UNITED ARAB REPUBLIC

THE INSTITUTE OF NATIONAL PLANNING

Memo. No. 838

DIGITAL COMPUTER SYSTEMS

By

Dr. Ing. Farid M. Badran

MAI. 1968.

CAIRO
3 MOHAMED MAZHAR - ZAMALEK

DIGITAL

COMPUTER

SYSTEMS

Dr. -Ing FARID M. BADRAN

Lecturer in Electrical Engineering

Cairo University

Institute of National Planning

PREFACE

This monograph is the outcome of the lectures presented by the author at the Institute of National Planning, Cairo. In fact, this institute has been carrying out pioneer work in the field of digital computation and numerical methods since it aquired an IBM 1620 in 1963, with limited input-output equipment. This was the first electronic stored-program digital computer introduced in Egypt. Soon the sucess of this installation opened the way for many others, also from other firms, namely NCR and ICT for commercial and scientific applications, the latest being the large-scale ICT 1905 E for the Cairo University (1968). Further, the Institute of National Planning has been carrying on many useful activities, including research work in numerical methods and econometrics, publication of technical and educational reports and memorundoms, compilation and publication of international bibliography on operations research and allied topics. The institute also provides regular courses as well as educational and computational assistance to many scientific, industrial and commercial organizations.

In this framework, this monograph is addressed to engineers and scientists who want to know the performance of digital computers, as well as to business men and managers who are mainly concerned with their economic aspect. It is written mainly for the non-expert in order to aquire a general understanding which he needs to effictively use the computer to solve problems in his own field. For this purpose main emphasis is made on principles and methods underlying the operation of the digital computer as a system. A detailed description of computer components would probably become obscelete within a few years due to the rapid development in this field.

Cairo, May 1968

Darid M. Badran

DIGITAL COMPUTER

SYSTEMS

part 1

INTRODUCTION TO

THE DIGITAL COMPUTER

CONTENTS

PART I

ZERO

Before the golden age of Arab civilization, the Romans used a very cumbersome number system, For example, 1776 is tediously symbolized in the Roman system as MDCCLXXVI, which meant: M (One thousand) plus D(five-hundred) plus C(one hundred) plus C(one hundred) plus L(fifty) plus X (ten) plus X(ten) plus V (five) plus I (one).

At the beginning of the thirteenth century, the Arabs introduced the Arabic system into Europe, and the Arabic notation of numbers is still used up today. Compared with the Roman system, the Arabic system has brought two advantages: the concept of ZERO as a number and the use of the principle of position, both unknown to the Romans.

The zero can be traced back to India, where it had the meaning or "blank". In the tenth century the Arab mathematicians Translated the Indian word for zero, namely SUNYA into the Arabic word SIFR, which means "void" or "nothing" in content. In the thirteenth century, sifr was latinized to ZIPHIRUM, which in course of time became the Italian zero, as it is used today. Infact, the use of zero marked a tremendous milestone in the thinking of mathimaticians and logicians, and without it modern logic and mathematics would be impossible.

The great ease and systemization brought about by the Arabic decimal number system enabled the development of calculating instruments and machines such Napier's Rods in 1617, Pascal's calculator in 1642, Leibnitz machine in 1694 and Babbage's difference engine in 1822. Now we come to the era of large scale digital computers capable of carrying out automatically large-scale mathematical problems at high speed. As F. Cajori wrote, "The miraculous powers of modern calculators are due to three inventions: the Arabic Notation, Decimal Fractions and Logarithms".

Chapter 1

INTRODUCTION

Man has the preveldge of a central position in the universe. This central position is exactly the mean between the tiny dimensions of the atom and the astronomical distances in the solar system. In figures, we have:

Seperation of nucleus (proton) and electron in the normal Hydrogen atom H^1 is 5.294×10^{-9} cm

Mean distance from earth to sun = 1.496×10^{13} cm

Geometric mean = $\sqrt{(5.294 \times 10^{-9})(1.496 \times 10^{13})} = 250$ cm.

This geometric mean is comparable with human dimensions. No wonder that nature has provided man with mind and talent to harness atoms and to conquer the universe. Man has reached these goals through hundreds of centnries of cultural development.

The cultural development of man has always been characterized by his success to extend his limited potentialities in some direction. Thus , in the prehistoric time of the neolithic world, settled agriculture took the place of hunting and food-gathering, which is actually an extention to man's ability. This caused a new division of labour which transformed social and cultural processes.

Man's continued efforts to extend his potentialities came to another outstanding climax in the 18th century. During the period from the middle 18th to middle 19th centuries, the trensition from agricultural and home industry to modern industrial period was caused by great changes in methods of manufacturing. The steam engine patented by James Watt in England in 1769 became a basic source of power. At the same time, coal mining became a great industry due to the adoption of coke in the smelting of iron, which greatly increased the production and uses of iron. Coal and iron took the place of wood, wind and water at the center of new industry. This was the time of the industrial revolution which started first in England, and concurrently in many other countries, particularly in the USA, France, Germany and Japan.

Tremendous social and economic changes accompanied the industrial revolution. The invention of new machines took both the industries and the workers from the homes to newly established factories, a fact which shifted the people from country to city and led to the founding of many factory towns and cities. This also led to the division of people into two classes, agricultural and industrial, capital and labour. Industrial countries followed an imperial policy, due to greater need of raw materials, of markets for their greatly increased production, and advantageous investment of newly acquired wealth. This also sharpened the differences between wealthy and poor societies, due to the failure of political leadership to keep pace with the numerous complex problems arising from the rapid advances in the production and distribution of goods. In fact, most of the problems of our time seem to be due to the fact that the tremendous changes taking place during the industrial revolution have not been fully interpreted or assimilated by governments.

Continued efforts in scientific research and technology have enabled man to extend his potentialities in all directions. Trains, motorcars, airplanes afford man high speed; telephony and telegraphy as well as broadcast extend his ability to communicate, television provides man an extention for his vision, while radar is an extention to his sensing organs. The research work during the last war as well as the tremendous recent developments in electronics have enabled man to extend the potentialities of his mind in several directions, namely in the speed of calculation, in storing information and in decision-making or choice between two alternatives. This has led to the invention of the automatic digital computer.

Since the mid-twentieth century, the world stands on the threshold of the computer or second industrial revolution. While the first industrial revolution has replaced human muscles by machines as the prime source of power, the computer revolution will effect a simillar extention in the mental sphere. The digital computer can store from thousands to millions of numbers in its storage unit or memory. It can perform millions of arithmetic operations in one second. This does not only save time, but complex scientific and mathematical problems that could not be tackled before because of the ext remely long time required, can now be very easily solved by the digital computer. To investigate closely this effect, let us compare the desk calculator with the digital computer. A desk calculator can multiply two 10-digit numbers in 6 seconds, while a high speed digital computer can do the same in $6x10^{-6}$ second. Hence, a digital computer can perform in one hour what the desk calculator do in 10^6 hours. Considering 8 working hours per day, then one million hours are 125000 days or 25000 working weeks. Calculating 50 working weeks per year, these are 500 years. Thus ONE HOUR of digital computer work is equivalent to 500 YEARS work of a desk calculator. Compared with the limited span of man's life, this is a rather unexpected fantastic extention. That is why the introduction of digital computers opens a new era in our civilization.

The first electronic computer were designed about the end of the the second world war for mathematical work in science and engineering. Recently they have been used much more widely to clerical and accounting routine work, to production planning, to the automatic control of industrial plant and machine tools, to insurance and banking, and to the reservation of passanger seats on airlines. In fact it is probably true to say that the volume of work being done by computers in the industrial and commercial fields is now greater in volume, if not perhaps in importance, than the work done in the fields of research and design.

No doubt the digital computer will introduce social and economic changes in human life that will proove to be comparable with those of the first industrial revolution. Production will be cheaper, profits will be more, and free time will increase. Beside control applications in industry and management, the digital computer has already been used to control motor car trafic In daily life, it will be possible to press a button at a computer terminal to get information about purchasing an article, or to get references about any specialized subject. A computerized university, where students get their courses from the computer is now becoming a reality. Thus, a system of self-instruction using well prepared courses stored in the computer will probably be a solution for the increasing shortage in tutorial staff at universities. As for the present time and the future, the scientific work for harnessing of atomic power and space travel would have been impossible without digital computers.

The advent of digital computers has led to the flourishing of numerical mathematics, and great efforts are now being made to develop new methods in numerical analysis. This is necessary in order to cope with the increase in the potentialities of the digital computer about thousand fold by electronic engineers in the last decade.

In the field of digital computer industry, computer companies had to merge with the large firms of electronics and communications, in order to benefit from recent advances in electronic components and production technology, such as integrated circuits, on economic basis. Thus, in Germany and England, only firms producing electronic components and peripheral equipment are manufacturing digital computers. The digital computer is not only used for arithmetical manipulation, but also for logic manipulation e.g. in calculating switching circuits of digital computers using Boolean algebra. Thus, the digital computer itself may also be programmed to simulate computers not yet built, and help to design new computers. This is of great importance, since the slow speed and limited capacity for quantitave detail of the engineer as a human being seriously hadicap him in completing a good design is the available time. Thus, the first generation of computers has given birth to a second and a third improved

Chapter 2

THE DIGITAL COMPUTER

2.1. GENERAL

Long before electronic digital computers were developed, there has been automatic systems which handled information, and processed it for other purposes than computation. We are all familier with the automatic telephone exchange, which receives dial pulses from the telephone set of the subscriber. In certain types of telephone exchanges, these dial pulses are stored in a register, then the automatic system controls the motion of selectors to the required subscriber line. Recently, the electro-mechanical relays and selectors have been replaced by fully electronic, transistorised circuits. In fact, the automatic exchange system can be represented by a programmed digital computer, to which the input is given by the calling subscriber, and the output leading to the called subscriber.

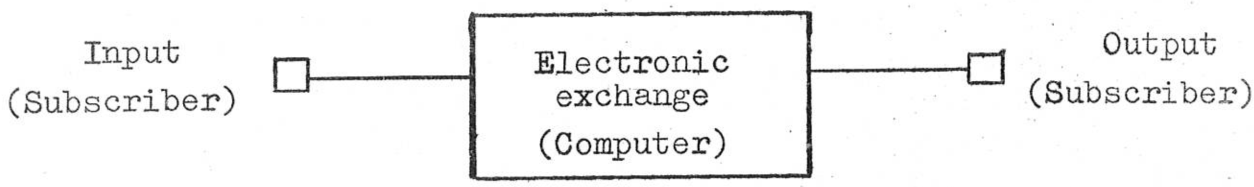


Fig. 2.1

computer generation. In fact, digital computers are machines that are seemingly able to reproduce themselves in a manner determined by their own capabilities.

Will the new changes due to the computer revolution be fully interpretted and assimilated by governments? Apart from increasing defence potentialities by computer-controlled radar systems, a new look in world polities is now prevailing. The era of colonies is over. Sound national economy based on highly specialized engineering industries is now the substitute. No wonder is then the merging of the largest electronic industries and computer companies in England into the new International & computer Limited, and the strong support of the Britsh government for the computer industry. In the doviet Union, the future of national development and political relations are no longer distated by rulers in the Cremlin, but planned by the professors of the Moscow University.

Alternatively, the digital computer itself can be considered as an exchange, to which only two subscribers are connected: input device and output device, the computer supplying the input data after processing to the output subscriber 2. Thus the computer serves to exchange date in one way between input and output after processing.

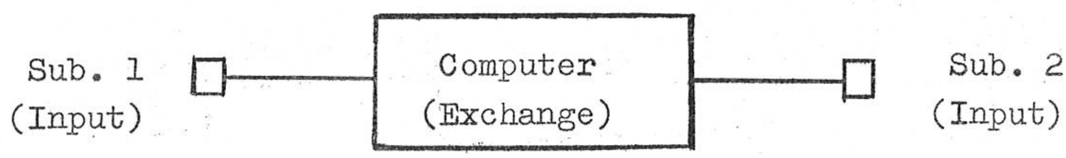


Fig 2.2

In general, exchanges and digital computers are both automatic switching systems: the exchange performs automatic switching for communication between subscribers, while the digital computer performs automatic switching for computation. As a matter of fact, automatic computers have made use of the development in exchange techniques since the technical computer terms such as: register, digits, counting chain, storage, control, random access, sequential access.... etc, . that have their origin from automatic exchanges.

2.2. BASIC CONCEPT

The digital computer consists of 5 units: input, storage, control, arithmetic and output (figure 2.3)

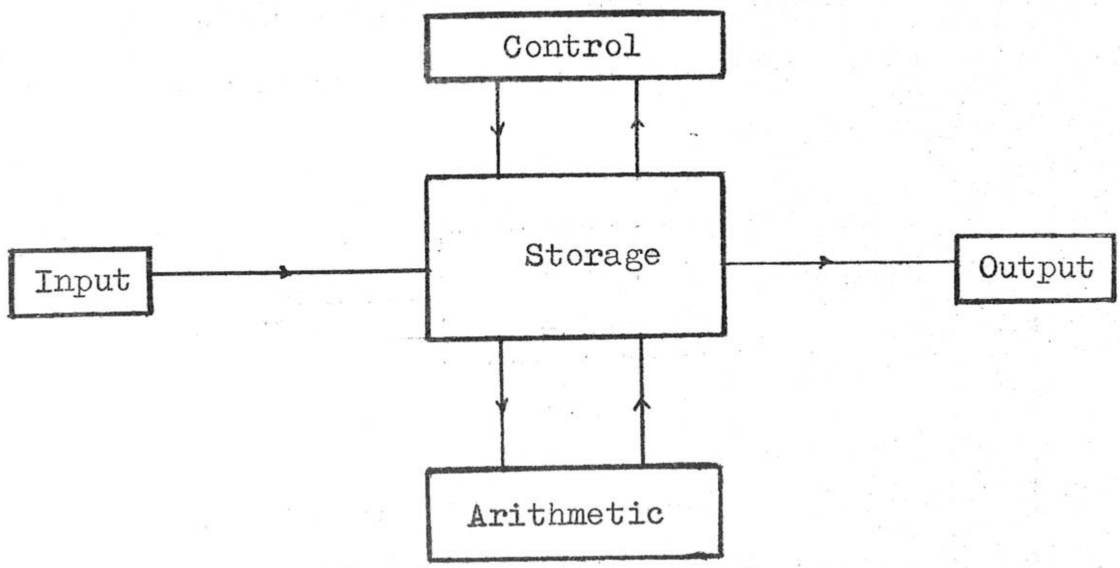


Fig. 2.3. Units of digital computer

In order to solve a problem by a digital computer, a complete set of instructions, called program, is to be prepared, reducing the solution procedure to a series of elementary steps such as addition, subtraction, multiplcation and division, that may be great in number. The data and instructions of the problem are entered into the computer by means of the input unit. The input information has to be expressed or coded in a form suitable for the computer. The input information may be punched on cards or paper tape or recorded on magnetic tape. The input unit reads the information from the card or tape and gives it to the storage unit in the computer.

One common type of storage unit is a magnetic drum rotating at high speed. The surface of the drum has magnetic properties that are used to record information. Magnetic heads are used to record and read the information stored on the drum. The access time to any location on the storage drum is less than or equal to one revolution time of the drum. Access time to the stored information is important, since it may delay further operations of the computer. The ability to store a certain amount of information is called the storage capacity of the computer. The storage capacity has to be chosed so as to cope with the problems handled by the computer. A small storage capacity may influence the programming of the computer. Each element in the storage unit is assigned a numerical address, the surface of the storage magnetic drum being divided into storage elements. In this way, the desired information in storage can be easily located, and be used in further operations.

The arithmetic unit consists of switching circuits using diodes, electronic tubes or transistors that operate on the discrete electric signals representing the numbers, and produce the results in the same form. This necessiates that the arithmatic unit would have a limited storage for holding the numbers involved in the calculation. It operates according to instructions stored in the storage, and can perform addition, subtraction, multiplication and division. Further the arithmetic unit has the ability to compare two numbers to find which is greater. In this way it has the ability of decision making, i.e. choosing between 2 alternatives. Addition and subtraction are carried out at high speeds, while multiplication and division(that are derived from addition and subtraction) at relatively slower speeds. The addition of two 10 decimal digits may take 100 usec., while multiplication of the same numbers may require 1,000 usec. The digital computer is provided with a control unit which directs and coordinates the calculation procedure according to the stored program. The control unit reads an instruction from storage, identifies it, sets up the required operations in the arithmatic unit, and at the end reads the next instruction, and so on. Thus, the control unit interprets the program given into storage. The first instruction is manually passed to the control unit, while "stop" may be programmed as last instruction. In general, control is not one constructional unit, but it is distributed through out the computer parts.

By means of the output unit, we obtain the results of the computer operations. The computer output is ordinarily given as decimal numbers delivered to an electric typewriter, teleprinter, punched card, magnetic or punched tape. Since these devices operate at much lower speed than the computer, the results have first to be stored then delivered to the output unit.

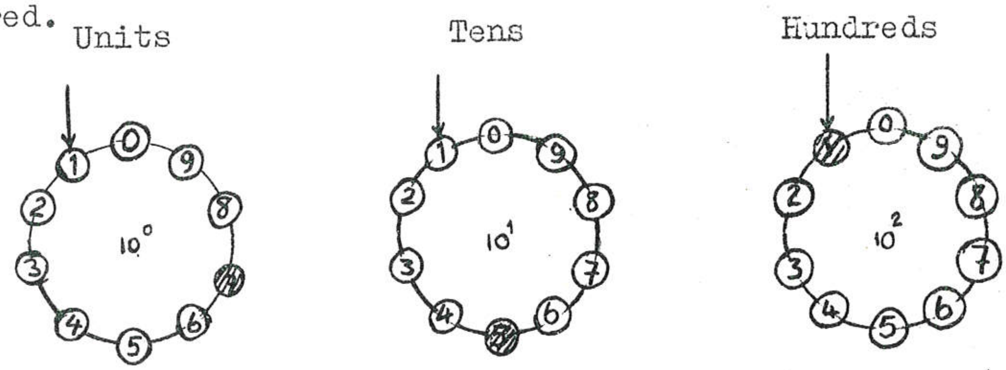
The development of digital computers is based on the following principles:

1. Solving problems by arithmetic operations,
2. Arithmetic operations are based on addition,
3. Operations are carried on integers or discrete quantitias.
4. These quantities are represented for computer operation in binary form, and this in electric signals having two conditions,
5. Storage of problems data, intermediate and final results,
6. High speed operation,
7. Instructions can be stored and used to control the operation of the computer.

2.3. NUMBER SYSTEM

The counting and storage of numbers is of fundamental importance both in exchange switching and computer switching. As an example, we may consider the register of the rotary system, having 5 chains each having ten relays each chain used to store one of the 5 digits of the subscriber number.

