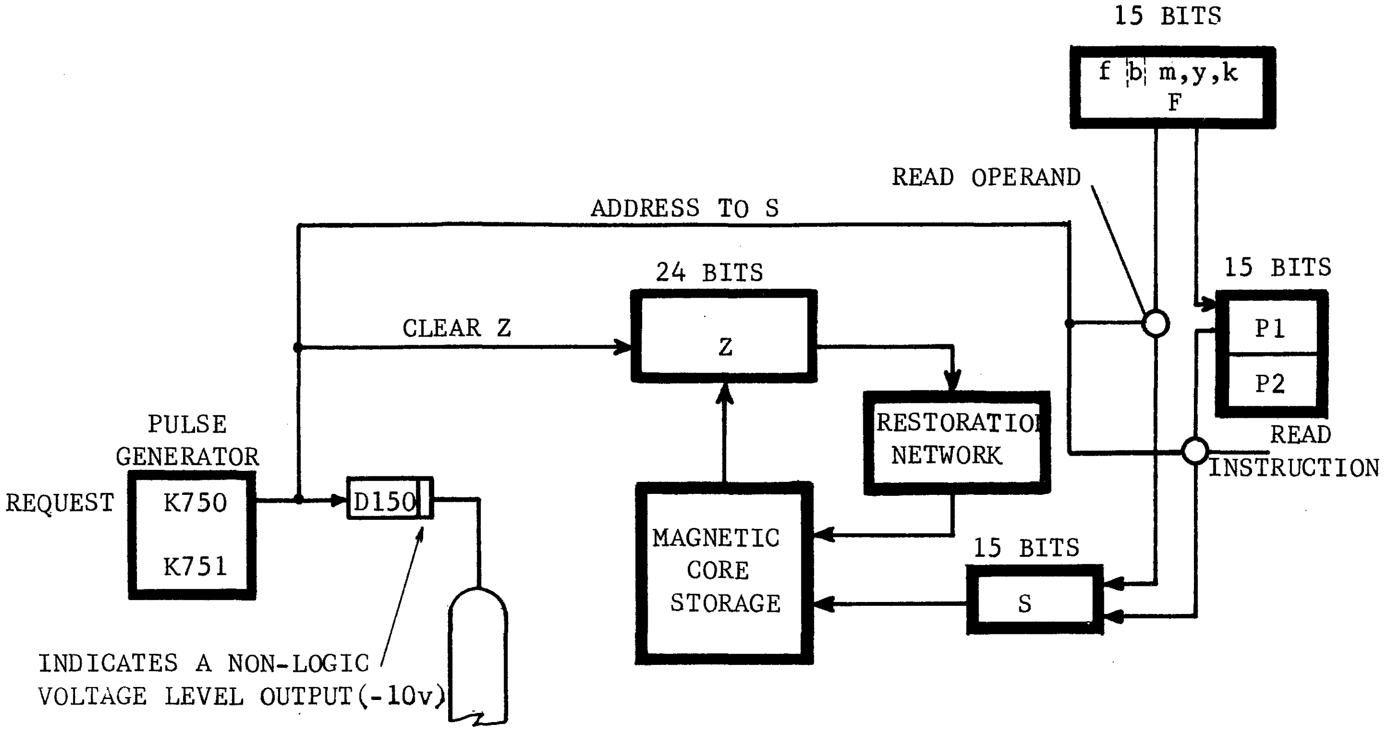


Figure 8-37. Initiation of a Storage Request

Start Read Clear Pulse Generator Flip-flop

The first pulse from the collapsing delay line results in the generation of read drive currents. The current is generated by a drive line transformer con to one end of each drive line. The other end of the selected X and Y drive lines is grounded through a 120-ohm resister, providing an impedence mismatch that results in 340 ma of current through the line. The coincident X and Y currents switch the one-state cores of the 24-bit word to zeros, inducing a voltage into the sense lines. Figure 8-38 shows how the "start read" pulse generates drive current and switches the core to the zero state. The pulse generator flip-flop is now cleared, having been set for 100 nanoseconds. The delay line starts to recharge at the top end which results in a 100 nanosecond pulse traveling along its length.

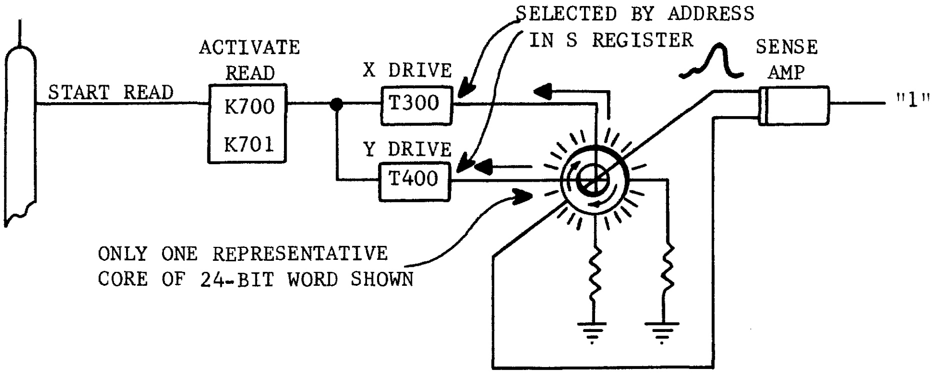


Figure 8-38. Generation of Read Current

Gate Word to Z register

About 200 nanoseconds after read current had been applied to the X and Y drive lines, the cores at the selected address start to switch. Zero-state cores do not switch and the voltage induced into the sense line is smaller and of shorter duration than that produced by the switching one-state cores (refer back to Figure 8-28). The "Gate Word to Z register" pulse from the delay line is reduced to a 60 nanosecond strobe by a special pulse-shaping circuit. By waiting until after flux activity of zero-state cores has subsided before generating the strobe pulse, only those cores originally in the ones state set corresponding bits in the Z register. Figure 8-39 illustrates how the strobe pulse gates the sense amplifier outputs to the Z register. The time relationship of the strobe pulse to the sensed voltage is also shown.

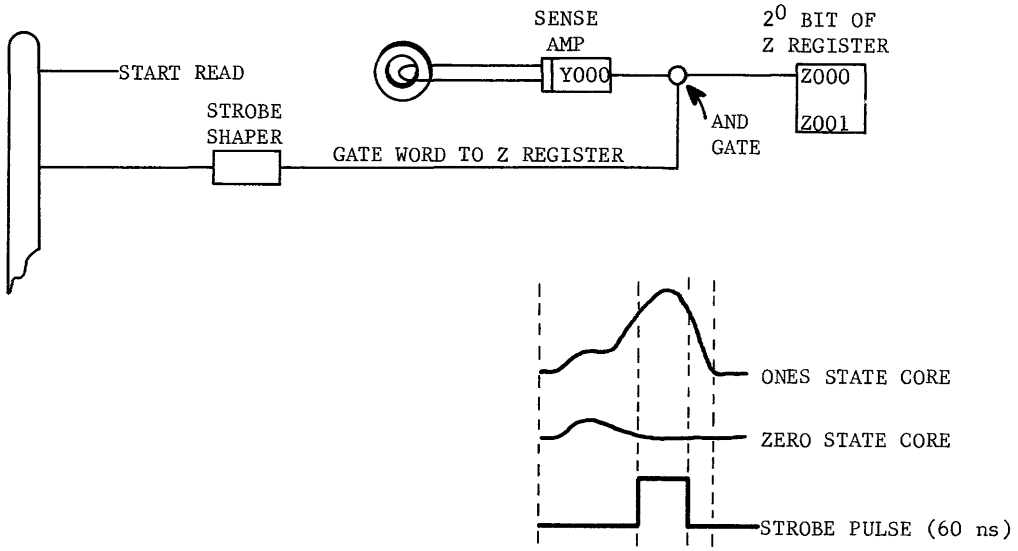


Figure 8-39. Strobe Memory to Z

End Read

Now that the selected memory word is reproduced in the Z register and all cores are in the zero state, the read current may be turned off. This is accomplished by clearing the same flip-flop that started the read portion of the memory cycle. The X and Y drive line transformers no longer have an input and read drive current ceases after about 425 nanoseconds (Figure 8-40). Note: A computer with larger cores in its memory would require a longer read cycle due to increased switching time of the greater core mass.

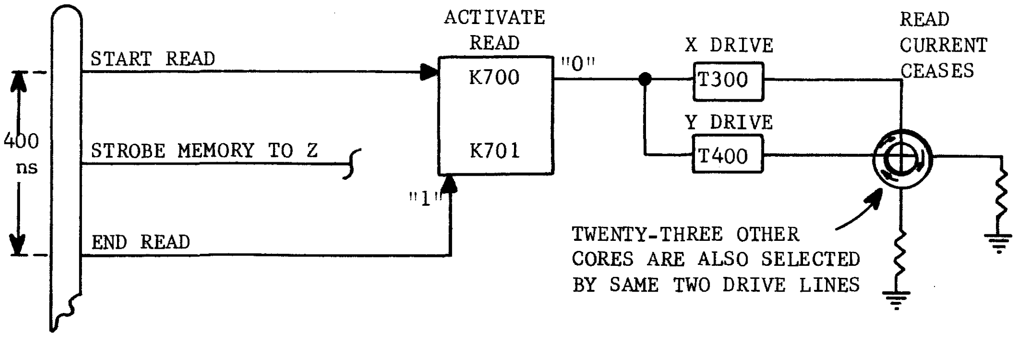


Figure 8-40. Termination of Read Current.

Inhibits On

The selected word is now contained in the Z register flip-flops. In addition to being used by the computer as an operand or an instruction, the word must also be restored back in its same location. If a zero is to be restored, an inhibit generator must be turned on to cancel out +½H and prevent the core from being switched to a one by the write drive current (refer back to Figure 8-33). The appropriate inhibit generators are turned on a few nanoseconds before write drive current. This insures the buildup of inhibit current before write current, insuring that no cores are accidentally switched to a one. Remember that inhibit current always opposes write current and is equal to ½ H (Figure 8-41)

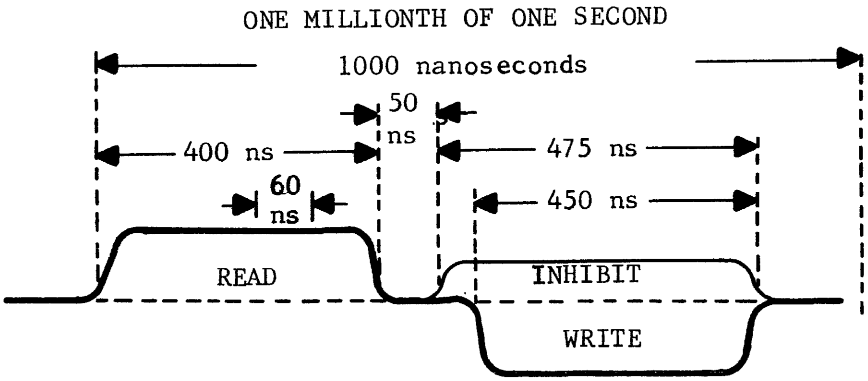


Figure 8-41. Time Relationship of Memory Cycle Operations

The clear side output of each Z register flip-flop is ANDed with the 475 nanosecond "Inhibit enable" signal. Inhibit current will be generated if the Z register flip-flop is cleared during that 475 nanosecond period (Figure 8-42).

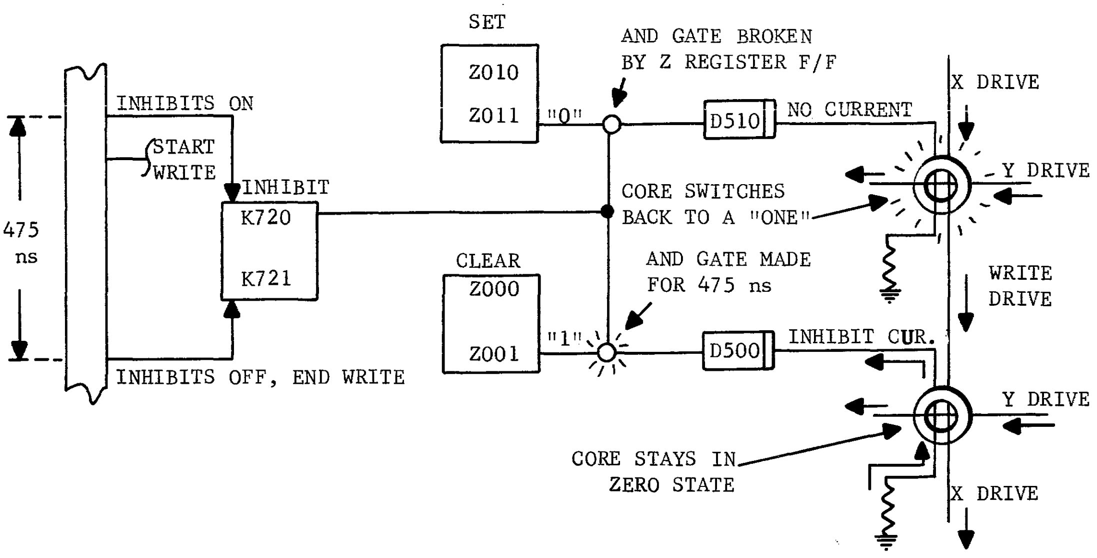


Figure 8-42. Inhibit Current Generation

START WRITE

The appropriate inhibit generators have been turned on and full write current is now applied to all cores of the selected word. If inhibit current is opposing the full write current of a given core, the write current is effectively reduced to + ½ H and the core remains in the zero state. The write current is in in the opposite direction of read current. This is accomplished by reversing current flow through the drive line transformers (compare Figures 8-38 and 8-43).

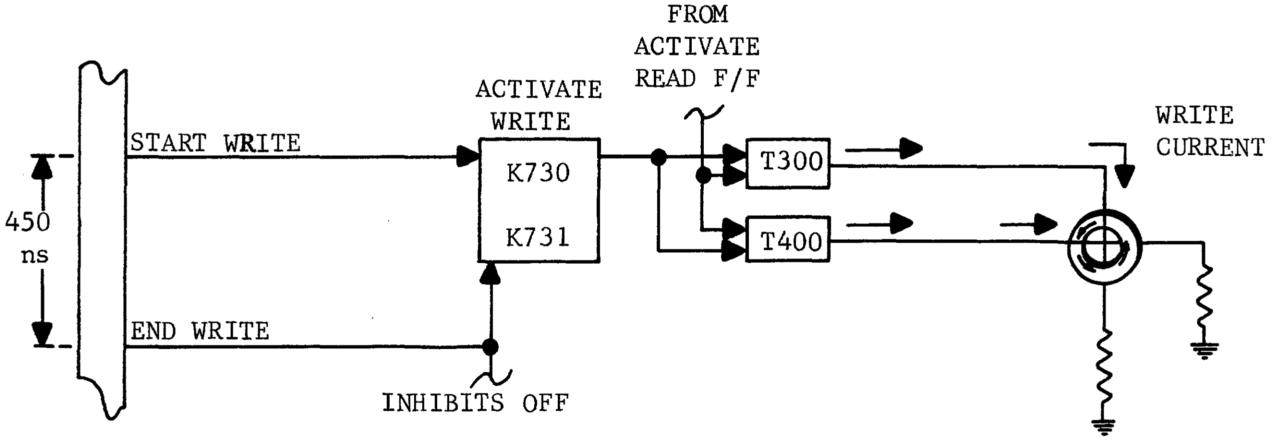


Figure 8-43. Generation of Write Current

END WRITE, INHIBIT OFF

These two signals clear the Activate Write and Inhibit flip-flops which, in turn, remove write and inhibit current from the lines. Flip-flops were used because the signals were required for approximately 400-450 nanoseconds in duration. Figures 8-42 and 8-43 show how the Activate Write and Inhibit flip-flops are cleared.

Clear S Register

The memory cycle is now essentially complete and the S register need no longer hold the address of that referenced location. To save time at the start of the next memory reference, the S register is cleared to all zeros in preparation for the ones transfer of the new address.

Discharge Drive Lines

The next memory request may be only a few nanoseconds after the one now being terminated. Although write current has been terminated, some time is required for runt pulses and noise caused by the switching cores to subside. That noise could adversely affect the reading of the next word, if allowed to decay at its own leisure. For this reason, all transformer cards are grounded shortly after removal of the write current. This action forces the instantaneous discharge of all drive lines and noise is quickly eliminated.

The entire memory cycle is now complete. The word was read from memory to the Z register where it could then be used as either an operand or an instruction. It was also restored back in its original location during the write cycle. One other operation must now be examined -- that of writing a new word in memory.

WRITING INTO MEMORY

Whether a word is being read from or written into memory, the same delay line generates the same signals and goes through the same Read and Write portions of the memory cycle. The only difference between the two operations is the source of information going into the Z register. Remember, the Z register determines what will be restored back in the referenced location.

If a new word is to be stored in memory, as during the execution of a Store A instruction, it is transferred to the X register and a Memory Request is issued for a store operation. The read portion of the memory cycle switches all cores at the selected location to zeros, exactly the same as when a word was being read from memory. The sense amps output lobical ones for those cores that contained ones, but the AND gates between the sense amplifiers and the Z register flip flops are broken. However, the same strobe pulse that gates the outputs of the sense amplifiers to the Z register now makes AND gates which transfers the contents of the X register. Figure 8-44 illustrates how one bit of an operand in the X register is transferred to the Z register by the strobe pulse.

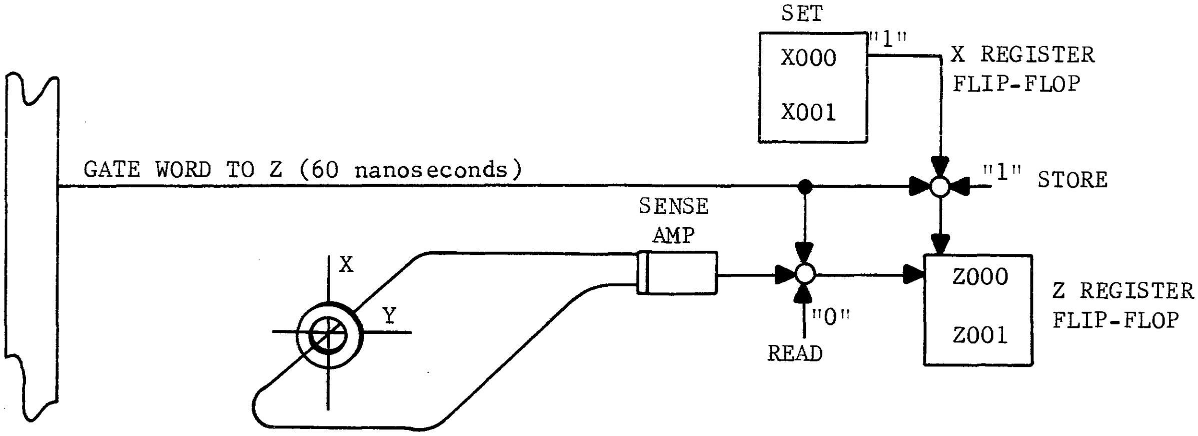


Figure 8-44. Storing a Word in Memory.

The read portion of the memory cycle ends and the word from the X register is now contained in the Z register. The Z register enables the appropriate inhibit generators and allows the new word to be stored in memory (refer back to Figure 8-42).

You should now understand why the same circuitry may be used for both reading from and writing into memory. The current instruction in the F register feeds the Function Translators which, in turn, determine whether the Memory Request is to read or store a word. For example: an ADA instruction requires an operand from storage; whereas a Store A instruction causes an operand to be written into storage.

The type of magnetic core storage explained in the foregoing test is known as a random access, co-incident current, destructive readout, four-wire magnetic core storage. The following diagram illustrates the complete circuitry required to read or store one bit of a 24-bit memory word.

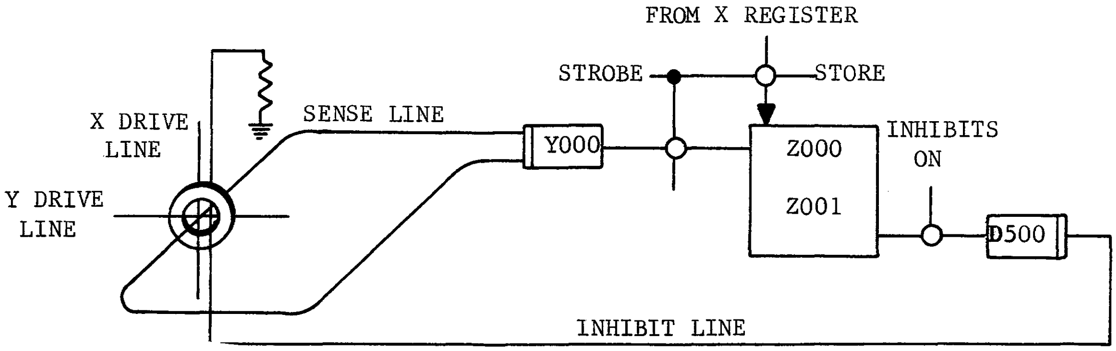


Figure 8-45. Complete Memory Circuit.

Several other types of random access storage devices exist or are being developed. Some memories have non-destructive readout capabilities which eliminates the necessity of restoring the word being read. Some storage devices use components other than magnetic cores for storing bits of information. Appendix A explains the theory of a random access, word organized, non-destructive readout, two-wire, thin-film memory.

SUMMARY

Several types of storage devices have been discussed in this chapter. The sequential access devices, such as the disk and drum, involved mechanical motion to position data and/or read heads before the information could be read. The disc is one of the very popular storage devices with modern super-computers because of its economical mass storage capability. Millions of bits of information may be stored on one disk memory device.

The main storage area of a digital computer is usually of the random access type because of the speed advantage during information access. The device may be magnetic core or thin-film, coincident current or word organized, and either destructive or non-destructive in reading. Although the cost per bit of these purely electronic storage devices is high, they are very reliable and allude to a faster computer system.

The four-wire magnetic core system explained in this chapter is quite common in present day computer systems. The two drive lines were used for both reading and restoring (or writing) a word by reversing the direction of current flow. The sense line is used only during the read portion of a memory cycle when the cores are switching to zero. The inhibit line is used only during the write portion of a memory cycle and then only if that bit is to contain a zero.

A memory cycle is initiated by a Memory Request from the Control section of the computer or from an I/o device. Once initiated, memory operations are automatically controlled by sequential pulses tapped off a delay line. The memory cycle reads and restores a word or may store a new word. The S register determines which memory location will be referenced by selecting one X and one Y drive line transformer and, consequently, the two drive lines. The storage word is read out into the Z register and then transferred to the F register as an instruction or to the X register as an operand.

Although the explanation of a magnetic core storage system and its memory cycle is necessarily quite lengthy, the time duration of a complete memory cycle is only slightly more than one millionth of one second. Those little iron doughnuts may be in continual use at that rapid pace without even getting stale. You now deserve a coffee (and little iron doughnuts) break.

The following review questions allow you to self-test your understanding of various memory devices and, specifically, the magnetic core type of storage. After completing the questions, check your solutions with those at the end of the chapter.

1) What two types of information may be stored in any memory device?

a)

b)

2) What is one of the most important factors to be considered in the design of a memory device? Why?

3) List two advantages and two disadvantages of each of the three memory devices discussed.

	Advantages	Disadvantages
Drum	1.____________________	1.__________________________
	2.____________________	2.__________________________
Disk	1.____________________	1.__________________________
	2.____________________	2.__________________________
Core	1.____________________	1.__________________________
	2.____________________	2.__________________________

4) Why are magnetic core memories utilized so often if they have the two disadvantages you listed?

5) Why are drum memories utlized so often if they have the two disadvantages you listed?

6) Making the assumption that a computer uses a drum storage device, can the one word access time be improved without increasing the speed of the drum or making it physically smaller?

If no, why not?

If yes, how?

7) A computer's memory consists of 4,096 24-bit words. Draw your design of the memory stack and indicate the number of wafers and matrix size.

8) How would you modify your design for question #7 if the memory size were increased to 32,768 words?

9) What are the advantages gained by increasing memory size?

10) What address length(number of bits) would be required to access each word of a

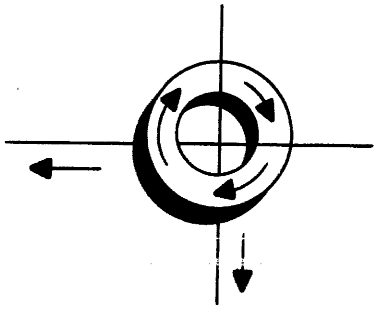
4096 word memory? ______________________

8192 word memory? ______________________

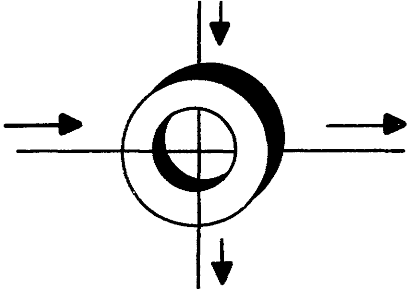
16,384 word memory? ______________________

32,768 word memory? ______________________

11) The following illustration indicates the magnetic state of a core. Will the core switch when electron flow through the drive lines is in the direction indicated by the arrows?



12) In what direction would the core be magnetized with electron flow through the drive lines as indicated by the following illustration?

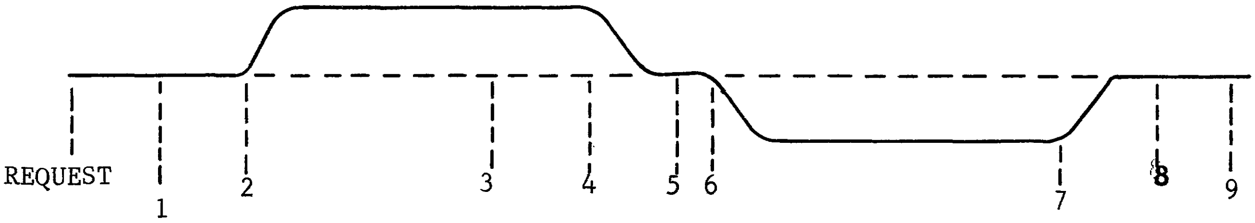


13) Draw the hysteresis loop for a half-selected core in the one state as read current is applied.

14) Explain how a word is written into memory and how the inhibit generators determine what state each core of the word will assume.

15) What determines whether a Memory Request is to read a word from memory or to store a new word into memory?

16) Each number in the following illustration identifies an event or events that occur during a memory cycle. List each of these events.



1.
2.
3.
4.
5.
6.
7.
8.
9.

17) A memory consists of 4096 words (one field) of 24 bits each. A random pattern of data is stored in 3000 locations, yet bit 17 of all 3000 words is always read as a "0". What would you do to correct the obvious malfunction?
Explain why or why not each of the following choices could cause the problem.

a. Failing sense Amplifier

b. An open drive line

c. An open sense line

d. Inhibit generator for bit 2^{17} is burned out

e. Someone stole one of the memory wafers

18) A memory malfunction exists and, after investigation, you decide that a drive line has opened. The failing addresses are all between 60000_8 and 67777_8. Which stack would contain the faulty drive line and which field would you replace?

19) The operand 66666666_8 is stored in memory. How many inhibit generators are turned on?

20) Without referring to the text, draw the circuit required to read and restore one bit of a memory word (core, sense amp, Z register flip-flop, etc.). Include the AND gates that allow the same circuit to be used to store a new bit.

A quick comparison of the solutions should indicate that you understand the concepts of a destructive read out, magnetic core memory. Compare these concepts with the thin-film, word-organized, non-destructive readout (NDRO) memory briefly explained in Appendix A, Volume III.

ANSWERS TO REVIEW QUESTIONS

1) a) Instructions

b) Data (operands)

2) Speed requirements of the computer system.

The applications for a new computer system dictate how fast it must operate. Generally, the slower, sequential devices are less expensive than the sophisticated, random-access memories. If memory access time is not a factor, a slower and more economical device may be used.

3) Drum:

Advantages	1)	High speed transfer for large blocks of data.
	2)	Relatively simple, both mechanically and electronically.
Disadvantages	1)	Long one-word access time.
	2)	Reliability is limited by mechanical characteristics.

Disk:

Advantages	1)	High speed transfer rate for large blocks of data.
	2)	Massive storage capacity.
Disadvantages	1)	Long one word access time.
	2)	Mechanically complex.

Core:

Advantages	1)	Completely random access.
	2)	Very high speed.
Disadvantages	1)	Relatively expensive.
	2)	Electronically complex.

4) If fast memory access time is a necessity, the speed advantage outweighs the added cost of the magnetic core memory. Although complex in design and operation, the electronic memories are very reliable and mechanical problems are practically eliminated.

5) In many applications where mass storage of large blocks of information is required, the advantages offered by drum storage far outweigh the disadvantages.

6) Another set of heads could be added half-way around the drum to decrease the initial access time by 50%. However, due to the added circuitry and more complex addressing techniques required, another type of storage device would probably be used instead of the drum.

7) Figure 8-12 illustrates the proper configuration. Each of the four planes on a wafer would contain a 64 x 64 matrix of 4096 cores.

8) See Figure 8-13.

9) A program consisting of 10,000 instruction and data words could not be entered into a 4K memory at one time. The program would have to be executed in segments of 4,000 instructions each. Increasing the word length of a computer allows larger operands to be expressed and makes the system more versatile.

10) a. 4096 words could be accessed with a 12-bit address (one field).

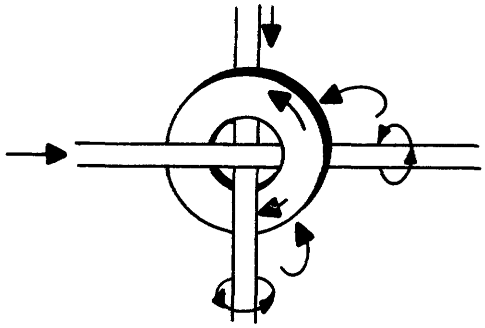
b. 13 bits

c. 14 bits

d. 15 bits

11) Yes, the coincident magnetic fields oppose the magnetic state of the core and it will switch.

12) Notice how the drive lines pass through the core and the direction of electron flow. The two magnetic fields are in opposition to each other and would cancel, effectively having no effect on the core.



13) Figure 8-23.

14) The word is transferred from the X register to the Z register which, in turn, determines which inhibit generators are turned on. A cleared flip-flop, representing a binary zero, in the Z register causes its inhibit generator to be turned on. The inhibit current opposes the write current and the core is not switched to a one during the write portion of the memory cycle.

15) The function translators determine what type of instruction is in the F register and consequently what type of operation is to be performed. For example, a store A instruction would write into memory whereas an ADA instruction would dictate a read operation from memory.

16)
1. Clear Z register, Address to S
2. Start Read
3. Gate word to Z register (from either sense amps or X register)
4. End Read
5. Inhibits on
6. Start write
7. End Write, inhibits off
8. Clear S register
9. Discharge drive lines

17)
a. The most probable cause would be sense amplifiers because it is a differential amplifier trying to detect the state of a core from the small voltage induced into the sense line. A large portion of memory malfunctions are caused by failing sense amplifiers.

b. An open drive line would affect all bits of 64 memory words because a drive line passes through 64 cores in each plane (Figure 8-17).

c. An open sense line is a possibility but not as probable as the sense amplifier (mechanical rather than electronic).

d. With a bad inhibit generator, no inhibit current would flow and the bit would always be switched to a one during the write portion of the memory cycle.

e. A memory wafer contains 4 bits of each of 4096 words. If the other three bits on that wafer (16,18,19) are being read correctly, there is no memory wafer thief in the crowd.

18) Stack 3, field 0 (Figures 8-13 and 8-14).

19) The birary equivalent of 66666666_8 contains 8 zeros. Therefore, eight inhibit generators would be turned on by the eight cleared flip flops in the Z register.

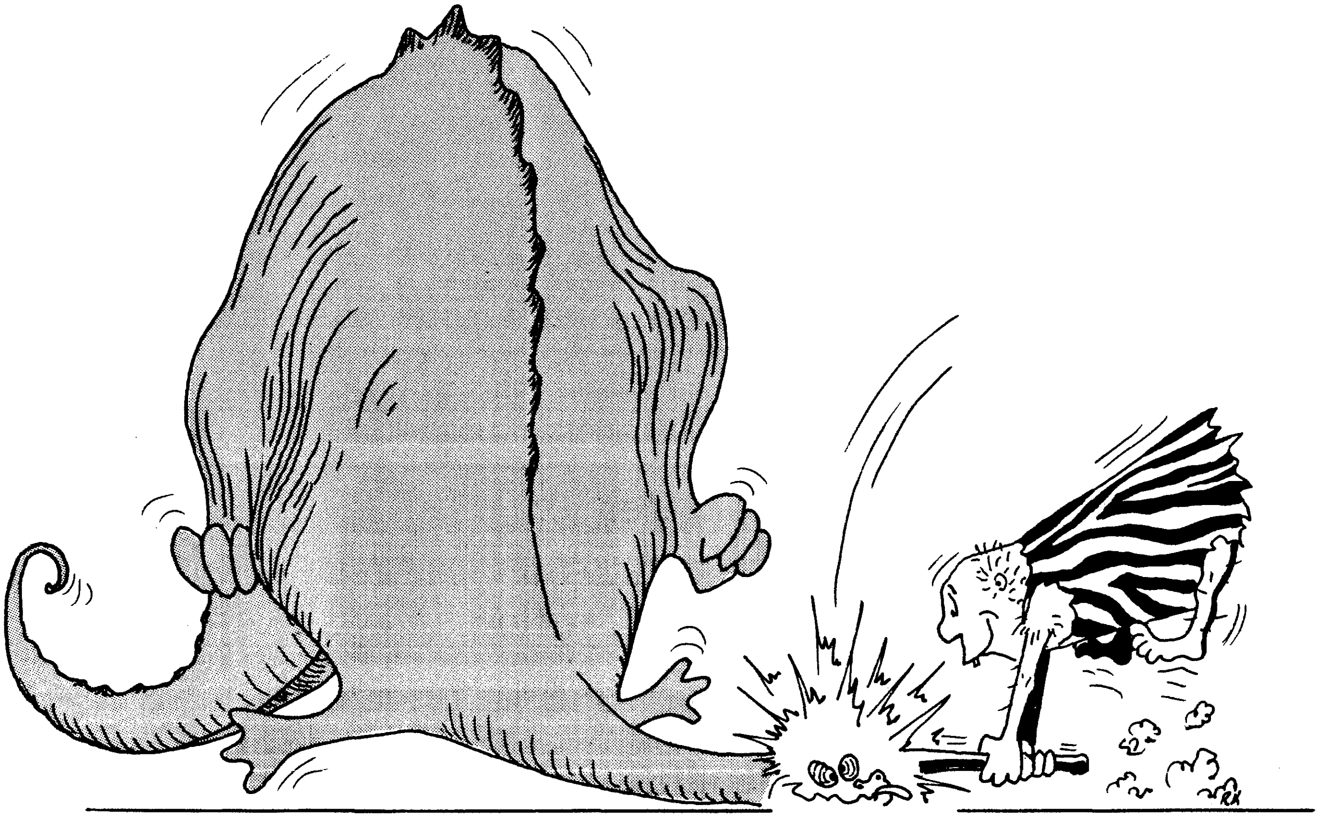
20) Your circuit would be similar to the one in Figure 8-45.

chapter VII

Arithmetic Section

chapter IX

Control Section



Control

CHAPTER IX

CONTROL SECTION

INTRODUCTION

The third section of the computer to be discusses is the control section. The two preceding chapters, Arithmetic and Memory, each explained a section the computer and each made reference to commands initiated by the control section. In Figure 9-1, the man is analogous to the control section of a computer. The adding machine and the pencil and paper are useless if the man is on a coffee (and little iron doughnuts) break. Likewise, a cobweb-covered in/out basket is indicative that the man is "out to lunch."

