BRPT DR. MGR GOVT. ARTS AND SCIENCE COLLEGE - PALACODE

DEPARTMENT OF COMPUTER SCIENCE

I B.SC (CS)

DIGITAL COMPUTER FUNDAMENTALS AND MICROPROCESSOR

19UCS01

DIGITAL COMPUTER FUNDAMENTALS AND MICROPROCESSOR – 19UCS01

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. 14

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 Book:

1. —Understanding Computers- Today and Tomorrow‖, Deborah Morley, Charles S. Parker, 1ST Edition, Thomson Course Technology, 2007
2. —8085 Microprocessor Programming and Interfacing‖, N.K.Srinath, PHI Publishing , 2005

UNIT I

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

Introduction:-

- In 19th Century the Charles Babbage attempted to mechanize sequence of calculations, eliminating the operator and designing a machine so that it would perform all necessary operations in a predetermined sequence.
- The machine was to perform the instructions dictated by the card automatically, not stopping until an entire sequence of instructions had been completed.
- In 1937 Howard Aiken, at Harvard, proposed to IBM that a machine could construct which would automatically sequence the operations and calculations performed.
- The machines used a combination of electromechanical devices, including many relays.

GENERATIONS OF MODERN COMPUTERS

- Computers are truly amazing machines.
- Based on the period of development and the features incorporated , they are classified into different generations.

The classification and the time period are:

1. First generation (1945-1956)
2. Second generation (1956-1963)
3. Third generation (1964-1971)
4. Fourth generation (1971-present)
5. Fifth generation (present and beyond)

First generation (1945-1956)

- Two engineers built the first computer using parts called vacuum tubes and named as ENIAC.
- Consists of 18,000 vacuum tubes, 70,000 resistors and 5 million soldered joints.
- Each computer had a binary-coded program called machine language.
- Difficult to program and limited its versatility and speed
- Magnetic drums are used for data storage.

Second generation (1956-1963)

- Transistors replaced the vacuum tubes
- Machine language were replaced by assembly language
- They have printers, tape storage memory and operating systems
- Stored program and programming language gave flexibility
- Languages like COBOL and FORTRAN came into common use

Third generation computers (1964-1971)

- Integrated circuits were developed by Jack Kilby
- Scientists fit more components on a single chip called semiconductor
- IBM announced a hardware-system/360 family of mainframe computers
- Operating systems allowed machines to run many different programs.

Fourth generation computers (1971-present)

- Large scale integration (LSI) fit many components onto one chip
- Ability to fit so much, increased their power, efficiency and reliability

- Diminished the size and price of computers
- Number of computers in use jumped from 2 million to 5.5 million
- A global web of computer circuitary links into a single network of information

Fifth generation (present and beyond)

- Aim to be able to solve highly complex problems require reasoning; intelligence and expertise
- Designed to people who are not computer experts
- They will not have a single processor, or a small number of tightly coupled processor
- Processing power is expressed in logical influences per second (lips)
- Performance are achieved through highly parallel architectures
- It will be a dominant force in IT (Information Technology)

APPLICATION OF COMPUTER

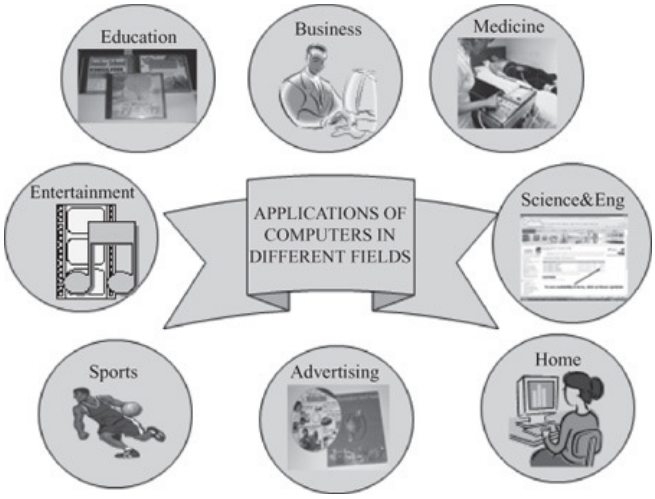
Computer is now playing a vital role in the lives of people today. It has revolutionized, the way one used to live. It gave a virtual world itself, where there are no barriers for communication, information sharing, idea sharing etc.

Computer In Business

The business process is under the IT revolution, which is transforming the way we do the business. The way our basic business operations like decision making, customer services, operations, marketing strategies, financial management, Human Resources management, etc. are done are being reformed with the use of computers.

The basic reasons for the use of IT in business are as follows:

1. Efficient Business Operations
2. Better Managerial Decision Making .
3. Gaining Competitive Advantage.
4. Business Process Re-Engineering
5. Solving Business Problems
6. Globalization of Business
7. Spontaneous Activity
8. Office Automation
9. Communication & Collaboration
10. Electronic Commerce
11. Interactive Marketing Etc.



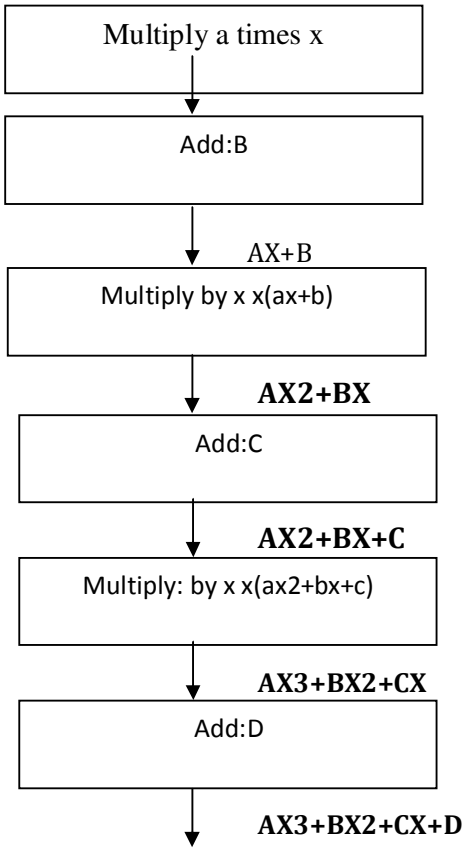
Scientific Applications

- The electronic digital computer is a valuable tool for studying the consequences of these laws.
- The exact procedure for solving a problem has been found, but the time required to perform the necessary calculations manually is prohibitive.
- Sometimes it is necessary to solve the same problem many times with different sets of parameters, and the computer is especially useful for solving the problems.

An algebraic formula is an expression of a mathematical relationship. Many of these laws of physics, electronics, chemistry etc., are expressed in this form.

To evaluate the expressions ax3+bx2+cx+d. given numerical values of a,b,c,d and x.The required steps are as follows.

1.Multiply a times x, yielding ax.
2. add b,yielding ax+b.
3.Multiply this by x, forming ax2+bx.
4.Add c, yielding ax3+bx+c
5. Multiply this by x :x(ax2+bx+c) or ax3+bx2+cx
6.Add d, obtaining ax3+bx2+cx+d
It would take several minutes to perform the calculations necessary to evaluate this algebraic expression for a single set of values by using manually operated calculators, but practically any computer could perform this series of operations several thousand times per second.



DIFFERENT TYPES OF COMPUTER SYSTEMS:

Batch processing is execution of a series of programs ("jobs") on a computer without manual intervention.

Jobs are set up so they can be run to completion without manual intervention. So, all input data are preselected through scripts, command-line parameters, or job control language.

An interactive programs which prompt the user for such input. A program takes a set of data files as input, processes the data, and produces a set of output data files. This operating environment is termed as "batch processing" because the input data are collected into batches of files and are processed in batches by the program

Interactive and Real-time Systems

The computers are using now is an *interactive system*. This means that it does not do anything until the user clicks a button or moves the mouse. Then it reacts to the user and could then possibly instruct the user to do something.

Some interactive systems are capable of *background processing*. This is where the system gets on with a job that does not require user input (e.g. virus scanning, down loading software updates, etc) while it is waiting for user input. Because computer systems are so fast they are able to get on with

things between key presses. The user is unaware that the background is going on as the system still reacts instantly to the user.

Real-time systems do not wait for user input but react to sensors. They are used to monitor machines to ensure that readings like temperatures, radiation levels, air pressure, etc. are correct and to react instantly if changes need to be made.

Another use for real-time systems is for systems used to keep track of ticket sales for concerts, flights, train tickets, etc.

- A Widely used input-output device is the terminal. This is an example of a keyboard which is typewriter like, generating a printed record when used, but also generating electric signals that can be used as computer input.
- Terminals are sometimes used in systems in which the console terminal is some distance from the computer. A special attachment called a modem, which makes it possible to transmit the electric signals generated by the terminal to the computer and receive the computer's response back over telephone lines.
- The user of the terminal simply dials the number at which the computer is located, establishes a connection and the users identify and right to use the computer and then proceeds to use the computer.
- A terminal with a CRT display which is similar to a telephone vision, thus providing a temporary display. These types of output device enable the computer to draw pictures or make graph as well as use printed characters.
- The computer terminal is portable. An attachment on the rear called an acoustic coupler which holds a telephone handset is provided so that when the telephone handset is placed in the attachment, computer can be dialed.
- Once the connection is made, the terminal then generates audio tones into the handset when keys are depressed on the keyboard. The terminal receiver also decodes coded tones representing characters generated by the computer, displaying the information received on the CRT display device.

- An acoustic coupler generates and receives audio signals from the handset while a modem uses electric signals and is connected directly into telephone jack.
- When a number of users share a computer, using the computer, sometimes via telephone lines, at the same time, the computer is able to be timeshared.

Computers can be generally classified by size and power as follows, though there is considerable overlap:

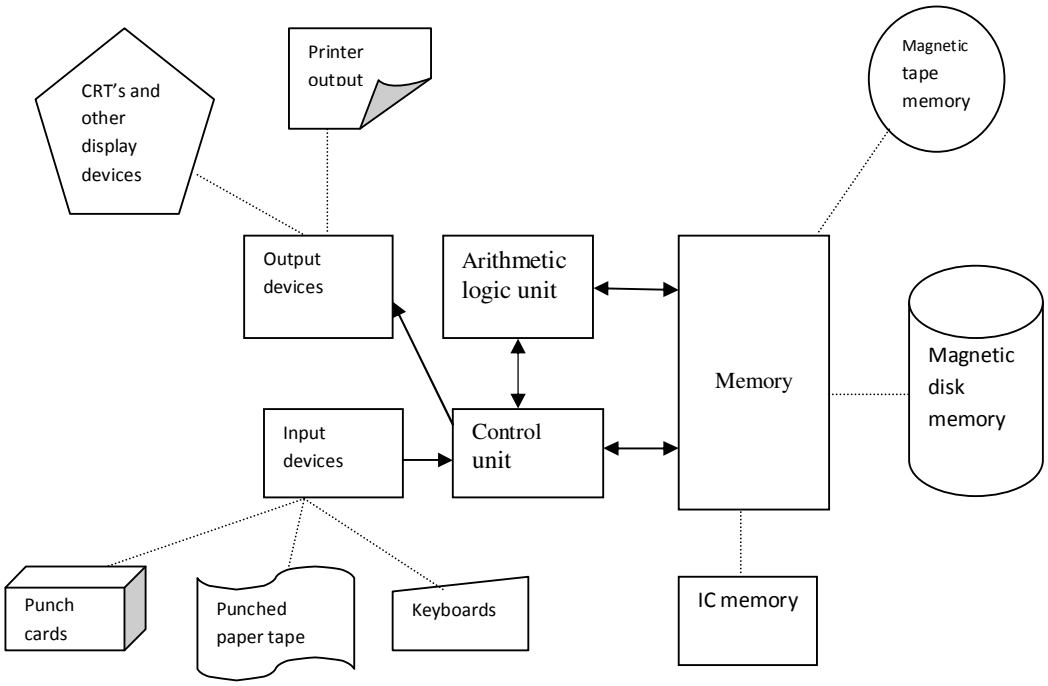
- Personal computer: A small, single-user computer based on a microprocessor.
- Workstation: A powerful, single-user computer. A workstation is like a personal computer, but it has a more powerful microprocessor and, in general, a higher-quality monitor.
- Minicomputer: A multi-user computer capable of supporting up to hundreds of users simultaneously.
- Mainframe: A powerful multi-user computer capable of supporting many hundreds or thousands of users simultaneously.
- Supercomputer: An extremely fast computer that can perform hundreds of millions of instructions per second.

COMPONENTS OF DIGITAL COMPUTER

The five major operational divisions of an electronic digital computer.

A digital computer may be divided into the following fundamental units:

1. Input device
2. Central Processing Unit (CPU), and
3. Memory Unit.
4. Arithmetic logic unit
5. Output device



Input: The input devices read the necessary data into the machine.In most general purpose computers, the instructions that constitute the program must be read into the machine along with all the data to used in the computatuions.

Examples : Keyboard,punch-card,tape readers.

Control: The control section of a computer sequence the operation of the computer, controlling the actions of all other units.The control circuitry interprets the instructions which constitute the program and then directs the rest of the machine in its operation.

Memory : Memory unit stores the data, instructions, intermediate results and output, temporarily, during the processing of data. This memory is also called the main memory or primary memory of the computer.

The input data that is to be processed is brought into the main memory before processing. The instructions required for processing of data and any intermediate results are also stored in the main memory. The output is stored in memory before being transferred to the output device.

Another kind of storage unit is also referred to as the secondary memory of the computer. The data, the programs and the output are stored permanently in the storage unit of the computer. Magnetic disks, optical disks and magnetic tapes are examples of secondary memory.

Arithmetic Logic unit:

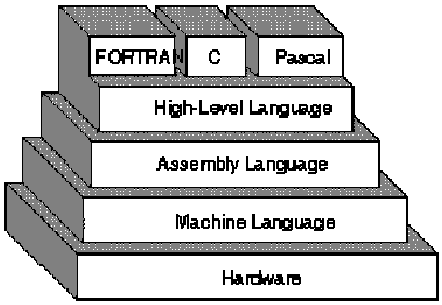
- ALU performs all the arithmetic (addition, subtraction, multiplication and division) and logic operations on the input data.
- The control unit tells the arthimetic logic unit which operation to perform and then sees that the necessary numbers are supplied.
- The arithmetic element can be compared to the calculating machines described previously in that the numbers to be used are inserted, and it is then directed to perform the necessary operations.

Output:

The output devices are used to record the results obtained by the computer and present them to the outside world. Most output devices are directed by the control element, which also causes the necessary information to be supplied to them.

Example: CRT display, printers, card punching machines and magnetic tapes drivers.

PROGRAMMING SYSTEMS:



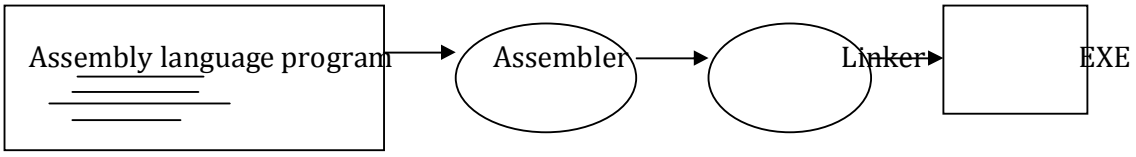
The various types of programming languages which is greatly facilitate the actual writing of programs. One of the first things the programming profession discovered was that the greatest aid to programming was the computer itself.

The program systems are two types:

1.Assemblers 2.Compilers

Assembler

An *assembler* is a program that accepts as input an assembly language program (source) and produces its machine language equivalent (object code) along with the information for the loader.



Executable program generation from an assembly source code

The purpose of this procedure is to enable programmers to write the operations they want the computer to perform in a manner that is simpler than machine language. The language which the programmer writes is called a programming language, and a program written in such a language is called an object program.

Advantages of coding in assembly language are:

- Provides more control over handling particular hardware components
- May generate smaller, more compact executable modules
- Often results in faster execution

Disadvantages:

- Not portable
- More complex
- Requires understanding of hardware details (interfaces)

Assembly Languages:

Each instruction to the computer in a programming language is called a statement. The basic characteristic which most distinguishes an assembly language is that each statement is translated by the assembly program to a single machine instruction word.

Mnemonic operations code:-

The programmer can write instructions to the computer using letters instead of binary numbers, and the letters which designate a given operations are arranged into a mnemonics code that convey the 'sense' of the instruction.

Symbolic referencing of storage addresses.

One of the greatest facilities offered the programmer is the ability to name the different pieces of data used in the program and to have the assembler automatically assign addresses to each name.

Convenient data representation:

This simply means that the programmer can write input data as, for instance, decimal numbers or letters, or perhaps in some form specific to the problem, and that the assembly program will convert the data from this form to the form required for machine computation.

