2024

Digital Electronics and Computer Organisation

Dr. Babasaheb Ambedkar Open University

Digital Electronics and Computer Organisation

Course Writer:

Ms.Khyati Shah

Assistant Professor,

Som-Lalit Institute of Computer Applications

Content Reviewer and Editor:

Prof. (Dr.) Nilesh K. Modi

Professor & Director School of Computer Science,

Dr. BabasahebAmbedkar Open University

ISBN No:

Printed and published by: Dr. BabasahebAmbedkar Open University, Ahmedabad While all efforts have been made by editors to check accuracy of the content, the representation of facts, principles, descriptions and methods are that of the respective module writers. Views expressed in the publication are that of the authors, and do not necessarily reflect the views of Dr. BabasahebAmbedkar Open University. All products and services mentioned are owned by their respective copyrights holders, and mere presentation in the publication does not mean endorsement by Dr. BabasahebAmbedkar Open University. Every effort has been made to acknowledge and attribute all sources of information used in preparation of this learning material. Readers are requested to kindly notify missing attribution, if any

Dr. Babasaheb Ambedkar Open University

BCAR-203

Digital Electronics and Computer Organisation

BLOCK-1: DIGITAL COMPUTER AND DATA REPRESENTATION

BLOCK-2: LOGIC CIRCUITS, COMPONENTS OF DIGITAL COMPUTER AND MEMORY

BLOCK-3: CPU DESIGN AND INSTRUCTION SET

BLOCK-4: ORGANIZATION OF INPUT- OUTPUT AND MEMORY

Digital Electronics and Computer Organisation

Contents

UNIT7 MEMORY

Memory Unit, Internal Structure of Memory Unit, Random Access Memory, Read only Memory, Types of Read Only Memory, Summary

UNIT 8 BUS AND MEMORY TRANSFER

Concept of BUS ,Three-state buffer,Memory Transfer, Instruction Execution, Interrupt, Summary

---

BLOCK 3: CPU DESIGN AND INSTRUCTION SET

UNIT9 REGISTER TRANSFER AND MICROOPERATION

Register Transfer Language, Memory Read Operation, Memory Write Operation, Micro operations, Summary

UNIT 10 ARITHMETIC MICRO OPERATIONS

Introduction, Binary Adder, Binary Adder-Subtractor, Binary Incrementer, Summary

UNIT11 BASIC COMPUTER ORGANIZATION

Instruction Code, Basic Computer Instructions ,Addressing, Instruction Cycle, Summary

UNIT12 CENTRAL PROCESSING UNIT

Introduction, Stack Organization, Addressing Modes, Program Interrupt, Summary

---

BLOCK 4: ORGANIZATION OF INPUT- OUTPUT AND MEMORY

UNIT13 INPUT-OUTPUT ORGANIZATION

Peripheral devices, Input-Output Interface, DMA, Summary

UNIT14 MEMORY ORGANIZATION

Memory Hierarchy, Main Memory, Auxiliary Memory, Cache Memory,Virtual Memory, Summary

BLOCK 1: DIGITAL COMPUTER AND DATA REPRESENTATION

Block Introduction

In this block-1 of Digital Electronics,I have emphasis on What is digital computer?,Types of signals, About hardware and software, instruction execution and interrupt.Basically, I introduced digital computer and how the hardware and software coordinates to perform instruction execution.I also described about interrupts and types of interrupt to understand instruction execution more Precisely.These will add better understanding about entire instruction execution cycle.

I have also emphasis on various number system and conversion between one number system to another number system.I tried to add also complements and codes for understanding of number system more betterly.

Block Objective

The objective of the block is to explain what is digital computer and how request for instruction execution is given by use is handle.Detailed study of computer organization is requied to understand the working of computer.

By learning this block of digital electronics student will learn about computer organization, concept of BUS, instruction execution cycle. Reader of this block, will know about working of computer inernally.

Students also learn various number system and conversion between one number system to another number systemBy adding complements and codes for understanding of number system more betterly.

Unit 1: DIGITAL COMPUTER

Unit Structure

1.1. Digital Computer

1.2. Signals in Digital Computer

1.3. Hardware and Software

1.4. Computer Oraganisation

1.5. Von Neumann Architecture

1.6. Summary

1.1 Digital Computer

Digital

A computer that performs calculations and logical operations with quantities represented as digits, usually in the binary number system.

A computer that accepts and processes data that has been converted into binary numbers.

The first electronic digital computers, developed in late 1940s, were used primarily for numerical computations.

Bit

A bit is short for binary digit and is a single unit of information that can have a value of either 0 or 1.A bit is represented by a lowercase b

All instructions that the computer executes and the data that it processes is made up of a group of bits.

Information is represented in digital computers in groups of bits. By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other discrete symbols, such as decimal digits or letters of the alphabet.

Another unit of data representing eight bits is an octet. An octet always consists of eight bits, no matter the architecture.

Byte

A byte is a storage unit capable of representing a single character, such as a letter, number or symbol. Technically speaking, a byte is a sequence of binary bits in a serialized data stream in data transmission systems.Byte is abbreviated with an uppercase B.

How many bits in a byte?

In most computers, one byte is equated to eight smaller units called bits, although the size of a byte has always been dependent on hardware.

Types of bytes

While bytes are measured in bit multiples, computer storage is typically measured in byte multiples. Due to massive increases in storage capacity over time, there are now eight additional units of measurement following the byte. The eight different

types of bytes currently used in computer architectures range from kilobytes (1,024 bytes) to yottabytes (1,024 zettabytes).

Byte multiples can be measured using two systems: base-2 or base-10. A base-2, or binary, system is commonly expressed as a rounded off decimal number.

Note: One kibibyte contains 1,024 bytes. One mebibyte contains 1,024 x 1,024 or 1,048,576 bytes.

Program

A sequence of instructions for the computer is called a program.

The data that are manipulated by the program constitute the database.

Layers in Modern Computer

A modern computer is collection of different multiple layers.Shown in below diagram.

Hardware

The inner most layer is hardware surrounded by system software and application software.

BIOS (Basic Input-Output System)

The BIOS or basic input-output system of a computer provides the computer with some of its most basic capabilities.

A computer can use the BIOS to start and perform some very basic functions even without an operating system installed.

One of the most important things the BIOS do is load the operating system when the computer is started.

It also manages the various devices connected to the computer, and acts as a sort of ‘traffic cop’ to direct instructions and data to the appropriate layers and components of the computer’s architecture.

Operating System

The operating system controls the capabilities of the computer’s hardware. In particular, the operating system carries out these responsibilities:

1. **Processor management** – identifying and prioritizing (scheduling) task
2. **Memory management** – managing flow of data in and out of RAM, managing virtual memory as needed
3. **Device management** – providing the interface between devices and the CPU
4. **Storage management** – managing permanent storage on hard drives and other media
5. **Application Interface** – managing communications between applications and the computer
6. **User Interface** – managing interactions with the user allowing them to control the computer

1.2 Signals in Digital Computer

What is a Signal?

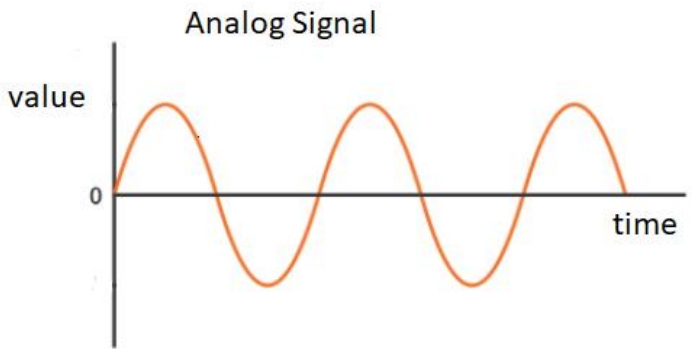
A signal is an electrical or electromagnetic current that is used to transfer data from one network to another.

There are two types of signals which are used in computer networking,

1. Analog signals
2. Digital signals

1. Analog Signals

These are the signals which can have an infinite number of different magnitude or values. They vary continuously with time. They are generated by signal generations hardware.



Examples of Analog Devices and Computers

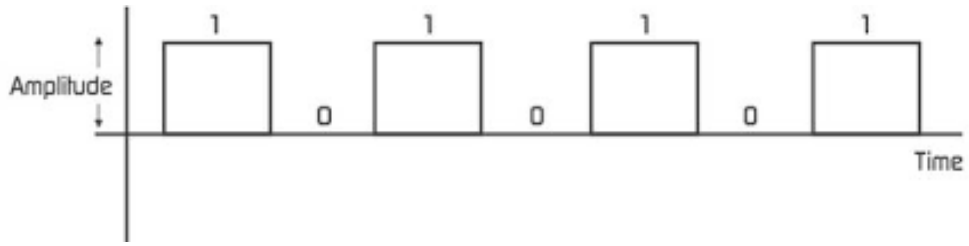
- Speedometer
- Tide Predictor
- Thermometer
- Analog clock
- Nomogram: a graphical calculating device

2. Digital signals

The signal, whose amplitude takes only limited values is called Digital signal.Digital signal are discrete, they contain only distinct values.

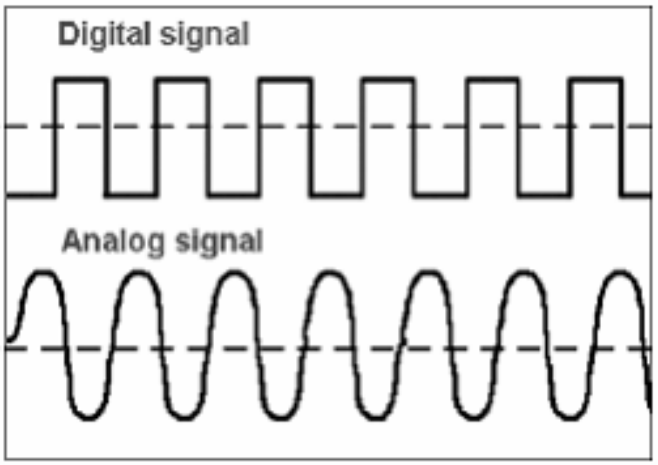
Digital signals carry binary data i.e. 0 or 1 in form of bits, it can only contain one value at a period of time. Digital signals are represented as square waves or clock signals.

Binary signals are examples of digital signals



The

following diagram shows the diagrammatic comparison of the digital and analog signals.



Analog and Digital Signal Difference

The following table shows the analog and digital signals difference in computer networking.

Analog Signal	Digital Signal
It uses a continuous range of values to represent information.	Digital signal uses discrete signals to represent information.
The bandwidth of the analog signal is low.	The bandwidth of a digital signal is high.
There is no fixed range in analog signals.	Digital signal has a finite range, i.e. 0 and 1
It is denoted by sine waves, as shown in the above diagram.	It is denoted by square waves as shown in the above diagram
Analog signals are used in temperature sensors, FM radio, photocells and light sensors.	Digital signals are used in computers, DVDs and CDs

1.3 Hardware and Software

Hardware

Hardware is the physical parts of computer.

Hardware refers to the physical elements of a computer. This is also sometime called the machinery or the equipment of the computer. Examples of hardware in a computer are the keyboard, the monitor, the mouse and the processing unit.

A computer's hardware is comprised of many different parts, but perhaps the most important of these is the motherboard. The motherboard is made up of even more parts that power and control the computer.

Software

Software is the electronic instructions that tell the computer to perform a task.

Software, commonly known as programs, consists of all the electronic instructions that tell the hardware how to perform a task.

These instructions come from a software developer in the form that will be accepted by the platform (operating system + CPU) that they are based on

Computer systems divide software systems into **two major classes**:

1) **System software:** Helps run computer hardware and computer system itself. System software includes operating systems, device drivers, diagnostic tools and more. System software is almost always pre-installed on your computer.

2) **Application software:** Allows users to accomplish one or more tasks. Includes word processing, web browsing and almost any other task for which you might install software. (Some application software is pre-installed on most computer systems.)

Difference between Application Software System Software

	Application Software	System Software
Definition	Application software is computer software designed to help the user to perform specific tasks.	System software is computer software designed to operate the computer hardware and to provide a platform for running application software.
Purpose	It is specific purpose software.	It is general-purpose software.
Environment	Application Software performs in a environment which created by System/Operating System	System Software Create his own environment to run itself and run other application.

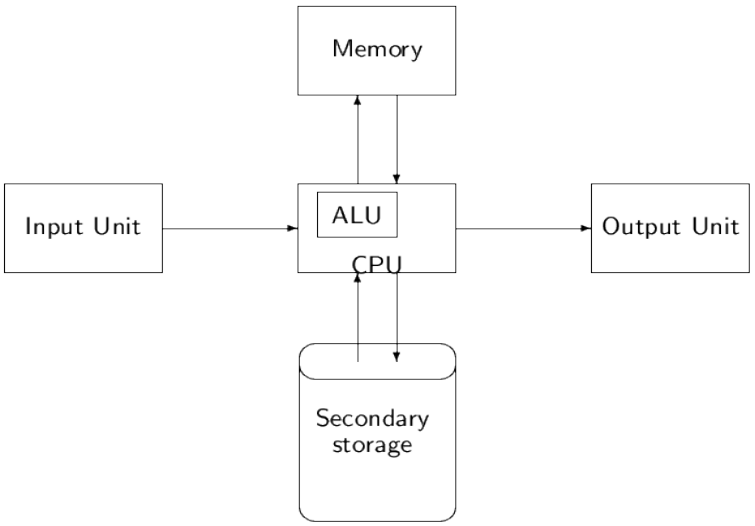
Execution Time	It executes as and when required.	It executes all the time in computer.
Essentiality	Application software is not essential for a computer to work.	System software is essential for a computer to work. Without System software (like operating systems), a computer is useless.
Number	The number of application software is much more than system software.	The number of system software is less than application software.
Example	Examples are: Windows, Dos, Unix, Linux, Norton Antivirus etc.	Examples are: MS Word, MS Excel, MS Power point etc.

1.4 Computer Organisation

The basic design of a computer includes how different parts of computer are organized and how various operations are performed between different parts of computer to do a specific task.

A computer has five functionally independent main parts:

- Input Unit
- Arithmetic and logic Unit
- Control unit
- Output Unit
- Memory Unit



Input Device

Computers accept coded information through input units, which read the data.

The most well-known input device is keyboard. Whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or the processor.

Memory Unit

The function of the memory unit is to store programs and data.

There are two classes of storage, called primary and secondary.

- **Primary storage** is a fast memory that operates at electronic speeds. Programs must be stored in the memory while they are being executed.

  Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called Random Access Memory (RAM).

- **Secondary storage** is used when large amounts of data and many programs have to be stored, particularly for information that is accessed infrequently. Ex:- magnetic disks and tapes and optical disks.

Difference between Primary Memory and Secondary Memory

	Primary Memory(main Memory)	**Secondary Memory(auxiliary Memory)**
Volatility	This is volatile.	This is Non-volatile.
Access Time	Access Time is higher Than Secondary memories.	Access Time is lower Than Primary memories.
Memory Status	This is a Temporary Memory.	This is a Permanent Memory.
Capacity	At present time, 512 MB to 8 GB RAMs are available.	At Present time 80 GB to 4 TB Hard Disc Drive are available.
Price	Higher than HDD/Secondary Memory	Lower Than primary Memory.
Connection	Connected via Slots.	Connected Via Cables.
Accessible	Primary memory is directly accessible to the CPU	Secondary memory is not directly accessible to the CPU
Speed	It is relatively fast memory.	It is slow in compare to main memory.

Arithmetic and Logic Unit

All the arithmetic or logic operation is initiated by bringing the required operands into the processor, where the operation is performed by the ALU.

Control Unit

The Control Unit is used to co-ordinate the operations of memory, ALU, input and output units.

The Control Unit determines the sequence in which computer programs and instructions are executed.