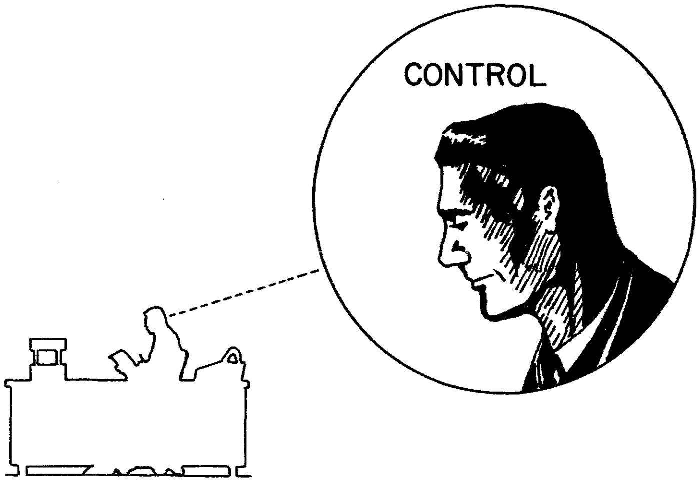


Figure 9-1. Man-sized computer

If the sections of a digital computer were classified in their order of importance, the control section would undoubtedly be first.

The control section has the responsibility of reading and executing instructions. Each of the stored-program instructions is read from memory into the F register where it is translated. The translation indicates what operation is to be performed and the control section generates the proper signals at the proper time to supervise the execution.

Possibly the man at the desk is an accountant, auditing financial reports. He removes a report from the basket and records some of the pertinent information on paper (the program is read into computer storage). The man reads the report and determines that two numbers should be added together (reading and translating an instruction). The numbers are added on the adding machine and the sum is formed (execution of ADA instruction). The sum is then recorded on a piece of paper (STA instruction) which is then placed in the out basket (output from computer storage to an I/O device, possibly a line printer). The entire process was under the CONTROL of the man and was performed in a logical sequence.

The objective of this chapter is to familiarize you with the overall function of the control section and the methods used whereby instructions are read and executed.

COMPUTER BLOCK DIAGRAM

The following block diagram illustrates the control section of the basic computer.

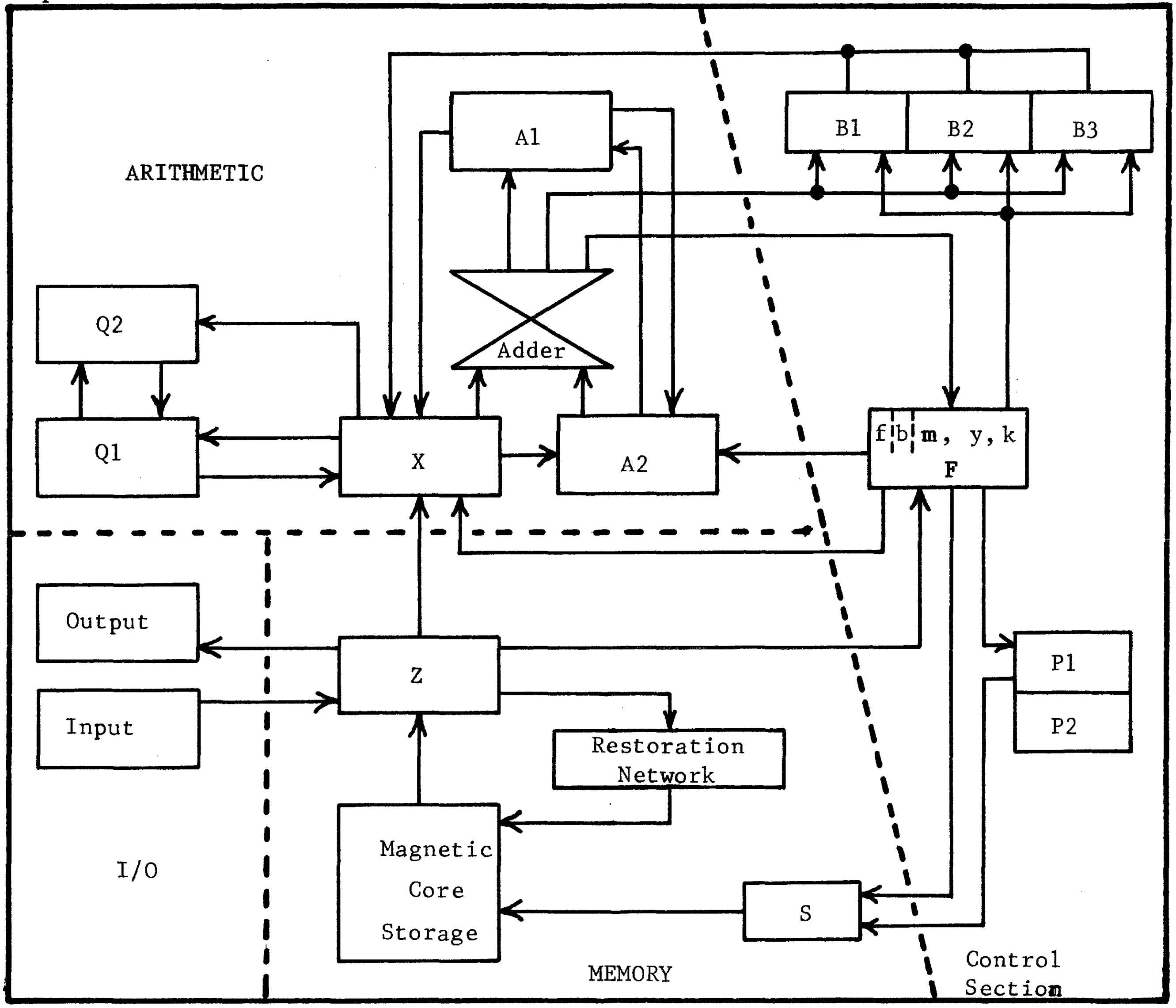


Figure 9-2. Basic Computer Block Diagram

It is important to realize that the diagram does not show all of the logic circuits of the computer, but does show the major components (memory, adder, registers) and how they are interconnected. Although it appears that the control section consists of only the B, F, and P registers, it also contains the logic circuitry required for the reading and execution of instructions.

READING THE INSTRUCTION

Before a program can be executed, it must be placed in computer storage. The operator then "master clears" the computer, sets the P register to the address of the first instruction, and presses the GO button on the console. The control section of the computer then generates a command that initiates the reading of the instruction from storage. The subsequent events are illustrated by Figure 9-3.

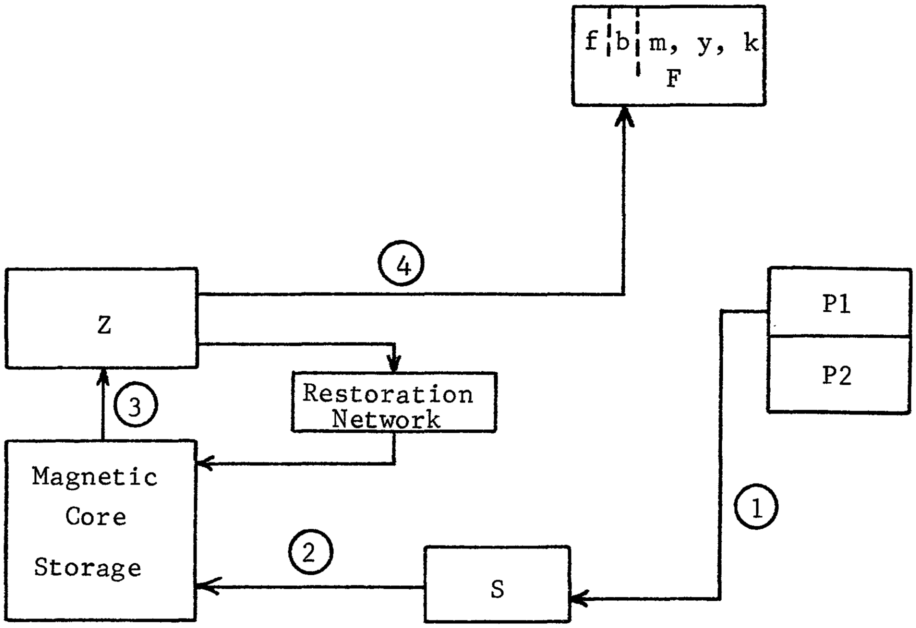


Figure 9-3. Reading an Instruction

You will recall from the preceding chapter that when an operand or an instruction is to be read from storage, the control section issues a Memory Request. The request pulses a delay line that controls the entire memory sequence, including steps 1, 2, and 3, indicated on the preceding diagram.

1. Address to S register (P→S)

2. Start read

3. Gate word to Z register

Upon receipt of a "Memory Request" from the control section, the memory section generated the commands to transfer the word to the Z register and restore it in memory. After the word has been gated to the Z register, the memory section sends a Reply to the control section which indicates that the word is ready to be transferred to the F register. Upon receipt of the Reply, the control section issues a command to transfer (Z) to F. The command transfers the instruction to the F register and translation begins.

Instruction translation is also considered as part of the reading sequence because, until after the computer has translated, it does not "know" what operations are to be performed. That entire sequence of events, from issuance of the Memory Request until the instruction has been translated, is known as the Read Next Instruction (RNI) Sequence (Figure 9-4).

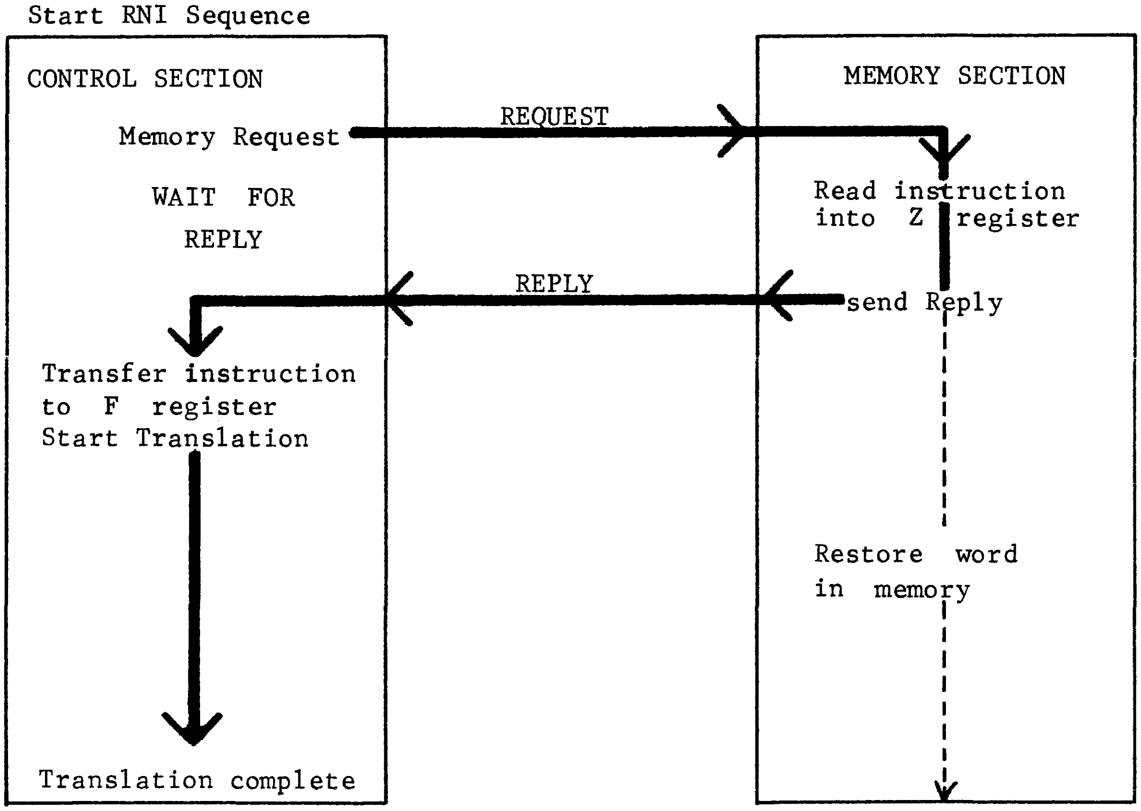


Figure 9-4. RNI Sequence

EXECUTING THE INSTRUCTION

The translated instruction in the F register must now be executed by another sequence. The proper sequence is selected by translators which have classified the circuit instruction according to its function. The other sequences are Read Address (RADR), Read Operand (ROP), and Store Operand (STO).

RADR Sequence

Entry is gained into the RADR sequence only if the "a" designator bit of the instruction specifies indirect addressing (a="1").

f	a	b	m

Indirect addressing is possible only if the address field (lower 15 bits) of the instruction is actually an address. However, all jump instructions have a jump address but only the UJP is indirectly addressable. A quick review of the 25 instructions listed in Chapter IV will show that only the following nine may be indirectly addressed.

ADA,I	30	a	b	m
DVA,I	51	a	b	m
LDA,I	20	a	b	m
LDQ,I	21	a	b	m
MUA,I	50	a	b	m
SBA,I	31	a	b	m
STA,I	40	a	b	m
STQ,I	41	a	b	m
UJP,I	01	a	b	m

Once entry has been gained into the RADR sequence, the "b" designator is examined first and, if designated, indexing is performed. Memory is then referenced at $m + (B^b)$, the modified address, and the lower 18 bits of the instruction are replaced with those from storage. If the new "a" designator is also a "1", indirect addressing is again required and a second pass through the RADR sequence is made (Figure 9-5).

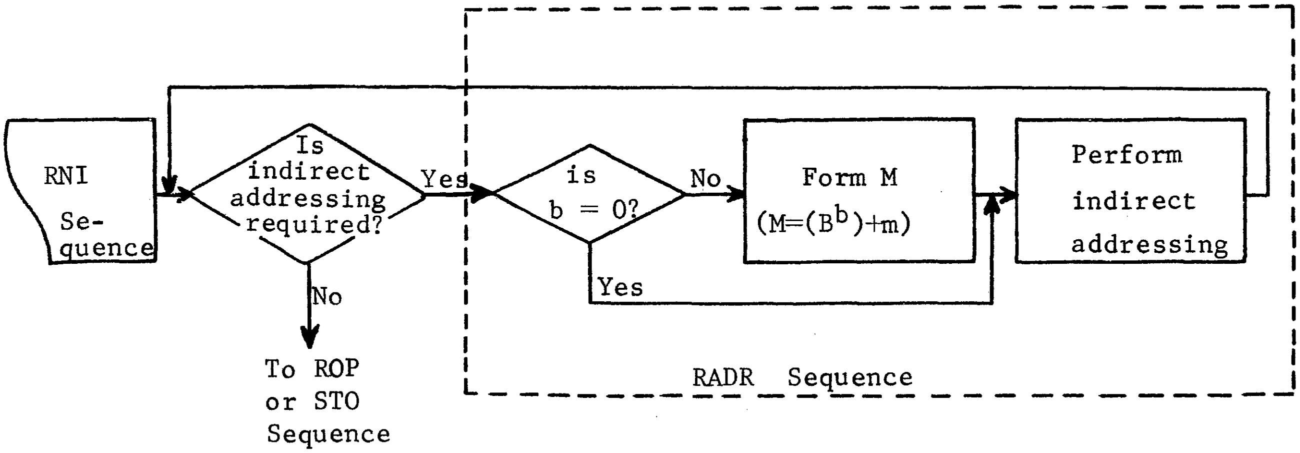


Figure 9-5. RADR Sequence

ROP Sequence

The ROP sequence is entered if the current instruction requires a word from memory. Another quick review of the instruction repertoire indicates that the following instructions require a word from memory and would, therefore, use the ROP sequence.

LDA	20	a	b	m	(M) ⟶ A
LDQ	21	a	b	m	(M) ⟶ Q
ADA	30	a	b	m	(A) + (M) ⟶ A
SBA	31	a	b	m	(A) - (M) ⟶ A
MUA	50	a	b	m	(M) X (A) ⟶ QA
DVA	51	a	b	m	(AQ) ÷ (M) ⟶ A

Each of the instructions could be indirectly addressed. In that case, the RADR sequence would precede the ROP sequence (Figure 9-6).

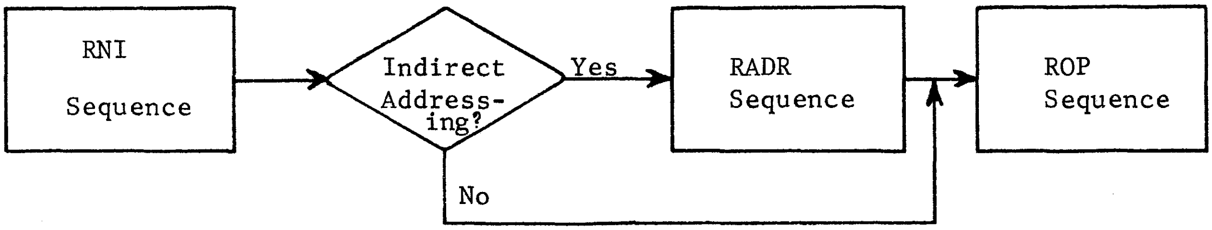


Figure 9-6. RNI, RADR, ROP Sequences

An instruction could specify address modification but may not specify indirect addressing. For example, the instruction 30 3 12345 does not specify indirect addressing and, therefore, the RADR sequence would not be used. However, address modification is specified $(M=(B^3)+m)$ and is accomplished in the ROP sequence before the operand is read from storage. After the operand has been obtained, it is added to the contents of the A register and execution of the instruction is then complete.

STO Sequence

If the instruction in the F register specifies that an operand is to be written into memory, the STO sequence is used instead of the ROP sequence. Only three of the twenty-five instructions listed in Chapter IV would use the STO sequence; they are STA, STQ, and RTJ. The RTJ instruction stores a return address (P+1) at the location designated by the instruction to allow return to the main program (review the RTJ instruction, if necessary). If the instruction is STA or STQ, addressing $(m+(B^b)=M)$ may be required to form the address (M) at which the operand will be stored. Addressing is not possible on the RTJ instruction.

The remaining sixteen instructions do not use either the ROP or STO sequence because the use of memory is not required. For these instructions, execution is accomplished in the same RNI sequence that allowed the instruction to be read from memory. Another RNI sequence would then be initiated to read the next instruction.

The control section is divided into three sub-sections; main control, arithmetic control, and block control.

MAIN CONTROL

The reading and translating of instructions is the primary function of main control. After translation, main control may initialize arithmetic control or block control if the current instruction requires the use of either of those two sections. After arithmetic control has been initiated, main control starts to read the next instruction from storage while the current instruction is still being executed.

ARITHMETIC CONTROL

All instructions that utilize a portion of the arithmetic section of the computer require the use of arithmetic control. Once initiated, an Arithmetic Busy FF is set which prevents another instruction from using the arithmetic section until the current instruction has been executed.

BLOCK CONTROL

All I/O instructions are executed as a function of block control.Because it is primarily I/O-orientated, block control will be discussed in the next chapter.

The commands generated by the control section are a result of pulses from a timing chain. The pulses are tapped off the chain at desired intervals. Main control has its own timing chain that is independent of those used by its two sub-sections: arithmetic control and block control. Figure 9-7 illustrates the relationship between the sequences and the timing chain for a 30 1 12345 instruction.

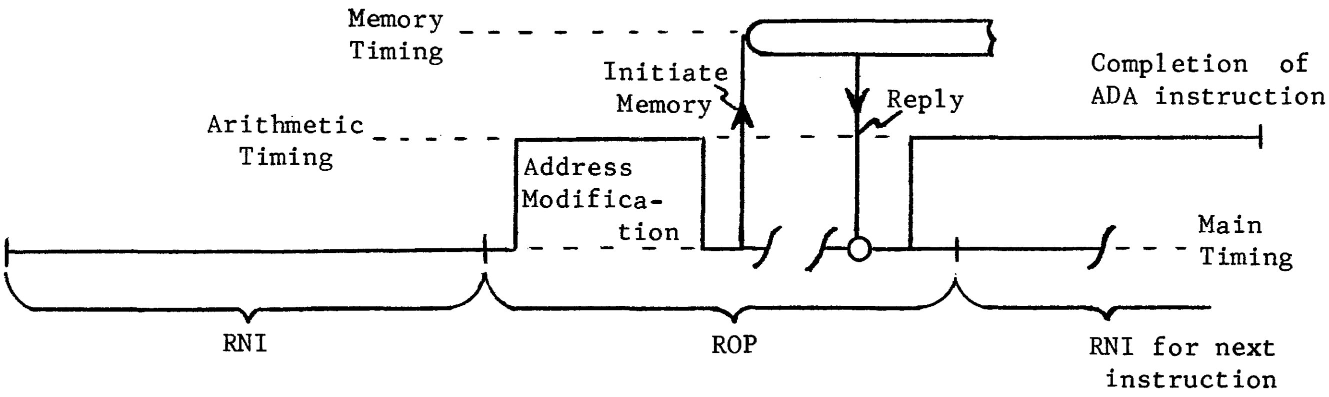


Figure 9-7. ADA Instruction Timing

A sequence may consist of commands from several timing chains. In the preceding figure, the ROP sequence was made up of commands from main control, arithmetic control, and the Reply from the memory delay line. Arithmetic Timing is used to perform address modification (indexing). Main timing then generates a Memory Request to obtain the operand. After memory has been initiated, it sends a Reply back to main timing. Arithmetic Timing is then used to perform the addition of the two operands while main timing reads the next instruction from memory. Some instructions, such as a multiply, require a considerable amount of time for execution. The next instruction following the multiply could be read and executed before the multiply is finished, but only if the next instruction is executed by the RNI sequence and does not require arithmetic timing. The only instructions that satisfy both requirements are SLS, SJ1-6, HLT, and UJP. Simultaneous execution is possible because there is an auxiliary function register (F2) that holds the upper nine bits of the previous instruction while the 24-bit F1 register holds the next instruction. Function translators for each register allow two operations to be occurring at the same time. Table 9-1 contains a list of the instructions and the sequences required for execution.

Table 9-1

	RNI SEQUENCE ONLY			RNI	ROP
	AZJ			ADA	**
	ENA			DVA	**
	ENA,S			LDA	**
	ENI			LDQ	**
	ENQ			MUA	**
	ENQ,S			SBA	**
*	HLT				
	INI				
	ISD			RNI	STO
	ISI		*	RTJ	
	SHA			STA	**
	SHQ			STQ	**
	SHAQ				
*	SJI-6				
*	SLS				
*	UJP	**			

* instruction does not require arithmetic timing.

** instruction may be indirectly addressed and would then require the use of the RADR sequence immediately after the RNI sequence.

NOTE

If the UJP instruction has a "b" designator of 1-3 (indexing required), arithmetic timing would be required because address modification is performed in the adder.

The importance of the computer control section should now be quite evident. A detailed analysis of the execution of instructions will be made to further enhance your understanding.

DETAILED CIRCUIT ANALYSIS

Before a program can be executed, it must be contained in magnetic core storage. Master Clear, set the P register to the address of the first instruction (keyboard entry from console), and press the GO button. Program execution then progresses until some type of Stop instruction is read. All operations between the initial start and the reading of the Stop instruction are automatic, initiated by the control section. The RNI sequence is activated when the GO button on the console is pressed. During the reading of any instruction, RNI has five major functions.

1) Ensure proper conditioning of the P register.
   Upon initial start, the P register is not advanced because the first instruction is to be read from (P). After execution of the first instruction, the P register will be advanced at the beginning of RNI before memory is initiated to read the next instruction. If the previous instruction was an ISI or an ISD <u>and</u> conditions are met, the P register will be advanced twice (to P+2) before the next instruction is read. If the previous instruction was a jump and jump conditions are met, P is not advanced but is forced to the new address in the lower 15 bits of the F register.

2) Read the instruction from storage and place it in the F register.
   The current instruction must be placed in the F register before execution can be accomplished because the function translators are fed by the F register (review Chapter VI).

3) Sense for interrupts.
   When an abnormal condition arises during program execution, it may be desirable to interrupt normal operations and alleviate the abnormal condition. For example, if an ADA instruction produced overflow, subsequent arithmetic operations would only propogate the incorrect result. The interrupt routine is a special program that analyzes the abnormal condition and performs the necessary corrections before returning to the main program. The computer would not stop and, except for an interrupt light on the console, the operator might not be aware that an interrupt had occurred.

4) Translate the instruction.
   Translation must be accomplished to inform the control section as to which functions must be performed to ensure proper execution. The translation also determines which sequence should follow RNI (RADR, ROP, STO, or another RNI).

5) Stop the computer if a stop instruction is read, and stop conditions are met (SLS and stop switch set or HLT).

> After the instruction has been read and translated by the RNI sequence, it must be executed. If the instruction is an ADA, the ROP sequence is initiated at the end of RNI and the addition is then performed. An STA instruction would require the STO sequence at the end of RNI to store the contents of the A register in memory.

The following diagram (Figure 9-10) illustrates how each of the sequences is initiated at the end of RNI by setting either the RADR, ROP, or STO flip-flops. The flip-flop that goes set is determined by the translation of the current instruction. The entire ROP sequence logic is not included in the diagram but sufficient logic is included to illustrate how the ADA, SBA, and LDA instructions are executed and how indexing is performed at the beginning of the ROP sequence.

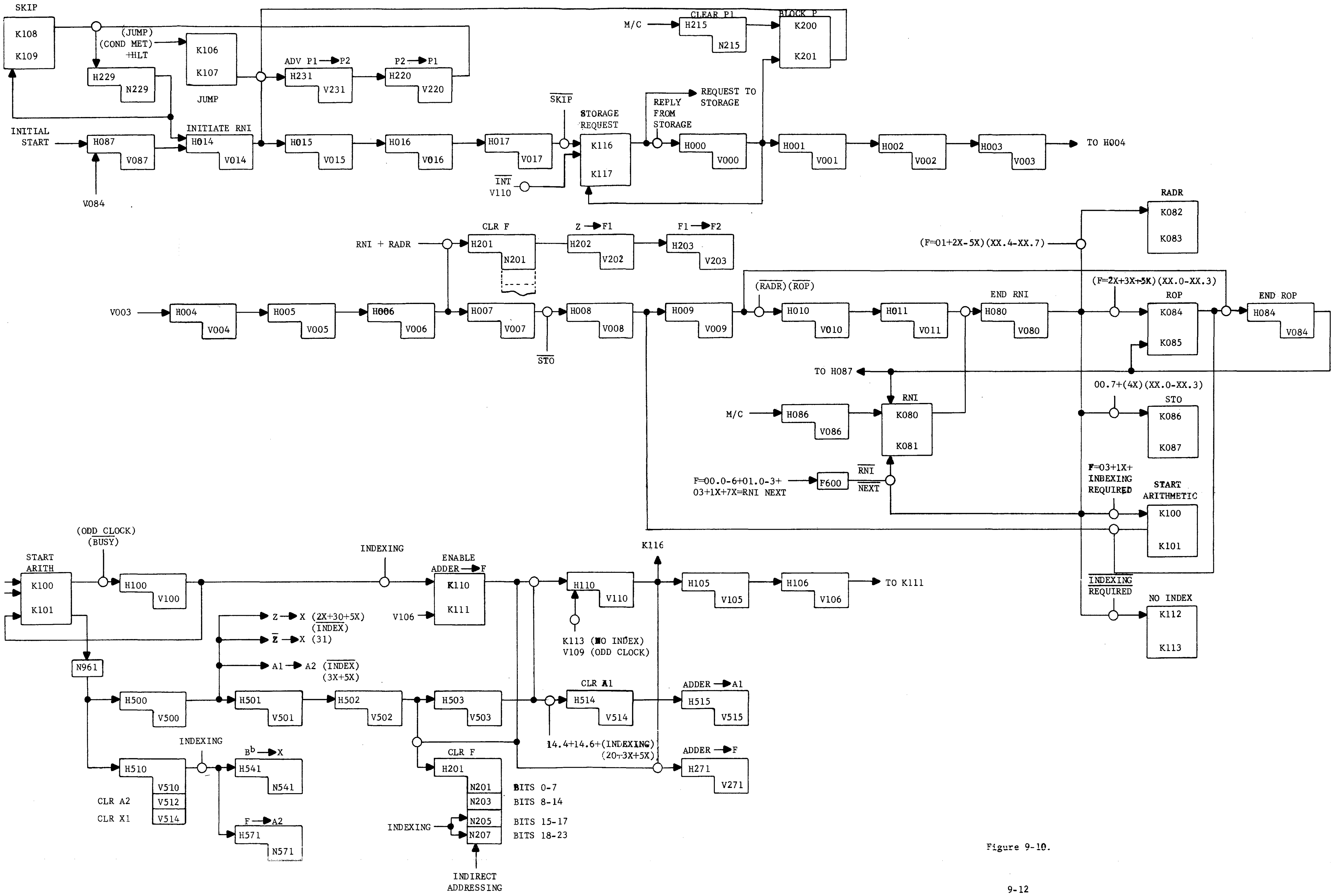

SKIP
K108
K109
(JUMP)
(COND MET)
+HLT
K106
K107
JUMP
H229
N229
ADV P1→P2
H231
V231
P2→P1
H220
V220
M/C
CLEAR P1
H215
N215
BLOCK P
K200
K201
SKIP
STORAGE
REQUEST
REPLY
FROM
STORAGE
REQUEST TO
STORAGE
INITIAL
START
H087
V087
V084
INITIATE RNI
H014
V014
H015
V015
H016
V016
H017
V017
K116
K117
INT
V110
H000
V000
H001
V001
H002
V002
H003
V003
TO H004
RADR
K082
K083
RNI + RADR
CLR F
H201
N201
Z→F1
H202
V202
F1→F2
H203
V203
(F=01+2X-5X)(XX.4-XX.7)
V003
H004
V004
H005
V005
H006
V006
H007
V007
STO
H008
V008
H009
V009
(RADR)(ROP)
H010
V010
H011
V011
END RNI
H080
V080
(F=2X+3X+5X)(XX.0-XX.3)
ROP
K084
K085
END ROP
H084
V084
TO H087
00.7+(4X)(XX.0-XX.3)
RNI
K080
K081
M/C
H086
V086
STO
K086
K087
F=00.0-6+01.0-3+
03+1X+7X=RNI NEXT
F600
RNI
NEXT
F=03+1X+
INDEXING
REQUIRED
START
ARITHMETIC
K100
K101
INDEXING
REQUIRED
NO INDEX
K112
K113
(ODD CLOCK)
(BUSY)
START
ARITH
K100
K101
H100
V100
INDEXING
ENABLE
ADDER→F
K110
K111
V106
K116
H110
V110
K113 (NO INDEX)
V109 (ODD CLOCK)
H105
V105
H106
V106
TO K111
Z→X (2X+30+5X)
(INDEX)
Z→X (31)
A1→A2 (INDEX)
(3X+5X)
N961
H500
V500
H501
V501
H502
V502
H503
V503
CLR A1
H514
V514
ADDER→A1
H515
V515
14.4+14.6+(INDEXING)
(20+3X+5X)
ADDER→F
H271
V271
INDEXING
H510
V510
V512
V514
CLR A2
CLR X1
B^b→X
H541
N541
F→A2
H571
N571
CLR F
H201
N201
N203
N205
N207
BITS 0-7
BITS 8-14
BITS 15-17
BITS 18-23
INDEXING
INDIRECT
ADDRESSING

Figure 9-10.

Figure 9-10 illustrates the logic circuitry for the RNI and ROP sequences. Assuming that the program:

```
00100    20 0 00103

00101    30 1 00074

00102    00 0 00100

00103    00 0 00200

00104    00 0 00600
```

is to be executed, the following operations must be accomplished.