As a matter of fact, the first digital computers used telephone relays for storage. Consider for example the number 157. It can be stored in a storage device having 3 relay chains of 10 relays each. The first chain representing the units digit, the second chain the tens digit and the third chain the hundreds digit. Thus, a total of 30 relays or storage elements is required.



N = 157

Fig. 2.4 Decimal number storage

The number 157 can be represented by a series of powers to the base 10:

$$157 = 1 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$$

$$= \sum_{k=o}^{2} a_k \, 10^k$$

In general, any number can be represented by a power series to any convenient base or radix, B.

$$N = \sum_{k=o}^{m} a_k \, B^k$$

where $0 \leqslant a_k \leqslant (B-1)$

Since many storage elements such as relays, electronic tubes, transistors, magnetic materials ... etc. have only two stable conditions, it is convenient to choose B = 2 as the base for a number system, called binary system. While in the decimal system we can use the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, in the binary system we have only two digits: 0 and 1, each digits of them representing one of the 2 stable conditions of a storage element.

The number 157 expressed in binary system is 10011101. This can be stored in 8 relay chains of 2 relays each. Thus, the total number of relays or storage elements is 16, which is here almost 50% less than in the decimal system (Figure 2.5).

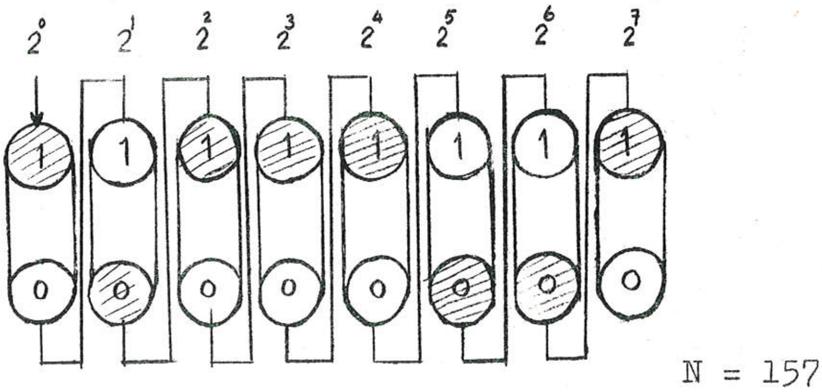


Fig. 2.5. Binary number storage

Binary numbers can be easily converted to decimal numbers:

$$(10111)_2 = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4$$

$$= 1 + 2 + 4 + 0 + 16$$

$$= (23)_{10}$$

Conversion of decimal numbers to binary numbers is performed by successive division by 2. Thus, the binary equivalent for 174 is obtained as follows:

2	174	
2	87	0
2	43	1
2	21	1
2	10	1
2	5	0
2	2	1
2	1	0
	0	1 ↑

The binary equivalent is obtained from the remainder.

$$(174)_{10} = (10101110)_2$$

Binary arithmetic is simple, it has simillar rules to those of the decimal number system, as seen from the tables for binary addition and binary multiplication.

x	0	1
0	0	0
1	0	1

Binary multiplication table

+	0	1
0	0	1
1	1	10

Binary addition table

2.4. SWITCHING ALGEBRA AND CIRCUITS

Now that we have discussed the binary arithmetic used in the digital computer, we can investigate the switching circuits that carry out the binary arithmetic operations in the computer.

These switching circuits are based on the algebra introduced by George Boole in 1847, that assumes only two values for a variable, 0 and 1. It was C.E. shannon who established in 1938 the application of Boolean algebra to switching circuits.

A switching circuit may have n inputs, each input considered as a boolean variable, and m ouptuts, that are functions of the n input of variable (figure 2-6)

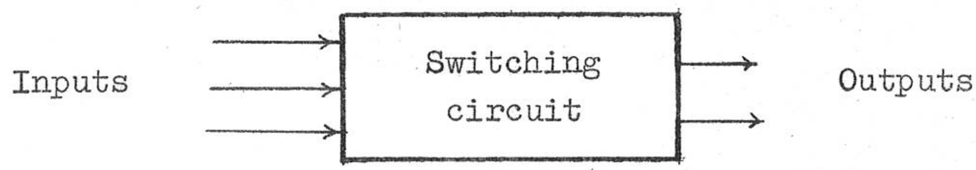
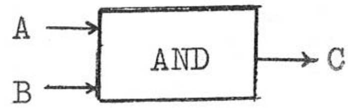


Fig. 2-6

Binary arithmetic operations, represented by boolean expressions, are based on three fundamental logic circuits or gates : the "AND" circuit, the "OR" circuit and the "inverter", also called NOT circuit



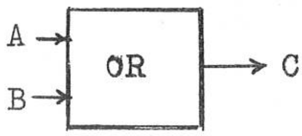
C = A and B

$C = A \cdot B$

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

Fig. 2-7. AND function



C = A or B

$C = A + B$

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

Fig. 2-8. OR function

C = not A

$C = \bar{A}$

A	C
0	1
1	0

Truth table

Fig 2-9. NOT function

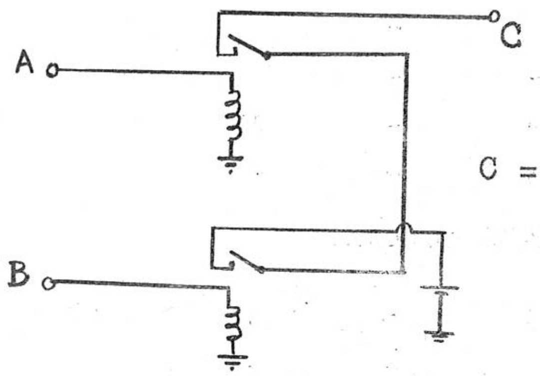
or "not" circuit. In each case, the value of the output as a function of the 0 and 1 values of the input variables is given by a truth table

The AND circuit supplies an output (1) when it simultaneously has an input on the first lead, "and" the second lead, and so on, i.e. when each of the input variables has the value 1.

The OR circuit supplies an output (1) when there is an input on the first lead, "or" the second lead, and so on, i.e. when any of the input variables has the value 1.

The NOT circuit supplies an output (1) when there is "not" an input, i.e. when the input variable has the value 0, and vice versa.

In the first computers, these circuits were realised with telephone relays (figure 2-12), which were later replaced by vacuum tubes, then by semiconductor diodes, and later by transistors.

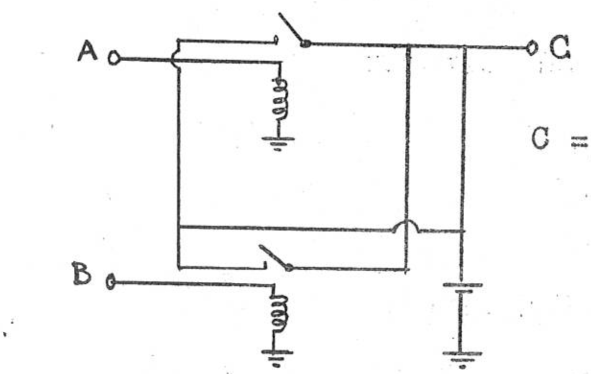


$C = A\ B$

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

Fig 2-10. Relay AND circuit

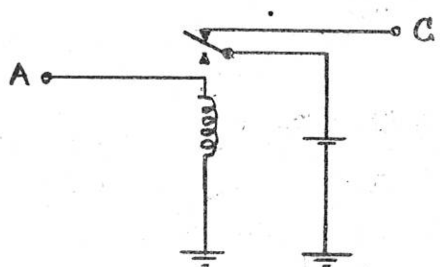


$C = A+B$

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

Fig 2-11. Relay OR circuit

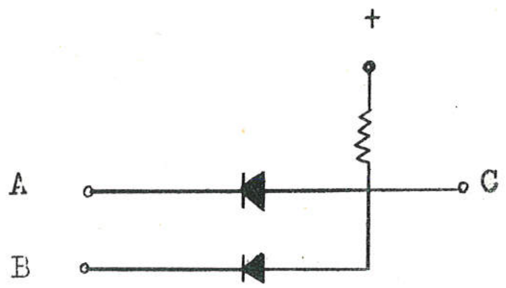


$C = A$

A	C
0	1
1	0

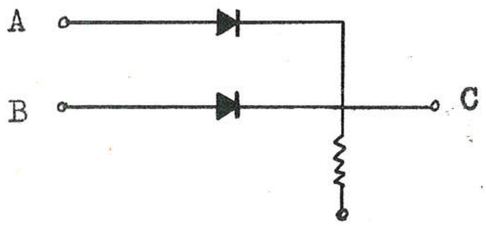
Truth table

Fig. 2-12 Relay NOT circuit



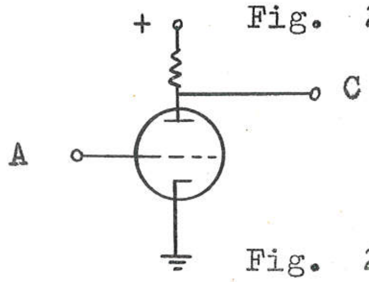
A	B	C	= AB
−	−	−	
−	+	−	
+	−	−	
+	+	+	

Fig. 2-13. Diode AND circuit



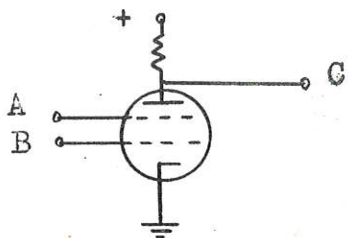
A	B	C	= A+B
−	−	−	
−	+	+	
+	−	+	
+	+	+	

Fig. 2-14. Diode OR circuit



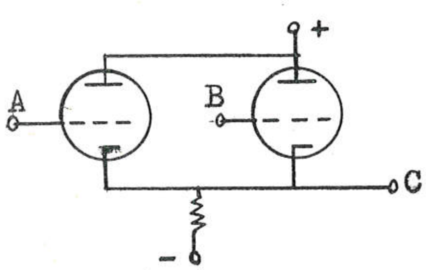
A	C	= $\overline{A}$
−	+	
+	−	

Fig. 2-15. Triode NOT circuit



A	B	C	= $\overline{AB}$
−	−	+	
−	+	+	
+	−	+	
+	+	−	

Fig. 2-16. Tetrode NOT AND circuit



A	B	C	= A+B
−	−	−	
−	+	+	
+	−	+	
+	+	+	

Fig. 2-16. Triodes OR circuit

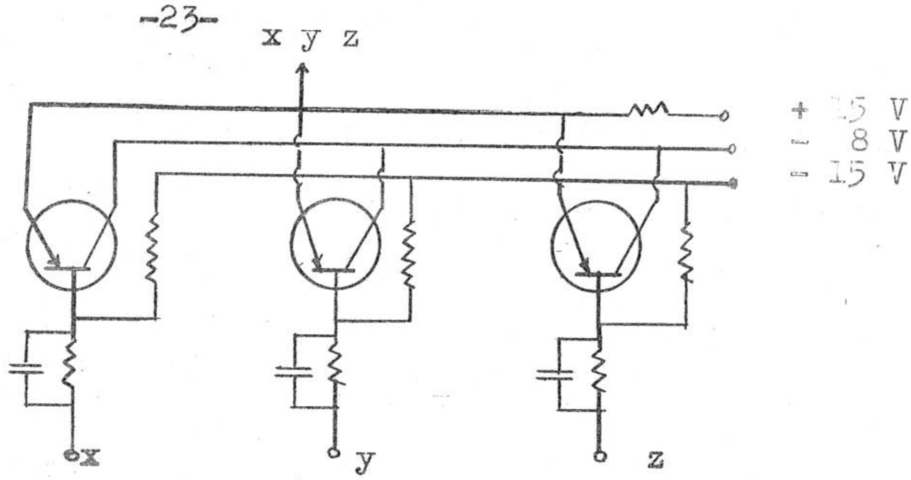
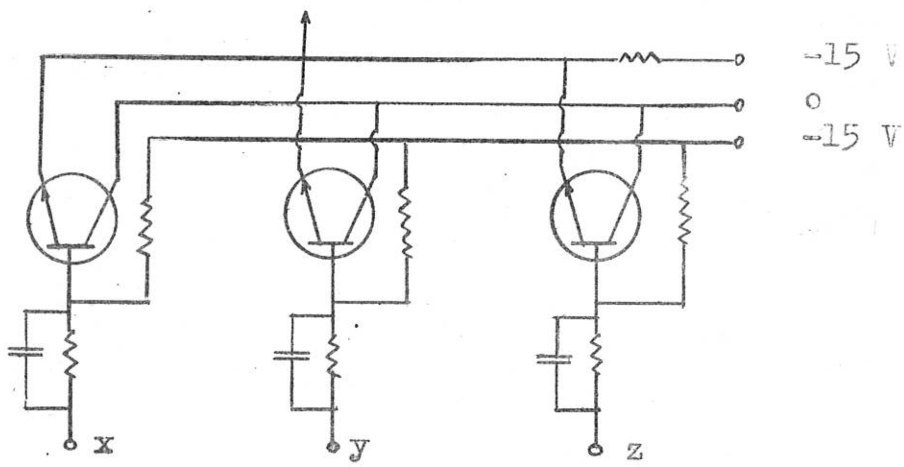
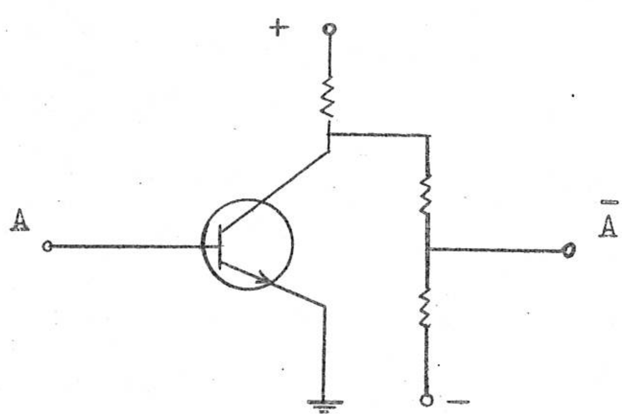


Fig. 2-17
-n-p transistor
ND circuit



ig. 2-18
-p-n transistor
R circuit



ig. 2-19
-p-n transistor
OT circuit

The given examples show how electronic circuits using diodes, tubes and transistors can perform the operations of boolean algebra: and, or and not. A switching circuit is built from a number of AND, OR and NOT units according to the boolean function describing the switching circuit.

For Boolean functions or expressions there are several useful theorems based on the basic operations or postulates explained before, namely

AND	OR
1.1 = 1	0 + 0 = 0
1.0 = 0	0 + 1 = 1
0.1 = 0	1 + 0 = 1
0.0 = 0	1 + 1 = 1

NOT $\bar{1} = 0$; $\bar{0} = 1$

For each boolean expression there is a compliment expression, obtained by chaning each (and) to an (or), changing each (1) to (0), and each (0) to (1), and complimenting the variables. For example, the compliment of $1.\bar{A} + B\bar{C} + 0$

is $(0 + A) \; (\bar{B} + C) \,.\, 1$

When the first expression equals 1, its compliment is equal to 0, and vice-versa.

If in forming the compliment, we do not compliment the variables, then we get the dual.

For example, for $1.\bar{A} + B\bar{C} + 0$ we get the dual $(0 + \bar{A})(B + \bar{C}).\ 1$

In deriving and establishing boolean theorems, they are presented as dual pairs. It can be easily seen that if a theotram is valid, then its complimentary theorem must be also valid, since every postulate has a complimentary postulate. Furthermore, since a theotherm must satisfy all values of the variables, then the dual of theorem is valid if the copliment is valid. The validity of any theorem in boolean algebra can be easily proved by perfect induction, i.e. by testing their validity for all values of the variables involved, 0 or 1.

1)	$0.X = 0$	$1 + X = 1$
2)	$1.X = X$	$0 + X = X$
3)	$XX = X$	$X + X = X$
4)	$XX = 0$	$X + \overline{X} = 1$
5)	$XY = YX$	$X + Y = Y + X$
6)	$XYZ = X(YZ) = (XY)Z$	$X+Y+Z = X+(Y+Z) = (X+Z)+Z$
7)	$\overline{XY..Z} = \overline{X}+\overline{Y}+...+\overline{Z}$	$\overline{X+Y+...+Z} = \overline{X}\overline{Y}...\overline{Z}$

8) $\overline{f}(X,Y,...,Z,\text{and},\text{or}) = f(\overline{X},\overline{Y},...,\overline{Z},\text{or},\text{and})$

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

9) $XY+XZ = X(Y+Z)$ $\qquad$ $(X+Y)(X+Z) = X+YZ$

10) $XY+X\overline{Y} = X$ $\qquad$ $(X+Y)(X+\overline{Y}) = X$

11) $X + XY = X$ $\qquad$ $X(X + Y) = X$

12) $X + \overline{X}Y = X + Y$ $\qquad$ $X(\overline{X} + Y) = XY$

12a) $ZX + Z\overline{X}Y = ZX + ZY$

12b) $(Z + X)(Z+\overline{X}+Y) = (Z+X)(Z+Y)$

13) $XY + \overline{X}Z + YZ = XY + \overline{X}Z$

13a) $(X + Y)(\overline{X} + Z)(Y + Z) = (X + Y)(\overline{X} + Z)$

14) $XY + \overline{X}Z = (X + Z)(\overline{X} + Y)$

14a) $(X + Y)(\overline{X} + Z) = XZ + \overline{X}Y$

The following examples illustate the application of these theorems to switching networks.

Example 1:

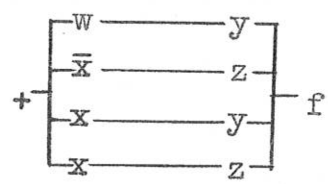
The shown relay network satisfies the boolean expression

$f = wy + \overline{x}z + xy + xz$

By simplification this reduces

$f = y(w+x) + z(x + \overline{x})$

$\quad = y(w + x) + z$



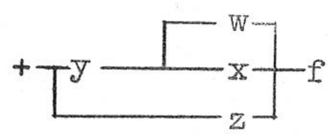


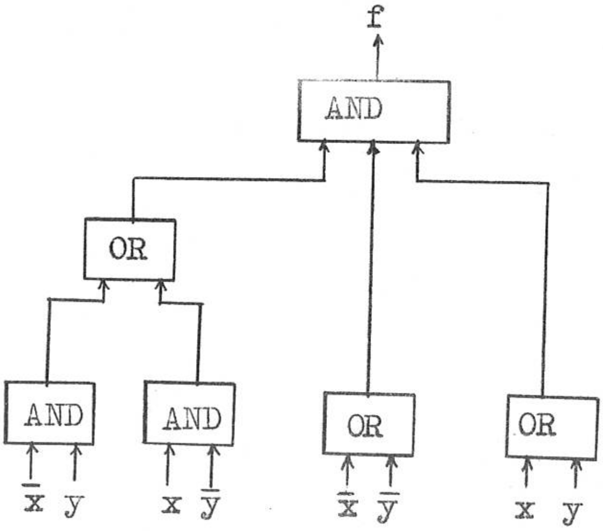
Fig. 2-20

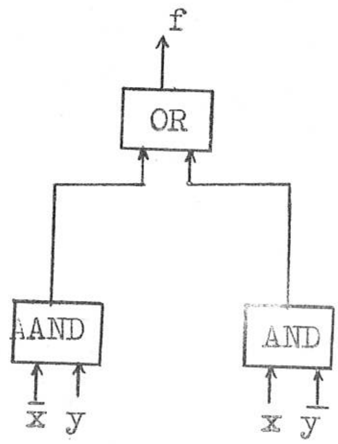
This simplified expression gives an equivalent circuit, sparing 4 contacts.