Output Unit

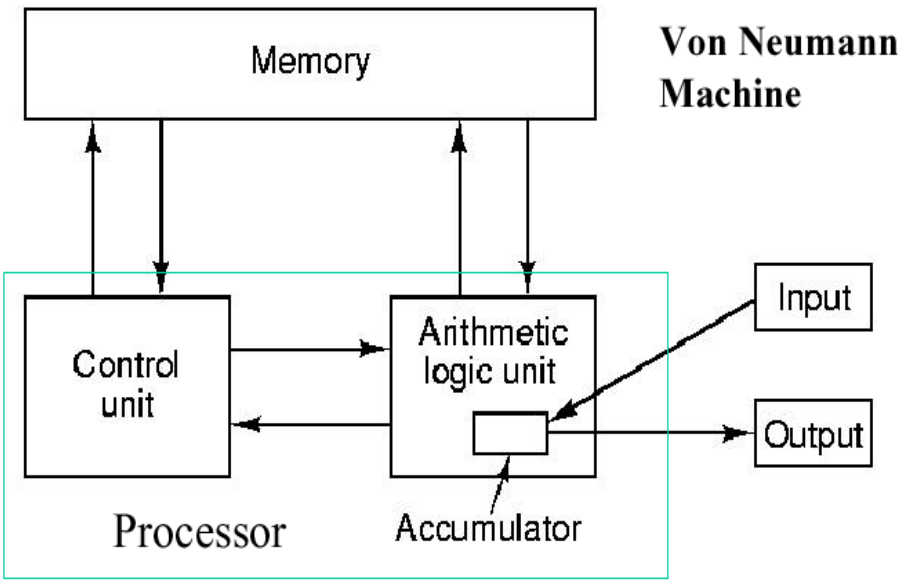
Its function is to send processed result to the outside world. Ex:- Printer.

1.5 Von Neumann Architecture

Von Neumann Architecture is comprised of the five classical components (input, output, processor, memory).

The processor is divided into an arithmetic logic unit (ALU) and control unit. Within the processor, the ALU data path mediates data transfer for computation.

A Von Neumann Architecture computer performs following sequence of steps and computer repeatedly performs the following cycle of events.



1. Fetch an instruction from memory.
2. Fetch any data required by the instruction from memory.
3. Execute the instruction (process the data).
4. Store results in memory.
5. Go back to step (1).

Von Neumann computers spend a lot of time moving data to and from the memory, and this slows the computer. This is called the von Neumann bottleneck.

1.9 Summary

In this chapter you have learned about:

- Digital Computer
- Signals in Digital Computer
- Hardware and Software
- Computer Oraganisation
- Von Neumann Architecture

Unit 2: Data Representation and Conversions

Unit Structure

2.1. Number System

2.2. Number System Conversion

2.3. Floating Point Numbers

2.4. Normalization of Floating Point Numbers

2.5. Binary Addition

2.6. Binary Subtraction

2.7. Summary

2.1 Number System

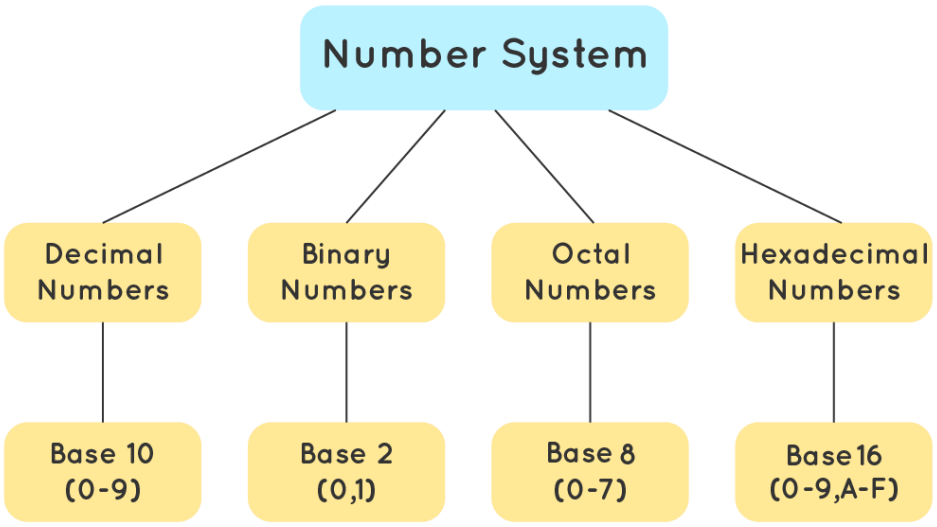
Definition

A number system is defined as the representation of numbers by using digits or other symbols in a consistent manner. The value of any digit in a number can be determined by a digit, its position in the number, and the base of the number system. The numbers are represented in a unique manner and allow us to operate arithmetic operations like addition, subtraction, and division.

Types of Number Systems

There are different types of number systems in which the four main types are:

- Decimal number system (Base - 10)
- Binary number system (Base - 2)
- Octal number system (Base - 8)
- Hexadecimal number system (Base - 16)



Decimal Number System

The decimal number system consists of 10 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 and is the most commonly used number system. We use the combination of these 10 digits to form all other numbers. The value of a digit in a number depends upon its position in the number.

The decimal number system is the system that we generally use to represent numbers in real life. If any number is represented without a base, it means that its

base is 10. For example: 723_{10}, 32_{10}, 4257_{10} are some examples of numbers in the decimal number system.

Binary Number System

The binary number system uses only two digits: 0 and 1. It means a 2 number system.The numbers in this system have a base of 2. Digits 0 and 1 are called bits and 8 bits together make a byte.

For example: 10001_2, 111101_2, 1010101_2 are some examples of numbers in the binary number system.

Representation of a Binary Number

MSB			Binary Digit					LSB
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Octal Number System

The octal number system uses eight digits: 0,1,2,3,4,5,6 and 7 with the base of 8. The advantage of this system is that it has lesser digits when compared to several other systems, hence, there would be fewer computational errors. Digits like 8 and 9 are not included in the octal number system.

For example: 35_8, 23_8, 141_8 are some examples of numbers in the octal number system.

Hexadecimal Number System

The word hexadecimal comes from Hexa meaning 6, and decimal meaning 10. So, in a hexadecimal number system, there are 16 digits. In the hex system, the numbers are first represented just like in the decimal system, i.e. from 0 to 9. Then, the numbers are represented using the alphabet from A to F.

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2.2 Number System Conversion

Numbers can be represented in any of the number system categories like binary, decimal, hex, etc. Also, any number which is represented in any of the number system types can be easily converted to other.

The conversion of one base number to another base number considering all the base numbers such as decimal, binary, octal and hexadecimal with the help of examples. Here, the following number system conversion methods are explained.

- Binary to Decimal Number System
- Decimal to Binary Number System
- Octal to Binary Number System
- Binary to Octal Number System
- Binary to Hexadecimal Number System
- Hexadecimal to Binary Number System

Number System Conversion Table

Binary Numbers	Octal Numbers	Decimal Numbers	Hexadecimal Numbers
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Number System Conversion Methods

Number system conversions deal with the operations to change the base of the numbers. For example, to change a decimal number with base 10 to binary number with base 2. We can also perform the arithmetic operations like addition, subtraction, multiplication on the number system.

The methods to convert the number of one base to the number of another base starting with the decimal number system. The representation of number system base conversion in general form for any base number is;

$(Number)_b = d_{n-1}\, d_{n-2} \text{——}. d_1\, d_0 \,.\, d_{-1}\, d_{-2} \text{—-}\, d_{-m}$

In the above expression, $d_{n-1}\, d_{n-2} \text{——}. d_1\, d_0$ represents the value of integer part and $d_{-1}\, d_{-2} \text{—-}\, d_{-m}$ represents the fractional part.

Also, d_{n-1} is the Most significant bit (MSB) and d_{-m} is the Least significant bit (LSB).

Decimal to Other Bases

Converting a decimal number to other base numbers is easy. We have to divide the decimal number by the converted value of the new base.

Decimal to Binary Number:

Suppose if we have to convert decimal to binary, then divide the decimal number by 2.

Example 1. Convert $(25)_{10}$ to binary number.

Operation	Output	Remainder
25 ÷ 2	12	1(MSB)
12 ÷ 2`	6	0
6 ÷ 2	3	0
3 ÷ 2	1	1
1 ÷ 2	0	1(LSB)

Therefore, from the above table, we can write,

$(25)_{10} = (11001)_2$

Decimal to Octal Number:

To convert decimal to octal number we have to divide the given original number by 8 such that base 10 changes to base 8. Let us understand with the help of an example.

Example 2: Convert 128_{10} to octal number**.**

Operation	Output	Remainder
128÷8	16	0(MSB)
16÷8	2	0
2÷8	0	2(LSB)

Therefore, the equivalent octal number = 200_8

Decimal to Hexadecimal:

Again in decimal to hex conversion, we have to divide the given decimal number by 16.

Example 3:Convert 128_{10} to hex.

Operation	Output	Remainder
128÷16	8	0(MSB)
8÷16	0	8(LSB)

Therefore, the equivalent hexadecimal number is 80_{16}

Here MSB stands for a Most significant bit and LSB stands for a least significant bit.

Other Base System to Decimal Conversion

Binary to Decimal

In this conversion, binary number to a decimal number, we use multiplication method, in such a way that, if a number with base n has to be converted into a number with base 10, then each digit of the given number is multiplied from MSB to LSB with reducing the power of the base. Let us understand this conversion with the help of an example.

Example 1. Convert $(1101)_2$ into a decimal number.

Solution: Given a binary number $(1101)_2$.

Now, multiplying each digit from MSB to LSB with reducing the power of the base number 2.

$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 8 + 4 + 0 + 1$

$= 13$

Therefore, $(1101)_2 = (13)_{10}$

Octal to Decimal:

To convert octal to decimal, we multiply the digits of octal number with decreasing power of the base number 8, starting from MSB to LSB and then add them all together.

Example 2: Convert 22_8 to decimal number.

Solution: Given, 22_8

$2 \times 8^1 + 2 \times 8^0$

$= 16 + 2$

$= 18$

Therefore, $22_8 = 18_{10}$

Hexadecimal to Decimal:

Example 3: Convert 121_{16} to decimal number.

Solution: $1 \times 16^2 + 2 \times 16^1 + 1 \times 16^0$

$= 16 \times 16 + 2 \times 16 + 1 \times 1$

$= 289$

Therefore, $121_{16} = 289_{10}$

Hexadecimal to Binary Shortcut Method

To convert hexadecimal numbers to binary and vice versa is easy, you just have to memorize the table given below.

Hexadecimal Number	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

You can easily solve the problems based on hexadecimal and binary conversions with the help of this table. Let us take an example.

Example: Convert $(89)_{16}$ into a binary number.

Solution: From the table, we can get the binary value of 8 and 9, hexadecimal base numbers.

8 = 1000 and 9 = 1001

Therefore, $(89)_{16} = (10001001)_2$

Octal to Binary Shortcut Method

To convert octal to binary number, we can simply use the table. Just like having a table for hexadecimal and its equivalent binary, in the same way, we have a table for octal and its equivalent binary number.

Octal Number	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Example: Convert $(214)_8$ into a binary number.

Solution: From the table, we know,

2 → 010

1 → 001

4 → 100

Therefore,$(214)_8 = (010001100)_2$

Practice Problems on Number System Conversion

1. Convert 146_{10} into a binary number system
2. Convert $1A7_{16}$ into the decimal number system
3. Convert $(110010)_2$ into octal number system
4. Convert $DA2_{16}$ into the binary number system
5. Convert 4652_8 into the binary number system

2.3 Floating Point Numbers

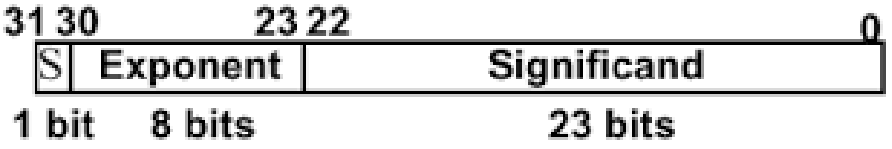
Floating point numbers are numbers that contain floating decimal points.

A floating-point number is one where the position of the decimal point can "float" rather than being in a fixed position within a number.

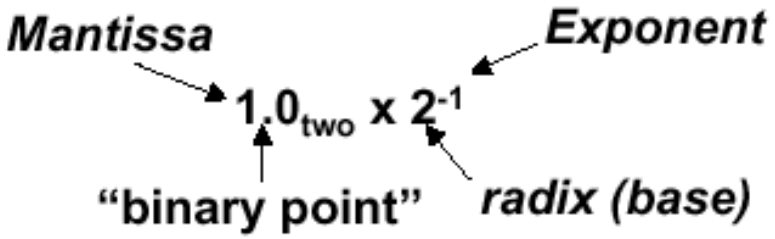
Examples of floating-point numbers are 1.23, 87.425, and 6.5 etc.
Different programming languages or systems may have different size limits or ways of defining floating-point numbers

Storing Floating Point

We split a Floating Point number into sign, exponent and mantissa as in the following diagram showing 23 bits for the mantissa ,8 bits for the exponent and 1 bit for sign bit:



Example



2.4 Normalization of Floating Point Numbers

Before a floating-point binary number can be stored correctly, its mantissa must be normalized.

The process is basically the same as when normalizing a floating-point decimal number. For example, decimal 1234.567 is normalized as 1.234567 x 103 by moving the decimal point so that only one digit appears before the decimal.

The exponent expresses the number of positions the decimal point was moved left (positive exponent) or moved right (negative exponent).

Similarly, the floating-point binary value 1101.101 is normalized as 1.101101 x 23 by moving the decimal point 3 positions to the left, and multiplying by 23. Here are some examples of normalizations:

Binary Value	Normalized As	Exponent
1101.101	1.101101	3
.00101	1.01	-3
1.0001	1.0001	0
10000011.0	1.0000011	7

2.5 Binary Addition

Binary addition is one of the binary operations. The binary addition operation works similarly to the base 10 decimal system, except that it is a base 2 system. The binary system consists of only two digits, 1 and 0. Most of the functionalities of the computer system use the binary number system.

Rules of Binary Addition

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

In fourth case, a binary addition is creating a sum of (1 + 1 = 10) i.e. 0 is written in the given column and a carry of 1 over to the next column.

Example − Addition

0011010 + 001100 = 00100110

1 1	carry
0 0 1 1 0 1 0	$= 26_{10}$
+ 0 0 0 1 1 0 0	$= 12_{10}$
0 1 0 0 1 1 0	$= 38_{10}$

2.6 Binary Subtraction

Binary subtraction is one of the four binary operations, where we perform the subtraction method for two binary numbers (comprising only two digits, 0 and 1)

There are four rules of binary subtraction.

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

when we subtract one from zero, we must borrow one from the next higher order digit to lower the digit by one, and the residual number left here is likewise 1.

Example − Subtraction

00100101 - 00010001 = 00010100

0		*borrows*
0 0 ~~1~~ ^{1}0 0 1 0 1	=	$37_{(\text{base } 10)}$
- 0 0 0 1 0 0 0 1	=	$17_{(\text{base } 10)}$
0 0 0 1 0 1 0 0	=	$20_{(\text{base } 10)}$

2.7 Summary

In this chapter you have learned about:

- Number System
- Number System Conversion
- Floating Point Numbers
- Normalization of Floating Point Numbers
- Binary Addition
- Binary Subtraction

UNIT 3: COMPLEMENTS

Unit Structure

3.1. Signed binary number

3.2. Complements of number

3.3. Subtraction using 1’s Complement

3.4. Subtraction using 2’s Complement

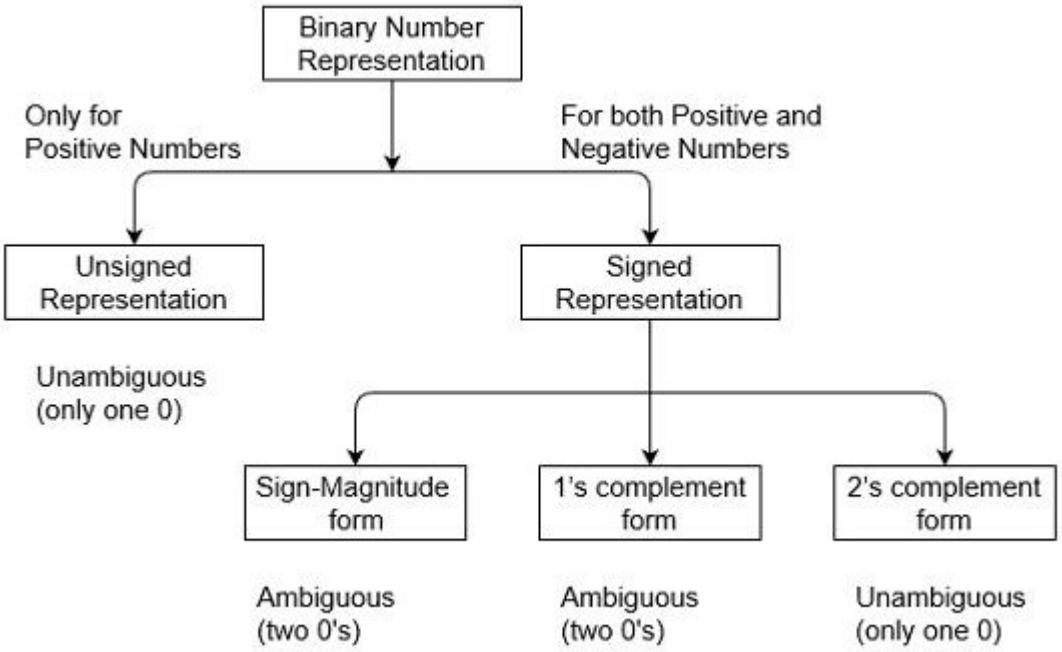
3.5. Summary

3.1 Signed binary number

Representation of Binary Numbers:

Binary numbers can be represented in signed and unsigned way. Unsigned binary numbers do not have sign bit.

whereas signed binary numbers uses signed bit as well or these can be distinguishable between positive and negative numbers. A signed binary is a specific data type of a signed variable.