1) Read and translate the LDA instruction (RNI)

2) Execute the LDA instruction (ROP)

3) Read and translate the ADA instruction (RNI)

4) Perform indexing ($M=(B^1)+m$) (ROP)

5) Execute the ADA instruction (ROP)

6) Read and translate the HLT instruction (RNI)

7) Stop the computer (RNI)

The operations are performed by generating the proper commands in the proper sequence (refer to Figure 9-10). That proper sequence is as follows:

1. Master clear the computer. This sets the RNI flip-flop (Zone J2) and the Block P F/F (D1) in addition to clearing all operational registers.

2. Set the P register to 00100 (address of the first instruction).

3. Set (B1)=00010

4. Press the GO button on the console. When the Go button is pressed, the initial start circuitry pulses H087 (Zone A1) which initiates RNI to read the LDA instruction.

LDA INSTRUCTION

The P register is not advanced on the first pass through RNI because the AND gate is broken by the Set Block P flip-flop. The storage request FF is set to hold the pulse until the Reply comes back from storage. Commands from the memory section gate the address from P to S and cause the instruction to be read from storage into the Z register (review Chapter VIII if necessary). As soon as the reply comes back from storage, the gate is made into H000 and the RNI sequence continues. The Block P FF is cleared in preparation for the advancement of P on the next pass through RNI. The seven phase-time wait (V000 to V006) is to allow time for memory to place the instruction in the Z register. The output of V006 clears the F register and places the instruction in F1 and the upper nine bits also in F2.

The output from V008 cannot set the Start Arithmetic FF because the ROP FF is not set. The cleared ROP FF also prevents the output of V009 from pulsing the End ROP control delay, H084. The output of V009 does make the gate into H010 (not RADR and not ROP). Two phase times later, the output of V011 pulses the End RNI control delay which clears the RNI FF and ends the first RNI sequence. The LDA instruction is now in the F register and has been translated.

The End RNI control delay enables a gate to set RADR, ROP, or STO, depending on the current instruction. It also enables either the Start Arithmetic or the No Index FF. The current instruction is 20 0 00103, which causes the ROP and No Index FFs to set. If indirect addressing had been specified, RADR would have set instead of ROP.

The setting of the No Index FF indicates that Indexing is not required and that the arithmetic section will not be used at this time. The No Index FF makes the AND gate into H110 (Zone D2), which requests storage for the operand if no interrupts are pending.

The memory section obtains the address from the lower fifteen bits of the F register, initiates storage to honor the request, and sends back the Reply. The same timing that read the instruction (H000 -- H009) is now used again to read the operand. However, the F register is not cleared by V006 on this pass because an ROP sequence is now in progress instead of RNI.

The output of V008 sets the Start Arithmetic FF and the output of V009 now ends the ROP sequence (although the operand has not yet been placed in the A register). The AND gate into H010 cannot be made at this time because the ROP FF is still set for one more phase time.

The End ROP delay now clears the ROP FF, sets the RNI FF, and pulses H087, to read the ADA instruction. Arithmetic timing is also in progress to place the first operand in the A register.

The Start Arithmetic FF is duplicated (Zone A2) to show how it initiates the arithmetic timing. It pulses H100 whose output is fed back to clear K100/K101. The Enable Adder → F FF cannot set because indexing is not being performed. When the Start Arithmetic FF sets, N961 outputs a logical "1" for one phase time (until the flip-flop is cleared), which pulses H510 (clears A2 and X1) and H500. The output of H500 gates the operand from Z to X.

The cleared A2 register and the X register now feed the adder and propogation begins. The A1 register is cleared (V514) in preparation for the output of the adder. The output of V515 then gates the operand (adder output) into the A register.

ADA INSTRUCTION

While the first instruction was being executed, the RNI sequence was using main timing to read the ADA instruction. On this pass through RNI, the P register will be advanced because the Block P FF is clear and the previous instruction did not specify a jump condition. If the previous instruction had been an ISI or an ISD and skip conditions were met, the Skip FF would allow the P register to be advanced twice before the storage request was issued.

The RNI sequence is identical to that for the LDA instruction except that the Start Arithmetic FF is set instead of No Index FF at the end of RNI. Arithmetic Timing is now used to form M. The output of V100 (B2) sets the Enable Adder →F FF (C2). The output of V510 gates (B1) to X and (F) address to A2; the output of V502 clears the lower 15 bits of F; and V503 makes the AND gate into H110. The output of V110 gates the output of the adder (M) to F address and also issues the Storage Request for the operand.

The ROP sequence is the same for the ADA instruction as for the LDA except that indexing was performed at the start of ROP. Although A2 is cleared by V510 (B3), it is restored by the A1→A2 pulse out of V500. Except for the A1→A2 pulse, the LDA and ADA instructions are identical.

HLT INSTRUCTION

The third instruction in the program is the HLT. While the arithmetic timing was being used to execute the ADA instruction, main timing was reading the HLT instruction. The End RNI control delay cannot set any of the five flip-flops (Zone K) and cannot clear the RNI FF. However, a GO FF (not shown) is cleared which causes the computer to stop at the end of RNI. The jump flip-flop is set in preparation for a jump to m if the computer is restarted. When the computer stops, the console displays the contents of the registers.

A = 00001000

Q = 00000000

F = 00 0 00100

B1 = 00010

If indirect addressing had been specified on either the LDA or ADA instruction, the End RNI control delay would have set the RADR FF instead of ROP. After completion of the indirect addressing, the ROP FF would be set and the instruction would be executed in the normal manner.

The following timing chart illustrates the sequence of commands issued by main and arithmetic timing to accomplish the execution of the previously-mentioned program.

MAIN TIMING

ARITHMETIC TIMING

LDA INSTRUCTION

Initiate RNI

Storage Request for instruction

WAIT FOR REPLY

CLR BLOCK P

CLR F

(Z)→F

F1→F2

END RNI

    Set NO INDEX

    Set ROP

    Clear RNI

Storage Request (V110) for operand

WAIT FOR REPLY

Set START ARITHMETIC

END ROP

Set RNI

	MAIN TIMING	ARITHMETIC TIMING	
ADA INSTRUCTION	Initiate RNI	CLR A2 and X1	LDA INSTRUCTION
		Z→X	
	Advance P to 00101		
		Clear A1	
	Storage Request for ADA instruction	Adder A1	
	WAIT FOR REPLY		
	Clear F		
	(Z)→F1		
	F1→F2		
	END RNI		
	Set START ARITHMETIC (b=1)		
	Set ROP		
	Clear RNI		
		Clear A2 and X1	INDEXING
		Set ENABLE ADDER→F (B^1 → X and (X) → A2 Clear lower 15 bits of F	
	STORAGE REQUEST for operand	Adder output (M)→F	
	WAIT FOR REPLY		
	Set START ARITHMETIC		
	END ROP		
	Set RNI		

	MAIN TIMING	ARITHMETIC TIMING	
	Initiate RNI	CLR A2 and X1	ADA INSTRUCTION
		Z→X	
	Advance P to 00102		
		Clear A1	
	Storage Request for HLT instruction	Adder→A1	
HLT INSTRUCTION	WAIT FOR REPLY		
	Clear F		
	(Z)→F		
	F1→F2		
	END RNI		
	Clear GO		
	Computer Stops		

The preceding Timing chart is similar to those published for each computer. They make up a manual called COMMAND TIMING in which the entire timing for each instruction is listed. The command timing charts are very helpful in the step-by-step analysis of an instruction and for troubleshooting malfunctions. Figure 9-11 illustrates the actual LDA timing for the 3200 computer. Notice the main timing commands on the left and the arithmetic timing commands on the right. Although some terms are different from those previously discussed, essentially the same operations are being performed.

20 (LDA) LOAD A

Function: Load A with a 24-bit quantity from storage address M.

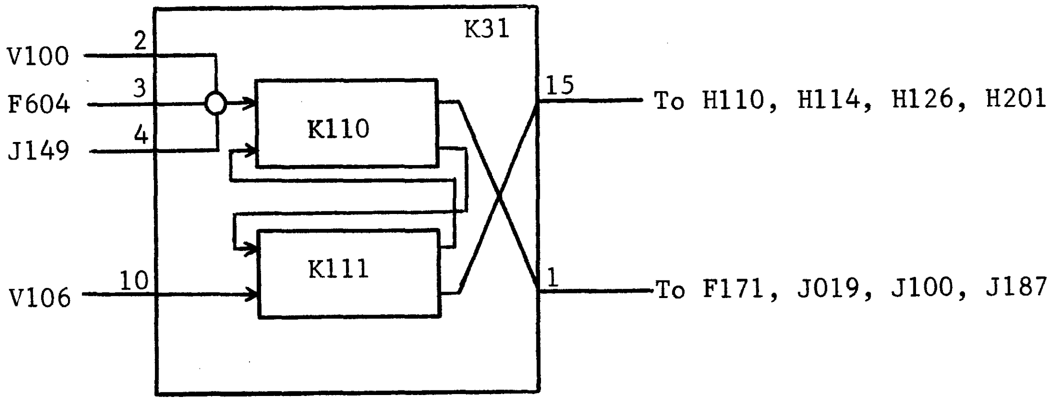
Main Control Timing | Arithmetic Timing

TIME	TERM	COMMAND	REMARKS	TIME	TERM	COMMAND	REMARKS
			This timing chart begins at V008 of a ROP sequence initiated by decoding a 20 instruction.				Indexing and indirect addressing are applicable to this instruction, but are not included in this timing chart.
V008	K^{104}_{105}	Set Start Arith 2	The 24 - bit operand from location M (M = m+B^b) is now in the DBR.				
V009 (V109)	H084 H104	End ROP		N691	H510	Clear X^1 and A^2	
					H500		
V084	K^{084}_{085}	Clr ROP		V500	H501 H531	$I^4 \longrightarrow X^1$	X^1 and A^2 now cleared. $\overline{\text{DBR}} \longrightarrow I^4$ enabled by F translations.
	K^{080}_{081}	Set RNI					
	H087						
(V104)	K^{104}_{105}	Clr Start Arith 2					
V087	H014	Initiate RNI	Main control now reads the next instruction while the arithmetic section simultaneously completes the LDA.	V501	H502		$\overline{\text{DBR}} \longrightarrow I^4 \longrightarrow X^1$ now occurs. This places (M) in X^1. (A^2) = Zero. Adder propagation begins.
				V502	H503		
				V503	H504 H514	Clear A^1	
				V504 (N544)	H505 H515	$I^0 \longrightarrow A^1$	A^1 now cleared. $\overline{\text{Sum}} \longrightarrow I^0$ statically enabled.
				N50-			$\overline{\text{Sum}} \longrightarrow I^0 \longrightarrow A^1$. This places (M) in A^1.
							Instruction execution is complete; arithmetic timing ends.

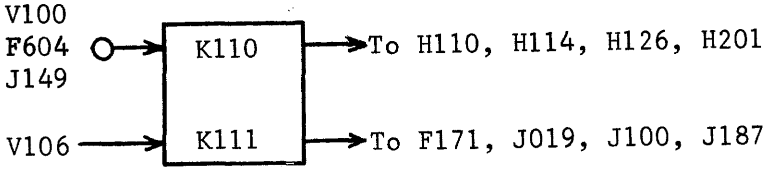
Figure 9-11. Command Timing.

Another valuable publication is the File of Equations. Each logic component in the computer is listed by term number. Also included is the card type, input terms, and output terms. The pin numbers of inputs and outputs aid trouble-shooting procedures. Figure 9-12 is an extract from a File of Equations. It is possible to construct a logic diagram using only that information. For example, the two terms K110 and K111 make up a flip-flop and are both on the same card.

The inputs to K110 are K111(feedback) + (V100·F604·J149). Its outputs feed F171, J019, J100, J187, and K111 (feedback). The card is located on Chassis #2, Q row, card 55 (2Q055A) and is a K31 type card.



The AND gate is actually made at the input to the card but would be logically represented externally.



```
         K106=K107*V120      *J110 V102*J105 V011*V011 I924,
           1 /  I /  2    3 /  4    5 /  6   10 / 11   12   13
                2Q067A H66  H231  K107  K953

         K107=K106*N209,
          15 /  I / 14
                2Q067B H66  F507  H211  H750  J060  J220  K106

         K108=K109*V184 J115*V202 F270 J126*V202 F418 J109 K002,
           1 /  2 /  3    4 /  5    6   10 / 11   12   13   14   15
                2Q041A K58  H217  K000  K109

         K109=N227*N217*M162*K108 K118,
           1 /  2 /  3 /  4 /  5    6
                2Q042A K24  H217  J301  K108  K114  K118

Flip-  ( K110=K111*V100 F604 J149,
       (   1 /  I /  2    3    4    5    6
       (        2Q055A K31  F171  J019  J100  J187  K111
       (
Flop   ( K111=K110*V106,
       (  15 /  I / 10   11   12   13   14
                2Q055B K31  H110  H114  H126  H201  K110

         K112=K113*V084 F403      *V082 F504      *V380 F524 F600,
           1 /  I /  2    3     4 /  5    6   10 / 11   12   13
                2Q056A K36  K113

         K113=K112*N110,
          15 /  I / 14
                2Q056B K36  H110  H750  K112

         K114=K115*V103 K109,
           1 /  I /  2    3
                2Q043A H46  F507  K115

         K115=K114*V102,
           6 /  I /  4    5
                2Q043B H46  K114

         K116=K117*     V117      *V125 K209,
           1 /  I /  2 /  3     4 /  5    6
                2Q050A K32  J014  J181  J213  K099  K117  K203

         K117=K116*N005*N152      *V317,
          15 /  I / 10 / 11   12 / 13   14
                2Q050B K32  H115  J216  K116  T650

         K118=K109*N151 J455*V000 K207 K204*V280 J063 J194,
           1 /  2 /  3    4 /  5    6   10 / 11   12   13   14   15
                2Q040A K58  K109

         K120=K121*V010 J499 J583      *N010 H080 J499 K497 F328,
           1 /  2 /  3    4    5     6 / 10   11   12   13   14   15
                2Q076A K12  H220  H220  J119  J214  J493  K121  V109
```

Figure 9-12. File of Equations.

Although the publications illustrated are not actually part of the control section and not even part of the computer, they do provide valuable information and enhance a better understanding of the computer.

You should now be able to generate the necessary commands to execute any instructions with the possible exception of a MUA or DVA. The following questions should provide an indication as to how well you understand the Control Section. The answers are at the end of the chapter.

Answer the following questions:

1. What is the function of the control section?

2. How many sequences does the control section have? List them and explain when each would be used.

3. What is the reason for having two different timing chains?

4. Why would arithmetic timing be used to execute an INI instruction?

5. Referring to figure 9-10, how would execution of an ADA instruction differ from the execution of an SBA instruction?

6. How could you modify figure 9-10 to enable the execution of an LDQ instruction?

7. Using only the block diagram (figure 9-2), generate the command timing for the INI instruction (15 3 00020). Assume the instruction is already in the F register.

8. What is the function of the 9-bit F2 register?

9. Which instruction would use only the RNI and RADR sequences during its execution?

10. Why is the use of control delays so prevalent in the control section?

11. Why is a flip-flop (Storage Request) set each time a request is sent to storage?

12. Assuming a two phase-time wait between the Memory Request and the Reply, how much time would be required to execute a 30 1 12345 instruction? Use figure 9-10 and give your answer in microseconds. Start with the output of V014 and stop when it outputs a pulse the second time.

SUMMARY

The third section of a digital computer discussed was the control section. Its responsibilities are quite varied, which tends to make it more difficult to understand than either the Arithmetic or Memory sections.

The control section generates the commands required to read and execute instructions and all of its operations are meant to achieve that overall objective. Place yourself in the role of a design enginecr and, with the block diagram, make a sequential list of the commands necessary to cause an instruction to be executed. When a list has been made for each instruction in the repertoire, combine the lists together and you will have the commands to be generated by the control section.

The execution of similar instructions is normally controlled by a sequence. The ROP sequence was used for those instructions requiring an operand from memory and the STO sequence was used to place an operand into memory. The RNI sequence was used to read instructions from memory. Each instruction could have its own sequence, but this method would require more hardware and would, therefore, increase the cost of the computer.

The design of the control section is a determining factor in the speed of a computer. Sequential operations should follow each other as closely as possible to save time. However, timing problems may occur if a command is generated before the previous operation is complete.

The final section to be discussed is the I/O section.

After its functions have been presented, the four sections of a computer will be integrated with the I/O equipments to constitute a computer system. At that time, the importance of the control section will become even more significant.

ANSWERS TO REVIEW QUESTIONS

1. Controls the execution of instructions by issuing the proper commands in the proper sequence.

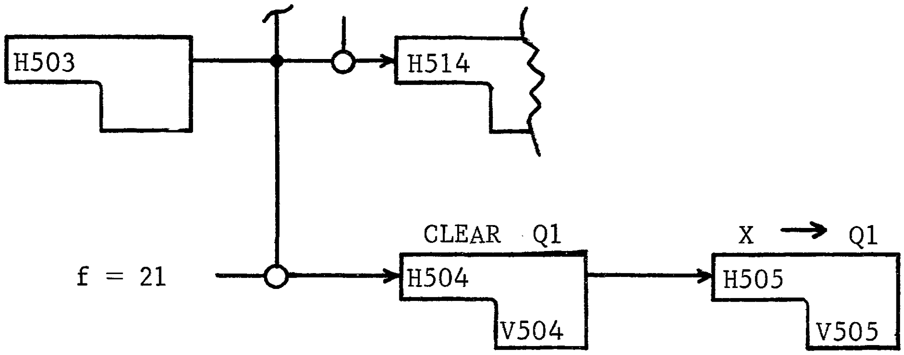
2. RNI Read instructions.
RADR Indirect Addressing.
ROP Execution of an instruction requiring an operand from storage.
STO Execution of an instruction storing an operand in memory.

3. Arithmetic Timing controls those operations concerned with the arithmetic section, whereas main timing generates the commands to read instructions. The two timing chains provide the capability for performing two simultaneous operations -- executing one instruction and reading another.

4. The arithmetic $(m+(B^b) \rightarrow B^b)$ is performed in the adder which requires the use of arithmetic timing.

5. V500 would initiate a $\bar{Z} \rightarrow X$ command for the SBA instruction and a $Z \rightarrow X$ command for the ADA instruction; the only difference.

6.



7.

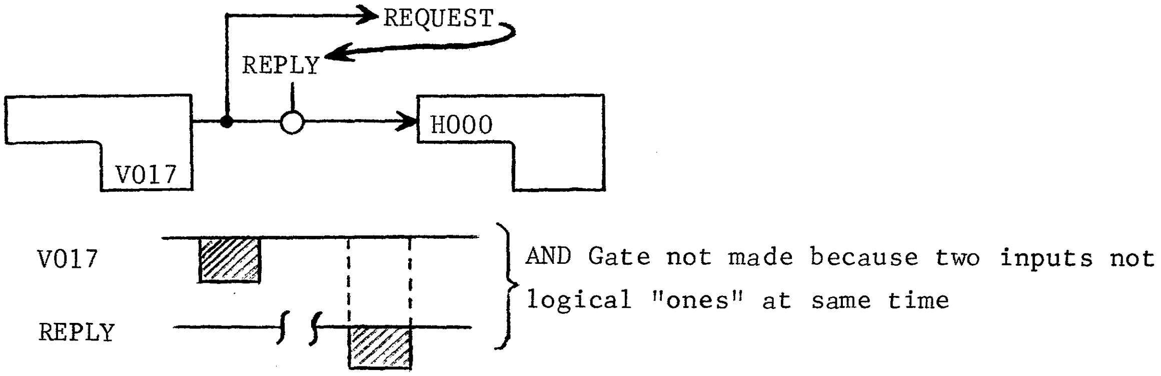
MAIN TIMING	ARITHMETIC TIMING
START ARITHMETIC	
	F address→A2
	$(B^b)\rightarrow X$
Clear B^b	
Adder → B^b	

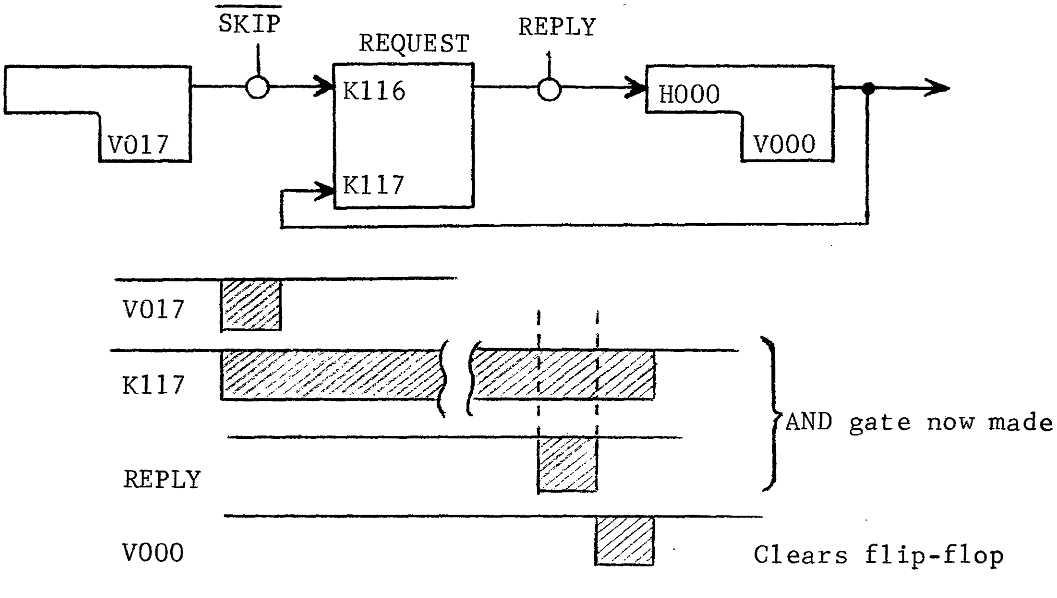
8. Arithmetic timing is controlled by function translators fed by F2, whereas main timing is controlled by F1 translators. This allows arithmetic timing to process one instruction while main timing is executing the next (providing the next instruction does not require arithmetic timing and the first requires a long time -- 5-10 microseconds -- for execution).

9. A UJP with indirect addressing (01 4 12345)

10. Commands must be issued in sequence with time allotted between operations to one to be accomplished before the next is initiated. The control delay provides the necessary time interval before the next operation is initiated. Each control delay is synchronized by the same master clock and, therefore, provides exactly the same delay as all others in the computer.

11. The output of a control delay is a pulse of only 62.5 nsec in duration. If the Reply was not there to make the AND gate when the delay had a "1" output, the pulse would die and the computer would stop. By using the flip-flop, the logical "1" is preserved for as long as necessary until the Reply is received from memory.


REQUEST
REPLY
H000
V017
V017
REPLY
AND Gate not made because two inputs not logical "ones" at same time

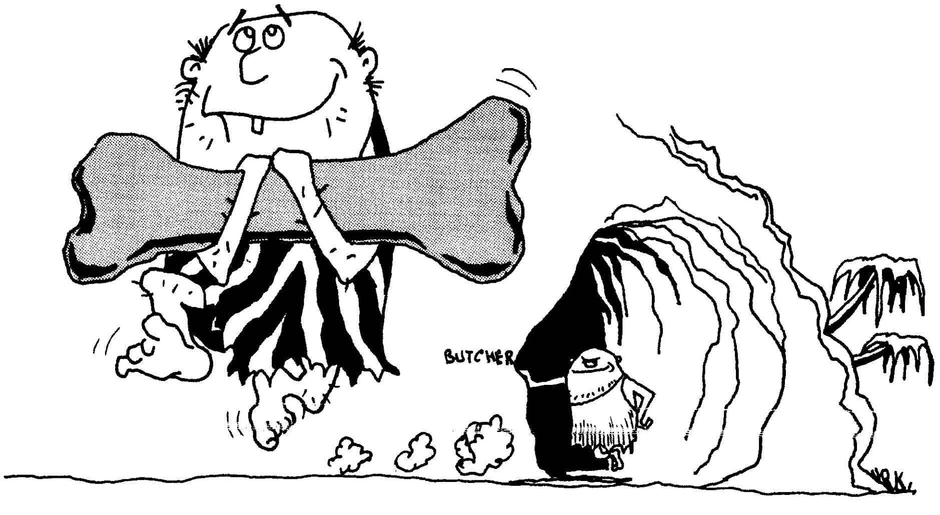


12. About 39 phase times, or less than 2.5 microseconds.

chapter x

Input-Output Section

Input



Output

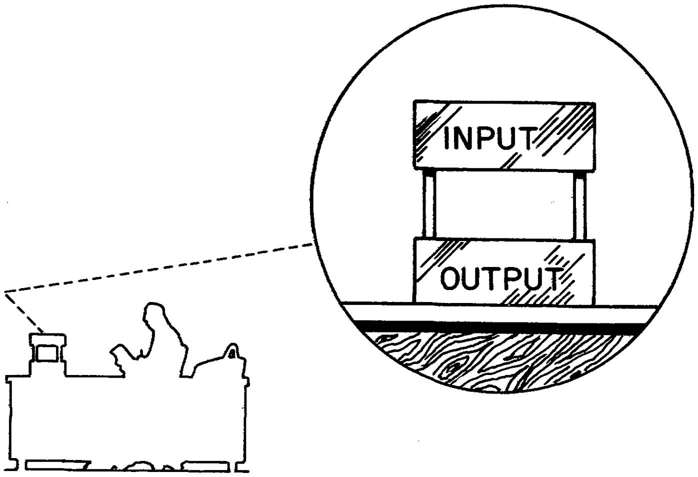
CHAPTER X

INPUT-OUTPUT SECTION

INTRODUCTION

The final section of a digital computer to be discussed is the I/O section. Although input and output are two different operations, they are both controlled by the same section. References are sometimes made to the five sections of a computer but because input and output are combined, this text will consider them collectively as the fourth section.

A different perspective of our man-sized computer indicates that there are two baskets on his desk. One is labelled IN and the other is labelled OUT -- indicative of the input and output sections of the computer.



You learned in the preceding chapter that all operations, including input and output, are under the supervision of the control section. The part of the control section that supervises input and output is known as block control because blocks of data are being transmitted between the computer and I/O devices. It is the objective of this chapter to explain the function of the common I/O devices and how each communicates with the computer.

I/O BLOCK DIAGRAM

The block diagram (Figure 10-1) illustrates how an I/O device is connected to the computer. Each I/O device must communicate with the computer through a controller (synchronizer) which synchronizes the slow mechanical operations of the device with the much faster electronic operations of the computer.

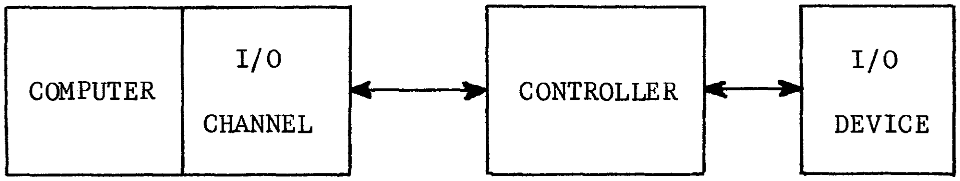


Figure 10-1. I/O Block Diagram.

The computer may have as many as eight I/O channels, each capable of bidirectional data transfer. Each channel may have up to eight controllers connected to it for a maximum of 64 controllers. Some controllers, such as those for magnetic tape units, can accommodate eight units. If a fully-expanded system had nothing except tape units as I/O devices, the computer could have 512 tape units connected to it (Figure 10-2).

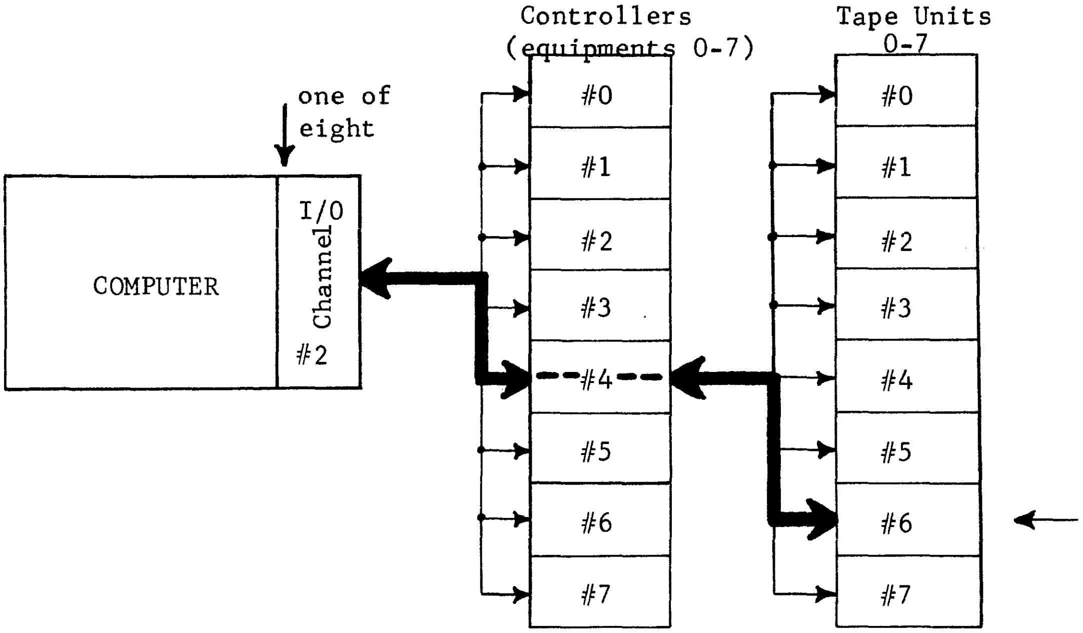


Figure 10-2. I/O Channel-Controller-Unit Connections.

In order to communicate with tape unit #6, an instruction would be executed to connect to channel #2, Equipment #4, and Unit #6. Once connected, another instruction would be executed specifying READ (input to computer) or WRITE (output from computer) and the specific block of memory to be used.

Each type of I/O device must have a special type of controller. For example, a system consisting of a card reader, a card punch, a line printer, and four tape units would have a card reader controller, a card punch controller, a printer controller, and one tape unit controller (Figure 10-3).

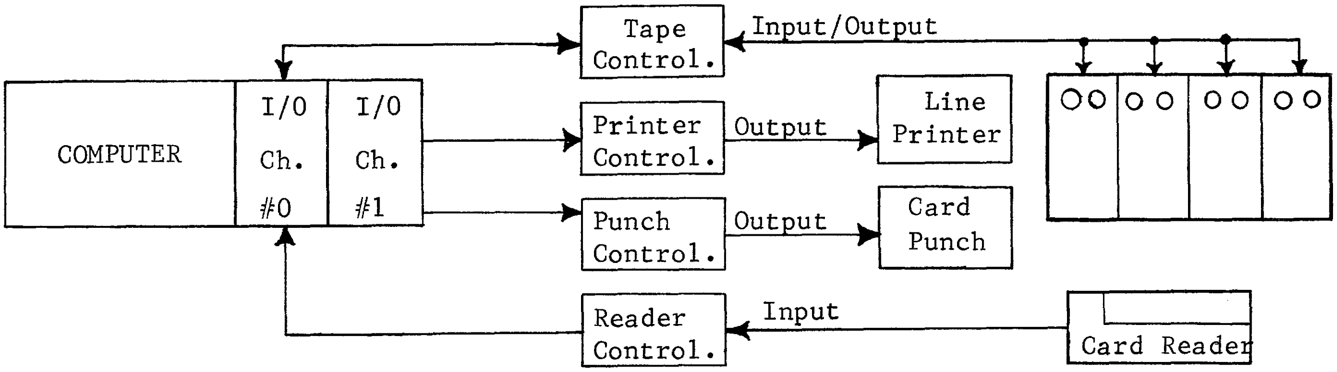


Figure 10-3. Computer System.

All four controllers could have been connected (physically) to a single channel, but the efficiency of the system would be diminished.

Many operations use tapes and the printer; if on the same channel, only one device could be connected (electrically) at any specific time.

TYPES OF PERIPHERAL EQUIPMENTS

There are many different types of peripheral devices. Most of them are either for input or for output. For example, some of the more common output devices are:

1) Card Punches
2) Magnetic Tapes
3) Paper Tape Punches
4) Line Printers
5) Magnetic Disks
6) Visual Displays

7) Plotters

8) Typewriter

Some of the more common input devices are:

1) Card Readers

2) Paper Tape Readers

3) Magnetic Tapes

4) Magnetic Disks

5) Optical Page Readers

6) Typewriter

Magnetic tape units, magnetic disks, and the typewriter are listed as both input and output devices. Each is capable of two-way communication with the computer. All I/O operations must be initiated by the computer. This is accomplished by reading an instruction from the program in the same manner as those instructions explained in chapter IV.

I/O INSTRUCTIONS

An I/O operation may be initiated by executing three instructions. If an input from a magnetic tape unit is desired, the program must first connect to the specific equipment, next specify the desired mode of operation, and then initiate the input.

The connection to the desired equipment is made by executing a Connect instruction. The instruction requires two consecutive memory locations and would be written according to the following format.

		23 18	17 15	14 12	11 00
CON	P	77	0	Ch	X
	P+1	Reject instruction			

Ch = I/O channel designator, 0-7

X = 12-bit connect code. Bits 09-11 select the controller and bits 00-02 select a unit connected to the controller (applicable to tape unit controller only).

Referring to Figure 10-2, the instruction to connect to tape unit #6 would appear as 770 24XX6. The instruction sends the 12-bit connect code to the controller on the selected channel. If the connection is made, the controller sends back an immediate Reply and the next instruction is read from P+2. If a Reply is not received within 100 microseconds, or if the channel is already busy, the reject instruction is read from P+1. The reject instruction may be a jump back to P, keeping the computer in a tight loop until the connect is made. The fallacy with this procedure is that the computer may "hang up" in an indefinite loop if the controller cannot be connected.

A Select Function instruction follows the connect and is used to specify the desired type of operation to be performed. The Select instruction also requires two consecutive memory locations.