Example 2:

The following switching circuit satisfies the boolean expression

$$f = (\bar{x}y + x\bar{y})\ (\bar{x} + \bar{y})\ (x + y)$$





By simplification this reduces to

$$f = (\bar{x}y + x\bar{y})\ (\bar{x}y + x\bar{y})$$

$$= \bar{x}y + x\bar{y}$$

Fig. 2.21

which is the exclusive OR circuit shown.

2.5. BINARY ADDERS

Switching circuits can be designed to perform binary arithmetic operations in the arithmetic unit. A switching circuit for adding two binary digits or bits is called a half adder. It has two inputs A and B, and gives two outputs, the sum S and carry C.

A	0	0	1	1
B	0	1	0	1
Sum S	0	1	1	0
Carry C	0	0	0	1

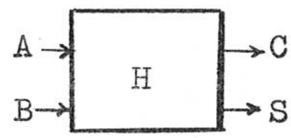


Fig. 2-22. Half adder

The condition to be satisfied by the half adder give the sum S and carry C as functions of A and B:

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

$$C = AB$$

Thus, the half adder is given by the following switching circuit:

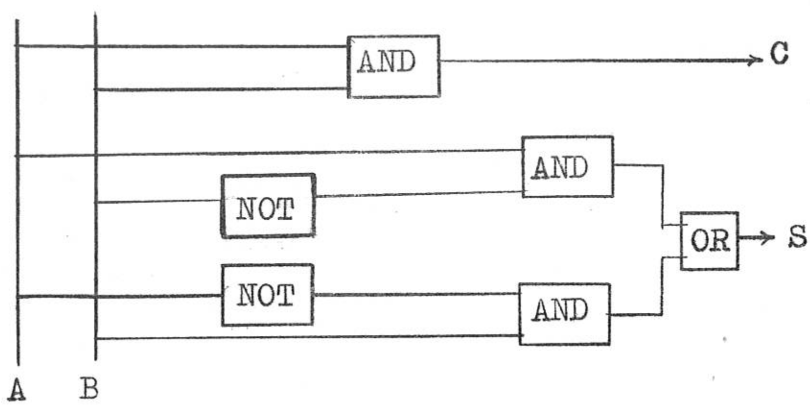


Fig. 2-23. Half adder circuit

The half adder circuit can be simplified:

$$\begin{aligned} S &= A\bar{B} + \bar{A}B \\ &= (A + B)(\bar{A} + \bar{B}) \\ &= (A + B)(\overline{AB}) \end{aligned}$$

The simplified circuit contains two switching units less than the first half adder circuit (figure 2-24).

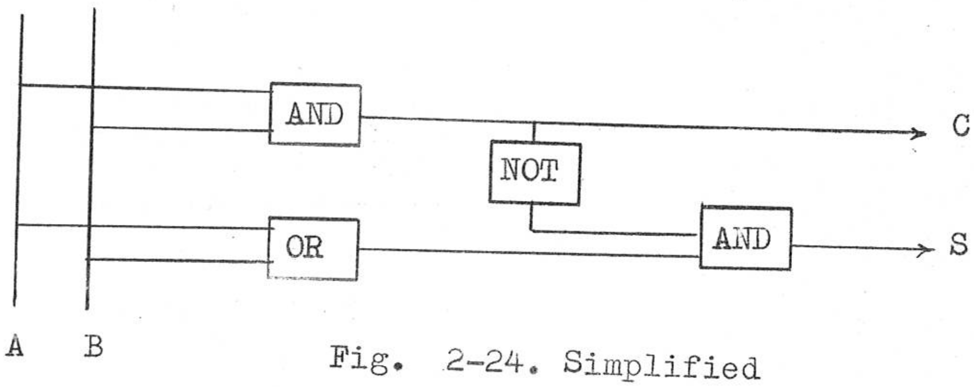


Fig. 2-24. Simplified half adder circuit

Three binary digits are added by a full adder. This is a switching circuit having three inputs A,B and C_1 and two outputs: the sum S and carry C, given by the expressions

$$S = A\,\bar{B}\,\bar{C}_1 + \bar{A}\,B\,\bar{C}_1 + \bar{A}\,\bar{B}\,C_1 + A\,B\,C_1$$

$$C = A\,B + C_1\,(A + B)$$

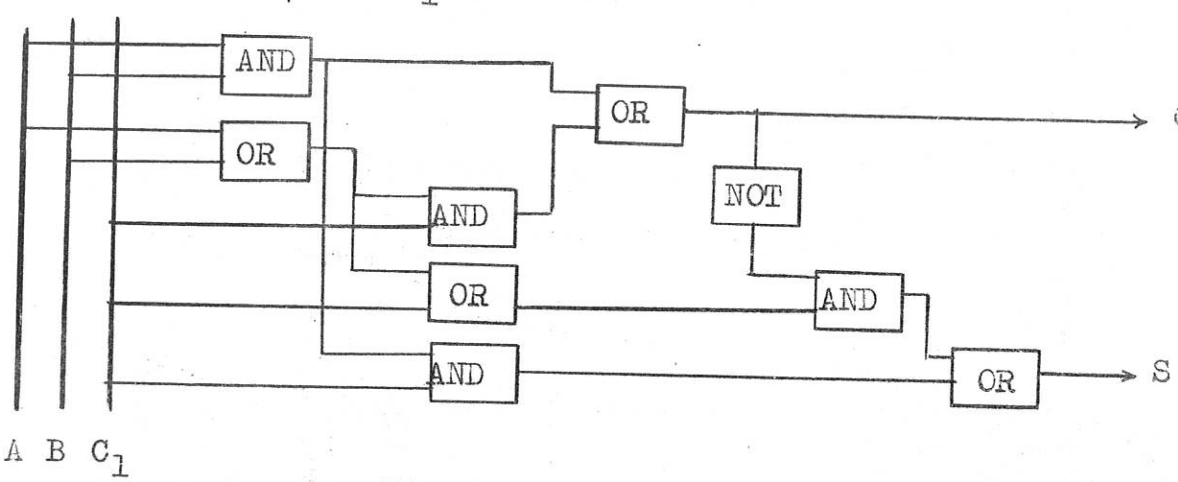


Fig. 2-25. Full adder

Using two half adders and an OR circuit, a full adder can be obtained (figure 2-26).

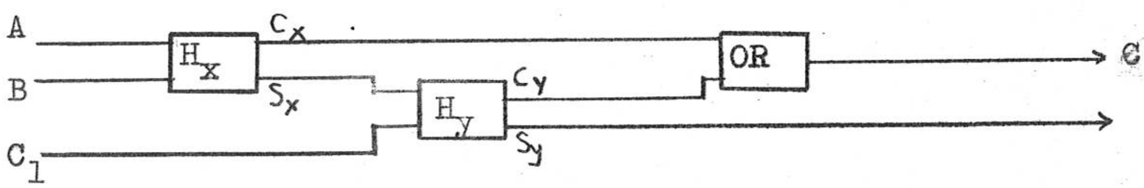


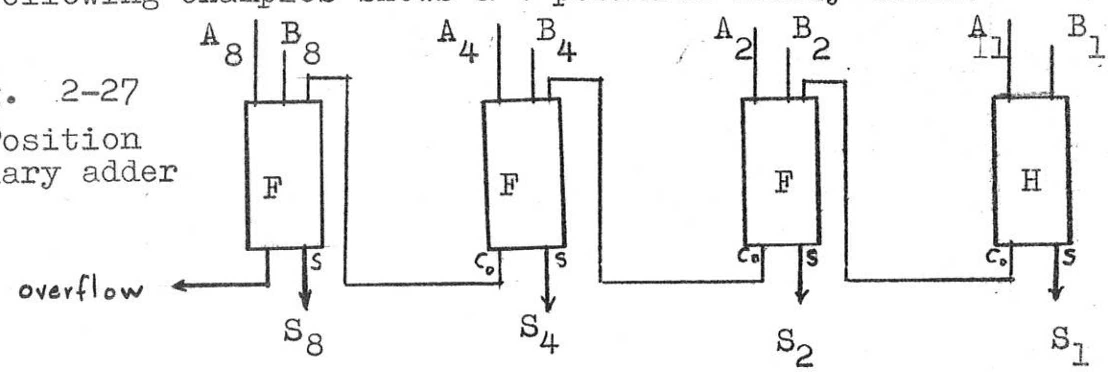
Fig. 2-26. Full adder

$$C_x = A\ B$$
$$S_x = A\ \bar{B} + \bar{A}\ B$$
$$C_y = C_1\ (A\ \bar{B} + \bar{A}\ B)$$
$$S = S_y = (A\ \bar{B} + \bar{A}\ B)\ \bar{C_1} + (\bar{A}\ \bar{B} + A\ B)\ C_1$$
$$= A\ \bar{B}\ \bar{C_1} + \bar{A}\ B\ \bar{C_1} + \bar{A}\ \bar{B}\ C_1 + A\ B\ C_1$$
$$C = A\ B + A\ C_1 (A\ \bar{B} + \bar{A}\ B)$$
$$= A\ B + A\ C_1 + B\ C_1$$
$$= A\ B + C_1 (A + B)$$

Using half adders and full adders, new adders can be derived for adding binary numbers having several digits.

The following examples shows a 4-position binary adder.

Fig. 2-27
4-Position binary adder



Multiplication can be performed by repeated addition. In binary arithmetic, every multipler digit is either 0 or 1. Thus for multiplying two digits, the multiplicand is added once for multiplication by 1, and shifting. If the number has n bits, there at most n additions. Subtraction in the computer is a process of complementary addition. This complementing process is equivalent to changing the sign. Then it is possible to proceed as in addition.

Although the complement subtraction method looks cumbersome in the decimal system, it is very simple in the binary number system. In the decimal number system, we complement any number with respect to 9. For example, the complement of 123456789 is

```
  9 9 9 9 9 9 9 9 9
- 1 2 3 4 5 6 7 8 9     ( number )
  -----------------
  8 7 6 5 4 3 2 1 0     ( complement )
```

Subtraction using complements is performed as follows:

1. Add the complement of the subtrahend to the minuend,
2. If there is a final carry-over, add it to the least significant digit (the right-hand digit). For example, 153-42= 111

Ordinary Subtraction	Complement Method
153 (minuead)	153
- 042 (subtrahend)	957 (complement of 042)
111 (difference)	1 110 sum
	1 final carry
	111 difference.

In the binary system, the complement of 0 is 1, and the complement of 1 is 0. The same procedure holds as for decimal numbers. Notice that both numbers must have equal number of digits. This is shown in the following example.

Decimal	Binary	
15	1111	minuend
− 12	0011	complement of 1100
3	1 0010	sum
	→ 1	final carry
	11	difference

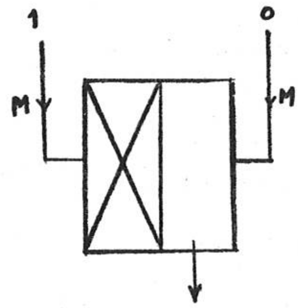
Division can be performed by successive subtraction. The arithmetic unit has a number of registers for storing the binary numbers on which it operates. The register can be formed by a series of flip-flop circuits. For a register to store a number of n bits, n flip-flops are needed (Fig. 2-28).

2.6. FLIP-FLOP CIRCUIT

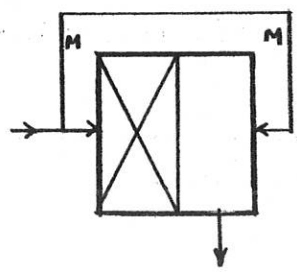
The flip-flop or bistable multivibrator (figure 2-29) consists of two identical stages, containing two almost identical triodes. Thus we have $R_{g1} = R_{g2}$, $R_{L1} = R_{L2}$, Symbol for a flip-flop

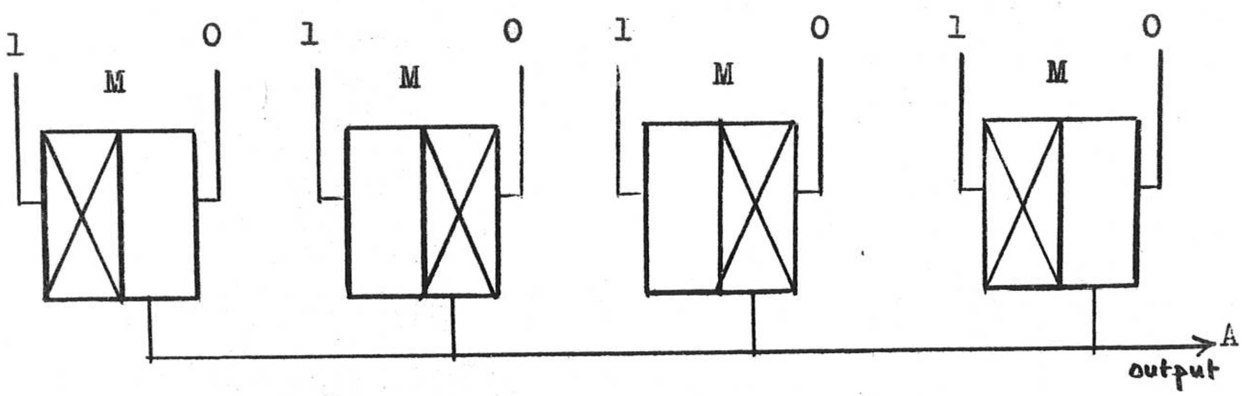
M and Mo , inputs

The crossed half is off.



Flip-flop as
binary counter





Register comprising four flip-flops

Stored binary number 0110

Fig. 2.28. Flip-flop Storage

$R_{c1} = R_{c2}$. The control grid of each triode is coupled to the anode of the other triode by a voltage divider consisting of R_c and R_g. The values of R_c, R_g and E_{cc} have been so chosen that when one triode is conducting with its grid at zero potential, the other triod's grid voltage given by R_c and R_g will cut off that triode. Assume that triode 1 is conducting, while triode 2 is cut off, and let a triggering negative voltage pulse be applied simultaneously to each grid. Since triode 2 is at cut off, then its grid becomes more negative. But for the conducting triode 1, the negative pulse at its grid tends to reduce I_{b1}, thus increasing E_{b1}. When E_{b1} is raised, the grid boltage of triode 2 is raised and triode 2 stárts to conduct, till triode 1 is cut off and triode 2 becomes conducting. In this way, each negative pulse applied to the two grids changes the state of operation.

A transistor flip-flop circuit using two p-n-p power transistors is shown in figure 2-30.

Whenever a triggering pulse is applied to its terminals, one of the transistors will conduct, while the other will cut off. The conducting transistor has negative base-emitter bias, while the cut off transistor has positive base-emitter bias (reverse bias) and is therefore not conducting.