Program listing:

An important feature of most assemblers is their ability to print for the programmer a listing of the source program and the object program, which is in machine language.

Error detection:

An assembler program will notify the programmer if an error has been made in the usage of the assembly language.

Assembly language uses a mnemonic to represent each low-level machine operation or opcode. Some opcodes require one or more operands as part of the instruction, and most assemblers can take labels and symbols as operands to represent addresses and constants, instead of hard coding them into the program.

```
mov ax,1
mov bx,F001
mov cx,2
mov dx,F002
add bx,dx
adc ax,cx
```

High-level programming language

A **high-level programming language** is a programming language with strong abstraction from the details of the computer. In comparison to low-level programming languages, it may use natural language elements, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. memory management), making the process of developing a program simpler and more understandable relative to a lower-level language.

There are three models of execution for modern high-level languages:

Interpreted

Interpreted languages are read and then executed directly, with no compilation stage. A program called an *interpreter* reads each program line following the program flow, converts it to machine code, and executes it; the machine code is then discarded, to be interpreted anew if the line is executed again.

Compiled

Compiled languages are transformed into an executable form before running. There are two types of compilation:

Machine code generation

Some compilers compile source code directly into machine code. This is the original mode of compilation, and languages that are directly and completely transformed to machine-native code in this way may be called "truly compiled" languages.

Intermediate representations

When a language is compiled to an intermediate representation, that representation can be optimized or saved for later execution without the need to re-read the source file.

When the intermediate representation is saved, it is often represented as byte code. The intermediate representation must then be interpreted or further compiled to execute it.

Virtual machines that execute byte code directly or transform it further into machine code have blurred the once clear distinction between intermediate representations and truly compiled languages.

Translated

A language may be translated into a lower-level programming language for which native code compilers are already widely available. The C programming language is a common target for such translators.

NUMBER SYSTEM

We use the decimal numbers or the decimal number system for our day-to-day activities. But computers understand only 0s and 1s – the machine language.

DECIMAL NUMBER SYSTEM

The base or radix of a number system is defined as the number of digits it uses to represent the numbers in the system. Since decimal number system uses 10 digits-0 through 9- its base or radix is 10. The decimal number system is also called base-10 number system.

For example, consider the number 3256
3256 = 3000+200+50+6 (or)
3256 = 3*10^3+2*10^2+5*10^1+6*10^0

The weight of each digit of a decimal number system depends on its relative position with in the number and is called the positional number system.

BINARY NUMBER SYSTEM

- The base or radix of the binary number system is 2. It uses two digits 0 and 1.
- Data is represented in a computer system by either the presence or absence of electronic or magnetic signals in its circuitary.
- This is called a binary, or two state representations of data since the computer is indicating only two possible states or conditions

11001
11001 = 1*2^4+1*2^3+0*2^3+0*2^1+1*2^0
= 16+8+0+0+1
= 25

The binary number system has only two symbols, 0 and 1.They are commonly called a **bit**.

BINARY-DECIMAL CONVERSION

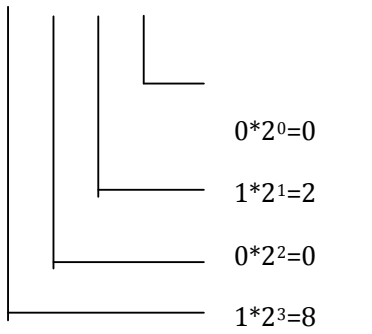
To convert a binary number to its decimal equivalent we use the following expression.

The weight of the nth bit of a number from the right hand side=nth bit * 2^n-1.

After calculating the weight of each bit, they are added to get the decimal value as shown in the following examples.

1010 = 1*2^3+0*2^2+1*2^1+0*2^0
= 8+0+2+0
= 10

1 0 1 0



$0*2^0=0$

$1*2^1=2$

$0*2^2=0$

$1*2^3=8$

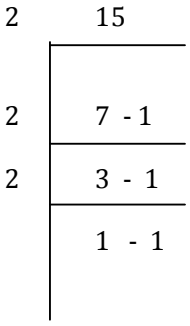
10 $(1010)_2=10_{10}$

DECIMAL-BINARY CONVERSION

Decimal numbers are converted into binary by a method called Double Dabble method.

In this method the mantissa part of the number is repeatedly divided by two and noting the reminders, which will be either 0 or 1.

This division is continued till the mantissa becomes zero. The reminders, which are noted down during the division, are read in the reverse order to get the binary equivalent.



The binary equivalent of 15 is (1111)2

If the decimal number has a fractional part, then the fractional part is converted into binary by multiplying it with 2. only the integer of the result is noted and the fraction is repeatedly multiplied by 2 until the fractional part becomes 0.

2	25	0.25 * 2 = 0.50
2	12 - 1	0.50 * 2 = 1.00
2	6 - 0	1.00 * 2 = 2.00
2	3 - 0	
2	1 - 1	

The binary equivalent of 25.25 is 11001.012

BINARY ADDITION

X	Y	X+Y
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0 carry 1**

The addition of 101101 and 1111 (which are 45 and 15 in the decimal system) is done as follows.

Binary	Decimal
101101	**45**
001111 +	**15 +**
111100	**60**

The addition of 1111011 and 11011 (which are 123 and 27 in the decimal system) is

Binary	Decimal
1111011	**123**
0011011 +	**27 +**
10010110	**150**

BINARY SUBTRACTION

X	Y	X-Y
0	**0**	**0**
0	**1**	**1 with a borrow 1**
1	**0**	**1**
1	**1**	**0**

The subtraction of 1111 from 101101 (which are 15 and 45 in the decimal system) is

Binary	**Decimal**
101101	**45**
001111 -	**15 –**
011110	**30**

The subtraction of 11011 from 1111011 (which are 27 and 123 in the decimal system) is

Binary	**Decimal**
1111011	**123**
0011011 -	**27 –**
1100000	**96**

COMPLEMENTS

- Computers use complemented numbers or complements to perform subtraction.
- In the binary number system, there are two types of complements
  - ✓ 1's complement
  - ✓ 2's complement
- In the decimal number system, there are two types of complements
  - ✓ 9's complement
  - ✓ 10's complement
- The complement is also defined as the difference between the number and the base raised to the nth power minus one.

9'S COMPLEMENT

- The 9's complement of a decimal number is obtained by subtracting each digit of the number from 9.
- For example, the 9's complement of 2 is 7 that of 123 is 876.

999 -

123

876

Add with first number 2

876

878

Result = Negative , So do 9's complement for the result

999

878

121

10'S COMPLEMENT

- The 10's complement of decimal number is obtained by adding one to the 9's complement of the number.
- For example, the 10's complement of 43 and 21 is 22.

First number : 43

Second number : 21

10's complement of 21 → 99 - 21 = 78 + 1 = 79

Add with first number → 43 + 79 = 122

Carry

Result = Positive, so remove the carry.

1'S COMPLEMENT

- The 1's complement in the binary system is similar to the 9's complement in the decimal system. To get 1's complement of a number
- Find the 1's complement of the second number ie) 0 → 1 , 1 → 0
- Add with the first number
- If carry occurs, result = positive, Add carry with the result.
- If no carry occurs, result = negative, Find 1's complement of the result.

Example:

1011 → 1000

First number : 1011

Second number : 1000

1's complement of the second number → 0111

Add with first number → 1011

Carry ← 10010

Result = Positive , So add carry with the result 0010 + 1 = 0011.

- For example, the 1's complement of 1010 is 0101, that of 1111 is 0000.

2'S COMPLEMENT

- The 2's complement of the binary system is similar to the 10's complement in the decimal system. To get the 2's complement of a number, add 1 to the 1's complement of the number.

- Find the 2's complement of the second number. 2's complement = 1's complement + 1.
- Add with the first number
- If carry occurs, result = positive, Remove the carry.
- If no carry occurs, result = negative, Find 2's complement of the result.

Example:

101 → 100

First number : 101

Second number : 10

2's complement of the second number → 100

Add with first number → 101

Carry ← 1001

Result = Positive , So remove the carry , result = 001.

- For example, the 2's complement of 1010 is 0110, that of 1111 is 0001.

SIGNED AND UNSIGNED NUMBER REPRESENTATION

- We put a plus (+) or minus (-) sign before the number to represent its sign.
- To represent a positive number a 0 is placed before the binary number. Similarly, to represent a negative number, a 1 is placed before the binary number.

Example

+15 can be represented by 0 1111
-15 can be represented by 1 1111

- There is only one way to represent a positive number, but there are different ways to represent a negative number. These are
  - ✓ Signed – magnitude representation
  - ✓ Signed 1's complement representation
  - ✓ Signed 2's complement representation
- When all the bits of the computer word are used to represent the number and no bit is used for sign representation, it is called unsigned representation number.

Fixed – point representation of numbers

- In fixed – point representation, all numbers are represented as integers or fraction.
- Signed integer or BCD numbers are referred to as fixed point representation because they contain no information regarding the location of the decimal point or the binary point.
- The binary or decimal point is assumed to be at the extreme right or left of the number.
- For example,
- If we want to multiply 23.15 and 33.45 this can be represented as 2315 X 3345. the result will be 7743675. the decimal point has to be placed by the user to get the correct result, which is 774.3675.

Floating – point representation of numbers

- In most computing applications, fractions are used very frequently. So a system of number representation that automatically keeps track of the position of the binary or decimal point is better than the fixed point is called as the floating – point representation of numbers.
- A number, which has both an integer part and a fractional part, is called a real number or a floating point number.
- These numbers can be either positive or negative. The number can be expressed as a combination of a mantissa and an exponent.

Consider the example, the number 123.23 is represented in the floating – point system as

Sign	**Sign**
0.12323	0 0 3
Mantissa	**exponent**

- The zero is the leftmost position of the mantissa and exponent indicates the plus sign.
- A negative number say –123.23 can be expressed as follows.

Sign	**Sign**
1.12323	0 03
Mantissa	exponent

BINARY CODED DECIMAL

- The BCD is the simplest binary code that is used to represent a decimal number. In the BCD code, 4 bits represent a decimal number.
- For example, 2 is represented as 0010. If a decimal number consist of more than 1 digit, each decimal digit is represented individually by its 4 – bit binary equivalent.
- For example, 123 is represented by

123
0001 0010 001

- Computers perform subtraction using complements and there is difficulty in forming complements when numbers are represented in BCD.

GRAY CODE

- The gray code is binary code. It is used in shaft encoder, which indicates the angular position of a shaft in digital form.

- The bits are arranged in such a way that only one bit changes at a time when we make change from one number to the next. Its use reduces the error in reading shaft position.

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

EXCESS – 3 CODE

- Excess – 3 codes are formed by adding 3 to the decimal number and then forming the binary coded number.
- For example convert 29 to excess – 3 code

2 9
3 3
5 12

5 → 0101
12 → 1100

Therefore 0101 1100 in the excess – 3 code stands for decimal 29

Decimal	BCD	Excess – 3 code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

ASCII Code

- ASCII stands for American Standard Code for Information Interchange. ASCII code is used extensively in small computers, peripherals, instruments and communication devices.
- It is a seven – bit code.
- Microcomputers using 8 – bit word length use 7 bits to represent the Basic code. The 8^{th} bit is used for parity or it may be permanently 1 or 0.
- With 7 bits up to 128 characters can be coded.
- 01000001 is a code for A and 00110000 is a code for 0 and so on.

ASCII – 8 Code

- A new version of ASCII is the ASCII – 8 code, which is an 8 – bit code. With 8 bits, the code capacity is extended to 256 characters.

EBCDIC Code

- EBCDIC stands for Extended BCD Interchange code. It is the standard character code for large computers.
- It is an 8 – bit code without parity
- A 9^{th} bit can be used for parity
- With 8 bits up to 256 characters can be coded
- In ASCII – 8 and EBCDIC, the first four bits known as Zone bits and the remaining 4 bits represent digit values.
- While in ASCII, the first three bits are zone bits and remaining 4 bits represent digit values.

Bits, Bytes and Words

- A byte is a basic grouping of bits (binary digits) that the computer operates on as a single unit.
- It consists of 8 bits and used to represent a character by the ASCII and EBCDIC coding systems.
- The capacity of a computers primary storage and its secondary storage device is usually expressed in terms of bytes.
- A word is a grouping of bits usually larger than a byte that is transferred as a unit between primary storage and the register of the ALU and the control unit.

OCTAL NUMBER SYSTEM

- It refers to the base-8 number system, which uses just unique symbols from 0 to 7.

Decimal to octal conversion

- Consider an example, 125.25_{10}. To convert a decimal to octal number, it is repeatedly divided by 8.

```
8 | 125
  |
8 | 15 - 5
  |
    1  - 7
```

- If the decimal number has a fractional part, then the fractional part is converted into octal by multiplying it with 8.it is repeatedly multiplied by 8 until the fractional part has become zero.

ie) 0.25 * 8 = 2.00

$125.25_{10} = 175.2_8$

HEXADECIMAL NUMBER SYSTEM.

- Hexadecimal number system uses 16 as the base or radix.
- This base-16 number system contains unique symbols. The numbers 0 to 9 and the letters A to F.

Hexadecimal	Decimal	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011

C	12	1100
D	13	1101
E	14	1110
F	15	1111

- To convert a value from hexadecimal to binary, translate each hexadecimal digit into 4-bit binary equivalent.
- For example, the hexadecimal number **3F7A** translates in to the following binary number **0011 1111 0111 1010.**

Hexadecimal to decimal system.

For example, hexadecimal number 2D.2 is converted to decimal by

2 D

D * 16^0 = 13

2 * 16^1 = 32

45

Fractional part

0.2 * 16^{-1} = 0.125

$2D.2_{16} = 45.125_{10}$

Decimal to hexadecimal system.

- To convert,it is repeatedly divided by 16.
- For example, Decimal number 425.125 is converted to hexadecimal by

16	425	
16	26	- 9
	1	- 10

- If the decimal number has a fractional part, then the fractional part is converted into hexadecimal by multiplying it with 16. it is repeatedly multiplied by 16 until the fractional part become 0.

0.125 * 16 = 2.0

$425.125_{10} = 1A9.2_{16}$

2 Marks

1. What is Computer? **(NOV 2017)(APR 2017)**
2. What are the major components of a Computer?
3. Write the difference types of a Computer
4. What is Word length?
5. What is the speed of the computer?
6. What is UNIVAC? **(NOV 2017)**
7. Write the different categories of digital computer
8. How PC is differing from Work station? **(NOV 2018)**
9. What is minicomputer?
10. Write the difference between Mini And Micro computer.
11. What is decimal number system? **(APR 2017)**
12. What is binary number system?
13. What is Complement?
14. Write the Hexa decimal number system.
15. What is octal number system?
16. Write Excess 3 and Gray codes.
17. Explain Byte, Bit, and Word. **(NOV 2018)**
18. Define Batch processing.
19. What is Terminal? Input / Output devices
20. Define A to D converter
21. Define D to A converter.
22. Define Access time.

5 Marks

1. Write the application of Computer. **(NOV 2017) (NOV 2018)**
2. What is Programming Language? **(NOV 2017)**
3. Explain decimal complement.
4. Write the types of Computer. **(APR 2017)**
5. What is binary add and sub code?
6. Explain signed and unsigned.
7. Explain floating point number. **(NOV 2018)**
8. Explain BCD. **(APR 2017)**

10 Marks

1. How the digital computers are classified? **(APR 2017)**
2. Explain Complement system.
3. Explain BCD, EBDCIC, Gray code.
4. Explain octal and Hexa decimal system.
5. Write the component of digital computer system.**(NOV 2018)**
6. Write the different types of computer.
7. Write about High level and Assembly programming language. **(NOV 2017)**

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

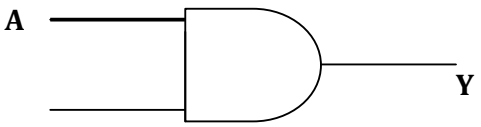
A gate is simply an electronic circuit, which operates on one or more input signals to produce an output signal.

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.

Circuit diagram



B

Truth table

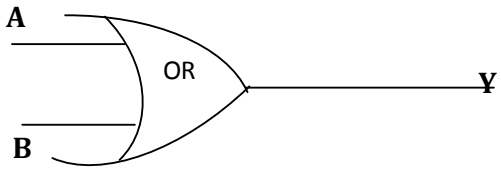
A	B	Y
0	0	0
0	1	0
1	0	0

1	**1**	**1**

OR GATE

The OR gate is similar to the operation of arithmetic addition.

Circuit diagram



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.

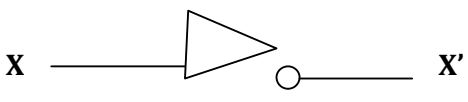
Truth table

A	B	Y
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.