There are three types of representations for signed binary numbers. Because of extra signed bit, binary number zero has two representation, either positive (0) or negative (1) These are:

1. Sign-Magnitude form
2. 1's complement form
3. 2's complement form

3.2 Complements of Numbers

In a number system, there is a fixed range of numbers and when you pick any number from the system, you have a complement for that number in the number system.

Types of Complements

There are two types of complements.

1. r-1's complement or Diminished Radix Complement.
2. r's complement or Radix Complement.

1. r-1's complement or Diminished Radix Complement

The (r-1)'s complement of a number in any number system with base r can be found out by subtracting every single digit of a number by r-1.

Suppose a number N is given

N = Number

r = base of the number

n = Number of digits in the number.

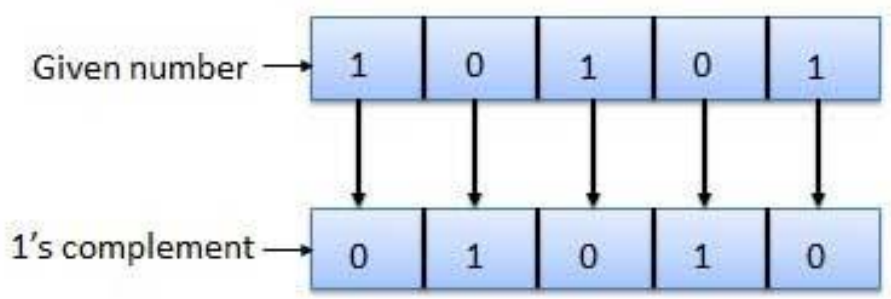
The r-1's complement of the number is given by the formula.

$$(r^n - 1) - N$$

Examples

I. 1's complement

The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.



II. 9's Complement

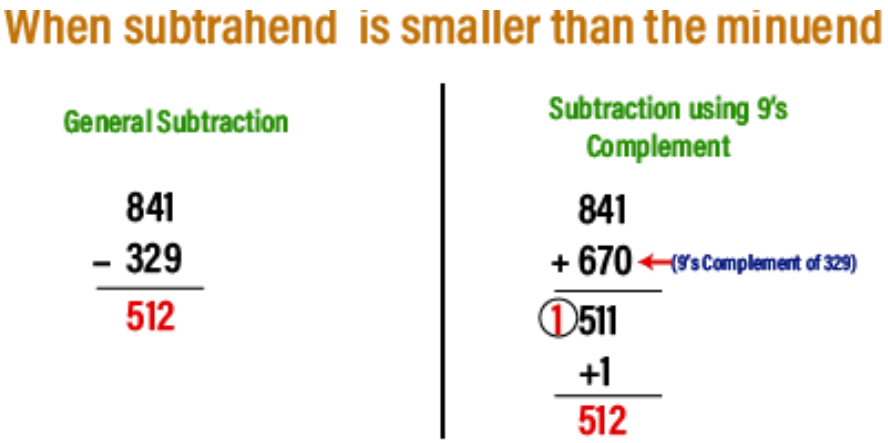
The 9's complement is used to find the subtraction of the decimal numbers.

The 9's complement of a number is calculated by subtracting each digit of the number by 9.

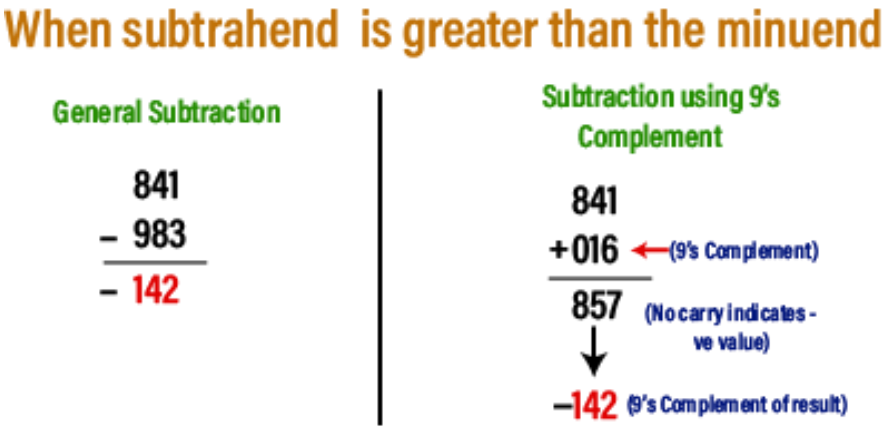
For example, suppose we have a number 1423, and we want to find the 9's complement of the number. For this, we subtract each digit of the number 1423 by 9. So, the 9's complement of the number 1423 is 9999-1423= 8576.

There **are two possible cases** when we subtract the numbers using 9's complement.

a) For subtracting the smaller number from the larger number using 9's complement, calculate 9's complement of smaller number or negative number, and by adding complement and first number, the result will come in the formation of carry. At last, we will add this carry to the result obtained previously.



b) For subtracting the larger number from the smaller number using 9's complement, calculate 9's complement of negative number, and by adding complement and first number, than no carry is generated than result is negative and make 9's complement of result and put negative sign.



2. r's complement or Radix Complement

The r's complement of a non-zero number in any number system with base r can be calculated by adding 1 to the LSB of its (r-1)'s complement.

To find radix complement of a number N where

N = Number

r. = base of the number

n = Number of digits in the number.

The r's complement of the number is given as

rn – N or r-1's complement + 1 => [(rn -1) – N] + 1

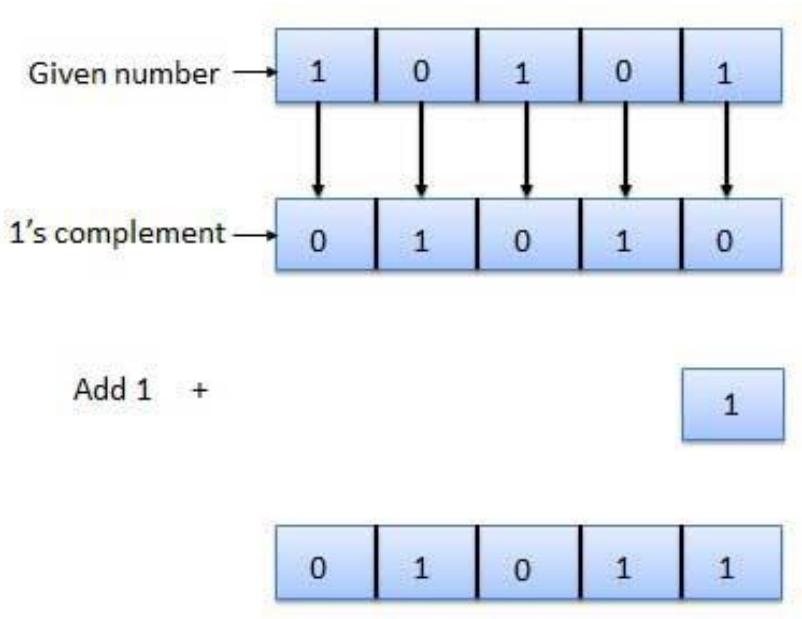
Examples

I. <u>2's complement</u>

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

2's complement = 1's complement + 1

Example of 2's Complement is as follows.



II. 10's Complement

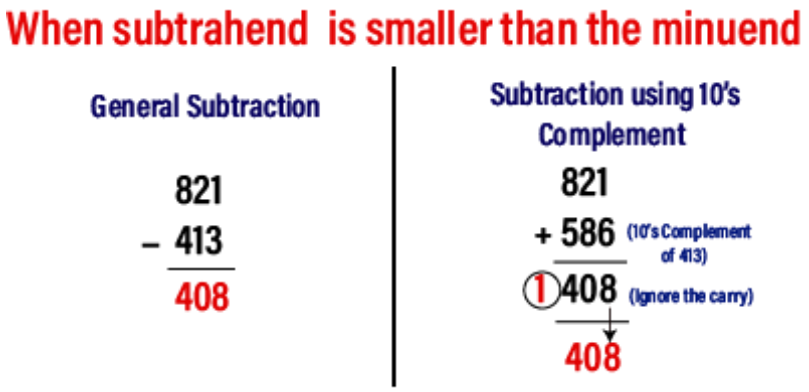
The 10's complement is used to find the subtraction of the decimal numbers.

The 10's complement of a number is calculated by subtracting each digit by 9 and then adding 1 to the result. Simply, by adding 1 to its 9's complement we can get its 10's complement value.

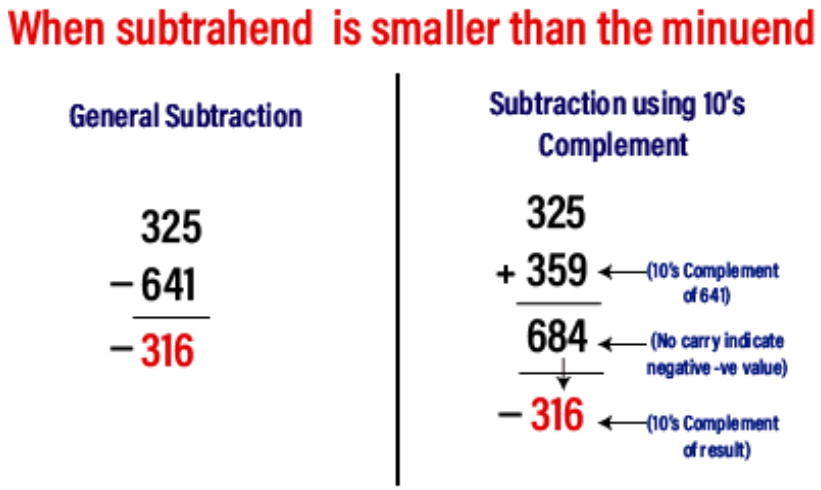
For example, suppose we have a number 1423, and we want to find the 10's complement of the number. For this, we find the 9's complement of the number 1423 that is 9999-1423= 8576, and now we will add 1 to the result. So the 10's complement of the number 1423 is 8576+1=8577.

There are **two possible cases** when we subtract the numbers using 10's complement.

a) For subtracting the smaller number from the larger number using 10's complement, calculate 10's complement of smaller number or negative number, and by adding complement and first number, the result will come in the formation of carry, then ignore carry.



b) For subtracting the larger number from the smaller number using 10's complement, first calculate 10's complement of negative number, and by adding complement and first number, than no carry is generated than result is negative and make 10's complement of result and put negative sign.



3.3 Subtraction using 1's Complement

These are the following steps to subtract two binary numbers using 1's complement

- In the first step, find the 1's complement of the subtrahend(negative number).
- Next, add the complement number with the minuend(first number).
- If got a carry, add the carry to its LSB.
- If carry is not generated than take 1's complement of the result which will be negative

Example 1: 10101 - 00111

We take 1's complement of subtrahend 00111, which comes out 11000. Now, sum them. So,

10101+11000 =1 01101.

In the above result, we get the carry bit 1, so add this to the LSB of a given result, i.e., 01101+1=01110, which is the answer.

Example 2: 10101 - 10111

We take 1's complement of subtrahend 10111, which comes out 01000. Now, add both of the numbers. So,

10101+01000 =11101.

In the above result, we didn't get the carry bit. So calculate the 1's complement of the result, i.e., 00010, which is the negative number and the final answer.

3.4 Subtraction using 2’s Complement

These are the following steps to subtract two binary numbers using 2's complement

- In the first step, find the 2's complement of the subtrahend.
- Add the complement number with the minuend.
- If we get the carry by adding both the numbers, then we discard this carry and the result is positive
- If carry is not generated than take 2's complement of the result which will be negative.

Example 1: 10101 - 00111

We take 2's complement of subtrahend 00111, which is 11001. Now, sum them. So,

10101+11001 =1 01110.

In the above result, we get the carry bit 1. So we discard this carry bit and remaining is the final result and a positive number.

Example 2: 10101 - 10111

We take 2's complement of subtrahend 10111, which comes out 01001. Now, we add both of the numbers. So,

10101+01001 =11110.

In the above result, we didn't get the carry bit. So calculate the 2's complement of the result, i.e., 00010. It is the negative number and the final answer.

3.9 Summary

In this chapter you have learned about:

- Signed binary number
- Complements of number
- Subtraction using 1’s Complement
- Subtraction using 2’s Complement

UNIT 4: CODES

Unit Structure

4.1 Binary codes

The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as **binary code.**

Binary codes can be classified into two types.

1. Weighted codes
2. Unweighted codes

If the code has positional weights, then it is said to be weighted code. Otherwise, it is an unweighted code.

1) 8 4 2 1 code

The weights of this code are 8, 4, 2 and 1. This code has all positive weights. So, it is a positively weighted code.

This code is also called as natural BCD (Binary Coded Decimal) code.

Decimal Digit	8421 Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

2) 2 4 2 1 code

The weights of this code are 2, 4, 2 and 1.This code has all positive weights. So, it is a positively weighted code.

It is an unnatural BCD code. Sum of weights of unnatural BCD codes is equal to 9.

It is a self-complementing code. Self-complementing codes provide the 9’s complement of a decimal number, just by interchanging 1’s and 0’s in its equivalent 2421 representation.

Decimal Digit	8421 Code	2421 Code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111

4.2 Excess-3 code

The Excess-3 code is nonweighted binary code. Following are steps to convert binary number to Excess-3 code.

Steps

- **Step 1** -- Convert decimal to BCD.
- **Step 2** -- Add $(3)_2$ to BCD number.

Example − convert $(1001)_{BCD}$ to Excess-3.

Step 1 − Convert to decimal

$9_{10=}(1001)_{BCD}$

Step 2 − Add 3 to decimal

$(1001)_2 + (0011)_2 = (1100)_2$

Step 3 − Convert to Excess-3

$(1100)_2 = (12)_{10}$

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Table showing all three codes together

Decimal Digit	8421 Code	2421 Code	Excess 3 Code
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

4.3 GRAY code

It is a non-weighted code .Gray code is a coded binary representation of a decimal digit which has a change in 1-bit position for consecutive digits.

Gray code is a coded binary representation of a decimal digit which has a change in 1-bit position for consecutive digits.

Gray Code sequences have to be converted to Binary or Binary Coded Decimal (BCD) if they are used in mathematical computations or for displays.

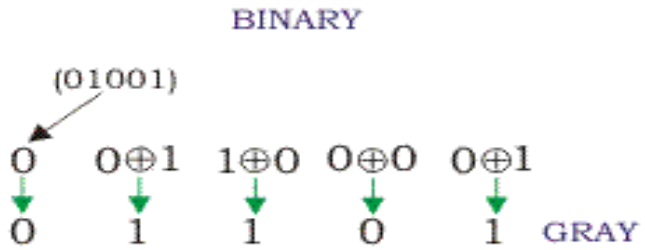
Binary to Gray Code Conversion

Steps given below elaborate on the idea on this type of conversion.

1. The M.S.B. of the gray code will be exactly equal to the first bit of the given binary number.

2. Now the second bit of the code will be **exclusive-or of the first and second bit** of the given binary number, i.e**if both the bits are same the result will be 0 and if they are different the result will be 1.**

3. The third bit of gray code will be equal to the exclusive-or of the second and third bit of the given binary number. Thus the Binary to gray code conversion goes on. One example given below can make your idea clear on this type of conversion.