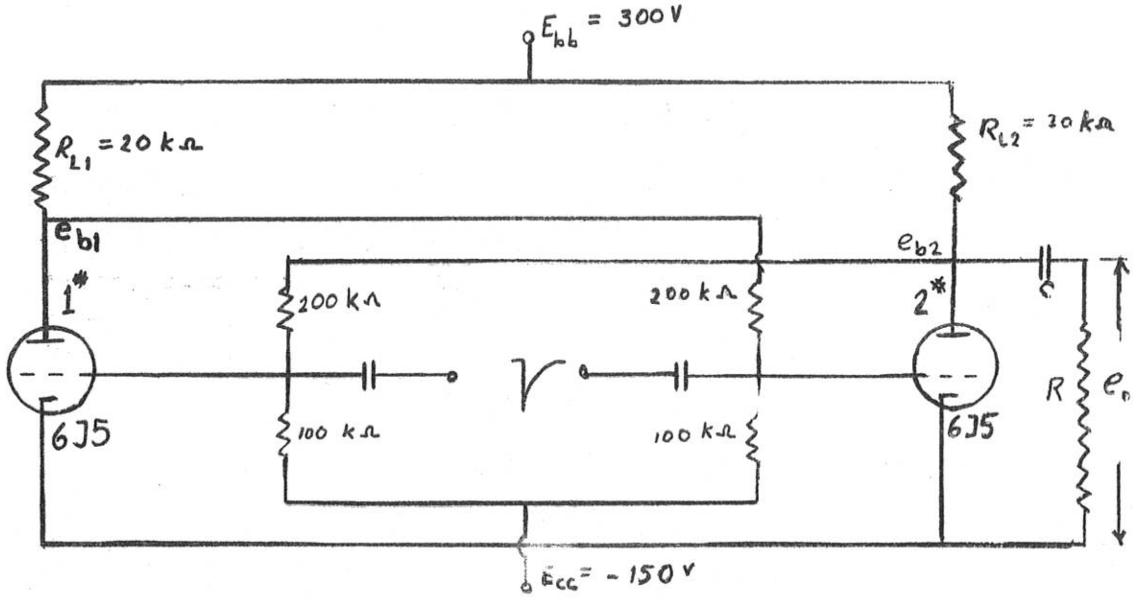


Fig. 2-29. Flip-flop tube circuit

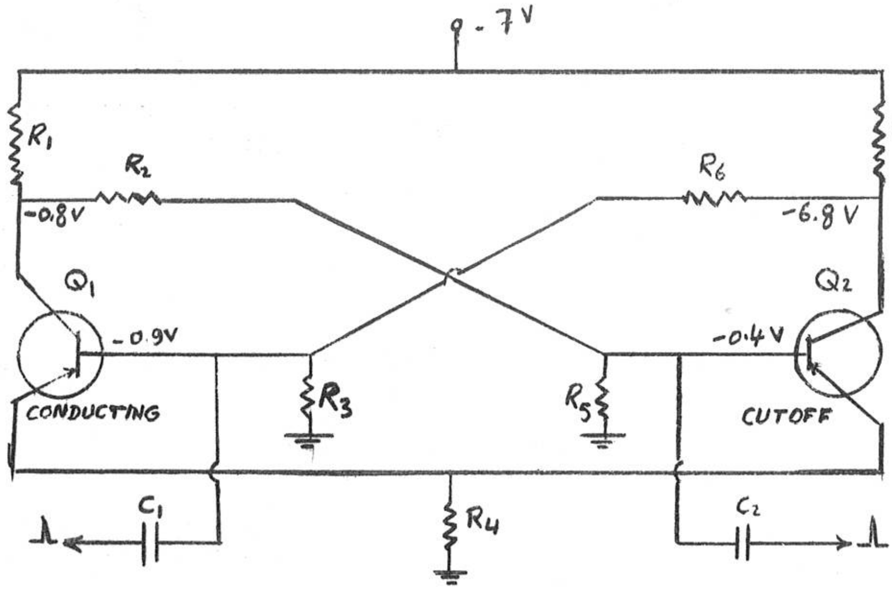


Fig. 2-30. Flip-flop transistor circuit

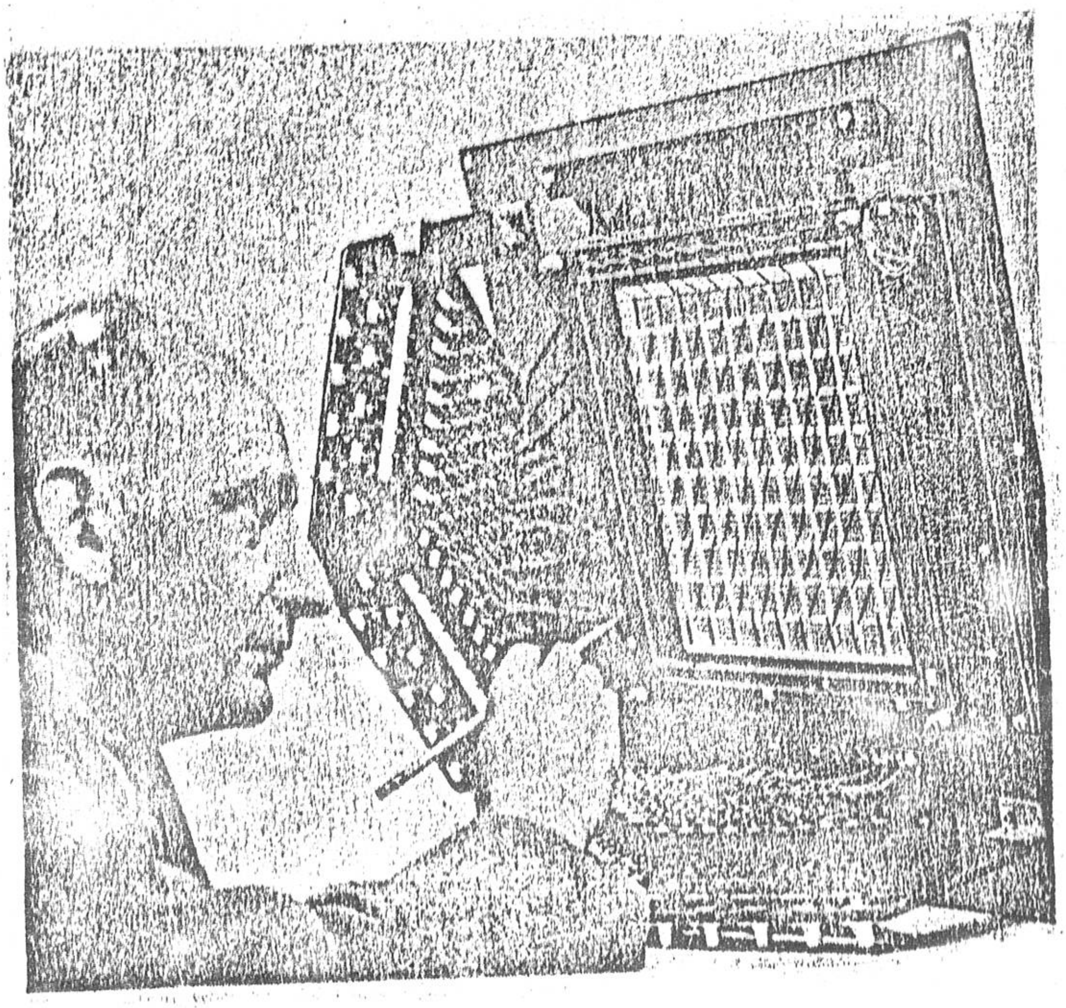
2.7. STORAGE

Although flip-flop circuits can be used for storage, yet they are quite expensive. While the first electronic digital computers used tubes for storage in the storage unit, they can not be used now due to the high cost and the huge numbers required.

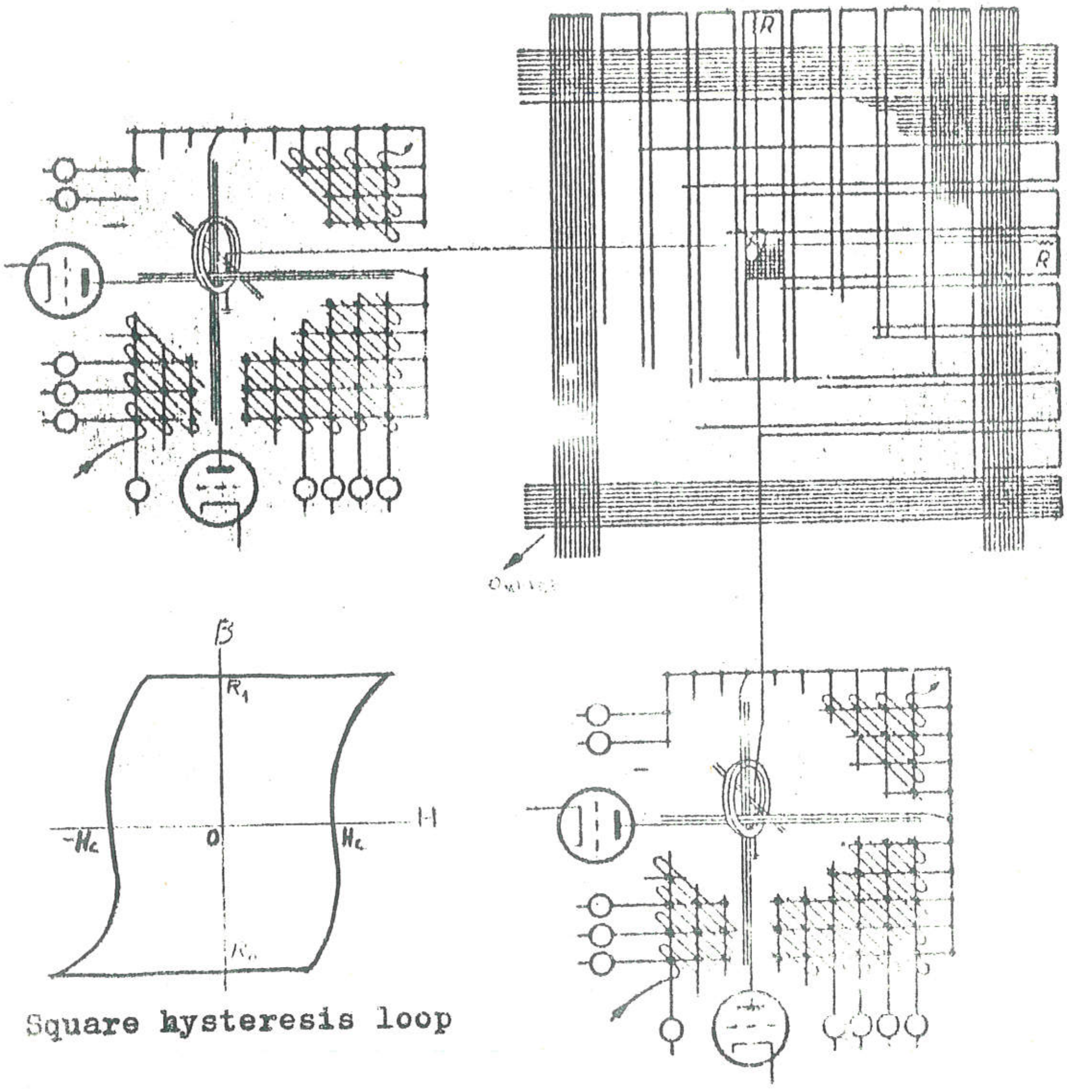
To-day, magnetic materials are mostly used for storage of considerable capacity in the form of magnetic tape, magnetic drum and magnetic cores.

2.8. MAGNETIC DRUM STORAGE

The magnetic drum is a further development for tape recording where powdered iron oxide as the magnetic materials is coated permanently on the outside surface of the drum. Each bit is stored on an area element about 0.01 inch circumferentially by 0.1 in axially, so that a drum of length 3 feet and diameter 3 feet can store about 4 million bits. A line of storage locations around the circomference of the drum is called a track or channel, which can store 1000 to 10000 bits, depending on the drum diameter. As the diameter of the drum increases, the storage cost per bit decreases, but the access time increases. The drum rotates at constant high speed past a group of heads. Associated with each track are one, two or three heads to read, erase and write on the drum surface, and still several read heads may be spaced around one track to decrease the access time.

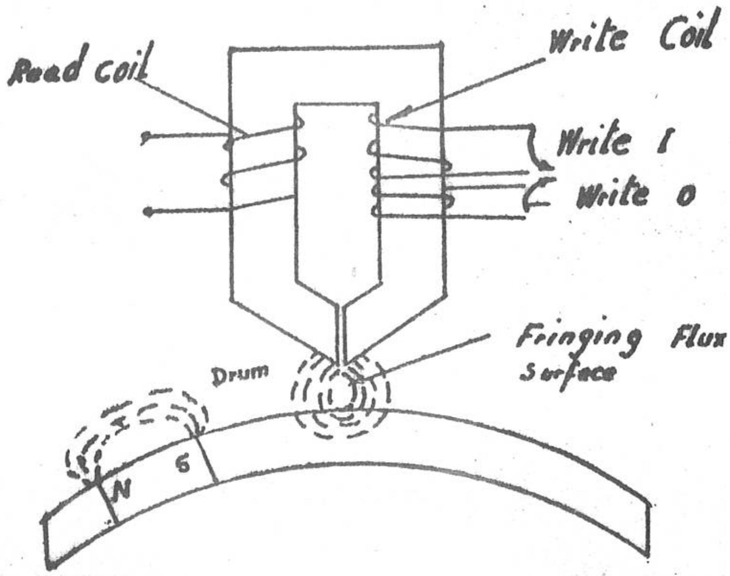


Dr. Rajchman of RCA examining
his 10000 core matrix, Fig. 2-32.

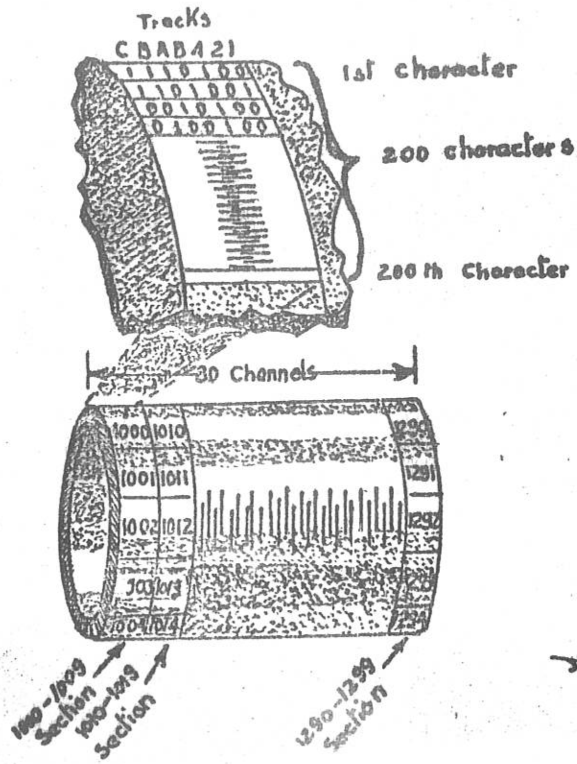


Square hysteresis loop

Fig. 2-32. 10000 magnetic core matrix and the associated 200 switshing circuits.



Drum Recording



ig. 2-31. Schematic, Drum Storage of IBM 705 Data Processing system

However, for practical commerical computers only one head per track is mostly used. The magnetic drum has nonvolatile storage with cyclic access time of a few milliseconds.

One bit of 1 or 0 is stored in each area element of a track by different magnetisations of the materials, for 1 the magnetisation being in one direction parallel to the direction of rotation, the 0-magnetisation being in the anti-parallel direction. If no magnetisation is required for 0, then erasing is done by alternating current.

The magnetic material should have a high coercive force and high remanence to obtain a satisfactory high reading level.

2.9. MAGNETIC CORE STORAGE

The storage element used in this type is a ferromagnetic ring of very small dimensions, for example the outer diameter may be of the order 1.3 to 2.2 mm. The outstanding property of these core rings is a hysteresis loop of almost square shape. The residual magnetic induction in such a square loop is essentially identical with the saturated induction. In the state R_1 it stores a "1", while in the state R_0 it stores a "0". Thus it can be used as a storage element. Due to the tiny size of the cores, they have small weight and take a rather small space, that is usually less than the associated control circuitry.

In each core of the whole core assembly pass two perpendicular wires. To write 1 into the core, half of the current needed to magnetise the core is sent through each wire, so that only the core at the intersection of these two wires is magnetised. No other core in one of the two wires can be magnetised at the same moment, since this half current is not suffecient for its magnetisation. The direction of the magnetising current pulse determines one of the two states : 1 or 0 which can be retained indefinetly.

In order to read the information stored in a core, a sense wire passes through the core. The sense wire is common for all cores in one plane. When it is required to determine whether a core has stored 0 or 1, we send a current through the horizontal and vertical wires so as to change or flip the selected core to the 0 state. If the core was originally in the 0 state, no change takes place in the core state and no current is induced in the sense wire. But if the core was in the 1 state, it changes or flop to the 0 state, inducing a current in the sense wire which is detected at the terminals of the wire.

Therefore, reading destroys the stored information in the core. This drawback is remedied by restoring back automatically all 1 state cores, while other cores of 0 state remain the same. This operation is carried out by a fourth wire that passes through every core in one plane, like the vertical wire. The complete operation of reading out the information in one or more cores and restoring that information or replacing it with new information (writing) is termed storage cycle. The access time for core storage is smaller than magentic drum storage and is equal for all storage locations. The core storage has great reliability and is of the random access type. It has also the advantage of longer life. It has been developed at at the Massachusettes Institute of Technology,USA.

2.10. CONTROL

The control unit of the digital computer consists of switching circuits as in the arithmetic unit. It contains one or more registers which are used to store instructions, beside other small registers which can store the adresses of one or more instructions to be performed subsequently. The control unit has switching circuits to decode the instructions, i.e. open or close (according to the instruction) paths to different locations in the storage unit, activate the proper circuits in the arithmetic unit, and control the input and output devices. Thus, the control unit directs the computer to perform the required operations on the given numbers according to the given sequence of instructions or program. The control unit has a master clock which is a source of standard timing signals for sequencing the computer operation.

2.11. COMPUTER CODE

For each basic operations, the digital computer should receive a specific instruction expressed as a combination, numerical and alphabetic. Such instructions are called the computer code. The program for solving a problem has to be written in the computer code, which is the language understood by the computer. Computer codes are not standard, but are generally simillar. Each instruction, expressed in the computer code, has associated with it the storage adress where the required data is stored. Some instruction perform arithmetic operations upon numbers stored in the storage. An important instruction is the test-for-minus command, which is the basis of decision making by the computer. The instruction instructs the computer to check the algebraic sign of the number in the arithmetic unit: if the sign is negative, the computer looks for its next instruction in a certain adress in storage while if the sign is positive, the computer proceeds with the next step in the program.

Another important instruction is the unconditional transfer instruction, which commands the computer unconditionally to go to the particular storage adress for its next instruction. The test-for-minus and the unconditional-transfer instructions give considerable flexibility in programming the digital computer.

The following table gives the instruction list of a hypothetical computer, with the alphabetic symbol and numeric code assigned for each operation described in the list (figure 35). According to this instruction list, a program will be prepared for solving a problem in the following example.

Figure 2-35. Typical Computer Instruction List

Command	Alphabetic symbol	Numeric Code	Operation Performed
Clear, add	CAD	01	The arithmetic unit is cleared and the number in specified storage adress is entered into the arithmetic unit.
Add	AD	02	The number in the specified storage adress is added to the number in the arithmetic unit, and the sum is retained in the arithmetic unit.
Subtract	SU	03	The number in the specified storage adress is subtracted from the number in the arithmetic unit, and the difference is retained in the arithmetic unit.

In our example we prepare the program of instructions to solve y_1, and use it again to compute y_2 by just substituting x_2 for x_1, and then compute y_3 by substituting x_3 for x_1. In fact, the program will be so arranged that after y_1 is computed, the program for computing y_1 is modified by the computer in order to compute y_2 and again modified to compute y_3.
When y_1, y_2 and y_3 have been obtained, the computer is instructed to print out the required results. The last instruction in the program will be the stop instruction, so that the computer can turn itself off.

USED STORAGE LOATIONS

The first step is to store the constants a and b into some arbitrary storage locations, e.g. 3000 and 3001, respectively. The given data x_1, x_2 and x_3 are stored in the locations 2000, 2001 and 2002 repsectively.

As mentioned before, we shall have to change our program for computing y_1 to get y_2, then change it once more to get y_3. For this purpose, some additional numbers will be needed. So we store the numbers 0001, 0001 and 0004 in the locations 3002, 3003 and 3004 respectively. For the computed results y_1, y_2 and y_3 we assign the storage locations 1000, 1001 and 1002 respectively.

Storage location	Quantity
1000	y_1
1001	y_2
1002	y_3
2000	x_1
2001	x_2
2002	x_3
3000	a
3001	b
3002	00 0001
3003	00 0001
3004	00 0004

Figure 2-36 Stored Quantities

The program instructions are stored in the location 000 to 0019 as shown in the following table.

Storage Location	Alphabetic Code	Numeric Code	Address
0000	CAD	01	3000
0001	MU	04	2000
0002	AD	02	3001
0003	ST	06	1000
0004	CAD	01	3002
0005	AD	02	3003
0006	ST	06	3002
0007	SU	03	3004
0008	TE	07	0013
0009	PR	08	1000
0010	PR	08	1001
0011	PR	08	1002
0012	STOP	10	0000
0013	CAD	01	0001
0014	AD	02	3003
0015	ST	06	0001
0016	CAD	01	0003
0017	AD	02	3003
0018	ST	06	0003
0019	UT	09	0000

Figure 2-37. Example Program-Storage Locations

OPERATION PROCEDURE

Once all the problem data and the coded program are stored in the assigned locations, the computer can proceed with the solution. It seeks its first instruction in storage location 000, then it seeks its second instruction in storage location 0001, and then in 0002, and so on, unless specifically instructed to do otherwise.