Circuit diagram



Truth table

X	X'
0	**1**

1	**0**

The NOT gate produce high output, when the input is low and low output when the input is high.

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.

Evaluation of an expression containing parenthesis

- Consider the example **X + Y(X' + Y').** since the expression contains both **X'** and **Y'**. The values of **X** and **Y** are complemented and then **X' + Y'** is performed.
- Then the value of **X' + Y'** is **AND**ed by the values of Y forming **Y(X'+Y')** in the table.
- Finally the value of **Y(X'+Y')** are added (**OR**ed) to the values for **X** and the expression is evaluated.
- The final column indicates that the function **X + Y(X' + Y')** is equivalent to **X+Y**.
- This is called proof by perfect induction which means evaluating each possible values for the variables and noting that both expressions then have the same value.

X	Y	X'	Y'	X'+Y'	Y (X'+Y')	X+Y (X'+Y')
0	0	1	1	1	0	0
0	1	1	0	1	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	1

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

1.	**0 + X**	**X**
2.	**1 + X**	**1**
3.	**X + X**	**X**
4.	**X + X'**	**1**
5.	**0 . X**	**0**
6.	**1 . X**	**X**
7.	**X . X**	**X**
8.	**X . X'**	**0**
9.	**X'**	**X**
10.	**X + Y**	**Y + X**

11.	**X . Y**	**Y . X**
12.	**X+(Y+Z)**	**(X+Y)+Z**
13.	**X(YZ)**	**(XY)Z**
14.	**X(Y+Z)**	**XY + XZ**
15.	**X + XZ**	**X**
16.	**X(X+Y)**	**X**
17.	**(X+Y)(X+Z)**	**X + YZ**
18.	**X + X'Y**	**X+Y**
19.	**XY+YZ+YZ**	**XY + Z**

Commutative law

Commutative law expresses the fact that the order in which a combination of terms is performed does not affect the result of the combination.

1. X **+** Y = Y + X(Rule 10) – Commutative law of addition
2. X. Y = Y**.** X(Rule 11) – Commutative law of multiplication.

Associative law

Associative law states that in the logical addition of several terms, the sum which will be obtained if the the first term is added to the second and then the third term is added will be the same as the sum obtained if the second term is added to the third and then the first term is added .

1. X + (Y + Z) = (X + Y) +Z- (Rule -12)- Associative law of addition
2. X (YZ) = (XY) Z(Rule 13) – Associative law of multiplication

Distributed law

Distributive law states that the product of a variable(X), a sum(Y+Z) is equql to the sum of the products of the variable multiplied by each term of the sum, X (Y + Z) = X. Y + X. Z.

1. X (Y + Z) = X. Y + X. Z
2. X + Y. Z = (X + Y). (X+ Z)

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)

XX' + XZ.

XZ. **XX' =0**

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

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 un complemented 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

**********ALL THE BEST**********

UNIT-II

2 Marks

1. Write the use B. A
2. What is AND, OR, NOR gate? **(NOV 2018) (NOV 2017)**
3. What is double conversion? **(NOV 2018)**

5 Marks

1. Explain about Gates. **(NOV 2018)**
2. How to evaluate the logical expression? **(NOV 2017)**
3. What is demorgan's theorem? **(NOV 2018)**
4. Write the basic duality of Boolean algebra.

10 Marks

1. Write the Basic law of Boolean Algebra.**(NOV 2017)**
2. Write the derivation of Boolean algebra. **(NOV 2018)**

UNIT III

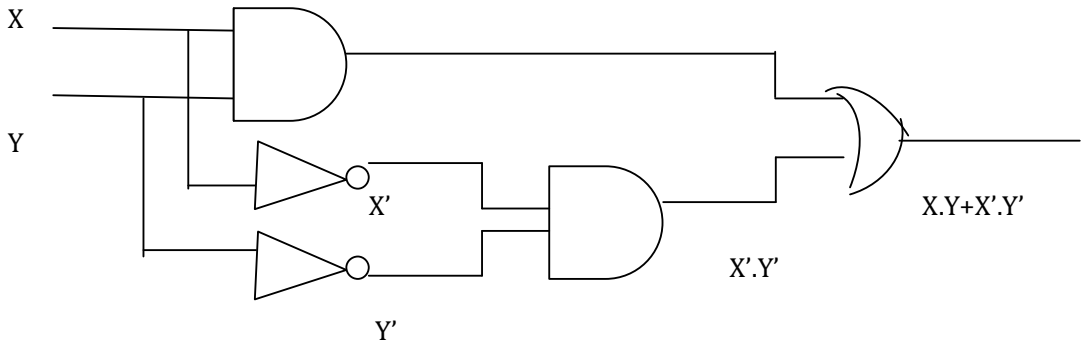
Interconnecting Gates – Sum of products (SOP) and Products of sums (POS) – Derivation of products of sums expressions – Derivation of three Input variable expression – NAND gates and NOR gates - The Map method for simplifying expressions – Sub cube and covering – product of sums expressions – Don't cares.

Interconnecting gates

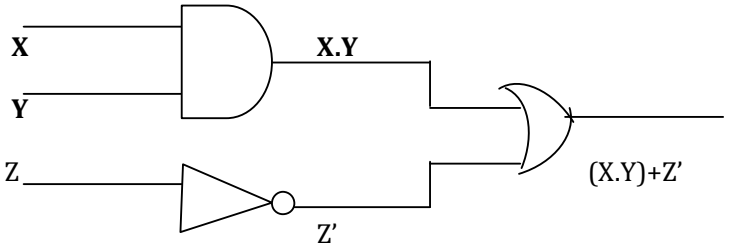
- The OR gates, AND gates and inverters can be interconnected to form gating or logic networks.
- This interconnecting study of gates is very useful to build switching networks.
- The Boolean algebra expression corresponding to a given gating network can be derived by systematic progressing from input to output on the gates.

Example

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



2. (X . Y) + Z'



SUM OF PRODUCTS AND PRODUCT OF SUMS

- Certain types of Boolean algebra expression lead to gating networks which are more desirable from most implementation viewpoints.

Two usable forms for Boolean expressions:

Terms:

Product term

- A product term is a single variable or the logical product of several variables.
- The product term may or may not be complemented.

Sum term

- A sum term is a single variable or the sum of several variables.
- The sum term may or may not be complemented.

Example

X is both a sum term and product term.

X.Y is a product term.

X+Y is a sum term.

X+Y.Z is neither a sum term nor a product term.

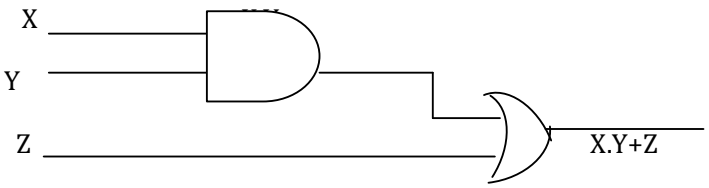
Sum of products expression

- A sum of products expression is a product term or several product terms logically added.
- This is also called as Maxterm.

Ex: X.Y+Z

X'.Y'+X'.Y.Z'

X+Y



Product of sums expression

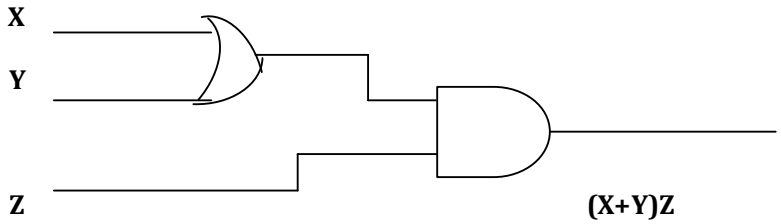
A product of sums expression is a sum term or several sum terms logically multiplied

This is also called as Minterm.

Ex: (X+Y).(X+Y').(X'+Y')

(X+Y+Z).)X+Y').(X'+Y')

(X+Y)Z



DERIVATION OF PRODUCT OF SUMS EXPRESSION

The method for arriving at the desired expression is as follows.

- Construct a table for the input and output values.
- Construct an additional column of sum terms containing complemented and uncomplemented variables.
- In each row of the table, a sum term is formed.
- If the input value for a given variable is 1, then the variable will be complemented and if 0, then it is not complemented.
- The desired expression is the product of the sum terms from the rows where output is 0.
  EX:

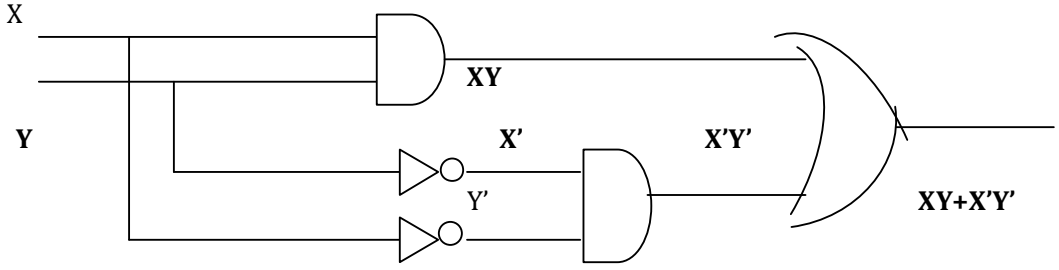
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

X	Y	Z	Sum terms	M
0	0	1	X + Y	M0
0	1	0	X + Y'	M1
1	0	0	X'+ Y	M2
1	1	1	X' + Y'	M3

- Selecting those sum terms for which the output is **0** and multiplying them now form a product of sums expression.
- **0** appears in second and third rows. So,

  Z = (X + Y')(X' + Y)
  Z = XX'+XY+X'Y'+Y'Y
  Z = XY+X'Y'.

The sum of products expression is found to be **XY+X'Y'**.



DERIVATION OF A THREE INPUT VARIABLE EXPRESSION.

- While deriving a three input variable expression, two columns will be added this time, one containing the sum of products and the other containing product of sums terms.

Example:

Draw the sum of products and product of sums and logic gates for

Minterm (m2, m3, m6)

Maxterm (m0, m1, m4, m5, m7)

- The sum of products term are derived from the minterm **m2, m3 , m6**.
- The product of sums are derived from the term **M0, M1, M4, M7**.
- The sum of products have output 1.

Sum of products expression

X	Y	Z	O/P	Minterm	Fundamental product
0	0	0	0	m0	X'Y'Z'
0	0	1	0	m1	X'Y'Z
0	1	0	1	m2	X'YZ'
0	1	1	1	m3	X'YZ
1	0	0	0	m4	XY'Z'
1	0	1	0	m5	XY'Z
1	1	0	1	m6	XYZ'
1	1	1	0	m7	XYZ

Sum of products term = m2, m3, m6

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

= X' (YZ' + YZ) + XYZ'

= X' (Y+Y) + XYZ'

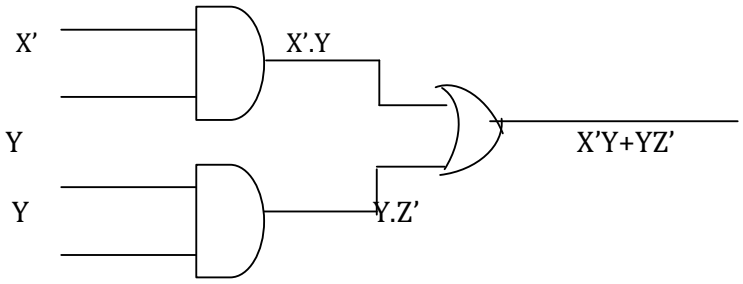
= X'Y + XYZ'

= Y (X' + XZ')

= X'Y+YZ'

Sum of products expression is **Y (X' + Z')**

Logic gate for sum of products term



Product of sums expression

X	Y	Z	O/P	Maxterm	Fundamental sum
0	0	0	0	M0	X+Y+Z
0	0	1	0	M1	X+Y+Z'
0	1	0	1	M2	X+Y'+Z
0	1	1	1	M3	X+Y'+Z'
1	0	0	0	M4	X'+Y+Z
1	0	1	0	M5	X'+Y+Z'
1	1	0	1	M6	X'+Y'+Z
1	1	1	0	M7	X'+Y'+Z'

Product of sums term = M0, M1, M4, M5, M7

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

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

= (XX'+XY+X'Y+YY)(X'+Y'+Z')

=(XY+X'Y+Y)(X'+Y'+Z')

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

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

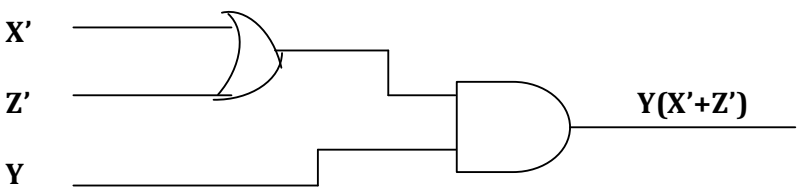
= Y+YX'+YY'+YZ'

= Y+Y (X'+Z')

= Y (X'+Z')

The product of sums expression is **= Y (X'+Z')**

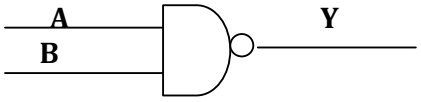
Logic gate for product of sums term



NAND GATES OR UNIVERSAL LOGIC GATE

- The NAND gate produce the high output when any one of the input is zero and produce the low output when both the inputs are one. The NAND gate is the opposite of AND gate.

Circuit diagram



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

OR-to-NAND gate network

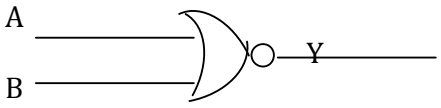
- If one of the two input lines to a two-input NAND gate contained the input A+B and the other containing A+B. The output from the NAND gate would be

  (A+B)(C+D) = AB + CD

NOR GATE

- The NOR gate produce high output when both the inputs are low and a high output when any one of the input is high.

Circuit diagram



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

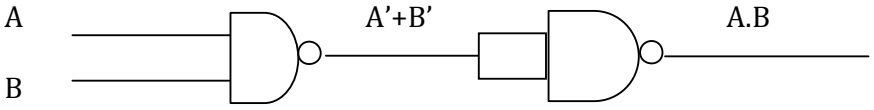
AND-to-NOR gate networks

If one of the input lines to a two-input NOR gate containing the signal **A . B** and the other input lines contained the signal **C . D**. The output from the NOR gate would be

(A.B + C.D) = (A + B) (C +D)

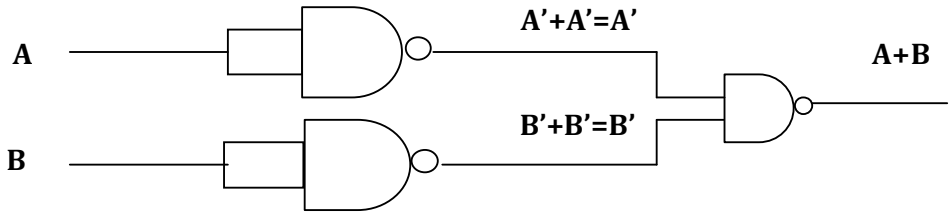
AND from NAND gates

The AND gate can be made from two NAND gates.



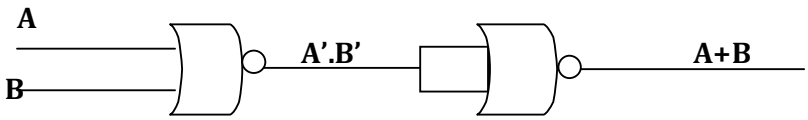
OR gate from NAND gates

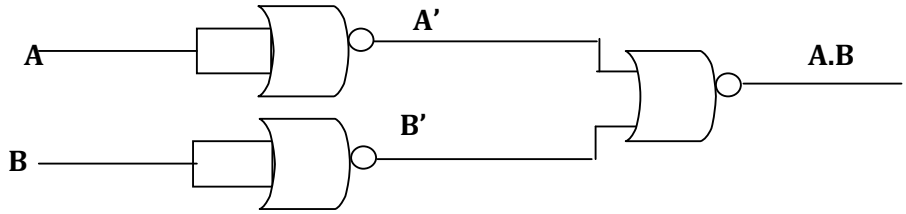
The OR gate can be made from the NAND gates.



AND and OR gates from NOR gates

- The NOR gate can also be used to form any Boolean function and a two-level NOR gate network yields the same function as a two-level OR-to-AND gate network.





MAP METHOD FOR SIMPLIFYING EXPRESSIONS

Karnaugh map

- The table of combinations provides a nice way to list all values of a Boolean function.
- The particular type of map used to represent the sum of products expression for the function is called karnaugh map.
- The map is filled in by placing 1s in the squares or cells for each term which leads to a output 1.
- Each case lists 2^n different product terms, formed in exactly n variables, in a different square.

Two variable map method

- Two variables are represented in four minterms.
- The map consists of four squares, one for each minterm.