Following table shows binary value and gray code value of decimal number.

Decimal numbers	Binary code	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

4.4 Alphanumeric codes

Alphanumeric codes are basically binary codes.

We can write alphanumeric data, including data, letters of the alphabet, numbers, mathematical symbols and punctuation marks by this code which can be easily understandable and can be processed by the computers.

The most common **alphanumeric codes** used these days are **ASCII code** and **EBCDIC code**.

1. ASCII Code

The full form of **ASCII code** is American Standard Code for Information Interchange. It is a seven bit code based on the English alphabet.

ASCII code has 128 characters

2. EBCDIC code

The EBCDIC stands for Extended Binary Coded Decimal Interchange Code.

EBCDIC is an IBM code for representing characters as numbers. Although it is widely used on large IBM computers, All the IBM computers and peripherals use this code. It is an 8 bit code and therefore can accommodate 256 characters

Char	EBCDIC
A	1100 0001
B	1100 0010
C	1100 0011
D	1100 0100
E	1100 0101
F	1100 0110
G	1100 0111
H	1100 1000
I	1100 1001

4.5 Summary

In this chapter you have learned about:

- Binary code
- Excess-3 code
- GRAY code
- Alphanumeric codes

BLOCK 2: LOGIC CIRCUITS, COMPONENTS OF DIGITAL COMPUTER AND MEMORY

Block Introduction

In this block-2 of digitasl electronics, I have tried to emphasis on: logic gates,components of digital computer and memory. Basically, I introduced concept of logic gates which helps in further circuit building. Along with this I have covered combinational circuits, flip-flops to understand how various operations are perform by various circuits in computer.

I hace also emphasis on registers,counters, internal structure and types of memory to understand how data is store and access from memory for any instruction.

Block Objective

The objective of the block is to explain logic gates which are building blocks for building any circuit. Students will able to learn about various circuits, flip-flops and meory.

By learning this block of digital electronics student will also learn about more details of how data is transfer between registers and memory . Reader of this block, will know about memory and memory transfer.

Unit 5: Digital Logic Circuit

Unit Structure

5.1 Logic Gates

Definition

Logic gates are the basic building blocks of any digital system.

It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on certain logic.

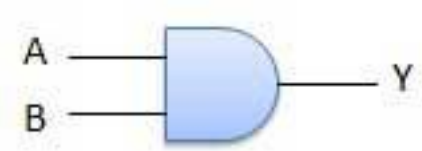
1. AND Gate

A circuit which performs an AND operation is shown in figure. It has n input (n >= 2) and one output.

The AND gate produces the AND logic function, that is, the output is 1 if input A and input B are both equal to 1; otherwise the output is 0. The algebraic symbol of the AND function is the same as the multiplication symbol

Y	=	A AND B AND C ..... N
Y	=	A.B.C ..... N
Y	=	ABC ..... N

Logic diagram



Truth Table

Inputs		Output
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

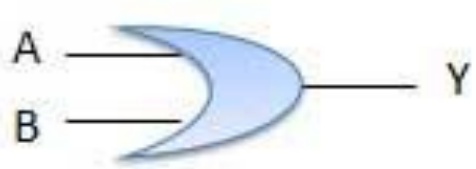
2. OR Gate

A circuit which performs an OR operation is shown in figure. It has n input (n >= 2) and one output.

The OR gate produces the inclusive-OR function; that is, the output is 1 if input A or input B or both inputs are 1; otherwise, the output is 0. The algebraic symbol of the OR function is +

Y = A OR B OR C ..... N
Y = A + B + C ..... N

Logic diagram



Truth Table

Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate

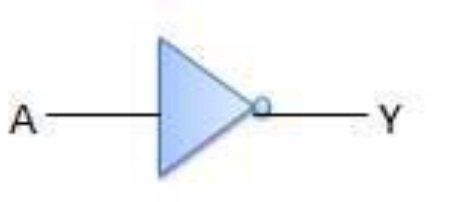
NOT gate is also known as Inverter. It has one input A and one output Y.

The inverter circuit inverts the logic sense of a binary signal. It produces the NOT, or complement, function.

The algebraic symbol used for the logic complement is a bar over the variable symbol.

Y = NOT A
Y = $\overline{A}$

Logic diagram



Truth Table

Inputs	Output
A	B
0	1
1	0

4. NAND Gate

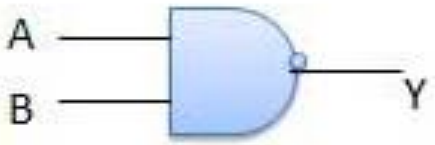
A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.

The NAND function is the complement of the AND function, as indicated by the graphic symbol, which consists of an AND graphic symbol followed by a small circle.

The designation NAND is derived from the abbreviation of NOT-AND.

Y = A NOT AND B NOT AND C ..... N
Y = A NAND B NAND C ..... N

Logic diagram



Truth Table

Inputs		Output
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

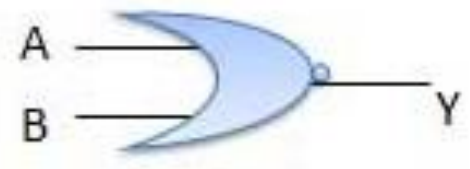
5. NOR Gate

A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.

The NOR gate is the complement of the OR gate and uses an OR graphic symbol followed by a small circle.

Y = A NOT OR B NOT OR C ..... N
Y = A NOR B NOR C ..... N

Logic diagram



Truth Table

Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

What are Universal Gates?

In Electronic physics there are AND, NOT and OR gates are the basic gates, we can create any logic gate or any Boolen expression by combining a mixture of these gates.

NOR gates and NAND gates have the particular property that any one of them can create any logical Boolen expression if appropriately designed so they are called a universal gate.

A universal gate is a logic gate which can implement any Boolean function without the need to use any other type of logic gate. The NOR gate and NAND gate are universal gates. This means that you can create any logical Boolean expression using only NOR gates or only NAND gates.
In practice, this is advantageous since NOR and NAND gates are economical and easier to fabricate than other logic gates.

6. XOR Gate

XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.

The exclusive-OR gate has a graphic symbol similar to the OR gate except for the additional curved line on the input side.

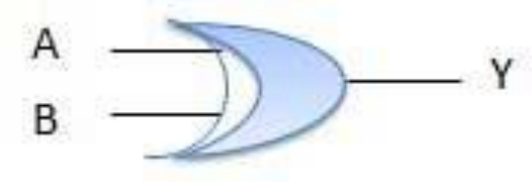
The output of the gate is 1 if any input is 1 but excludes the combination when both inputs are 1.

$$Y = A \text{ XOR } B \text{ XOR } C \ldots\ldots N$$

$$Y = A \oplus B \oplus C \ldots\ldots N$$

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

Logic diagram



Truth Table

Inputs		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

7. XNOR Gate

XNOR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-NOR gate is abbreviated as EX-NOR gate or sometime as X-NOR gate. It has n input (n >= 2) and one output.

The exclusive-NOR is the complement of the exclusive-OR, as indicated by the small circle in the graphic symbol.

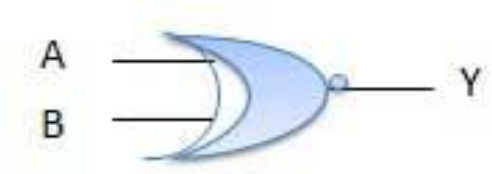
The output of this gate is 1 only if both the inputs are equal to 1 or both inputs are equal to 0.

$Y = A \text{ XOR } B \text{ XOR } C \ldots\ldots N$

$Y = A \odot B \odot C \ldots\ldots N$

$Y = \overline{A}\,\overline{B} + AB$

Logic diagram



Truth Table

Inputs		Output
A	B	$A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

5.2 Boolean Algebra

Boolean algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as Binary Algebra or logical Algebra.

Boolean algebra was invented by George Boole in 1854.

Rule in Boolean Algebra

Following are the important rules used in Boolean algebra.

- Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
- Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as B. Thus if B = 0 then B = 1 and B = 1 then B = 0.
- ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
- Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

Boolean Laws

There are six types of Boolean Laws.

Commutative law

Any binary operation which satisfies the following expression is referred to as commutative operation.

$$\text{(i) } A.B = B.A \qquad \text{(ii) } A + B = B + A$$

Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

Associative law

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

$$\text{(i) } (A.B).C = A.(B.C) \qquad \text{(ii) } (A + B) + C = A + (B + C)$$

Distributive law

Distributive law states the following condition.

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

AND law

These laws use the AND operation. Therefore they are called as AND laws.

(i) $A.0 = 0$ (ii) $A.1 = A$

(iii) $A.A = A$ (iv) $A.\overline{A} = 0$

OR law

These laws use the OR operation. Therefore they are called as OR laws.

(i) $A + 0 = A$ (ii) $A + 1 = 1$

(iii) $A + A = A$ (iv) $A + \overline{A} = 1$

Inversion law

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.

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

1. Boolean Algebra simplification rules

1. $A + \overline{A} = 1$	2. $A + A = A$
3. $A \cdot A = A$	4. $A \cdot \overline{A} = 0$
5. $A \cdot (B + C) = A \cdot B + A \cdot C$	6. $A + 0 = A$
7. $A + 1 = 1$	8. $A \cdot 1 = A$
9. $A \cdot 0 = 0$	10. $A \cdot B = B \cdot A$
11. $A + B = B + A$	12. $B \cdot (A + \overline{A}) = B$
13. $A + A \cdot B = A$	14. $A \cdot (A + B) = A$
15. $A + \overline{A} \cdot B = A + B$	16. $A \cdot (\overline{A} + B) = A \cdot B$
17. $\overline{A + B} = \overline{A} \cdot \overline{B}$	18. $\overline{A \cdot B} = \overline{A} + \overline{B}$

De Morgan's Theoram

De Morgan has suggested two theorems which are extremely useful in Boolean Algebra. The two theorems are discussed below.

Theorem 1

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

NAND = Bubbled OR

The left hand side (LHS) of this theorem represents a NAND gate with inputs A and B, whereas the right hand side (RHS) of the theorem represents an OR gate with inverted inputs.

This OR gate is called as Bubbled OR.

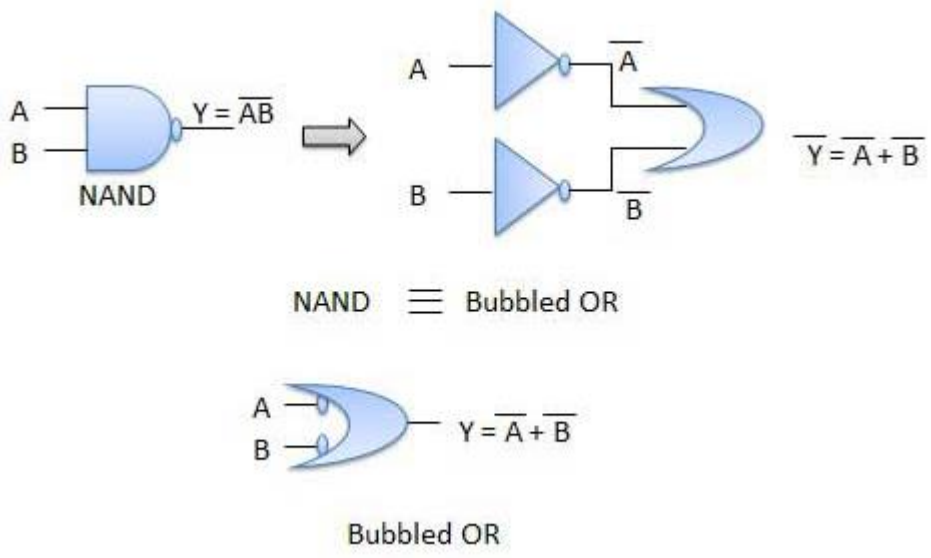


Table showing verification of the De Morgan's first theorem −

A	B	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

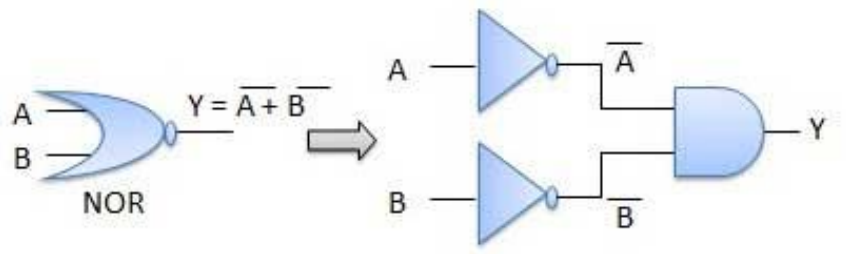
Theorem 2

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

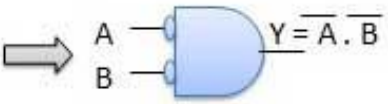
NOR = Bubbled AND

The LHS of this theorem represents a NOR gate with inputs A and B, whereas the RHS represents an AND gate with inverted inputs.

This AND gate is called as Bubbled AND.



NOR ☰ Bubbled AND



Bubbled AND

Table showing verification of the De Morgan's second theorem −

A	B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

5.3 Boolean Function

A Boolean Function is described by an algebraic expression called Boolean expression which consists of binary variables, the constants 0 and 1, and the logic operation symbols.

The function F is equal to 1 if x is 1 or if both y' and z are equal to 1, F is equal to 0 otherwise. Saying y'=1 is equivalent to saying that y=0.

So F is equal to 1 if x=1 or if yz=01.

Truth Table

The relationship between a function in a truth table and its binary variables can be represented in a truth table.

There are eight possible combination for assigning bits to the three variables x, y and z.

Logic Diagrams and Expressions

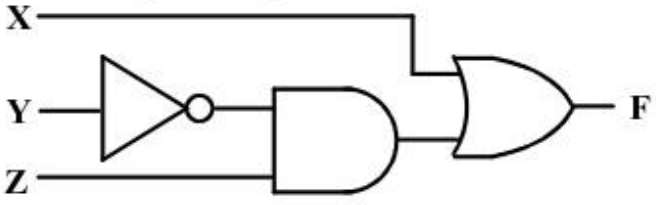
Truth Table

X Y Z	$F = X + \overline{Y} \cdot Z$
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

Logic Equation

$$F = X + \overline{Y}\,Z$$

Logic Diagram



- **Boolean equations, truth tables and logic diagrams describe the same function!**

5.4 Integrated Circuits

An IC is a collection of electronic components – resistors, transistors, capacitors, etc. – all stuffed into a tiny chip, and connected together.

Digital ICs and these are designed by using multiple numbers of digital logic gates, multiplexers, flip flops and other electronic components of circuits.

The various gates are interconnected inside the chip to form the required circuit. The chip is mounted in a ceramic or plastic container, and connections are welded by thin gold wires to external pins to form the integrated circuit.

The number of pins may range from 14 in a small IC package to 100 or more in a larger package.

Classification of ICs (Integrated Circuits)

Below is the classification of different types of ICs basis on their chip size.

- SSI: Small scale integration. 3 – 30 gates per chip.
- MSI: Medium scale integration. 30 – 300 gates per chip.
- LSI: Large scale integration. 300 – 3,000 gates per chip.
- VLSI: Very large scale integration. More than 3,000 gates per chip.

An IC can be further classified as being digital, analog or a combination of both. The most common example of a modern IC is the computer processor, which consists of billions of fabricated transistors, logic gates and other digital circuitry.

The circuit technology is referred as a digital logic family. Each logic family has its own basic electronic circuit.

There are many different logic families of integrated circuit following are popular among them.

1. TTL (transistor-to-transistor logic)

Transistor-transistor logic (TTL) is a class of integrated circuits which maintain logic states and achieve switching with the help of bipolar transistors. Many TTL logic gate s are typically fabricated onto a single integrated circuit (IC).