Let us assume that the computer is set into operation, and follow the operation procedure of the computer, step by step, according to the stored program.

The frist 4 commands cause the computer to calculate y_1 as follows:

01	3000	Brings(a) from storage location 3000 to the arithmetic unit, clears the arithmetic unit of any number remaining there from previous operations, and adds(a) to the arithmetic unit.
04	2000	Brings the contents of the storage location 2000, which is x_1, to the arithmetic unit and multiplies it with the contents of the arithmetic unit, a, retaining the product ax_1 in the arithmetic unit.

02	3001	Adds the contents of storage location 3001, which is the constant b, to the contents of the arithmetic unit, ax_1, retaining the sum $(ax_1 + b)$ in the arithmetic unit.
06	1000	Stores the contents of the arithmetic unit in the storage location 1000. Thus, $y_1 = ax_1 + b$ is now stored in location 1000.

Now that the computation of y_1 is finished, the computer obtains instructions from the program to compute y_2, then y_3. Specifically, the computer has to find out (by program instructions) whether or not all required values of y have been calculated. This is the decision-making ability of the computer.

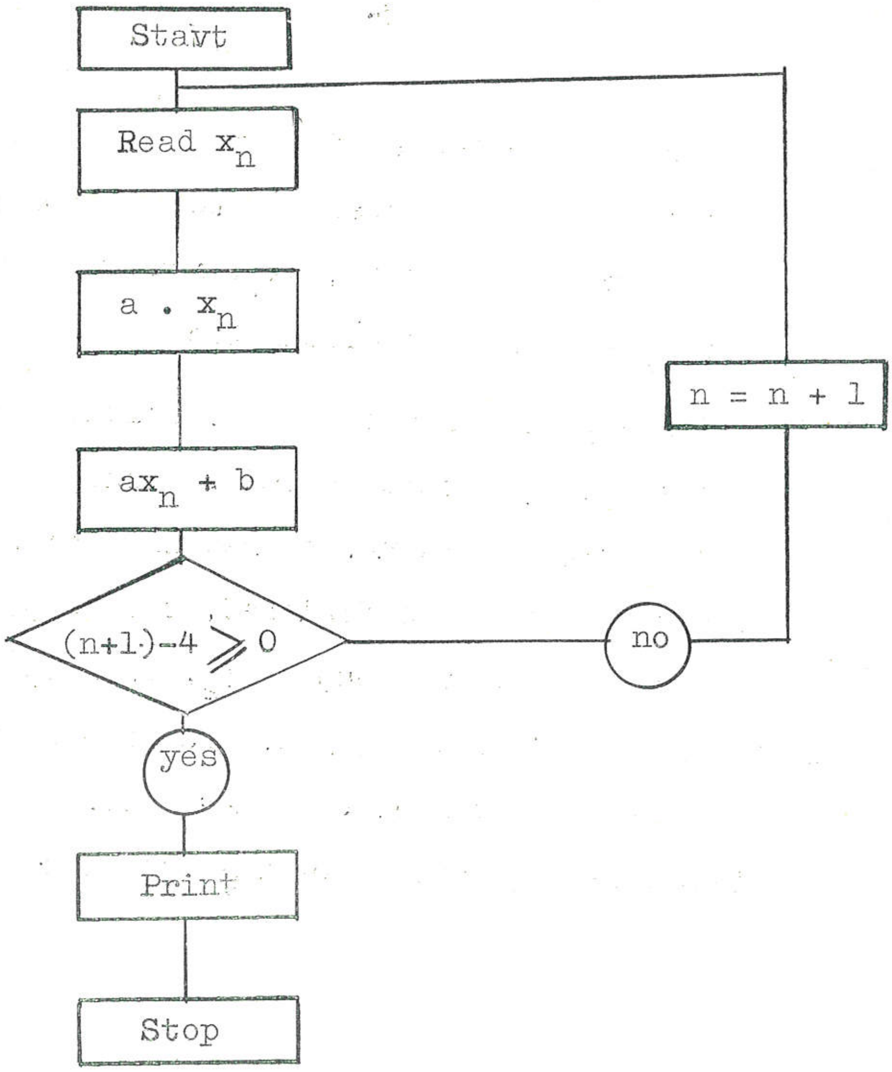


Figure 2-38. Flow Chart, Example Program

01 3002 Brings the number from storage location 3002, clears the arithmetic unit, and adds the number to the arithmetic unit. The number 00 0001 is now in the arithmetic unit.

02	3003	Brings the number from storage location 3003 and adds to it the number in the arithmetic unit. The number in the arithmetic unit is now 00 0002
06	3002	The contents of the arithmetic unit is now stored in storage location 3002 and also retained in the arithmetic unit. The contents of storage location 3002 has now been changed from 00 0001 to 00 0002, and 00 0002 is still held in the arithmetic unit.
03	3004	Subtracts the number in storage location 3004 (00 0004) from the contents of the arithmetic Unit (00 0002): the difference is negative.
07	0013	Tests for the presence of a negative number in the arithmetic unit. Since there is a negative number, control is transferred to storage location 0013.

Now that the computer has to compute another value of y, it shifts control to storage location 0013, where instruction for modifying the first four steps of the program begin and continue in sequence.

01   0001   Brings the number from storage location 0001, clears the arithmetic unit, and adds it to the arithmetic unit. The number brought from storage location 0001 is 04 2000, which is one of the instructions for calculating y.

02   3003   Brings the number from storage location 30003 (00 0001) and adds it to the number in the arithmetic unit. The number in the arithmetic unit is now 04 2001.

06   0001   Stores the number present in the arithmetic unit in storage location 0001. Thus the contents of storage location 0001 is now 04 20001.

The instructions have now been changed, so that when computing y,x is obtained from storage location 2001. Before the computer proceeds to compute y_2, the program has to be changed again in order to store the result y_2 in the storage location assigned for y_2. This is carried out as follow:

01   0003   Clears the arithmetic unit and enters the number from storage location 0003 (06 1000) into the arithmetic unit. In the arithmetic unit is now the storage location of y, which is 1000.

02	3003	Adds the number in storage location 3003 (00 0001) to the number in the arithmetic unit. The arithmetic unit now reads 06 1001, which is the instruction for storing y_2 in location 1001.
06	0003	Stores the number present in the arithmetic unit in storage location 0003.
09	0000	Control is uncoditionally transferred back to storage location 0000, and the computing cycle starts over.

The computer is now ready to compute y_2 by executing the first four commands of the program again. On this second cycle, y_2 is calculated according to the modified program, and its value is stored in storage location 1001. Then the test is made again to see whether or not three values of y have been computed. Since only two have been computed, the program is again modified and y_3 is computed and stored in storage location 1002. The test is made again to see whether or not three values of y have been computed. The test for negative is made from the commands stored in the storage locations 0004, 0005, 0006, 0007 and 0008. Before this last test, the number in storage location 3002 is 00 0003. When performing this last test, the number is increased by 0001 again to become 00 0004 in the arithmetic unit. Now, when subtracting the contents of storage location 3004 (00 0004) from the number 00 0004 in the arithmetic unit, the difference is not negative, but zero. The computer interprets zero as positive number, hence directed to continue in sequence to the instruction in storage location 0009.

The only remaining operations are to print out the results and then to stop.

08	1000	Prints out the contents of storage location 1000, which is y_1.
08	1001	Prints out the contents of storage location 1001, which is y_2.
08	1002	Prints out the contents of storage location 1002, which is y_3.
10	0000	Stops the computer.

In this way, the computer carries out the required computations according to the instructions given given in the prepared program, each instruction being represented by a number code specific for the computer used.

The demonestrated programming example shows the ability of the digital computer to decide between two alternatives, and how the computer can store the program instructions and modify them according to the program. Otherwise it could have been possible to program the computer to compute y_1, then independantly y_2 and also y_3, without modifying the original instruction of the program. But when we have to repeat the same computation hundr-eds of times, then it is a great advantage to modify each time the basic program by the computer itself. The development of modern digital computers started about the and of the second world war.

The principle of storing the program instructions and other data in the computer storage has been introduced by von Neumaun in 1945. The first computer of this type was built at the Cambridge University in 1949.

If the sequence of control signals is built into the computer, it becomes a special-purpose computer that can perform only the calculations of the problem defined by that sequence of operations.

2.13. FLOW-CHARTING

The sequence of operations on data as well as the logical steps in alternative paths of processing required to solve a problem are usually drawn in a block diagram, called flowchart. Figure 2 - 38 shows the flow-chart of the above example program. The actual coding of the problem i.e. programming is usually based on the flow-chart.

In general, the flow-chart indicates the steps in the solution procedure in terms of the basic arithmetic and logical processes of the digital computer:

1. Arithmetic
2. Transfer
3. Decision .

Figure 2-38 shows these processes in the flowchart of the example program. Each process or step is described in few words in enclosed boxes or rectangles. The flowchart emphasises the ability of the computer to repeat operations, called looping. Thus, if it is required to calculate 1000 values of y for given 1000 values of x in the example program, and to write the program to calculate each case seperately in turn the program would be large and time consuming to write; also the flowchart would be large. The rectangle or box enclosing the statement n = n+1 means replace the value of (n) by (n+1), which indicates the ability of the computer to modify its own program. The box (n+1)-4 $\geqslant$ 0 represents the decision-malking ability of the computer. In fact, the ability of the digital computer to repeat instructions (looping) combined with the facilities of modifying and skipping over instructions (transfer), makes a great reduction in the number of instructions required to perform a certain calculation.

In scientific or mathematical applications, it may sometimes not be necessary to draw up a detailed flowchart, since the procedure generally follows definite rules and may be kept clearly in mind. In commerical work, however, there are usually many exceptions, many accountancy checks, several procedures to be integrated, so that a detailed flowchart must be thoroughly prepared. This also promotes team work in vast organizations. The flowchart helps to prevent and detect errors in a proposed solution. It may lead to discover methods for the saving of time and labour. The flowchart makes the method of solution much more understandable to enyone who has to take part in the work or to review it. Flowcharting assists greatly to organize thinking about a complex procedure.

As revisions are made in the system, the flowchart has to be kept up to date.

2.14. INTERNAL INFORMATION REPRESENTATION

In commercial applications, a decimal number system is desirable because input and output are typically decimal. On the other hand, a binary number system is most suitable for computer design since it is easier to design and build digital equipment with electronic two-state devices.

One way to solve this problem is to keep input and output in decimal number system, and to design the computer to operate with pure binary numbers. However, this requires converting input from decimal to binary and output back to decimal. The computer is usually programmed to carry out the number system conversion.

In commerical applications, however, the amount of processing (i.e. arithmetic operations) is comparalively small compared with input and output volumes. This makes the number system conversion unfavourable. Another way to solve this problem is in the form of a compromise. Each decimal digit is sperately represented in a binary code, thus avoiding the task of converting the number as a whole into binary . The obtained numbers are called binary-coded decimal numbers, BCD. However, this greater ease of decimal-binary conversion is gained at the cost of longer numbers, since binary-coded decimal numbers are longer than the corresponding pure binary numbers. Thus, in processing binary-coded decimal numbers, more circuitry is required. Each individual decimal digit is represented by 4 binary digits, while to represent one decimal digit in pure binary form, only 3.33 bits or binary digits are needed (information content per decimal digit (out of the 10 decimal digits 0,1,2,3,....9 is = $\log_2 10 = 3.33$ bits). This shows that the pure binary system will result into a computer circuitry more economic by the factor. 0.825 that in the case of using a binary-coded decimal system,

Now, four bits give sixteen different bit combinations ($2^4 = 16$), but only ten are needed for binary coding of the decimal digits 0,1,2,3,....9. Therefore many different binary-coded decimal schemes may be used. One called "8421" derives from the value assigned to each bit position as shown below.

Decimal Digit	B C D
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

As an example, the decimal number 5497.658 in BCD becomes:

Decimal	5	4	9	7	.	6	5	8
B C D	0101	0100	1001	0111	.	0110	0101	1000

Now, the binary number equivalent to $(5497.658)_{10}$ is $(1010101111101.101010001)_2$ which contains 22 bits, while the same number in binary-coded decimal contains 28 bits. The binary-coded decimal system is preferred in commercial computers, while the pure binary system is preferred in scientific computers.

Information held into the computer as problem data or program instructions is thus represented in binary form, and this as electric signals having two conditions, pulse or no pulse. This is in principle simillar to the telegraph code consisting of combinations of marks (current)and spaces (no current) representing letters and numerals. Thus, in a digital computer, numerals and characters (letters, special signs, ... etc.) are represented by combinations of pulses or 1 and 0 digits as a binary code.

Digital computers differ in the method of representing information in binary code. A digital computer may either use an alphanumeric code i.e. representing decimal digits and letters or a numeric code, the letters being represented by two-digit combinations. A computer in which each storage location can hold one numeric or one alphabetic character is called "character machine".

Most computer codes are self-checking, so that any fault in a code combination is detected automatically by the computer. For this purpose an extra bit (digit 1 or digit 0) is added to each code combination, called the parity bit, in order to make the sum of "1" bits in a character code combination equal to an odd or even number, as specified. If a parity bit is generated by the machine so as to make the total number of the 1 bits in any code combination equal to an odd number. This is called odd parity. Even party may be also used.

Parity bits are generated as characters(or numerals) enter the storage unit of the computer. Parity is then automatically checked as characters are subsequently

a) moved within the storage unit

b) extracted from the storage unit to be transfered to the arithmetic unit.

c) extracted from the storage unit for output via a peripheral device.

THE SIX-BIT NUMERIC CODE

This code is essentially a numeric code, i.e. represents decimal digits in binary form. To represent letters and special characters, two digits are used. Of the six bits, 4 bits represent a decimal digit in binary form. The four bit positions have the binary values 1,2,4,8 respectively. The fifth bit is called the flag bit, F. Its presence (1) denotes + sign, while its asbscence (0) denotes - sign. The sixth bit is a redundaut bit (C) for odd parity checking. It is present (bit 1) only if the other positions contain an even number of bits.

An example for representing letters by this numeric code is shown below. The letter A is shown represented by code of decimal digits 4 and 1.

Digit 4	Digit 1	
0 0 0 1 0 0	0 0 0 0 0 1	Code bits
C F 8 4 2 1	C F 8 4 2 1	value

Fig.2-39. Numeric Code, letter A

Decimal digit	C	F	8	4	2	1
0	1	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	1	0	0	0	1	1
4	0	0	0	1	0	0
5	1	0	0	1	0	1
6	1	0	0	1	1	0
7	0	0	0	1	1	1
8	0	0	1	0	0	0
9	1	0	1	0	0	1

Fig.2-4. Six-bit numeric code

SEVEN-BIT ALPHANUMERIC CODE

In the six bit numeric code, only four bits are used to record information. This gives $2^4 = 16$ different code combination, which just give 10 combinations for the ten decimal digits 0,1,2,....,9, while letters have to be recorded as two-digit combinations. There may be a lot of waste or uneconomic use of storage locations.

If we use a 6-unit code, the number of possible (different) code combinations becomes 64, allowing to represent decimal digits, letters as well as special signs. The six bits are divided into 4 numeric bits of binary position values 1,2,4 and 8 respectively, and two zone bits. For decimal digits, the two zone bits are both 0 bits, while zone combinations 10, 01 and 11 are used with numeric combinations to represent letters and special signs. To the 6 bits a redundant seventh bit is added for parity checking. An example of the 7-bit alphameric or character code is shown in figure 2-41 .

MULTI-MODE WORDS

"Word" is defined as a fixed number of characters that are treated as a unit. Word length is fixed by the computer designer and incorporated in the circuitry. Common word lengths are 10 or 12 decimal digits, and 24, 36 or 48 bits. A word may also represent numbers in pure binary code.

In the early scientific computers large words having 32 to 39 bits were used. For scientific work it was necessary to hold large numbers in binary form, e.g. for word length of 39 bits, the most significant bit would have the value of 2^{38} according to the position of the binary point; the corresponding decimal values are

Character Description	Printed Symbol	MACHINE CODE							Character Description	Printed Symbol	MACHINE CODE						
		C	ZONE		NUMERIC						C	ZONE		NUMERIC			
			A	B	2^3	2^2	2^1	2^0				A	B	2^3	2^2	2^1	2^0
Zero	0	1	0	0	0	0	0	0	Colon	:	1	0	1	1	1	0	1
One	1	0	0	0	0	0	0	1	Apostrophe	,	1	0	1	1	1	1	0
Two	2	0	0	0	0	0	1	0	Plus zero	+0	0	0	1	1	1	1	1
Three	3	1	0	0	0	0	1	1	Minus	-	0	1	0	0	0	0	0
Four	4	0	0	0	0	1	0	0	J	J	1	1	0	0	0	0	1
Five	5	1	0	0	0	1	0	1	K	K	1	1	0	0	0	1	0
Six	6	1	0	0	0	1	1	0	L	L	0	1	0	0	0	1	1
Seven	7	0	0	0	0	1	1	1	M	M	1	1	0	0	1	0	0
Eight	8	0	0	0	1	0	0	0	N	N	0	1	0	0	1	0	1
Since	9	1	0	0	1	0	0	1	O	O	0	1	0	0	1	1	0
Space		1	0	0	1	0	1	0	P	P	1	1	0	0	1	1	1
Quarter	¼	0	0	0	1	0	1	1	Q	Q	1	1	0	1	0	0	0
At the rate of	@	1	0	0	1	1	0	0	R	R	0	1	0	1	0	0	1
Open Parenthesis	(	0	0	0	1	1	0	1	Pound	£	1	1	0	1	0	1	1
Close Parenthesis	)	0	0	0	1	1	1	0	Asterisk	*	0	1	0	1	1	0	0
		1	0	0	1	1	1	1			0	1	0	1	1	1	1
Ampersand	&	0	0	1	0	0	0	0	Half	½	1	1	1	0	0	0	0
A	A	1	0	1	0	0	0	1	Oblique	/	0	1	1	0	0	0	1
B	B	1	0	1	0	0	1	0	S	S	0	1	1	0	0	1	0
C	C	0	0	1	0	0	1	1	T	T	1	1	1	0	0	1	1
D	D	1	0	1	0	1	0	0	U	U	0	1	1	0	1	0	0
E	E	0	0	1	0	1	0	1	V	V	1	1	1	0	1	0	1
F	F	0	0	1	0	1	1	0	W	W	1	1	1	0	1	1	0
G	G	1	0	1	0	1	1	1	X	X	0	1	1	0	1	1	1
H	H	1	0	1	1	0	0	0	Y	Y	0	1	1	1	0	0	0
I	I	0	0	1	1	0	0	1	Z	Z	1	1	1	1	0	0	1
Plus	+	0	0	1	1	0	1	0	Comma	,	0	1	1	1	0	1	1
Period	.	1	0	1	1	0	1	1	Per cent	%	1	1	1	1	1	0	0
Semicolon	;	0	0	1	1	1	0	0	Equals	=	0	1	1	1	1	1	0

Fig. 2-41, Alphanumeric or Character Code

274,877,906,944 or

0.0000000000036 37978807091713

Such figures are usually too high to be used in normal commercial practice. Modern commerical high power word computers have a smaller word length, eg. of 24 bits. One parity bit can be used with the 24 bits. The 24:bits treated as one word may be used as 6 decimal characters or 4 alphanumeric characters. These modes are alternative to using the 24 bits as one binary number, which is 10963812. A computer having its storage divided into words is called a word machine.