SEL	P	77	1	Ch	X
	P+1	Reject Instruction			

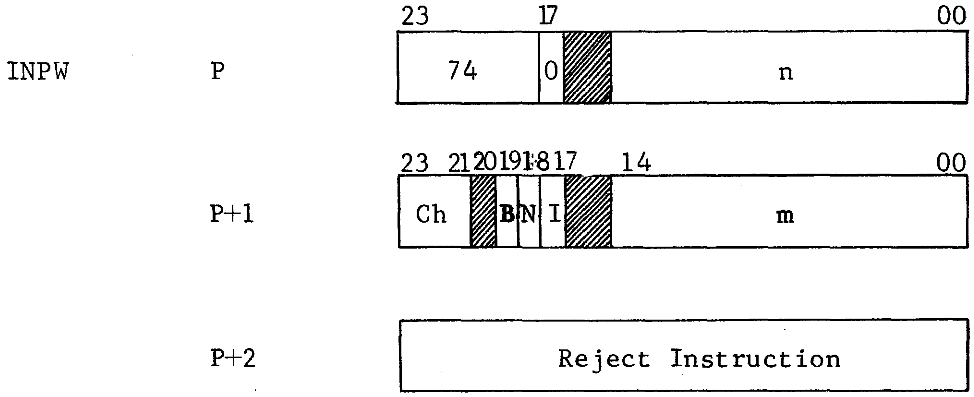
Ch = channel designator

X = 12-bit function code. Each device has its own unique set of function codes. They are listed in a booklet which contains the codes for each controller. If a tape controller has been connected, a Select instruction could be used to select a specific recording density, binary or coded information, tape motion codes, or one of several others. Each will be explained with the individual unit.

The reject instruction at P+1 serves the same function as for the Connect. A reject will occur if:

1. A connect has not previously been made.
2. The 12-bit code is not defined for that controller.
3. Equipment busy or not ready.
4. Channel busy.

After the equipment has been connected and the appropriate operations selected, the actual input or output is initiated. An input is initiated by the INPW (word addressed input to storage) instruction and an output by the OUTW (word addressed output from storage) instruction. Each instruction requires three memory locations, but this is all that is required to transfer a block of data into or out of storage. Once the operation is initiated, the computer continues with its program simultaneously with the data transfer.



NOTE: Shaded areas should contain zeros.

B = backward storage (data is entered starting al last address first)

Ch = I/O channel designator

I = Interrupt computer upon completion

N = "0" for assembly, "1" for no assembly. Input can be into lower 12 bits only or into all 24 bits of storage location.

m = address of first storage location to which data is to be transferred.

n = last word address+1 of data block. If n = 01000, last word will be transferred into storage location 00777. The instruction is read from two storage locations, P and P+1. If the channel is busy, the reject instruction is read from P+2. If not, the next instruction is read from P+3.

The only difference between the INPW and OUTW instruction format is the function code. 74 0 indicates INPW whereas an OUTW instruction would have a function code of 76 0.

The same three instructions, CON, SEL, and INPW or OUTW, are used to initiate operations on all types of equipment except the typewriter. The typewriter is not connected through an I/O channel and communications between it and the computer are accomplished with two special instructions, CTI (Set console typewriter input) and CTO (Set console typewriter output).

A program to read 100 words from the tape unit indicated in Figure 10-2 could appear as follows:

```
00100      77 0 24006          Select channel #2, Equipment #4,
                               Unit #6

00101      Reject instruction

00102      77 1 20001          Select binary mode of operation

00103      Reject instruction

00104      77 1 20004          Select density of 200 frames per inch

00105      Reject instruction

00106      74 0 01001          Input - Last word will be in 01000

00107      20 0 00700          Channel #2 - First word to be trans-
                               ferred into 24 bits of location
                               00700. If bit 18 is set (21 0 00700),
                               the information would be placed in
                               only the lower 12 bits of each
                               storage location (00700-01000) and
                               only half as much data would be
                               transferred from the tape unit.

00110      Reject instruction

00111      Next instruction of program.
```

The instructions from locations 00101, 00103, 00105, and 00110 will not be read unless the preceding operation cannot be initiated.

Block control supervises the entire input operation and leaves the computer free to execute instructions. All eight channels can be busy at the same time but would alternately share block control. One word would be transferred via channel 0 followed by a word on another channel and the third word via a third channel. Block control has its own memory consisting of 100_8 24-bit storage locations called a register file.

Sixteen of those locations are permanently allocated specifically for the eight I/O channels. When either an INPW or OUTW instruction is executed, the first address and last address+1 are placed in the appropriate register file location. For example, as the computer executes the INPW instruction 740 01001, the equivalent of 01001 (last word address+1) will be automatically placed in the register file at address 12 (channel #2). The instruction at P + 1 contains the first word address (00700) and the equivalent of it will be placed in the register file at address 02.

If the INPW instruction had specified channel #7 instead of #2, the LWA + 1 and FWA would have been placed in register file locations 17 and 07 respectively. Remember, the register file is a separate memory from computer storage and is reserved specifically for block control. The remaining register file locations are for other block control operations not discussed in this text.

TRANSFER OF DATA

Once the input has been initiated, the transfer of data is supervised by block control as it communicates with the controller through the I/O channels. There are several types of I/O channels but this discussion will assume that the channel can accomodate a full 24-bit data word. The N designator (bit 18) of the INPW instruction would then be a "1", since no assembly is required.

The channel requests data from the controller which, in turn, obtains two 6-bit bytes of information from the tape unit. The controller then sends the 12 bits of data to the I/O channel which are placed in the upper half of a 24-bit register in the I/O channel (Figure 10-4).

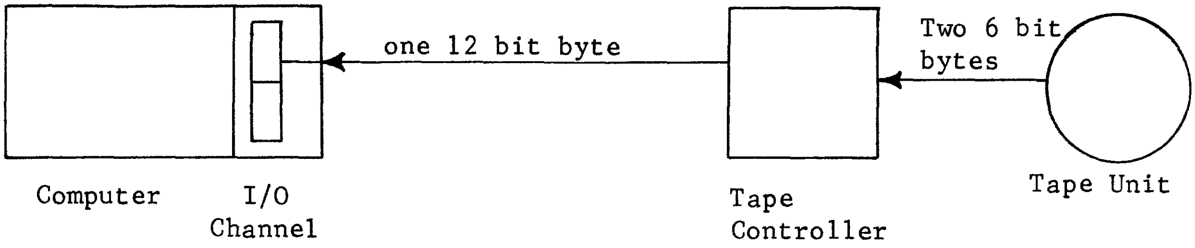


Figure 10-4. Data Transfer.

As soon as the I/O channel has received the first 12-bit byte of data, it sends another request to the controller. The second 12-bit byte is sent to the I/O channel and placed in the lower half of the register.

The 24-bit data channel now has a 24-bit word ready to be stored in memory. It notifies block control by issuing a channel request. Block control has a channel scanner which alternately monitors each channel. When the scanner reaches the requesting channel number, it stops and the request is then recognized by block control.

All memory modules are connected together through a common data bus which could be in use by the computer and not immediately available (Figure 10-5).

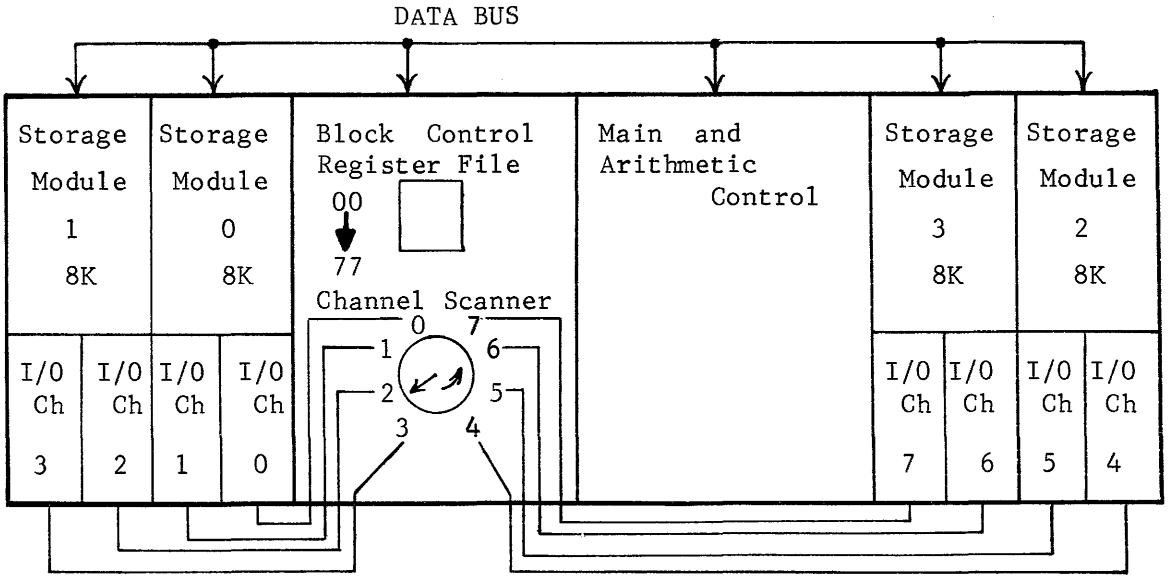


Figure 10-5. Eight Channel System with 32K of Storage

As soon as block control acquires use of the data bus, it notifies the requesting I/O channel. The register file (location 02 for channel #2) contains the address in computer storage into which the data word will be placed. Block control reads that address from the register file and then makes a storage request. The effective starting address is then updated by one and placed back in the register file at address 02. When the operation is repeated for the next data word, the word will go into the next consecutive memory location.

As soon as computer storage recognizes the request, block control gates the word from the I/O channel into a data bus register and into computer storage. The last word address+1 is read from the register file (address 12 for channel #2) and compared with the current address after it has been updated. If they are equal, the transfer of data is complete; if not, another request for data is sent to the controller and the process is repeated. This type of operation is known as a buffered data transfer because computer operation is completely independent of the I/O operation. A non-buffered operation would require the computer to suspend its execution of instructions until the I/O operation is finished.

An output operation would be quite similar except that the transfer of data is from storage to the I/O device.

FUNCTIONAL ANALYSIS OF I/O EQUIPMENTS

Now that you understand how the computer communicates with Block control, the I/O channel, and the controller, the function of several types of I/O devices and their controllers will be explained.

MAGNETIC TAPE

One of the more common types of I/O devices is the magnetic tape unit. As previously stated, up to eight units may be connected to one controller but only one of the eight may be electrically connected and communicating with the computer at any given time. If other controllers are connected to the same I/O channel, only one of them may be in communication with the computer.

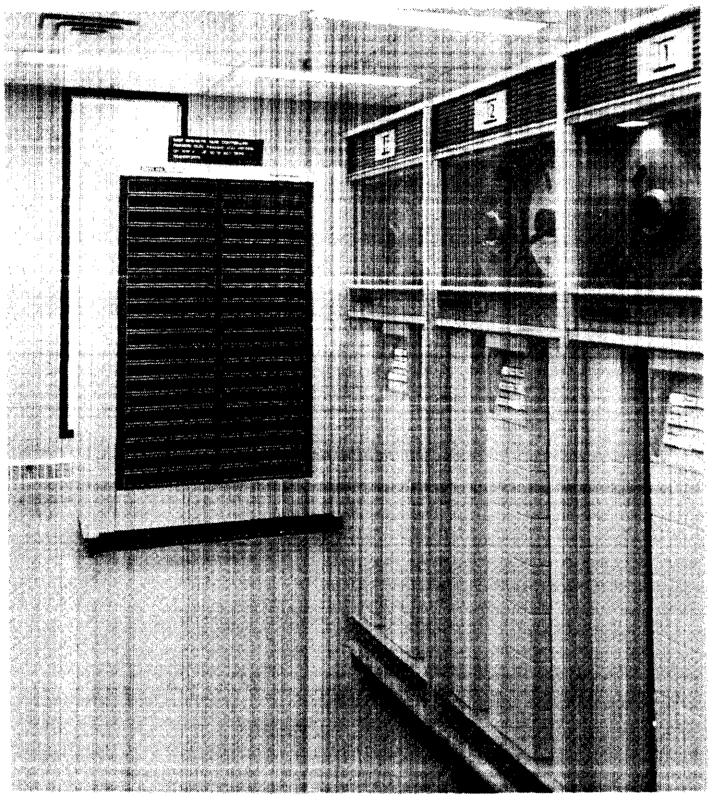


Figure 10-6. Magnetic Tape Units and Controller.

Three magnetic tape units and the controller are illustrated in Figure 10-6. Magnetic tape is used as a storage medium for large volumes of data. Information may be recorded on tape and repeatedly read back into the computer as desired. The tapes may be stored for long periods of time with very little degeneration in the quality of information. If data is lost, automatic checking circuits notify the computer by causing a program interrupt.

The primary function of the tape unit is to move tape past the heads to enable the reading or writing of data. The physical layout of the tape unit is illustrated in Figure 10-7.

The supply reel feeds tape into the right vacuum chamber, over a drive capstan and under the head assembly, over the second drive capstan, through the left vacuum chamber and is taken up by the take-up reel. Tape motion is controlled by the two drive capstans, each turning in opposite directions. The capstans are hollow and can have either vacuum or air pressure applied, depending upon the desired direction of tape travel.

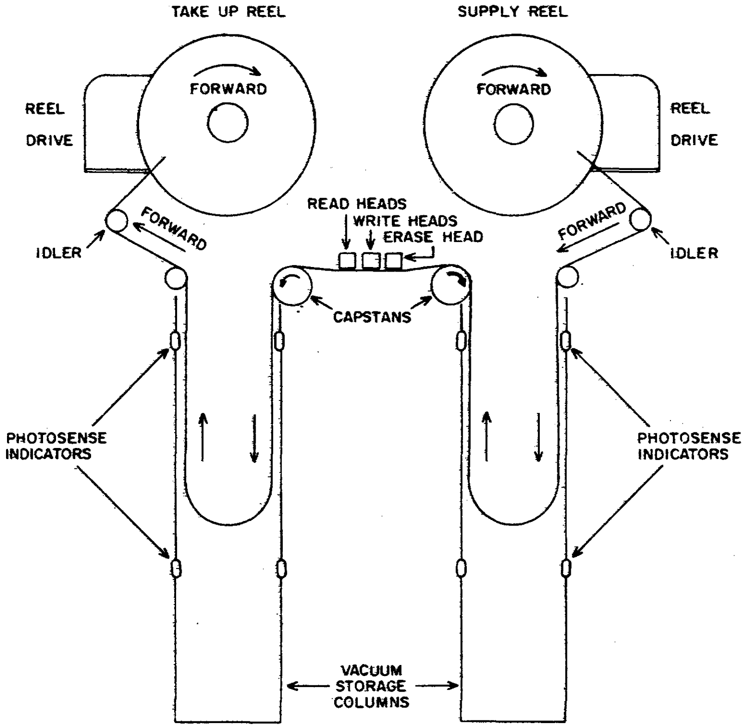


Figure 10-7. Magnetic Tape Unit

If it is desirable to move tape forward (from supply to take-up), pressure is applied to the right capstan and vacuum is applied to the left capstan. The pressure forms an air cushion between the tape and the clockwise-turning capstan. The tape is forced into contact with the counter-clockwise turning capstan by the vacuum and the tape moves under the head assembly from right to left. Tape speed of the units illustrated in Figure 10-6 is 150 inches per second, but may vary with different types of units.

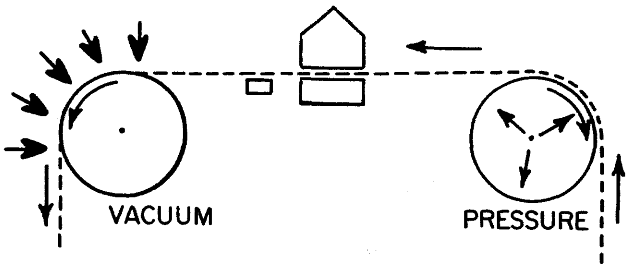


Figure 10-8. Forward Tape Motion.

Reverse motion is attained by reversing the pressure and vacuum in the two capstans. Tape motion is stopped by applying pressure to both capstans and vacuum to the brake (Figure 10-9).

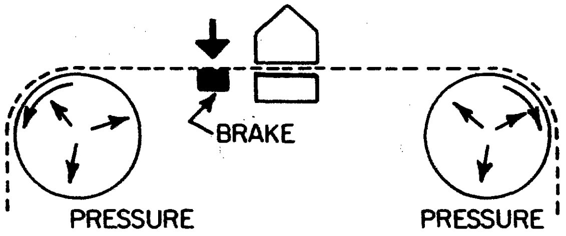


Figure 10-9. Stopping Tape Motion.

Tape motion is controlled by the two drive capstans and the brake. The motion of the take-up and supply reels is controlled by two photocells in each vacuum column (Figure 10-7). The loop of tape in each column acts as a buffer for tape motion when the tape is suddenly started or stopped. The photocells in the left vacuum column control the take-up reel and the photocells in the right column control the supply reel. If the drive capstan moves tape out of the right column faster than the supply reel is putting it in, the loop becomes shorter. As soon as it moves above the top photocell, the detector signals the supply reel motor to turn forward. If tape moves below the bottom photocell, it signals the supply reel motor to turn in the reverse direction. In the left column, the top photocell signals the take up reel to turn in the reverse direction while the bottom photocell sends the forward signals. A constant vacuum in each chamber maintains a stable loop of tape and prevents flutter.

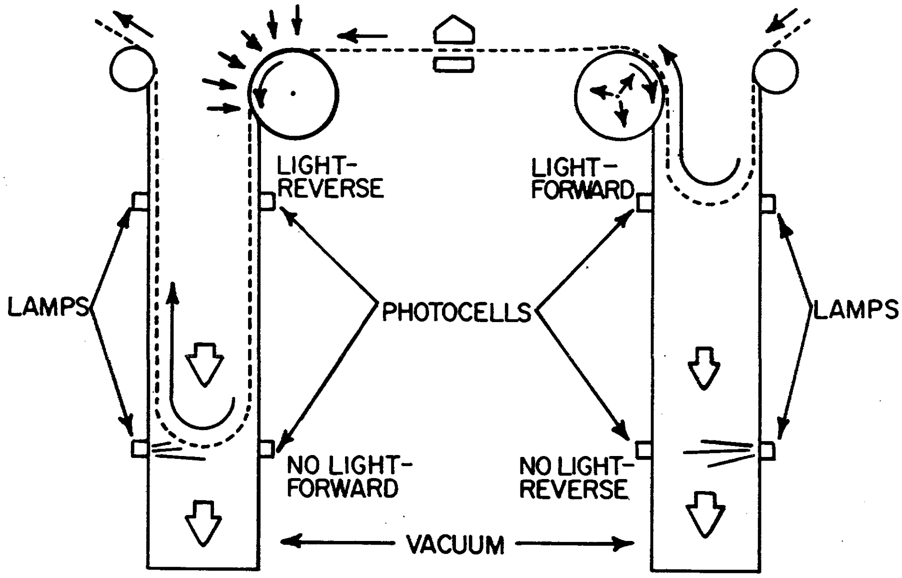


Figure 10-10. Vacuum Column Photocells.

Tape Format

The tape consists of a plastic-base film of 1 to 1½-mil thickness coated on one side with ferrous oxide suspended in an organic binder. Recording is accomplished by magnetizing minute areas of the tape surface in a pattern determined by the data to be recorded. The recording area of the tape is 2400 feet in length plus an extra ten feet at the beginning for a leader and 18 feet at the end. A reflective marker at each end of the recording area notifies the tape unit when either end has been reached. The first marker (10 feet from beginning of reel) is the Load Point marker and the one at the end of the recording area is the End Of Tape marker. The beginning and end of the recording area is detected by two photocell sensors. When the reflective marker passes over its respective sensor, the change in reflected light is detected (Figure 10-11).

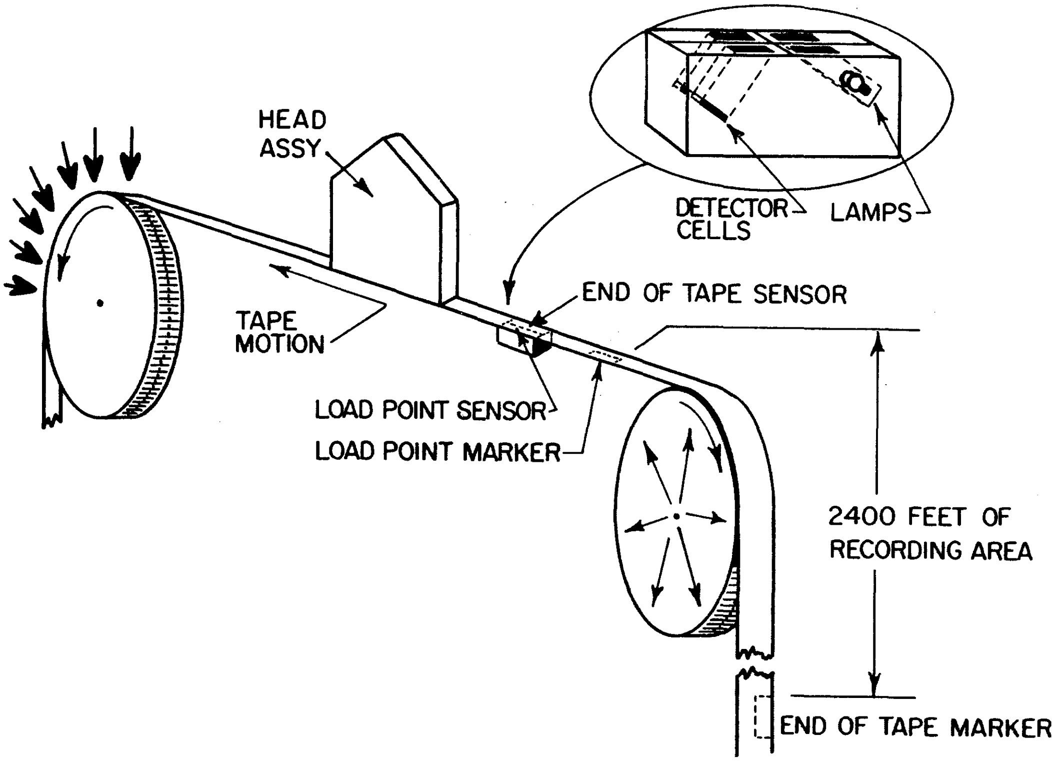


Figure 10-11. Load Point and End Of Tape Sensors.

The Load Point marker is placed on the outside edge of the tape away from the tape unit whereas the End Of Tape marker is placed on the inside edge toward the unit. Both markers are on the plastic side (bottom) away from the recording surface.

Tape width may vary but common widths are from ½ inch to 1 inch. If ½-inch tape is used, recording is accomplished by simultaneously magnetizing small areas that lie in seven parallel tracks. Six tracks are for the recording of data. The seventh is used to record a summation check bit of the other six and is known as the vertical parity track.

Writing on Tape

When a block of data is to be written on tape, it leaves computer memory as a series of 24-bit words. The first 24-bit word reaches the 24-bit data channel where it is disassembled into two 12-bit bytes. The upper 12 bits are then sent to the tape synchronizer for further disassembly into two 6-bit bytes. The byte corresponding to the upper six bits of the word is then recorded on tape along with the parity bit generated by the synchronizer. The synchronizer then sends the next byte and parity is sent to the tape unit and that frame is recorded on tape immediately following the first.

The data channel then gates the lower 12 bits of the word to the synchronizer which then records two more 7-bit frames on tape. Another word is then read from storage which is again recorded as four consecutive frames of seven bits.

The parity bit for each frame serves as a means of detecting lost data. After a frame is written on tape, the frame passes the read head where the frame is read and compared with the parity bit. If the recording area is excessively worn and information failed to be written, it is immediately detected as a Write Parity Error. Assuming that the tape was good at recording time but received damage through subsequent use, the lost data would be detected as a Read Parity Error the next time the tape is read.

The parity bit may be recorded in either of two formats -- odd or even. Formats cannot be mixed during a given operation. Odd parity indicates that the total number of bits in the track are odd. If the total number of bits in the track (including the parity bit) is even, the recording is considered to be in even parity. For example, if the word from memory was 12345670, it would be written on tape in four frames as

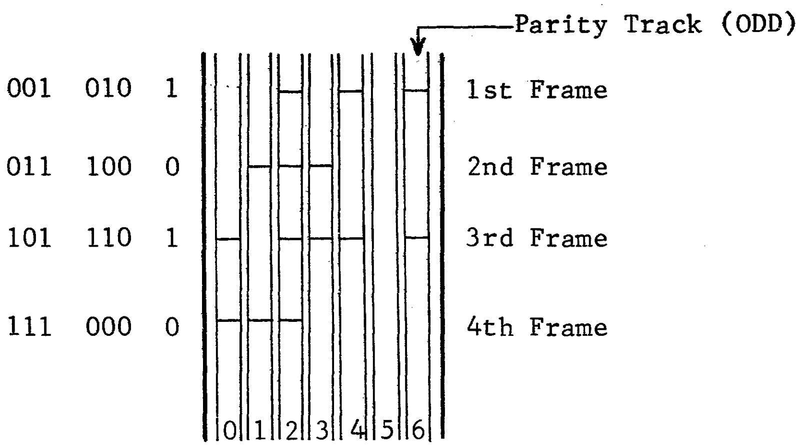
000 010

011 100

101 110

111 000

The binary recording of data is always written in odd parity and would appear as



If data is to be written in binary coded decimal (BCD) format, even parity is used and the total number of ones per frame is always even.

The frames would actually be much closer together than indicated. Information may be recorded at densities of 200 (low), 556 (high), and 800 (hyperdensity) frames per inch. At hyperdensity, 200 words could be written on one inch of magnetic tape and the entire contents of a 32K memory could be recorded on less than 14 feet of tape. Figure 10-12 illustrates how much tape is required to record 350_8 words (232_{10}) on tape. The same block of information was recorded at three different densities and then extracted from tape by a special process. The top track contains the parity bits and is known as the parity track.

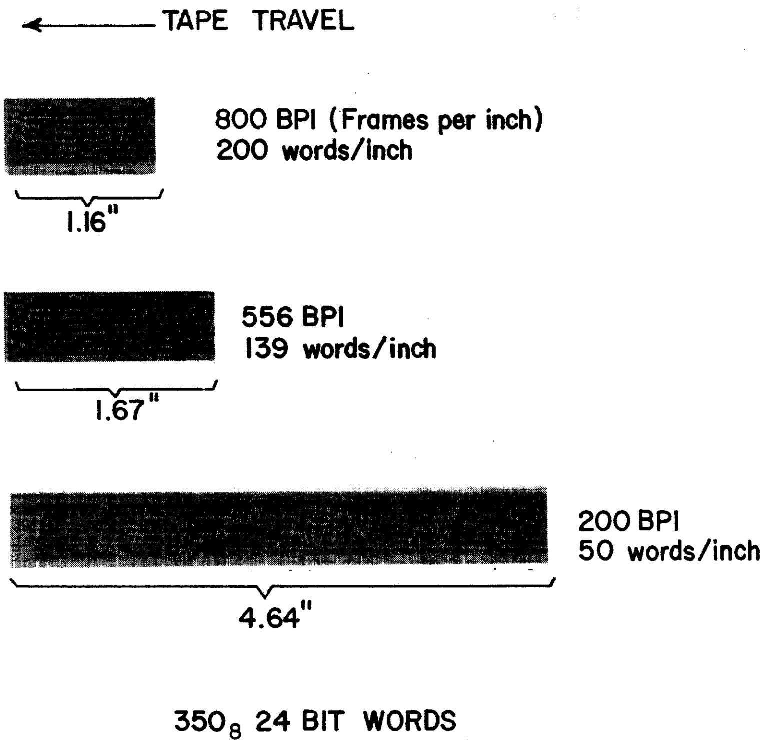


Figure 10-12. Information Recorded on Tape.

The reading and writing of data on tape is accomplished by moving the tape under the head assembly. The tape first passes under the erase head. If writing, the erase head removes all old information and leaves the tape magnetically clean. Tape then moves under the write heads, one for each track, which records the information and then moves under the read heads where it is immediately read back to check parity. The head assembly is illustrated in Figure 10-13.

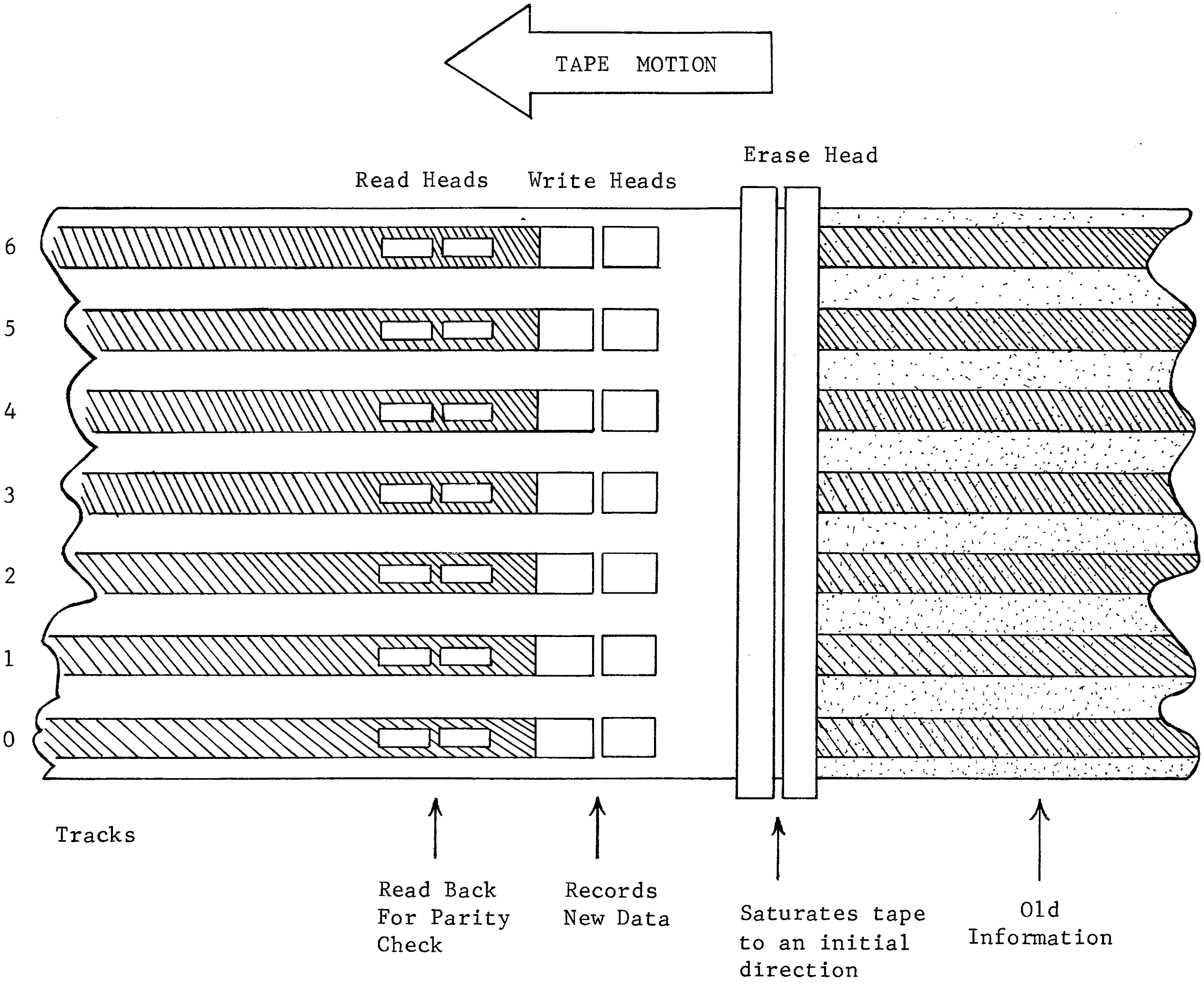


Figure 10-13. Head Assembly (Top view) WRITING.

Notice that the write heads record the information in relatively wide tracks but the read heads are not as wide and read only the center of the track. The erase head overlaps both edges of the tape and all old information is removed.

If the computer is reading information into storage, the Erase and Write heads are disabled to prevent the alteration of previously stored data (Figure 10-14).