TTL integrated circuits (ICs) were widely used in applications such as computers, industrial controls, test equipment and instrumentation

A TTL device employs transistor s with multiple emitters in gates having more than one input. Transistor-transistor logic is one of the reasons that integrated circuits are so widely used, as they are less expensive, more reliable and faster

There are different sub-categories or families for transistor-transistor logic, such as:

- Standard transistor-transistor logic
- Fast transistor-transistor logic
- Schottky transistor-transistor logic
- High power transistor-transistor logic
- Low power transistor-transistor logic
- Advanced Schottky transistor-transistor logic

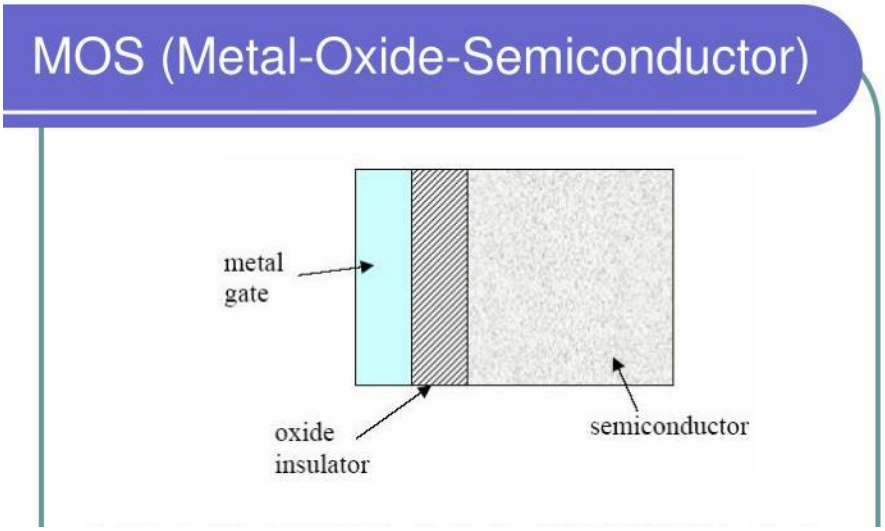
2. ECL(Emitter-coupled logic)

emitter-coupled logic (ECL) is a high-speed integrated circuit

ECL is used in systems such as supercomputers where high speed is essential.

3. MOS(Metal Oxide Semiconductor)

MOS is a method of creating transistors. MOS consists of three layers, a metal conductor, insulating silicon layer, and a semiconductor silicon layer.



4. CMOS

CMOS stands for "Complementary Metal-Oxide-Semiconductor." It's the name of a manufacturing process used to create processors, RAM, and digital logic circuits, and is also the name for chips created using that process.

CMOS is an on-board, battery powered semiconductor chip inside computers that stores information. This information ranges from the system time and date to system hardware settings for your computer.

When you make changes to your BIOS configuration, the settings are not stored on the BIOS chip itself. Instead, they are stored on a special memory chip, which is referred to as "the CMOS."

The picture shows an example of the most common CMOS coin cell battery (Panasonic CR 2032 3V) used to power the CMOS memory.

CMOS Battery

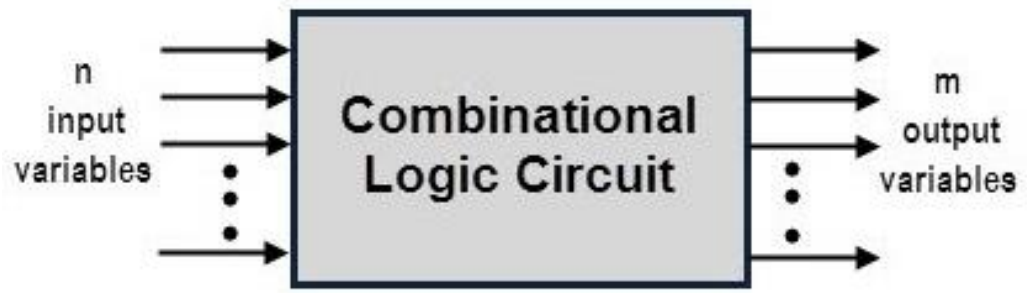
5.5 Combinational circuits

A combinational circuit is a connected arrangement of logic gates with set of inputs and outputs.

A combinational circuit comprises of input variables, logic gates and output variables. The logic gates accepts the inputs and depending on the type of functioning of the logic gate, output signals are generated from them.

Diagram below shows the combinational circuit having n inputs and m outputs.

The n number of inputs shows that there are 2^n possible combinations of bits at the input. Therefore, the output is expressed in terms m Boolean expressions.



Implementation steps

- Identify the input and output variables.
- Determine the relation between the input and output variables.
- Construct truth table according to input output specifications.
- Find out the Boolean expressions for outputs in terms of inputs.
- Minimize the Boolean expressions.
- The logic diagram is drawn.

1. Half Adder

With the help of half adder, we can design circuits that are capable of performing simple addition with the help of logic gates.

The half adder circuit has two inputs: A and B, which add two input digits and generate a carry and sum.

Block Diagram

Truth Table

INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1-bit adder can be easily implemented with the help of the XOR Gate for the output ‘SUM’ and an AND Gate for the ‘Carry’.

When we need to add, two 8-bit bytes together, we can be done with the help of a full-adder logic.

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

Carry = A•B

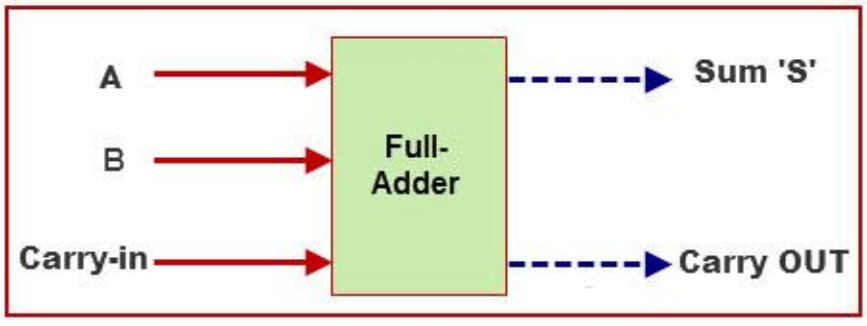
Circuit

2. Full Adder

A full-adder has three inputs and two outputs

The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S.

Block Diagram



Truth Table

INPUTS			OUTPUT	
A	B	C-IN	C-OUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

So, we can implement a full adder circuit with the help of two half adder circuits. At first, half adder will be used to add A and B to produce a partial Sum and a second half adder logic can be used to add C-IN to the Sum produced by the first half adder to get the final S output.

Full Adder Logic Circuit

5.6 Flip-Flops

A flip-flop is a type of circuit that contains two states, 0 or 1(on or off and are often used to store state information. By sending a signal to the flip-flop, the state can be changed.

Flip-flops are used in a number of electronics, including computers and communications equipment. Flip-flop circuits are interconnected to form the logic gates for the digital integrated circuits (IC s) used in memory chips and microprocessors.

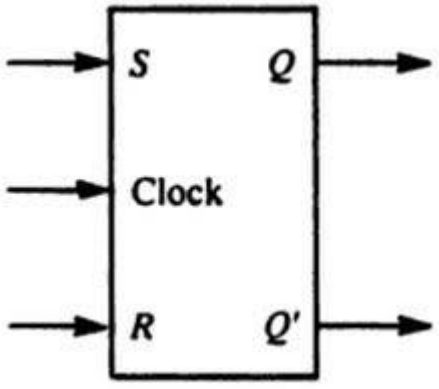
There are several different kinds of flip-flop circuits, with designators such as T (toggle), S-R (set/reset), J-K and D (data).

A flip-flop typically includes zero, one, or two input signals as well as a clock signal and an output signal. Some flip-flops also include a clear input signal to reset the current output.

SR Flip Flop

SR flip-flop is one of the most common types of flip-flop. SR stands for Set Reset.

Diagram shows the graphic symbol of SR flip-flop. The flip-flop has three inputs S (set), R, (reset) and C (clock)



S	R	Q
0	0	No change
0	1	0
1	0	1
1	1	Indeterminate

Operation of SR flipflop

If there is no signal at the clock input C, the output of the circuit cannot change the values of S and R.

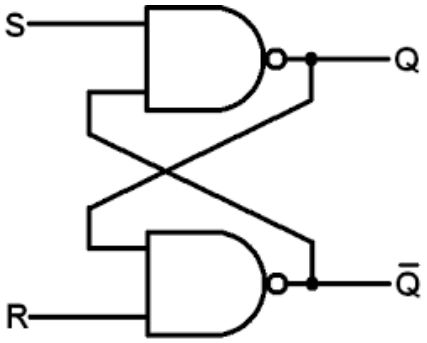
If S=1 and R=0 when C changes from 0 to 1, output Q is set to 1.

If S=0 and R=1 when C changes from 0 to 1, output Q is cleared to 0.

If both S and R are 0 during the clock transition, the output does not change.

When both S and R are equal to 1, the output is unpredictable and may go to either 0 or 1.

Logic Diagram

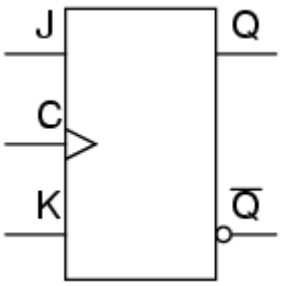


JK flip Flop

The JK Flip Flop is the most widely used flip flop. It is considered to be a universal flip-flop circuit. The sequential operation of the JK Flip Flop is same as for the RS flip-flop with the same SET and RESET input.

The JK Flip Flop name has been kept on the inventor name of the circuit known as Jack Kilby.

The intermediate condition of SR flip-flop is defined in JK.



Truth Table

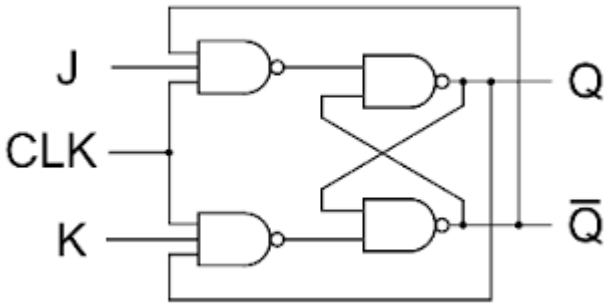
J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

Operation of JK flipflop

The J input is equivalent to the S (set) input of SR flip-flop and K input is equivalent to R (clear) input.

Instead of indeterminate condition, the JK flip-flop has a complement condition (toggle).

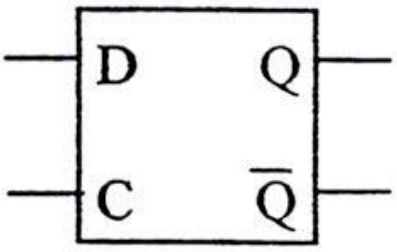
Logic Diagram



D Flip-flop

The D-type flip-flop is constructed from a gated SR flip-flop with an inverter added between the S and the R inputs to allow for a single D (data) input.

The D Flip Flop is by far the most important of the clocked flip-flops as it ensures that ensures that inputs S and R are never equal to one at the same time.



Input D	Circuit Action $Q_{(t+1)}$
0	0
1	1

Simply, for positive transition on clock signal,

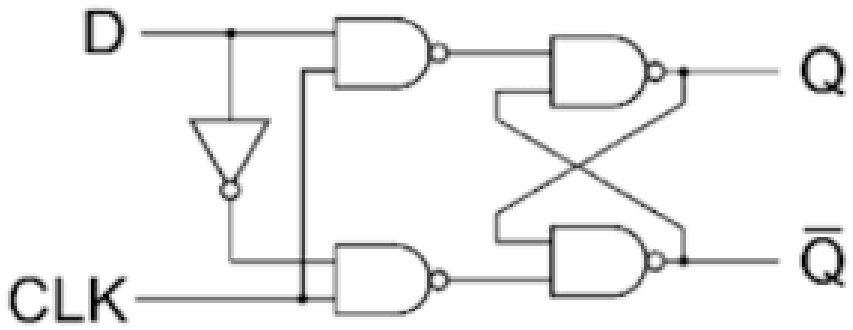
If D = 0 => Q = 0 so flip flop is reset.

If D = 1 => Q = 1 so flip flop is set.

Operation of D flip-flop

- If the clock signal is high (rising edge to be more precise) and if D input is high, then the output is also high and if D input is low, then the output will become low.

Logic Diagram



5.7 Summary

In this chapter you have learned about:

- Logic Gates
- Boolean Algebra
- Boolean Function
- Integrated Circuit
- Combinational Circuit
- Flip-flops

Unit 6: Components Of Digital Computer

Unit Structure

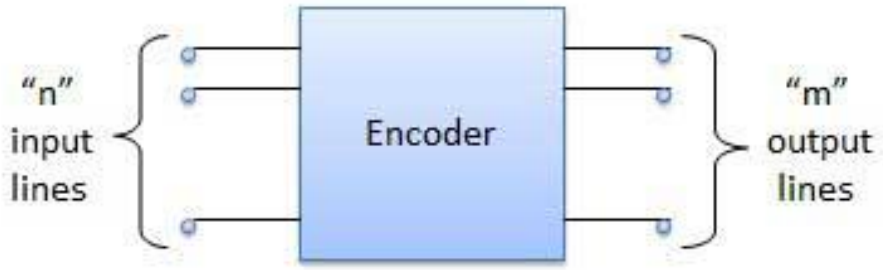
6.1 Encoder

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder.

An encoder has n number of input lines and m number of output lines.

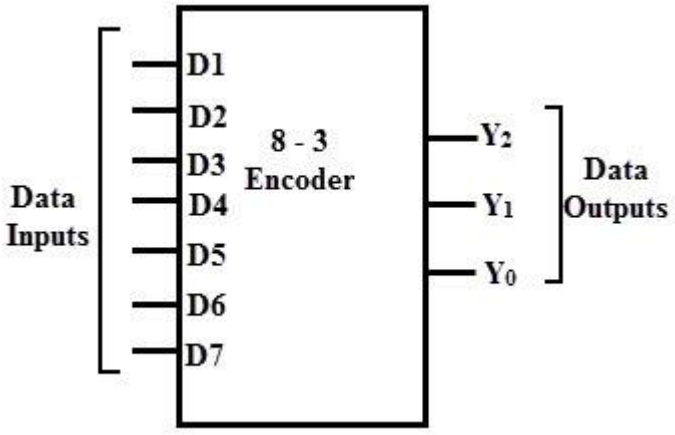
An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word.

Block diagram



Octal to Binary Encoder

Block Diagram



An octal to binary encoder has 8 input lines D0 to D7 and 3 output lines Y0 to Y2.

Truth table

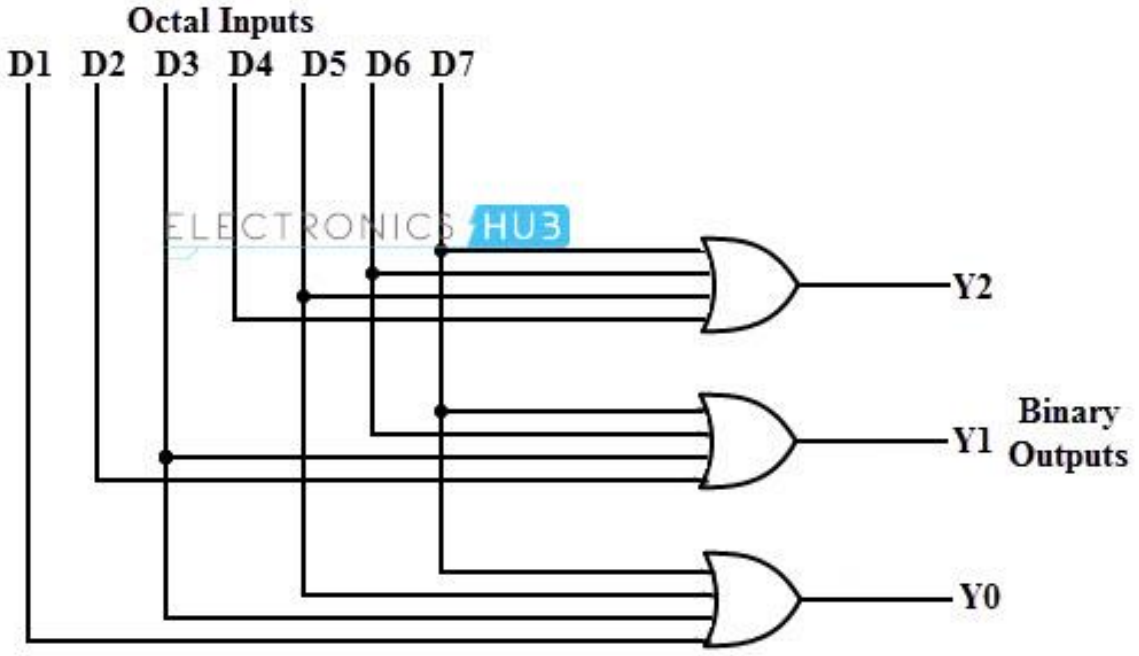
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

From the truth table, the outputs can be expressed by following Boolean Function.

