DIGITAL COMPUTER FUNDAMENTALS AND MICROPROCESSOR

UNIT – I

Introduction: Application of Computer - Different types of Computer systems - Basic components of Digital Computer System - Programming Languages; Number Systems.

UNIT – II

Boolean Algebra and Gate Networks: Fundamentals concepts of Boolean Algebra – Logical Multiplication AND Gates, OR Gates, and Inverters – Evaluation of logical Expressions – Basic Law of Boolean Algebra – Simplification of expressions – De Morgan's theorems – Basic Duality of Boolean Algebra - Derivation of a Boolean Expression.

UNIT - III

Interconnecting Gates – Sum of products (SOP) and Products of sums (POS) – Derivation of products of sums expressions – Derivation of three Input variable expression – NAND gates and NOR gates - The Map method for simplifying expressions – Sub cube and covering – product of sums expressions – Don't cares.

UNIT – IV

Microprocessors, Microcomputers and Assembly Language: Microprocessors - Microprocessor instruction set and Computer Languages-From large computers to single chip Microcontrollers; Microprocessor Architecture and Microcomputer systems: Microprocessor Architecture and its operations – Memory – I/O devices; 8085 Microprocessor Architecture and Interfacing: The 8085 MPU – Examples of a 8085 based Microcomputer – Memory interfacing.

UNIT –V

Programming the 8085: Introduction to 8085 Instructions ; Code conversion: BCD to Binary conversion – Binary to BCD conversion – BCD to seven segment LED code conversion – Binary to ASCII and ASCII to binary code conversion – BCD addition – BCD subtraction.

TEXT BOOKS

1. Digital Computer Fundamentals‖.(6TH Edition) Thomas C.Bartee, 6th Edition T.M.H Publisher, New Delhi, 1991.(UNIT I, II & III)
2. ―Microprocessor Architecture Programming and Application with the 8085‖. Ramesh Gaonkar, 5th Edition. (UNIT IV & V)

REFERENCE BOOKS:

1. Deborah Morley, Charles S. Parker, "Understanding Computers-

Today and Tomorrow", 1stEdition, Thomson Course Technology, 2007

3. N.K. Srinath,"8085 Microprocessor Programming and Interfacing", PHI Publishing, 2005.

UNIT -- I

Introduction: Application of Computer - Different types of Computer systems - Basic components of Digital Computer System - Programming Languages; Number Systems.

What is Computer:

•Computer is an electronic device that is designed to work with Information. *The Term computer* is *derived from the Latinterm*'**computare**', this means to *calculate*or *Programmable machine.*

•**Computer cannot do any thing without a Program**. It represents the decimal numbers through a string of binary digits. The Word 'Computer' usually refers to the **Central Processing unit plus InternalMemory**.

•Charles Babbage is called the "Grand Father" of the computer. The First mechanical computer designed by Charles Babbage was called **Analytical Engine.**It uses read-only memory in the form of punch cards.

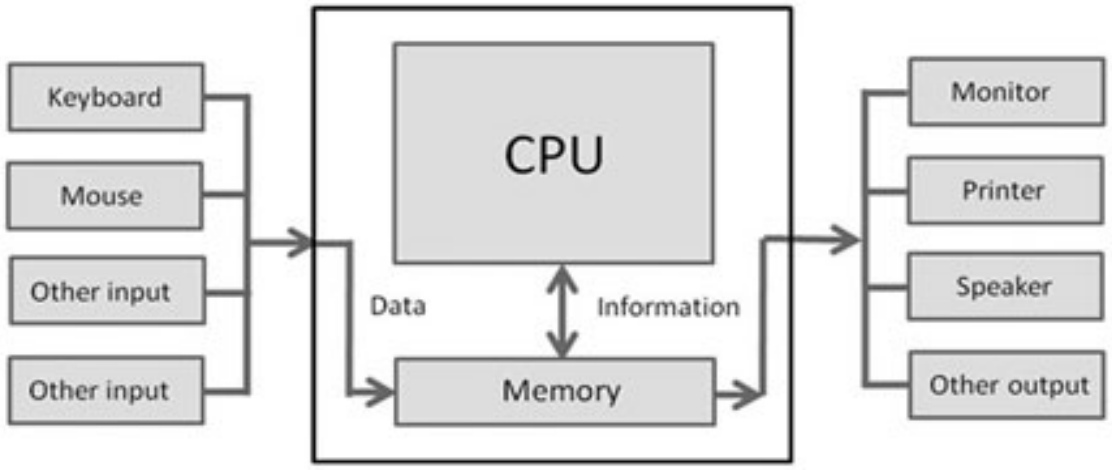
•Computer is an advanced electronic device that takes raw data as input from the user and processes these data under the control of set of instructions (called program) and gives the result (output) and saves output for the future use. It can process both numerical and non-numerical (arithmetic and logical) calculations

Digital Computer

•The basic components of a modern digital computer are:Input Device, Output Device, Central Processor Unit(CPU), mass storage device and memory. A Typical modern computer uses LSI Chips.

•Four Functions about computer are:

1. To Accept data- *Input*
2. To Process data- *Processing*
3. To Produce output- *Output*
4. To Store results- *Storage*



Input(Data):

- Input is the raw information entered into a computer from the input devices. It is the collection of letters, numbers, images etc.

Process:

•Process is the operation of data as per given instruction. It is totally internal process of the computer system.

Output:

•Output is the processed data given by computer after data processing. Output is also called as Result. We can save these results in the storage devices for the future use.

Applications of Computer

Business

•A computer has high speed of calculation, diligence, accuracy, reliability, or versatility which made it an integrated part in all business organisations.

•Computer is used in business organisations for: •Payroll calculations

•Budgeting

•Sales analysis

•Financial forecasting

•Managing employees database

•Maintenance of stocks etc

Banking

•Today banking is almost totally dependent on computer.

•Banks provide following facilities:

•Banks provide online accounting facility, which includes current balances,deposits, overdrafts, interest charges, shares, and trustee records.

•ATM machines are making it even easier for customers to deal with banks.

Insurance

•Insurance companies are keeping all records up-to-date with the help of computers. The insurance companies, finance houses and stock broking firms are widely using computers for their concerns.

•Insurance companies are maintaining a database of all clients with information showing
•procedure to continue with policies

•starting date of the policies

•next due installment of a policy

•maturity date
•interests due
•survival benefits
•bonus

Education

•The computer has provided a lot of facilities in the education system.

•The computer provides a tool in the education system known as CBE (Computer Based Education).

•CBE involves control, delivery, and evaluation of learning.

•The computer education is rapidly increasing the graph of number of computer students.

•There are number of methods in which educational institutions can use computer to educate the students.

•It is used to prepare a database about performance of a student and analysis is carried out on this basis

Marketing

•In marketing, uses of computer are following:

•**Advertising-**With computers, advertising professionals create art and graphics, write and revise copy, and print and disseminate ads with the goal of selling more products.

•**At Home Shopping-**Home shopping has been made possible through use of computerized catalogues that provide access to product information and permit direct entry of orders to be filled by the customers

HealthCare

•Computers have become important part in hospitals, labs, and dispensaries.

The computers are being used in hospitals to keep the record of patients and medicines. It is also used in scanning and diagnosing different diseases.

ECG, EEG, Ultrasounds and CT Scans etc., are also done by computerized machines.

•Some major fields of health care in which computers are used are:

•**Diagnostic System-**Computers are used to collect data and identify cause of illness.

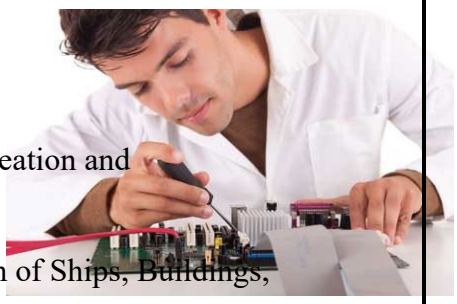
•**Lab-diagnostic System-**All tests can be done and reports are prepared by computer.

•**Patient Monitoring System-**These are used to check patient's signs for abnormality such as in Cardiac Arrest, ECG etc.

•**Pharma Information System-**Computer checks Drug-Labels, Expiry dates, harmful drug's side effects etc.

•**Surgery:** Nowadays, computers are also used in performing surgery.

Engineering Design



•Computers are widely used in Engineering purpose.

•One of major areas is CAD (Computer aided design). That provides creation and modification of images. Some fields are:

•**Structural Engineering-** Requires stress and strain analysis for design of Ships, Buildings, Budgets, Airplanes etc.

•**Industrial Engineering-**Computers deal with design, implementation and improvement of integrated systems of people, materials and equipments.

•**Architectural Engineering-**Computers help in planning towns, designing buildings, determining a range of buildings on a site using both 2D and 3D drawings.

Government

•Computers play an important role in government. Some major fields in this category are:

•Budgets

•Sales tax department

•Income tax department

•Male/Female ratio

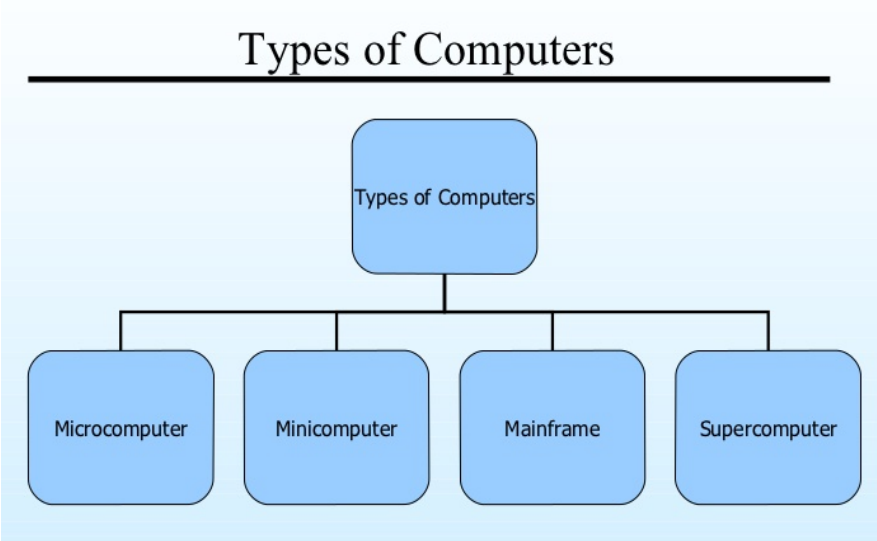
•Computerization of voters lists

•Computerization of driving licensing system

•Computerization of PAN card

•Weather forecasting

Different Types of computer System



- We have **four different computer types** classified according to their performance, power, and size.
- A computer is an electronic device that accepts data, processes it, stores, and then produces an output.
- There are different computer types available depending on the number of users they can support at any one time, their size, and power.

MINI COMPUTERS

- **Minicomputers** — Minicomputers are mid sized computers.
- In terms of size and power, minicomputers are ranked below *mainframes*.
- A minicomputer is a multiprocessing system capable of supporting from 4 to about 200 users simultaneously.
- The use of the term Minicomputer has diminished and they have merged with servers.

Minicomputer

- **Advantage**
  - Cater to multiple users
  - Lower costs than mainframes
- **Disadvantage**
  - Large
  - Bulky

MAINFRAME COMPUTERS

- **2. Mainframe computers** — These are large and expensive computer types capable of supporting hundreds, or even thousands, of users simultaneously.
- Thus, they are mostly used by governments and large organizations for bulk data processing, critical applications, transaction processing, census, industry and consumer statistics among others.
- They are ranked below supercomputers

Mainframe

- **Advantage**
  - Supports many users and instructions
  - Large memory
- **Disadvantage**
  - Huge size
  - Expensive

SUPER COMPUTERS

- **Supercomputers are Very Fast and Most Powerful**
- **1. Supercomputers –** Supercomputers are very expensive, very fast, and the most powerful computers we have in the world.
- Supercomputers are optimized to execute a few number of programs.

- This makes it possible for them to execute these few programs at a **very high speed**.
- Due to their inhibiting cost, they are used in high end places like in scientific research centres.
- The supercomputer consists of thousands of processors making it clock very high speeds measured by petaflops.

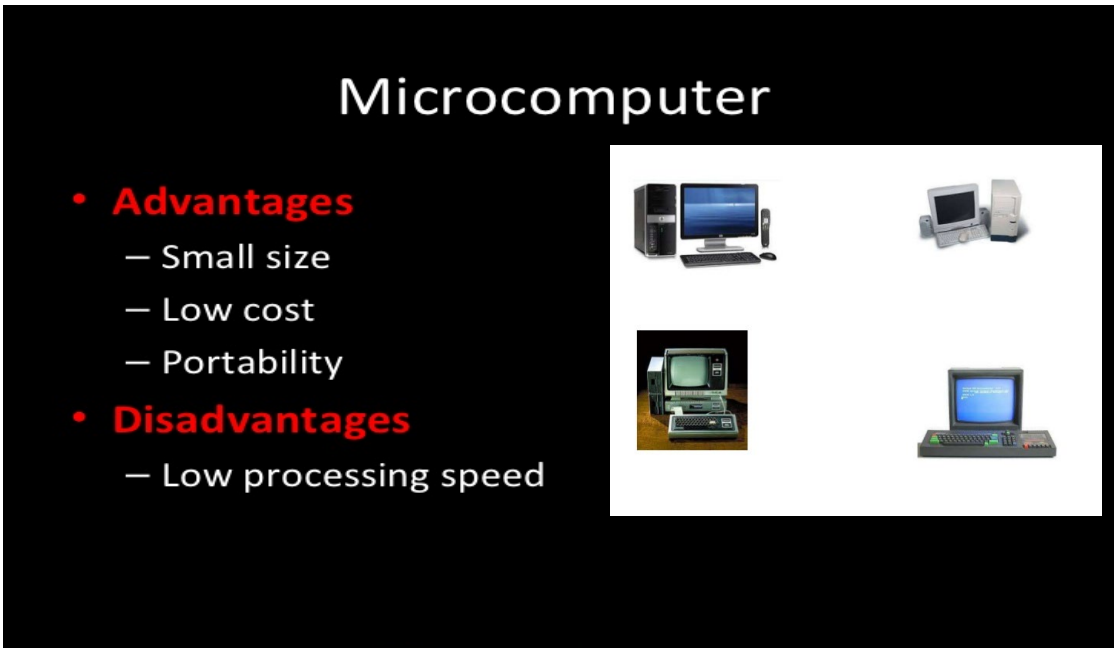
- These computer types are also very large in size due to the numerous parts and components involved in their design.
- A good example of a Supercomputer is **Tianhe-2:** TH-IVB-FEP Cluster; National Super Computer Center in Guangzhou, China; 3.12 million cores (33.86 petaflop/s).

Super Computer

- **Advantage**
  - Speed
- **Disadvantage**
  - Generate a large amount of heat during operation

MICRO COMPUTER

- **Microcomputers or Personal computers** – A personal computer is a computer designed to be used by one user at a time.
- The term *microcomputer* relates to microprocessor which is used with a personal computer for the purpose of processing data and instruction codes.
- These are the most common computer types since they are not very expensive.

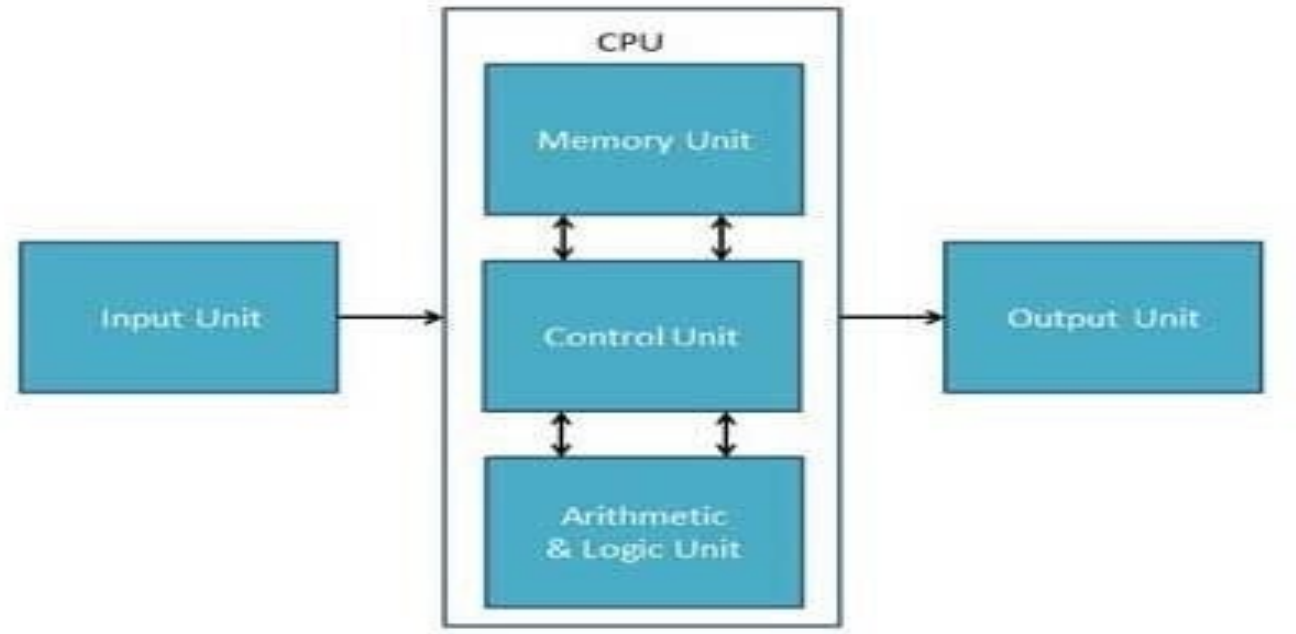


Basic components of Digital Computer System

All types of computers follow the same basic logical structure and perform the following five basic operations for converting raw input data into information useful their users.

S.No.	Operation	Description
1	Take Input	The process of entering data and instructions into the computer system.
2	Store Data	Saving data and instructions so that they are available for processing as and when required.
3	Processing Data	Performing arithmetic, and logical operations on data in order to convert them into useful information.

4	Output Informatio n	The process of producing useful information or results for the user,such as a printed report or visual display.
5	Control the	Directs the manner and sequence in which all of the above operations are performed.



Input Unit

This unit contains devices with the help of which we enter data into the computer. This unit creates a link between the user and the computer. The input devices translate the information into a form understandable by the computer.

CPU (Central Processing Unit)

CPU is considered as the brain of the computer. CPU performs all types of data processing operations. It stores data, intermediate results, and instructions (program). It controls the operation of all parts of the computer.

CPU itself has the following three components −

- ALU (Arithmetic Logic Unit)
- Memory Unit
- Control Unit

Output Unit

The output unit consists of devices with the help of which we get the information from the computer. This unit is a link between the computer and the users. Output devices translate the computer's output into a form understandable by the users.

Central Processing Unit (CPU) consists of the following features −

- CPU is considered as the brain of the computer.
- CPU performs all types of data processing operations.
- It stores data, intermediate results, and instructions (program).
- It controls the operation of all parts of the computer.

Memory or Storage Unit

This unit can store instructions, data, and intermediate results. This unit supplies information to other units of the computer when needed. It is also known as internal storage unit or the main memory or the primary storage or Random Access Memory (RAM).

Its size affects speed, power, and capability. Primary memory and secondary memory are two types of memories in the computer. Functions of the memory unit are −

- It stores all the data and the instructions required for processing.
- It stores intermediate results of processing.
- It stores the final results of processing before these results are released to an output device.
- All inputs and outputs are transmitted through the main memory.

Control Unit

This unit controls the operations of all parts of the computer but does not carry out any actual data processing operations.

Functions of this unit are −

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It manages and coordinates all the units of the computer.
- It obtains the instructions from the memory, interprets them, and directs the operation of the computer.
- It communicates with Input/Output devices for transfer of data or results from storage.
- It does not process or store data.