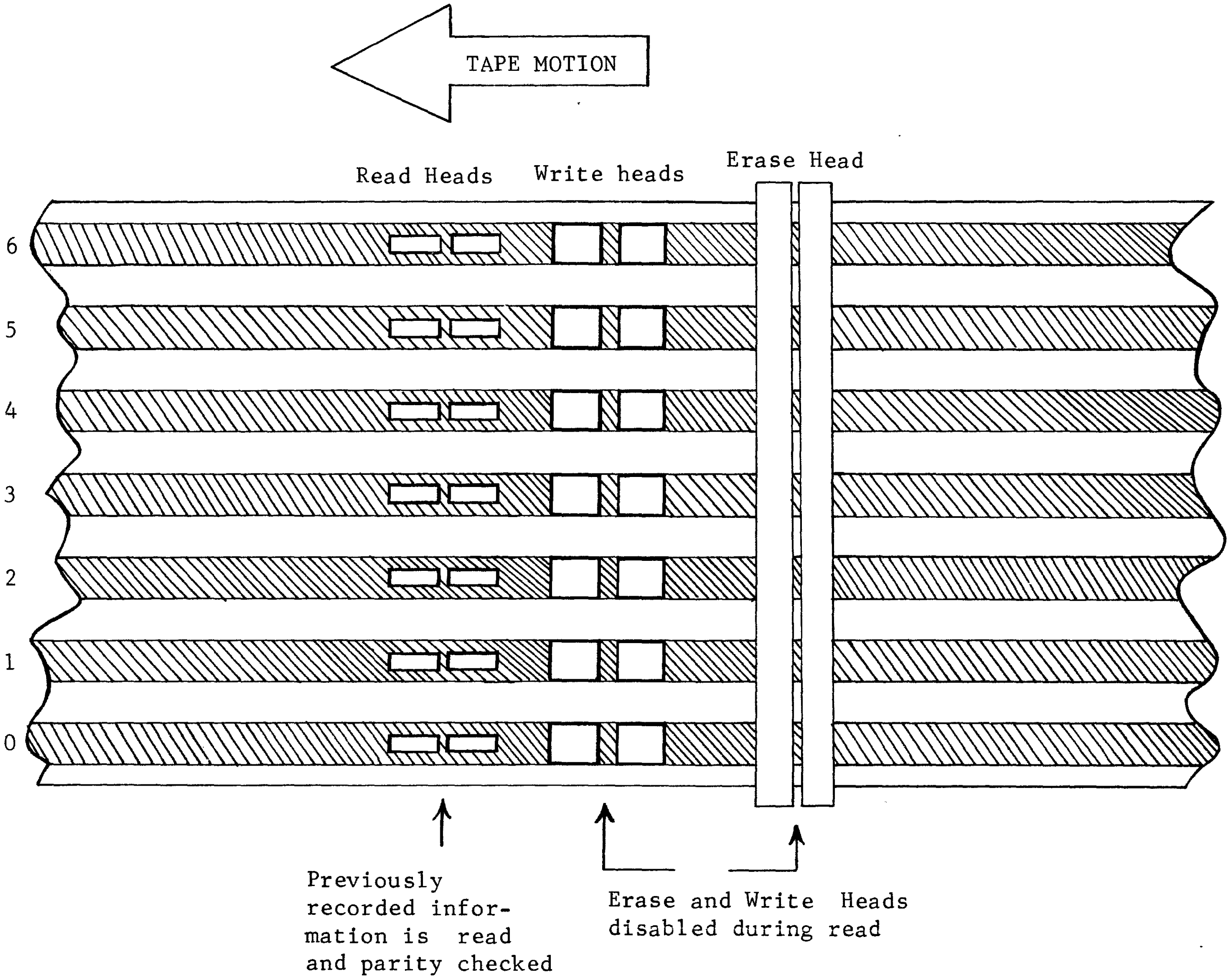

TAPE MOTION
Read Heads
Write heads
Erase Head
6
5
4
3
2
1
0
Previously recorded information is read and parity checked
Erase and Write Heads disabled during read

Figure 10-14. Head Assembly Reading (Top view).

Each head is a tiny electromagnet. The minute gap at the poles is adjacent to the recording surface. Current flow through the coil of the magnet produces a magnetic field at the gap which magnetizes a tiny spot on the tape (Figure 10-15).

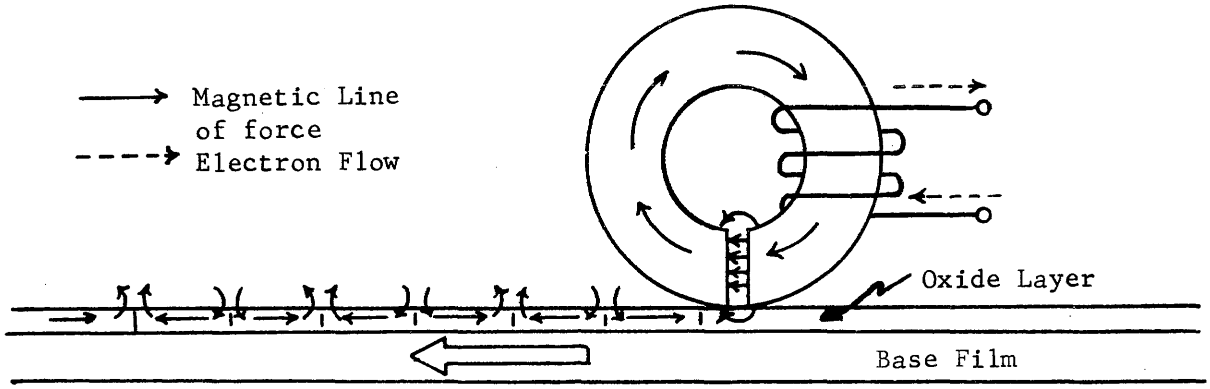


Figure 10-15. Write Head Configuration.

Current is continuously flowing through the Write head during a write operation and does not cease between frames. Each time a "1" bit is to be recorded on tape, the direction of current through the coil is reversed. This method of recording is referred to as the "non-return to zero, change on ones, indiscrete" method. Indiscrete refers to the fact that a change in current flow through the coil, regardless of direction, is indicative of a "1" bit. Figure 10-16 illustrates how one bit of several frames would be recorded on tape.

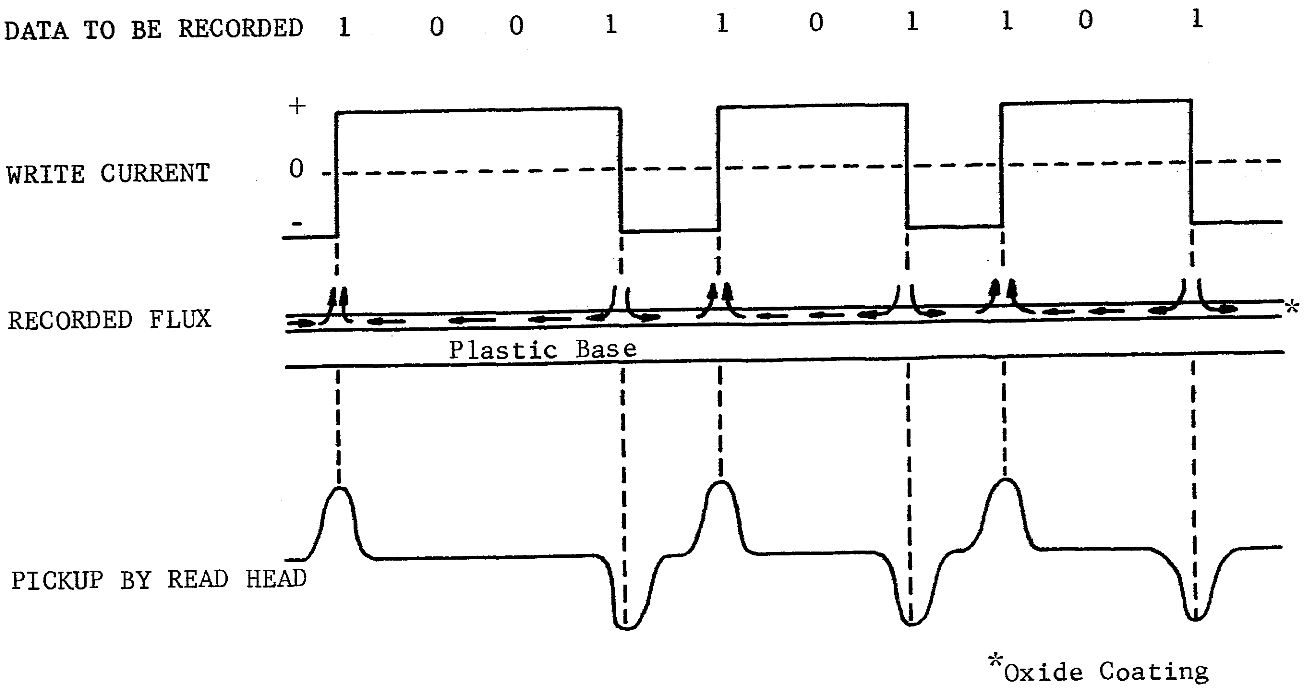


Figure 10-16. Non-Return-To-Zero (NRZ) Recording.

As the tape passes under the Read head, the magnetized areas induce current flow in the pickup coil. Each pulse of current through the pickup coil indicates the presence of a recorded "1" on tape.

Since there are seven tracks across the tape, seven individual heads are required for writing and seven more are required for reading.

Each time an output to magnetic tape is initiated, the entire block of data is written in continuous frames and forms a record. When recording in odd parity, at least one track in each frame would contain a "1" bit. If a memory location contained all zeros, the four frames would be written on tape as follows:

$$\left.\begin{array}{lll} 000 & 000 & 1 \\ 000 & 000 & 1 \\ 000 & 000 & 1 \\ 000 & 000 & 1 \end{array}\right\} \quad \text{binary representation}$$

When information is read from tape, each frame is identified by a "1" in at least one of the tracks and a Sprocket pulse is generated. A BCD character of 00 is not used when writing on tape; all other characters would generate a Sprocket pulse. When all of the information has been read, the absence of Sprocket pulses indicates that the end of the record has been reached.

Longitudinal Parity

Another type of parity is also generated as data is recorded on magnetic tape. Longitudinal parity provides a means of checking all of the "1" bits in each track for the entire record, including the parity track. When the entire record has been recorded, a Check Character is written in even parity after a skip of three blank frames. For example, if the record consisted of only two words from memory and these words were 71 72 73 74 and 41 32 23 14, they would appear on tape as follows:

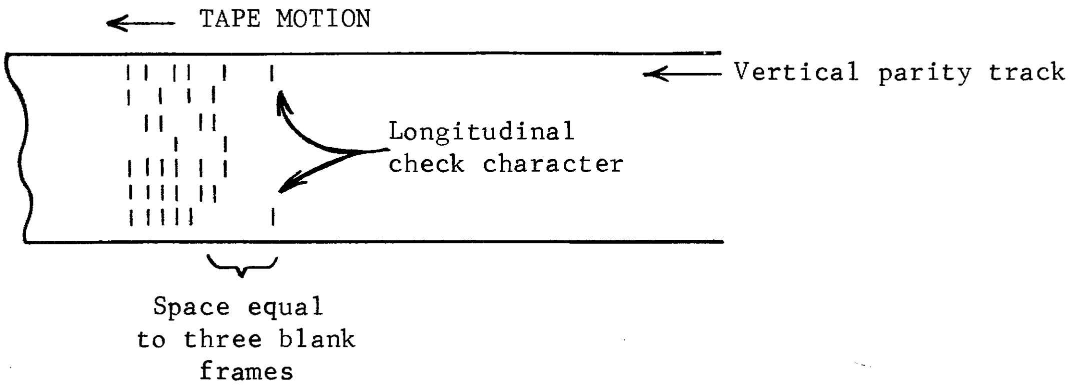

TAPE MOTION
Vertical parity track
Longitudinal check character
Space equal to three blank frames

Figure 10-17. Longitudinal Check Character.

Vertical parity is not checked on the Check Character but is checked on all frames containing data.

If another output is initiated on the same tape unit, the controller will cause tape to move 3/4 of an inch before starting to record the new record. The record gap separates the blocks of data and allows a specific record to be located on tape. After the desired number of records have been recorded, a File Mark may be written by executing a Select instruction.

$$\underbrace{77\ 1}_{\text{SEL}}\qquad \underset{\substack{\uparrow\\ \text{CH}}}{2}\underbrace{0015}_{\text{CODE FOR "WRITE FILE MARK"}}$$

When the Write File Mark instruction is executed, the code is sent to the controller. The controller causes tape to move six inches and then writes an octal 17 in even parity as an End of File mark. After a skip of three blank frames, the Check Character for the file mark is written. Because the Check Character is written in even parity, it also appears as an octal 17 (Figure 10-18).

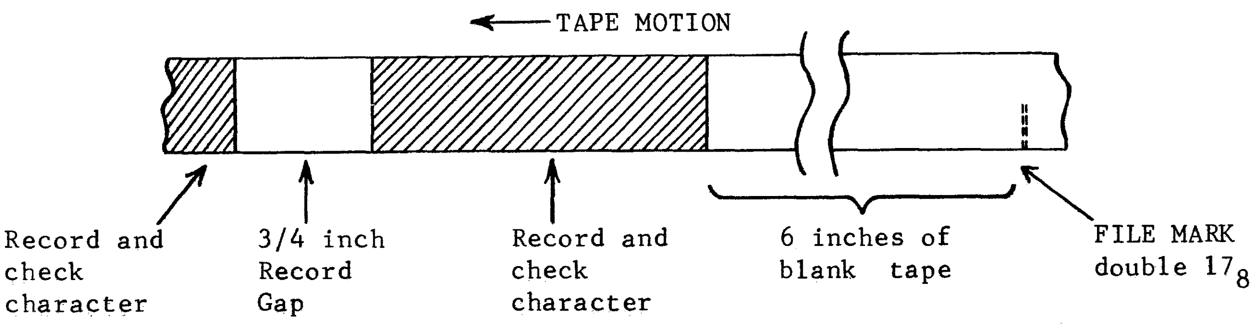


Figure 10-18. Writing a File Mark.

A comparison can be made between a reel of magnetic tape and an ordinary book, such as this text, containing chapters.

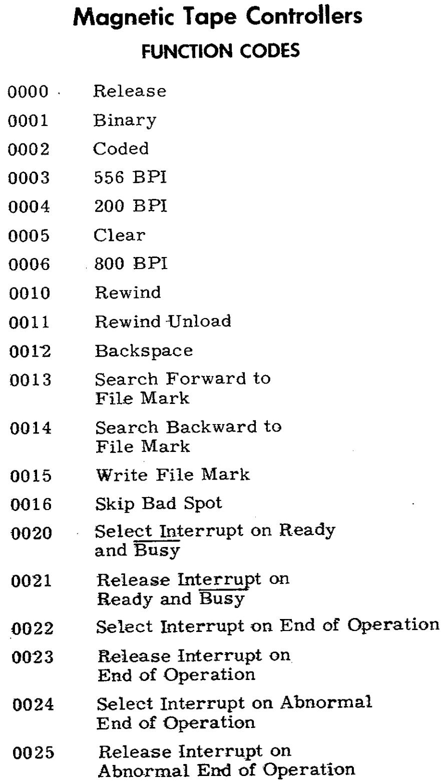
Each file is comparable to a chapter in the book and each record in the file may be compared to the pages in a chapter. If the desired information is contained in file #2, it can be referenced by executing a Select instruction to search forward to file mark. This causes the first file to be ignored and the subsequent initiation of an INPW instruction would read data from file #2.

In addition to the transfer of data, the controller communicates with the selected tape unit to control tape motion and to select desired operations (read, write, density, etc.). Table 10-1 is one page from the peripheral equipment code book and illustrates those codes that would be included in the lower 12 bits of a Select instruction to initiate a desired operation. For example, a 77 1 20010 instruction would cause the selected tape unit on channel #2 to rewind back to the Load Point Marker. With that as a reference point, any information on that reel of tape may be accessed.

The overall function of the tape controller is to synchronize the speed of the tape unit and the speed of the computer. It accepts control signals from the computer and generates a response, checks parity for each frame of information read from tape, and generates parity for each frame written on tape. It also generates and checks the Check Character for each record and generates the octal 17 to be written as a File Mark.

As data is transferred, the controller must accept a 12-bit byte from the I/O channel and disassemble it into frame-size bytes for the tape unit. During input, two frame-size bytes are assembled by the controller and sent to the I/O channel. When the 24-bit I/O channel has received two 12-bit tytes from the controller, a request is issued to the computer and the word is placed in storage at the address determined by the contents of the register file location for that channel (01-07).

TABLE 10-1. TAPE CONTROLLER SELECT CODES



Magnetic Tape Controllers

FUNCTION CODES

0000	Release
0001	Binary
0002	Coded
0003	556 BPI
0004	200 BPI
0005	Clear
0006	800 BPI
0010	Rewind
0011	Rewind Unload
0012	Backspace
0013	Search Forward to File Mark
0014	Search Backward to File Mark
0015	Write File Mark
0016	Skip Bad Spot
0020	Select Interrupt on Ready and $\overline{\text{Busy}}$
0021	Release Interrupt on Ready and $\overline{\text{Busy}}$
0022	Select Interrupt on End of Operation
0023	Release Interrupt on End of Operation
0024	Select Interrupt on Abnormal End of Operation
0025	Release Interrupt on Abnormal End of Operation

Magnetic tape is highly acceptable as a storage medium for the following reasons:

1. Extremely large storage capacity per reel of tape (nearly six million words could be stored on one 2400-foot reel).

2. Tape is relatively inexpensive.

3. Tape reels are easily changed on the tape unit.

4. Large volumes of data are easily transported.

There are some disadvantages of magnetic tape storage:

1. Data must be accessed sequentially (a record near the end of the tape must be accessed by moving tape starting at the Load Point Marker and counting File Marks. Then the desired record in the file must be read into storage.)

2. Tape speed must be limited to reduce head and tape surface wear, to maintain close velocity specifications, and to allow start and stop tape movement within acceptable time limits.

The special design features of Control Data tape units make magnetic tape storage reliable for the following reasons:

1. Vacuum Tape Drive -- Tape movement is controlled by vacuum and pressure in the drive capstans instead of the friction-type pinch rollers.

2. Less wear -- The recording surface of the tape comes in contact only with the nylon idler as it moves from supply to take-up reel (Figure 10-7).

3. Tape cleaning -- A specially-designed vacuum tape cleaner removes all foreign particles from the recording surface as tape passes under the head assembly. This prevents minute dust particles from scratching the recording surface which could result in lost data.

Because of the advantages of magnetic tape as a storage device, magnetic tape handling equipment is normally included as an integral part of a computer system.

CARD EQUIPMENT

Another common and highly accepted input-output medium is the punched card. Programs may be read into the computer via a card reader and after assembly or compilation, the machine language object program may be punched on a card punch. This provides a "hard" copy of the program and eliminates re-assembly or recompilation time if the program is to be re-used.

The standard card contains 12 horizontal rows and 80 vertical columns (Figure 10-19).

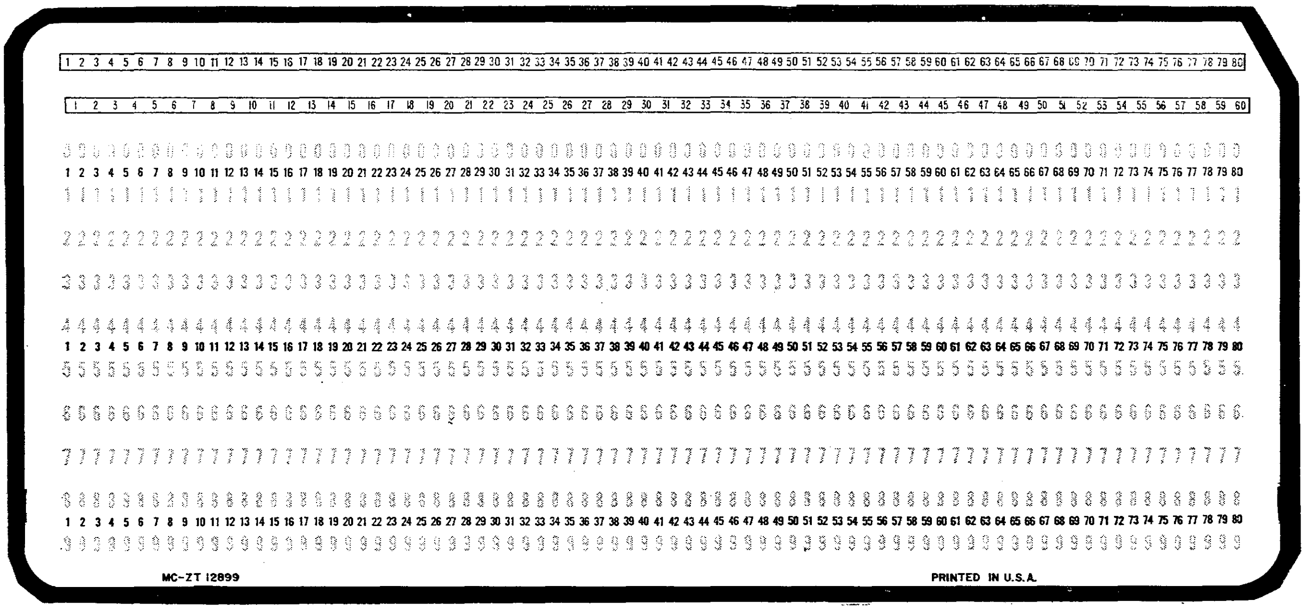


Figure 10-19. 80-Column Card.

Another commonly-used card is the 51-column card (Figure 10-20).

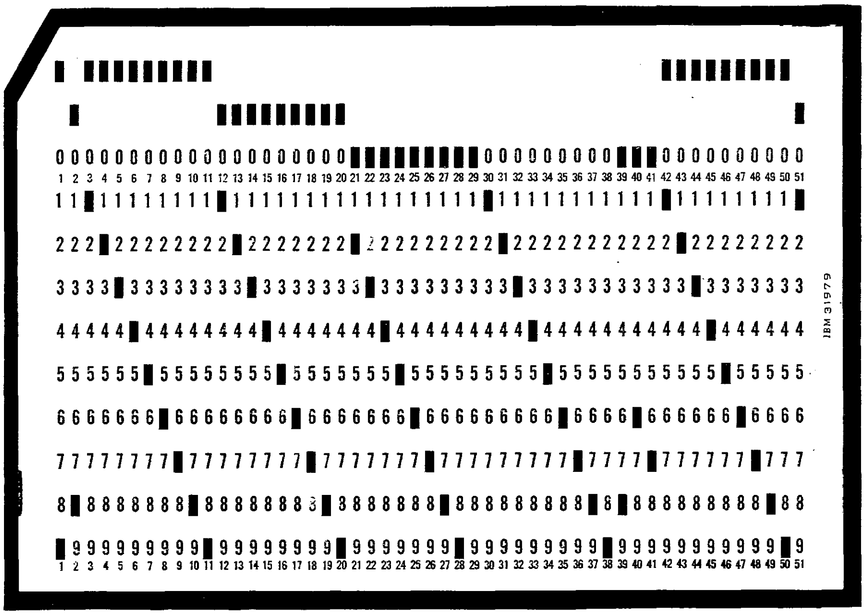


Figure 10-20. 51-Column Card.

Data may be punched into a card in any of the following codes:

1. Hollerith
2. Binary coded decimal (BC)
3. Row binary
4. Column binary

Hollerith Coding

When a program is initially written, the programmer gives the coding forms to a keypunch operator. The source program is then manually punched into a source deck which may then be read into the computer.

However, a review of Chapter IV will indicate that a source deck may contain alphanumeric characters that are not compatable to the binary number system of the computer. In 1889, Dr. Herman Hollerith invented a code that could be used to express alpha, numeric, and special characters by a combination of holes in a card. This code is still used today and is the basis for the Hollerith card.

In the Hollerith format, each vertical column contains one alphanumeric character (maximum of 80 character per card). This type of card is especially advantageous because the character punched into each column may be printed at the top. The source program can then be visually inspected for keypunch errors before it is fed to the computer. A Hollerith card with each special character is illustrated in Figure 10-21.

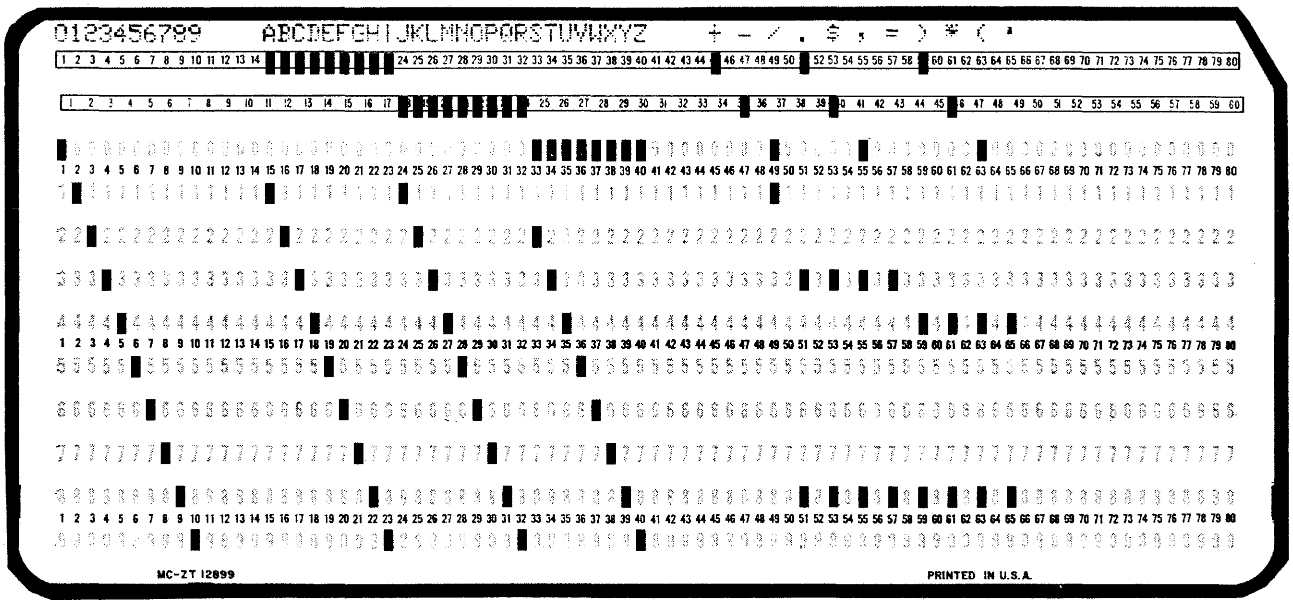


Figure 10-21. Hollerith Punched Card.

Each column could have holes punched in any of the 12 rows but notice that a maximum of three holes is sufficient to express any Hollerith character. Also, for future reference, notice that rows 7, 8, and 9 may be punched but no character requires a punch in both 7 and 8 or both 7 and 9.

All cards punched on a keypunch are in Hollerith format. Although only one character is contained in a column, the advantage of the print-out on the card justifies the wasted capacity.

Binary Coded Decimal (BCD)

In BCD format, each column contains two 6-bit BCD characters. The same characters expressed in Hollerith may also be expressed in BCD. By using the BCD format instead of Hollerith, the capacity of the card is doubled. In Hollerith, the character A would be represented by punches in rows 12 and 1; the BCD code for "A" is 61 and would result in punches in rows 12, 11, and 3 or rows 4, 5, and 9. A card punched in BCD is illustrated in Figure 10-22. The most significant bit of the first BCD character is in row 12. The card may be read by turning it over and reading in binary from left to right.

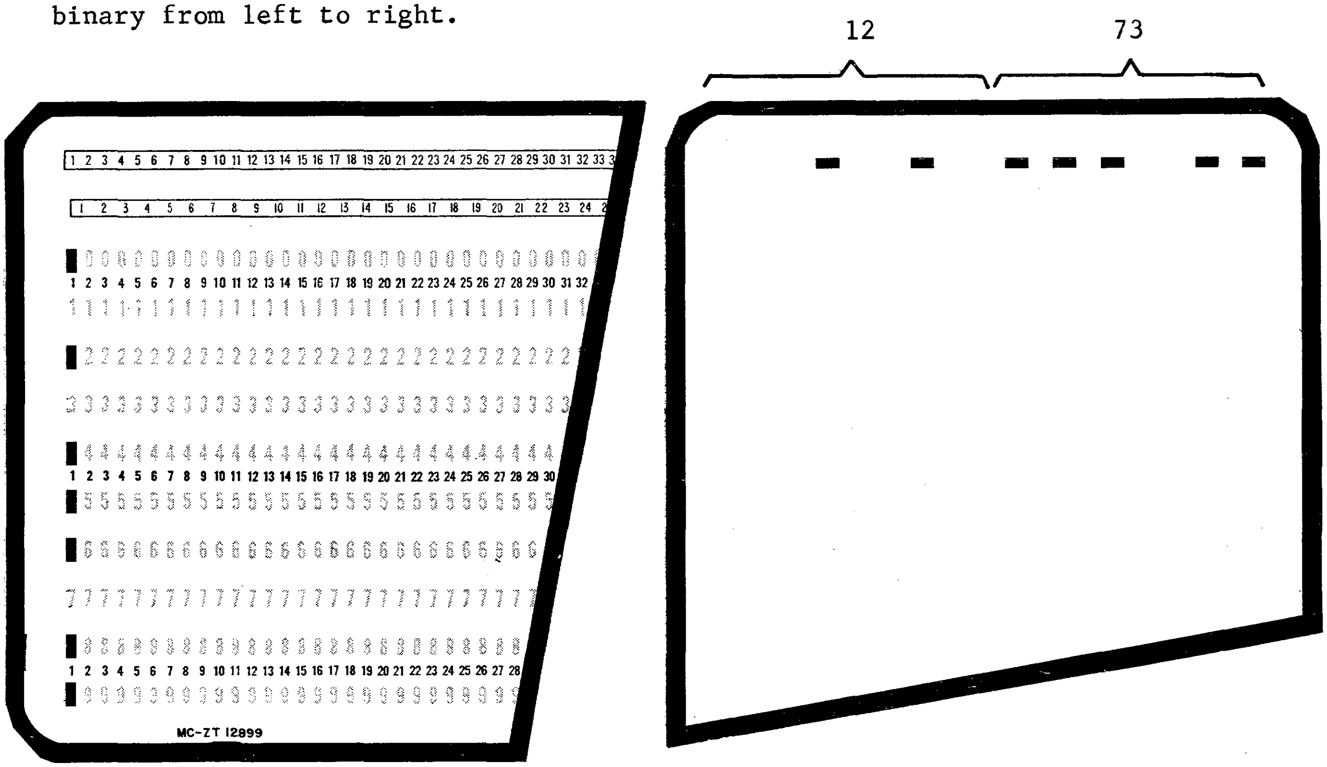


Figure 10-22. BCD Card.

Row Binary

A binary card contains only numbers such as are used internally in the computer. Once a program has been assembled or compiled into machine language instructions, that program may be punched out in binary.

At some later date, the binary object deck may be read back into the computer exactly as it appears on the card. Row binary indicates that the card is punched row by row (Figure 10-23) instead of by column (Figure 10-24).

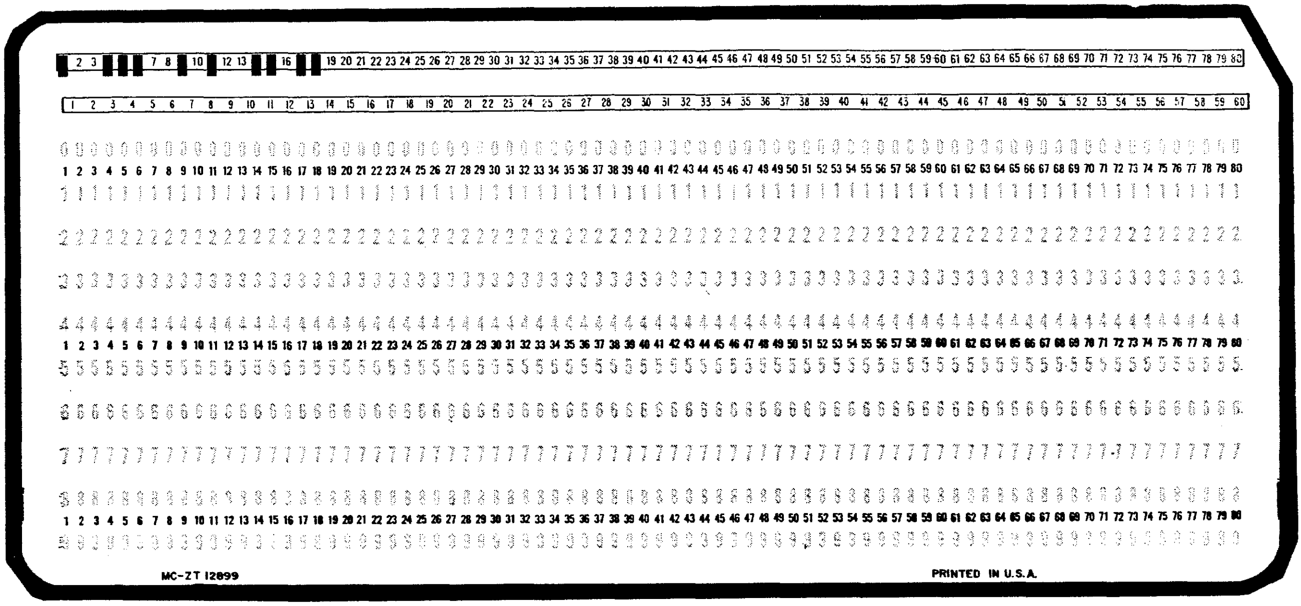


Figure 10-23. Row Binary

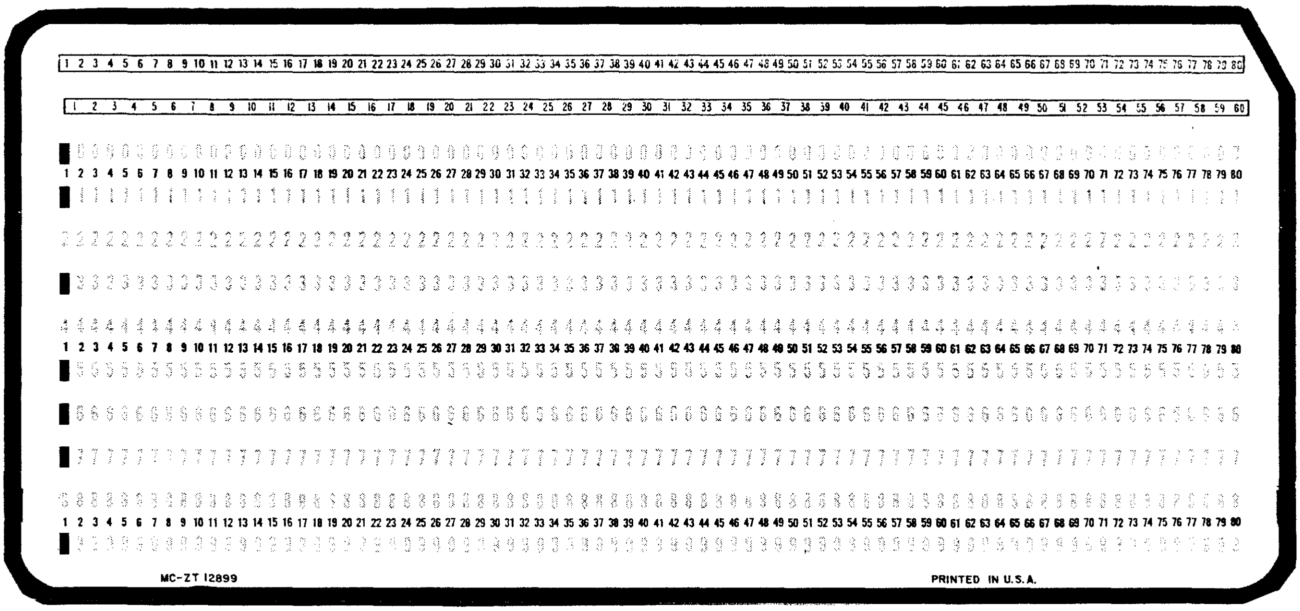


Figure 10-24. Column Binary.