Y0 = D1 + D3 + D5 + D7

Y1 = D2 + D3 + D6 + D7

Y2 = D4 + D5 + D6 + D7



6.2 Decoder

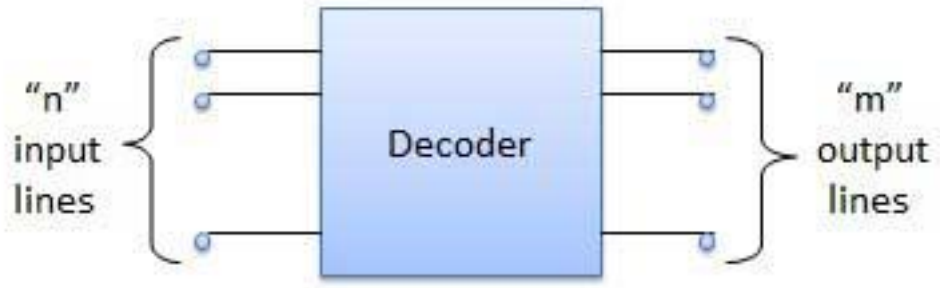
A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique information from n input lines to a maximum of 2^n unique output lines.

A decoder is a circuit that changes a code into a set of signals. It is called a decoder because it does the reverse of encoding

Decoding is the conversion of an encoded format back into the original sequence of characters.

A common type of decoder is the line decoder which takes an n-digit binary number and decodes it into 2n data lines.

Block diagram



2-to-4 line decoder

Truth Table

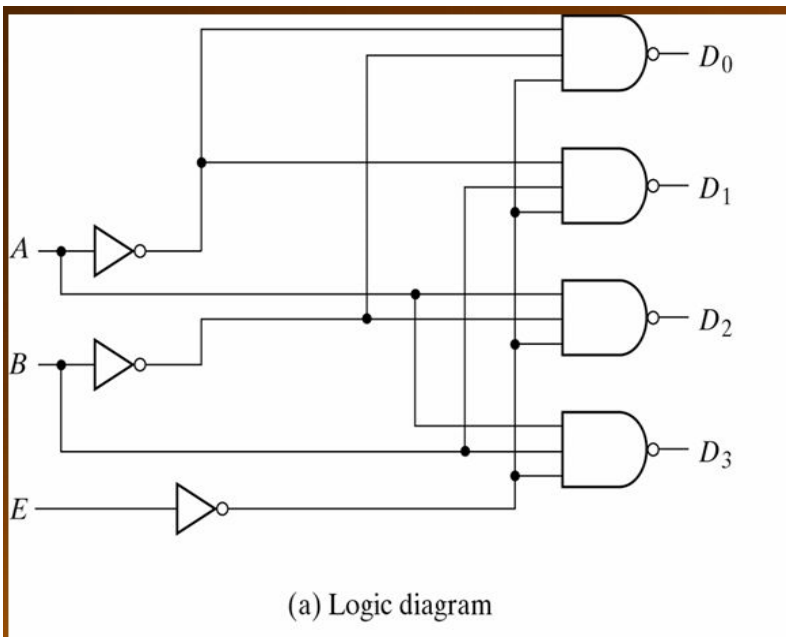
The circuit operates on complemented outputs and complemented enable input E.

The decoder is enable when E is equal to 0, only one output is equal to 0 at a given time and other three are equal to 1.

The circuit is disabled when E is equal to 1.

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Logic Circuit



(a) Logic diagram

3-to-8 line decoder

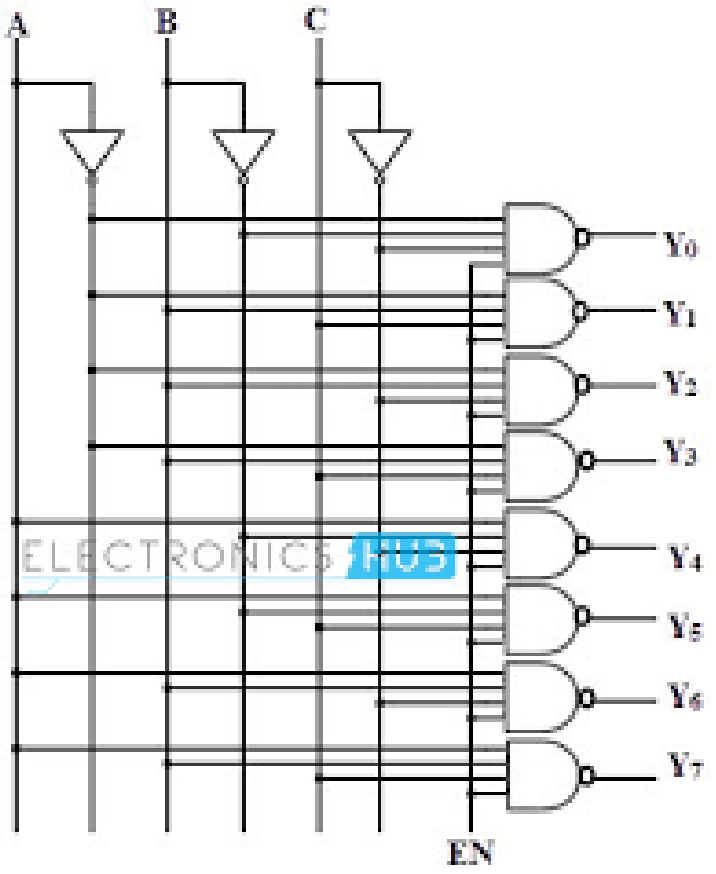
Truth Table

When enable input E is equal to 0, the output are equal to 0.

When enable input E is equal to 1, decoder operates in normal condition. For each possible combination there are seven outputs that are equal to 0 and only one that is equal to 1.

Inputs				Outputs							
EN	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Logic Circuit



6.3 Multiplexer

A multiplexer is a digital circuit has multiple inputs and single output.

The selection of the n inputs is done by the select input.It has one output selected at a time, it is known as data selector.

A multiplexer has

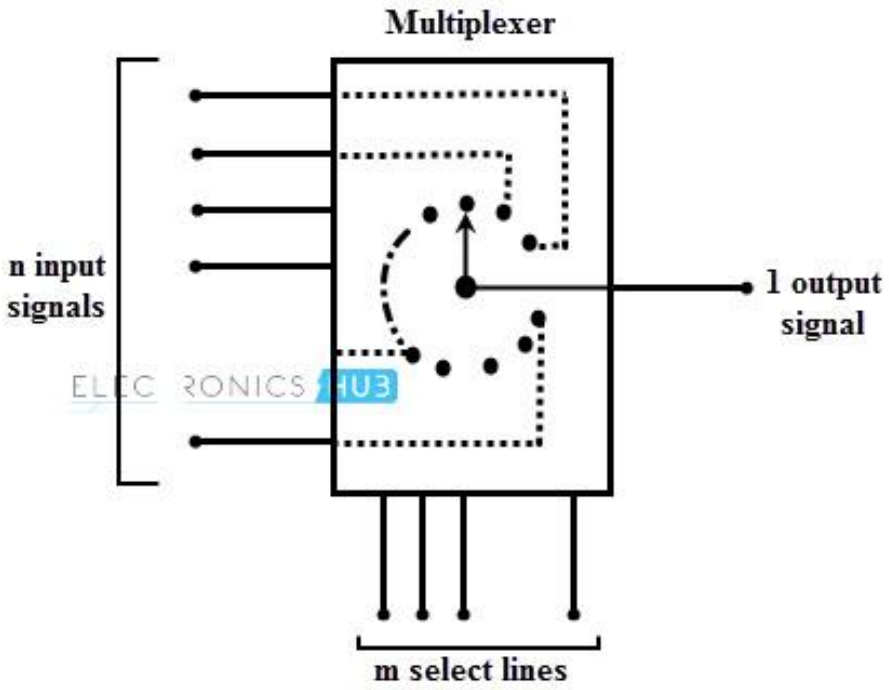
N data inputs(multiple)

1 output (single)

M select inputs with $2^M=N$

Application of Multiplexer

Block Diagram



Types

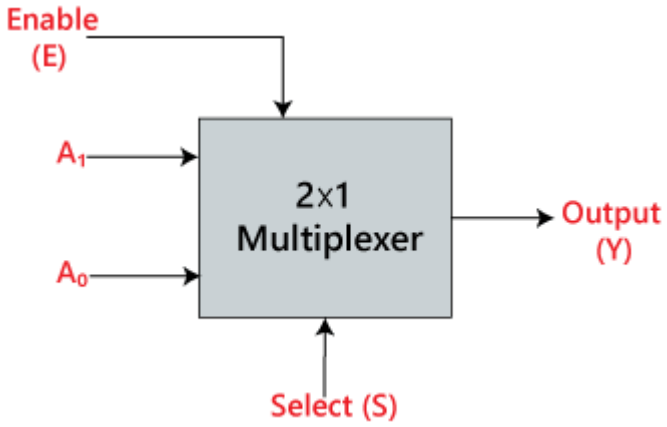
- 2-to-1 (1 Select line)
- 4-to-1 (2 Select line)
- 8-to-1 (3 Select line)
- 16-to-1 (2 Select line)

2-to-1 (1 Select line) Multiplexer

In 2×1 multiplexer, there are only two inputs, i.e., A_0 and A_1, 1 selection line, i.e., S_0 and single outputs, i.e., Y. On the basis of the combination of inputs which are present at the selection line S_0, one of these 2 inputs will be connected to the output.

The block diagram and the truth table of the 2×1 multiplexer are given below.

Block Diagram



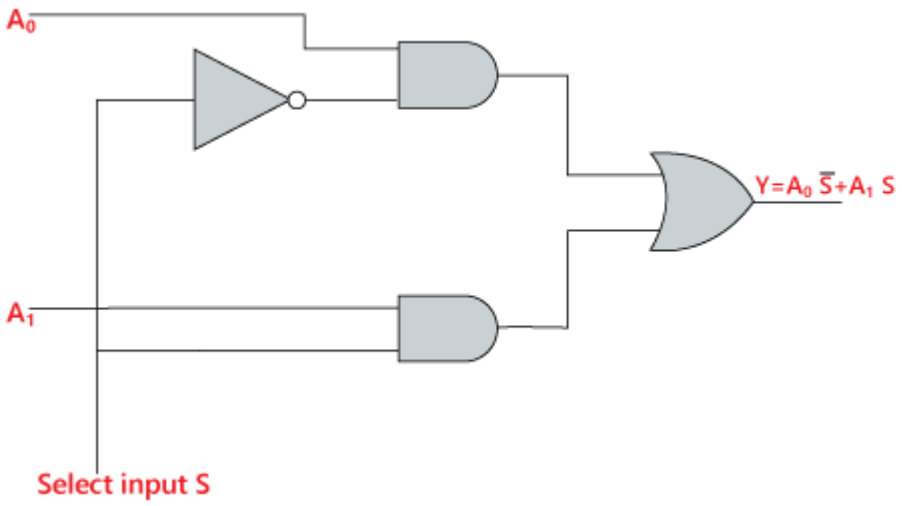
Truth Table:

INPUTS	Output
S_0	Y
0	A_0
1	A_1

The logical expression of the term Y is as follows:

$Y=S_0'.A_0+S_0.A_1$

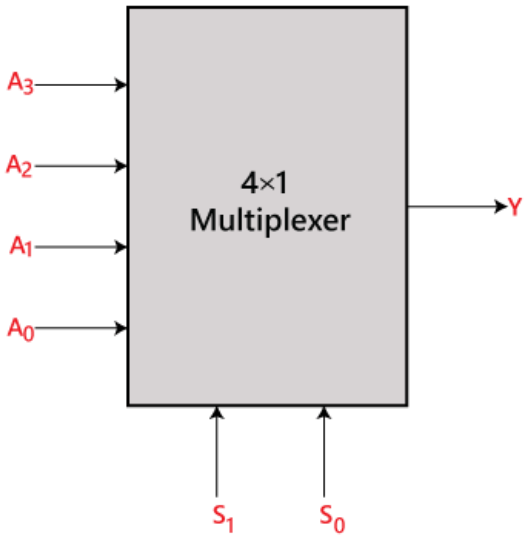
Logical circuit of the above expression is given below:



4-to-1 (2 Select line) Multiplexer

In the 4×1 multiplexer, there is a total of four inputs, i.e., A_0, A_1, A_2, and A_3, 2 selection lines, i.e., S_0 and S_1 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S^0 and S_1, one of these 4 inputs are connected to the output. The block diagram and the truth table of the 4×1 multiplexer are given below.

Block Diagram



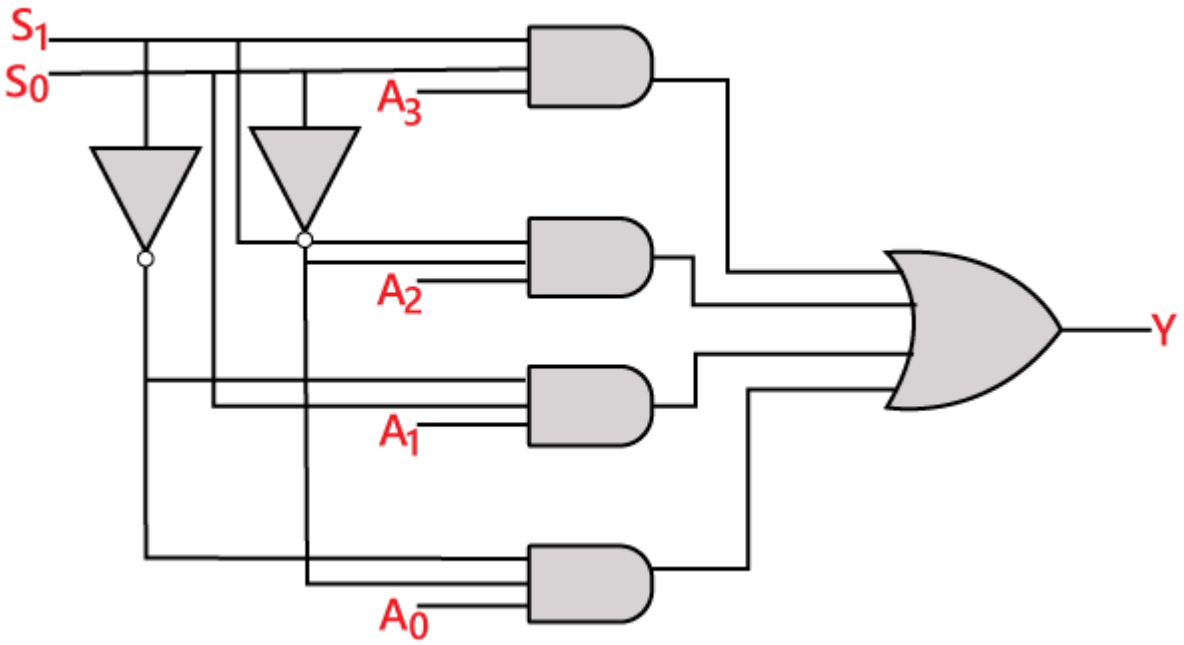
Truth Table:

INPUTS		Output
S_1	S_0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

The logical expression of the term Y is as follows:

$Y=S_1' S_0' A_0+S_1' S_0 A_1+S_1 S_0' A_2+S_1 S_0 A_3$

Logical circuit of the above expression is given below:

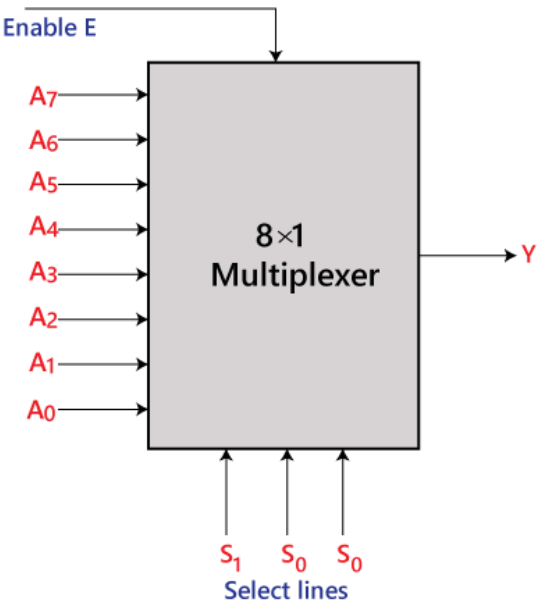


8-to-1 (3 Select line) Multiplexer

In the 8 to 1 multiplexer, there are total eight inputs, i.e., A_0, A_1, A_2, A_3, A_4, A_5, A_6, and A_7, 3 selection lines, i.e., S_0, $S_1$and S_2 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S_0, S_1, and S_2, one of these 8 inputs are connected to the output.