ALU (Arithmetic Logic Unit)

This unit consists of two subsections namely,

- Arithmetic Section
- Logic Section

Arithmetic Section

Function of arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication, and division. All complex operations are done by making repetitive use of the above operations.

Logic Section

Function of logic section is to perform logic operations such as comparing, selecting, matching, and merging of data.

Types of Programming Languages

There are two **types of programming languages**, which can be categorized into the following way

1. Low level language

Machine language (1GL)

Assembly language (2GL)

2. High level language

Procedural-Oriented language (3GL)

Problem-Oriented language (4GL)

Natural language (5GL)

1. Low level language

This language is the most understandable language used by computer to perform its operations. It can be further categorized into:

MachineLanguage(1GL)

Machine language consists of strings of binary numbers (i.e. 0s and 1s) and it is the only one language, the processor directly understands. Machine language has an Merits of very fast execution speed and efficient use of primary memory.

Merits:

- It is directly understood by the processor so has faster execution time since the programs written in this language need not to be translated.
- It doesn't need larger memory.

Demerits:

- It is very difficult to program using 1GL since all the instructions are to be represented by 0s and 1s.
- Use of this language makes programming time consuming. It is difficult to find error and to debug.
- It can be used by experts only.

Assembly Language

Assembly language is also known as low-level language because to design a program programmer requires detailed knowledge of hardware specification. This language uses mnemonics code (symbolic operation code like 'ADD' for addition) in place of 0s and 1s. The program is converted into machine code by assembler. The resulting program is referred to as an object code.

Merits:

- It is makes programming easier than 1GL since it uses mnemonics code for programming. Eg: ADD for addition, SUB for subtraction, DIV for division, etc.
- It makes programming process faster.
- Error can be identified much easily compared to 1GL.
- It is easier to debug than machine language.

Demerits:

Programs written in this language is not directly understandable by computer so translators should be used.

¨ It is hardware dependent language so programmers are forced to think in terms of computer's architecture rather than to the problem being solved.

¨ Being machine dependent language, programs written in this language are very less or not protable.

¨ Programmers must know its mnemonics codes to perform any task.

2. High level language

Instructions of this language closely resembles to human language or English like words. It uses mathematical notations to perform the task. The high level language is easier to learn. It requires less time to write and is easier to maintain the errors. The high level language is converted into machine language by one of the two different languages translator programs;**inter preteror compiler.**

Procedural-Oriented language(3GL)

Procedural Programming is a methodology for modeling the problem being solved, by determining the steps and the order of those steps that must be followed in order to reach a desired outcome or specific program state. These languages are designed to express the logic and the procedure of a problem to be solved. It includes languages such as Pascal, COBOL, C, FORTAN, etc.

Merits:

¨ Because of their flexibility, procedural languages are able to solve a variety of problems.

¨ Programmer does not need to think in term of computer architecture which makes them focused on the problem.

¨ Programs written in this language are portable.

Demerits:

¨ It is easier but needs higher processor and larger memory.

¨ It needs to be translated therefore its execution time is more.

Problem-Oriented language(4GL)

It allows the users to specify what the output should be, without describing all the details of how the data should be manipulated to produce the result. This is one step ahead from 3GL. These are result oriented and include database query language.
Eg: Visual Basic, C#, PHP, etc.
The objectives of 4GL are to:

- Increase the speed of developing programs.
- Minimize user's effort to botain information from computer.
- Reduce errors while writing programs.

Merits:

¨ Programmer need not to think about the procedure of the program. So, programming is much easier.

Demerits:

¨ It is easier but needs higher processor and larger memory.
¨ It needs to be translated therefore its execution time is more.

Natural language(5GL)

Natural language are stil in developing stage where we could write statrments that would look like normal sentences.

Merits:

¨ Easy to program.
¨ Since, the program uses normal sentences, they are easy to understand.¨ The programs designed using 5GL will have artificial intelligence (AI). ¨ The programs would be much more interactive and interesting.

Demerits:

¨ It is slower than previous generation language as it should be completely translated into binary code which is a tedious task.
¨ Highly advanced and expensive electronic devices are required to run programs developed in 5GL. Therefore, it is an expensive approach.
These are the different **types of programming languages** with their merits and demerits.

Memory

UNIT	ABBREVIATION	STORAGE
Bit	B	Binary Digit, Single 1 or 0
Nibble	-	4 bits
Byte/Octet	B	8 bits
Kilobyte	KB	1024 bytes
Megabyte	MB	1024 KB
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB
Petabyte	PB	1024 TB
Exabyte	EB	1024 PB
Zettabyte	ZB	1024 EB
Yottabyte	YB	1024 ZB

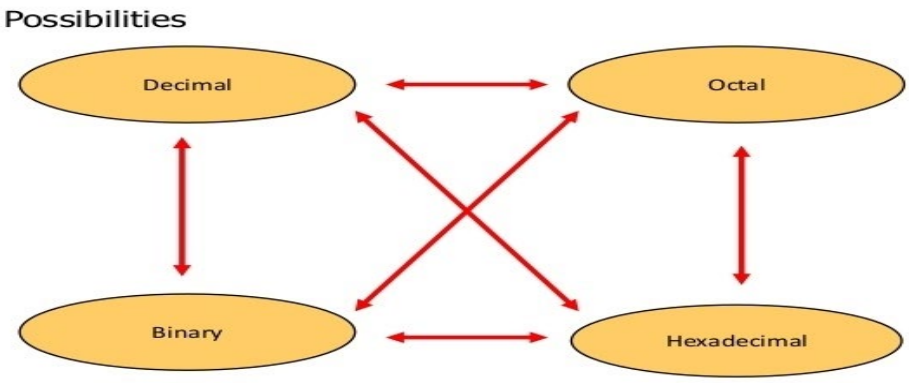
Storage units (www.byte-notes.com

Number systems

A Number system defines a set of values used to represent the quantity. It has different types of number systems

- Convert decimal numbers to binary.
- Convert binary numbers to decimal.
- Convert decimal numbers to Octal.
- Convert Octal number to decimal numbers.
- Convert decimal numbers to hexadecimal.
- Convert hexadecimal numbers.
- Convert binary numbers to Octal.
- Convert binary numbers to hexadecimal.
- Convert Octal number to binary.
- Convert hexadecimal Numbers to binary.

Conversion Among Bases



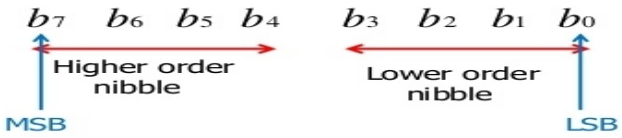
Different Number Systems

- ✓ Decimal Number System
  - Base/Radix 10
- ✓ Binary Number System
  - Base/Radix 2
- ✓ Octal Number System
  - Base/Radix 8
- ✓ Hexadecimal Number System
  - Base/Radix 16

Terms related to Binary Numbers

✓ **BYTE:** A byte is a combination of 8 binary bits.

✓ The number of distinct values represented by a byte is 256 ranging from 0000 0000 to 1111 1111.



Terms related to Binary Numbers

✓ **NIBBLE:** A nibble is a combination of 4 binary bits.

Examples, 1110
0000
1001
0101

Binary Number System

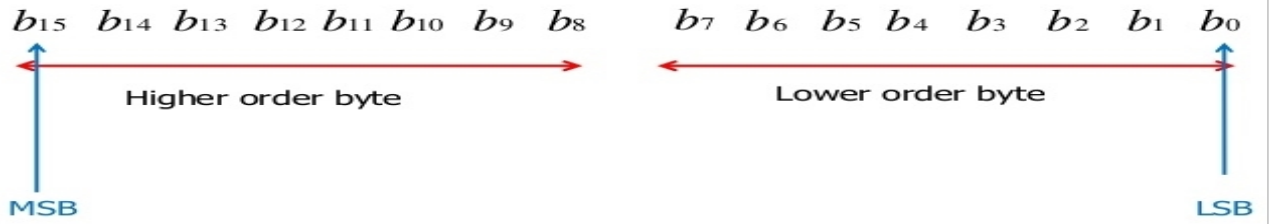
✓ A binary digit is called a **"Bit"**

✓ A binary number consists of a sequence of bits, each of which is either a 0 or a 1.

✓ The binary point separates the integer and fraction parts

Terms related to Binary Numbers

✓**WORD:** A word is a combination of 16 binary bits. Hence it consists of two bytes.



Terms related to Binary Numbers

✓**DOUBLE WORD:** A double word is exactly what its name implies, two words.

-It is a combination of 32 binary bits.

Bits and Bytes

•A binary digitis a single numeral in a binary number.

•Each 1 and 0 in the number below is a binary digit: – 1 0 0 1 0 1 0 1

•The term “binary digit” is commonly called a “bit.”

•The total number of digits used in a number system is called its base or radix. its grouped together is called a “byte.”

DecimalNumberSystem

•The prefix “deci-” stands for 10

•The decimal number system is a Base 10 number system:

1. There are 10 symbols that represent quantities:

2.0, 1, 2, 3, 4, 5, 6, 7, 8, 9

4. Each place value in a decimal number is a power of 10.

Background Information

•Any number to the 0 (zero) power is 1.

$4^0 = 1$ $16^0 = 1$ $1{,}482^0 = 1.$

•Any number to the 1st power is the number itself.

$10^1 = 10$ $49^1 = 49$ $827^1 = 827$

Binary Numbers

•The prefix “bi-” stands for 2

•The binary number system is a Base 2 number system:

*There are 2 symbols that represent quantities:0, 1

*Each place value in a binary number is a power of 2.

1)Binary Number System

A Binary number system has only two digits, which are 0 and 1. Every number (value) is represented with 0 and 1 in this number system. The base of binary number system is 2, because it has only two digits. Though DECIMAL (No 3) is more frequently used in Number representation, BINARY is the number system form which the system/machine accepts.

2**)Octal number system**

Octal number system has only eight (8) digits from 0 to 7. Every number (value) is represented with 0,1,2,3,4,5,6 and 7 in this number system. The base of octal number system is 8, because it has only 8 digits.

3) Decimal number system

Decimal number system has only ten (10) digits from 0 to 9. Every number(value) is represented with 0,1,2,3,4,5,6, 7,8 and 9 in this number system. The base of decimal number system is 10, because it has only 10 digits.

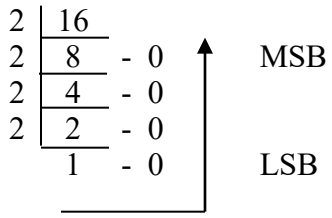
4)Hexadecimal number system

A Hexadecimal number system has sixteen (16) alphanumeric values from 0 to 9 and A to F. Every number (value) represents with 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F in this number system. The base of hexadecimal number system is 16, because it has 16 alphanumeric values. Here, we have 0 to 9, representing 0 – 9 but from 10, we have A is 10, B is 11, C is 12, D is 13, E is

NUMBER SYSTEM	BASE	USED DIGIT	EXAMPLE
BINARY	2	0,1	$(111011)_2$
OCTAL	8	0,1,2,3,4,5,6,7	$(360)_8$
DECIMAL	10	0,1,2,3,4,5,6,7,8,9,	$(240)_{10}$
HEXA DECIMAL	16	0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F.	$(F0)_{16}$

Convert decimal numbers to binary

•16



ANS: $(16)_{10} = (1000)_2$

Convert binary numbers to decimal

•A binary number can be converted into a decimal number by adding the products of each bit and its weight.

METHOD: 1

•(101)

$$101_2 \qquad = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 0 + 1$$

$$= 5$$

ANS: $(101)_2 = (5)_{10}$

Binary Fractions

$$2^0 = 1$$

$$2^{-1} = \frac{1}{2^1} = \frac{1}{2} = 0.5$$

$$2^{-2} = \frac{1}{2^2} = \frac{1}{4} = 0.25$$

$$2^{-3} = \frac{1}{2^3} = \frac{1}{8} = 0.125$$

$$2^{-4} = \frac{1}{2^4} = \frac{1}{16} = 0.0625$$

$$2^{-5} = \frac{1}{2^5} = \frac{1}{32} = 0.03125$$

METHOD:2

$(1101.0111)_2$

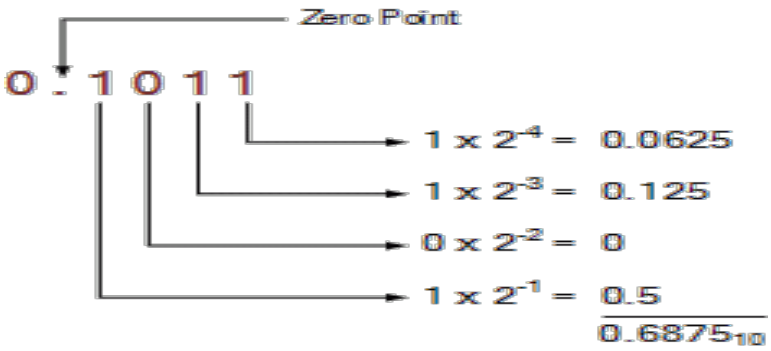
$$101.0111 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3}) + (1 \times 2^{-4})$$

$= 8 + 4 + 0 + 1 + 0 + 1/4 + 1/8 + 1/16$

$= 8 + 4 + 0 + 1 + 0 + 0.25 + 0.125 + 0.0625$

Ans:

$(1101.0111)_2 = 13.4375_{10}$



Octal Number System

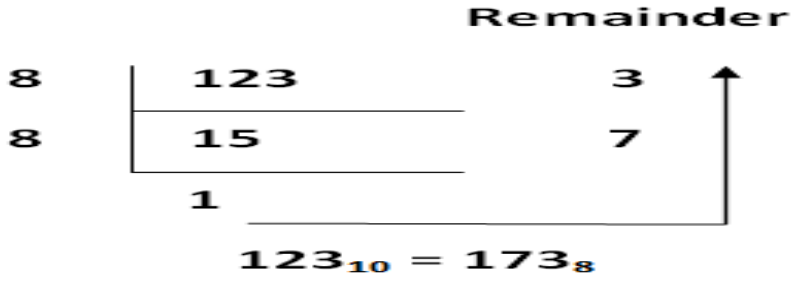
•The prefix "Oct -" stands for 8

•The Octal number system is a Base 8 number system:

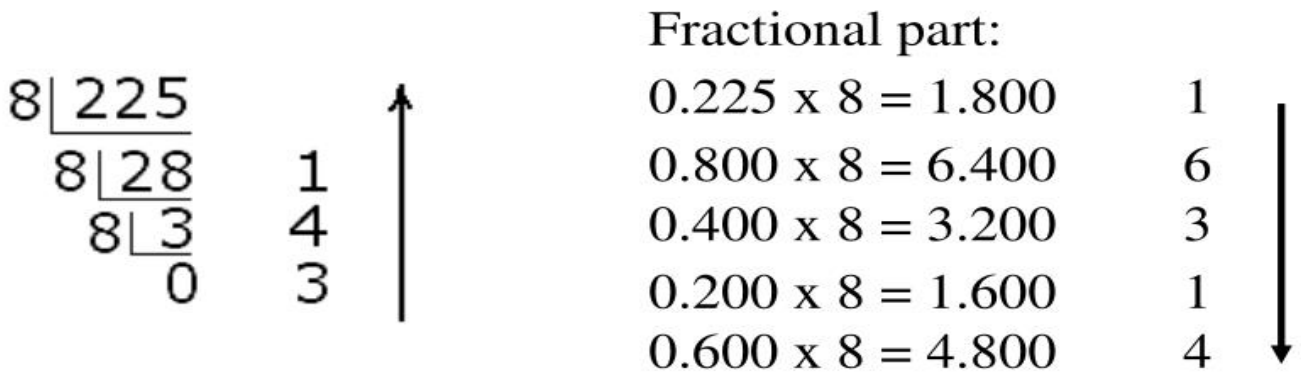
1. There are 8 symbols that represent quantities:
2. 0, 1, 2, 3, 4, 5, 6, 7
3. Each place value in a decimal number is a power of 8.

Convert decimal numbers to Octal

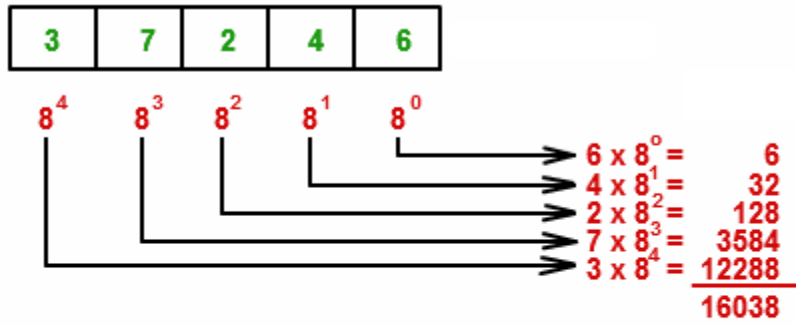
•To convert a decimal number to octal, we have to divide the decimal number by 8 repeatedly and collect the remainders from top to bottom



$$(225.225)_{10} = (341.16314)_8$$



Convert Octal numbers to decimal number



Octal = 37246

Decimal = 16038

- Example: 253.64_8

  $= (2\times8^2) + (5\times8^1) + (3\times8^0) + (6\times8^{-1}) + (4\times8^{-2})$

  $= 128 + 40 + 3 + 0.75 + 0.0625$

  $= 171.8125$

- Exercise: Convert 172.4_8 to decimal

  Answer: $172.4_8 = 122.5$

Hexadecimal Number System

•The prefix "Hexa -" stands for 16

- •The Hexa number system is a Base 16 number system:
- There are 16 symbols that represent quantities:
- 2.0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Each place value in a decimal number is a power of 16.

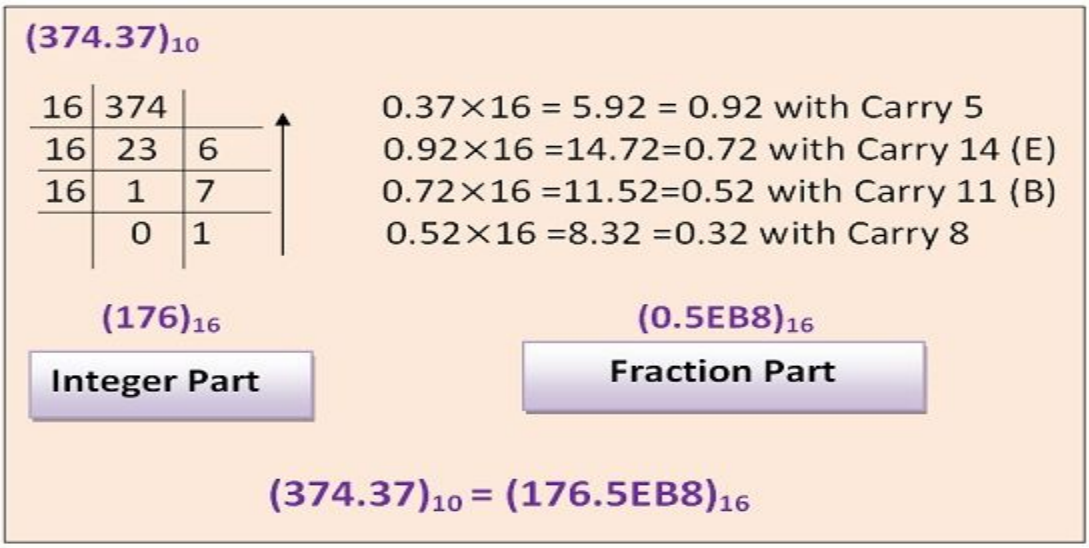
Hexadecimal Number System (HEX)

Decimal No.	Binary No.	Hex No.
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

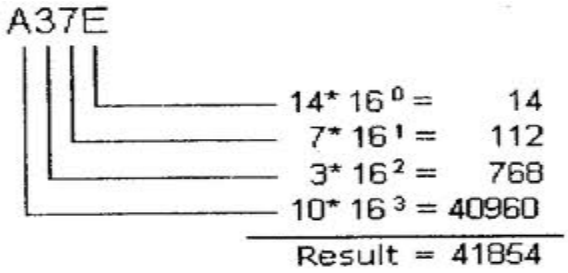
Decimal No.	Binary No.	Hex No.
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

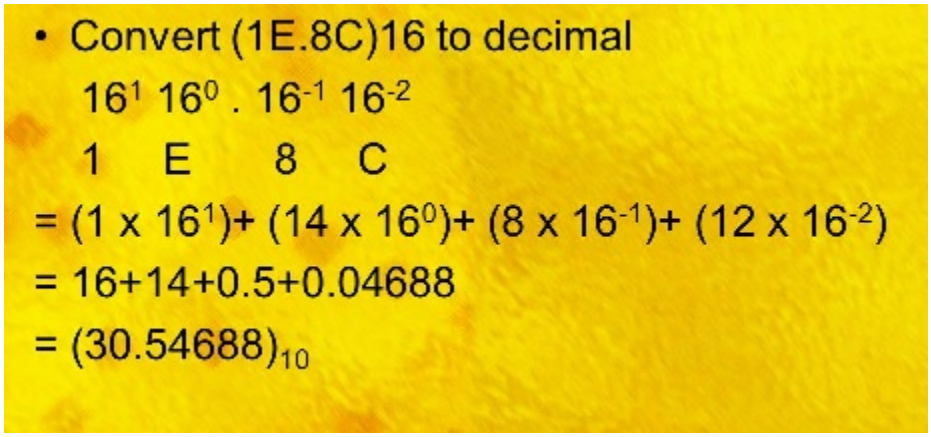
Convert decimal numbers to hexadecimal

•To convert a decimal number to octal, we have to divide the decimal number by 8 repeatedly and collect the remainders from top to bottom.