2.15. FLOATING POINT ARITHMETIC

After the digits representing a number have been determined, it is necessary to indicate the location of the decimal point, or in a binary computer, the binary point. If this location is the same for all number words of the storage unit, the computer is said to operate with a fixed point. This is similar to calculation by desk calculators or slide rules, with which the operator must keep track of the decimal point. Thus, in case of a fixed point computer, the location of the radix point is the programmer's responsibility. He can keep track of decimal points in arithmetical operations for numbers that are within the range of computer's word capacity; e.g. 0.0000001 to 9999999. Overflow arises when numbers are too large to fit into storage, which necessiates scaling of the operands before calculation. Thus tracking the location of the deciaml point is difficult and requires extra programming.

Another method of calculation called floating point can automatically account for the location of the decimal (or binary) point. This is usually accomplished by handling the number as a signed mantissa or fraction times the radix raised to an integral exponent. For example, the decimal number +88.3 might be written as $+ 0.883 \times 10^2$; the binary number +0.0011 as $+0.11 \times 2^{-2}$. One group of special instructions available in many computers deals with floating point arithmetic, which automatically handles decimal points in arithmetical operations.

A computer equipped with circuitry (hardware) for floating point operations lines up floating point numbers (so that exponents are equal) before addition or subtraction. In multiplication, the exponents are added. In division, the exponent of the divisor is subtracted from the exponent of the dividend. The product or quotient is adjusted to a fraction and the appropriate exponent. To avoid negative exponents in the computer word, common practice is to add the exponent to some positive base, e.g. 50. Examples of numbers are:

Ordinary Number	Floating Point mantissa	Floating Point Exponent	Computer word mantissa	Computer word Exponent
+ 5327.	+.5327	+ 4	+ 53270	54
- 368592.	-.368592	+ 6	- 36859	56
+ 0.4375	+.4375	0	+ 43750	50
- 0.00298	-.298	- 2	- 29800	48

It may be seen that the time required for addition and subtraction is much longer in the floating than in the fixed point arithmetic. The time required for multiplication, and similarly for division, is not so much affected. If floating point operations are built-in (hardware), some of the steps can be carried out simultaneously on paralell equipment, which shortens very considerably the time needed.

The choice of the word length in a word machine depends on choosing either fixed or floating point arithmetic. This is due to the fact that the length of the numbers depends on the effect of rounding errors. The rounding errors comitted in the millions of operations which make up a problem accumulate to such an extent that frequently three or more decimal places are lost in the course of computation. Therefore, the machine program must allow for the worst possible case, i.e. for the largest-possible number that can occur in such a series, and thus part of the space reserved for a number remains unused most of the time in fixed point calculation. However, this condition is absent in machines with floating point arithmetic, in which case a somewhat shorter word length is adequate. Thus, while a word length between 11 and 13 decimal digits is optimal for a fixed-point machine, 10 or 11 decimal digits are optimal word length for floating-point machines.

Floating point operations, if provided for the hypothetical computer discussed in this chapter, could be defined as follows:

Code	Operation Performed
FAD y x	Floating point add the contents of location x to the number in the arithmetic unit, content of storage location x is unchanged. Both numbers must be in the floating point representation.

Floating point subtraction, multiplication and division can be similarly defined.

In scientific computation, floating point operation is required in many problems. If the computer is built for fixed point operation, and floating point operation is accomplished by coding, a very considerable loss in efficiency results by a factor between 5 and 50 (depending on the computer design and code), with an increase in storage requirements. A computer with built-in (hardware) floating operation avoids these drawbacks, but requires more equipment and a more complicated design of the arithmetic unit, and consequently higher initial and maintenance cost.

In most commerical applications, fixed point arithmetic is usually preferable, since the arithmetic operations are much simpler than in scientific applications, and because of the high volume of input and output, this would also require fixed to floating point conversion and vice versa at input and output respectively.

Computers with a built-in floating point feature also perform arithmetical operations in fixed point ordinary arithmetic. Most computers place the fixed point at the left end of the number, so that all numbers in the machine are less than unity in absolute value. Some computers place the fixed point one decimal place (or 2 binary places) from the left end of the number, so that constants like 1,2,3, π , e , can be represented without scaling (Lubkin 1948).

2.16. SUBROUTINES ; SOFTWARE

In our discussion of floating point feature, two alternatives were discussed:

(a) built-in floating point feature, called hardware i.e. as integral part of the circuits and physical units from which the computer is built;

(b) coding a sequence of instructions in machine language to carry out the necessary floating point operations. since this will result into a self-contained program section, it can be pre-prepared to be used repeatedly and hence usually called "subroutine". Usually subroutines are designated as software to differentiate them from the computer hardware.

A subroutine can be entered from any point in the main program and is so constructed that, when the subroutine has been excuted, a return jump (unconditional transfer) is automatically made to the instruction immediately following the jump which entered the subroutine.

The subroutine may be required at several different points in the main program. Thus, it is stored once and jumping is made to that section or subroutine whenever it is required in the main program. This will save much storage space. Using a subroutine will save programming time as well as testing time.

Use of subroutines in common in scientific and engineering work that involves computing square routes, cubic routes, sin es consines and tangents, as well as for division. Sometimes the computer running time may be longer for a program built of subroutines than for a program especially prepared for the problem. Yet the cost of longer running time on the computer may be outweighted by reduced programming cost and shorter time required to prepare programs.

Subroutines are also repeatedly used in commercial applications, where it is more important to arrange for efficient input, output and machine utilization than to minimize the cost of programming. Some parts of commercial programs such as input, output, edditing and sorting may be handled by subroutines, for example in computing payrolls or in inventory control.

Beside subroutines as software facility supplied by computer firms, programs and routines are also supplied. These may be defined as follows:

PROGRAM: The complete sequence of instructions for a job to be performed on the computer. The program may comprise one or more computer runs.

ROUTINE: Any part of a program which deals with a particular aspect of the overall procedure.

There is a library of general purpose programs, routines and subroutines written for specific types of computers which are supplied by the computer firms. Each program, routine or subroutine is classified and numbered and has a specification sheet that contains sufficient information to enable the user to make the appropriate choice and is carefully planned to economize in the use of storage space and computer time. A library subroutine complies with a standard format, so that a complete subroutine usually comprises:

1) Specification sheet,
2) Flowchart,
3) Program sheets
4) Program card pack or paper tape.

Sofware services supplied by computer firms may also include:

UTILITY ROUTINES : available for repetitive data-handling procedures such as sorting or transfer of data from working storage to a more permenent area or medium (dumping).

COMPILERS : routines that before the disired computation is started convert a relatively machine-independant source program into a machine language program, e.g. Fortran compiler.

This shows the prime importance of the software facilities that computer firms give to the customer as aids for scientific as well as for commercial applications. These positively render the computer system more powerful, increase the variety of application , contribute greatly for economic operation, as well as in running and maintenance.

OPEN & CLOSED SUBROUTINES

If the subroutine is simply copied on the main program whenever it is needed, it is called open subroutine.

If the subroutine only appears once in the program, and is to be used any number of times, it is called closed subroutine. It may be placed anywhere in the store, and is called into use by a transfer instruction. Similarly, control is returned to the main program by another transfer instruction.

A complete program usually consists of a main program and several subroutines, some drawn from the library and some made specially for the particular job.

For dealing with the numeric data of the problem, the program has to include at least one number-reading subroutine as well as one output subroutine.

2.17. SERIAL AND PARALLEL OPERATION

Mode of operation refers to the way that bits, characters and words are read in or out of storage, and processed in the arithmetic unit. In serial operation, data in the form of bits characters and words are read one after another in a time sequence. In parallel operation, all bits are read simultaneously.

In a serial computer, the numbers to added are added one position after the other : first the units, then the tens, then the hundreds...etc. Each time, the carry is added to the next higher-order position. Therefore, the time required for serial operation depends on the number of digits in the quantities to be added. In a parallel computer, addition is performed on complete data words. Any two data words, regardless of the quantities in each word, can be added including carries at the same time, the difference, between the two modes of operation is shown by the following example.

Serial Addition

	Step 1	Step 2	Step 3	Step 4
First quantity	3728	3728	3728	3728
Second quantity	2115	2115	2115	2115
	1	1		
	3	53	853	5853

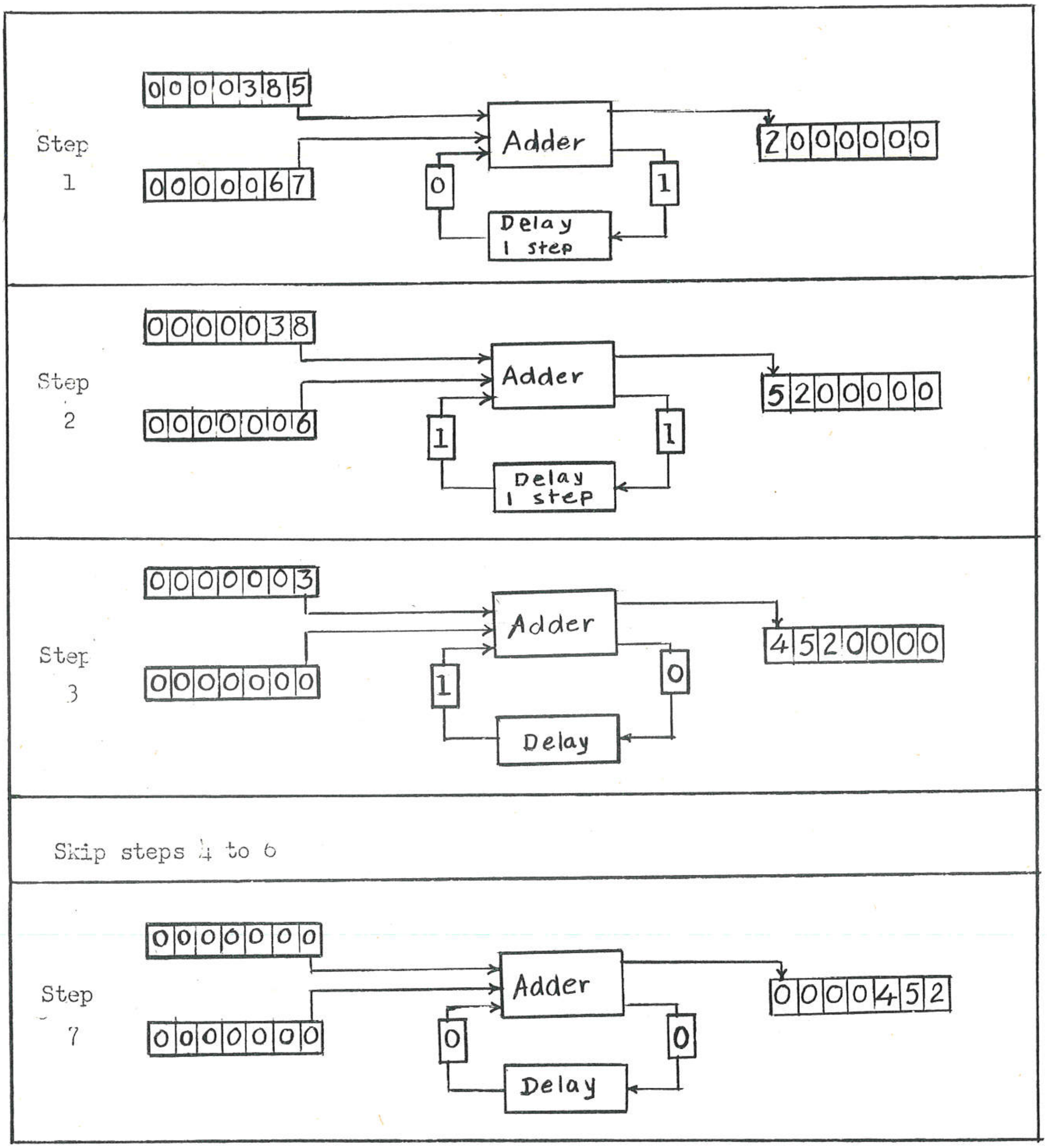


Fig. 2-42. Serial adder
Ex: 385 + 67 = 452

Parallel Addition

First quantity	0 0 3 7 2 8
Second quantity	0 0 2 1 1 5
Carry	1
Sum	0 0 5 8 5 3

Figure 2.42 shows the operation of serial adder. For simplicity, decimal numbers are used, while in the actual computer binary digits are employed. The right-hand digits of two operands are added to make one digit in the sum. Any carry is delayed and added in the next step. The operands are shifred one digit to the right after each step so that the next digits are obtained from the same positions as before. The sum is also shifted one position to the right after each step, and the new digit is introduced at the same point each time. Seven steps are necessary to add 2 seven-digit numbers, of which only the first three are shown in figure. The sum will, at the end of the operation, be positioned just as the operands were.

In actual computers, this method is simplified by putting the sum into the vacant positions, where one of the operands was. In the serial mode, the digits from the first column are added during one time interval. Any carry is added with digits from the second column during the next time interval. The operation time thus depends on the number of digit positions involved. Only one adder is required for the serial mode, regardless of the number of digits in the number.

In case of parallel adders, all digits of a number are handled at one time, by providing an adder for each digit position of a number. The faster speer speed of parallel operation is obtained at the expense of more components and ingenious circuitry.

2.18. OPERATIONS IN THE ARITHMETIC UNIT

To carry out the operations of additions, subtraction, multiplication and division, the arithmetic unit has registers, each capable of storing one computer word i.e. a binary number having n bits, which will hold the sum of previous numbers, transfers this sum to an adder for addition to another binary number and to receive and store the sum of this addition until it is required for the next operation or is deared. When such a register is combined with an adder, the whole is known as an accumulator, fig. 2-43.

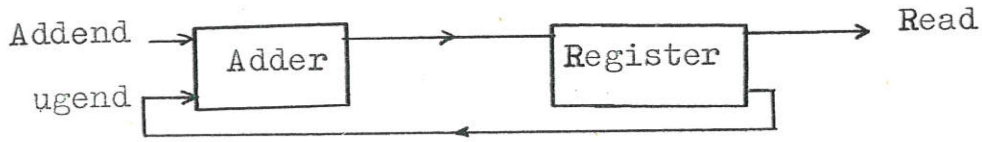


Fig. 2-43 The accumulator

In case of parallel operation, the accumulator has for each binary position its own adder, thus reducing the time of processing to a great extent.

Digital computers are usually so designed that all the n digits of a number stored in a register can be shifted one place in either direction in a sigle operation.

Now, in decimal arithmetic, multiphying a number by 10 the base means adding one decimal position more to the number e.g. 54 x 10 = 540 i.e. shifting the number to the left by one position. Similary, in binary arithmetic, where the base is 2, multiplication with $(10)_2$ means adding one binary position more to the number, e.g.

$$(101)_2 \times (10)_2 = (1010)_2$$

This is equivalent to moving or shifting the binary number 101 into the next more significant binary position thus becoming 1010. Thus, a left shift of one position is to multiply the binary number by 2, and a shift of k positions to the left multiplies the binary number with 2^k.

We now carry out a simillar analogy for division. In decimal arithmetic, dividing by 10 , the base, means decreasing the number of decimal positions by the least significant, e.g.

250/10 = 25 , i.e. shifting the number to the right by one positic Similarly, in binary arithmetic, where the base is 2, division by $(10)_2$ means decreasing one binary position from the number, which is the least significant one. For example,

$$(1110)_2 \ / \ (10)_2 \ = \ (111)_2$$

Thus, a right shift of one position is to divide the binary number by 2.

In order to give examples of arithmetic operations, we consider here the arithmetic unit of a fictitions computer, which cosists of an accumulator AC, and a multiplier-quotient register, MQ. The AC register has 16 bits. The first bit is the sign bit, the next bit the overflow bit, and the remaining bits may contain a binary number. The digit 1 will appear in the overflow bit if through an arithmetic calculation a number is obtained which is larger than the capacity of the computer register. The primary use of the AC is in addition and subtraction. It is also used for other operations. The MQ is a 15-bit register which is used with the AC for multiplication and division operations. We shall consider here floating-point numbers to be stored in the form

(sign)(characteristic) (fraction)

where characteristic = exponant + $(10000)_2$ For example, $(0.101\ 010\ 111)_2 \times 2^5$ stored as 10101 101010111

character. fraction

ADDITION

Integer addition is carried out in two steps in the AC. Suppose that the binary numbers 101 111 and 010 are to be added. The first step is to place one of the 2 numbers in the AC.

		AC				
0	0	00	000	000	101	111

The next step is to add the other number. The result is left in the AC.

		AC				
0	0	00	000	000	110	001

In the addition of floating-point numbers, the normalized form of one of the numbers must be altered so that the binary points be alterned so that the binary points line up. Consider the addition of 0.100×2^2 and $0.101\ 111\ x^{-3}$. We first place the first number in the AC.

		AC				
0	0	10	010	100	000	000

The second number must be changed to:

$$0.000\ 001\ 011\ 11 \times 2^2$$

We now add the fractional part to the fractions already in the AC, after dropping the last two digits 11 as insignificant.

AC

0 0 10 010 100 001 011

Under circomstances, an addition of two binary numbers may give an overflow from bit 8. For example,

$$\begin{array}{l} 0 \, . \, 101 \times 10^2 \\ 0 \, . \, 110 \times 10^2 \\ \hline 1 \, . \, 011 \times 10^2 \end{array}$$

In such a case, a shift to the right is made and the digit 1 is added to the characteristic. The result would appear in the AC as follows:

AC

0 0 10 011 101 100 000

SUBTRACTION

The process of subtraction is reduced to compliment addition. Consider the subtraction of the two binary numbers: 010 from 101 111. We first place 101 111 in the AC.

AC

0 0 00 000 000 101 111

The compliment of 010 is

11 111 111 111 101

Adding we get

			AC							MQ		
0	0	00	000	000	000	111	‖	010	000	000	000	101

The partial sum 111 appears as the three digits 111 in the AC and the second digit 1 in the MQ. Thus, in each shift the partial sum is shifted one position from the AC into the MQ, till at the end the final product appears in the MQ, and the contents of the AC becomes all zeros.