	X'	X
Y'	X'Y'	XY'
Y	X'Y	XY

Three variable map method

- Three variables are represented in 2^3 or 8 different minterms.
- The map consists of 8 squares.

X'Y'Z'	X'YZ'	XYZ'	XY'Z'
X'Y'Z	X'YZ	XYZ	XY'Z

Four variable map method

- Four variables are represented in 16 different minterms.
- The four-map method consists of 16 squares each for one minterm.

W'X'Y'Z'	W'XY'Z'	WXY'Z'	WX'Y'Z'
W'X'Y'Z	W'XY'Z	WXY'Z	WX'Y'Z
W'X'YZ	W'XYZ	WXYZ	WX'YZ
W'X'YZ'	W'XYZ'	WXYZ'	WX'YZ'

Pairs

- The map contains a pair of 1s that are horizontally or vertically adjacent to one another.

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

Quad

- The quad is a group of four 1s that are horizontally or vertically adjacent.

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

Octet

- Octet is a group of eight 1s, which can able to eliminate 3 variables and its complement.

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

Overlapping

- When more than one kind of techniques are used with 1s is called as Overlapping.

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

Rolling or continuous

- If the left side edge and the right side edge to be touching, the map is considered to be rolled or rolling the map.
- To represent this rolling, half circles are drawn around each pair.

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

Karnaugh maps for the expression :

XYZ'+ X'Y'Z

X'Y'+X'Y

1	
1	

1		1	

Sub cubes and covering

0	0	0	0
1	1	0	0

- A sub cube is a set of exactly 2 adjacent cells containing 1s.
- For m=0, sub cube consists of a single cell.
- For m=1, sub cube consists of a two adjacent cells.
- For m=2, sub cube consist of four adjacent cells.
- A sub cube containing 2 cells as 2 cube, 4 cells as 4 cube, 8 cells as 8 cube.

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

Product of sums expression AND Don't Care's

- There is a situation in which certain outputs are not specified in a problem.
- The outputs will never appear and such outputs are called Don't care outputs.
- Since the output values are of no importance, they may be filled in with either a 1 or 0.

W	**X**	**Y**	**Z**	Function Values
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0

0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	d
1	0	1	1	d
1	1	0	0	d
1	1	0	1	d
1	1	1	0	d
1	1	1	1	d

1	1	d	1
		d	
1	1	d	d
		d	d

UNIT-III

2 Marks

1. What are combinational Networks? **(NOV 2017)**
2. Write the usable forms of B.A
3. Write about NAND, NOR with suitable truth table. **(NOV 2017)**
4. What is Inverters? **(NOV 2018)**
5. What is POS?**(NOV 2018)**
6. What is SOP?

5 Marks

1. Write the Sum of Product and Product of sum. **(NOV 2018)**
2. Explain about NAND and NOR gate. **(NOV 2017) (NOV 2018)**
3. Simplify expression sub cube and covering**(NOV 2017)**.

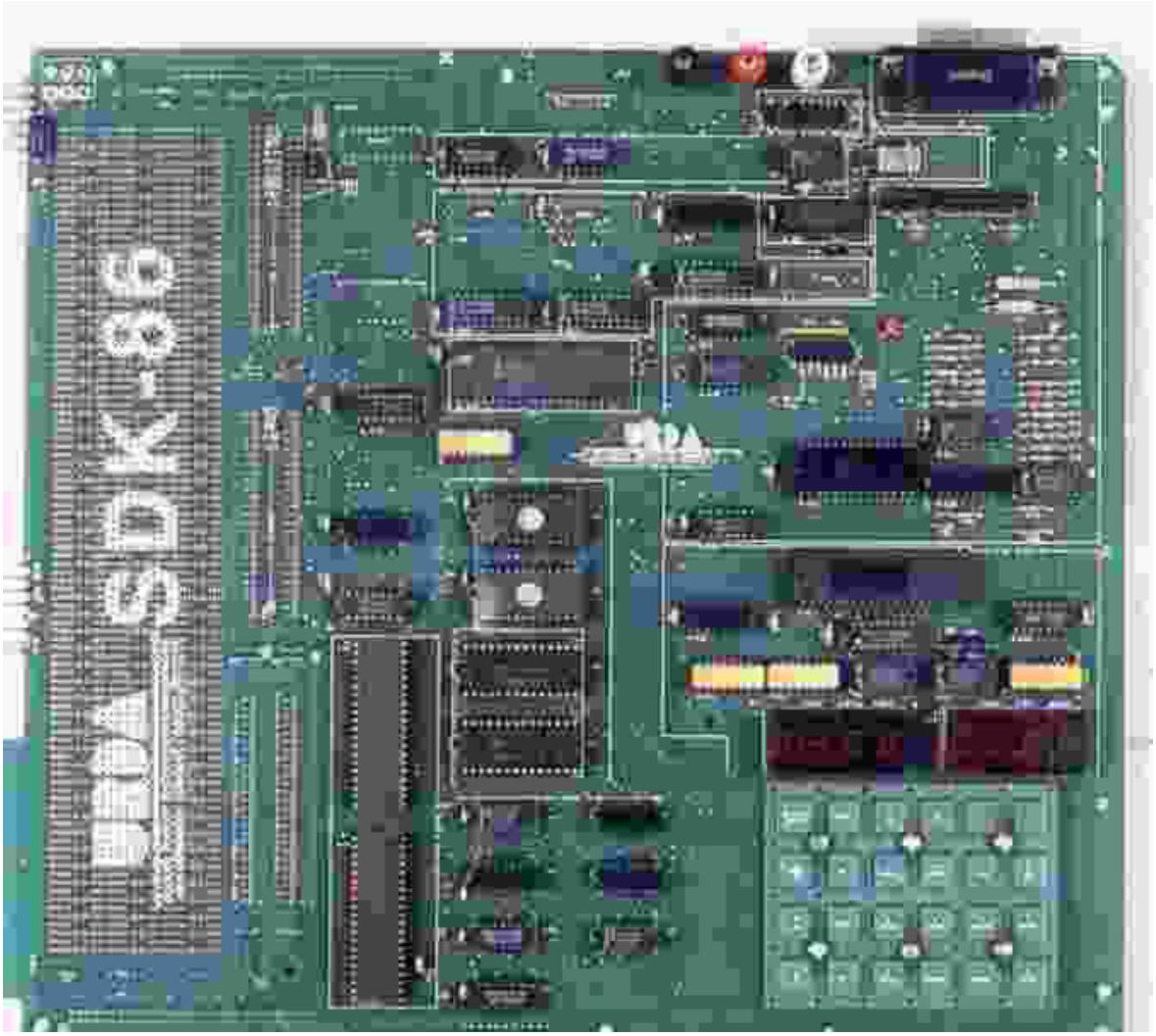
10 Marks

1. Write in briefly about KMAP method and simplify expression.
2. Write briefly about SOP and POS. **(NOV 2017)**
3. How to simplify the expression.
4. Explain about Inter-connecting Gates. **(NOV 2018)**

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. 14

MICROPROCESSORS



The word comes from the combination micro and processor. Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor's design.

Micro is a new addition. In the late 1960's, processors were built using discrete elements. These devices performed the required operation, but were too large and too slow.

In the early 1970's the microchip was invented. All of the components that made up the processor were now placed on 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. This instruction set defines what the microprocessor can and cannot do.

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: The microprocessor has a very narrow view on life. 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".**

The number of bits in a Microprocessor's word, is a measure of its "abilities". Words, Bytes, etc.The earliest microprocessor (the Intel 8088 and Motorola's 6800) recognized 8-bit words. They processed information 8-bits at a time. That's why they are called "8-bit processors". They can handle large numbers, but in order to process these numbers, they broke them into 8-bit pieces and processed each group of 8-bits separately.

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.

Produces: For the user to see the result of the execution of the program, the results must be presented in a human readable form. The results must be presented on an output device.

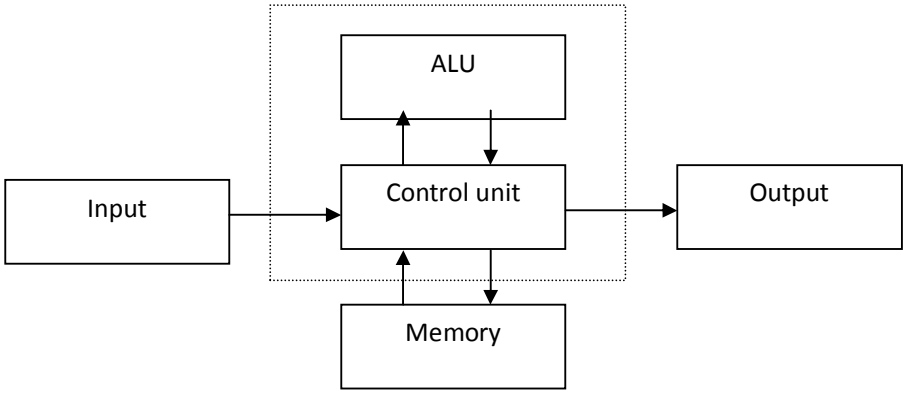
This can be the monitor, a paper from the printer, a simple LED or many other forms

Microprocessor

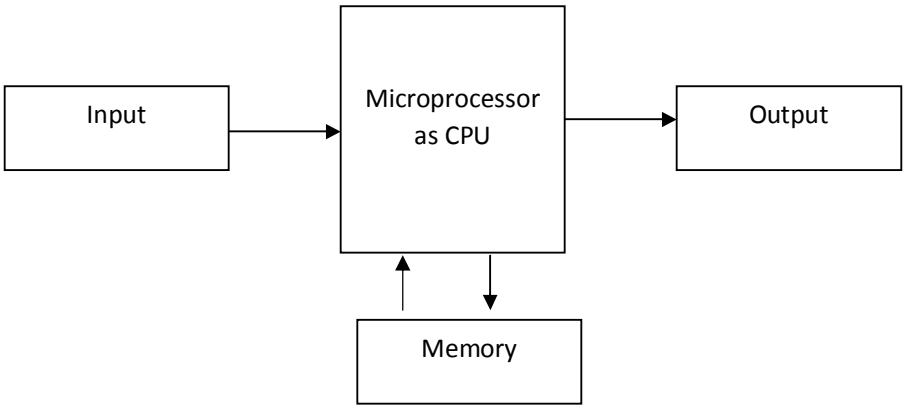
· **Digital** Computer-a programmable machine that processes binary data. It includes four components: CPU (ALU plus control unit), memory, input, and output.

· **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.

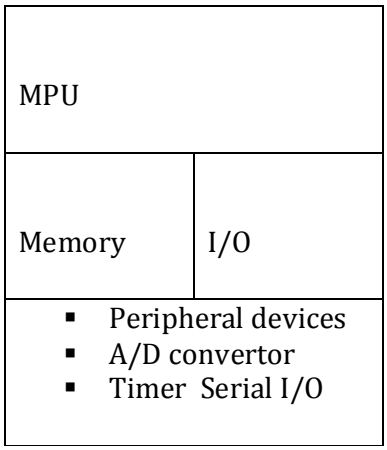
Block diagram of a computer



Block diagram of Microprocessor



Block diagram of Microcontroller



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

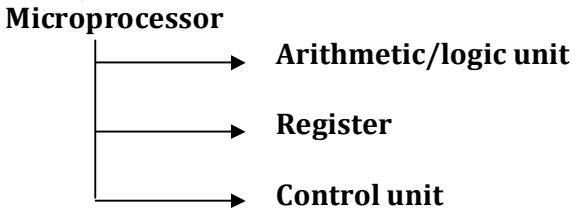
· **Control Unit**-The group of circuits that provides timing and signals to all operations in the computer and controls data flow.

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

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

Microprocessor-Based Systems

- The microprocessor is a clock-driven semiconductor device.
- It consists of electronic logic circuits manufactured by using either a large-scale integration (LSI) or very large-scale integration (VLSI) technique.
- It is capable of performing various computing functions. The microprocessor can be divided into three segments.



Arithmetic logic unit

- Various computing functions are performed on data.
- This unit performs arithmetic operations such as addition and subtraction and logic operation as AND, OR and exclusive OR.

Register array

- This area consists of various registers.
- These are primarily used to store data temporarily during the execution of a program.

Control unit

- This unit provides the necessary timing and control signals to all the operations in the microcomputer.
- It controls the flow of data between the microprocessor and memory and peripherals.

Memory

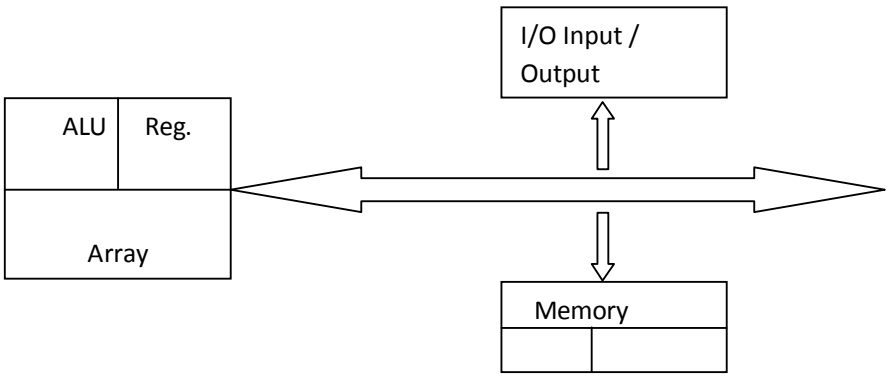
- Memory stores binary information as instructions and data provides information to the microprocessor.
- To execute programs, the microprocessor reads instructions and data from memory and performs the computing operations in the ALU.
- The memory block has two sections
  - ✓ Read-only memory (ROM)
  - ✓ Read/write memory (R/WM)

I/O (input/output)

- This section transfer binary information from the outside world to the microprocessor such as keyboard, switches etc.,
- The output section transfers data from the microprocessor to the outside world. They include magnetic tape etc.,

System bus

- It is a communication path between the microprocessor and peripherals.
- Group of wires to carry bits.
- There are several buses in the system and all peripherals share the common bus.
- The microprocessor communicates with only one peripheral at a time.



· **ROM**-Read-Only Memory. A memory that stores binary information permanently. The information can be read from this memory but cannot be altered.

· **R/WM**-Read/Write Memory. A memory that stores binary information during the operation of the computer. This memory is used as a writing pad to write user programs and data. The information stored in this memory can be easily read and altered.

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.

The term "nibble," which stands for a group of four bits, is also found in popular computer magazines and books. (A byte has two nibbles.)

TABLE 1

Microcomputer Applications

TABLE 1
Microcomputer Applications

Characteristics	Types: Microcomputer (Personal Computer)	Types: Single-Board Microcomputer	Types: Single-Chip Microcomputer
Price range	$1,000–$5,000	$100–$800	<$50
System memory size (ROM & R/WM)	512K—1.2M	256 bytes–2K	64–128 bytes
I/O	ASCII keyboard, CRT	Hex keyboard (rarely ASCII) LEDs and seven-segment LEDs	Keyboard, switches, alpha-numeric LEDs*
Languages used	Various types of high-level languages, assembly	Assembly	Assembly and C
Applications	Small business applications, word processing, instructional applications	Evaluation of microprocessors, assembly language instruction; as a subsystem	Industrial control and embedded systems
Storage memory	Floppy and hard disks	No storage memory	No storage memory

*These I/O devices will vary according to the type of microprocessor-based products.

The instruction is defined as a complete task (such as Add) the microprocessor can perform ; it can be made up of one or more words. Each machine has its own set of instructions based on the design of its CPU or its microprocessor. To be intelligible to the microprocessor, instructions must be written in binary language, also known as machine language.

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.

INC A INC stands for increment, and A represents the accumulator. This symbol suggests the operation of incrementing the accumulator content by one.

ADD A, B ADD stands for addition, and A and B represent the contents in the accumulator and register B, respectively. This symbol suggests the addition of the contents in register B and the accumulator.

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

Writing and Executing an Assembly Language Program

To write and execute an assembly language program manually on a single-board computer, with a Hex keyboard for input and LEOs (or seven-segment LEOs) for output, the following steps are necessary:

1. Write the instructions in mnemonics obtained from the instruction set supplied by the manufacturer.

2. Find the hexadecimal machine code for each instruction by searching through the set of instructions.

3. Enter (load) the program in the user memory in a sequential order by using the Hex keyboard as the input device.

4. Execute the program by pressing the Execute key. The answer will be displayed by the LEOs.

When the user program is entered by the keys, each entry is interpreted and converted into its binary equivalent by the monitor program. and the machine code is stored as eight bits in each memory location in a sequence. When the Execute command is given, the microprocessor fetches each instruction, decodes it. and executes it in a sequence until the end of the program.