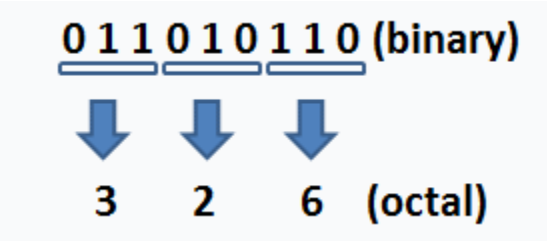


Convert hexadecimal numbers to decimal



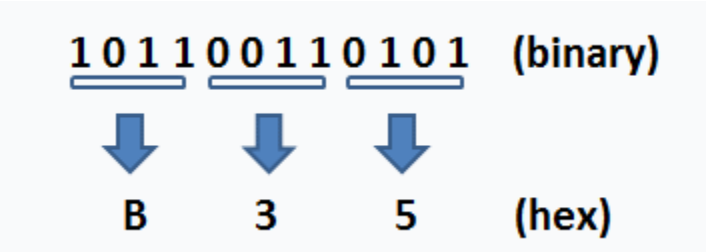


Convert binary numbers to Octal



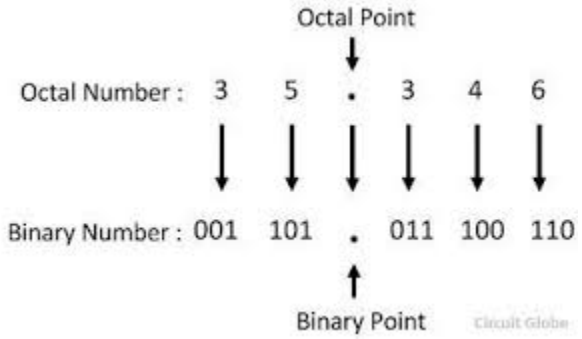
1) 10111.1_2
   $010111 . 100_2$
   $(2\ \ 7\ \ .\ 4\ )_8$

Convert binary numbers to hexadecimal

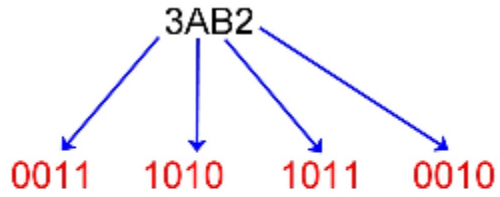


1) $(110.101)_2$
   $(0110.1010)_2$
   $(6 . A)_{16}$

Convert Octal number to binary



Convert hexadecimal Numbers to binary



$3AB2_{16} = 11101010110010_2$

UNIT -- II

Boolean Algebra and Gate Networks: Fundamentals concepts of Boolean Algebra – Logical Multiplication AND Gates, OR Gates, and Inverters – Evaluation of logical Expressions – Basic Law of Boolean Algebra – Simplification of expressions – De Morgan's theorems – Basic Duality of Boolean Algebra - Derivation of a Boolean Expression.

BOOLEAN ALGEBRA AND GATE NETWORKS

Modern Computers are designed and maintained, and their operation is analyzed, by using techniques and symbology from a field of mathematics called modern algebra. Algebraists have studied for over a hundred years mathematical systems called Boolean algebra.

- The name Boolean algebra honors a fascinating English mathematician, George Boole.
- He published a classic book on 1854, an investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities.
- Calculus of propositions and the algebra of sets, were based principally on Boole's work.

FUNDAMENTAL CONCEPTS OF BOOLEAN ALGEBRA.

The fundamental concepts may include the following that

- ✓ The variable used in Boolean equation has a unique characteristic.
- ✓ The two values assumed by a variable may be represented by the symbols 0 and 1.
- ✓ The original symbol proposed by Boole was '+'.
- ✓ For a given value of the variable, the function can be either 0 or 1.
- ✓ The rules for this operation can be given as follows: 0+0 =0
  0+1=1
  1+0=1
  1+1=1

This is logical addition table and represents a standard binary addition table except for the last entry.

Both x and y represent 1s, the value of x+y is 1.

LOGICAL MULTIPLICATION AND GATES

In Boolean Algebra '.' Symbol is used to represent Logical multiplication and AND operation.

The rules for this operation are as follows: 0.0=0
0.1=0
1.0=0
1.1=1

Both (+) and (.) obey a mathematical rule called the associative law. For instance

$$(X+Y)+Z = X+(Y+Z)$$
$$(X.Y).\ Z = X.\ (Y.Z)$$

We can simply write as **X + Y + Z** and **X. Y. Z**. For no matter in what order the operation is performed, the result is same.

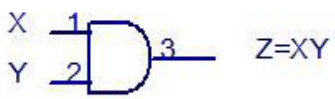
The + and . operations are physically realized by two types of electronic circuits called OR gates and AND gates.

GATE

A gate is simply an electronic circuit, which operates on one or more input signals to produce an output signal.

AND GATE

The AND gate is the logical circuit with the operation similar to logical multiplication. They produce the high output if all the inputs are in high state, this gate produce low output, when any one of the inputs is low

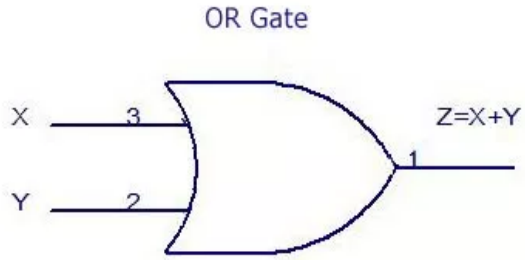


TRUTH TABLE

INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE

The OR gate is similar to the operation of arithmetic addition. The OR gate produce the high output when any one of the input is in high state. We get low output if either of the input is low.

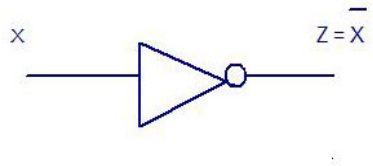


TRUTH TABLE

INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Complementation and inverters or NOT gate

- ✓ In Boolean algebra, we have an operation called complementation and the symbol is "_".
- ✓ The complement is physically realized by a gate called inverter or NOT gate.
- ✓ The NOT gate produce high output, when the input is low and low output when the input is high.



TRUTH TABLE

INPUT	OUTPUT
X	Z
0	1
1	0

EVALUATION OF LOGICAL EXPRESSION

- The table of values for the three operations is called tables of combinations.
- To study a logical expression, it is very useful to construct a table of values for the variables.
- Consider the expression **X+ YZ.** There are three variables in this expression **X, Y, Z**, each of which can assume the value **0** or **1**.

X	Y	Z	Z'	YZ'	X + YZ'
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

- ➢ The above is the truth table for the expression **X + YZ'**. The value **Z** is complemented and the complemented value is multiplied with **Y** to get **YZ'**.
- ➢ This column will have the value 1 only when both **Y** is a 1 and **Z'** is a 1.
- ➢ Now the value of **YZ'** is performed logical addition with the value of **X** and the final value is evaluated.

BASIC LAWS OF BOOLEAN ALGEBRA

- A list of basic rules by which Boolean algebra expressions may be manipulated are contained in this table.
- Each rule may be proved by using the proof by perfect induction.

BOOLEAN ALGEBRA RULES

TABLE 5 Boolean Identities.

Identity	*Name*
$\overline{\overline{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)x$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \overline{x} + \overline{y}$ $\overline{(x + y)} = \overline{x}\,\overline{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \overline{x} = 1$	Unit property
$x\overline{x} = 0$	Zero property

SIMPLIFICATION OF EXPRESSIONS

The rules may be used to simplify Boolean expression.

Consider the expression **(X + Y)(X+Y')(X'+Z).**

This can be simplified by

(X + Y) (X + Y) (X + Z)

Consider the two terms,

(X+Y) (X+Y')

XX + XY' + XY + YY'
X + XY' + XY + 0 **(YY = 0, XX = X)**
X + X(Y' +Y) **(Y' + Y = 1)**
X + X(1)
X + X **(X+X=1)**

X

Now multiply the term **(X + Z)**

X(X'+Z) X' + XZ.

XZ. **XX' =0**

So the expression **(X + Y) (X+Y') (X'+Z)** is reduced to **XZ**.

$$\overline{(A\overline{B} + \overline{A}B)}(A + B) = \overline{A\overline{B}}\,\overline{\overline{A}B}(A+B)$$
$$= (\overline{A}+B)(A+\overline{B})(A+B)$$
$$= (\overline{A}+B)(AA+AB + \overline{B}A + \overline{B}B)$$
$$= (\overline{A}+B)(A +AB +A\overline{B} + \overline{B}B)$$
$$= (\overline{A}+B)(A(1 + B + \overline{B}) + \overline{B}B)$$
$$= (\overline{A}+B)(A(1) + \overline{B}B)$$
$$= (\overline{A}+B)A$$
$$= A\overline{A} + AB$$
$$= AB$$

- Ex. Simplify - AB + A(B + C) + B(B + C)
- Solution - AB + AB + AC + BB + BC

  AB + AB + AC + B + BC

  AB + AC + B + BC

  AB + AC + B

  B+AC

- $(A + B)(A + C) = A + BC$
- This rule can be proved as follows:
- $(A + B)(A + C) = AA + AC + AB + BC$ (Distributive law)

  $= A + AC + AB + BC$ $(AA = A)$

  $= A(1 + C) + AB + BC$ $(1 + C = 1)$

  $= A.1 + AB + BC$

  $= A(1 + B) + BC$ $(1 + B = 1)$

  $= A.1 + BC$ $(A . 1 = A)$

  $= A + BC$

DEMORGAN'S THEOREM

Demorgan's theorem is very useful to design circuits in Boolean algebra.

The following two rules are the Demorgan's theorem.

1. X' + Y' = X' . Y'

2. (X . Y)' = X' + Y'

- The complement of any Boolean expression or a part of any expression may be found by means of these theorems.
- Two steps are used to form a complement.
  1. The (+) symbols are replaced with (.) and (.) symbol are replaced with (+) symbol.
  2. Each of the terms in the expression is complemented.

The complement of W'X+YZ' is done by two steps:

- ✓ The addition symbol is changed
- ✓ The complement of each term is formed. Ex:

(W'.X)'(Y.Z')'

can be written as

(W+X')(Y'+Z)

Since W and Z were already complemented, they become uncomplemented by the theorem

X' = X.

It is sometimes necessary to complement both sides of an equation. This may be done in the same way as before:

WX+YZ =0

Complementing both sides gives

(WX+YZ)' = 0'

(W'+X')(Y'+Z') =1

TRUTH TABLE FOR DEMORGAN'S THEOREM

INPUTS		OUTPUT
X	Y	Z
0	0	1
0	1	0
1	0	1
1	1	1

BASIC DUALITY OF BOOLEAN ALGEBRA

- The postulates and theorems which have been presented can all be divided into pairs.
- Boolean algebra is defined as **(0,1)** and by the two binary operator **(+, .)**

Example:

i. **(X+Y)+Z = X+(Y+Z)** is the dual of **(XY)Z = X(YZ)**
ii. **X + 0 =X** is the dual of **X. 1 = X.**
iii. **X + X = X** is the dual **X.X =X**
iv. **X +Y = Y + X** is the dual **X .Y = Y. X**

DERIVATION OF A BOOLEAN EXPRESSION

When designing a logical circuit, the logical designer works from two sets of known values.

1. The various states which the inputs to the logical network can take , and
2. The desired outputs for each input condition.

The logical expression is derived from these sets of values.

The truth table for two inputs X and Y in a logical network to give an output Z is as follows:

X	Y	Z
0	0	1
0	1	0
1	0	1
1	1	1

- It is necessary to add another column to the table.
- This consists of list of product terms obtained from input variables.
- The input is complemented when the input value is 0 and not complemented when the value is 1.

X	Y	Z	Product of terms
0	0	1	(XY)'
0	1	0	(XY)'
1	0	1	(XY)'
1	1	1	(XY)'

- Whenever Z is equal to 1, the X and Y product term from the same row is removed and formed into sum of products.

- The table for the expression **X + Y** is evaluated.

X	Y	Y'	X + Y'
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

- The last column of this table agrees with the last column of the desired function (the value of Z) and they are equivalent.

  There are now three terms, each product of two variables. The logical sum of these is equal to the expression desired. This type of expression is often referred to as canonical expansion for the function.

- The complete expression in normal form is

$$(XY)' + XY' + XY = Z$$

The left-hand side of the expression may be simplified as follows:

$$(XY)' + X\,(Y' + Y) = Z$$

$(XY)' + X\,(1) = Z$ $(XY)' + X = Z$

X + Y '= Z

Truth table for three input values X,Y and Z

X	Y	Z	Output A
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

A column is added to listing the inputs, A, Y,and Z according to their values in the input columns.

The product terms from each row in which the output is a 1 are collected ((XYZ)', X'YZ', XY'Z', and XYZ') and the desired expression is the sum of these products (X'Y'Z'+ X'YZ'+XY'Z'+XYZ').

Therefore, the complete expression in standard form for the desired network is

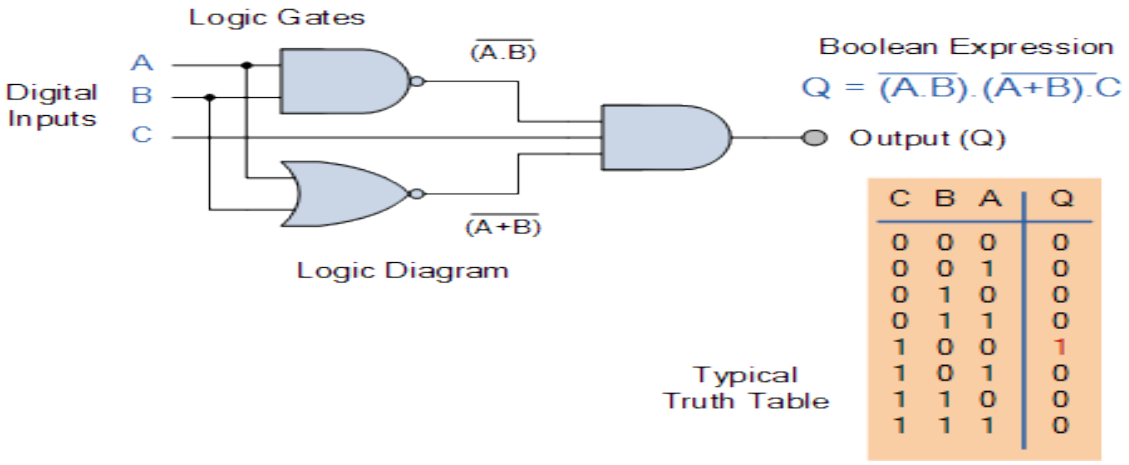
X'Y'Z' + X'YZ '+ XY'Z' + XYZ' =A

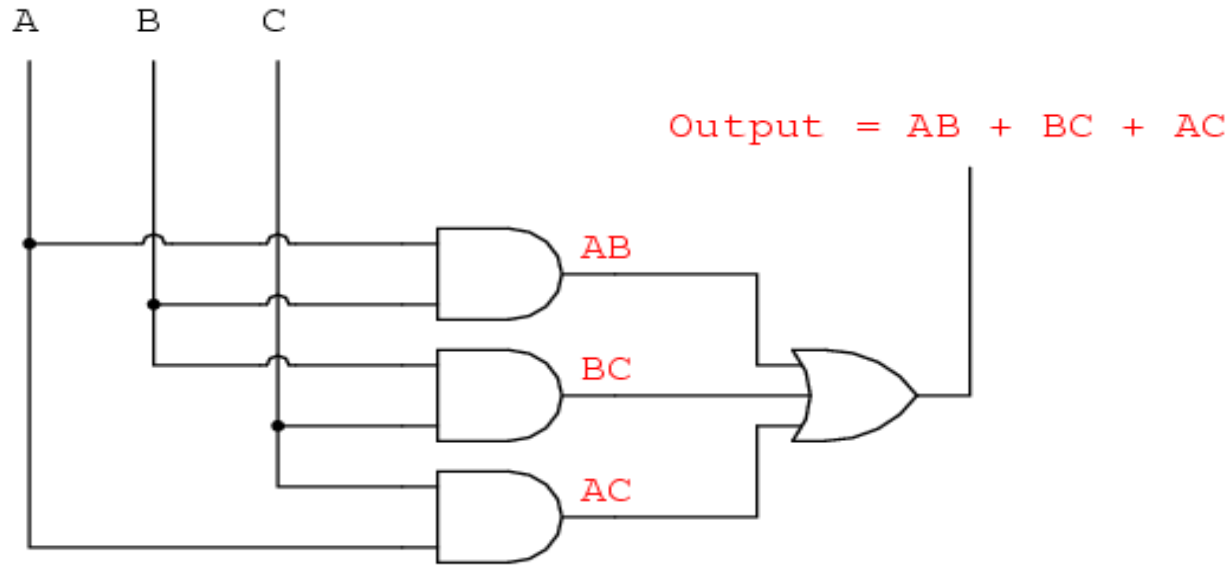
X'(Y'Z'+YZ')+X(Y'Z'+YZ') =A
X'(Z'(Y'+Y))+X(Z'((Y'+Y)) =A
X'Z'+XZ' =A
Z' =A

UNIT -- III

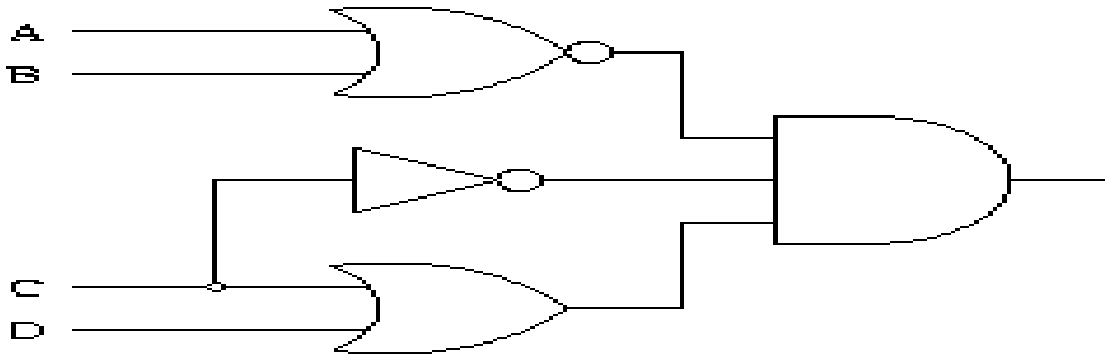
Interconnecting Gates – Sum of products (SOP) and Products of sums (POS) – Derivation of products of sums expressions – Derivation of three Input variable expression – NAND gates and NOR gates - The Map method for simplifying expressions – Sub cube and covering – product of sums expressions – Don't cares.