The block diagram and the truth table of the 8×1 multiplexer are given below.

Block Diagram



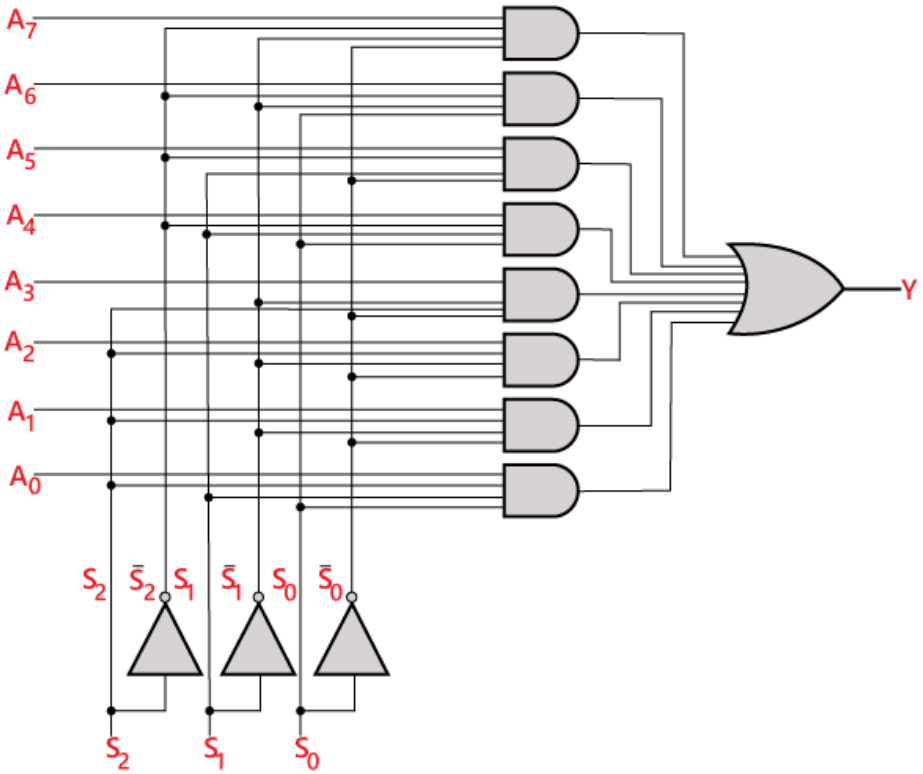
Truth Table

INPUTS			Output
S_2	S_1	S_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7

The logical expression of the term Y is as follows:

$Y=S_0'.S_1'.S_2'.A_0+S_0.S_1'.S_2'.A_1+S_0'.S_1.S_2'.A_2+S_0.S_1.S_2'.A_3+S_0'.S_1'.S_2\ A_4+S_0.S_1'.S_2\ A_5+S_0'.S_1.S_2\ .A_6+S_0.S_1.S_3.A_7$

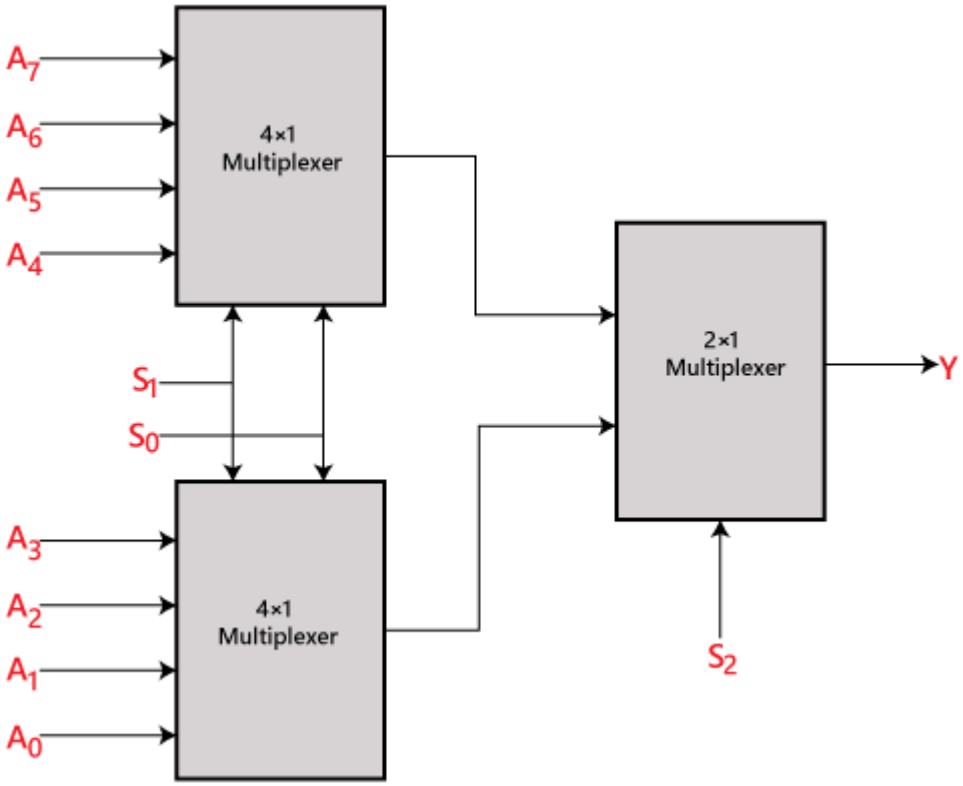
Logical circuit of the above expression is given below:



8 ×1 multiplexer using 4×1 and 2×1 multiplexer

We can implement the 8×1 multiplexer using a lower order multiplexer. To implement the 8×1 multiplexer, we need two 4×1 multiplexers and one 2×1 multiplexer. The 4×1 multiplexer has 2 selection lines, 4 inputs, and 1 output. The 2×1 multiplexer has only 1 selection line.

For getting 8 data inputs, we need two 4×1 multiplexers. The 4×1 multiplexer produces one output. So, in order to get the final output, we need a 2×1 multiplexer. The block diagram of 8×1 multiplexer using 4×1 and 2×1 multiplexer is given below.

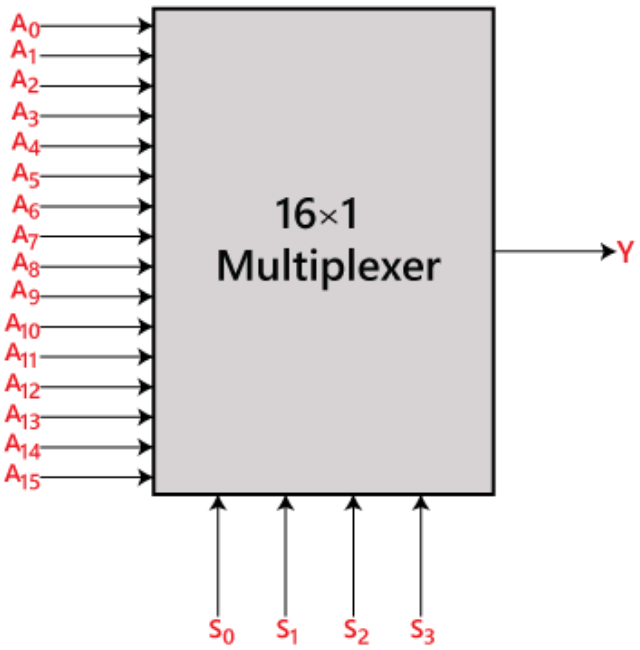


16-to-1 (4 Select line) Multiplexer

In the 16 to 1 multiplexer, there are total of 16 inputs, i.e., A_0, A_1, …, A_{16}, 4 selection lines, i.e., S_0, S_1, S_2, and S_3 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S_0, S_1, and S_2,S_3 one of these 16 inputs will be connected to the output.

The block diagram and the truth table of the 16×1

Block Diagram



Truth Table:

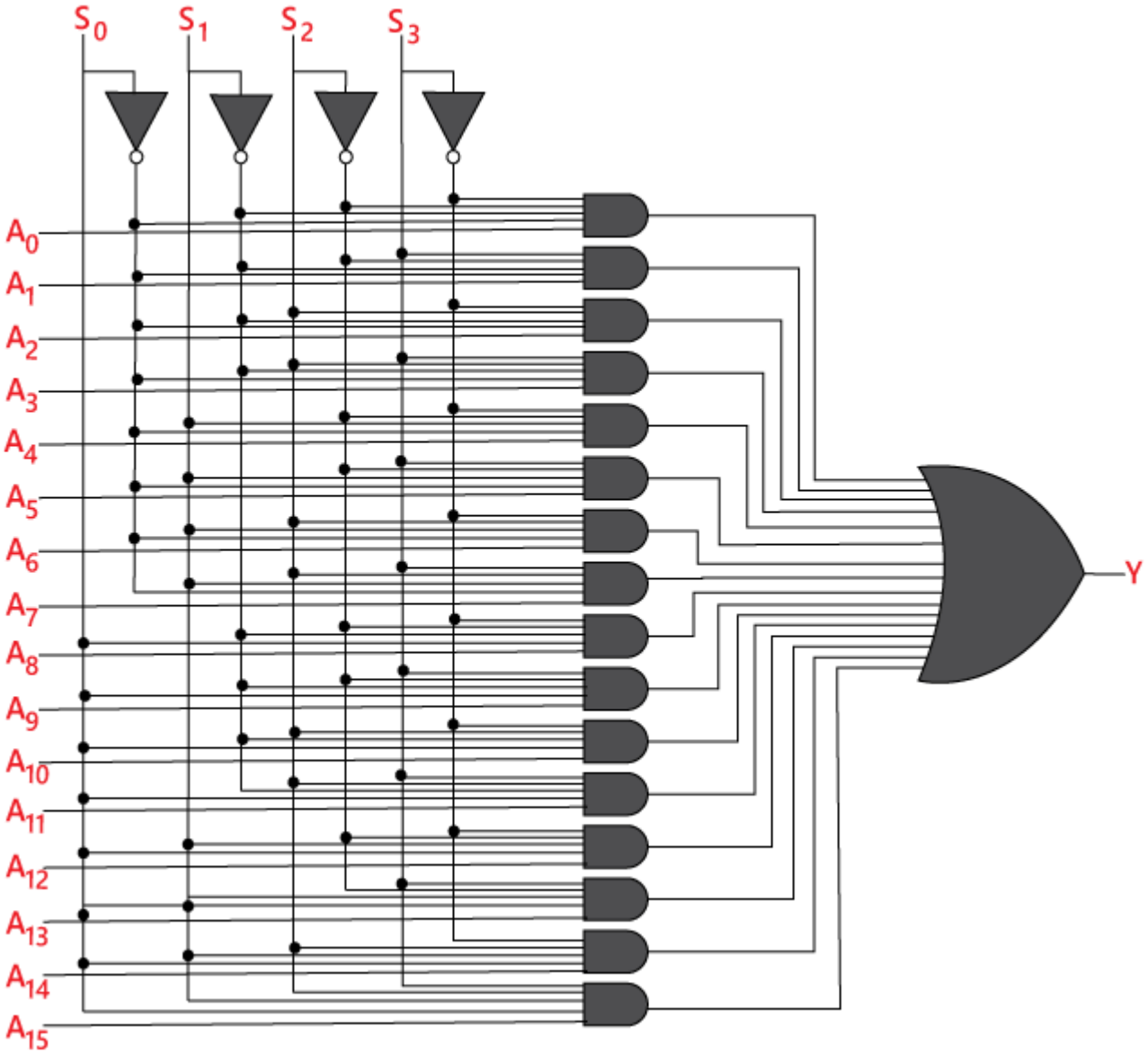
INPUTS				Output
S_0	S_1	S_2	S_3	Y
0	0	0	0	A_0
0	0	0	1	A_1
0	0	1	0	A_2
0	0	1	1	A_3
0	1	0	0	A_4
0	1	0	1	A_5
0	1	1	0	A_6
0	1	1	1	A_7
1	0	0	0	A_8
1	0	0	1	A_9
1	0	1	0	A_{10}
1	0	1	1	A_{11}
1	1	0	0	A_{12}
1	1	0	1	A_{13}
1	1	1	0	A_{14}
1	1	1	1	A_{15}

The logical expression of the term Y is as follows:

$Y=A_0.S_0'.S_1'.S_2'.S_3'+A_1.S_0'.S_1'.S_2\ '.S_3+A_2.S_0'.S_1'.S_2.S_3'+A_3.S_0'.S_1\ '.S_2.S_3+A_4.S_0'.S_1.S_2'.S_3'+A_5.S_0\ '.S_1.S_2'.S_3+A_6.S_1.S_2.S_3'+A_7.S_0\ '.S_1.S_2.S_3+A_8.S_0.S_1'.S_2'.S_3'+A_9\ .S_0.S_1'.S_2'.S_3+Y_10.S_0.S_1'.S_2.S_3\ '+A_11.S_0.S_1'.S_2.S_3+A_12$

$S_0.S_1.S_2\ '.S_3'+A_13.S_0.S_1.S_2'.S_3+A_14.S_0.S_1\ .S_2.S_3'+A_15.S_0.S_1.S_2'.S_3$

Logical circuit of the above expression is given below:

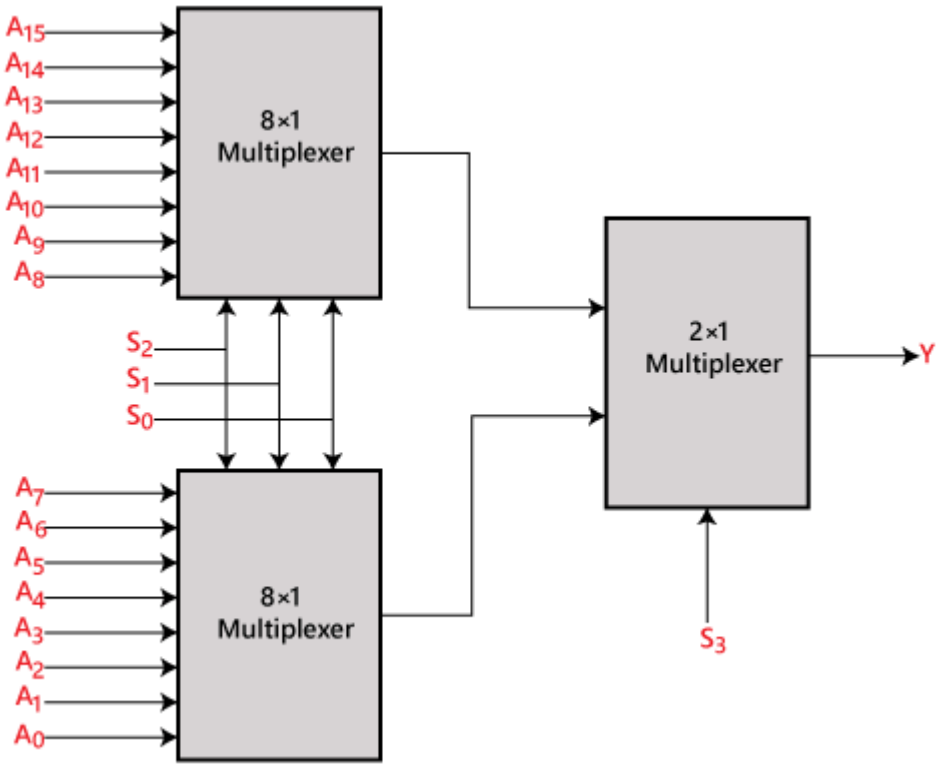


16×1 multiplexer using 8×1 and 2×1 multiplexer

We can implement the 16×1 multiplexer using a lower order multiplexer. To implement the 8×1 multiplexer, we need two 8×1 multiplexers and one 2×1

multiplexer. The 8×1 multiplexer has 3 selection lines, 4 inputs, and 1 output. The 2×1 multiplexer has only 1 selection line.