The manual assembly procedure is commonly used in single-board microcomputers and is suited for small programs

The assembler is a program that translates the mnemonics entered by the ASCII keyboard into the corresponding binary machine codes of the microprocessor. Each microprocessor has its own assembler because the mnemonics and machine codes are specific to the microprocessor being used, and each assembler has certain rules that must be learned by the programmer.

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. The primary difference between a compiler and an interpreter is in the process of generating machine code. The compiler reads the entire program first and then generates the object code, while the interpreter reads one instruction at a time, produces its object code, and executes the instruction before reading the next instruction.

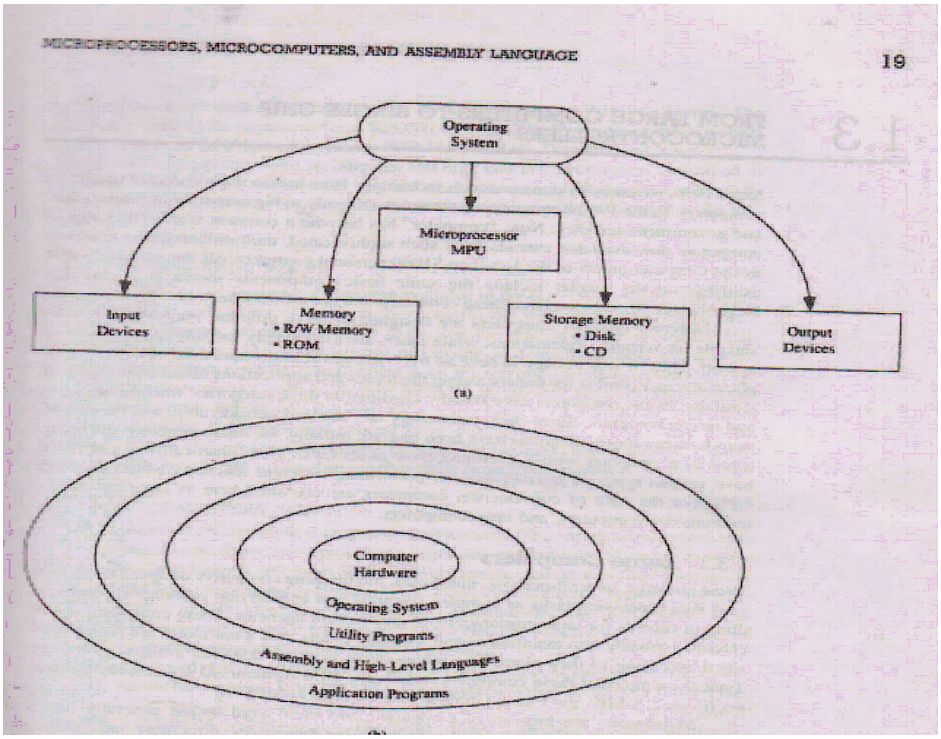
Thus, assembly language programs are compact and require less memory space; they are more efficient than the high-level language programs. The primary advantage of high-level languages is in troubleshooting programs, also known as debugging. It is much easier to find errors in a program written in a high-level language than to find them in a program written in assembly language.

In certain applications such as traffic control and appliance control, where programs are small and compact, assembly language is suitable. Similarly, in such real-time applications as converting a high-frequency waveform into digital data, program efficiency is critical. In real-time applications, events and time should closely match with each other without significant delay. Therefore, assembly language is highly desirable in these applications.

Operating systems:

The interaction between the hardware and the software is managed by a set of programs called an operating system of a computer. The computer transfer information constantly between memory and various peripherals such as printer, keyboard and

video monitor. It also stores programs on disk. The operating system is responsible primarily for storing information on the disk and for communication between the computer and its peripherals.



FROM LARGE COMPUTERS TO SINGLE-CHIP MICROCONTROLLERS

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. Thus, it is necessary to have an overview of the entire spectrum of computer applications as a context for understanding the topics and applications discussed in this text.

In 1970s, computers were broadly classified in three categories as Mainframe, Mini and Microcomputers. Since then, technology has changed considerably, and the distinctions between these categories have been blurred. Initially,the microcomputer was recognized as a computer with the microprocessor as its CPU. Now practically all computers have various types of microprocessors performing different functions within the large CPU. For the sake of convenience, computers are classified as large computers, medium-size computers 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. These computers can be classified broadly into mainframe and super computers, mainframe are further classified according to their sizes.

Mainframes are high-speed computers, and their word length generally ranges from 32 to 64 bits. They are capable of addressing megabytes of memory and handling all types of peripherals and a large number of users. They are the fastest computers, capable of executing billions of instructions per second, and are used primarily in research dealing with problems in areas such as global climate and high-energy physics.

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.

MICROCOMPUTERS:

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.

The floppy disk is a magnetic medium similar to a cassette tape except that is round in shape, like a record. Information recorded on these disks can be accessed randomly using disk drives. The hard disk is similar to the floppy disk except that the magnetic material is coated on a rigid aluminum base that is enclosed in a sealed container and permanently installed in a microcomputer. The hard disk has a large storage capacity; therefore, large and frequently used programs such as compilers, interpreters, system programs, and application programs are stored on the disk. The floppy disk is generally used for user programs and to make backup copies. The microcomputers are further classified according to their size, weight and portability. They are called laptop and notebook. These computers can be battery operated or use AC power and are carried easily from place to place. These are called laptop **(instead of desktop**) because the size is small enough to place them in one's lap.

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. Typically, these microcomputers include an 8-bit or 16-bit microprocessor, from 256 bytes to 8k bytes of user memory, a Hex keyboard, and seven-segment LEDs as display. The interaction between the microprocessor, memory , and I/Os in these small systems is looked after by a program called a system **monitor program**, which is generally small in size , stored in less than 2k bytes of ROM.When a single-board microcomputer is turned on , the monitor program is in charge of the system ; it monitors the keyboard inputs, interprets those keys ,stores programs in memory, sends system display to the LEDS and enables the execution of the user programs.

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.These are complete microcomputers on a chip, they are also known as microcontrollers. They are used primarily for such functions as controlling appliances and traffic lights.

Microprocessor architecture and microcomputer systems

Microprocessor architecture and its operations

- 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).

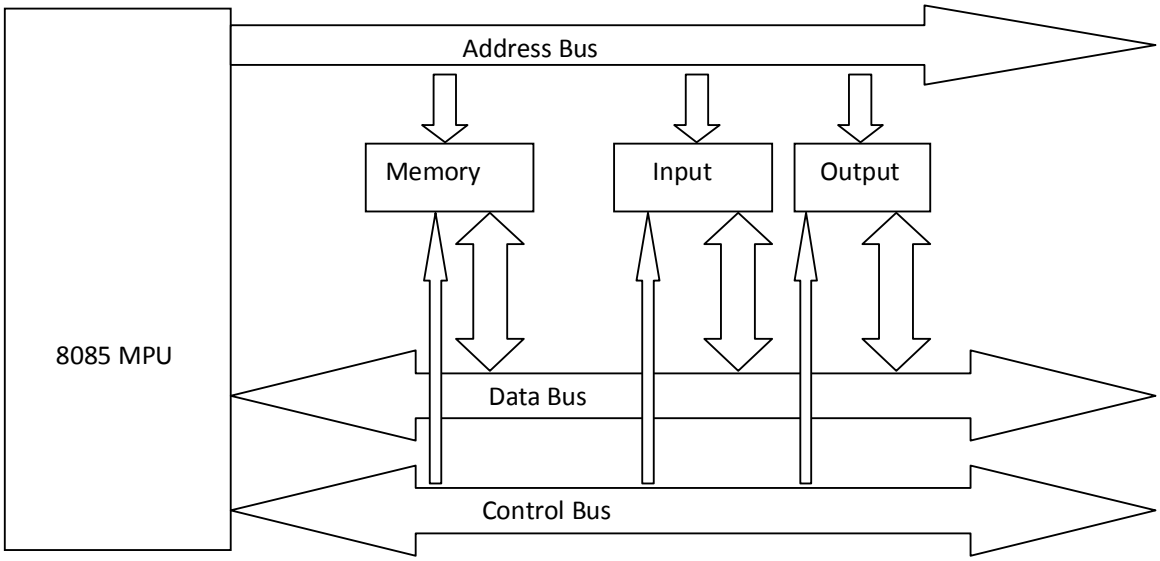
Microprocessor initiated operations and 8085 bus organization

- 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).

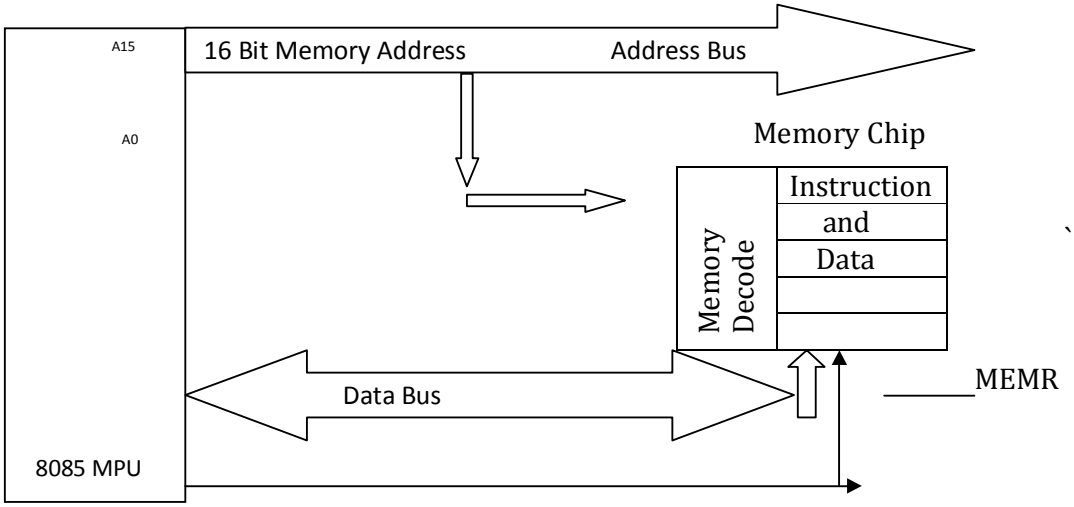
Control bus

- 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.

Internal data operations and the 8085 registers

- The internal architecture of the 8085 microprocessor determines how and what operations can be performed with the data.

- These operations are
    1. Store 8-bit data.
    2. Perform arithmetic and logical operations
    3. Test for conditions.
    4. Sequence the execution of instructions.
    5. Store data temporarily during execution in the defined R/W memory locations called the stack.



The 8085 programmable registers.

Flags

- The ALU includes 5 flip-flops which are set or reset after an operation according to the result.
- The results are stored mostly in the accumulator.

Accumulator (8)	Flag Register (8)
B Register (8)	C Register (8)
D Register (8)	E Register (8)
H Register (8)	L Register (8)
Program Counter (16)	
Stack Pointer (16)	

Z-zero flag

It is set to 1 when the result is zero otherwise it is reset.

CY-carry flag

If an arithmetic operations results in a carry, the CY flag is set, otherwise it is rweset.

S-sign flag

This flag is set if bit D7 of the result is 1.

P-parity flag

If the result has an even number of 1s, the flag is set. For an odd number of 1s, the flag is reset.

AC-auxiliary carry

If an arithmetic operation, when a carry is generated by digit D3 and passed to digit D4, the AC flag is set. The AC flag is used internally for the BCD arithmetic operations.

Peripheral or externally initiated operation

External devices can initiate the following operations, for which individual pins on the microprocessor are assigned:

Reset

- When the RESET pin is activated by an external key (also called a reset key), all internal operations are suspended and the program counter is cleared.
- The program execution can again begin at the zero memory address.

Interrupt

- The microprocessor can be interrupted from the normal execution of instructions and asked to execute some other instructions called a service routine.
- The microprocessor resumes its operation after completing the service routine.

Ready

- The 8085 has a pin called READY. If the signal at this READY pin is low, the microprocessor enters into a wait state.
- This signal is used primarily to synchronize slower peripherals with the microprocessor.

Hold

- When the HOLD pin is activated by an external signal, the microprocessor gives up control of buses and allows the external peripheral to use them.

Memory

- Memory is an essential component of a microcomputer system.
- It stores binary instructions and data for the microprocessor.
- The various types of memory can be classified into two groups.
    - Prime memory (main memory)
    - Storage memory

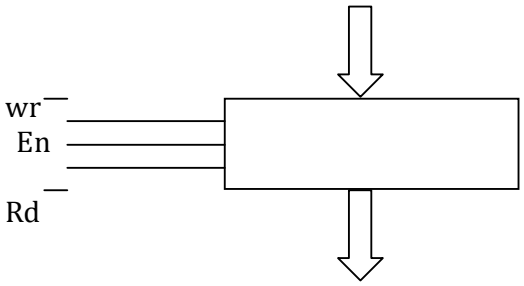
Example: Primary: Read/Write memory(R/WM) and Read-Only Memory (ROM)

- The number of bits stored in a register is called a memory word; memory devices are available in various word sizes.
- To communicate with memory, the MPU should be able to
    - ✓ Select the chip
    - ✓ Identify the register
    - ✓ Read from or write into the register

Flip-flop or latch as a storage element

- A memory is a circuit that can store bits-high or low, generally voltage levels or capacitive charges representing 1 and 0.
- A flip-flop or a latch is a basic element of memory.
- To write or store a bit in the latch, we need an input data bit (D) and the enable signal (EN).

- We can write into the latch by enabling the input buffer and read from it by enabling the output buffer.
- The write signal WR and the read signal RD acts as active low signals.
- The latch, which can store one binary bit, is called a memory cell.



- The figure shows four cells or latches grouped together. This is a register, which has four input lines and four output lines and can store four bits; thus the size of the memory word is four bits.

- To write into or read from any one of the registers, a specific register should be identified or enabled. This is a simple decoding function; a 2-to-4 decoder can perform that function.
- Two more input lines A1 and A0, called address lines, are required to the decoder. These two input lines can have four different bit combinations (00,01,10,11).

- Memory chip M1 is selected when A3 and A2 are both 0; therefore, registers in this chip are identified with the addresses ranging from 0000 to 0011(0 to 3).
- Similarly, the addresses of memory chip M2 range from 1000 to 1011(8 to B); this chip is selected only when A3 is 1 and A2 is 0.
- We need three lines to identify eight registers: two for registers and one for Chip select. However we used the fourth line for chip select also. This is called complete or absolute decoding.
- The requirements of a memory chip can be summarized as follows:
    1.A memory chip requires address lines to identify a memory register.
    2.A memory chip requires a chip select (CS) signal to enable the chip.
    3.The address lines connected to CS select the chip and the address lines connected to the address lines of the memory chip select the register.
    4.The control signal Read(RD) enables the output buffer and data from the selected register are made available on the output lines.

- A model of a typical memory chip represents the R/W memory and represents the Read-Only memory; the only difference between the two as far as addressing is concerned is that ROM does not need the WR signal.
- Internally, the memory cells are arranged in a matrix format-in rows and columns.

Memory map and addresses

- A memory map is a pictorial representation in which memory devicers are located in the entire range of addresses.
- Memory addresses provide the locations of various memory devices in the system.
- In computer systems, we define 1024 as 1K.

- The 1K-byte memory chip has 1024 registers with 8 bits each. Similarly, a group of 256 registers is defined as one page and each register is viewed as a line to write on.
- The term memory map is used generally for the entire address ranges of the memory chips in a given system.

Memory address range of a 1K memory chip

- If a chip includes more than 256 registers (e.g., 512 or 1024 registers), the number of low-order address lines will be higher than that of the high-order address lines.
- The memory chip has 1024 registers; therefore 10 address lines (A9-A0) are required to identify the registers. The remaining six address lines(A15-A10) of the microprocessor are used for the Chip Select (CS) signal.
- The memory chip is enabled when the address lines A15-A10 are at logic 0. the address lines A9-A0 can assume any address of the 1024 registers, starting from all 0s to all 1s.

Memory address lines

- For a chip with 256 registers, we need 256 binary numbers to identify each register. Each address line can assume only two logic states (0 and 1). Therefore we need to find the power of 2 that will give us 256 combinations.

Example:

Find x where $2^x = 256$. By taking logon both sides, we get

$\log 2^x = \log 256 \rightarrow x \log 2 = \log 256$

$x = \log 256 / \log 2 = 8$

Here x represents the number of address lines needed to obtain 256 binary numbers.

Memory word size

- Memory devices are available in various word sizes (1, 4 and 8) and the size of a memory chip is generally specified in terms of the total number of bits it can store.
- If a memory chip of size 1024 x 4 has 1024 registers and each register can store four bits, thus it can store a total of 4096 (1024 x 4).
- To design 1K-byte(1024 x 8) memory, we will need two chips, each chip will provide four data lines.