Shift to the right:

			AC						MQ			
0	0	00	000	000	000	011	‖	011	000	000	000	010

Since the rightmost digit in MQ is zero, the partial sum is unchanged. Next follows a shift:

			AC						MQ			
0	0	00	000	000	000	001	‖	011	100	000	000	001

Now, the rightmost digit in MQ is 1, so 101 is added to AC

			AC						MQ			
0	0	00	000	000	000	110	‖	011	100	000	000	001

Next follows a shift

			AC						MQ			
0	0	00	000	000	000	011	‖	001	110	000	000	000

		AC			
01	00	000	000	101	100

Now, putting zero in the second bit, and adding 1, we get finally the the result in AC.

		AC			
00	00	000	000	101	101

MULTIPLICATION

Before proceeding with multiplication using the arithmetic unit, let us consider the steps involved as in paper and pencil calculation. As an example of binary multiplication, we may consider the multiplication of 101 by 1011.

			Partial Products
Multiplicand	101		
Multiplier	1011		
	101	=	101 x 1 , no shift
	1010	=	101 x 10 , 1 shift
	00000	=	0 x 100 , 2 shift
	101000	=	101 x 1000, 3 shift
Add:Result =	110111		

The multiplication process consists of a sequence of additions and shifting operations, each partial product consisting either of the multiplicand shifted up a number of positions, or the number zero, depending on whether the corresponding binary digit of the multiplier is 1 or 0. In the arithmetic unit the partial products are usually added, one after the other, into the accumulator register. Multiplication of two n-digit binary numbers requires (n-1) shifts and (n-1) additions. These operations will be carried out in the arithmetic unit in steps: one shift followed by one additions, then one shift followed by one additions till all the 2(n-1) operations are completed. In each step. The partial sum (which is initialy the multipliand is shifted one digit to the right, then addition follows.

For multiplication in the arithmetic unit, the AC and MQ are used. Consider the multiplication of the two binary numbers: 001 011 by 101. The first step is to place 001 011 in MQ, and clear AC.

AC								MQ				
0	0	00	000	000	000	000	‖	000	000	000	001	011

The computer checks the first digit from the right position value 2°, shift) Since this digit is 1, the second number 101 is added to the AC.

AC								MQ				
0	0	00	000	000	000	101	‖	000	000	000	001	011

After addition follows one shift (i.e. multiplication). All digits in AC and MQ are shifted to the right one place except for the sign bit in MQ.

AC								MQ				
0	0	00	000	000	000	010	‖	010	000	000	000	101

Again we have a digit 1 in the rightmost of the MQ; so 101 is added to AC

AC								MQ				
0	0	00	000	000	000	111	‖	010	000	000	000	101

The partial sum 111 1 appears as 111 in the AC and 1 in the MQ. Thus, in each shift, the partial sum is shifted one position from the AC into the MQ, till at the end the final product appears in the MQ, and the contents of the AC becomes all zeros.

Shift to the right :

AC								MQ				
0	0	00	000	000	000	011	‖	011	000	000	000	010

Since the rightmost digit in MQ is zero, the partial sum is unchanged. Next follows a shift:

AC								MQ				
0	0	00	000	000	000	001	‖	011	100	000	000	001

Now, the rightmost digit in MQ is 1, so 101 is added to AC

AC								MQ				
0	0	00	000	000	000	110	‖	011	100	000	000	001

Next follows a shift

AC								MQ				
0	0	00	000	000	000	011	‖	001	110	000	000	000

This continues until a total of 14 shift take place. Indeed, after the remaining 10 shifts, the product appears in MQ as the final result.

AC								MQ				
0	0	00	000	000	000	000	‖	000	000	000	110	111

In case the product has more than 14 digits, an overflow appears into the rightmost bits of the AC, and its contents will not be zero.

Floating-point multiplication can be carried out by applying the same procedure to the fractions of the two numbers. Also, the exponents (not the characteristics) must be added. For example, multiplying 0.100 x 2^2 by 1.01 111 x2^{-3} gives 0.010 111 x 2^{-1}. The result in nomalized form 0.101 111 x 2^{-2} will appear in the MQ register.

MQ

001 110 101 111 000

Another approach to multiplication is by forming a multiplication table by adding the number to itself a given number of times and storing each total. An alternative is:form multiples of 0 through 9 times the multiplicand. The last method is more elaborate but faster..

DIVISION:

Simillar to multiplication, division is carried out in the arithmetic unit using the AC and the MQ. Consider the division of 001 011 by 101. We start by placing 001 011 in the MQ.

AC								MQ				
0	0	00	000	000	000	000	‖	000	000	000	001	011

Shift to the left one position, skipping the sigh bit in the MQ.

AC								MQ				
0	0	00	000	000	000	000	‖	000	000	000	010	110

The number in the AC is now compared with the divisor. Since this number is less than the divisor, another shift to the left takes place. In this example, this process continues until a total of 13 shifts has occured.

AC								MQ				
0	0	00	000	000	000	101	‖	010	000	000	000	000

When the number in the AC becomes equal to or greater than the divisor, the divisor is subtracted in the AC and 1 is placed in the rightmost bit of the MQ.

```
                AC                                       MQ
0 0  00   000   000   000   000 || 010   000   000   000   001
```

Shift to the left:

```
                AC                                       MQ
0  0  00   000  000   000   001 || 000   000   000   000   010
```

The number in the AC is less than the divisor. Fourteen shifts have been made, and the division operation is complete. The quotient 010 appears in the MQ and the remainder 001 in the AC. The signs of the dividend and the divisor are compared and the correct digit placed in the first bit of the MQ. The above steps correspond to the usual division:

```
            10
101    | 1 011        Quotient
         1 01
         ------
             1
             0
             -
             1        Remainder
```

Floating-point division can be carried out by applying the same procedure to the fractions of the two numbers. The exponent (not the characteristic) of the divisor must be subtracted from the exponent of the dividend. For example, dividing 0.101×2^{-3} by 0.100×2^{2} gives $1.011\ 110 \times 2^{-5} = 0.101\ 111 \times 2^{-4}$.
This would appear in the MQ as follows:

```
             MQ
001   100   101   111   000
```

CONTROL REGISTERS :

The control of the computer may have several registers. During the exectution of a program, the storage location of the next instruction appears in an instruction location .

2.19 CONTROL UNIT OPERATION:

The control unit of the computer contains registers, counters, and decoders.

An instruction register holds the program instruction that the computer is currently performing.
A decoder translates the operation part of an instruction by setting up the arithmetic unit circuits to perform the operation specified in the instruction.

The instruction counter in the control unit indicates the address of the instruction that is being executed. The instruction counter increases during each instruction execution cycle to indicate the address of the next instruction. If instructions in storage are being executed in sequence, the counter increases by one in each cycle.The counter can be reset to zero or to any desired instruction by a transfer or jump instruction.

In programming a loop, an index register contains the current value of the index, while the upper limit of the index is stored into an index limit register.

The operation of the control unit can be illustrated by the following example. Consider the instruction "AD 02 0268" that is stored in storage location 0062, which means : add the contents of storage location 0268 to the contents of the accumulator. Since the instruction specifies one operand only, the computer is called "single address" computer.

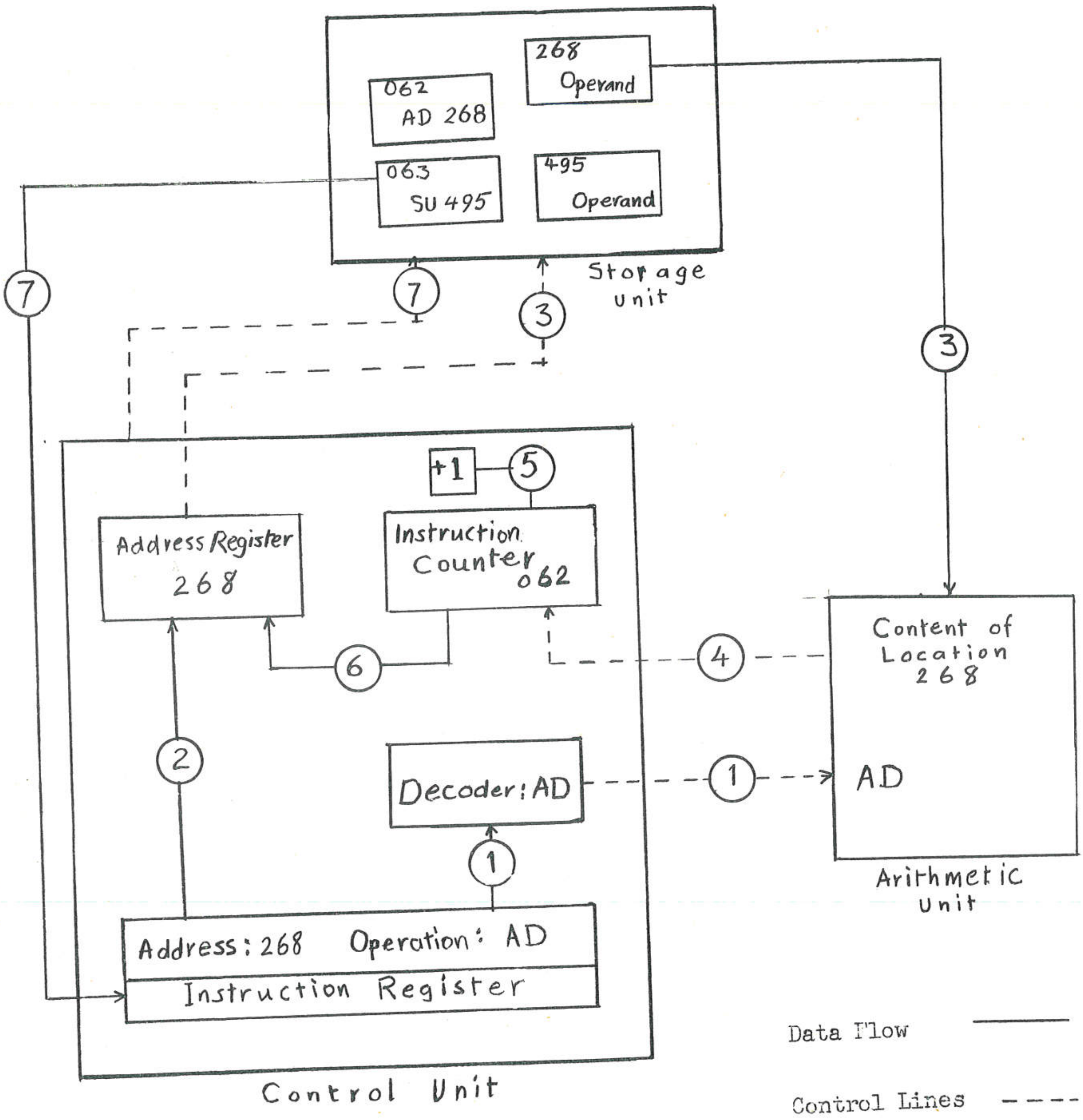


Fig. 2-44
Operation of Control Unit
(Single-address Computer)

Assume that computations are in process and we look in one operating cycle, just after an instruction is put in the instruction register. The instruction register contains an instruction such as "Ad 0268" copied from location 0062. The operating cycle can be described as consisting of 7 steps that are repeated as many times as are required to excute the program.

Referring to figure 2-44, the execution of the considered instruction proceeds as follows:

1. Transfer the operation part of the instruction, AD, from the instruction register to the decoder.
2. Transfer the address part of the instruction, 0268, from the instruction register to the address register.
3. Copy into the arithmetic unit the operand (which may be either data or an instruction) located at address 0268.
4. Execute the required operation, AD, in the arithmetic unit, Notify the control unit when the operation is executed.
5. Increase the number 0062 in the instruction counter by 1, to 0063 , to indicate the address of the next instruction.
6. Transfer the number 0063 from the instruction counter to the address register.
7. Get the instruction "SU 03 0495" located at address 0063 and put it into the instruction register.

Usually , the contents of the control registers as well as the **AC** and **MQ** arithmetic unit registers are displayed on the computer console by means of display lights.

2.20 THE MACHINE CYCLE:

As illustrated above, the computer periodically takes a new instruction out of the storage unit, decodes it, performs the specified operation, and then goes back to the storage unit for another instruction. This sequence is called a machine cycle. A single machine cycle must have a time period enough to allow:

1. An instruction to be extracted from storage.
2. The instruction to be fed to the interpreting device and decoded.
3. The command signal to be sent to the appropriate circuitry.
4. The specified operation to be performed.

If each machine cycle has the same time duration, regardless of the operation performed, the computer is called a synchronous computer. In some computers, the machine cycles are not of the same length, and hence called asynchronous machines.

Obviously, some arithmetic operations require more time to perform than others. In a synchronous computer, the machine cycle must be set to accomodate the slowest operation. Thus, time is wasted when fast operations are performed.

In asynchronous computers, a different length machine cycle is available for each operation. The computer looks for the next instruction immediately after the operation is completed, instead of waiting for the beginning of a fixed-length machine cycle. Thus, asynchronous computers are faster, but are more complicated.

The operation of the digital computer is synchronized by timing pulses produced by a central clock. These timing pulses coordinate the electrical signals that contral the calculating circuits, and garantee that each step of an operation is performed at the proper time in the sequence. Thus, during one machine cycle, timing pulses are applied in sequence to extract a new instruction from storage and transfer it to a register in the control unit where it is decoded, while the subsequent timing pulses of the same machine cycle will be routed through the calculating circuits and applied as command pulses. The routing action is performed by a network of gates controlled by an instruction decoding matrix. Most computers have a clock frequency or pulse repitition rate between 100 and 4000 kc/s.

2.21. SINGLE AND MULTI-ADDRESS INSTRUCTIONS:

Simillar to the hypothetical computer discussed in this chapter, many actual computers are so designed that only one address is specified in each instruction, and hence known as single-address machines. Instructions are obeyed in the order in which they are stored, and hence called sequential machine.

However, other instruction formats can be also used. A variation of the single-address is the one and a half address scheme. For example, "ADD 268, 063" may mean : add the content of location 268 to the quantity in the accumulation and get the next instruction from location 063. In this case, an instruction counter is not necessary. This scheme is useful only if instructions are placed in storage at intervals, instead of in sequence. The half address of each instruction may be used to minimize access to the next instruction. This type is called non-sequential machine.

A two-address instruction may contain an operation and two addresses, both of which may refer to operands. For example, "CAD 437, 438" might mean: clear the accumulator and add the content of storage location 437 and 438.

A <u>three-address</u> instruction specifies 3 operands. Two addresses indicate the operands involved, and the third specifies where to store the results. Thus, instruction " ADD 102, 252, 402" may mean: add contents of locations 102 and 252 and store the total in location 402. Instructions are executed in sequence and a control counter keeps track of the instruction being executed.

Each address scheme has some advantages. Multiple-address computers may be easier to program than a single-address computer, but multiple-address computers have more circuits and components. A single-address computer is in general, simpler to design than a multi-address computer, but needs more instructions to enable it to carry out a given calculation. Singl-address instructions might be faster for the simple operations involved in summing a series of numbers. On the other hand, a 3-address or 4-address (as 3-address plus location of next instruction) may be preferable to a single-address computer if many complex operations are involved.

2.22 RESIDUE NUMBER SYSTEMS

So far we discussed the parity check method which is a simple way of checking the binary code mainpulated by the computer. Using more rudundant bits than the one parity-bit, this number of redundant bits may uniquely identify the number itself. The original number can then be omitted altogether. The numbers obtained in this way are called residue numbers. These require a different arithmetic concept, called residue arithmetic.

The basic concept of the residue number representation is as follows. If we divide any integral number N by an integer (a), we get a quotient and a remainder at most (a-1). This remainder or residue is the number N expressed modulo (a). Taking several values for (a) we get several residues. By choosing prime numbers for (a), N can be uniquely expressed by its residues. For example, taking the first four prime numbers: 2, 3, 5, 7 gives the following table:

Fig. 2 - 45 Residue Table

N	Residues a=2	3	5	7	N	Residues a=2	3	5	7
0	0	0	0		8	0	2	3	1
1	1	1	1	1	9	1	0	4	2
2	0	2	2	2	10	0	1	0	3

N	a=2	3	5	7	N	a=2	3	5	7
3	1	0	3	3	11	1	2	1	4
4	0	1	4	4	12	0	0	2	5
5	1	2	0	5	13	1	1	3	6
6	0	0	1	6	14	0	2	4	0
7	1	1	2	0	15	1	0	0	1

The number 5 would be denoted by 1205 where

5 = 1 mod 2
5 = 2 mod 3
5 = 0 mod 5
5 = 5 mod 7

The notation is unique till it finally repeats after 2x3x5x7=210, which again would be denoted by 0000. The arithmetic operations with these numbers differ completely from ordinary arithmetic. Addition is accomplished by adding the corresponding residues and expressing these sums again by their residues. Eor example,

4 = 0144
+ 6 = 0016
10 = 0103

where 0 + 0 = 0 mode 2
1 + 0 = 1 mode 3
4 + 1 = 0 mode 5
4 + 6 = 3 mode 7

Addition and subtraction can be mechanized very easily, Multiplication in the residue system is affected by obtaining the modulo product of corresponding digits. Since no carries or repeated additions are involved, multiplication is faster than with ordinary binary binary numbers. However, division is not so simple, and also the detection of overflows.

In a computer using the residue system, the residues will be represented in binary form. In the above example, we would need 1 bit for the first digit, 2 bits for the second, 4 for the third, and 4 for the fourth, which makes 11 bits. In direct binary form, only 8 bits are needed. A computer performing residue arithmetic thus has more components than the usual binary computer. The use of residue arithmetic in computers is still under investigation.

In some computers, the internal parity check is replaced by moldulo three check on all operations (Telefunken TR-4), Also modulo 7 check on all operations has been used (Gamma 60, Bull). These checks use the residue of the stored number with respect to some modulus (divisor), in decimal machines, it is chosen to be 9, which makes the method equivalent to the elemantary rule of "casting out nines". The remainder or residue of a number after division by 9 is equal to that of the sum of its (decimal) digits. The check circuit in the computer has to add the digits of the number, casting off the 9's, and comparing with the prestored check sum. For binary computers, the modulus 15 or 31 (in general $2^k - 1$) for k-bit words, are advantageous. For example, the remainder of a binary number after division by 15 is equal to the sum obtained by adding the binary digits from right to left with their position values 1,2,4,8, 1,2,4,8,.. and costing of 15's. This check requires more circuitry than the simple parity check but is more reliable. The residue check can be used to check not only number storage, but also arithmetic operations, where the parity check is inadequate.