Interconnecting Gates





what is the logic expression following logic circuit



Sum of products (SOP)

- An SOP expression when two or more product terms are summed by Boolean addition.

Examples:

$$AB + ABC$$

$$ABC + CDE + \overline{B}C\overline{D}$$

$$\overline{A}B + \overline{A}B\overline{C} + AC$$

Also:

$$A + \overline{A}\overline{B}C + BC\overline{D}$$

- In an SOP form, a single over bar cannot extend over more than one variable; however, more than one variable in a term can have an over bar:

  Examples:

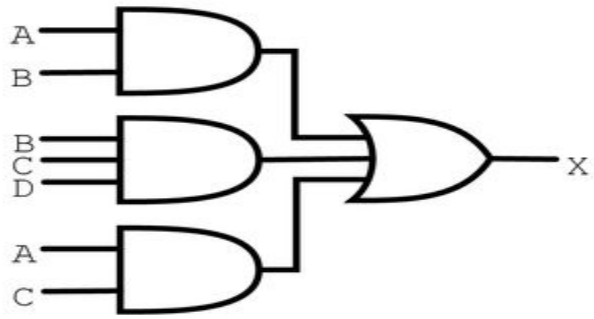
  $\overline{A}\overline{B}\overline{C}$ is ok

  But not:

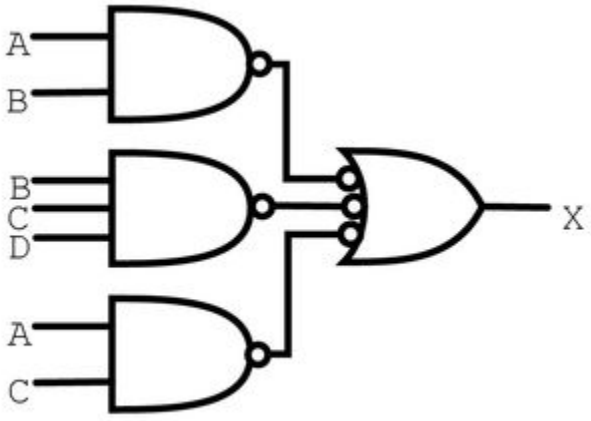
  $\overline{ABC}$

Implementation of an SOP

- $X=AB+BCD+AC$
- AND/OR implementation



- NAND/NAND implementation



General Expression → SOP

- Any logic expression can be changed into SOP form by applying Boolean algebra techniques.
  ex:
  A(B + CD) = AB + ACD
  AB + B(CD + EF) = AB + BCD + BEF
  (A + B)(B + C + D) = AB + AC + AD + BB + BC + BD
  (A + B) + C = (A + B)C = (A + B)C = AC + BC

The Product-of-Sums (POS)

- When two or more sum terms are multiplied, the result expression is a product-of-sums (POS):
  - Examples:

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

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

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

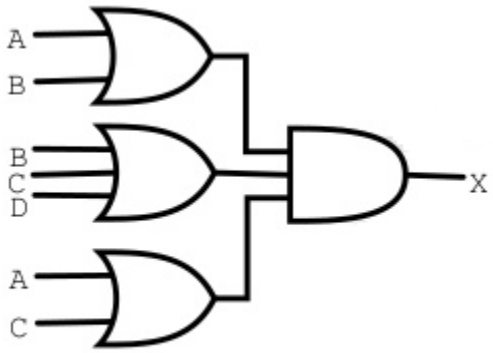
  - Also:

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

- In a POS form, a single over bar cannot extend over more than one variable; however, more than one variable in a term can have an over bar:
  - Examples:
    $\overline{A}+\overline{B}+\overline{C}$ is ok
  - Also:
    $\overline{A+B+C}$

Implementation of a POS

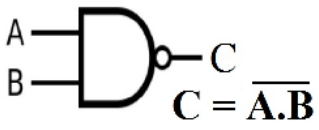
- ***X=(A+B)(B+C+D)(A+C)***
- **OR/AND implementation**



NAND Gate

- Complemented AND gate
- Generates an output signal of:
  - 1 if any one of the inputs is a 0
  - 0 when all the inputs are 1

NAND GATE



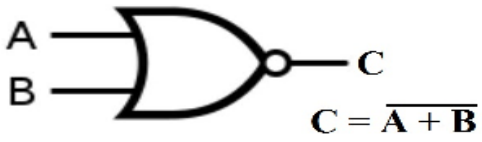
Truth Table

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

- Complemented OR gate
- Generates an output signal of:
  - 1 only when all inputs are 0
  - 0 if any one of inputs is a 1

NOR GATE



TRUTH TABLE

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

The Karnaugh Map

- **Feel a little difficult using Boolean algebra laws, rules, and theorems to simplify logic?**
- **A K-map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the <u>minimum expression</u>.**

What is K-Map

- **It's similar to truth table; instead of being organized (i/p and o/p) into columns and rows, the K-map is an array of cells in which each cell represents a binary value of the input variables.**
- **The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells.**
- **K-maps can be used for expressions with 2, 3, 4, and 5 variables.**
  - **3 and 4 variables will be discussed to illustrate the principles.**

Properties

- An n-variable K-map has 2^n cells with n-variable truth table value.
- Adjacent cells differ in only one bit .
- Each cell refers to a minterm or maxterm.
- For minterm m_i, maxterm M_i and don't care of f we place 1 , 0 , x .

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

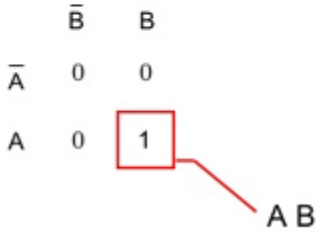
ABCD	ABCD
0000 - 0	1000 - 8
0001 - 1	1001 - 9
0010 - 2	1010 - 10
0011 - 3	1011 - 11
0100 - 4	1100 - 12
0101 - 5	1101 - 13
0110 - 6	1110 - 14
0111 - 7	1111 - 15

Simplification Process

- No diagonals.
- Only 2^n cells in each group.
- Groups should be as large as possible.
- A group can be combined if all cells of the group have same set of variable.
- Overlapping allowed.
- Fewest number of groups possible.

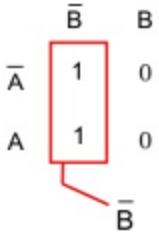
Two Variable K-map Grouping

Groups of One – 4

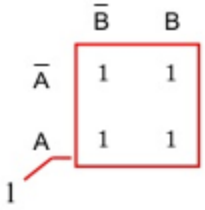


Two Variable K-Map Groupings

Groups of Two – 2

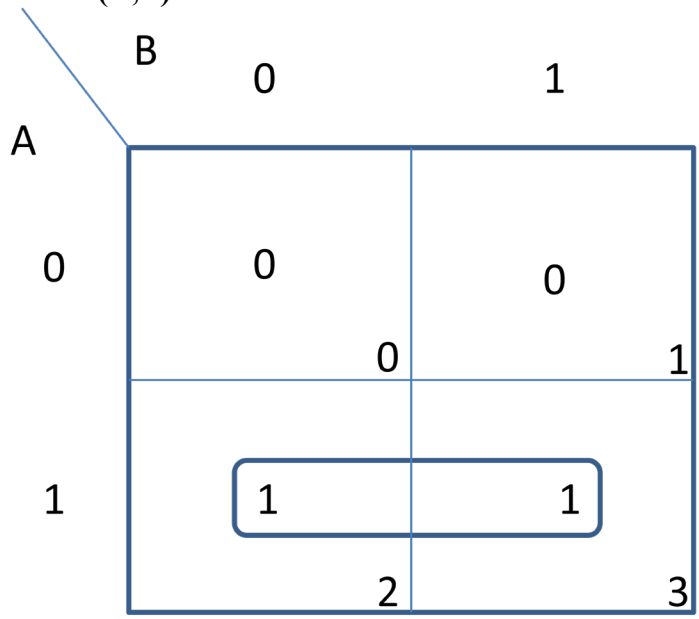


Group of Four

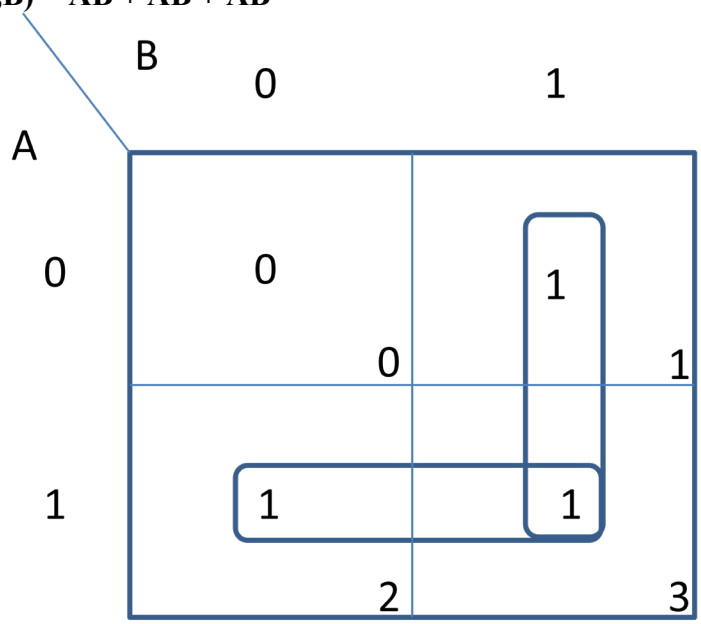


The 2 Variable K-Map

- **F(A,B) = AB + AB**

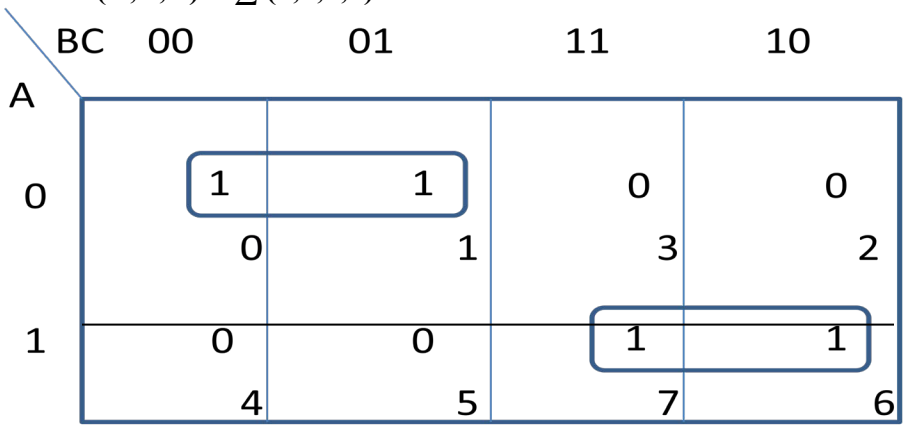


- **F(A,B) = AB + AB + AB**

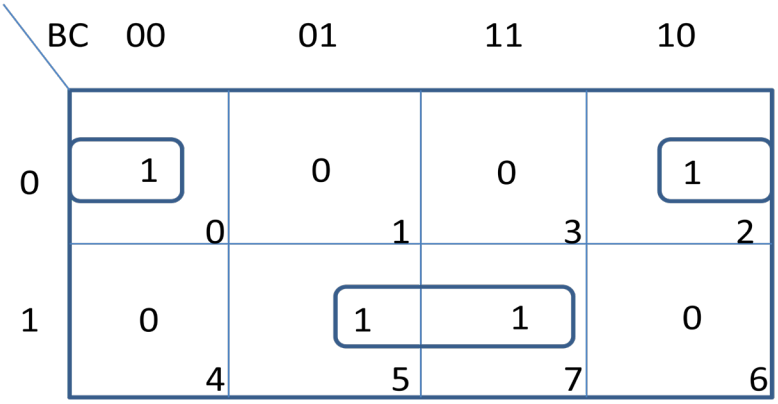


The 3 Variable K-Map

- **F(A,B,C) = ∑ (0,1,6,7)**



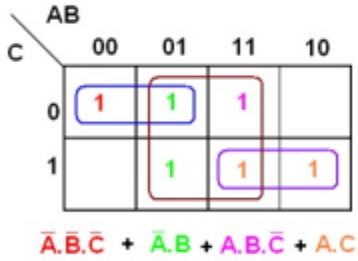
- **F(A,B,C) = ∑ (0,2,5,7)**



Karnaugh Mapping Worked Example

Solving problems using the Karnaugh mapping.

$$Z = f\ A,B,C\ = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B + A.B.\overline{C} + A.C$$



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

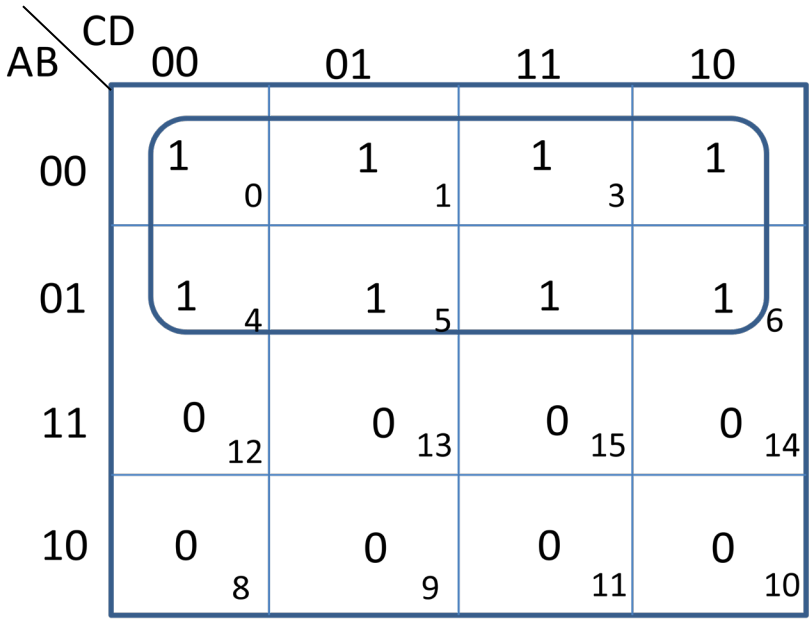
The 3 Variable K-Map

- **There are 8 cells as shown:**

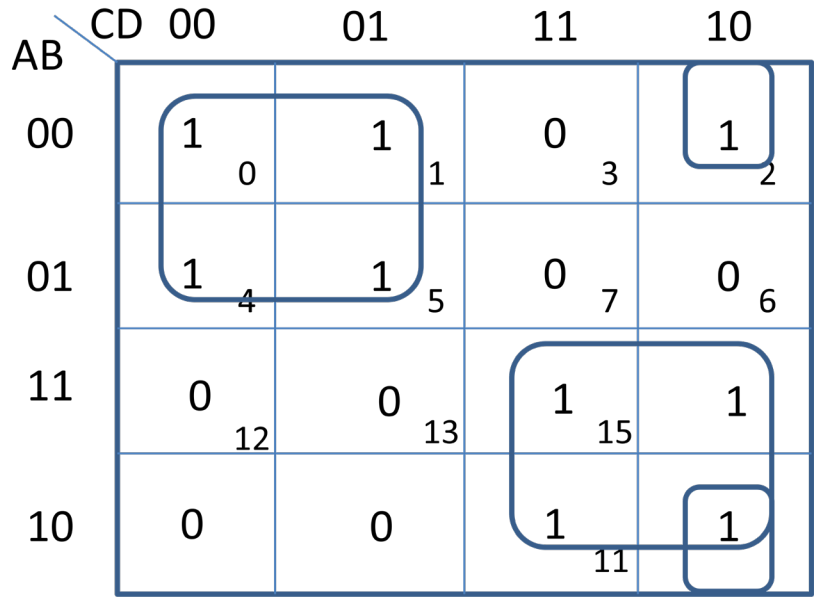
AB \ C	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\,\overline{C}$	$A\overline{B}C$

The 4-Variable K-Map

- **F(A,B,C,D) = $\sum$ (0,1,2,3,4,5,6,7)**



- **F(A,B,C,D) = ∑ (0,1,2,4,5,10,11,14,15)**

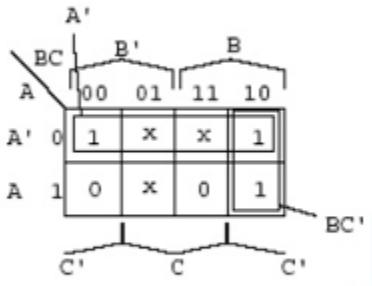


The 4-Variable K-Map

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Don't-care condition

- Minterms that may produce either 0 or 1 for the function.
- Marked with an 'x' in the K-map.
- These don't-care conditions can be used to provide further simplification.



Don't Care Conditions

Simplified sum-of-products (SOP) logic expression for the logic function F_4.

R	S	T	U	F_4
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	X
0	1	0	0	0
0	1	0	1	X
0	1	1	0	X
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	X
1	1	0	0	X
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

	$\bar{T}\bar{U}$	$\bar{T}U$	TU	$T\bar{U}$
$\bar{R}\bar{S}$	X	0	X	1
$\bar{R}S$	0	X	1	X
RS	X	0	0	0
$R\bar{S}$	1	1	X	1

$\bar{R}T$ and $R\bar{S}$

$$F_4 = \bar{R}T + R\bar{S}$$

UNIT -- IV

Microprocessors, Microcomputers and Assembly Language: Microprocessors - Microprocessor instruction set and Computer Languages-From large computers to single chip Microcontrollers; Microprocessor Architecture and Microcomputer systems: Microprocessor Architecture and its operations – Memory – I/O devices; 8085 Microprocessor Architecture and Interfacing: The 8085 MPU – Examples of a 8085 based Microcomputer – Memory interfacing

MICROPROCESSORS

- The word comes from the combination micro and processor.
- In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
- Micro is a new addition.
- In the late 1960's, processors were built using discrete elements.
- In the early 1970's the microchip was invented.
- All of the components that made up the processor a single piece of silicon.
- The size became several thousand times smaller and the speed became several hundred times faster. The "Micro" Processor was born.

The microprocessor is a **programmable device that takes in numbers**, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a result.

PROGRAMMABLE DEVICE:

- The microprocessor can perform different sets of operations on the data it receives depending on the sequence of instructions supplied in the given program.
- By changing the program, the microprocessor manipulates the data in different ways.

Instructions:

- Each microprocessor is designed to execute a specific group of operations.
- This group of operations is called an instruction set.

Takes in:

- The data that the microprocessor manipulates must come from somewhere.
- It comes from what is called "input devices".
- These are devices that bring data into the system from the outside world.
- These represent devices such as a keyboard, a mouse, switches, and the like.

NUMBERS:

- It only understands binary numbers.
- A binary digit is called a **bit** (which comes from **b**inary dig**it**).
- The microprocessor recognizes and processes a group of bits together. This group of bits is called a "**word".**

They processed information 8-bits at a time.

Later microprocessors (8086 and 68000) were designed with 16-bit words.

A group of 8- bits were referred to as a **"half-word" or "byte".**

A group of 4 bits is called a **"nibble".**

Also, 32 bit groups were given the name **"long word".**

Today, all processors manipulate at least 32 bits at a time and there exists microprocessors that can process 64, 80, 128 bits Words, Bytes, etc. The earliest microprocessor (the Intel 8088 and Motorola's 6800) recognized 8-bit words.