Memory and instruction fetch

- The primary function of memory is to store instructions and data and to provide that information by sending the address of a specific memory register on the address bus and enables the data flow by sending the control signal.
- To fetch the instructions located in memory location 2005H, the following steps are performed:
    1. The program counter places the 16-bit address 2005H of the memory location on the address bus.
    2. The control unit sends the memory read control signal (MEMR, active low) to enable the output buffer of the memory chip.
    3. The instruction (4FH) stored in the memory location is placed on the data bus and transferred (copied) to the instruction decoder of the microprocessor.
    4. The instruction is decoded and executed according to the binary pattern of the instruction.

Memory classification

- The memory can be classified into two groups: prime (system or main) memory and storage memory and storage memory.

- The R/WM and ROM are examples of prime memory; this is the memory the microprocessor uses in executing and storing programs.
- The other group is the storage memory, such as magnetic disks and tapes. This memory is used to store programs and results after the completion of program execution.
- The memory is divided into two groups:
    - ✓ Read/Write memory(R/WM)
    - ✓ Read only memory (ROM)

ROM(Read Only Memory)

- The ROM is a nonvolatile memory: it retains stored information even if the power is turned off.
- The concept underlying the ROM can be explained with the diodes arranged in a matrix format.
- The horizontal lines are connected to vertical lines only through the diodes.
- Each of the eight Horizontal rows can be viewed as a register with binary addresses ranging from 000 to 111.information is stored by the diodes in the register as 0s and 1s.
- The presence of a diode stores 1, and its absence stores 0. When a register is selected, the voltage of that line goes high, and the output lines, where diodes are connected, go high.

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.

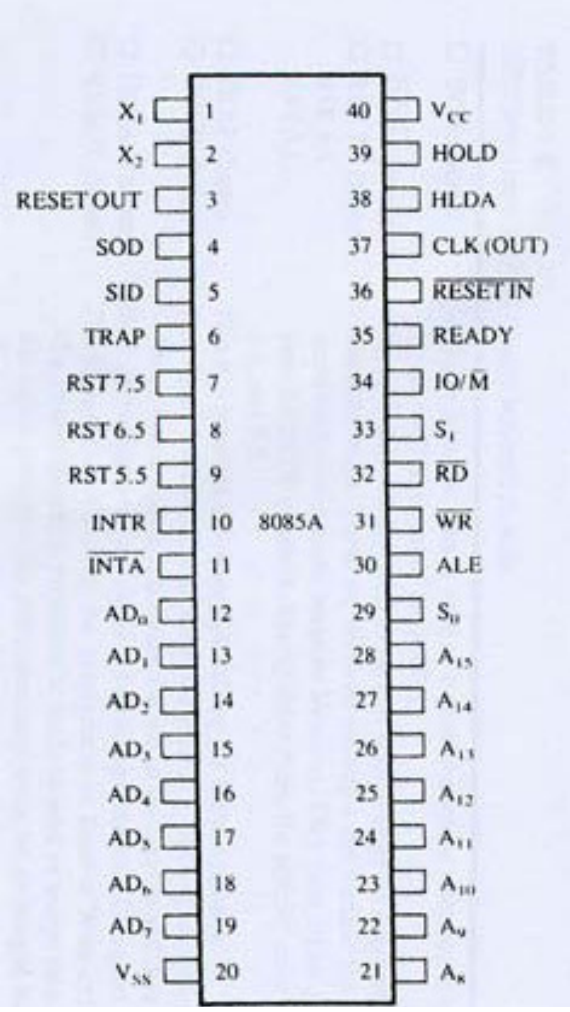
8085 microprocessor architecture and memory interfacing

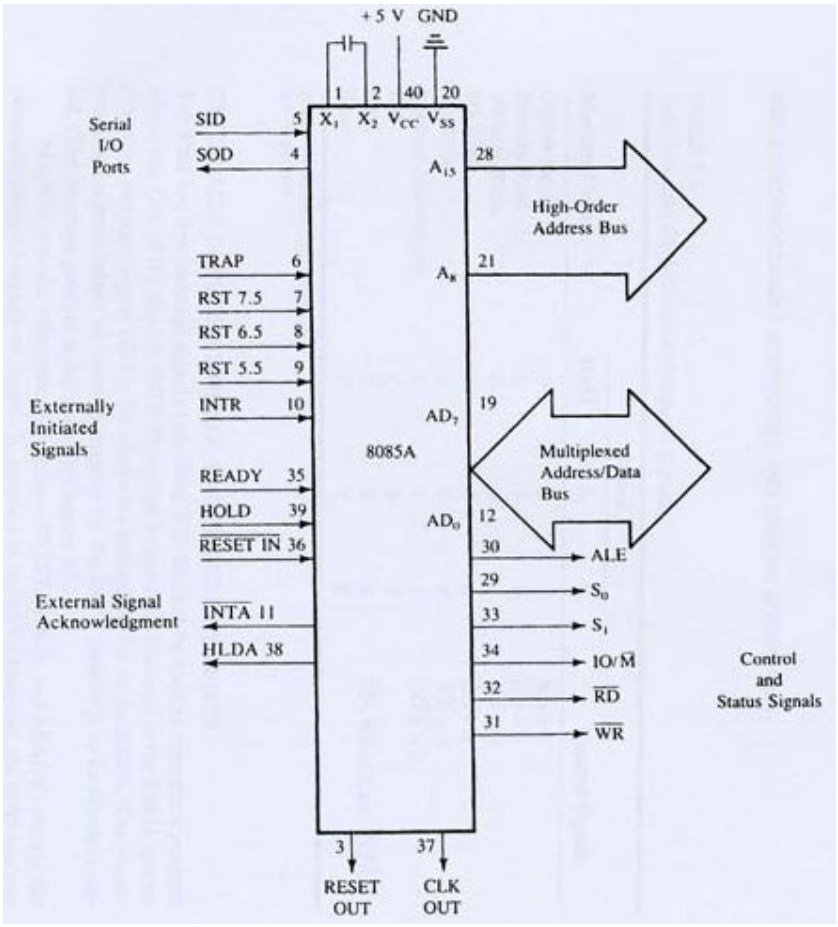
The 8085 MPU

- 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.

The 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
  - Serial I/O ports





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 Latch Enable

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

This signal indicates that the MPU is being reset. This signal can be used to reset other devices.

Serial IO ports

The 8085 have two signals to implement the serial transmission SID serial input data and SOD serial output data.

Microprocessor communication and bus timings

Let us examine the process of communication (reading and writing into memory) between the microprocessor and memory and the timings of these signals in relation to the system clock. The steps are as follows:

Step1:

The microprocessor places the 16-bit memory address from the program counter (PC) on the address bus. T, the high order address bus 20H is placed on the bus A15-A8, the low oreder address memory address 05H is placed on the bus AD7 – AD0, and the ALE signal goes high. Similarly, the status signal IO/M goes low, indicating that this is a memory-related operation.

Step2:

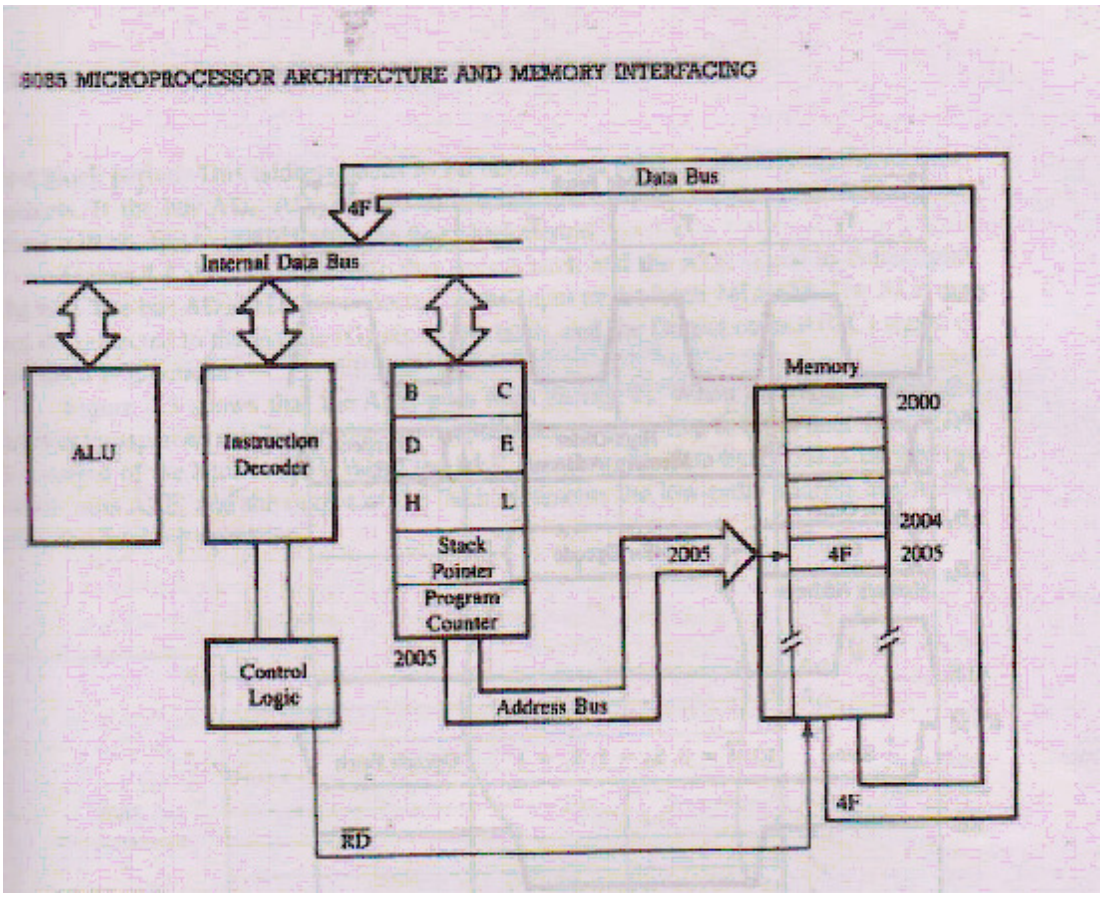
The control unit sends the control signal RD to enable the memory chip. The control signal RD is sent out during the clock period T2, thus enabling the memory chip.

Step 3:

The byte from the memory location is placed on the data bus. When the memory is enabled, the instruction byte (4FH) is placed on the bus AD7-AD0 and transferred to the microprocessor.

Step 4:

The byte is placed in the instruction decoder of the microprocessor, and the task is carried out according to the instruction. The machine code or the byte (4FH) is decoded by the instruction decoder, and the contents of the accumulator are copied into register C.



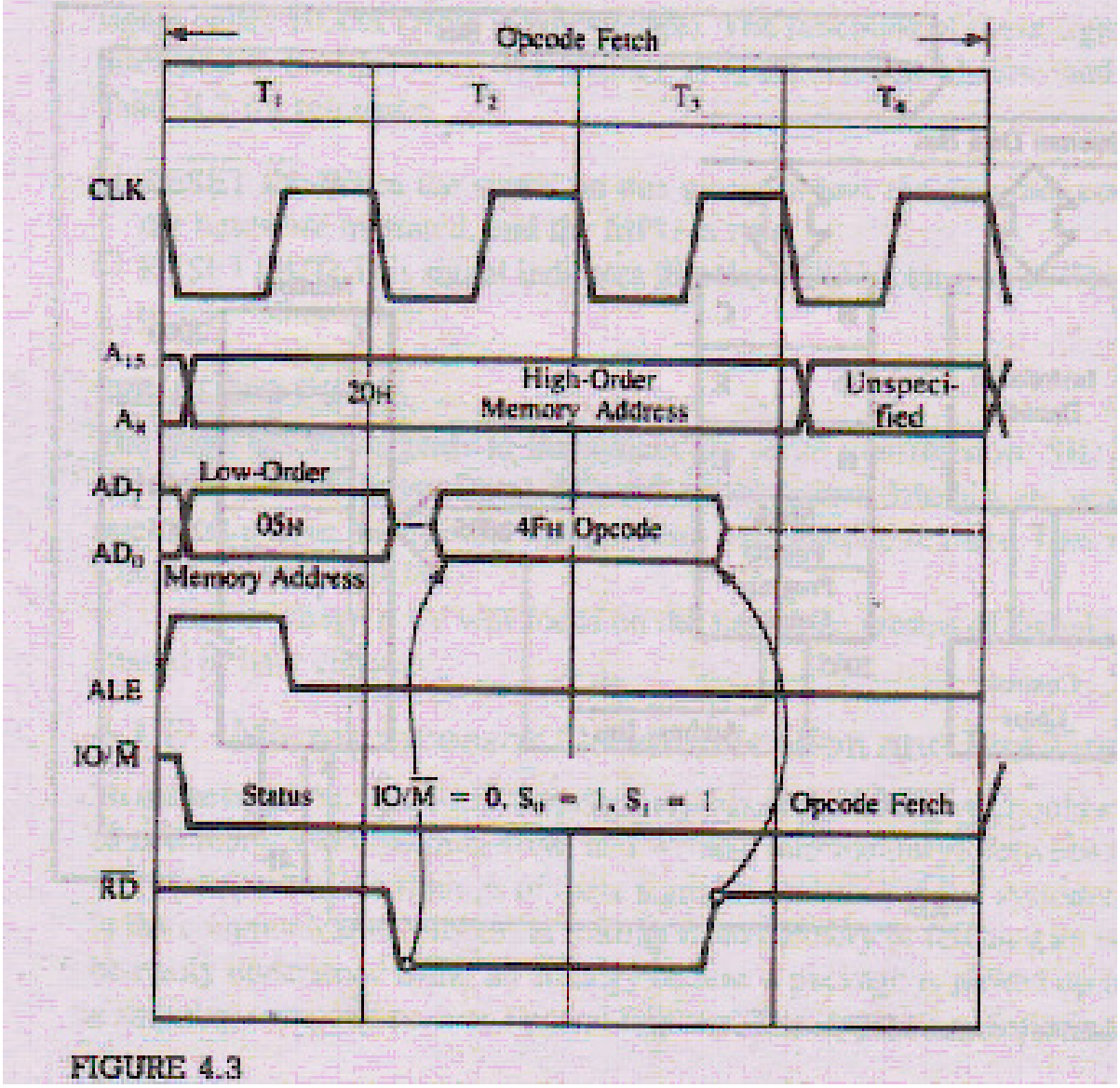
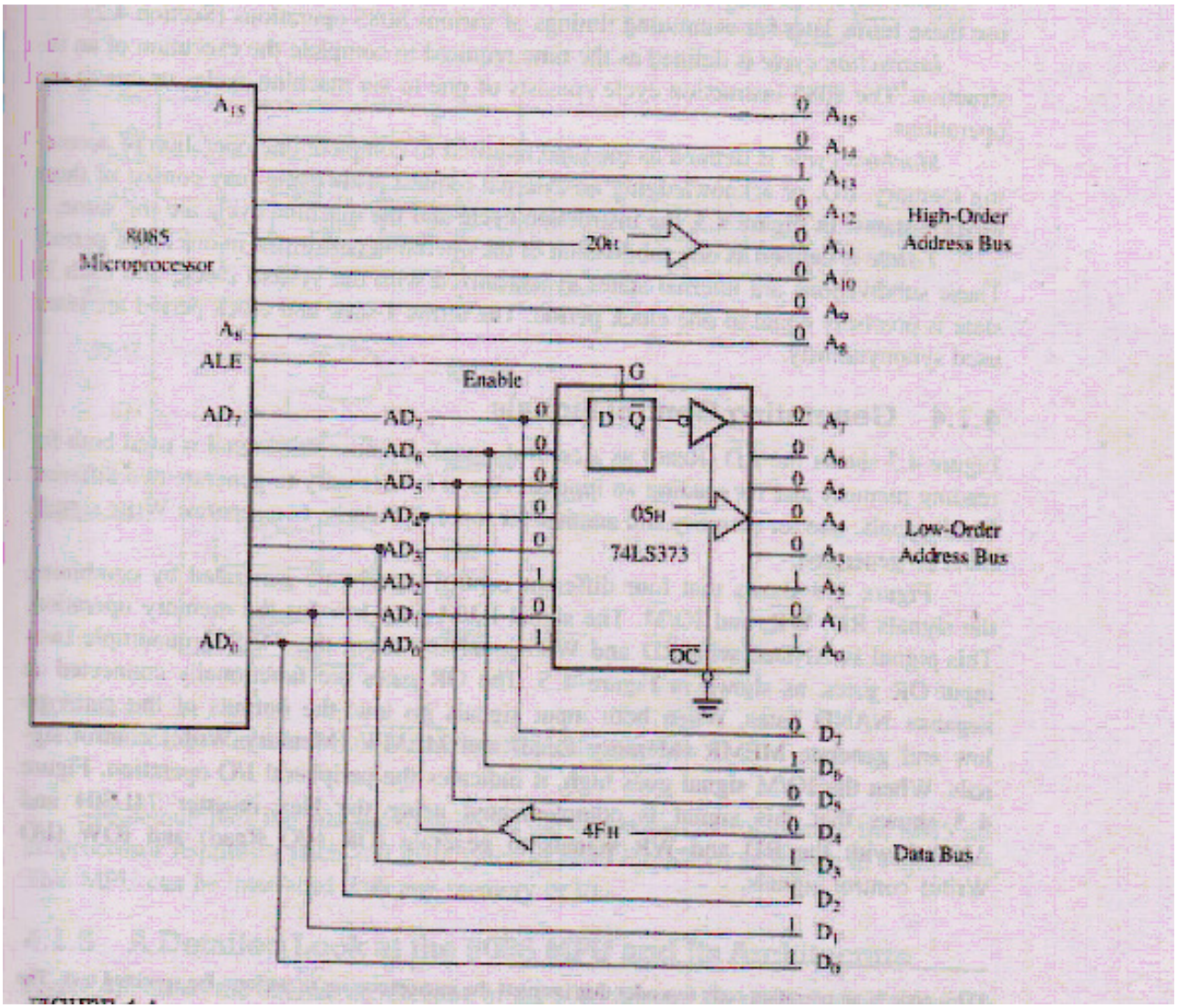


FIGURE 4.3

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.
- The diagram represents a latch and the ALE signal to demultiplex the bus. The bus AD7-AD0 is connected as the input to the latch 74LS373.
- The ALE signal is connected to the enable (G) pin of the latch and the output control (OC) signal of the latch is grounded.
- The ALE goes high during T1. When the ALE is high, the latch is transparent; this means the output changes according to input data.