Column Binary

The only difference between row binary and column binary is that the information is punched by column instead of by row. How the information is punched is determined by the type of equipment. Either type of card may be read into the computer but special manipulation of the data may be required. The column binary card may be read exactly the same as the BCD card. The column binary card is illustrated by Figure 10-24.

With four types of cards to be read, it looks as though some confusion should exist either in the card reader, the controller, or the computer.

However, the problem is not as complex as it may appear.

CARD READER

The function of a card reader is to move cards past a read station and extract the data by sensing holes punched in the card. Earlier models read the data with mechanical fingers that made electrical contact if a hole was punched. This method is satisfactory if the cards are not moved past the read station too rapidly. Increased demands for faster I/O equipments brought about the design of a different type of card reader. Instead of mechanical fingers, data is sensed photoelectrically. If a hole is punched in the card, the light is detected by a photocell. Using this method, cards may be moved past the read station at a very rapid rate. One such type is the CONTROL DATA® 405 Card Reader (Figure 10-25).

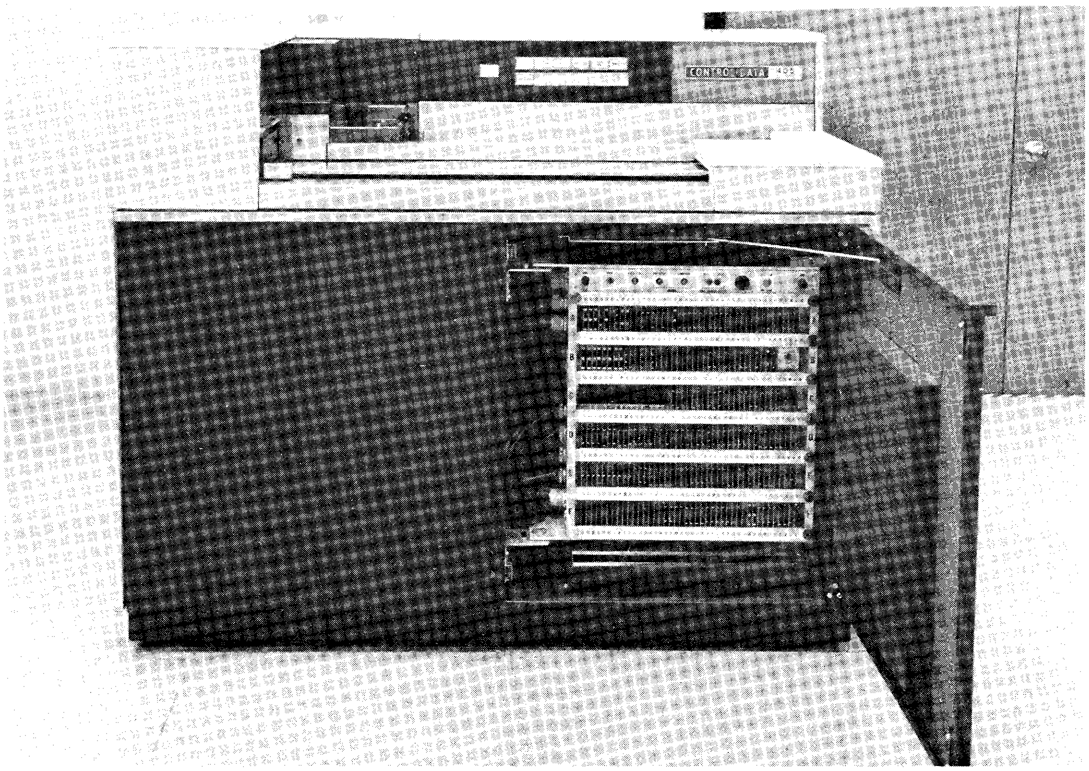


Figure 10-25. Card Reader.

As with all I/O equipments, the card reader must have a controller as an interface between the computer and the reader. The controller is contained in the bottom of the 405 Card Reader. Notice the Equipment Number Select Switch on the top row (Figure 10-25).

Cards are read at the rate of 1200 per minute with an input tray capacity of 4000 cards. Each card is read by two read stations and if the same information is not read by both stations, that card is routed to a secondary hopper and a read error is detected. Cards are picked from the input tray by a vacuum capstan and passed by the read station by pinch rollers. The light source is a high-intensity lamp focused on the two read stations through a prism (Figure 10-26).

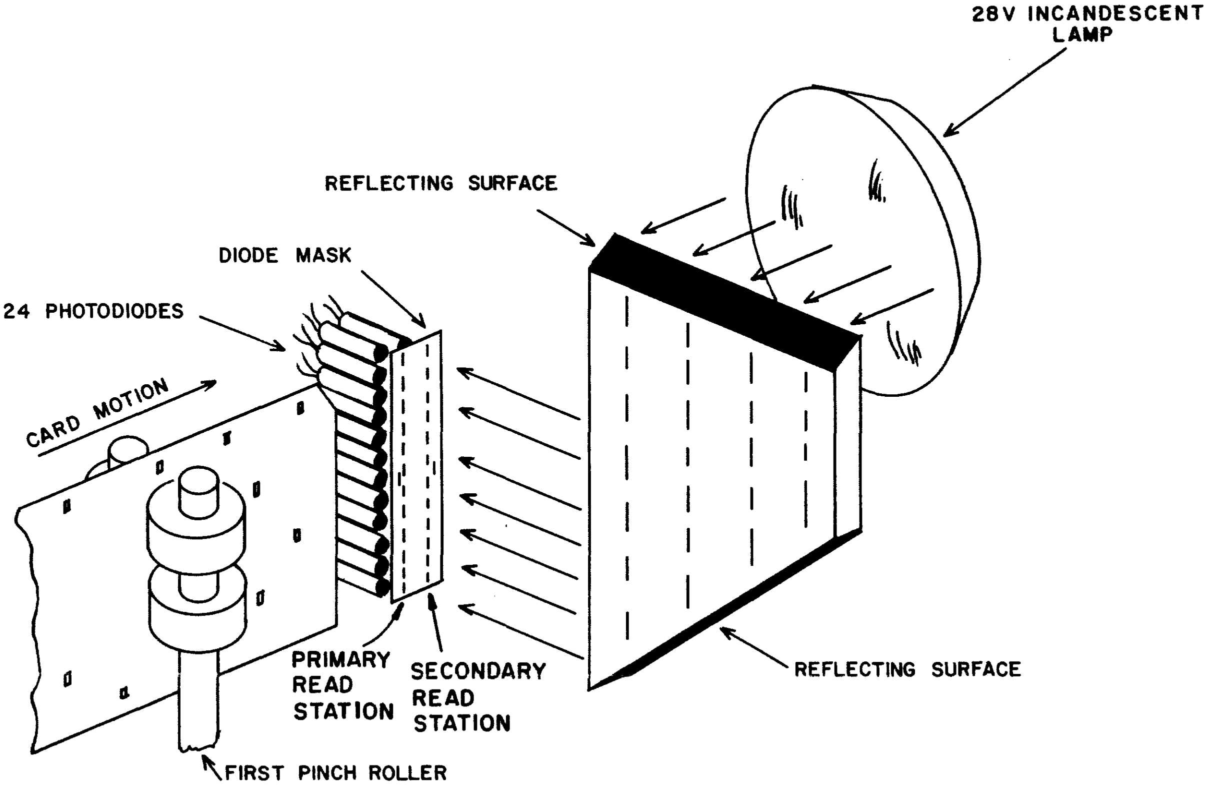


Figure 10-26. Light Source and Read Stations.

The card reader does not differentiate between Hollerith, BCD, and binary cards. The information is sent to the controller as it is read (by columns). Some controllers have a memory where all of the information from one card is stored. Others send the information directly to the I/O channel as it is read. In either type a conversion from Hollerith to internal BCD may be accomplished.

Internal BCD is a special binary coded decimal format used to facilitate the sorting of alphanumeric characters. For example, Table 10-1 illustrates that internal codes for the alpha characters are in ascending order (from 21 for an A to 71 for a Z). The commonly used (by the computer industry) external BCD codes are not as easily sorted because the magnitude of the code does not allude to alphabetical positioning. Further discussions relating to BCD characters will, therefore, assume them to be in internal BCD format.

TABLE 10-2. INTERNAL-EXTERNAL BCD CODES

Char Prd.	Int. BCD	Ext. BCD
A	21	61
B	22	62
C	23	63
D	24	64
E	25	65
F	26	66
G	27	67
H	30	70
I	31	71
J	41	41
K	42	42
L	43	43
M	44	44
N	45	45
O	46	46
P	47	47
Q	50	50
R	51	51
S	62	22
T	63	23
U	64	24
V	65	25
W	66	26
X	67	27
Y	70	30
Z	71	31

Cards read through the card reader are normally assumed to be punched in Hollerith. Each column of information from the card will be translated into a 6-bit BCD character by the controller.

Assume that an assembly language program is being read by the card reader and that a card contains ANSWER LDA SHEEP.

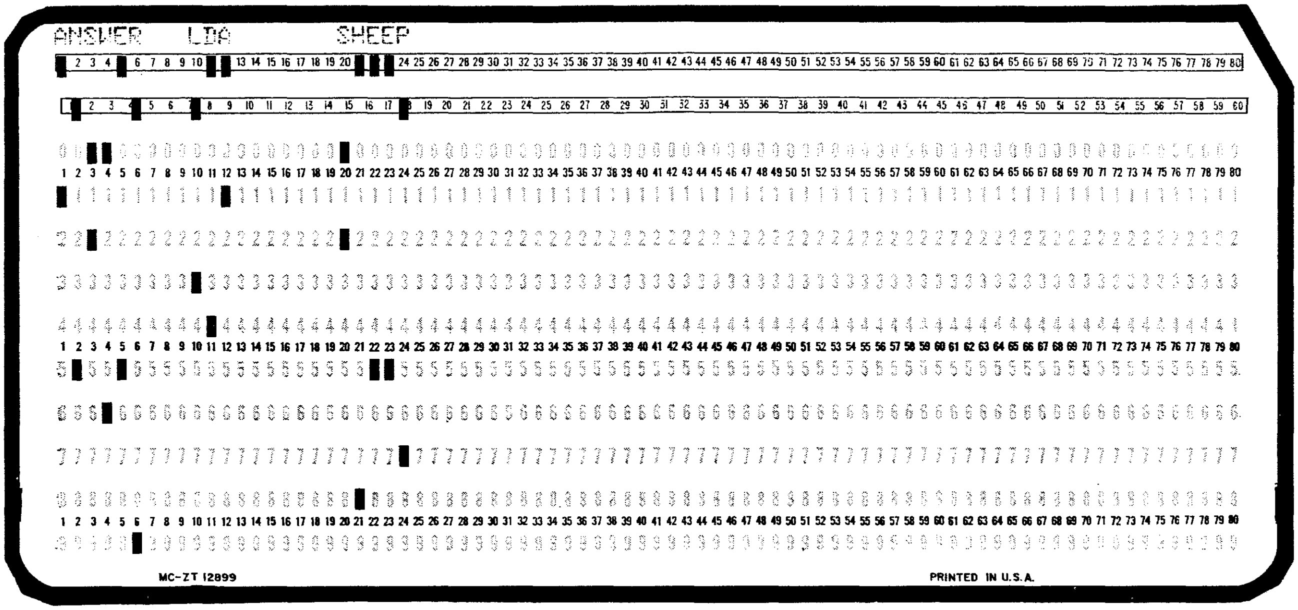


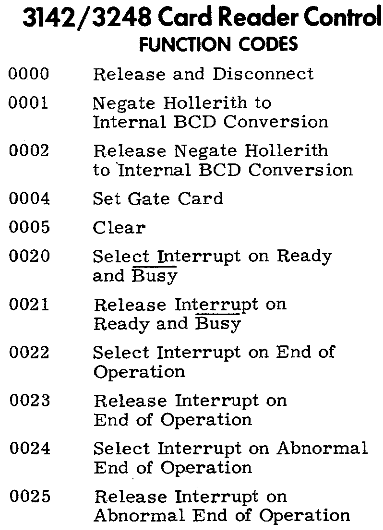
Figure 10-27. Hollerith Card from Source Program.

The A in column 1 is translated as an octal 21 by the controller and the N as an octal 45. If the controller does not have its own memory, that 12-bit byte is immediately sent to the I/O channel. As soon as the channel receives the next 12-bit byte (SW = 62 and 66), it issues a request to the computer and the 24-bit word is placed in storage as 21456266_8. The entire card is read into storage in 20 sequential storage locations (blanks are translated as an octal 60). The second storage location would contain the codes for ER blank blank, or 25 51 60 60. When the entire program has been read, the internal BCD codes are assembled into the machine language program.

If a deck of cards is to be read that are not coded in Hollerith, the Hollerith-to-BCD conversion may be suppressed by executing a Select instruction (Table 10-3).

$$\underbrace{77\ 1}_{\text{SEL}}\quad \underset{\text{CH\#0}}{\overset{0}{\downarrow}}\quad \underbrace{0001}_{\text{NEGATE HOLLERITH TO BCD CONVERSION}}$$

TABLE 10-3. CONTROLLER FUNCTION CODES



3142/3248 Card Reader Controller

FUNCTION CODES

0000	Release and Disconnect
0001	Negate Hollerith to Internal BCD Conversion
0002	Release Negate Hollerith to Internal BCD Conversion
0004	Set Gate Card
0005	Clear
0020	Select Interrupt on Ready and $\overline{\text{Busy}}$
0021	Release Interrupt on Ready and $\overline{\text{Busy}}$
0022	Select Interrupt on End of Operation
0023	Release Interrupt on End of Operation
0024	Select Interrupt on Abnormal End of Operation
0025	Release Interrupt on Abnormal End of Operation

The data on the card is then read exactly as it appears and the information from each column is sent to the I/O channel as a 12-bit byte. Twice as much information was contained on the card and, consequently, 40 storage locations will be required to store the data. If the deck was punched in column binary format, it could represent a program in machine language and no assembling would be required.

If the deck was in external BCD format, the controller would automatically translate it to internal BCD. When the program is in memory, the assembler must still assemble the internal BCD codes into machine language instructions.

The negation of the Hollerith-to-BCD conversion may also be accomplished by having rows 7 and 9 punched in column 1. A special signal to the controller causes the entire card to be read in binary because no conversion is accomplished. A quick look back to chapter 4 (Figure 4-33) will show that the SCOPE control cards all have a 7/9 multipunch in column 1. SCOPE is a monitor and consequently does not assemble alphanumeric characters. For this reason, control cards are read as binary cards even though an alphanumeric printout may appear at the top of the card (Figure 10-28).

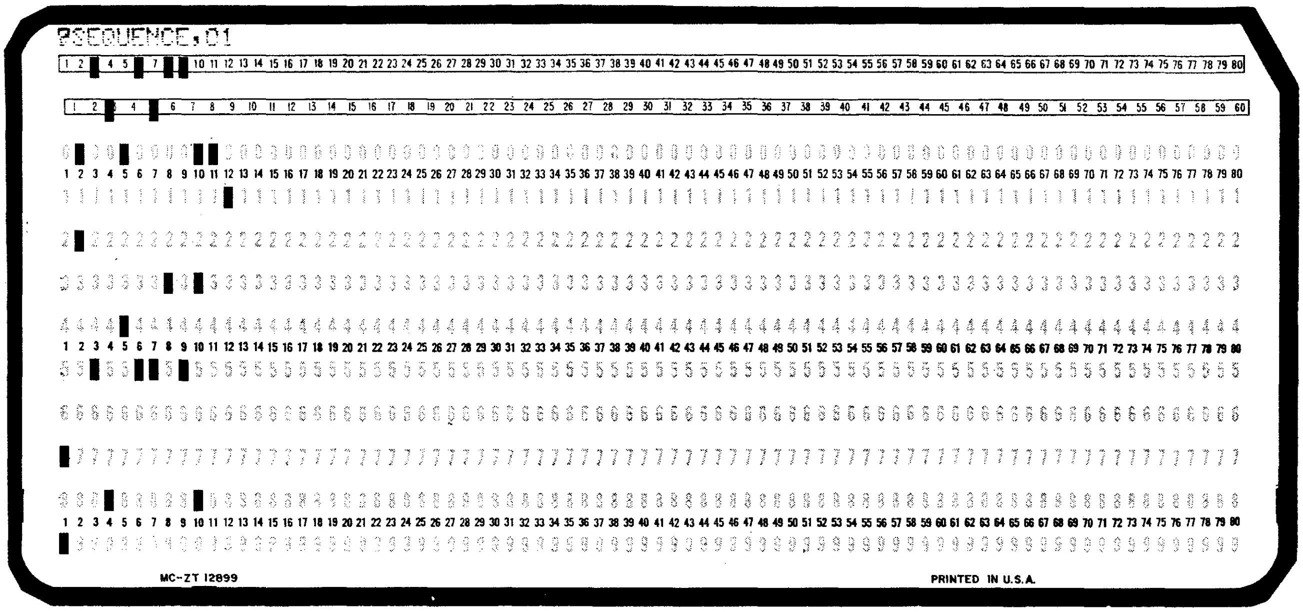


Figure 10-28. Binary Control Card (7/9 Multipunch).

Another special card is the 7/8 7/8 multipunched card. This is interpreted as an End Of File command to the computer. The resulting interrupt terminates the read operation and allows the computer to continue with a different routine.

The advantages of the punched card are:

1. Program changes are easily made.

2. Cards are relatively inexpensive.

3. The "hard copy" of the program may be permanently retained.

The only significant disadvantages are:

1. Cards wear out or may be damaged.

2. Storage space requirements are considerably greater than with other storage mediums. For example, the information that can be stored on a single reel of magnetic tape would require a stock of Hollerith cards over 180 feet high.

CARD PUNCH

The card punch is an output device used to produce an object deck or to reproduce any information that may be stored in memory. One type of card punch and its controller is illustrated in Figure 10-29.

This particular punch has an output rate of 250 cards per minute. The punch is considerably slower than the card reader because the cards are punched with mechanical knives. After the card is punched, it is immediately read to ensure that the correct information is contained on the card. The reader is of the mechanical type, allowable because of the slow rate at which the cards are punched. When a read error is detected, the card in question is laterally offset in the output hopper. It may then be removed from the deck and visually examined for errors.

Some controllers have a 40-word memory which holds the information to be punched on one card. The data is received from computer storage and repreduced in the special memory. Conversion to the desired format is then accomplished by the controller.

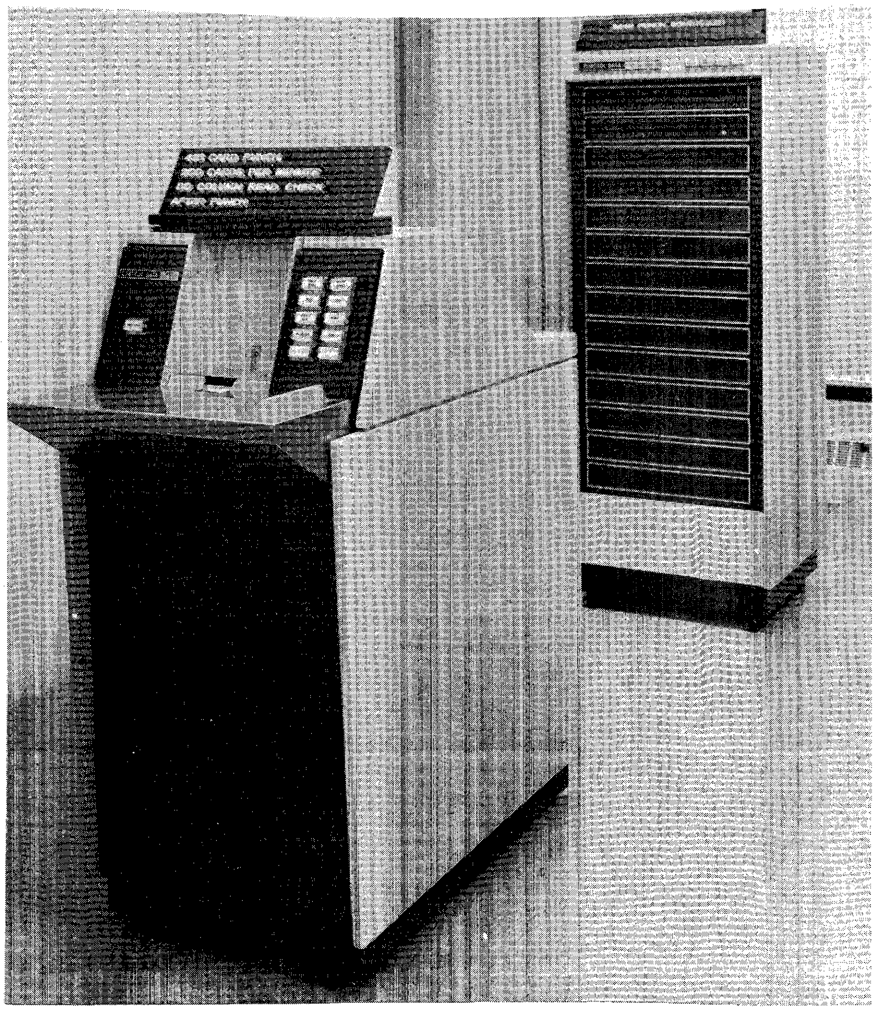


Figure 10-29. Card Punch and Controller.

The card is actually punched by row instead of by column. Using this method, a card may be punched in less time because only 12 mechanical operations are involved instead of 80. It appears as though the card would then be punched in row binary but rearrangement of the data before being sent to the punch results in a column arrangement.

The controller illustrated in Figure 10-29 does not have a separate memory and, therefore, all conversions and rearrangement of data must be accomplished in the computer by special routines. Data is sent to the controller in eight, 10-bit bytes where it is assembled in an 80-bit register. That information is then used ot punch row 9 on the card. The operation is performed 12 times to punch the 12 rows on each card.

LINE PRINTER

Another necessary I/O equipment in a computer system is a line printer. A printer provides the user with a readable copy of program listing and results. Special forms, such as payroll checks, may also be used. The advantage of the line printer is that all characters in an entire line are printed simultaneously. Using this method, printing rates of 1000 lines per minute are obtained. One type of line printer and its controller is illustrated in Figure 10-30. The cabinet on the right contains the printer; the controller is in the background. The printer paper can be seen through the glass of the printer cabinet. The projection at the back of the cabinet is the stacker which receives and fan-folds the printed output.

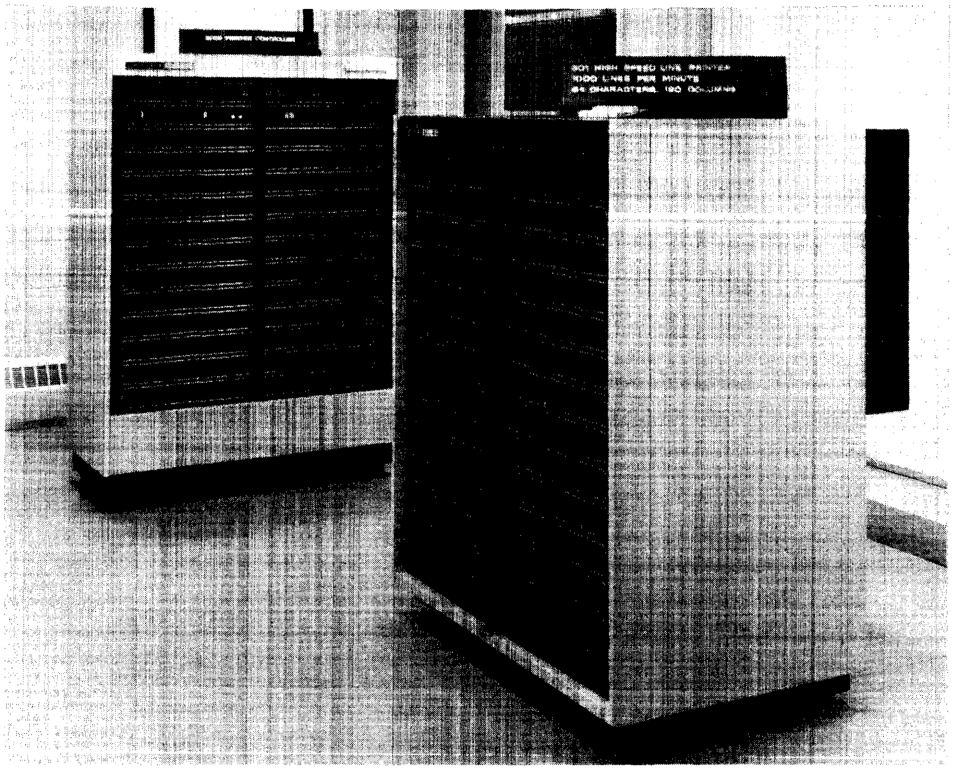


Figure 10-30. Line Printer and Controller.

The controller in Figure 10-30 allows the printer to be accessed by either of two channels. Notice the two equipment number switches inside the controller (top left). The printer has an output rate of up to 1000 lines per minute, 120 characters per line. Other print head assemblies may be used with up to 140 characters per line.

The print head has 64 rows of characters around its circumference, each row containing identical characters. The standard Fortran characters are contained in 48 rows with special characters in the remaining 16. Table 10-4 lists the BCD codes and the corresponding character for each.

TABLE 10-4. CHARACTERS AND CODES

External BCD	Internal BCD	Character Printed	External BCD	Internal BCD	Character Printed
00	12	: Colon	40		- minus
01		1	41		J
02		2	42		K
03		3	43		L
04		4	44		M
05		5	45		N
06		6	46		O
07		7	47		P
10		8	50		Q
11		9	51		R
12	00	0	52		∨ logical OR
13		= equals	53		$ dollar sign
14		≠ not equals	54		* asterisk
15		≤ less than or equal	55		↑ arrow up
16		% percent	56		↓ arrow down
17		[open bracket	57		> greater than
20	60	Blank (not loaded into memory)	60	20	+ plus
21	61	/ slash	61	21	A
22	62	S	62	22	B
23	63	T	63	23	C
24	64	U	64	24	D
25	65	V	65	25	E
26	66	W	66	26	F
27	67	X	67	27	G
30	70	Y	70	30	H
31	71	Z	71	31	I
32	72	] closed bracket	72	32	< less than
33	73	, comma	73	33	. period
34	74	(open parenthesis	74	34	) closed parenthesis
35	75	→ arrow right	75	35	≥ greater than or equal
36	76	≡ Identify	76	36	¬ logical NOT
37	77	∧ logical AND	77	37	; semicolon

The drum is continuously rotating at 1000 RPM and all characters on a line would be printed during one revolution. A character is printed by actuating a hammer which pushes the paper against a ribbon and the head (Figure 10-31).

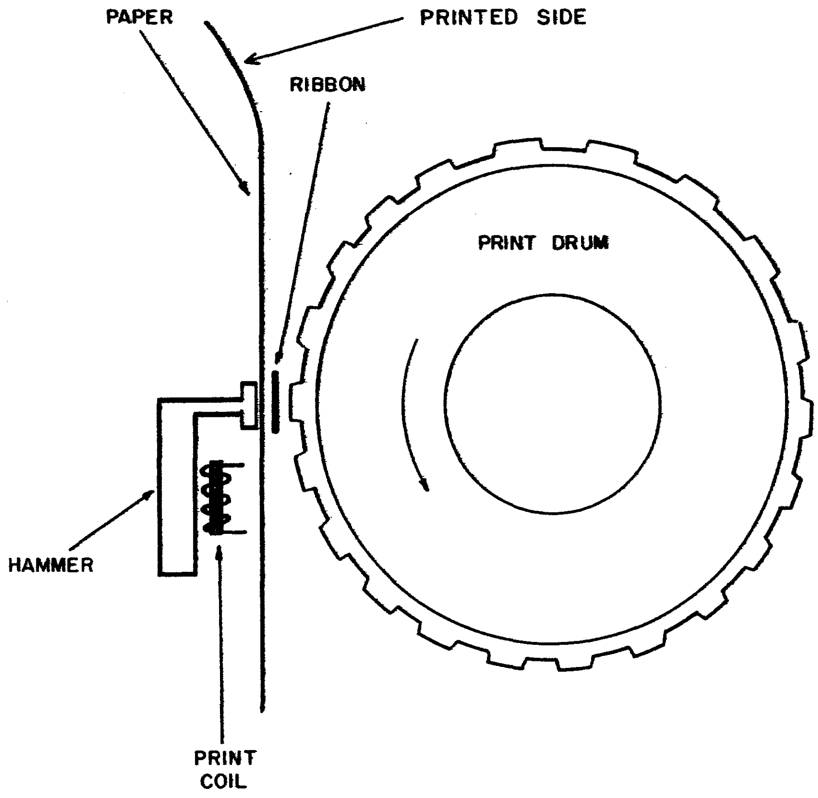


Figure 10-31. Print Head Assembly.

If the letter A is to be printed across the entire line, all hammers would be actuated at the same instant. Although the drum does not stop, smudging is eliminated by the rapid speed of the hammers. After a line has been printed, the paper is advanced and stopped before printing of the next line begins.

Line spacing on the printed page is achieved by selecting one of eight levels on a format tape in the printer. The format tape is a continuous mylar loop containing the same number of frames as lines on the printed page. A standard page could have 66 lines of print which would require a 66-frame format tape.

In addition to the eight levels, the format tape contains a row of feedholes. As paper advances, a cogged wheel engages the feedholes and advances the format tape. Level eight may contain only one hole, as it determines where the first line of print will appear on the page. Level 7 also contains one hole, which identifies the last line to be printed. The format tape may be punched to suit any desired application.

After the printer is connected to the I/O channel with a Connect instruction, the Select instruction, 77120015, would select level 5 of the format tape for post-print spacing (Table 10-5).

Figure 10-32 illustrates how the printed page would appear with level 5 of the format tape selected. The last line of form will not be printed because level 5 is not punched in frame 62. When the level 7 hole is sensed, paper will be advanced to the next level 8 hole (top of next form).

TABLE 10-5. PRINTER CONTROLLER FUNCTION CODES

FUNCTION CODES

0040	Release and Disconnect
0001	Single Space
0002	Double Space
0003	Advance to Last Line
0004	Page Eject
0005	Auto Page Eject
0006	Suppress Space
0010	Clear Format Selection
	Select Format Tape Level for Postprint Spacing:
0011	Level 1
0012	Level 2
0013	Level 3
0014	Level 4
→ 0015	Level 5
0016	Level 6
0020	Select Preprint Spacing
	Select Format Tape Level for Preprint Spacing:
0021	Level 1
0022	Level 2
0023	Level 3
0024	Level 4
0025	Level 5
0026	Level 6
0030	Select Interrupt on Ready and $\overline{\text{Busy}}$
0031	Release Interrupt on Ready and $\overline{\text{Busy}}$
0032	Select Interrupt on End of Operation
0033	Release Interrupt on End of Operation
0034	Select Interrupt on Abnormal End of Operation
0035	Release Interrupt on Abnormal End of Operation

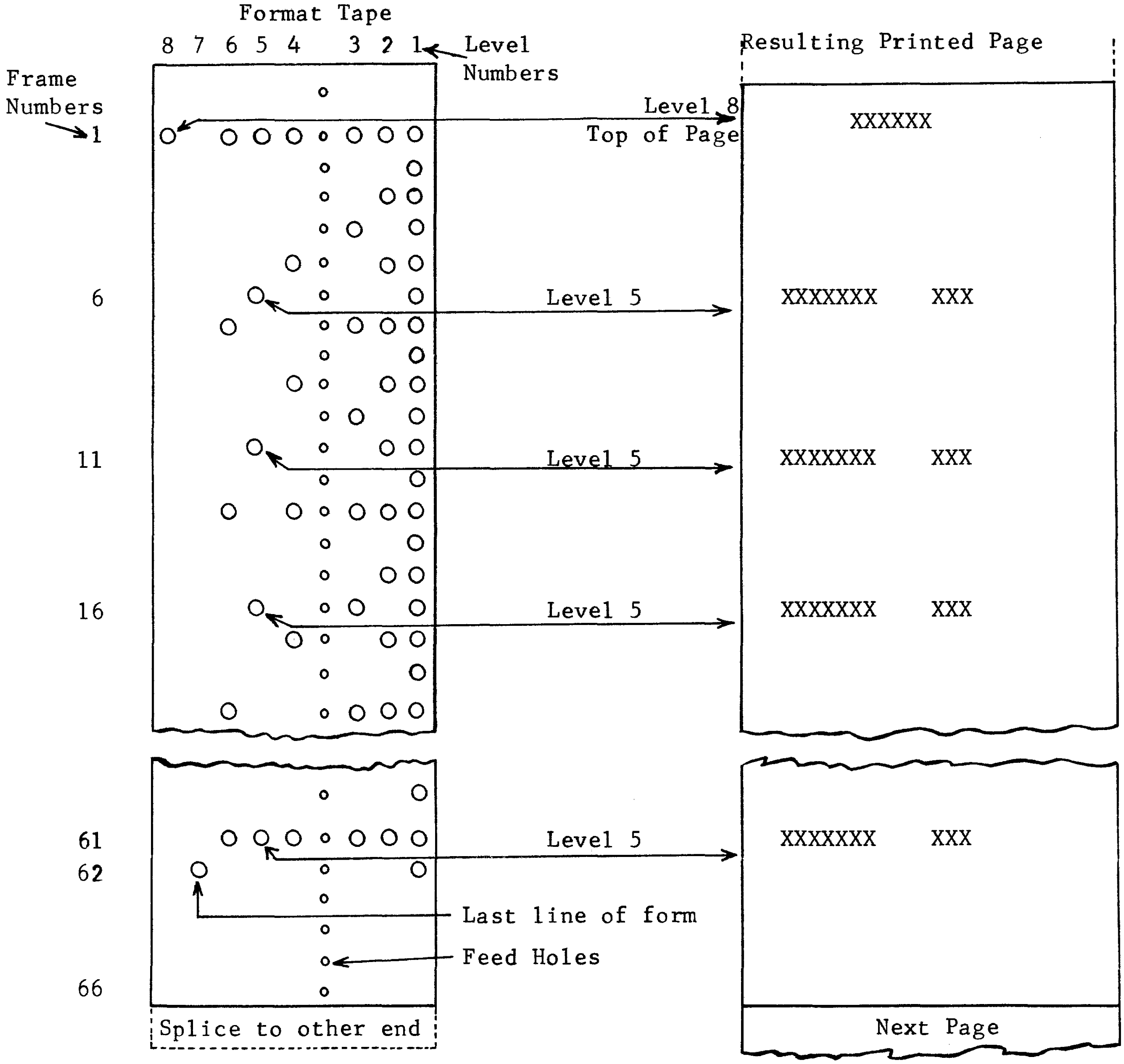


Figure 10-32. Format Tape And Page Layout

The format tape only determines which lines of the page will be printed; it does not determine what will be printed on these lines.