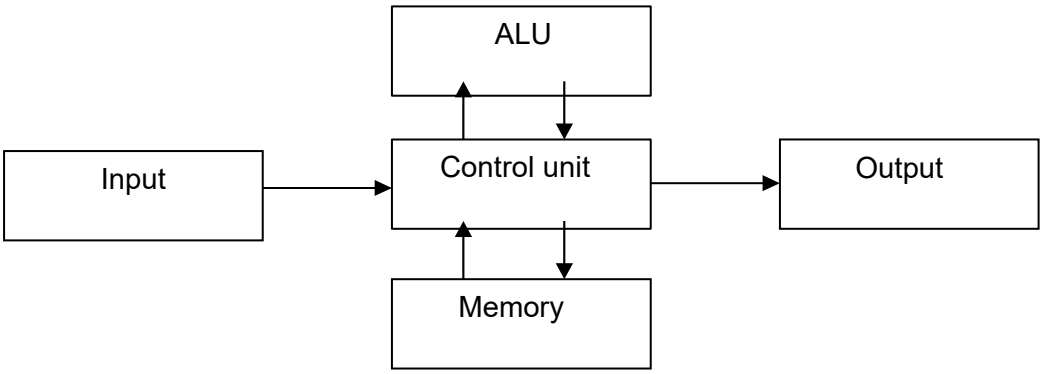
Memory is usually measured by the number of bytes it can hold. It is measured in Kilos, Megas and lately Gigas. A Kilo in computer language is 210 = 1024. So, a KB (KiloByte) is 1024 bytes. Mega is 1024 Kilos and Giga is 1024 Mega.

- Computer-a programmable machine that processes binary data.
- It includes four components:
- CPU (ALU plus control unit), memory, input, and output.

Microprocessor Digital

· **CPU**-the Central Processing Unit. The group of circuits that processes data and provides control signals and timing. It includes the arithmetic/logic unit, registers, instruction decoder, and the control unit.

Block diagram of a computer



· **ALU**-the group of circuits that performs arithmetic and logic operations. The ALU is a part of the CPU.

· **Control Unit**-The group of circuits that provides timing and signals to all operations in the computer and controls data flow.

· **Memory**-a medium that stores binary information (instructions and data).

· **Input** -a device that transfers information from the outside world to the computer

MICROPROCESSOR INSTRUCTION SET AND COMPUTER LANGUAGES

- Microprocessors recognize and operate in binary numbers.
- However, each microprocessor has its own binary words, instructions, meanings, and language.
- The words are formed by combining a number of bits for a given machine.

MACHINE LANGUAGE

The number of bits in a word for a given machine is fixed, and words are formed through various 'combinations of these bits. For example; a machine with a word length of eight bit scan have 756 (2^8) combinations of eight bits-thus a language of 256 words.

8085 MACHINE LANGUAGE

The Z80 is a microprocessor with 8-bit word length .Its instruction set (or language) is upward compatible with that of the 8080; the Z80 has 158 instruction types that include the entire 8080 set of 72 instruction types.

8085 ASSEMBLY LANGUAGE

Even though the instructions can be written in hexadecimal code, it is still not easy to understand such a program. Therefore, each manufacturer of microprocessors has devised a symbolic code for each instruction, called a mnemonic.

ASCII Codes

A computer is a binary machine; in order to communicate with the computer in alphabetic letters and decimal numbers. translation codes are necessary. The commonly used code is known as ASCII-American Standard Code for Information Interchange..

Another code, called EBCDIC (Extended Binary Coded Decimal Interchange Code) is widely used in IBM computers

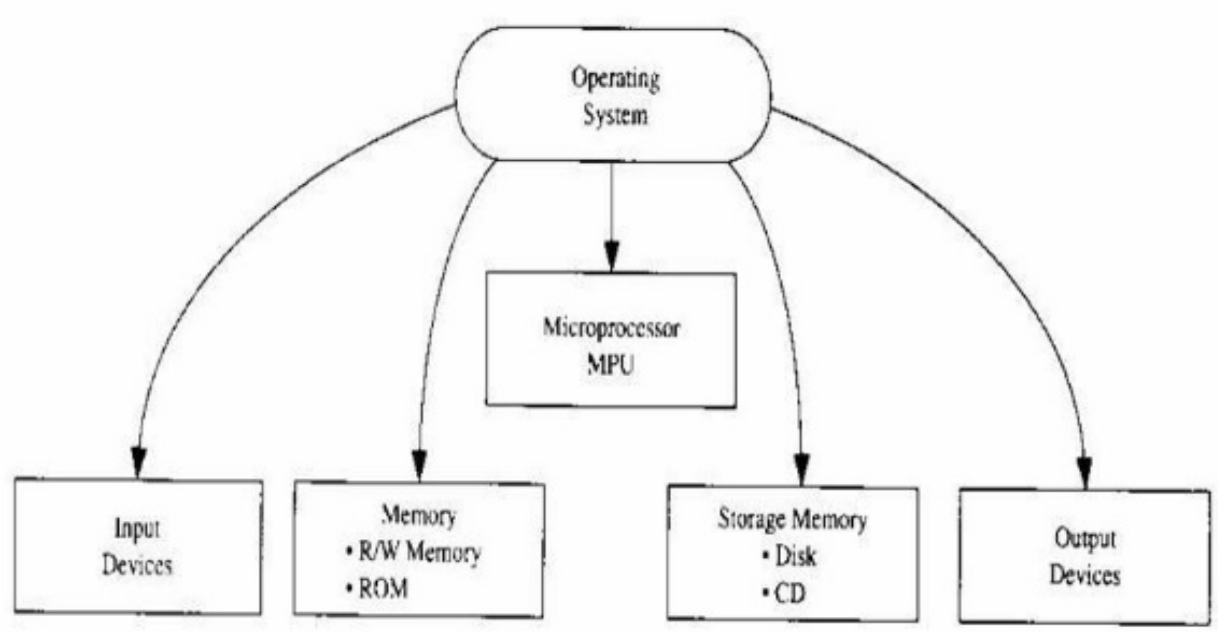
HIGH-LEVEL LANGUAGES

Programming languages that are intended to be machine-independent are called high-level languages. The list includes such languages as C, FORTRAN, BASIC, PASCAL, and COBOL.

Each microprocessor needs its own compiler or interpreter for each high-level language.

OPERATING SYSTEMS:

The interaction between the hardware and the software is managed by a set of programs called an operating system of a computer.



FROM LARGE COMPUTERS TO SINGLE-CHIP MICROCONTROLLERS

Different types of computers are designed to serve different purposes. some are suitable for scientific calculations, while others are used simply for turning appliances on and off.

In 1970s, computers were broadly classified in three categories as Mainframe, Mini and Microcomputers..

LARGE COMPUTERS:

These are large, general-purpose, multi-user, multitasking computers designed to perform such data processing tasks as complex scientific and engineering calculations and handling of records for large corporations or government agencies.

MEDIUM-SIZE COMPUTERS:

In the 1960s, these computers were designed to meet the instructional needs of small

colleges, the manufacturing problems of small factories, and the data processing tasks of medium-size businesses, such as payroll and accounting. These were called mini-computers. These machines were slower and smaller in memory capacity than mainframes.

MICRO COMPUTERS:

The 4-bit and 8-bit microprocessors became available in the nid-1970s, and initial applications were primarily in the areas of machines control and instrumentation. Present- day microcomputers can be classified in four groups: personal (or business) computers (PC), work stations, single-board, and single-chip microcomputers (microcontrollers).

PERSONAL COMPUTERS (PC):

These microcomputers are single-user system and are for a variety of purposes such as payroll, business accounts, word processing, legal and medical record keeping, personal finance, accessing internet resources (e-mail, web search, newsgroup), and instruction. They are also known as personal computers **(PC)** or desktop computers.

WORKSTATIONS:

These are high-performance cousins of the personal computers. They are used in engineering and scientific applications such as computer-aided design (CAD), computer- aided engineering (CAE), and computer-aided manufacturing (CAM).They generally include system memory and storage (hard disk) memory in gigabytes, and a high-resolution screen. The RISC processors tend to be faster and more efficient than the processor used in personal computers.

SINGLE-BOARD MICROCOMPUTERS:

These microcomputers are primarily used in college laboratories and industries for instructional purpose or to evaluate the performance of a given microprocessor. They can also be part of some larger system.

SINGLE CHIP MICROCOMPUTERS:-

These microcomputers are designed on a single chip, which typically includes a micro processor, 256 bytes of R/W memory, from 1K to 8K bytes of ROM, and several signal lines to connect I/Os.

Microprocessor architecture and microcomputer systems

- The Microprocessor is a programmable digital device, designed with registers, flip-flops and timing elements.
- It has a set of instructions, designed internally to manipulate data and communicate with peripherals.
- The process of data manipulation and communication is determined by the logic design of the microprocessor, called the architecture.
- All the various functions performed by the microprocessor can be classified in three general categories.

- ✓ Microprocessor-initiated operations
- ✓ Internal operations
- ✓ Peripheral operations

❖ The term micro processing unit (MPU) is defined as a group of devices that can perform these functions with the necessary set of control signals.

❖ This is similar to the term central processing unit (CPU).

Microprocessor initiated operations and 8085 bus organization

❖ The MPU performs primarily four operations.

1. Memory Read: Reads data from memory.
2. Memory Write: Writes data from memory.
3. I/O Read: Accepts data from input devices.
4. I/O Write: Sends data to output devices.

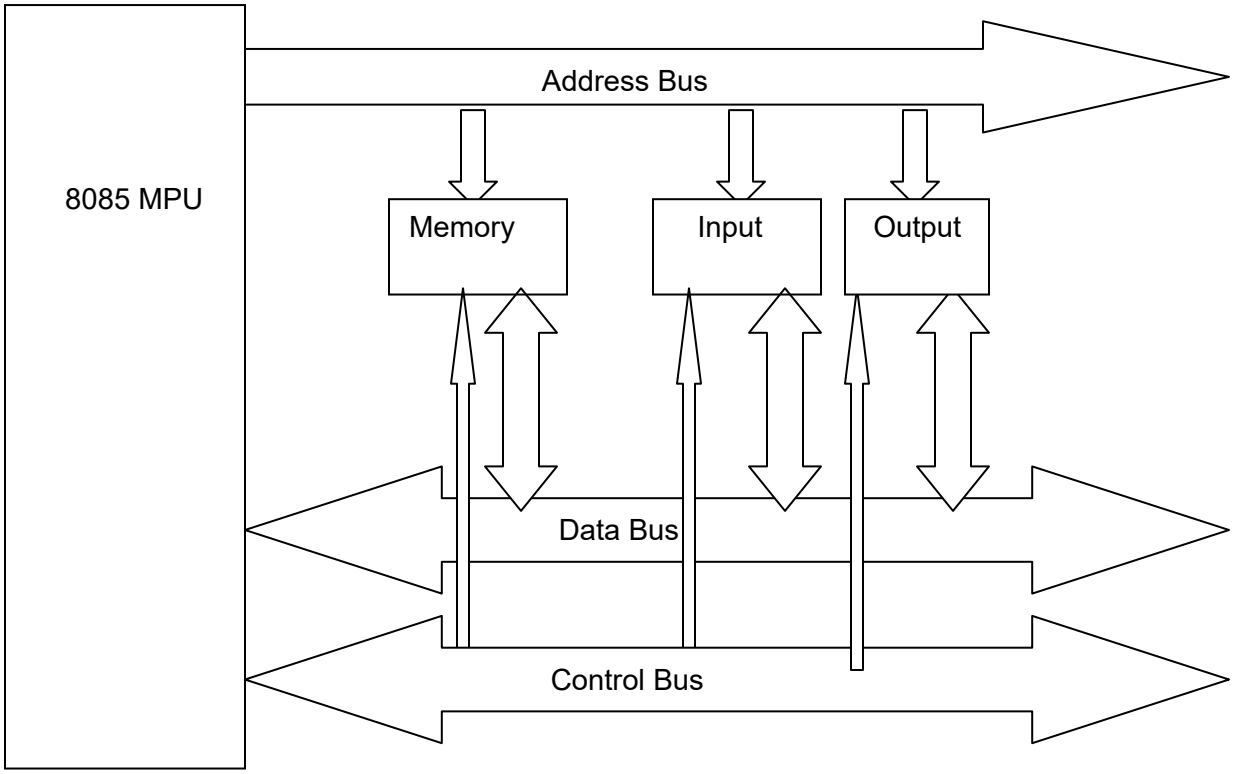
❖ These are communication process between the MPU and peripheral devices.

❖ To communicate with the peripheral (or a memory location), the MPU needs the following steps.

Step 1: Identify the peripheral or the memory location with its address. Step 2: Transfer binary information.

Step 3: Provide timing or synchronization signals.

To perform these functions using three sets of communication lines called buses



Address bus

- Group of 16 lines identified as A0 to A15.
- The address bus is unidirectional. Bits flow in one direction- from the MPU to peripheral devices.
- The MPU uses this bus to perform step1.
- In computer, each peripheral or memory location is identified by a number called an address, used to carry 16-bit address.

Data bus

- Group of 8 lines are used for data flow.
- Bi-directional data flow in both direction, between the MPU and memory and peripheral devices.
- The MPU uses this bus to perform step2.
- The 8 lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (2^8 = 256 numbers).

Control bus

- This bus is comprised of various single lines that carry synchronization signals.
- The MPU uses this bus to perform step3.
- The MPU generates specific control signals for every operation it performs.
- The MPU sends a pulse called Memory Read as the control signal.
- The pulse activates the memory chip and the contents of the memory location are placed

on the data bus.

Input and output devices (I/O) devices

- Input/output devices are the means through which the MPU communicates with " the outside world".
- The MPU accepts binary data as input from devices such as keyboards and A/D converters and sends data to output devices such as LEDs or printers.

I/Os with 8-bit addresses (peripheral-mapped I/O)

- The MPU uses eight address lines to identify an input or an output device. This is known as peripheral-mapped I/O.
- The eight address lines can have 256 (2 combinations) address. MPU can identify 256 input devices and 256 output devices with addresses ranging from 00H to FFH.
- The input and output devices are differentiated by the control signals, the MPU uses the I/O read control signals for input devices and I/O write control signals for output device.
- The steps in communicating with an I/O device are as follows.
  - ✓ The MPU places an 8-bit address on the address bus, which is decoded by external decode logic.
  - ✓ The MPU sends a control signal (I/O read or I/O write) and enables the I/O device.
  - ✓ Data are transferred using the data bus.

I/O with 16-bit addresses (Memory-Mapped I/O)

- The MPU uses 16 addresses lines to identify an I/O device.
- An I/O is connected as if it is a memory register known as memory mapped I/O.
- The MPU uses the same control signal and instructions as those of memory.
- In some microprocessors, all I/Os have 16-bit addresses. I/Os and memory share the same memory map.

8085 microprocessor architecture and memory interfacing

The 8085 MPU

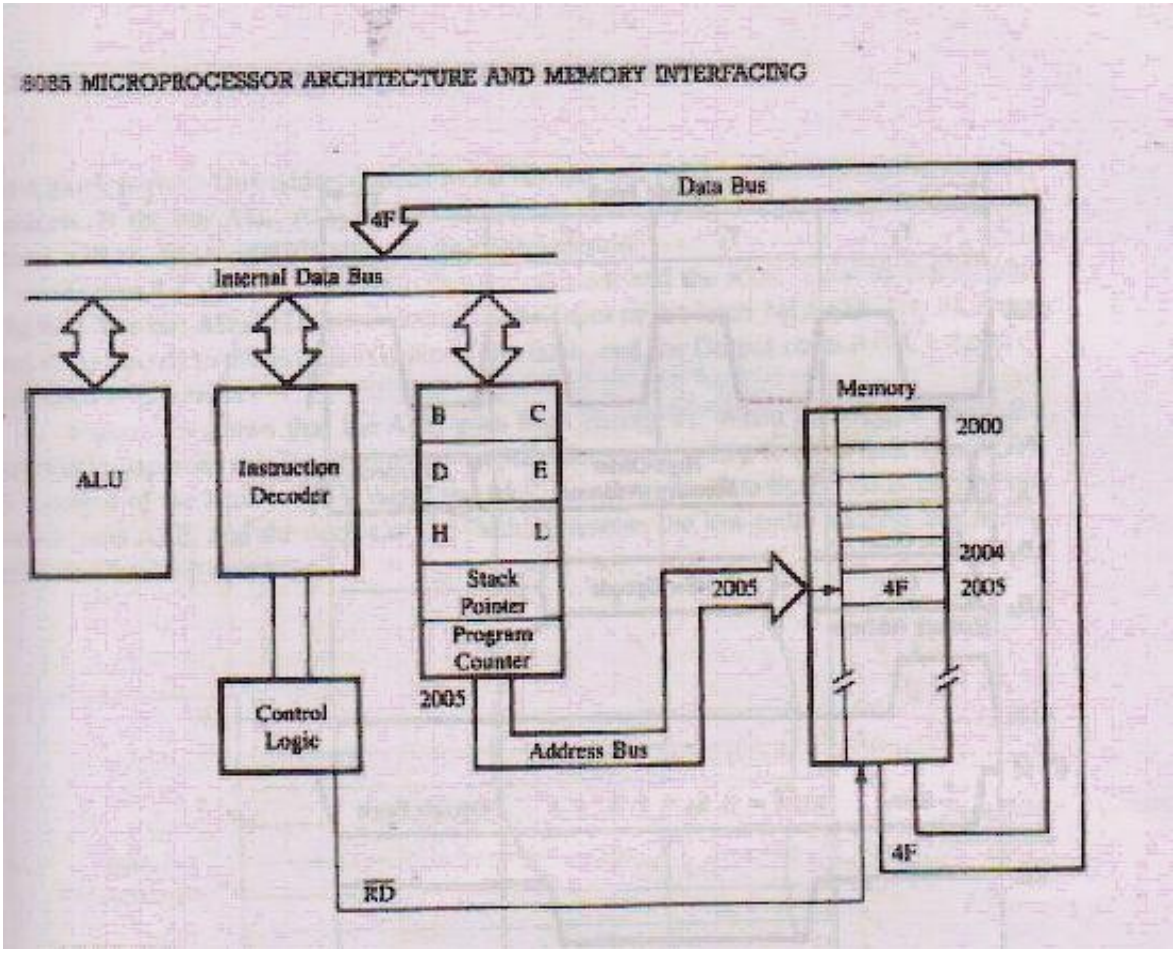
- The term micro processing unit (MPU) is similar to the term central processing unit (CPU).
- The MPU is a device or a group of devices that can communicate with peripherals provide timing signals, direct data flow, and perform computing tasks as specified by the instructions in memory.
- The 8085 microprocessor can almost qualify as an MPU, but with the following two limitations.
  - ✓ The low-order address bus of the 8085 microprocessor is multiplexed (time – shared) with the data bus. The buses need to be multiplexed.
  - ✓ Appropriate control signals need to be generated to interface memory and I/O with the 8085.