2.23. ERROR CORRECTING CODES

Beside errer detection in computer code during internal operations, there is also the possibility of using an errer correcting code using more redundant bits, although it is not up till now employed in computers. Thus, to the 4 bits representing a decimal number in binary form, 3 redundant bits are added. The construction and operation a Hamming code will be explained for single error correction.

Let the decimal number be represented in the binary coded decimal code by the bits x_3 , x_5 , x_6 , x_7. The 3 redundant bits x_1, x_2 , x_4 are obtained from the relations

$$x_4 + x_5 + x_6 + x_7 \equiv a_1$$

$$x_2 + x_3 + x_6 + x_7 \equiv a_2$$

$$x_1 + x_3 + x_5 + x_7 \equiv a_3$$

We choose x_4 to make a_1 even, x_2 to make a_2 even, x_2 to make a_2 even, and x_1 to make a_3 even. The values of a_1, a_2 and a_3 are calculated modulo 2. Hence, an (a) is zero if even, and 1 if odd. Thus we get a binary number $a_1\ a_2\ a_3$ in the range 0 to 7, and it can be shown that the value of this binary number gives the subscript (i) of the digit x_i in the set of 7, which is incorrect, and hence the error bit is located . If there is no error, the binary number $a_1\ a_2\ a_3$ is 000.

Consider for example the decimal number 6 in binary form 0110. We have $x_3 = 0$, $x_5 = 1$, $x_6 = 1$ and $x_7 = 0$. Caculating the redundant bits according to the above procedure, we get:

$x_4 = o$, $x_2 = 1$ and $x_1 = 1$

Hence the decimal number 6 in coded form becomes 1100110.

x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	1	0	0	1	1	0

Now, let an error take place such that x_3 becomes 1 instead of zero, In this case $a_1 = o$, $a_2 = 1$ and $a_3 = 1$. The binary number 011 represents 3 and so indicates an error in the third digit x_3.

It is further possible to detect and correct double errors, but more redundant bits will be needed. In general, the possibilities of detecting and correcting faulty codes increases with the number of redundant bits. But at the same time, the cost of the computer increases, since these extra bits must be stored, generated and checked. However, as the number of electrical components in the computer increases, the probability of malf-unction increases. Thus, an opposite approach to the computer reliability is to use no redundancy at all, as for example the computer Electro Data, a machine of comparable size, in wide use. For scientific computers, the recent improvements made in components favours a trend toward non-redundant codes.

It has been found that at least 90% of all the errors which can occur within the computer proper excluding its peripheral devices, are due to the malfunctioning of electronic components which will result in the loss of a single bit. The parity check will therefore detect about 90% of errors due to machine malfunctioning which are likely to be experienced in practice. This high standard in performance is due to the fact that the transistors used in modern computers are extremely reliable, and are not subject to occaisonal or intermittent failure. In fact the computer is expected to perform millions and billions of operations without error.

Arithmetic checks on the data being processed can be made by instructions at key positions in the program; selected types of operations can be tested in two ways in the computer :

1. Parallel testing involves simultaneous execution of an instruction along two paths in the equipment, and equality test.
2. Serial testing involves one set of circuits used to repeat the operation and perform an equality test. For example, the product of 543 x 978 can be compared with the product of 978 x 543 by subtracting one product from the other and testing for zero.

Usually special checking programs are used in order to check the operation of all instructions and computer units before starting the daily work of the computer.

APPENDIX 1

SWITCHING DEVICES

1. RELAYS:

The first digital computers developed during the second world war used telephone relays. In 1946, the Bell Telephone Laboratories in U.S.A. completed two almost identical machines called "Bell Relay Computers Model 5" based on ideas of G.R. Stibitz. They consisted entirely of telephone relays and teletype equipment, and required an average of 2 seconds per arithmetic operation.

The relay is a device which can open or close electric circuits. The telephone relay depends in its operation on an electo-magnet. It consists of a coil, yoke, armature and an air gap to complete the magnetic circuit, The relay has sets of contacts mounted on springs that can be operated by the relay armature. The telephone relay is neutral or non-polarized, because its operation does not depend on the direction ofthe current flowing in its coil. Figure (1) shows the construction of a telephone relay.

Shortly after the second World War, a relay computer was put into operation at the Birkbeck College, University of London, and another one was completed in Stockholm, Sweden (Kjellberg and Neovius, 1951) A relay computer based on a wartime German design was put into operation in Zurich, Switzerland; a simillar one with relay storage in Wetzlar, Germany. Theses relay computers had all

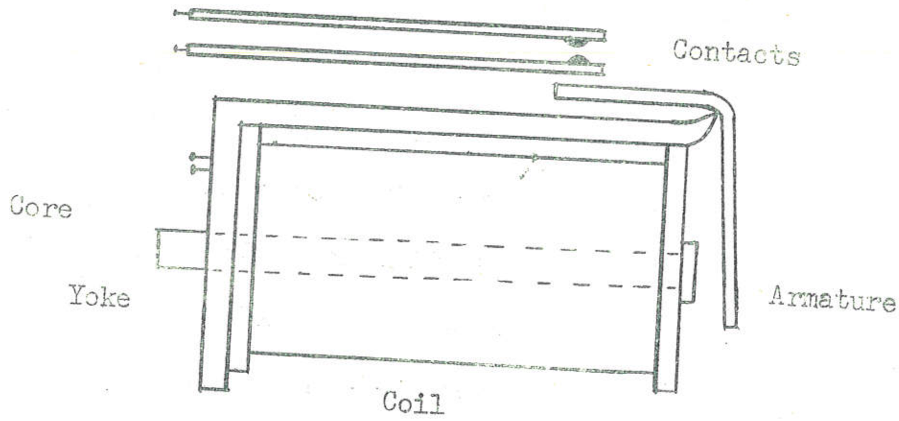


Fig. 1. Telephone Relay

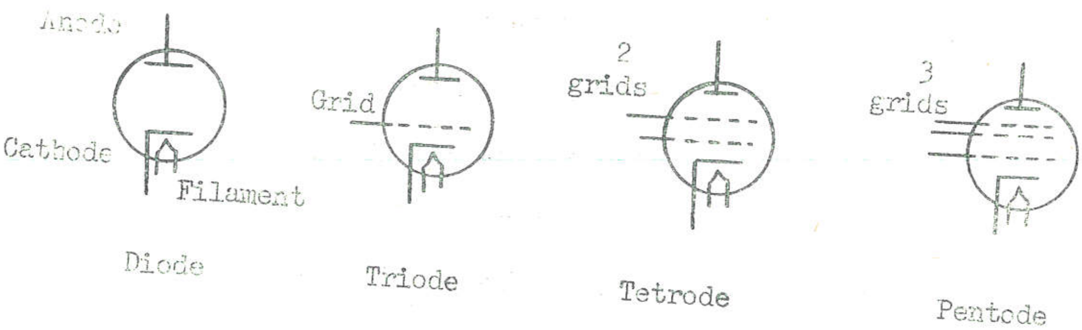


Fig. 2 Electron Tubes

the main features of today's electronic computers, except speed. Relay computer models are still made for teaching purposes.

2. ELECTRON TUBES

It was the use of electron tubes which revolutionized the computer field by making today's high-speed computation possible. The ENIAC was the first computer to use tubes instead of telephone relays. It was developed at the University of Pennsylvania, U.S.A. It proceeded simultaneously with the development of the Bell Relay Computers and was completed slightly earlier in 1946. It used 20,000 tubes as switching elements in arithmetic and control as well as for storage of very small capacity (at most 20 ten-digit numbers). Eniac was in continuous use until 1956.

Electron tubes are classified according to the number of electodes. A vacuum diode has 2 electrodes, anode and cathode. It will permit current to flow through it in only one direction. The cathode is heated to emit electrons. If the anode (also called plate) is made positive with respect to the cathode, electrons will be attracted to the anode, and current will flow. If, on the other hand, the anode is made negative with respect to the cathode, electrons willbe repelled back to the cathode. Thus, depending on the polarity of the anode to cathode voltage, The vacuum diode will be conducting (low resistance) or not conducting (infinite resistance), i.e. an on-off characteristic.

A vacuum triode is simillar in construction to a vacuum diode, but has an additional electrode called control grid, placed between anode and cathode. The grid is used to control the flow of electrons between cath and anode. The anode is made positive with respect to the cathode, so that electrons emitted by the cathode will be attracted to the anode. If the grid is set to zero volts or made slightly positive, electrons can pass through it unhindered to the anode. On the other hand, if the grid is sufficiently negative, it will completely stop or cut-off the flow of electrons. Thus, an on-off characteristic is obtained depending on the polarity of the grid voltage. For this reason, tubes are known in England as valves. Similarly, vacuum tetrodes (having 2 grids) and pentodes (having 3 grids) can be used in switching circuits to carry out logic operations. In turning vacuum tubes on and off, there is no mechanical inertia (as in relays) to overcome, which allows for high speed operation. Yet electron tubes have limited life, require large space, and produce a lot of heat (Eniac consumed 150 KW of electric power).

3. SEMICONDUCTOR DIODES AND TRANSISTORS:

Solid state semi-conductors are characterized by a resistivity higher than that of conductors, but less than that of insulators. A conductor has a resistivity of the order of 10^{-7} ohm-meter, while an insulator has a resistivity of about 10^{10} to 10^{16} ohm-meter. Between these limits, a semiconductor has a resistivity of one ohm-meter. This is due to the fact that the number of free electrons in the semiconductor at room temperature is not large, so that only one conducting electron exists for 10^2 atoms in Germanium.

Semiconductors such as the elements Silicon and Germanium have their atoms arranged in crystals. A Crystal is an orderly array of atoms located in repetitive geometric patterns. The positive charge left when an electron is removed is called a "hole". When an electric field is established within the semiconductor, both holes and electrons migrate under its influence. The conductivity of Germanium and Silicon can be greatly increased by the addition of a small amount of impurity. By introducing Autimoney into Germanium crystals in the ratio of 1 atom Antimony to 10^8 atoms of Germanium, then in a semiconductor sample of 10^{10} atoms we get 100 free electrons introduced by the Antimony atoms and about 5 electrons normally present at room temperature in the semiconductor.

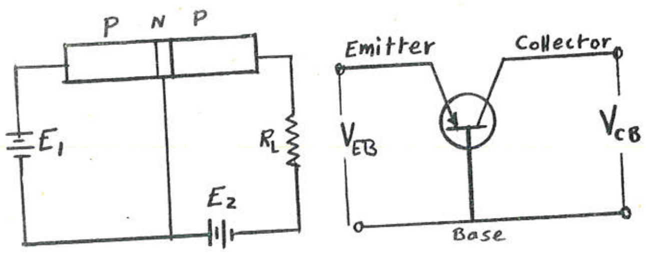
If the impurity atoms introduced have 5 valence electrons such as Antimony or Arsenic (Germanium and Silicon have only 4), the number of free electrons will be increased by the number of impurity atons (doners) added, so that the ratio of free electrons to holes becomes very large. Such a semiconductor is called n-type semiconductor.

If the impurity atoms introduced have three valence electrons such as Boron, Galium and Indium, there will be a tendency to move electrons so that holds (positive charges) are introduced in the semiconductor, since the impurity atoms (acceptors) can accept electrons from surrounding semiconductor atoms.

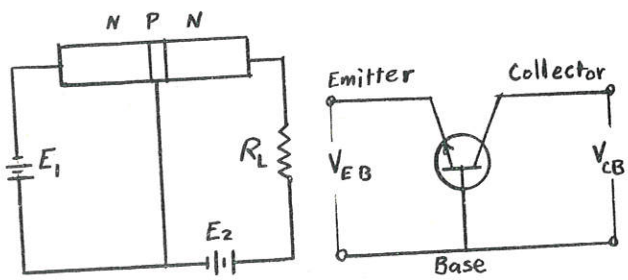
Such a semiconductor is called p-type semicondutor.

A semiconductor diode is formed by establishing contact between one p-typ and one n-type semiconductor we get a p-n junction, where a diffusion action takes place. The holes from the p-type tend to spread into the n-type semiconductor, while free electrons from the n-type tend to spread into the p-type semiconductor. Such a junction has a forward direction when the positive pole of a battery is connected to the p-type terminal and the negative pole to then-type terminal. Conduction in the forward direction across the p-n junction is great due to holes and free electrons increased by the impurity atoms, and a steady current flows across the junction. By reversing the polarity of the junction terminals, the holes (charge carriers) diffuse across the junction and establish an electric field hindering further diffusion, thus giving a small current in the backward direction due to thermally produced electron-hole pairs only. Conduction across the p-n junction in the reverse direction can be greatly increased by placing two junctions in tandem, producing a p-n-p junction transistor (figure 3).

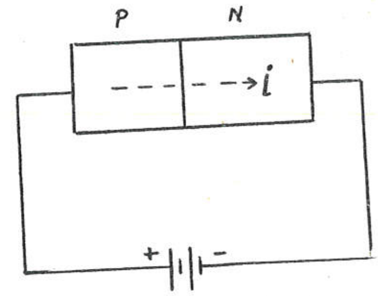
The p-n junction to the left is biased in the forward direction by the battery E_1, while the n-p junction to the right is biased in the reverse direction by the battery E_2. Thus, holes from p region to the left diffuse across the left p-n junction in the forward direction, so that for a small voltage E_1 a relatively large forward current across this junction is produced. The n region is made very thin, so that the holes from p region to the left can



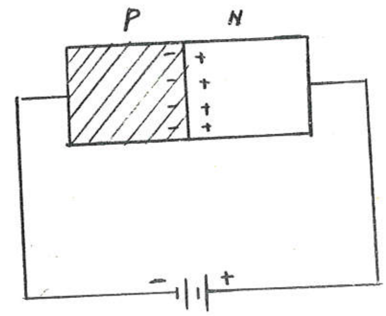
The p-n-p Junction Transistor



The n-p-n Junction Transistor



Forward Direction



Reverse Direction

The p-n-p Junction Diode

Fig. 3. Semi-conductor Diode and Transistor

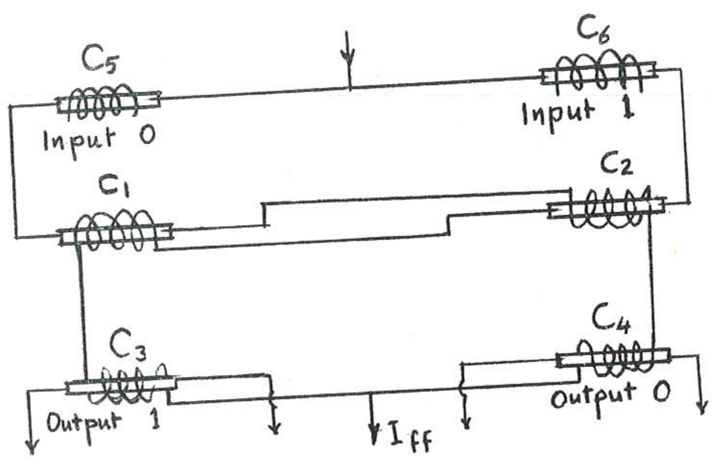


Fig. 4. Cryotron Flip-Flop

travel through the n region without recombining andcross the n-p junction to right p region in the reverse direction. Thus a small change in E_1 causes a large change in the current through the load resistance R_L, producing a high voltage drop across R_L, i.e., the transistor can give amplification. The p-n-p regions of the transistor are called emitter, base and collector respectively. The emitter emits carriers or holes into the n region, callled base. The p-region to the left is called the collector which captures most of the carriers given by the emitter.

In the n-p-n junction transistors, electrons from the n-type emitter diffuse across the emitter-base n-p junction in the forward direction, pass through the thin p-type base without recombining, to the n-type collector in the reverse direction. Both p-n-p and n-p-n types are used, and the symbols adopted to distinguish between them are as follows: in the p-n-p type the arrowhead points outwards the base and in the n-p-n type the opposite is the case. In both types the arrowhead indicates the direction of easy current flow into the emitter electrode. This is positive to negative and hence opposite to the travel of electrons.

Transistors have now been developed to a high reliability standard and because of their size, insignificant weight, machanical strength and lower power consumption have almost completely displaced electon tubes in the construction of computers since about 1957. Another great advantage in that they do not generate and dissipate so much heat as electron tubes.

Special arrangements to ventilate and cool the computer room are therefore unnecessary. The big advance lay in the fact that a heated cathode (or filament) was not required as a source of electrons, thus saving considerabl input power and eliminating a major source of heat. An equally important transistor characteristic is its potentially long life: the probability of tens of thousands of hours of operation as compared to hundreds of hours of electron tubles.

The transistor is also a device that operates at lower voltages than vacuum tubles and this characteristic, coupled with lower operating temperatures, made it possible to develop an entirely new line of associated components, resistors, capacitors, switches, etc. since size is proportional to voltage, power handling capacity, and temperature.

4. CRYOTRONS

The unique properties of materials at very low temperatures (below 125°K) are utilized in cryogenic electronics. At temperatures near absolute zero there is almost complete loss of resistivity, which is a phenomenon called super conductivity. When the super-conductive material is placed in a large magnetic field, the superconductor may be driven out of the superconducting state into the normal resistive state.

A switching device based on superconductivity is the cryotron. It consists of a control coil of Niobium wire surrounding another very small rod of

Tantalum wire, the gate. When immersed in liquid Helium, this element becomes superconductive. If, however, a current is passed through the Niobium control coil, the Tantalum wire recovers back its normal resistivity. By connecting the gate of one cryotron in series with the control coil of another, a switching action can be achieved.

Figure(4) shows a cryotron flip-flop circuit. Let C_3 be conducting, i.e. the supply current is flowing through the control coil of C_4 and a comparatively high resistance is present at its gate terminals. In order to switch from C_3 to C_4 an input signal is applied to the control coil of C_5. A high resistance appears across its terminals and the supply current is now shunted through the gates of C_5 passing through the coil of C_3 and so cutting off the output from cryotron C_3 and obtaining instead an output from C_4.

Cryotron circuits are cheap, small and reliable. The fact that cryotrons must operate in liquid Helium is a major drawback. Up till now, no computer employing cryotrons is yet commercially available.

5. PRINTED CIRCUITS

In the early 1950's another major development beside the transistor opened up entirely new possibilities for size reduction in digital computers. This is the technique of printed circuits which entirely eliminated hand soldering and hand-tooled wiring in the computer. Printed wiring provides the structural mounting, electrical insulation and interconnections of many small electronic components on a single sheet of material. It is specially suited for computer circuitry because of its repetitive nature. The component leads are inserted either manually or automatically in the prepared holes in the insulating board on which a metallic wiring pattern had been automatically formed. In a signle dip or pass-over molten solder, all component leads are fused to the printed or etched interconnecting lines. Thus, transistors and printed circuits have helped to make computer smaller, lighter, more reliable, as well as less expensive. All these advantages can be easily seen every day in the small handy transistor radio receivers.