Data contained in 30 storage locations would print one line of 120 characters. If a line is to contain blanks, the internal BCD code for a blank (60) must be in memory at those character positions. For example, suppose you wish to print out a poem entitled THE MORTIFIED MORTICIAN. The title is to start 5 spaces from the left margin of the page. The spaces and the title would require 28 character positions, or 7 storage words. If the data starts at address 01000, the OUTW instruction would contain 01000 as the first word address and 01007 as the last word address +1. The contents of those storage locations would be:

01000	60	60	60	60	blank	blank	blank	blank
01001	60	63	30	25	blank	T	H	E
01002	60	44	46	51	blank	M	0	R
01003	63	31	26	31	T	I	F	I
01004	25	24	60	44	E	D	blank	M
01005	46	51	63	31	0	R	T	I
01006	23	31	21	45	C	I	A	N

Another output could be initiated to print the first line of the poem. The printer remains connected until some other equipment on that channel is connected or a Select instruction to "release and disconnect" (771 X 0040) is executed (Table 10-5). When the second output is initiated, the printer automatically advances paper to the selected line.

The entire poem may be printed with one output buffer operation. However, the remaining 23 storage locations for the title must be filled with blanks (60), as must the unused character positions for all other lines in the poem.

When an output to the printer is initiated, the 30 words to print a complete line are sent to the controller in 12-bit bytes. Each 6-bit portion is translated and loaded into a special memory containing a single core for each character on the drum (Figure 10-33).

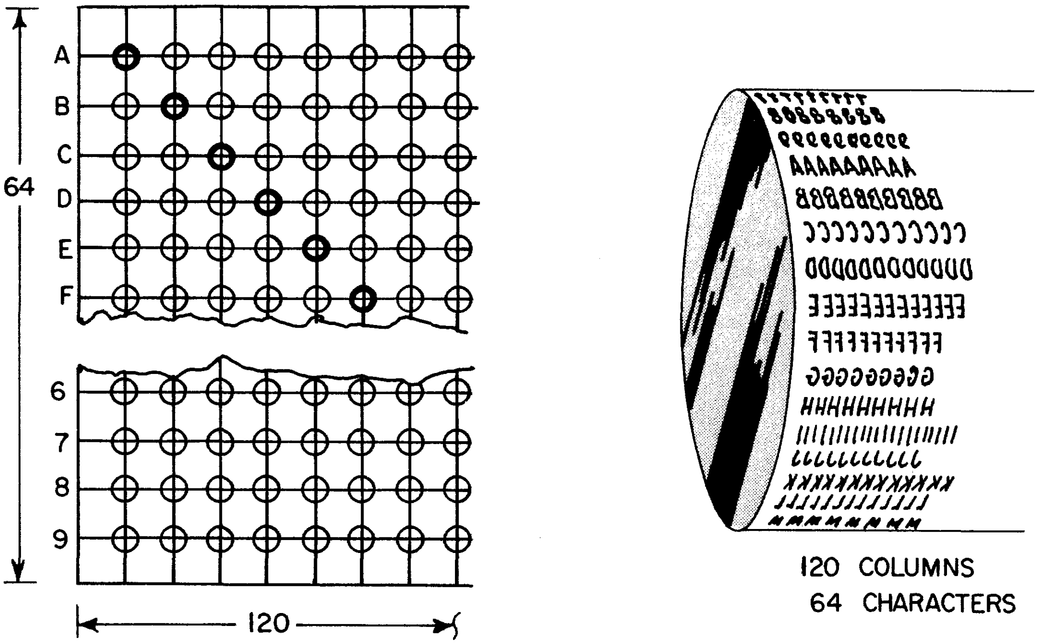


Figure 10-33. Controller Memory Configuration.

Assume that the alphabet is to be printed on one line. The upper 12 bits of the first word would be sent to the controller as 2122. The first 6-bit portion would be translated and would set the column 1 "A" core. The other byte would set the column 2 "B" core. The channel then sends another 12 bits which, when translated, would set the column 3 "C" and the column 4 "D" cores.

As each character is written into memory, a column counter is advanced which selects the proper vertical string of cores (Figure 10-34).

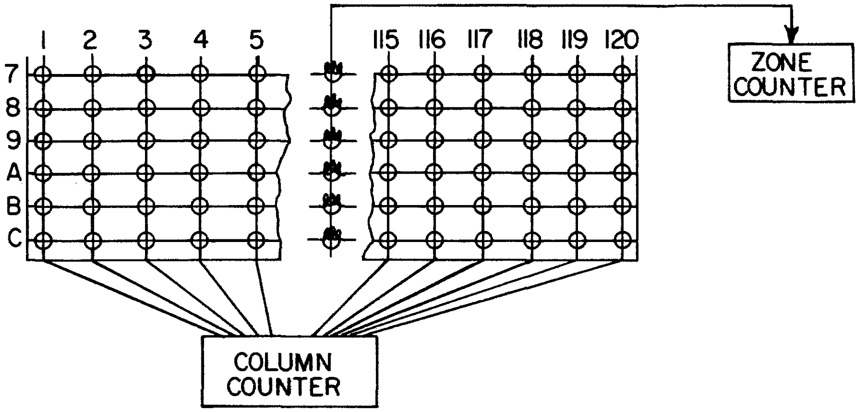


Figure 10-34. Column Selection.

The translation of the code selects the proper horizontal line and the core at the intersection of the two lines is switched to a one.

Another column of cores called a zone column has only the horizontal lines passing through them but the line is looped through the core three times. Each different character to be printed switches one core in the zone column. A zone counter is advanced as each core switches, which will indicate how many different characters are to be printed.

When the column counter reaches a count of 120, or the output buffer is terminated, that line is ready to be printed. Assume that the entire line was to contain nothing but the letter A. The "A" core in each column would be switched to a one and the zone counter would be at a count of one (only one character). When the print drum came around to the A line, the A string of cores would be read. As each core switches back to zero, the sense amplifier outputs energize the hammers and the entire line of A's is printed. The zone counter is decremented to a zero, which signifies that the line has been printed. Paper is advanced as selected, and the memory is again loaded with the next 120 characters to be printed.

If the next line is to contain the alphabet (Figure 10-33), the zone counter would be advanced to 26. As the alphabet is printed, the zone counter is decremented for each type of character. After all 26 letters have been printed, the zone counter reaches zero which, again, advances paper and reloads memory.

The advantage of this method is that the drum need not make a complete revolution before reloading memory. As soon as memory is loaded and paper has been advanced, the next row of characters to reach the hammers is printed. For example, if the line to be printed contains the letter T and the T row on the drum is the next to reach the hammers, all T's will be printed at the same time.

The morTified morTician

When the drum rotates around to the A row, all A's will be printed.

The morTified morTiciAn

Then the C row

The morTified morTiCiAn

Then the D row, etc.

If only the 48 standard FORTRAN characters are used, memory can be reloaded and paper advanced while the drum is rotating past the 16 special characters. One complete line would then be printed for each drum revolution and the output rate would be 1000 lines per minute.

The high-speed line printer is a necessary equipment in any versatile computer system.

PAPER TAPE DEVICES

Another medium for input/output communications is via paper tape. Information is recorded by the presence or absence of holes. A section of tape is illustrated in Figure 10-35 that has been punched in assembly mode. The hole at level 6 is not information but only indicates the starting frame of a 24-bit word. Reading in binary from left to right, the first word would be 73600514.

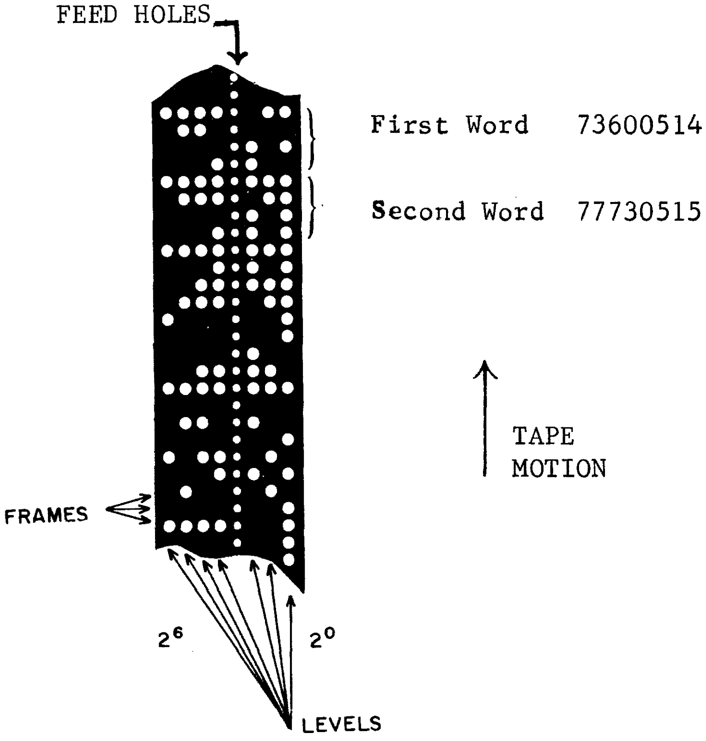


Figure 10-35. Paper Tape Format.

Assembly mode indicates that four frames are to be assembled into one word and read into computer memory. Data may also be recorded in Character mode. Character mode tape may be identified by the absence of holes in the 7th level (2^6).

To prepare punched tape, blank tape is moved past a punch station containing the necessary punch and die assembly required to punch a one-word frame.

The tape is momentarily halted. The word to be punched is sent from the computer which conditions the necessary punch mechanism to make the holes.

The tape is then advanced one frame width and the process is repeated. This method of recording data is relatively slow but permanent records of short programs and routines are easily obtained.

Once the tape has been prepared, it may be read back into the computer through the paper tape reader. The tape is read optically be detecting light where holes exist in the frame. The feedholes indicate when a frame is positioned over the photodiodes. After a frame is read, the tape is advanced to position the next frame over the read station. Input rate is about 250 frames a second as compared to a maximum of 30,000 frames a second with magnetic tape. As with other I/O devices, the reader and punch must have a controller to interface with the computer.

If paper tape equipment is desired as part of the computer system, the reader, punch, and controller may be obtained in a single cabinet. One type of paper tape station is illustrated in Figure 10-36. The reader is on the left and the punch is on the right.

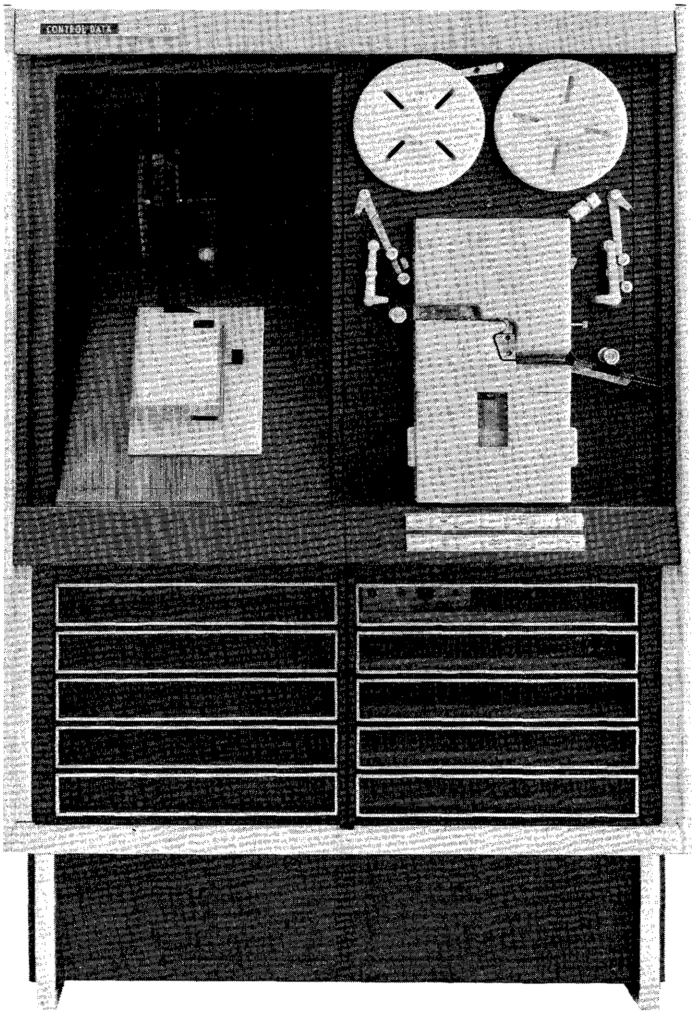


Figure 10-36. Paper Tape Reader/Punch.

PLOTTERS

The plotter is a high-speed, two-axis, digital recorder used for plotting one variable against another. Recording is done by the movement of a pen over the surface of the recording paper. The Y axis is produced by the lateral movement of the pen carriage and the X axis is produced by the rotation of the drum chart. The X and Y axes may be reversed, depending upon the information to be plotted. Z axis modulation incorporates the use of a pen solenoid which raises or lowers the pen from the recording surface in response to electrical input signals.

The motors used in plotters are of a bidirectional type on both the X and Y axes. Each input signal causes the pen carriage or the drum to move a predetermined distance in either a negative or positive direction. The rate of speed is determined by the type of motors used.

The recording paper used for plotting may be as small as the average sized notebook paper or may be in rolls of 100 feet or more. Movement of the paper is accomplished by sprocket teeth engaging sprocket holes at the paper's edge. The feed and takeup mechanism is bidirectional.

There are six operating modes in a typical plotter: Drum Up, Drum Down, Carriage Left, Carriage Right, Pen Up and Pen Down. The modes are controlled by electrical input pulses.

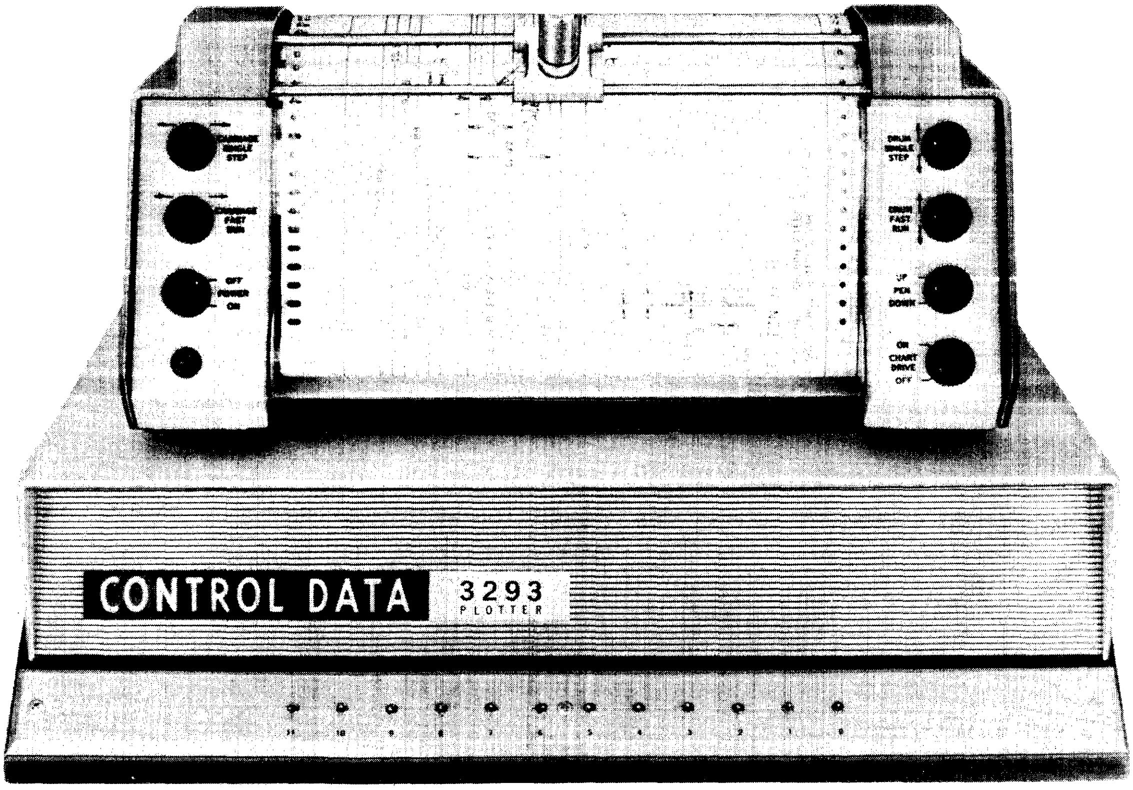


Figure 10-37. Incremental Plotter.

By the use of special adapters, plotters may be connected to a number of different types of general-purpose computers. One type of plotter is illustrated in Figure 10-37.

DISK RECORDING

The effective memory size of a computer may be increased by the addition of disk storage drives to the system. The disk equipment may be a large-capacity disk file or a smaller capacity disk pack.

The disks are constructed of closely-machined aluminum with a magnetic oxide coating. Data is recorded by positioning an arm containing the head assembly over a selected track and magnetizing minute areas of the surface in circular tracks around the disk. Multiple recording surfaces increase the storage capacity but data is read from only one track of one surface at a time. One type of disk pack is illustrated in Figure 10-38.

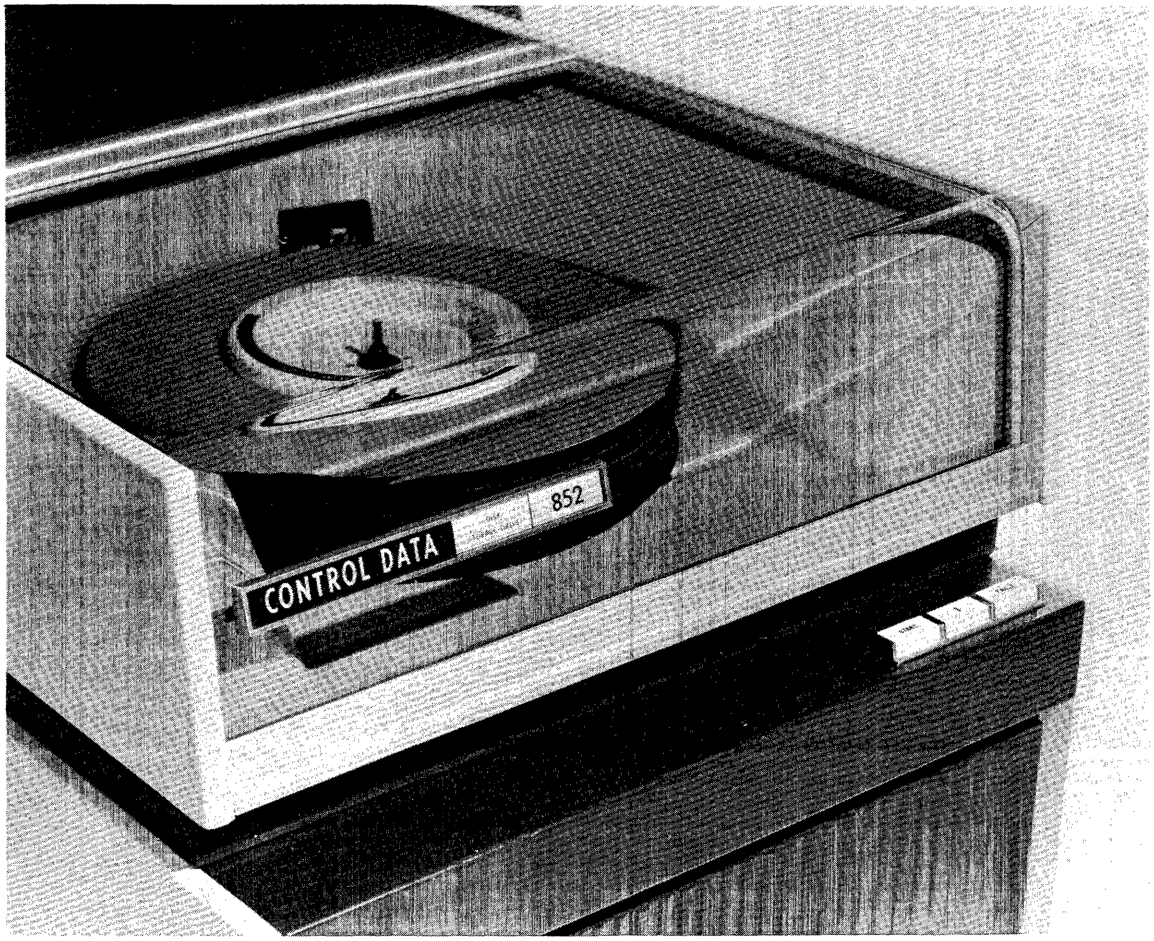


Figure 10-38. Disk Storage Drive.

The smaller disk pack has one advantage not available with the disk files. The six disks may be interchanged as a unit similar to reels of magnetic tape. With this feature, data may be recorded and then placed in storage. The unit in Figure 10-38 has a capacity of nearly three million BCD characters. Recording density averages slightly more than 800 bits per inch. Each of the 10 recording surfaces contains 100 tracks for data and revolves at 1500 revolutions per minute. The data transfer rate is over 77,000 characters per second (20,000 computer words) which compares favorably with magnetic tape. One advantage over magnetic tape is that data may be accessed in less than 0.2 seconds.

A computer system can have a greatly increased storage capacity with the addition of magnetic disk equipment.

OPTICAL PAGE READER

One of the recent developments in the I/O equipment line is the optical page reader (Figure 10-39).

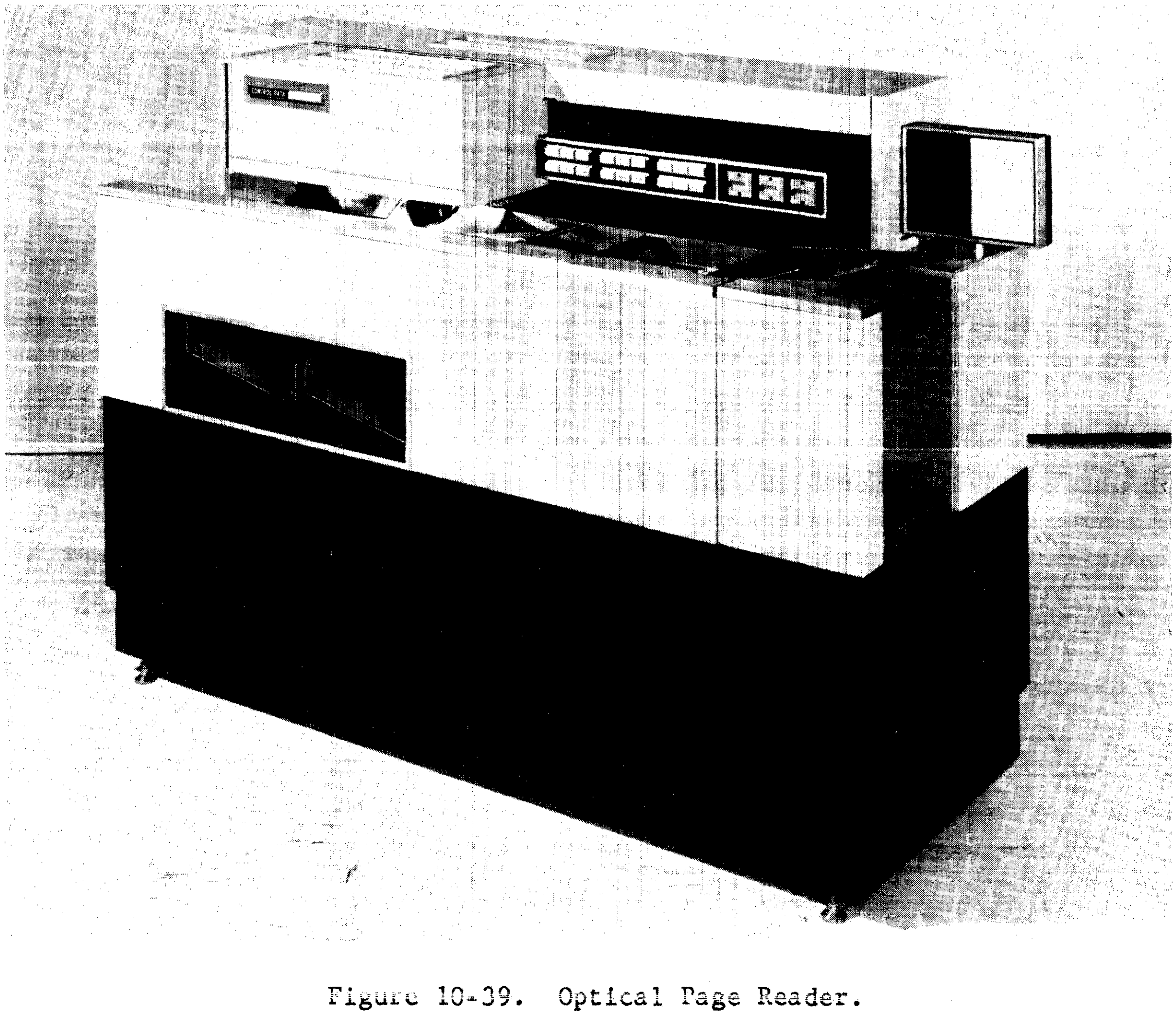
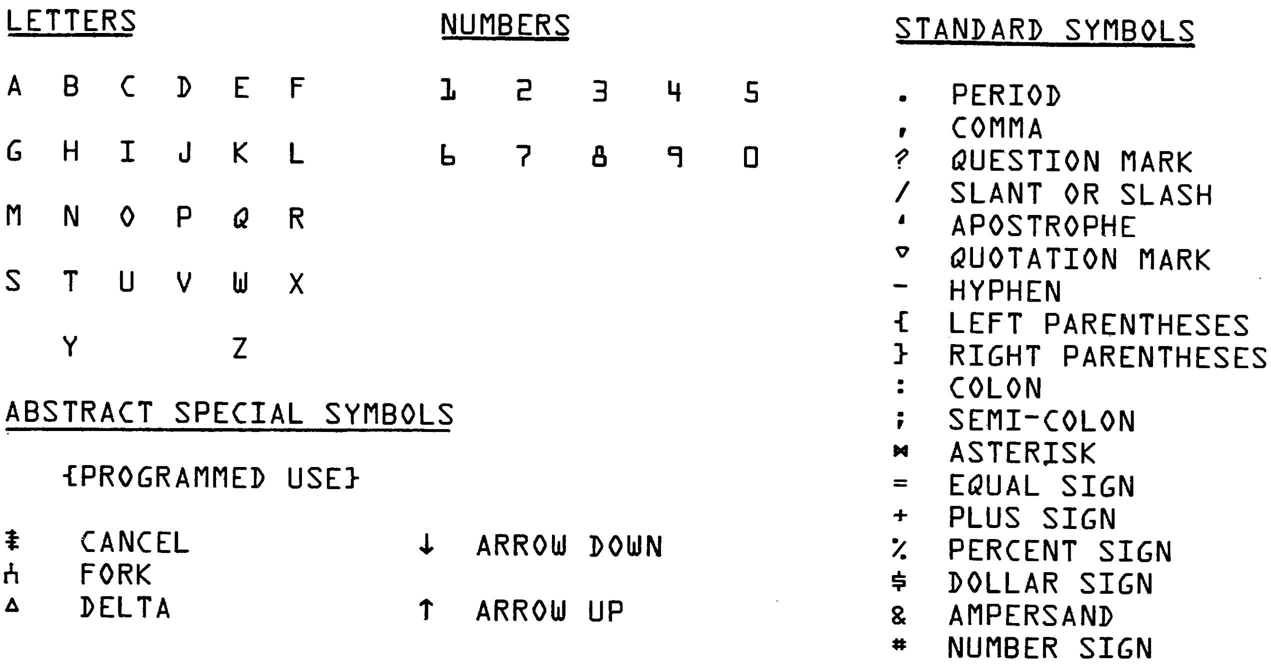


Figure 10-39. Optical Page Reader.

This unique device scans a printed page and outputs the BCD codes for the printed characters to a computer. It can be programmed to ignore lines that contain mistakes, yet read the remaining data on the page. Characters are read at a rate of 370 per second. If a line contains a character that is misaligned, illegible, or marginal, the line is re-scanned. A stack of documents may be placed in the input hopper where they are automatically fed to the read station. A standard 8½ by 11-inch document with six full lines per inch can be read in fourteen seconds. Smaller size or partially-blank documents are read proportionately faster.

The documents are prepared on a special typewriter to provide greater variation of character font than available on a standard typewriter (Figure 10-40).



LETTERS

A B C D E F

G H I J K L

M N O P Q R

S T U V W X

Y Z

NUMBERS

1 2 3 4 5

6 7 8 9 0

STANDARD SYMBOLS

.	PERIOD
,	COMMA
?	QUESTION MARK
/	SLANT OR SLASH
'	APOSTROPHE
▿	QUOTATION MARK
-	HYPHEN
{	LEFT PARENTHESES
}	RIGHT PARENTHESES
:	COLON
;	SEMI-COLON
⋈	ASTERISK
=	EQUAL SIGN
+	PLUS SIGN
%	PERCENT SIGN
$	DOLLAR SIGN
&	AMPERSAND
#	NUMBER SIGN

ABSTRACT SPECIAL SYMBOLS

{PROGRAMMED USE}

‡	CANCEL
⋔	FORK
Δ	DELTA
↓	ARROW DOWN
↑	ARROW UP

Figure 10-40. Special Character Font for Page Reader.

The special font provides more accuracy in character recognition and consequently eliminates re-scanning.

With the addition of an optical page reader to a computer system, documents that could be read only by man can now be read by a computer.

TYPEWRITER

A typewriter is usually included on the console of a computer system to provide a communication link between the operator and the machine. Operations may be initiated by the operator and information messages may be typed by the computer. Software programs, such as COMPASS and FORTRAN, are easily monitored and controlled by simple statements sent to the computer.

The slow, serial operation of the typewriter (15 characters per second) makes it impractical by input/output standards. However, it affords that necessary link between operator and machine and is, therefore, a common equipment on a computer system (Figure 10-41).

Many other types of I/O equipments exist that have not been mentioned. The more common types have been briefly discussed to provide you with a better understanding of their functions. When connected to a computer, these equipments transform the computer into a computer system.

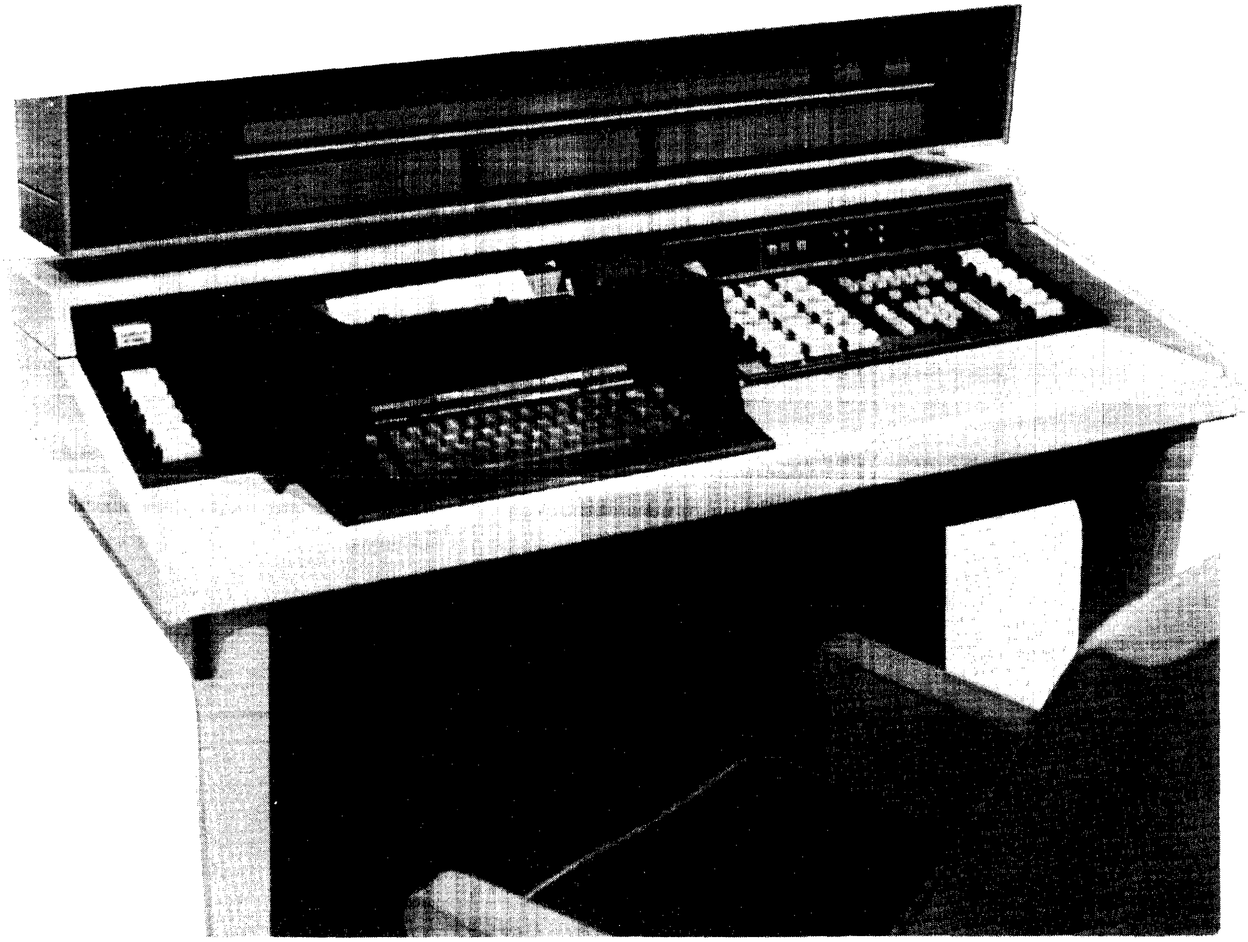


Figure 10-41. Console Typewriter.