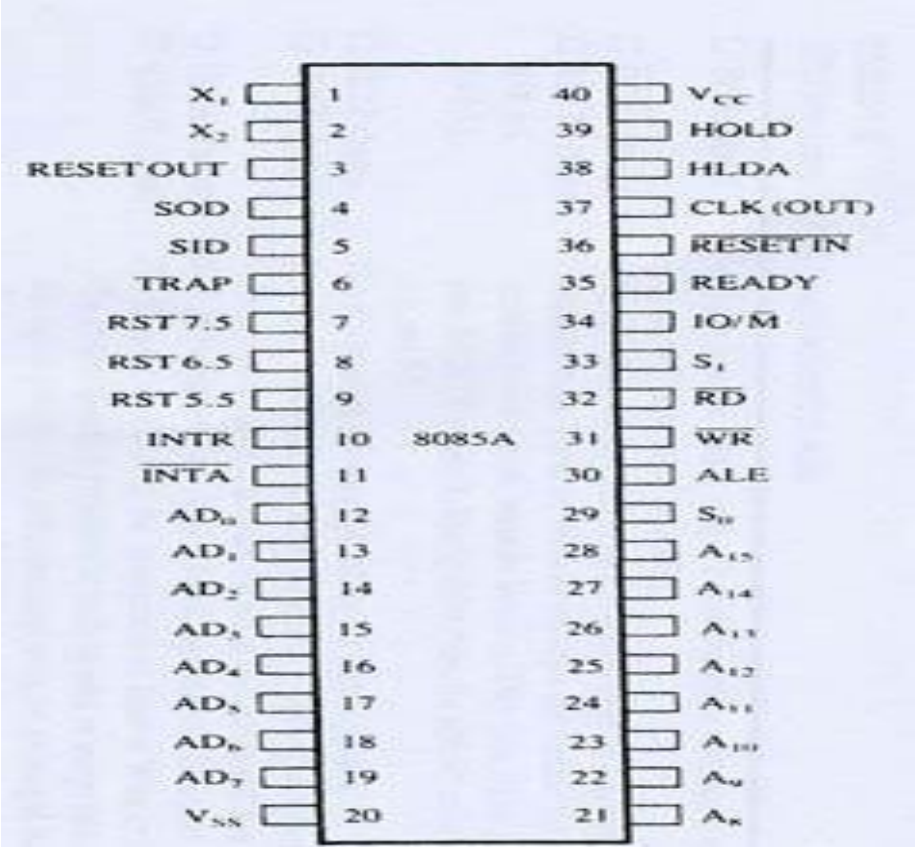
8085 microprocessor

- The 8085A is an 8-bit general-purpose microprocessor capable of addressing 64K of memory.
- The device has forty pins, requires a +5v single power supply and can operate with a 3-

MHz single-phase clock.

- The 8085 is an enhanced version of its predecessor 8080A meaning that 8085 instruction set includes all the 8080A instructions and some additional instructions.
- The entire signal can be classified in to six groups.
    - Address – bus
    - Data-bus
    - Control and status signals
    - Power supply and frequency signals
    - Externally initiated signals





Address

The 16 signal lines which are used as the address bus are split in to two segments.

A15-A8 are unidirectional and used for the most significant bits called the high-order address.

Multiplexed address/databus

The signal lines AD7-AD0 are bidirectional. They serve a dual purpose. They are used as the low-order address bus as well as the data bus.

During the earlier part, they are used as low-order bus. During the later part, they are used as databus.

Control and Status Signals

This includes two control signals (RD and WR) three status signals (I/O, S1 and S0) and one special signal (ALE) to indicate the beginning of the operation.

ALE – Address LatchEnable

This signal is used primarily to latch the low-order address from the multiplexed bus

and generate a separate set of eight address lines A7 – A0.

RD – Read

This signal indicates that the selected I/O or memory device is to be read and data are available on the data bus.

WR – Write

This signal indicates that the data on the data bus are to be written into a selected memory or I/O location.

IO/M

This signal is used to differentiate between I/O and memory operations. When it is high, it indicates an I/O operation; when it is low, it indicates a memory operation.

S1 and S0

These signal signals, similar to IO/M can identify various operations, but they are rarely used in small systems.

Power Supply and clock Frequency

Vcc - +5v power supply

Vss – ground reference

X1, X2: a crystal is connected at three two pins.

CLK (OUT) – clock output. This signal can be used as the system clock for other devices.

EXTERNALLY INITIATED SIGNALS, INCLUDING INTERRUPTS

The 8085 have five interrupt signals that can be used to interrupt a program execution. In addition to the interrupts, three pins-RESET, HOLD and READY-accept the externally initiated signals as inputs.

INTR – interrupt request.

This is used as a general purpose interrupts; it is similar to the INT signal of 8080A. INTA- interrupt acknowledge.

it is used to acknowledge an interrupt. RST 7.5, RST 6.5, RST 5.5 – restart interrupts.

It transfers the program control to specific memory locations. They have higher priority than the INTR interrupt.

TRAP

Non-maskable, highest priority interrupt.

HOLD

Indicates that a peripheral is requesting the use of the address and data buses.

HLDA – hold acknowledge:

Acknowledges the Hold request.

READY

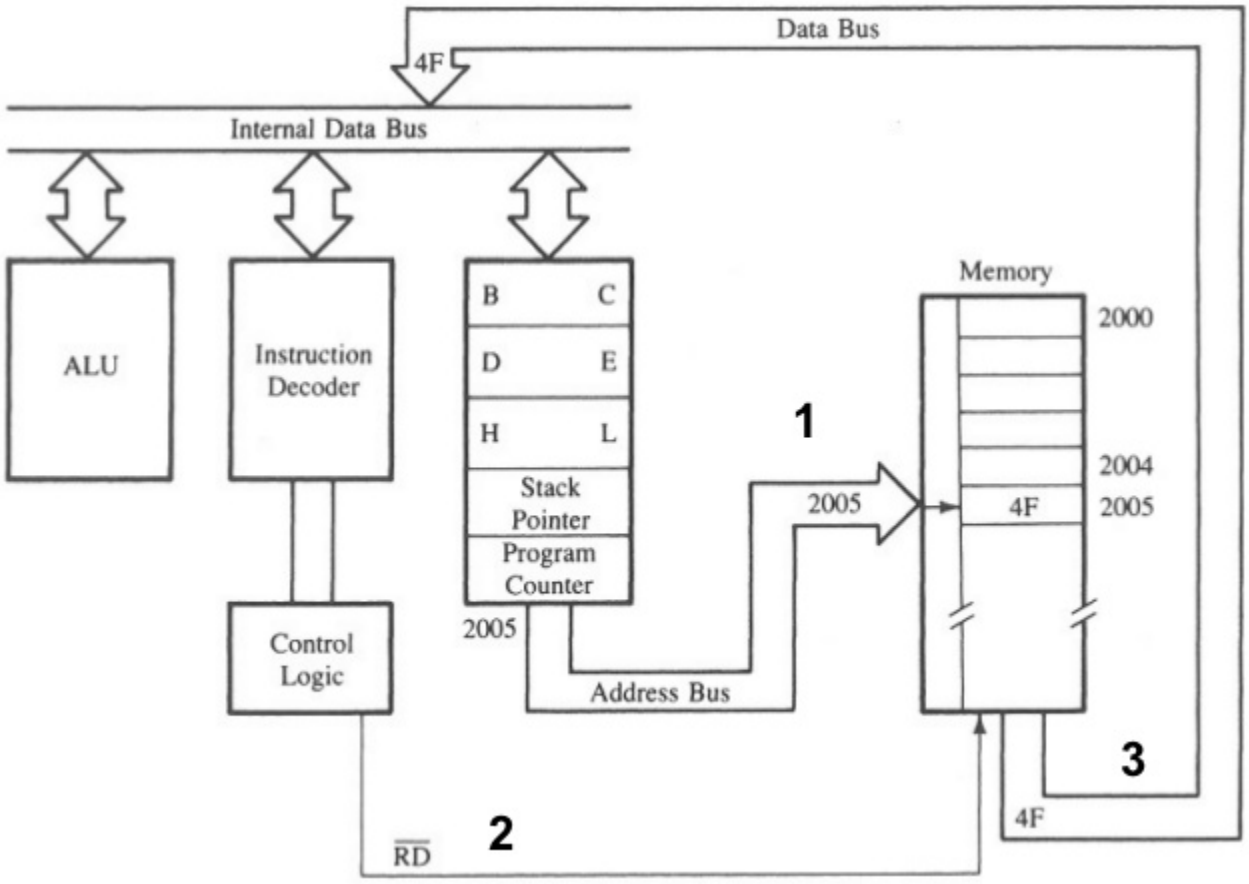
This signal is used to delay the microprocessor read and write cycles until a slow responding peripheral is read to send or accept data.

RESET IN

When the signal on this pin goes low the program counter is set to zero.

RESET OUT

ThissignalindicatesthattheMPUisbeingreset.Thissignalcanbeusedtoresetotherdevices.



DEMULTIPLEXING THE BUS AD7 – AD0

- The address on the high-order bus (20H) remains on the bus for three clock periods. However, the low-order address (05H) is lost after the first clock period.
- This address needs to be latched and used fro identifying the memory address. If the bus AD7- AD0 is used to identify the memory location (2005H), the address will change after first clock period.

Instruction cycle

It is defined as the time required to complete the execution of an instruction. The 8085 instruction cycle consists of one to six machine cycles or one to six operations.

Machine cycle

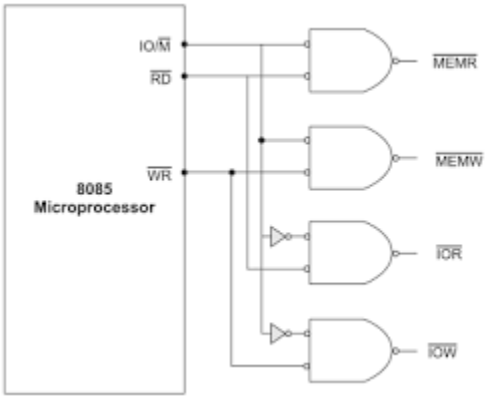
It is defined as the time required to complete one operation of accessing memory, I/O, or acknowledge an external request. This cycle may consist of three to six T-states.

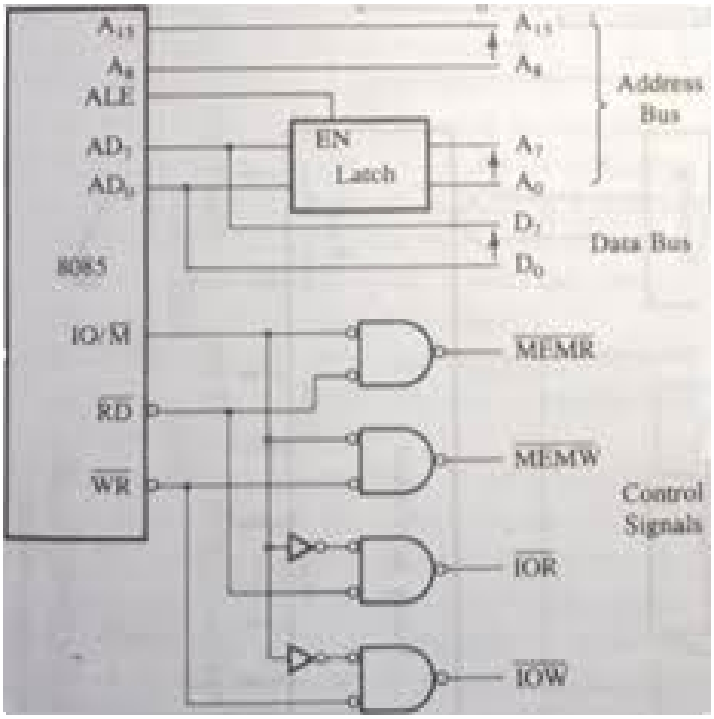
T-state

It is defined as one subdivision of the operation performed in one clock period. These subdivisions are internal states synchronized with the system clock, and each T-state is equal to one clock period.

Generating control signals

- RD is used as a control signal. Because this signal is used both for reading memory and for reading an input device.
- It is necessary to generate two different read signals: one for memory and another for input.
- Four control signals are generated by combining the signals RD, WR, and IO/M. the signal IO/M goes low for the memory operation.
- This signal is ANDed with RD and WR signals by using the 74LS32 quadruple two-input OR gates.
- The OR gates are functionally connected as negative NAND gates.
- When both input signals go low, the outputs of the gates go low and generate MEMR (memory read) and MEMW (memory write) control signals.
- When the IO/M signal goes high, it indicates the peripheral I/O operation.
- The signal is complemented using the hex inverter 74LS04 and ANDed with the RD and WR signals to generate IOR (I/O read) and IOW (I/O write) control signals.
- The MPU can be interfaced with any memory or I/O.



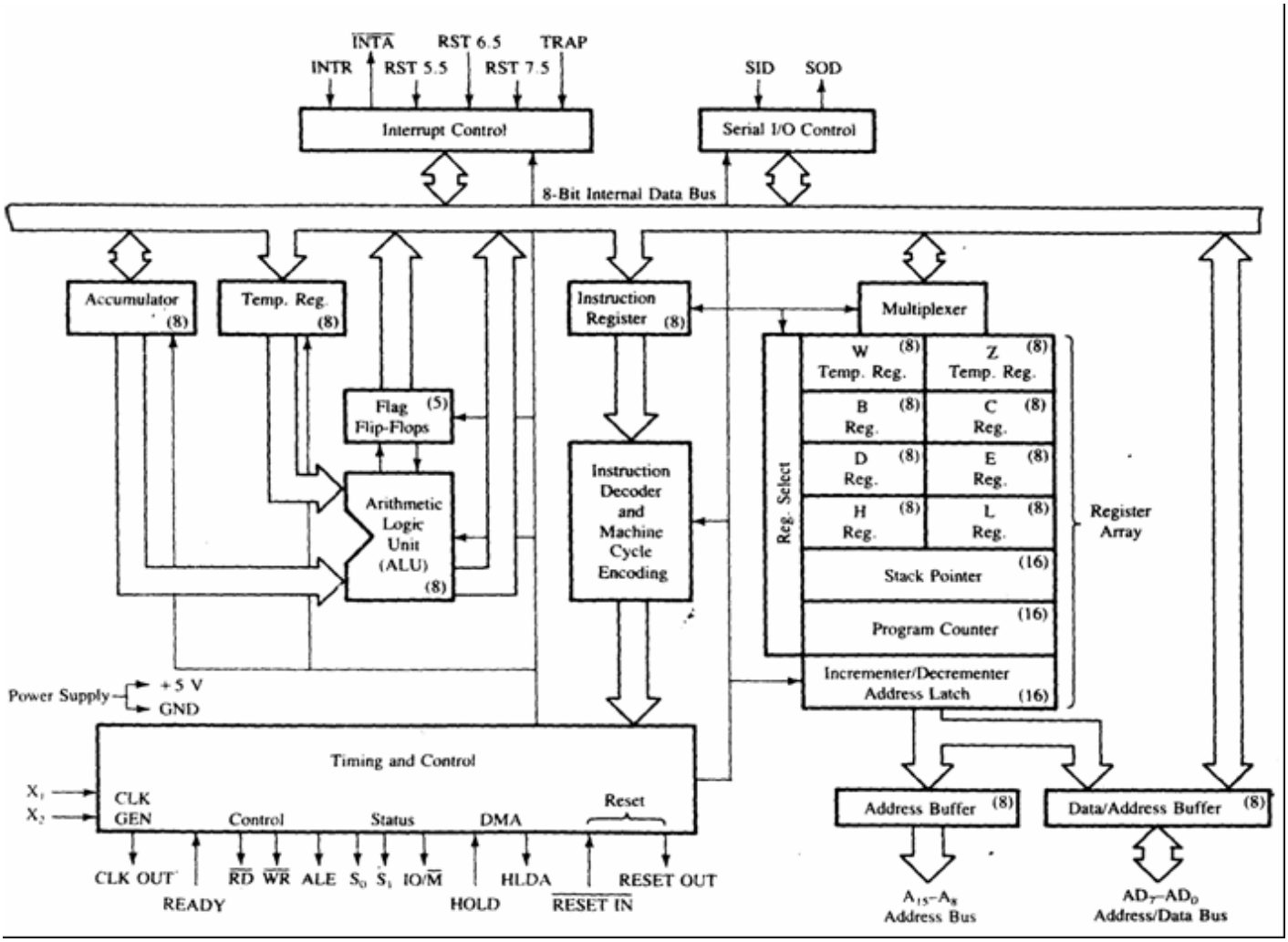


Block diagram of 8085 microprocessor

- The internal architecture of the 8085 beyond the programmable registers it includes the
  - ✓ ALU (arithmetic/ logic unit)
  - ✓ Timing and control unit

- ✓ Instruction register and decoder
- ✓ Register array
- ✓ Interrupt control

Serial I/O control



ALU (Arithmetic Logic Unit)

- ❖ The arithmetic/logic unit performs the computing functions.
- ❖ It includes the accumulator, the temporary register, the arithmetic and logic circuits and five flags.
- ❖ The temporary register is used to hold data during an arithmetic/logic operation.
- ❖ The result is stored in the accumulator and the flags are set or reset according to the result of the operation.
- ❖ The flags generally reflect data conditions in the accumulator-with some expections. For the description of the flag [Refer page no:].
- ❖

Timing and control unit

- The unit synchronizes all the microprocessor operations with the clock and generates the control signals for necessary communication between the microprocessor and peripherals. The RD and WR signals are syncpulses indicating the availability of data on data bus.

Instruction register and decoder

- The instruction register and the decoder are part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register.
- The decoder decodes the instruction and establishes the sequence of events to follow.
- The instruction register is not programmed and cannot be accessed through any instruction.

Register array

- In addition to the 8085 programmable registers, two additional registers called temporary registers W and Z are included in the register array.
- These registers are used to hold 8-bit data during the execution of some instruction.

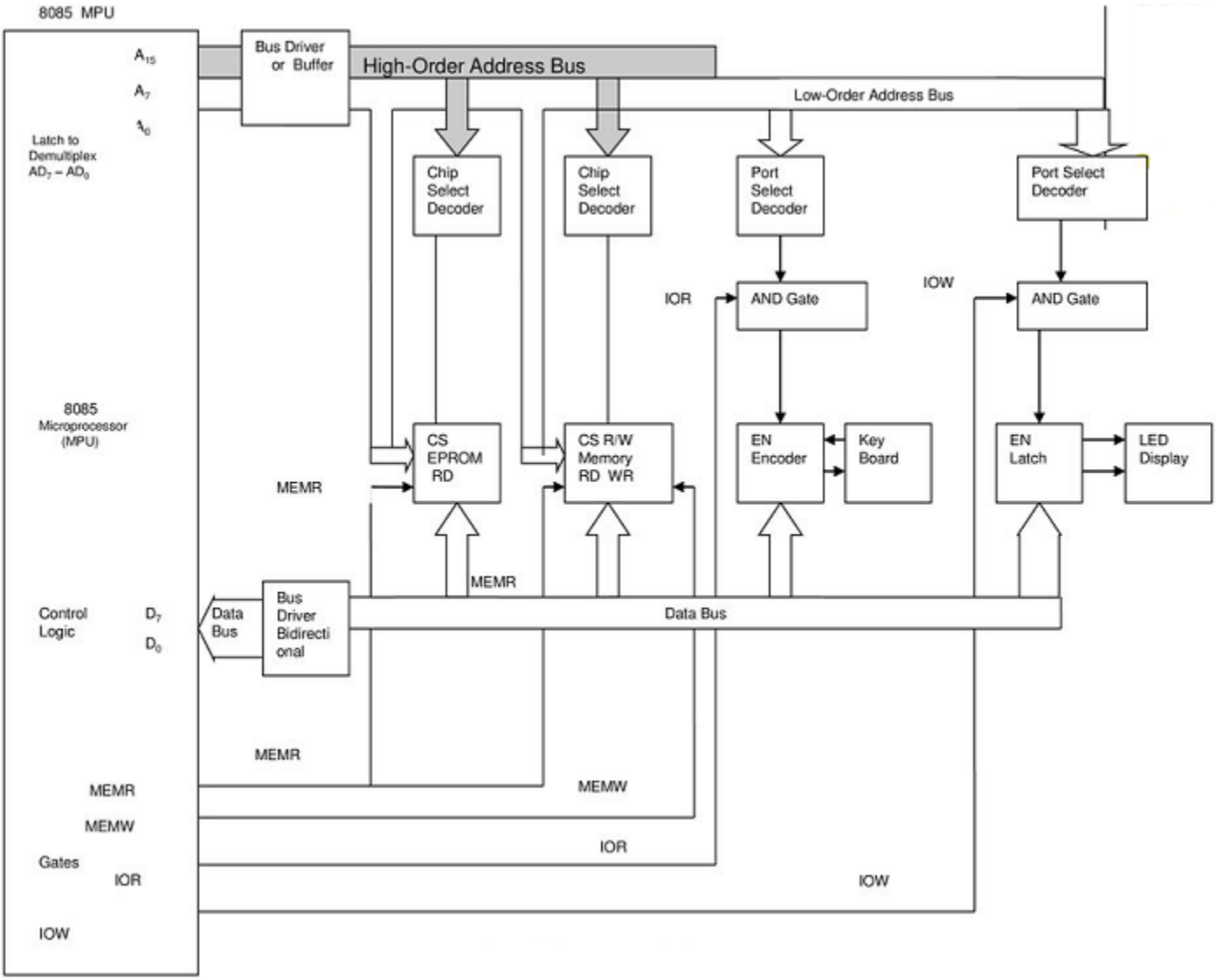
Decoding and executing an instruction

- Assume that the accumulator contains data byte 82H, and the instruction MOV C, A(4FH) is fetched.
- To decode and execute the instruction, the following steps are performed.