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.

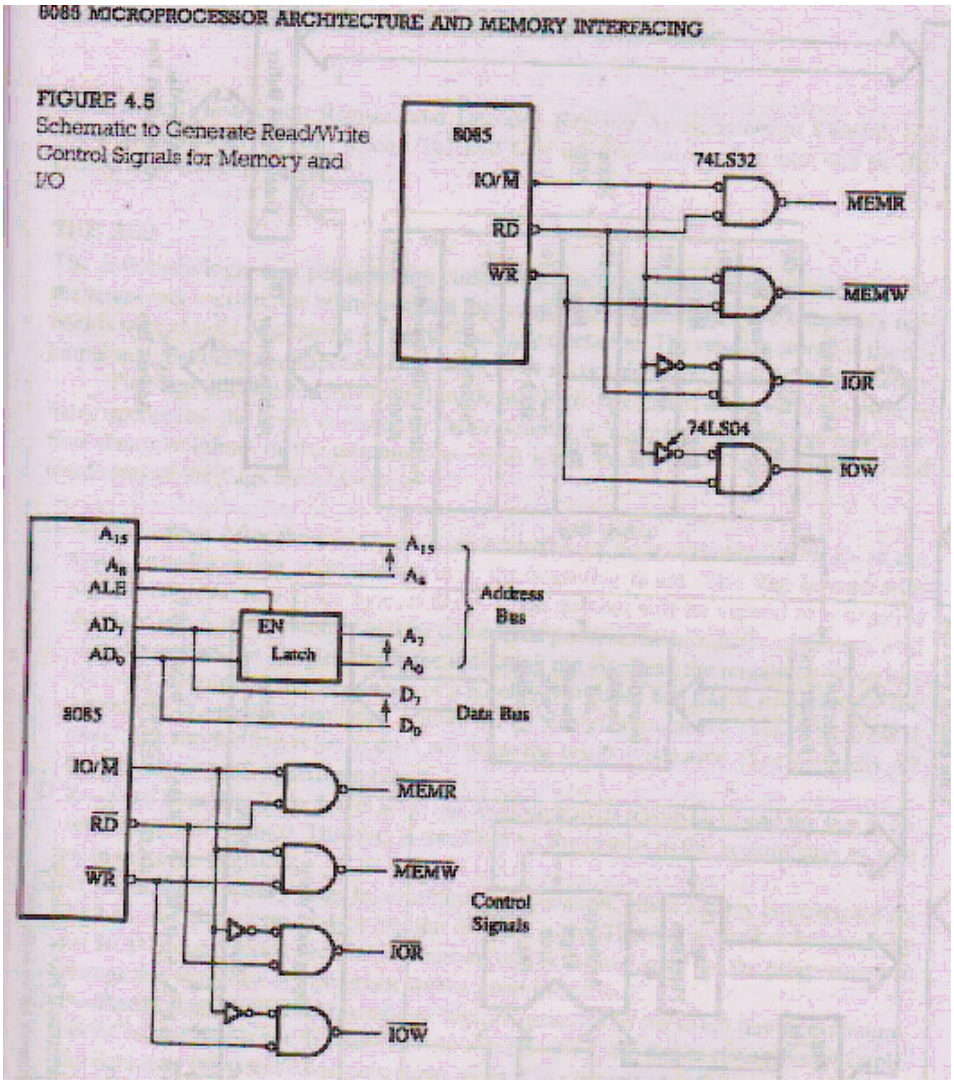
T-state

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.

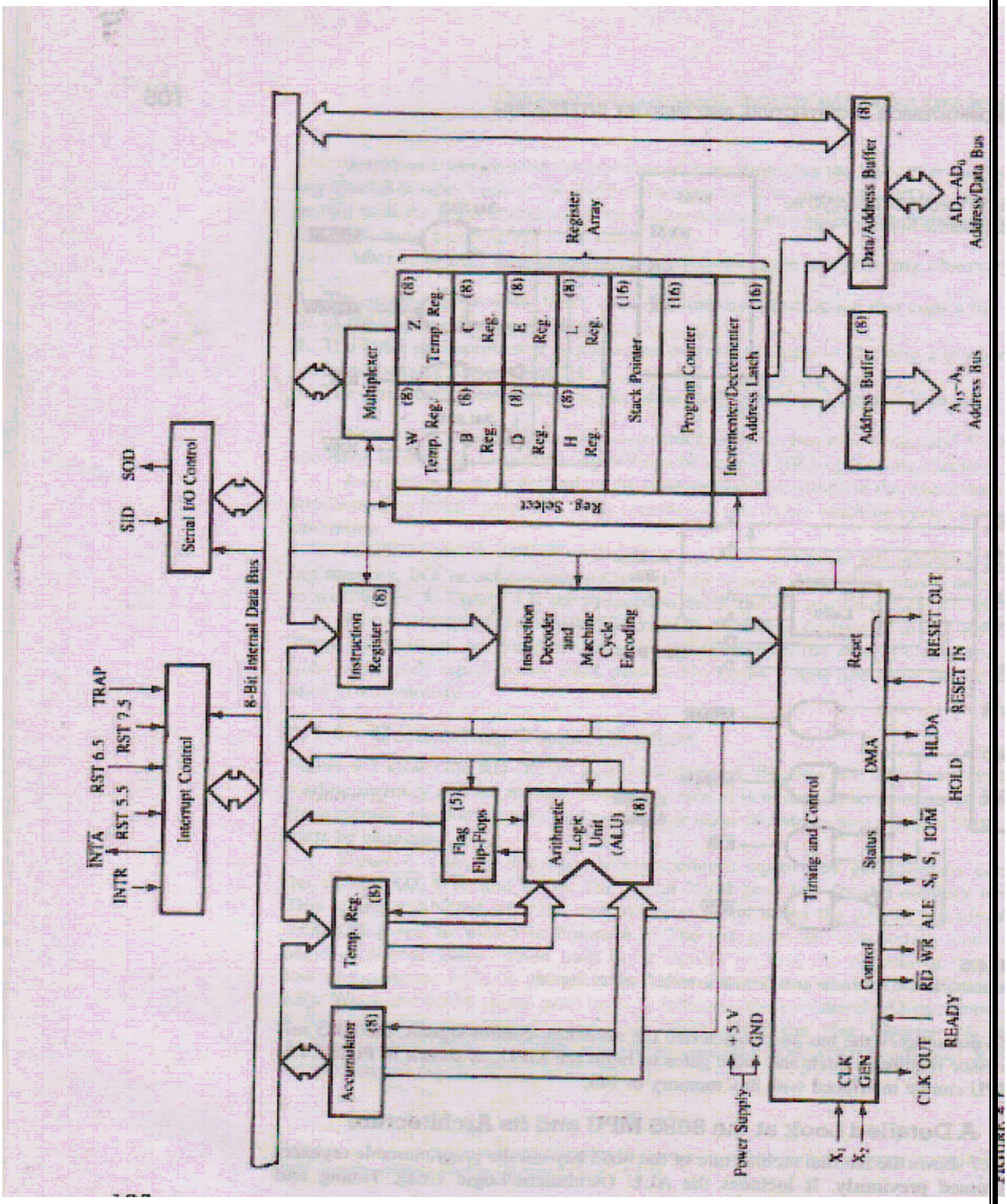
- Four control signals are generated by combining the signals RD, WR, and IO/M. the signal IO/M goes low for the memory operation.
- This signal is ANDed with RD and WR signals by using the 74LS32 quadruple two-input OR gates.
- The OR gates are functionally connected as negative NAND gates.
- When both 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 sync pulses 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.

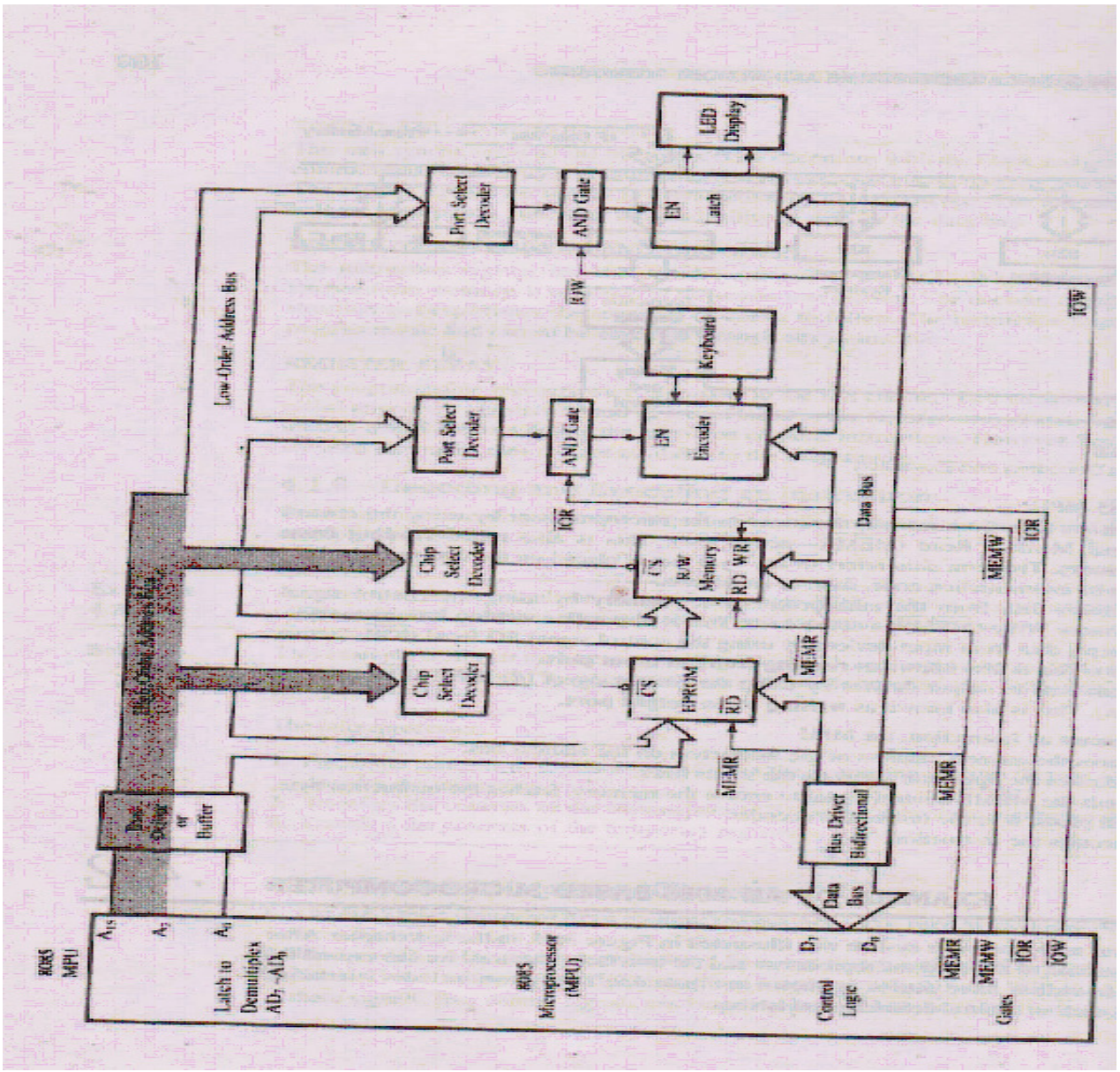
Decoding and executing an instruction

- 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.

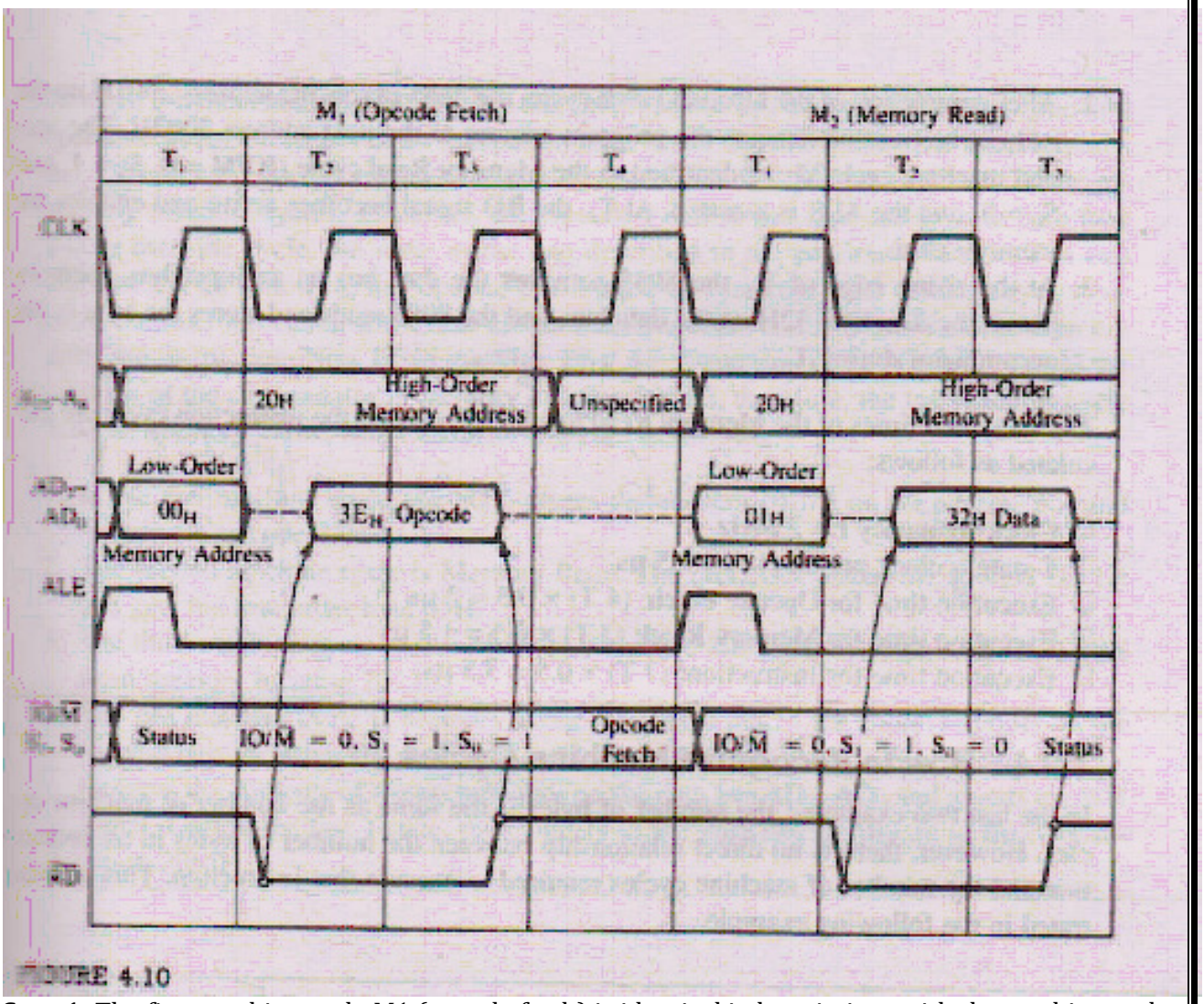


FIGURE 4.10

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.