CABLING

Now that you understand the function of the computer and some of the more common peripheral equipments, we should discuss the manner in which units can be cabled together into an operating system.

Before installation of the computer, the area is air-conditioned and false flooring is installed. The false floor serves two functions:

1) It provides a means of running the interconnecting cables under the floor.

2) Conditioned air may be forced under the floor to the individual equipments. Fans in each cabinet draw air through holes in the floor (under the equipments) to provide cooling for the electronic circuits.

The first component of the system to be installed is the computer main frame and I/O channels. Then the console is installed and connected to the computer by nine cables running under the false floor (Figure 10-42). Each of the cables contains 61 conductors, providing enough circuits to display the contents of all of the registers and to accomodate the many signals from the typewriter, keyboard, and various switches.

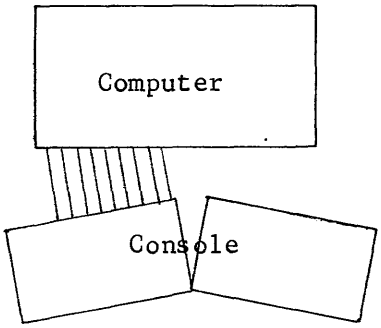


Figure 10-42. Main Frame and Console Cabling.

The next step is to decide which equipments are to be connected to each channel. The system is to have a card reader, a card punch, a line printer, and four magnetic tape units. By using dual-channel controllers, the line printer and the tapes may be connected to two channels. Because the number of peripheral units is relatively small, two I/O channels should be suffiecient.

Most of the I/O equipments contain 1604-type logic circuitry, even though the computer itself contains 3600-type circuitry. Special cards are used to convert from one type of logic signal to the other type. The same I/O equipments may be used on any type of computer, regardless of the type of logic used. For this reason, 1604-type logic is used and logic levels are converted where necessary.

Data transmission within a cable is accomplished by using a twisted-pair, impedence-matching transmission line. The line-to-line potential is always 0.5 volts. The respective polarity of the two lines determines whether a "1" or a "0" is being transmitted. The cable is terminated at each end by a transmitter and a receiver card. Logic level voltages (-5.8 or -1.1 volts) feed the transmitter card at one end of the cable. At the other end of the transmission line, a receiver card translates the signal and converts it back to a logic level voltage. If the equipment receiving the signal contains 1604 logic circuits, a modified receiver card converts the data into 1604 logic level voltage (-3.0 or 0.5 volts). A single transmitter card is capable of feeding up to 20 receivers along a 200-foot cable.

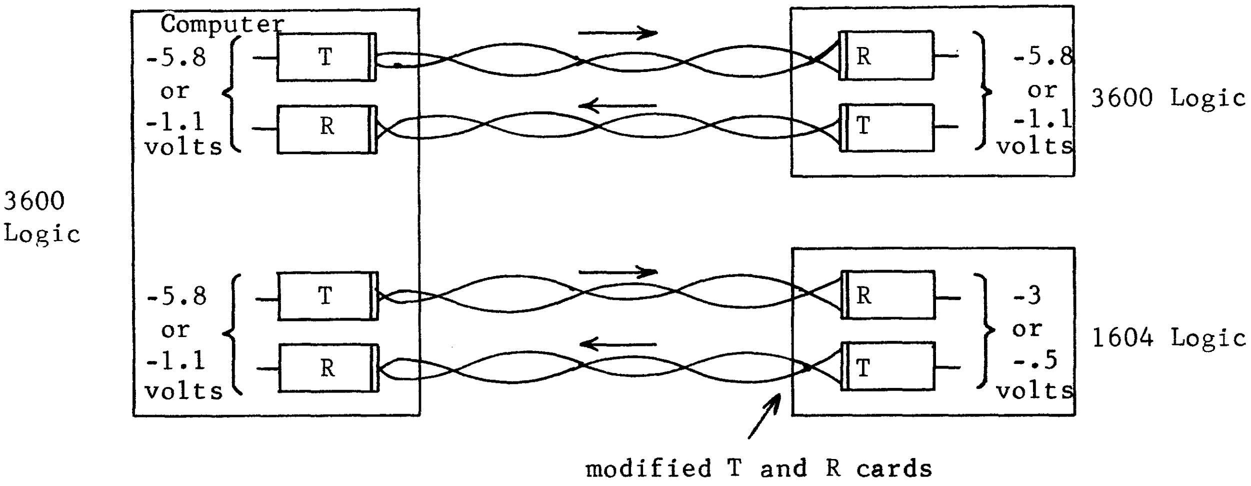


Figure 10-43. Transmission of Data (3600-Type).

If data is to be transmitted between two equipments that both use 1604-type logic, transmission is accomplished using L and M cards. All equipments are connected to a common ground and data is transmitted on a single conductor. The potential varies from -16 volts, for a logical "0", to 0 volts, for a logical "1" (Figure 10-44).

Some peripheral controllers have 1604-type logic while others have 3600 logic. The system under discussion has four controllers. The tape controllers use 3600-type logic but the other three use 1604 logic.

The card reader, the line printer, and the magnetic tape units are to be connected to channel 0. The card punch, the line printer, and the magnetic tape units are to be connected to channel 1.

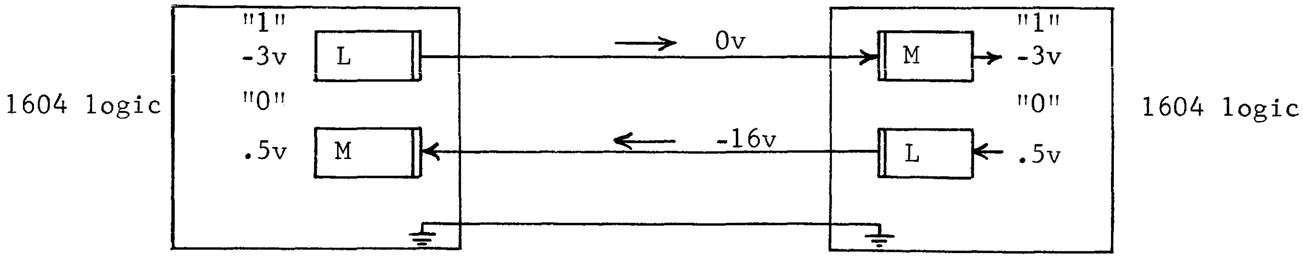


Figure 10-44. Transmission of Data (1604-Type).

The controllers are connected in series to their respective channels. Two cables connect the I/O channel to a controller, one for data and the other for control signals. Another set of cables connects that controller with the next controller. This process continues until all controllers on that channel have been connected (Figure 10-45).

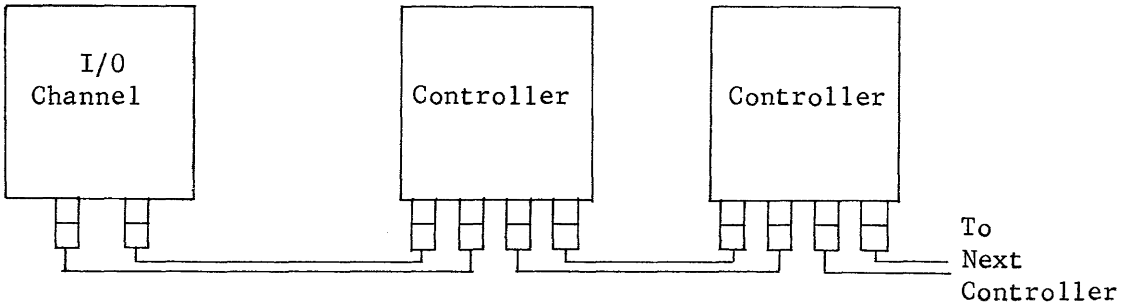


Figure 10-45. Series Cabling.

If 25-foot cables are used to interconnect the maximum of eight controllers, the total cable length of 200 feet is within the specifications for the transmitter cards. Figure 10-46 illustrates how the controllers would be connected to the previously-assigned channels.

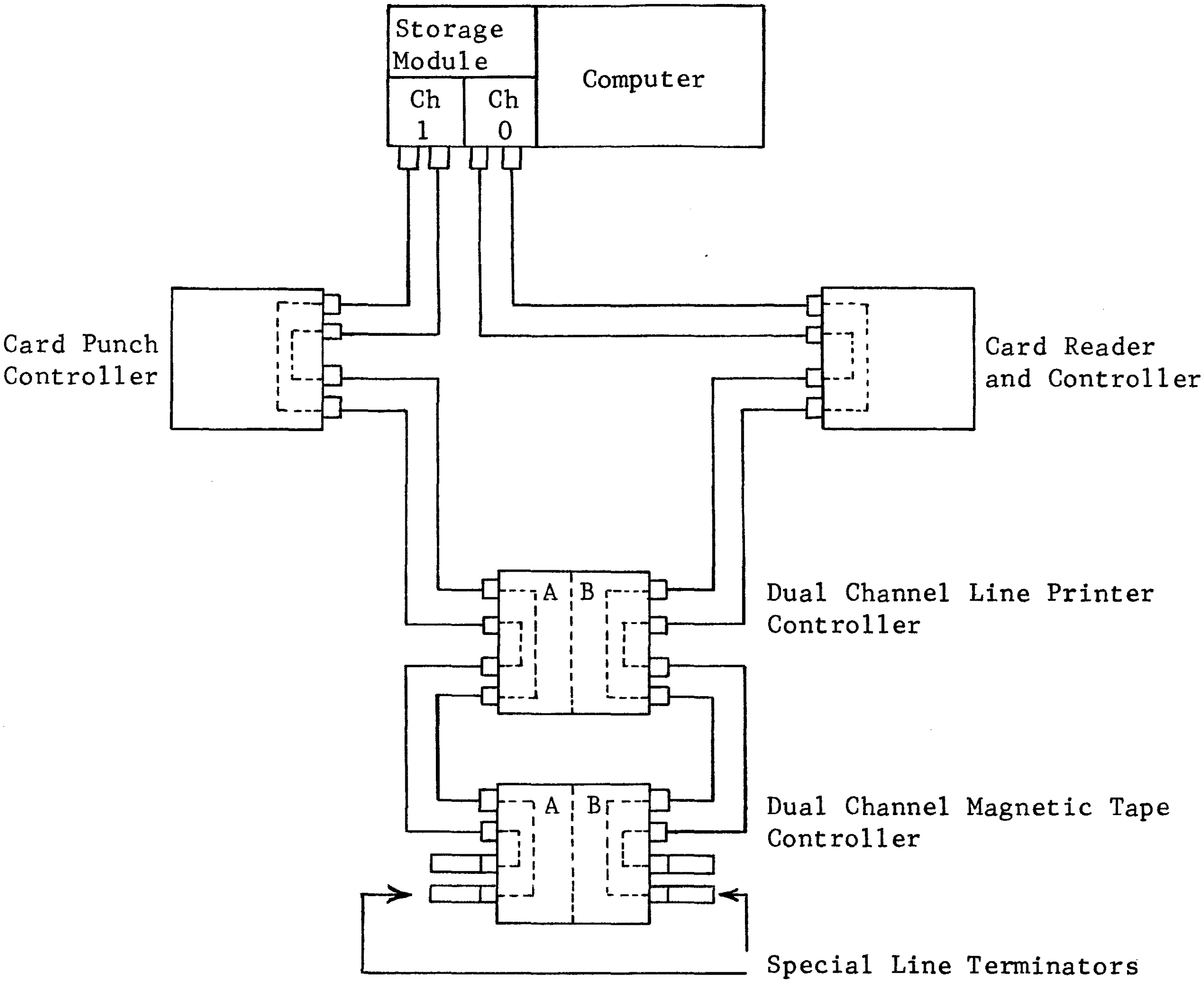

Storage Module
Ch 1
Ch 0
Computer
Card Punch Controller
Card Reader and Controller
A
B
Dual Channel Line Printer Controller
A
B
Dual Channel Magnetic Tape Controller
Special Line Terminators

Figure 10-46. Controller Cabling

Now that the controllers are cabled into the system, the I/O equipments themselves can be connected. Figure 10-47 illustrates how the card punch is connected.

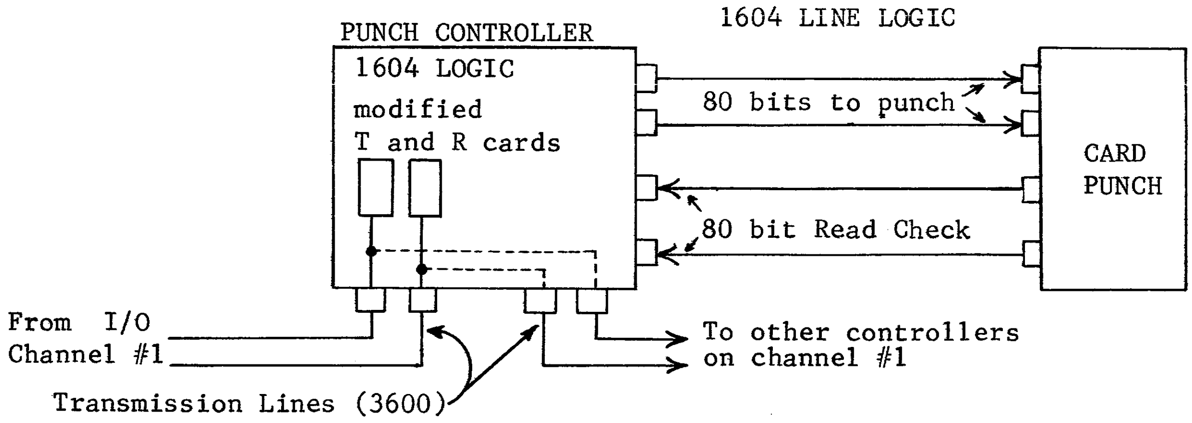


Figure 10-47. Card Punch Cabling

The card reader and controller are in the same cabinet, which obviates the necessity of external cabling. The line printer cabling is illustrated by Figure 10-48.

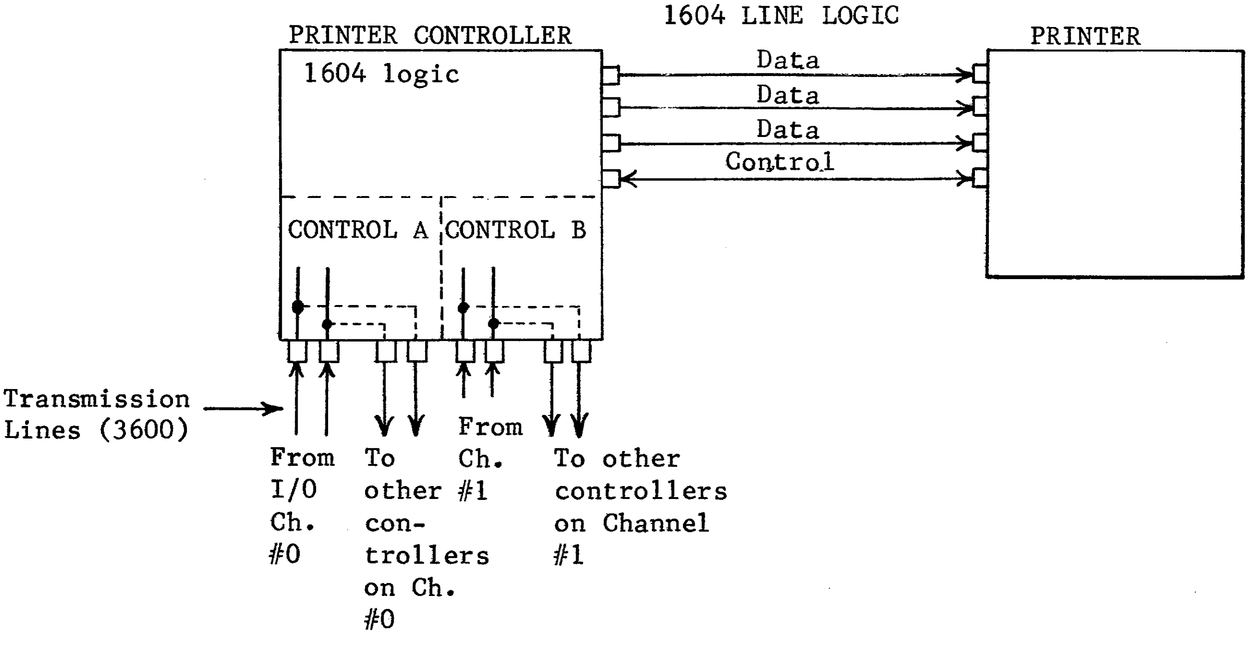


Figure 10-48. Line Printer Cabling

The last equipments to be connected to the system are the magnetic tape units. Although the controller can handle up to eight units, only four

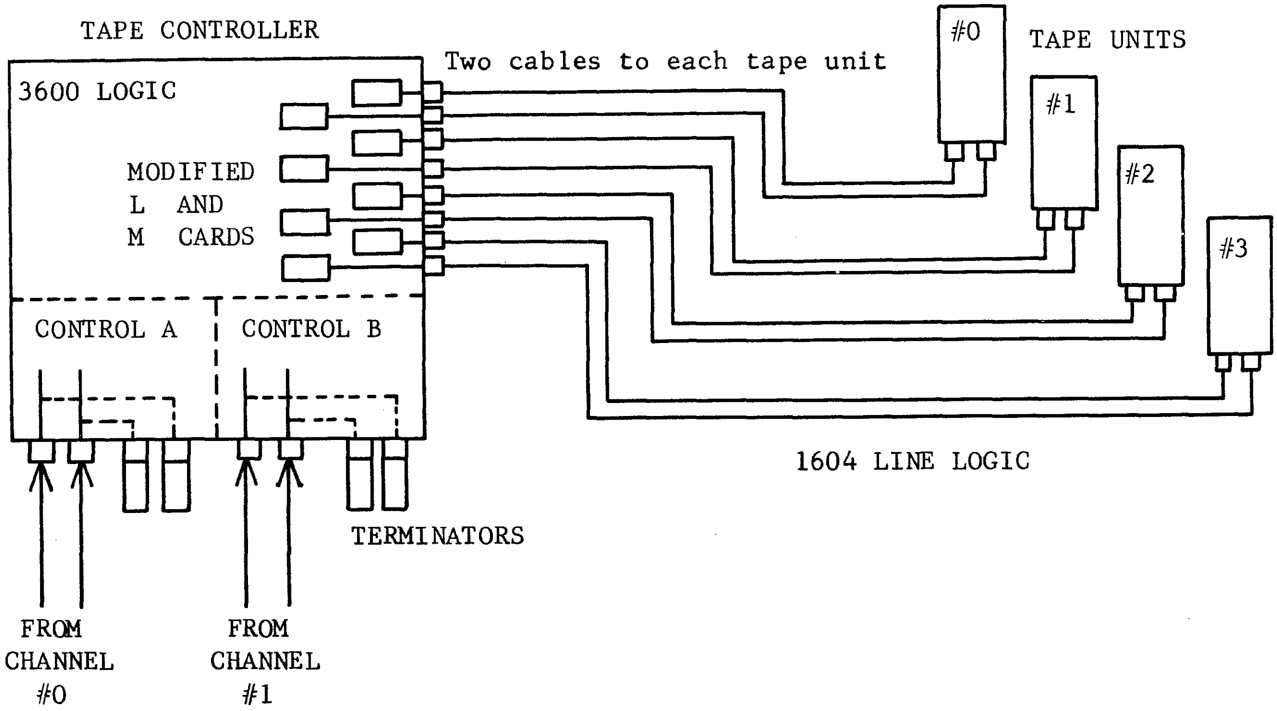


Figure 10-49. Tape Unit Cabling.

The entire system is now connected and ready to begin many hours of accurate computation. Notice that each transmission line is terminated at the last controller connected to each channel. The terminators are to match the impedence of the transmission line thereby eliminating line noise and standing waves.

The entire system, with interconnecting cabling, is illustrated in Figure 10-50. The function of each equipment should now be evident and its value to a computer system more meaningful.

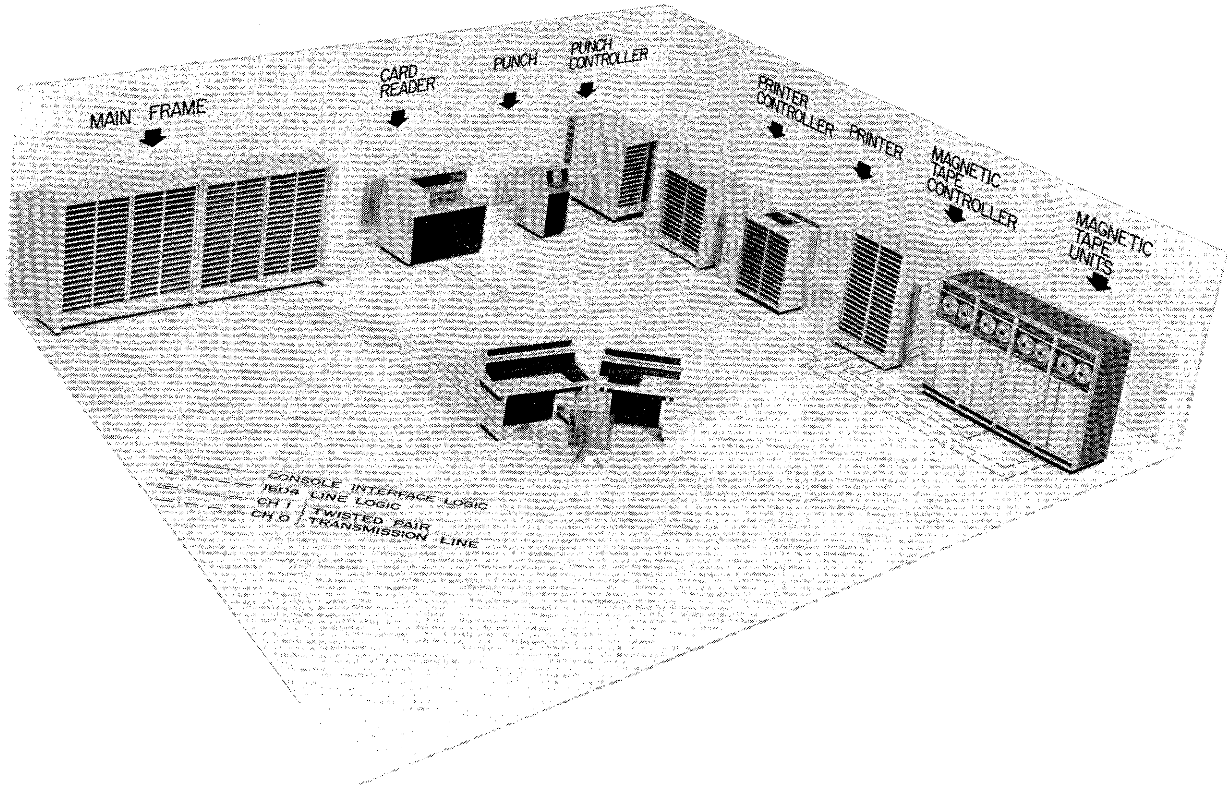


Figure 10-50. Computer System.

1. What is the function of a peripheral controller?

2. Why is it necessary to have a different controller for each type of I/O equipment?

3. A tape unit controller can have eight tape units connected to it. Why is a line printer controller restricted to a single line printer?

4. Why are four cables required between the line printer and its controller?

5. What is meant by a buffered operation?

6. How does the computer know when enough information has been transferred?

7. What is the function of an I/O channel?

8. What happens if two channels have data to be placed into storage at the same instant?

9. List four output devices. List four input devices. List three devices that may be used for both input and output operations.

1.
2.
3.
4.

1.
2.
3.
4.

1.
2.
3.

10. What are the advantages of magnetic tape over paper tape?

11. What are the advantages of magnetic disk equipment over magnetic tape?

12. Why is it necessary to have a line printer in a computer system in addition to the console typewriter?

13. List three formats for recording data on punched cards. Explain each.

14. Why are punched cards used instead of magnetic tape?

15. What is meant by "non-return-to-zero" in reference to magnetic recording?

16. What is parity? Why is it used?

17. How does a computer using 3600 logic communicate with an I/O equipment using 1604 logic? Why do the I/O equipments have 1604 logic instead of the faster 3600 type?

18. Why is it necessary to have a memory in the card reader controller?

19. What is the function of the zone counter in the line printer controller? What count would it contain if the words CONTROL DATA CORPORATION were to be printed on one line?

20. Why are cards punched by row instead of by column?

SUMMARY

The discussion of the fourth and final section of a digital computer should enhance your understanding of a computer system. Input/output operations are initiated by the program but then continue under control of the block control section. Requests for data transfers are recognized by a scanner which monitors each I/O channel. When a request is recognized, the current address is obtained from the register file and updated in preparation for the next word. Transfer rates depend on the speed of the I/O device.

The function of some of the more common I/O equipments has been explained. Input/output media may be magnetic tape, paper tape, punched cards, visual displays, or printed documents. Even though these equipments are not part of the actual computer, a basic understanding of each is necessary to full appreciate the operation of the I/O section.

Finally, the I/O equipments were connected to the computer. Cables were laid under the false flooring and interconnected to provide the desired system configuration. The four sections of a computer and the I/O equipments now constitute a functional computer system.

ANSWERS TO REVIEW QUESTIONS

1. A controller acts as an interface to synchronize the slower I/O equipments and the computer.

2. Each type of I/O device performs a different function in a different way and must, therefore, communicate accordingly.

3. It is quite common to have four to eight tapes on a system but one line printer is quite often sufficient. The added circuitry would not be justified for most systems, as it would not be used.

4. The printer controller contains the information to be printed in a special memory. The line to be printed could be all of the same character and up to 140 characters would then be printed simultaneously.

5. The computer initiates the operation but then resumes normal program execution -- sharing memory with the I/O operation. Data is buffered in and/or out while memory is also being alternately shared for program execution.

6. The register file contains the starting and last word address+1. After one word has been buffered, the starting address (current address) is updated by one to the address of the next word. When the starting (current) and LWA + 1 addresses are equal, the operation is complete.

7. The I/O channel communicates between the controller and the computer. It assembles and disassembles storage words from 12 to 24 bits and generates the required control signals between computer and controller.

8. The scanner monitors the channels in sequence. Even though both have data to be transferred, the first one checked by the scanner will be answered first. After that one word has been transferred, the scanner is released and the other request is honored.

9.
   1. Card Punch
   2. Magnetic Tape
   3. Typewriter
   4. Printer

   1. Card Reader
   2. Page Reader
   3. Magnetic Tape
   4. Magnetic Disk

   1. Magnetic Tape
   2. Magnetic Disk
   3. Typewriter

10. Much faster, more compact. However, paper tape is fine for short programs and routines.

11. Although disk equipments read data serially, transfer rates are relatively high and nearly comparable to a high speed tape unit. The prime advantage is that data on any disk surface may be accessed

within 0.2 seconds as compared to over 3 minutes for a record near the end of a reel of tape.

12. The console typewriter is a serial device suitable only for the output of short phrases whereas a line printer may print an entire line of up to 140 characters simultaneously. Line printer output would approximate 120,000 characters a minute as compared to less than 1000 characters per minute for a fast typewriter.

13. Binary. Data is punched in the card to represent numbers. No alpha characters may be represented. Information is placed in memory as it appears on the card with each 3-bit tyte representing an octal number.

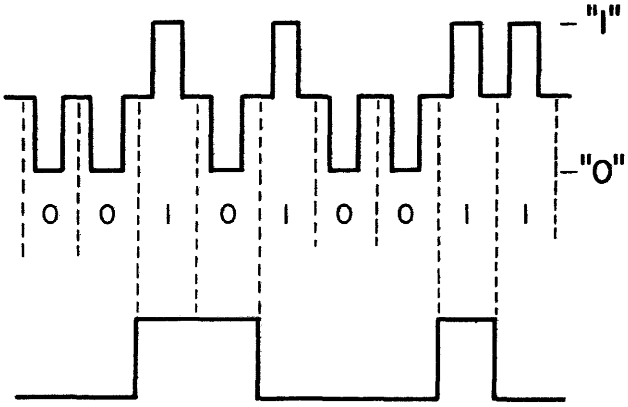
    Binary Coded Decimal (BCD) format may be used if alpha and numerical characters are to be represented. Each 6-bit byte represents a BCD character.

    Hollerith format is used for requirements similar to BCD. However, each column on the card (12 bits) represents one character. Data contained on the card may be printed at the top to facilitate interpretation. Each 12-bit byte is reduced to a 6-bit BCD character before being placed in storage.

14. Although magnetic tape is faster, the source program must first be placed on cards. If the program is to be run on a fast computer, data is sometimes transferred from cards to tape to take advantage of the faster input medium.

15. Using discrete recording techniques, the direction of magnetization of the surface determines the presence of a "1" or a "0". After each bit is recorded, flux density must be returned to zero before recording the next bit. Using the NRZ method, flux direction changes each time a "1" is to be recorded but need not return to the zero state. The NRZ method allows increased packing densities to be used.

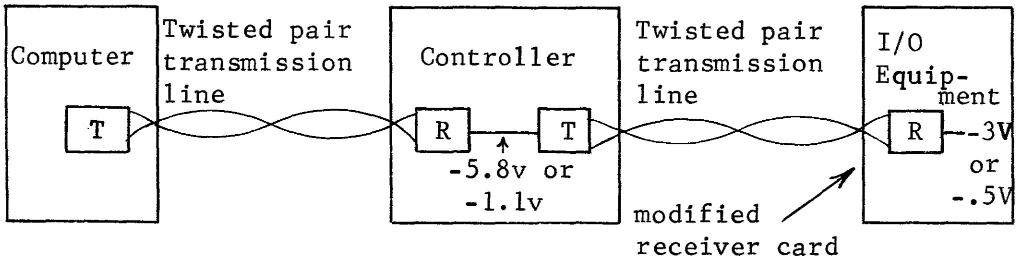
123 RETURN TO ZERO METHOD



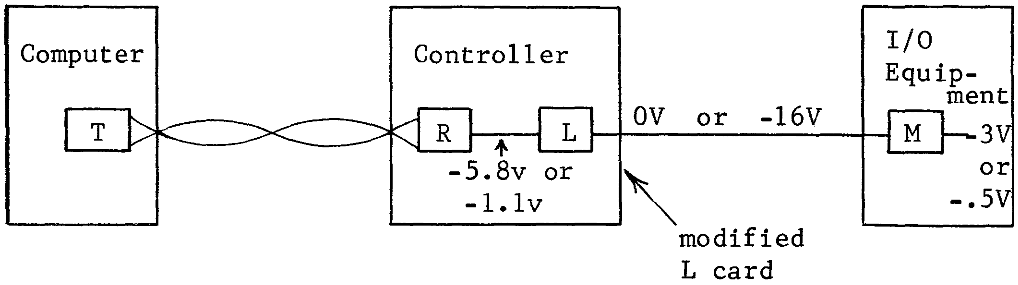
123 NRZ METHOD
EACH CHANGE REPRESENTS A ONE

16. Parity provides a means of checking for lost data. By recording information with a parity bit to make the total number of bits odd or even, a lost bit is easily detected.

17. Special logic cards convert from one logic level to another. For example, suppose a computer and a peripheral controller use 3600 logic but the I/O equipment uses 1604 logic. Communications between the equipment and the computer could be by either of the two examples.



3600 line to 1604 logic



3600 logic to 1604 line

The I/O equipments may be used on any type of computer system. When the computer has logic other than the 1604 type, special circuit cards are used for transposition from one logic to another. Most I/O equipments are relatively slow which make the 1604-type of logic quite suitable.

18. The line printer may print the contents of 30 storage locations at the same instant. Therefore, it is necessary to transfer the contents of those thirty locations into a memory which can then be "dumped" on the line printer as required.

19. The zone counter determines how many different types of characters are to be printed on a line. By decrementing the zone counter as each character(s) is printed, the memory can be reloaded when the count reaches zero instead of watiing for a complete drum revolution. 12 octal.

20. By punching by row, twelve mechanical operations are required instead of 80. Although special manipulation of data is required, the time saving more than justifies the special handling.

CONTROL DATA

CORPORATION