- The microprocessor:
    1. Places the contents of the data bus(4FH) in the instruction register and decodes the instruction.
    2. Transfers the contents of the accumulator (82H) to the temporary register in the ALU.
    3. Transfers the contents of the temporary register-to-register C.

Example of an 8085-based microcomputer(8085 single-board microcomputer system)

- The 8085 MPU module includes devices such as the 8085 microprocessor, an octal latch and logic gates.
- The octal latch demultiplexes the bus AD7-AD0 using the signal ALE, and the logic gates generate the necessary control signals.
- The system includes interfacing devices such as buffers, decoders and latches.
- It has a demultiplexed address bus, the data bus and the four active control signals: MEMR, MEMW, IOR, IOW. It uses a unidirectional bus driver for the address bus and bi-directional bus driver for the data bus.



The 8085 machine cycles and bus timings

- The 8085 microprocessor is designed to execute 74 different instruction types. Each instruction has two parts.
  - ✓ Operation code
  - ✓ Operand
- The opcode is a command such as ADD.
- The operand is an object to be operated on, such as a byte or the contents of a register.
- All the instructions in a 8085 are divided into a few basic machine cycles are these are further divided into precise system clock periods.
- The external communication functions can be divided into 3 categories.
  - ✓ Memory read and write
  - ✓ I/O read and write
  - ✓ Request acknowledge
- The opcode fetch cycle by the status signals (IO/M = 0, S1 = 1, S0 = 1); the active low IO/M signal

indicates that it is a memory operation, and S1 and S0 being high indicate that it is an opcode fetch cycle.

Opcode fetch machine cycle

The first operation is the opcode fetch.

It needs to get the machine code from the memory register where it is before the microprocessor can begin to execute the instruction.

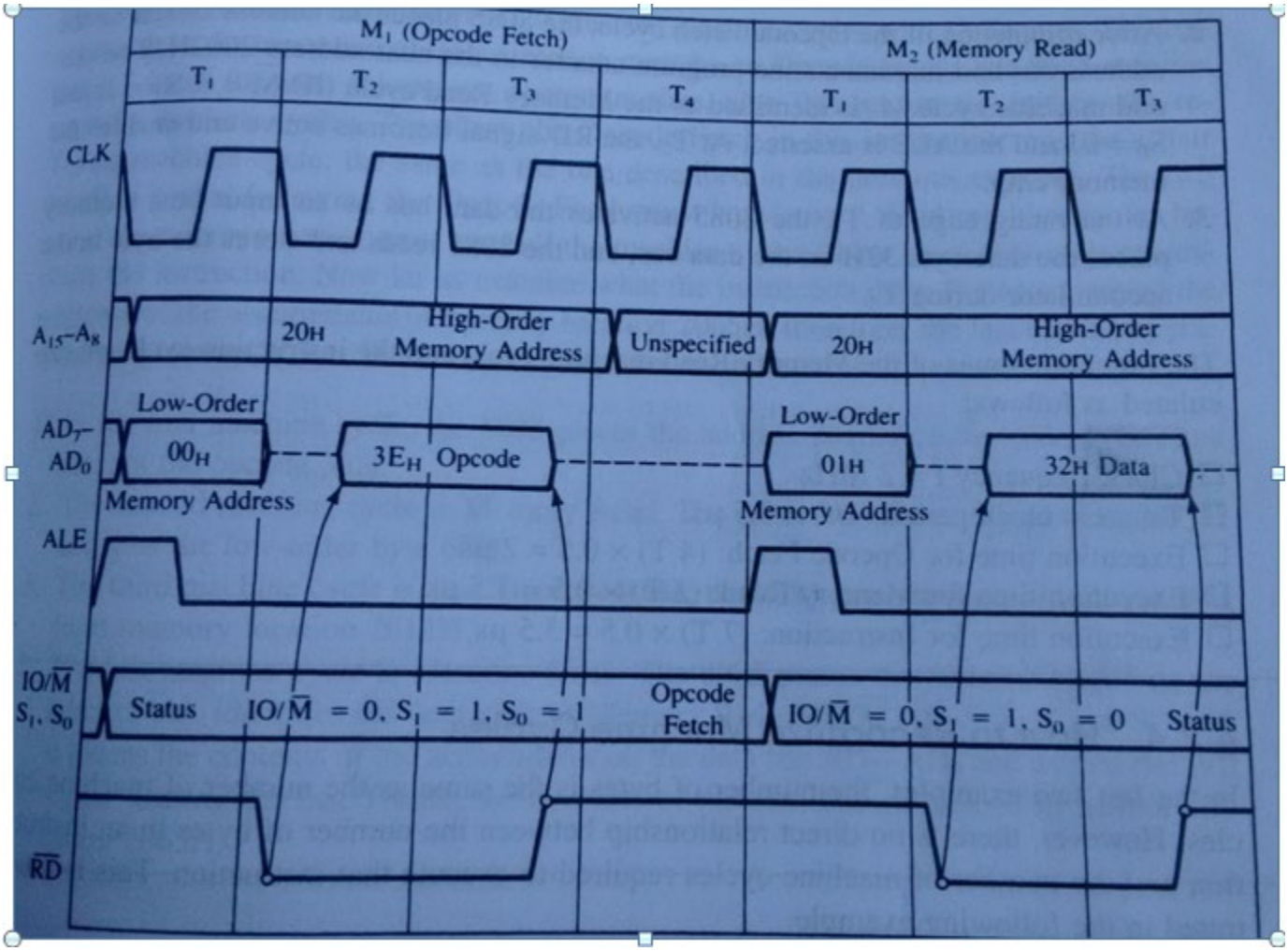
The 8085 fetches the machine code, using the address and the data buses and the control signal.

The opcode fetch cycle is called the M1 cycle and has four T- states.

The 8085 uses the first three states T1- T3 to fetch the code and T4 to decode and execute the code.

Memory read machine cycle

- To illustrate Memory read machine cycle, we need to examine the execution of a 2-byte or 3- byte instruction because in a 1-byte instruction the machine code is an opcode fetching. Therefore the operation is always an opcode fetch.
- Consider two machines codes- 0011 1110(3EH) and 0011 0010(32H) are stored in memory locations 2000H and 2001H.
- the first machine code represents the opcode to load the data byte in the accumulator, and the second code(32H) represents the data byte to be loaded in the accumulator.



Step 1: The first machine cycle M1 (opcode fetch) is identical in bus timings with the machine cycle except for the bus contents.

- At T1, the microprocessor identifies that it is an opcode fetch cycle by placing 011 on the status signals (IO/M = 0, S1 = 1 and S0 = 1).
- It places the memory address (2000H) from the program counter on the address bus, 20H on A15-A8, and 00H on AD7-AD0 and increments the program counter to 2001H to point to the next machine code.

Step 2: After completion of the opcode fetch cycle, the 8085 places the address 2001H on the address bus and increments the program counter to the next address 2002H.

- The second machine cycle M2 is identified as the memory read cycle (IO/M = 0, S1 = 1 and S0 = 0) and the ALE is asserted. At T2 the RD signal becomes active and enables the memory chip.

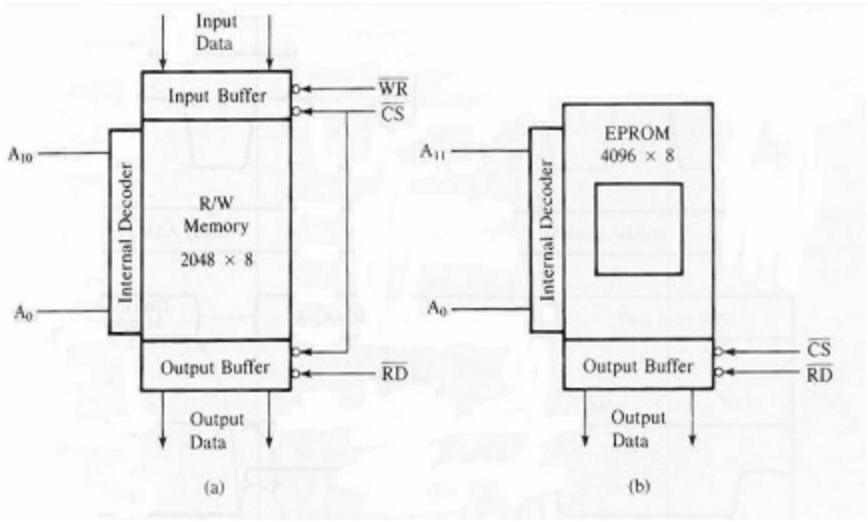
Step 3: At the rising edge of T2, the 8085 activates the data bus as an input bus and memory places the data byte 32H on the data bus, and the 8085 reads and stores the byte in the accumulator during T3.

Memory interfacing

- Memory is an integral part of a microcomputer system.
- While executing a program, the microprocessor needs to access memory quite frequently to read instruction codes and data stored in memory.
- The interfacing unit enables this access.
- Memory has certain signals requirements to write into and read from and write into memory.
- The interfacing process involves designing a circuit that will match the memory requirements with the microprocessor signals.

Memory structure and its requirements

- A typical R/W memory chip has 2048 registers and each register can store eight bits indicated by eight input and eight output data lines.
- The chip has 11 address lines A10 – A0, one chip select (CS) and two control lines read (RD) to enable the output buffer and write (WR) to enable the input buffer.
- The internal decoder is also used to decode the address lines.
- A typical EPROM (erasable programmable read-only memory) has 4096 (4K) registers. It has 12 address lines A11 - A0, one chip select (CS), and one read control signal.

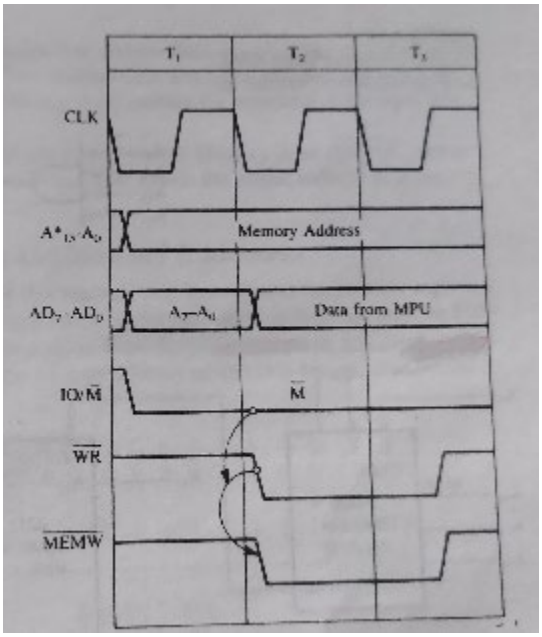


Basic concepts in memory interfacing

The primary function of memory interfacing is that the microprocessor should be able to read from and write into a given register of a memory chip. To perform these operations, the microprocessor should

- ✓ Be able to select the chip
- ✓ Identify the register
- ✓ Enable the appropriate buffer

- ❖ The 8085 place a 16-bit address on the address bus, and with this address only one register should be selected. For the memory chip, only 11 address lines are required to identify 2048 registers.
- ❖ We connect the low order address lines A10-A0 of the 8085-address bus to the memory chip.
- ❖ The remaining address lines (A15 – A11) should be decoded to generate a chip select (CS) signal unique to that combination of address logic.
- ❖ The 8085 provides two signals-IO/M and RD can be combined to generate MEMR control signal that can be used to enable the output buffer by connecting to the memory signal RD.
- ❖ To write into a register, the microprocessor performs similar steps as it reads from a register.
- ❖ The 8085 places the address and data and asserts the IO/M signal.
- ❖ The IO/M and WR signals can be combined to generate the MEMW control signals that enables the input buffer of the memory chip and stores the byte in the selected memory register.

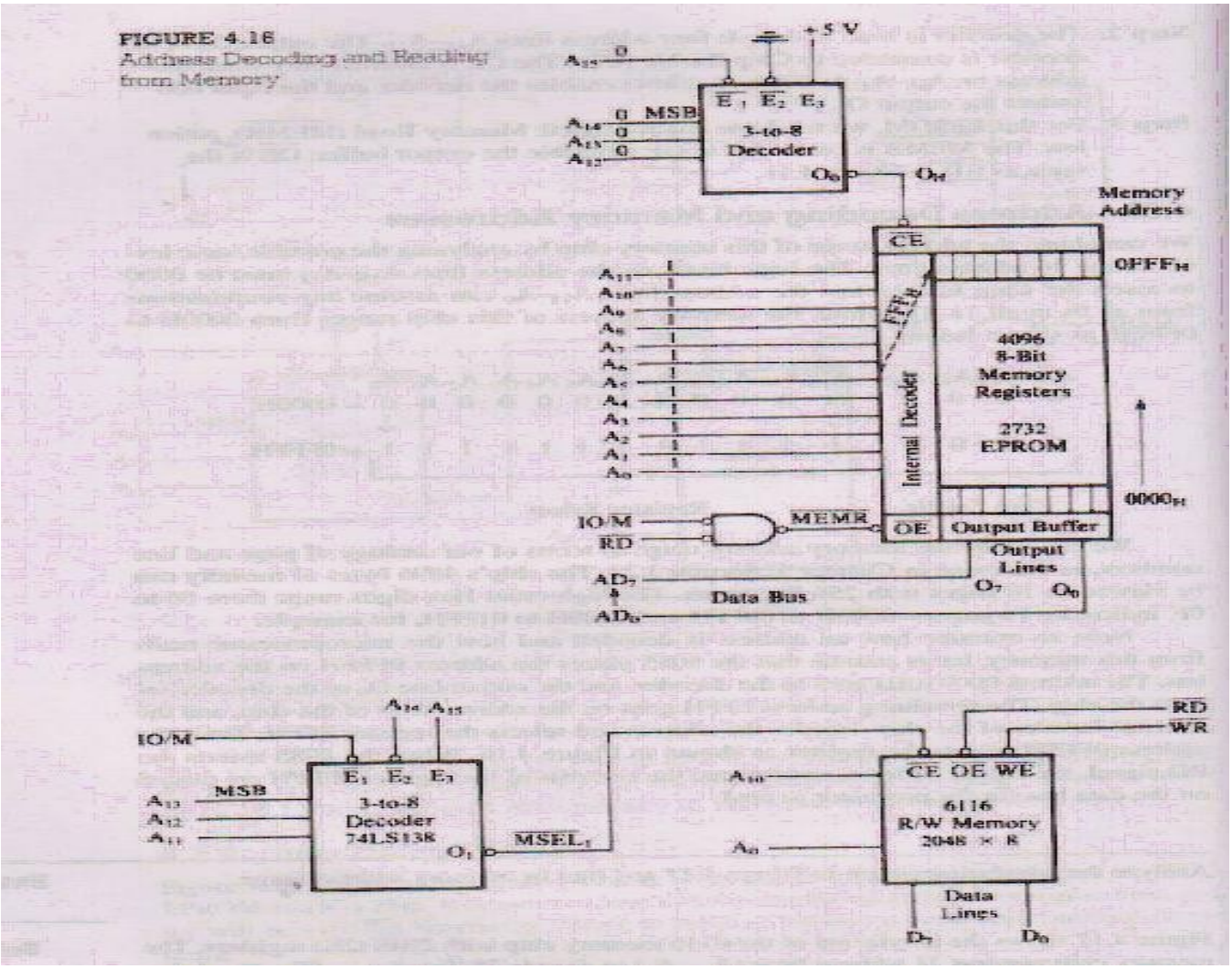


Demultiplexed Address Bus

Address decoding

- This process is to identify a register for a given address.
- The address lines (A11-A0) are connected to the memory chip and the remaining four address lines (A15-A12) are decoded.
- Two methods of decoding these lines are
  - Using a NAND gate
  - 3 to 8 decoder
- The output of the NAND gate goes active and selects the chip only when all the address lines are at logic 1(A15-A12).

- We can obtain the same result by using O7 of the 3 to 8 decoder, capable of decoding eight different input addresses.
- Three lines can have eight different logic combinations from 000 to 111. each input combination is identified by corresponding output line.
- If enable lines are active, the lines E1 and E2 are enabled by grounding and A15 must be at logic 1 to enable E3.



Interfacing circuit

- The above figure shows an interfacing circuit using a 3 to 8 decoder to interface the 2732 EPROM chip.

Step 1: The address lines A11-A0 are connected to pins A11-A0 of the memory chip to address4096 registers.

Step 2: The decoder is used to decode four address lines A15-A12. the output O0 of the decoder is connected to chip enable (CE). The CE is asserted only when the address on A15-A12 is 0000. A15 enables the decoder and the output asserts the output O0.

Step3: we need one control signal: memory read(MEMR), active low. The MEMR is connected to OE to enable output buffer. OE is same as RD.

Address decoding and memory addresses

- The logic levels on the address lines A15-A12 must be 0000 to assert the chip enable, and the address lines A11-A0 can assume any combinations from all 0s and 1s. the memory address of the chip ranges from 0000H to 0FFFH will be as follows.

- The chip's 4096 bytes of memory can be viewed as 16 pages with 256 lines each. The high- order hex digits range from 00 to0F, indicating 16 pages-0000H to 00FFH and 0100H to 01FFH.

UNIT V

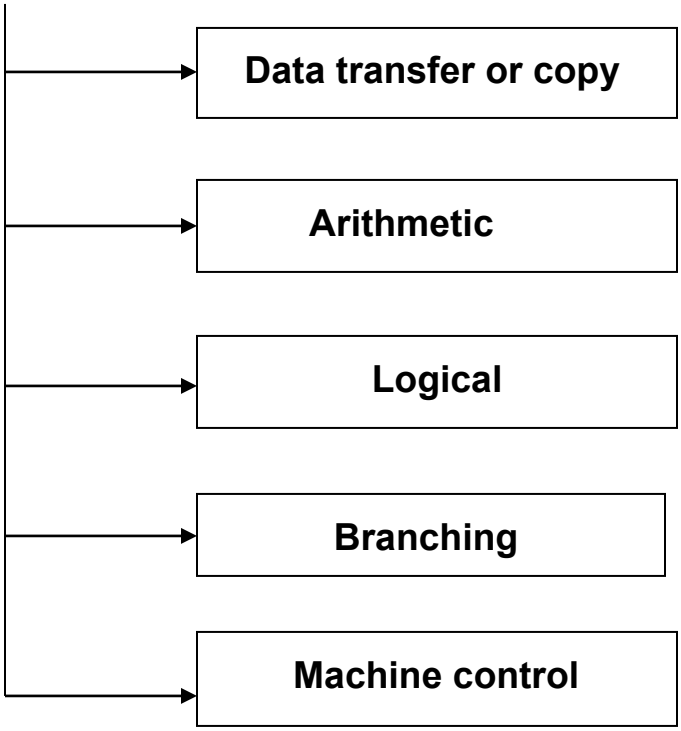
Programming the 8085: Introduction to 8085 Instructions ; Code conversion: BCD to Binary conversion – Binary to BCD conversion – BCD to seven segment LED code conversion – Binary to ASCII and ASCII to binary code conversion – BCD addition – BCD subtraction.