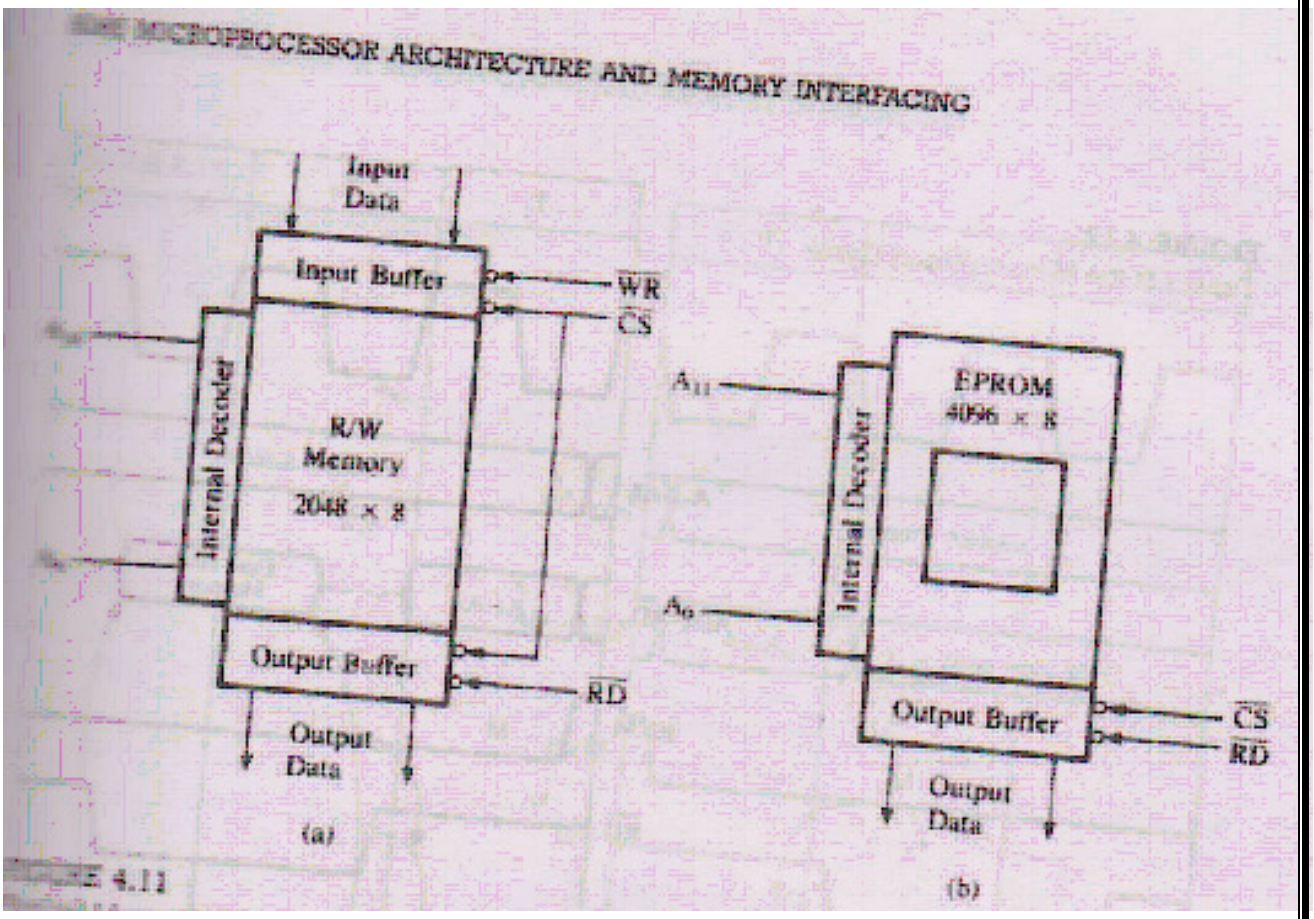
Memory interfacing

- 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.

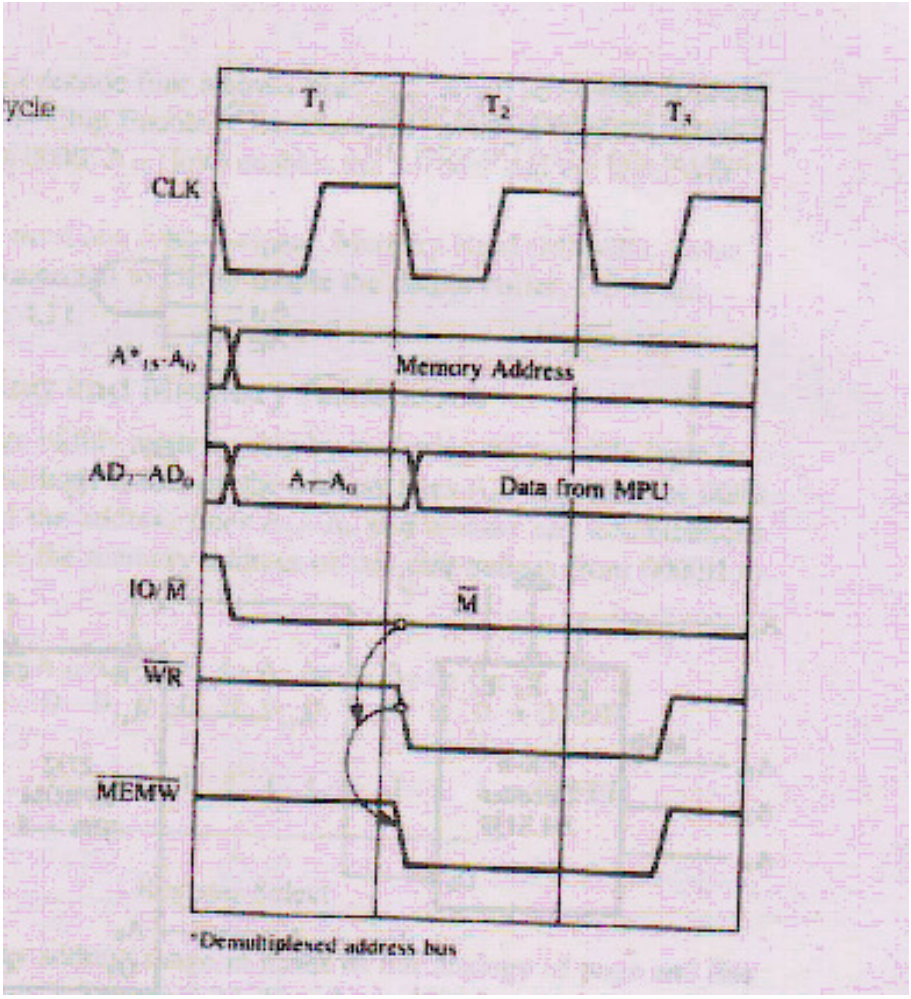


Basic concepts in memory interfacing

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

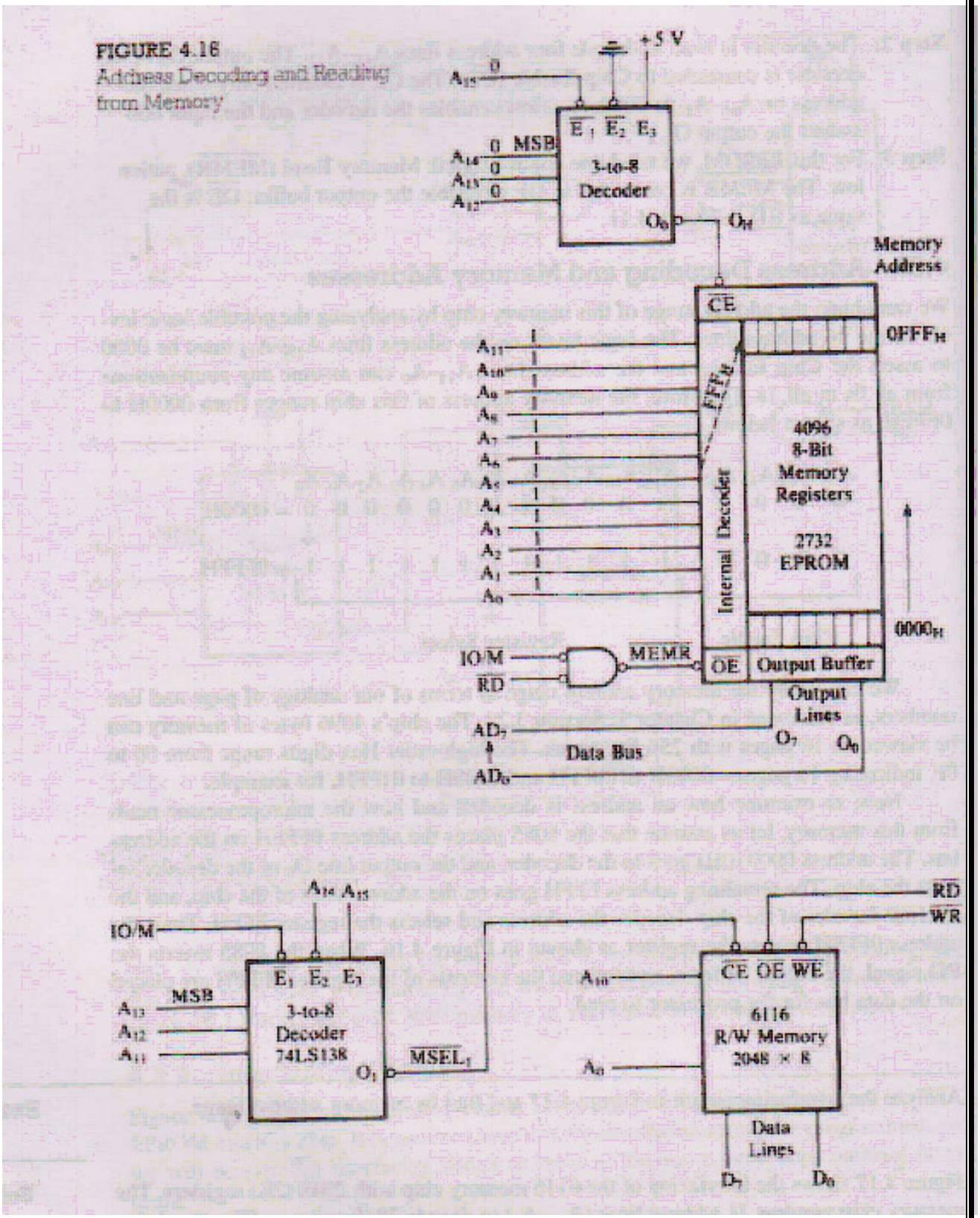
- ❖ The 8085 place a 16-bit address on the address bus, and with this address only one register should be selected. For the memory chip, only 11 address lines are required to identify 2048 registers.
- ❖ We connect the low order address lines A10-A0 of the 8085-address bus to the memory chip.
- ❖ The 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.



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 address 4096 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 combiunations 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.

2 Marks

1. What is MP?
2. What is MC? **(APR 2018)**
3. What is bus?
4. What is RAM?
5. What is ROM?
6. What is called Instruction?
7. What is Mnemonics?
8. What is Compiler?
9. What is Interpreter? **(APR 2018)**
10. What is Accumulator?
11. What is Source code?
12. What is Assembler?
13. What is Opcode?
14. What is Register? **(NOV 2017)**
15. What is Flag?
16. What is program counter?
17. What is Stack pointer?
18. What is data transfer?**(NOV 2017)**
19. What is BCD?
20. What is Instruction set?
21. What is Address bus? **(NOV 2018)**
22. What is Data bus?
23. What is Control bus?
24. What is Memory? **(NOV 2018)**
25. What is Buffer?
26. What is Serial I/O port?

5 Marks

1. Write short notes on Assembly language. **(NOV 2018) (NOV 2017)**
2. Discuss about 8085 programming Model. **(APR 2018)**
3. What is Instruction Set? **(NOV 2018)**
4. Write Short note on opcode and data format?
5. What is Memory and explain in detail. **(APR 2018)**
6. Write short notes on Memory classification**(NOV 2017)**

10 Marks

1. Write about Microprocessor architecture and its operation with neat diagram. **(NOV 2017)**
2. Explain in detail about 8085 Architecture. **(APR 2018)(NOV 2018)**
3. Write about Memory Interfacing.
4. Write about I/O devices peripheral map and Memory map I/O.

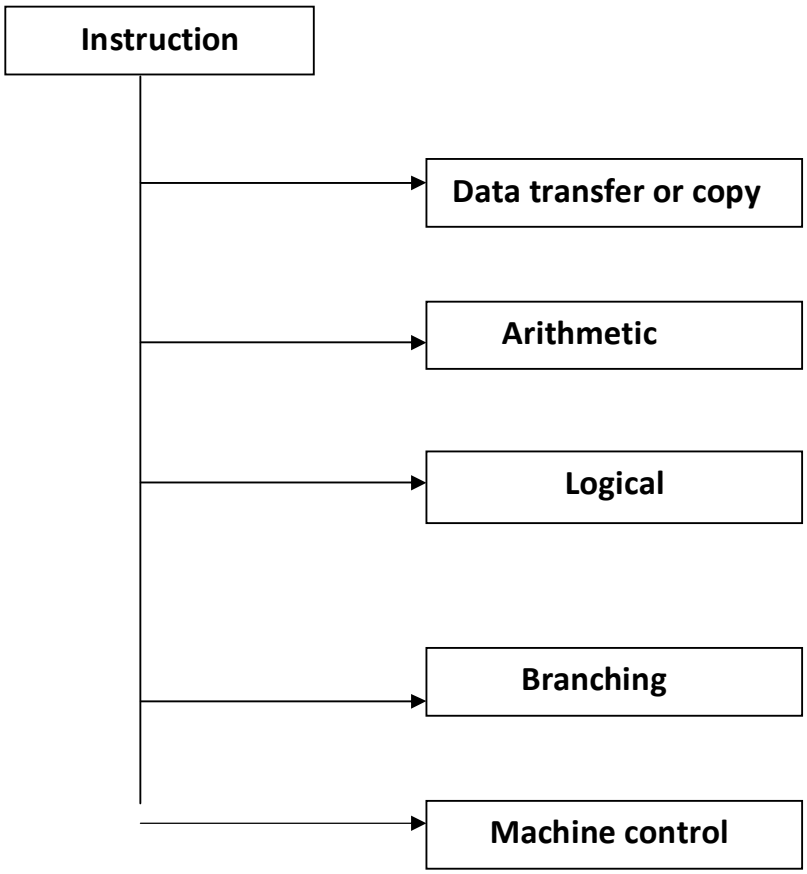
Unit V

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.



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-bit port address	**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-bit port address	**Input from Port** ✓ This is a 2-byte instruction ✓ 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.

Opcode	Operand	Description

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 the 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 accumulator.
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 and 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.

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**

		(if D7 = 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.

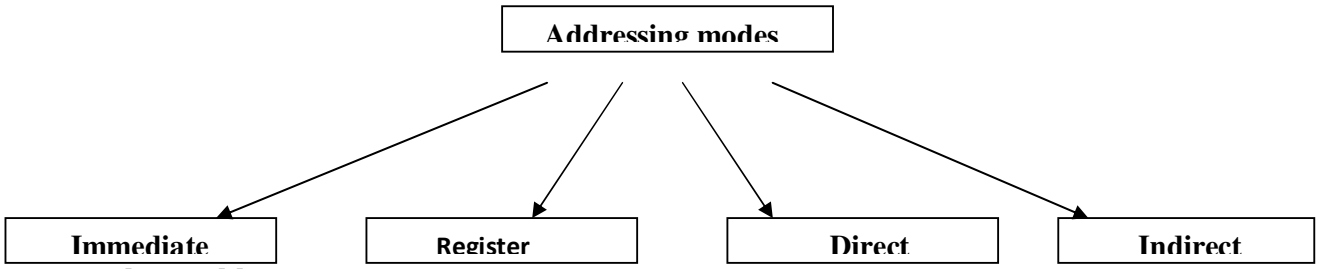
The

Opcode	Operand	Description
JMP	16-bit	***Jump*** Enables the programmer to set up continuous loops

Different types of addressing modes available in 8085 microprocessor.

ADDRESSING MODES

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



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.,

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.

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

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.

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.

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

$BCD_1 = 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1$

$BCD_2 = 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1$

- 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
	RRC	
	RRC	
	RRC	
	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, other wise divide by 10 repeatedly until the remainder is less than 10. the quotient is BCD_2.

Step 3:

The remainder from $step_2$ is BCD_1.

Example Quotient

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's complement
	INR A	Find 100's complement
	ADD B	Add minuend to 100's complement
	DAA	Adjust for BCD
	RET	Return

****All the Best****