PROGRAMMING TO 8085

INTRODUCTION THE 8085 INSTRUCTIONS

- Each instruction in the program is a command, in binary, to the microprocessor to perform an operation. The entire group of instruction called the instruction set. The instruction can be classified into five different categories.

Instruction



Data transfer or copy instruction

- The primary function of the microprocessor is copying data, from a register called the source to another register called the destination. This copying function is called as the data transfer function.
- The data transfer instructions copy data from a source into a destination without modifying the contents of the source.
- The data transfer instructions do not affect the flags

Opcode	Operand	**Description**
MOV	Rd, Rs	**Move** ✓ **This is a 1 byte instruction** ✓ **Copies data from source register Rs to destination register Rd.**
MVI	R, 8-bit	**Move Immediate** ✓ This is a 2 byte instruction ✓ Loads the 8 bits of the second byte into the register specified.
OUT	8-bitportaddress	**Output to Port** ✓ This is a 2-byte instruction ✓ Sends the contents of the accumulator to the output port specified in the second byte.
IN	8-bitportaddress	**Input from Port** instruction ✓ This is a 2-byte ✓ Accepts data from the input port specified in the second byte, and loads into the accumulator.
LXI	Rp, 16-bit	✓ **Load register pair** ✓ Load 16-bit data in a register pair. ✓ The second byte is loaded in the low-order register of the register pair. ✓ The third byte is loaded in the high-order register pair. ✓ There are four such instructions in the set. The operands B, D, and H represent BC, DE and HL register pairs.

LDA	16-bit port address	**Load Accumulator Direct** ✓ This is a 3-byte instruction. ✓ It copies the data byte from the memory location specified by the 16-bit address in the second and third byte. ✓ The second byte is the low order memory address. ✓ The third byte is the high-order memory address. ✓ The addressing mode is direct.
LDAX	Rp	**Load Accumulator Indirect** ✓ This is a 3-byte instruction. ✓ It copies the data byte from the memory location specified by the 16-bit address in the second and third byte.
		✓ The second byte is a low-order memory address. ✓ The third byte is a high-order memory address. ✓ The addressing mode is direct.
STA	16-bit port address	**Store Accumulator Direct** ☐ This is a 3-byte instruction. ☐ It copies data from the accumulator into the memory location specified by the 16-bit operand.
STAX	Rp	**Store Accumulator Indirect**. ✓ This is a 1-byte instruction. ✓ It copies data from the accumulator to the memory specified by the content of either BC or DE registers.

ARITHMETIC OPERATIONS

- The arithmetic operations are add, subtract, increment and decrement. The add and subtract operations are performed in relation to the content of the accumulator.
- The increment or the decrement operations can be performed in any register.

Opcode	Operand	Description
ADD	R	**Add** ✓ This is a 1-byte instruction. ✓ Add the contents of register R to the contents of th accumulator.
ADI	8-bit	**Add Immediate** ✓ This is a 2-byte instruction. ✓ Adds the second byte to the contents of the accumulator.
SUB	R	**Subtract** ✓ This is a 1-byte instruction ✓ Adds the second byte to the contents of the accumulator.
SUI	8-bit	**Subtract Immediate** ✓ This is a 2-byte instruction ✓ Subtracts the second byte from the contents of the accumulato
INR	R	**Increment** ✓ Increases the contents of register R by 1.
DCR	R	**Decrement** ✓ Decreases the contents of register R by 1.
INX	Rp	**Increment Register Pair.** ✓ This is a 1-byte instruction. ✓ It treats the contents of two registers as one 16-bit number an increases the contents by 1. ✓ The instruction set includes four instructions.
DCX	Rp	**Decrement Register Pair** ✓ This is a 1-byte instruction. ✓ It decreases the 16-bit contents of a register pair by 1. ✓ The instruction set includes four instructions.

LOGIC OPERATIONS

- The logic operations are performed in relation to the contents of the accumulator.
- The logical instructions are AND, OR, EX OR and NOT.

Opcode	Operand	Description
ANA	R	**Logical AND with Accumulator** ✓ This is a 1-byte instruction ✓ Logically ANDs the contents of the register of the register R with the contents of the accumulator.
ANI	8-bit	**AND Immediate with Accumulator** ✓ This is a 2-byte instruction. ✓ Logically ANDs the second byte with the contents of the accumulator.
ORA	R	**Logically OR with Accumulator** ✓ This is a 1-byte instruction. ✓ Logically ORs the contents of the register R with the contents of the accumulator.
ORI	8-bit	**OR Immediate with Accumulator** ✓ This is a 2-byte instruction ✓ Logically ORs the second byte with the contents of the accumulator.
XRA	R	**Logically Exclusive-OR with Accumulator** ✓ This is a 1-byte instruction. ✓ Exclusive-ORs the contents of register R with the contents of the accumulator.
XRI	8-bit	**Exclusive-OR Immediate with Accumulator** ✓ This is a 2-byte instruction. ✓ Exclusive ORs the second byte with the contents of the Accumulator.
CMA		**Complement Accumulator** ✓ This is a 1-byte instruction that complements the contents of the accumulator. ✓ No flags are affected.
RLC		**Rotate Accumulator left** ✓ Each bit is shifted to the adjacent left position. Bit D7 becomes D0. ✓ CY flag is modified according to bit D7.

RAL		**Rotate Accumulator left through carry** ✓ Each bit is shifted to the adjacent left position. ✓ Bit D7 becomes the carry bit and the carry bit is shifted into D0. ✓ The carry flag is modified according to bit D7.
RRC		**Rotate Accumulator right** ✓ Each bit is shifted right to the adjacent position. Bit D0 becomes D7. ✓ The carry flag is modified according to bit D0.
RAR		**Rotate Accumulator Right through Carry** ✓ Each bit is shifted right to the adjacent position. Bit D0 becomes the carry bit, and the carry bit is shifted into D7.
CMP		**Compare with Accumulator** ✓ This is a 1-byte instruction. ✓ It compares the data byte in register or memory with the contents of the accumulator. ✓ If (A) < (R/M), the CY flag is set and the Zero flag is reset. ✓ If (A) = (R/M), the Zero flag is set and the CY flag is reset. ✓ If (A) > (R/M), the CY and zero flags are reset.
CMI		**Compare Immediate with accumulator** ✓ This is a 2-byte instruction, the second byte being 8-bit data. ✓ It compares the second byte with (A). ✓ If (A) < 8-bit data, the CY flag is set and the Zero flag is reset. ✓ If (A) = 8-bit data, the Zero flag is set, and the CY flag is reset. ✓ If (A) > 8-bit data, the CY and Zero flags are reset.

BRANCH OPERATIONS

- The branch instructions allow the microprocessor to change the sequence of a program either conditionally or unconditionally.
- All jump instructions in the 8085 are 3-byte instructions. The second byte specifies the low-order memory address and the third byte specifies the high-order memory address.

CONDITIONAL JUMP

Opcode	Operand	Description
JC	16-bit	**Jump On Carry** (if result generates carry and CY = 1)
JNC	16-bit	**Jump On No Carry** (CY = 0)
JZ	16-bit	**Jump On Zero** (if result is zero and Z = 1)
JNZ	16-bit	**Jump On No Zero** (Z = 0)
JP	16-bit	**Jump On Plus** (if D7 = 0, S = 0)
JM	16-bit	**Jump On Minus** (ifD7=1,S=1)
JPE	16-bit	**Jump On Even Parity** (P = 1)
JPO	16-bit	**Jump On Odd Parity** (P = 0)

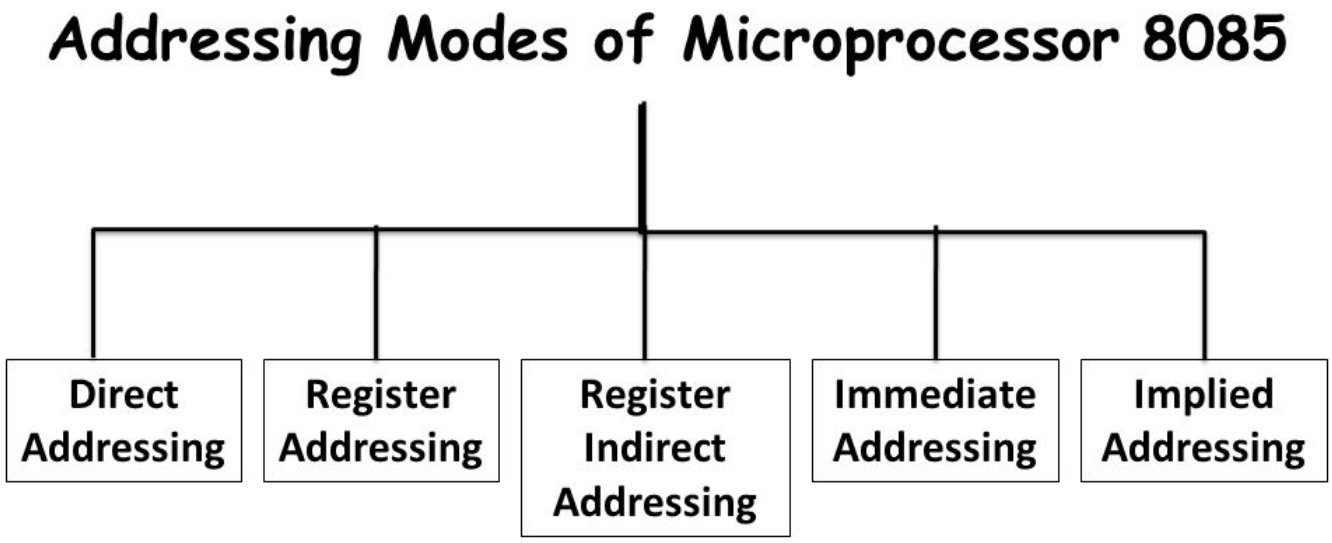
Unconditional jump

The 8085 instruction set includes one unconditional jump instructions.

Opcode	Operand	Description
JMP	16-bit	Jump Enablestheprogrammertosetupcontinuousloops

ADDRESSING MODES

- The various format of specifying the operands are called addressing modes. The 8085 instruction set has the following modes.



Direct addressing

- Simply giving the complete binary address in memory is the most direct way to locate an operand or to give an address to jump.

Example: **LDA 3A**

The 8-bit address in the memory to be loaded into the accumulator

Register addressing

- The instruction specifies a register pair that contains the memory address where the data are located.

Example: **MOV Rd, Rs**

- It moves the content of source to the destination register. The operand is stored in one of the CPU register.

Indirect addressing

- The instruction indicates a register pair that contains the address of the next instruction to be executed.

Example: **MOV M, C**

- The above instruction moves the contents of the C register into the memory address stored in register pair.

Immediate addressing

- Immediate addressing refers to move the immediate data to any of the registers, accumulators or memory.

Example: **MVI R, data**

- The above example moves the value of R (immediate value of R) to the data.
- The immediate addressing will have the data as a part of the instruction.

- The move instructions will have immediate instructions MVI, similarly ADI, SUI, ANI etc.,

Implied/ implicit addressing

- In implied/implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself

Examples:

- CMA (finds and stores the 1's complement of the contains of accumulator A in A)
- RRC (rotate accumulator A right by one bit)
- RLC (rotate accumulator A left by one bit)

CODE CONVERSION

BCD to BINARY Conversion

Describe the procedure for BCD to binary conversion and write the program for the same.

- The conversion of a BCD number into its binary equivalent employs the principle of positional weighting in a given number.

For example $72_{10} = 7 \text{ X } 10 + 2$.

- The digit 7 represents 70, based on its second position from the right, so its binary equivalent requires multiplying the second digit by 10 and adding the first digit.

$72_{10} = 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0$

$BCD_1 = 00000111$

$BCD_2 = 00000111$

- Multiply BCD_2 by 10, add the answer to the BCD_1.
- The multiplication of BCD2 by 10 can be performed by various methods.

Location	Program	Explanations
START	**LXI SP, STACK**	Initialize the stack pointer
	LXI H, 9000	Input buffer location
	LXI B, 9002	Output buffer location
	MOV A, M	Move the content of the input to accumulator
	CALL BCDBIN	Call the subroutine BCDBIN
	STAX B	Store the accumulator content to B register
	HLT	Terminate the program
BCDBIN	**PUSH B**	Save BC register
	PUSH D	Save DE register
	MOV B, A	Move the accumulator content to B register
	ANI 0FH	Mask most significant four bits

	MOV C, A	Move the accumulator content to C register.
	MOV A, B	Move the B register to accumulator
	ANI F0H	Mask the least significant four bits
	RRC	Convert most significant four bits into unpacked BCD_2
	MOV D, A	Save BCD_2 in D register
	XRA A	A → 00
	MVI E, OAH	Set E as multiplier of 10
SUM	**ADD E**	Add 10 until D = 0
	DCR D	Decrement the D register by 1
	JNZ SUM	
	ADD C	Add BCD_1
	POP D	Retrieve previous contents
	POP E	
	RET	Return to calling subroutine

BCD to BINARY conversation

- The conversion of binary to BCD is performed by dividing the number by the powers by dividing the number by the power of ten; the division is performed by the subtraction method.
- For example, assume the binary number is 1 1 1 1 1 1 1 1 FFH = 255
- To represent this number in BCD requires 12 bits or three BCD digits labelled as BCD_3 (MSB), BCD_2 and BCD_3(LSB).

0 0 1 0 0 1 0 1 0 1 0 1

BCD_3 BCD_2 BCD1

- The conversion can be performed as follows

Step 1:

If the number is less than 100, go to step 2; otherwise, divide by 100 or subtract 100 repeatedly until the remainder is less than 100. the quotient is the most significant BCD digit BCD_3.

Step 2: If the number is less than 10, go to step 3, otherwise divide by 10 repeatedly until the remainder is less than 10. the quotient is BCD_2.

Step 3: The remainder from step2 is BCD_1.

Location	Program	Explanations
START	**LXI H, 8050**	Point HL index where binary number is stored
	MOV A, M	Move the content of memory to accumulator
	CALL BCD_1	Call the subroutine BCD_1
BCD_1	**LXI H, 8060**	Point HL index where the BCD number is stored
	MVI B, 64H	Load 100 in register B
	CALL BCD_2	Call conversion
	MVI B, 0A	Load 10 in register B
	CALL BCD_2	Call the BCD_2 subroutine
	MOV M, A	Move the accumulator content to memory location.
	RET	Return
BCD_2	**MVI M, FFH**	Load 255 to the memory location
XX	**INR M**	Increment the memory location by one.
	SUB B	Subtract the content of B register to accumulator
	JNC XX	Subtract until less than power of 10.
	ADD B	Add the content of B register pair
	INX H	Increment the HL register pair
	RET	Return

BCD to seven segment LED code conversion

- A set of three packed BCD numbers representing time and temperature are stored in memory locations starting at XX50 H.
- The seven segment codes of the digits 0 to 9 for a common cathode LED are stored in memory locations at XX70H and the output buffer memory is reserved at XX90H.

Location	Program	Explanation
START	**LXI SP,STACK**	Initialize stack pointer
	LXI H, XX50 H	Point HL where BCD digits are stored
	MVI D, 03H	Number of digits to be converted is placed in D.
	CALL UNPAK	Call subroutine to unpack BCD numbers
	HLT	End of conversion
UNPAK	**LXI B,BUFFER**	Point BC index to the buffer memory
NXTBCD	**MOV A, M**	Get packed BCD number
	ANI FOH	Mask BCD_1
	RRC	Rotate four times to place BCD_2
	RRC	
	RRC	
	RRC	
	CALL LEDCOD	Find seven segment code
	INX B	Point to next buffer location

	MOV A, M	Get BCD number again
	ANI OFH	Separate BCD_1
	CALL LEDCOD	
	INX B	
	INX H	Point to next BCD
	DCR D	One conversion complete reduce BCD count
	JNZ NXTBCD	If all BCDs are not yet converted, go back to convert next BCD
LEDCOD	**PUSH H**	Save HL contents of the caller
	LXI H, CODE	Point index to beginning of seven-segment code
	ADD L	Add BCD digit to starting address of the code
	MOV L, A	Point HL to appropriate code
	MOV A, M	Get seven-segment code
	STAX B	Store code in buffer
	POP H	
	RET	

Binary to ASCII and ASCII to binary code conversion

Binary to ASCII

Location	Program	Explanation
START	**LXI SP, STACK**	Initialize the stack pointer
	LXI H, 9050 H	Point where the value reside
	LXI H, 9060 H	Point where ASCII is stored
	MOV A, M	Move the content of memory location to accumulator
	MOV B, A	Move the accumulator to B register
	RRC	Rotate four times
	RRC	
	RRC	
	RRC	
	CALL ASCII	Call the subroutine
	STAX D	Store the accumulator to DC register pair
	INX D	Increment the D register
	MOV A, B	Move the B register to accumulator
	CALL ASCII	Call the subroutine
	STAX D	Store the accumulator to DC register pair
	HLT	Terminate the program
ASCII	**ANI 0FH**	And immediate to the accumulator
	CPI 0AH	Compare the accumulator with data
	JC CODE	Perform the loop until accumulator is less than the value

	ADI 07H	Add 07 to accumulator
CODE	**ADI 30H**	Add 30H to accumulator
	RET	Return

ASCII to BINARY conversion

Location	Program	Explanation
ASCBIN	**LDA 6100**	Load the content to the accumulator
	CALL SUB	Call the subroutine SUB
	STA 6102	Store the accumulator content to 6102
	HLT	Terminate the program
SUB	**SUI 30H**	Subtract immediately 30H from the accumulator
	CPI 0AH	Check whether number is between 0 and 9
	RC	If yes, return to main program
	SUI 07H	If not, subtract 7 to find number between A and F.
	RET	

BCD addition

Location	Program	Explanation
START	**LXI SP, STACK**	Initialize the stack pointer
	LXI H, 9000H	
	MVI C, COUNT	Load register C with the count of BCD number to be added
	XRA A	Clear the accumulator
	MOV B, A	Move the accumulator to register
NXT	**CALL BCDADD**	Call the subroutine
	INX H	Increment the HL register pair
	DCR C	Decrement the C register
	JNZ NXT	If all numbers are added goto next step otherwise go back
	LXI H, 9063H	Point index used to store the BCD1 first
	CALL UNPACK	Unpack the BCD stored in the accumulator
	MOV A, B	Move the B register to the accumulator
	CALL UNPAK	Call the subroutine
	HLT	Terminate the program
BCDADD	**ADD M**	Add packed BCD byte and adjust it for BCD sum
	DAA	
	RNC	If no carry go back to next BCD

	MOV D, A	If carry is generated save the sum from the accumulator to D
	MOV A, B	Move the B register to accumulator
	ADI 01H	Add 01H
	DAA	Decimal adjust BCD from B
	MOV B,A	Save adjusted BCD in B
	MOV A, D	Place BCD_1 and BCD_2 in accumulator
	RET	Return
UNPAK	**MOV D, A**	Save BCD number
	ANI OFH	Mask high order BCD
	MOV M, A	Store low order BCD
	DCX H	Point to next memory
	MOV A, D	Get BCD again
	ANI F0H	Mask low order BCD
	RRC	Convert the MSB bits to unpacked BCD
	RRC	
	RRC	
	RRC	
	MOV M, A	Move the content of accumulator to memory
	DCX H	Point to next memory location
	RET	Return

BCD subtraction

When subtracting two BCD numbers

Location	Program	Explanation
SUBBCD	**MVI A, 99H**	
	SUB C	Find 99'scomplement
	INR A	Find 100's complement
	ADD B	Addminuendto100'scomplement
	DAA	Adjust for BCD
	RET	Return