For getting 16 data inputs, we need two 8 ×1 multiplexers. The 8×1 multiplexer produces one output. So, in order to get the final output, we need a 2×1 multiplexer. The block diagram of 16×1 multiplexer using 8×1 and 2×1 multiplexer is given below.

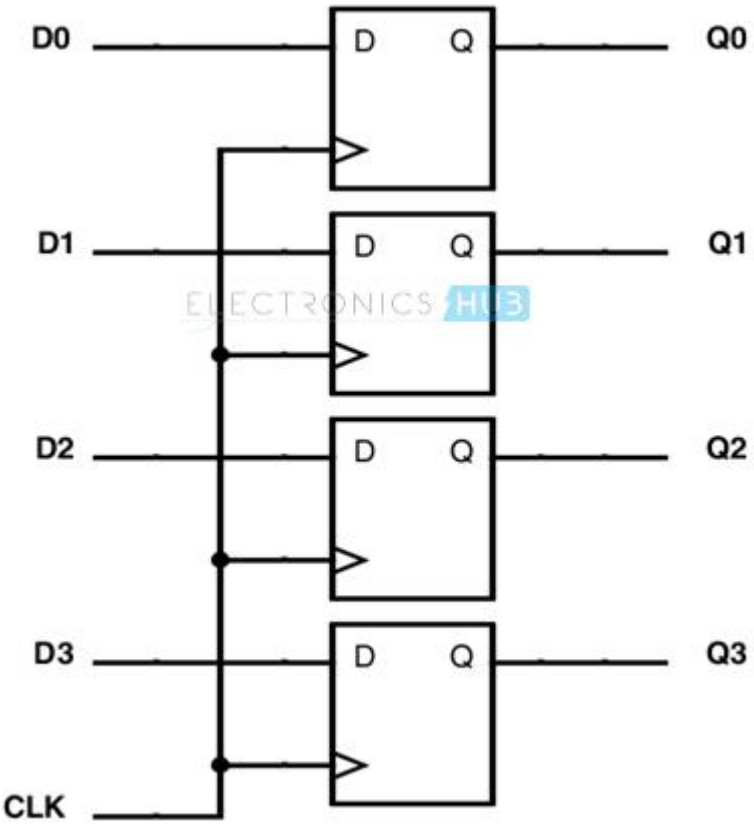


6.4 Registers

Register is a group of flip-flop capable of storing one bit of information. The n-bit register will consist of n number of flip-flop and it is capable of storing an n-bit word.

The binary data in a register can be moved within the register from one flip-flop to another. A register consists of a group of flip-flops and gates.

Following diagram shows register constructed with four D flip-flop.



The common clock input triggers all flip-flops on the rising edge of each pulse, and the binary data available at the four inputs are transferred into 4-bit register.

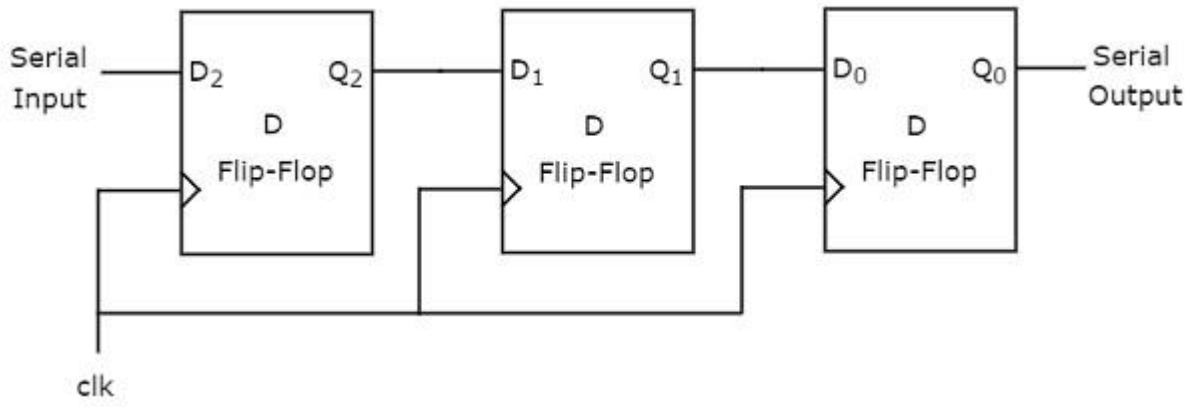
The clear input is useful for clearing the register.

Register load- The transfer of new information into a register is referred as loading.

6.5 Shift Registers

Shift Registers are sequential logic circuits, capable of storage and transfer of data. The shift register is capable of shifting bits either towards right hand side or towards left hand side.

They are made up of Flip Flops which are connected in such a way that the output of one flip flop could serve as the input of the other flip-flop, depending on the type of shift registers being created.



The output of a given flip-flop is connected to the D input of the flip-flop at its right. The clock is common to all flip-flops.

The serial input determines what goes into leftmost position during shift. The serial output determines what is taken from the rightmost flip-flop.

Bidirectional Shift Registers

Bidirectional shift register shift in both directions.

Following are the four types of shift registers based on applying inputs and accessing of outputs.

- Serial In - Serial Out shift register
- Serial In - Parallel Out shift register
- Parallel In - Serial Out shift register
- Parallel In - Parallel Out shift register

6.6 Binary Counters

What is a Counter?

In a digital logic system or computers, **counter can count and store the number of time any particular event or process have occurred, depending on a clock signal.**

A binary counter is a hardware circuit that is made out of a series of flip-flops. The output of one flip-flop is sent to the input of the next flip-flop in the series.

A binary counter can be constructed from J-K flip-flops by taking the output of one cell to the clock input of the next.

There are **two types of counters**

1. **Asynchronous counters**
2. **Synchronous counters**

1. Asynchronous Counters

If the flip-flops do not receive the same clock signal, then that counter is called as Asynchronous counter.

The output of system clock is applied as clock signal only to first flip-flop. The remaining flip-flops receive the clock signal from output of its previous stage flip-flop. Hence, the outputs of all flip-flops do not change affect at the same time.

Disadvantage: When counting a large number of bits, due to the chain system, **propagation delay** by successive stages became too large which is very difficult to get rid off. In such a situation, Synchronous counters are faster and reliable. There are also **counting errors** in Asynchronous Counter when high clock frequencies are applied across it.

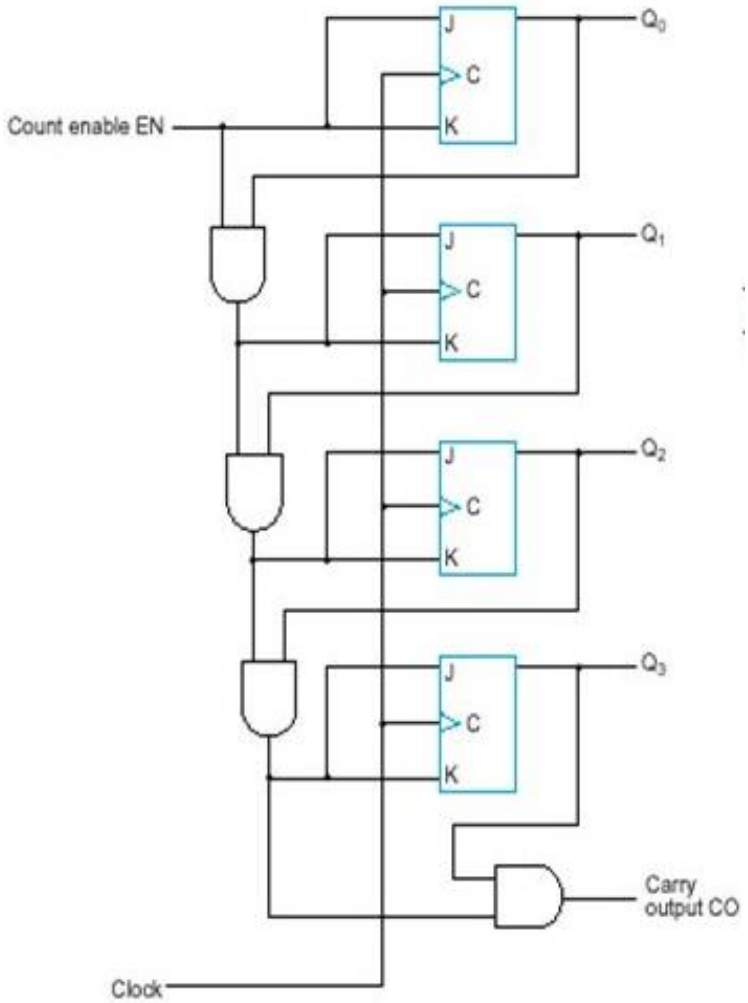
2. Synchronous Counter

The counters which use clock signal to change their transition are called “Synchronous counters”. This means the synchronous counters depends on their clock input to change state values.

The external clock signal is connected to the clock input of every individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in “synchronization” with the clock signal.

The result of this synchronization is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

In synchronous counters, all flip flops are connected to the same clock signal and all flip flops will trigger at the same time.



(a) Logic diagram

J and K inputs of flip-flops are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.

The C inputs of all flip-flops receive a common task.

If count enable is 0, all J and k inputs are maintained at 0 and the output of counter does not change.

The first stage Q_0 is complemented when counter is enabled.

Each of the other flip-flops are complemented when all previous least significant flip-flops are equal to 1.

6.7 Summary

In this chapter you have learned about:

- Encoder
- Decoder
- Multiplexer
- Registers
- Shift Registers
- Binary Counters

Unit 7: Memory

Unit Structure

7.1 Memory Unit

Memory unit is the amount of data that can be stored in the storage unit. This storage capacity is expressed in terms of Bytes.

A Memory Unit is a collection of storage cells together with associated circuits needed to transfer information in and out of storage.

Units of Memory

The storage capacity of the memory is expressed in various units of memory. These are as follows:

Bit

A microprocessor uses binary digits 0 and 1 to decide the OFF and ON state respectively, of various circuits. Furthermore, a bit is the smallest unit of representation in the binary language.

Nibble

A nibble is a collection of 4 bits.

Byte

A byte is the representation of a group of 8 bits. Moreover, a byte is a unit that expresses any word, symbol, or character in the computer language. Besides, computer memory is always in terms of multiples of bytes.

Word

A computer word is similar to a byte, as it is also a group of bits. Moreover, a computer word is fixed for each computer. At the same time it varies from computer to computer. Besides, the length of a computer word is the **word-size or word length**. Therefore, a computer stores information in the form of computer word.

Kilobyte

It is the most common unit of memory which is the smallest of all. But, it is greater than the byte.

The abbreviation for kilobytes is ‘KB’.

It contains 1000 bytes. Besides, it is synonyms to kibibytes which contain 1024 (2^10) bytes.

Megabytes usually measures the size of text documents, graphics of websites, individual files, etc.

Megabyte

The abbreviation for megabyte is ‘MB’.

It contains 1000,000 bytes. Besides, it is synonyms to mebibytes which contains 1048576 (2^20) bytes.

Kilobytes usually measure the size of large files. For example high-resolution images, songs, storage of compact disks, etc.

Gigabyte

The abbreviation for the gigabyte is ‘GB’ or ‘gigs’.

It contains 1000,000,000 bytes. Besides, it is synonyms to gibibytes which contain 1073741824 (2^30) bytes.

Kilobytes usually measure the capacity of storage devices.

Terabyte

The abbreviation for terabytes is ‘TB’.

It contains onetrillion bytes. Besides, it is synonyms to tebibytes which contains 2^40 bytes.

Kilobytes usually measure the capacity of large storage devices, for example, HDDs (Hard Disk Drives).

Petabyte

The abbreviation for petabyte is ‘PB’.

It contains 10^15 bytes. Besides, it is synonyms to pebibytes which contains 2^50 bytes.

Petabytes usually measure the total data storage in large networks or server farms. For example, the data in Google or Facebook data servers is around more than 10 PBs.

Exabyte

The abbreviation for exabyte is ‘EB’.

It contains 10^18 bytes. Besides, it is synonyms to exbibytes which contains 2^60 bytes.

The exabyte unit is so large that it does not even measure the storage of large cloud servers. Rather, it can be used to measure the amount of data transfer over the internet for a certain time limit.

Zettabyte

The abbreviation for zettabyte is ‘ZB’.

It contains 10^21 bytes. Besides, it is synonyms to zebibytes which contains 2^70 bytes.

It can measure a huge amount of data. In fact, the whole data in the world is just a few zettabytes.

Yottabyte

The abbreviation for yottabyte is ‘YB’.

It contains 10^24 zettabytes. Besides, it is synonyms to yobibytes which contains 2^80 bytes.

It is a tremendously huge unit of measurement. Therefore, it has no practical use.

7.2 Internal Structure of Memory Unit

The internal structure of a memory unit is specified by the number of words it contains and the number of bits in each word.

Special input lines called address lines select one particular word.

Each word in memory is assigned an identification number, called an address, staring from 0 and continuing with 1, 2, 3, up to 2^k - 1 where k is the number of address lines.

The selection of a specific word inside the memory is done by applying the k-bit binary address to the address lines.

A decoder inside the memory accepts this address and opens the paths needed to select the bits of the specified word.

Computer memories may range from 1024 words, requiring an address of 10 bits, to 232 words, requiring 32 address bits.

- **K(Kilo)** is equal to 2^{10}
- **M(Mega)** is equal to 2^{20}
- **G(Giga)** is equal to 2^{30}

Two major types of memories are used in computer systems:

1. Random Access Memory(RAM)
2. Read Only Memory (ROM)

7.3 Random Access Memory

RAM (Random Access Memory) is the internal memory of the CPU for storing data, program, and program result.

It is a read/write memory which stores data until the machine is working. RAM is volatile, As soon as the machine is switched off, data is erased.

Data in the RAM can be accessed randomly but it is very expensive.

RAM is of two types

- Static RAM (SRAM)
- Dynamic RAM (DRAM)

Now, let's understand types of RAM by below showing table as difference

DRAM	SRAM
1. Constructed of tiny capacitors that leak electricity.	1.Constructed of circuits similar to D flip-flops.
2.Requires a recharge every few milliseconds to maintain its data.	2.Holds its contents as long as power is available.
3.Inexpensive.	3.Expensive.
4. Slower than SRAM.	4. Faster than DRAM.
5. Can store many bits per chip.	5. Can not store many bits per chip.
6. Uses less power.	6.Uses more power.
7.Generates less heat.	7.Generates more heat.
8. Used for main memory.	8. Used for cache.

Write and Read Operations

The two operations that a random access memory can perform are **the write and read operations**.

The write signal specifies a transfer-in operation and **the read signal** specifies a transfer-out operation.

On accepting one of these control signals. The internal circuits inside the memory provide the desired function. The steps that must be taken for the purpose of transferring a new word to be stored into memory are as follows:

1. Apply the binary address of the desired word into the address lines.
2. Apply the data bits that must be stored in memory into the data input lines.
3. Activate the write input.

The memory unit will then take the bits presently available in the input data lines and store them in the specified by the address lines.

The steps that must be taken for the purpose of transferring a stored word out of memory are as follows:

1. Apply the binary address of the desired word into the address lines.
2. Activate the read input.

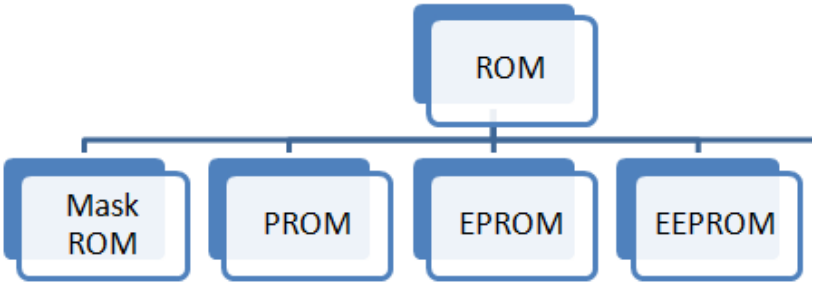
The memory unit will then take the bits from the word that has been selected by the address and apply them into the output data lines. The content of the selected word does not change after reading.

7.4 Read only Memory

ROM stands for Read Only Memory. The memory from which we can only read but cannot write on it. This type of memory is non-volatile.

The information is stored permanently in such memories during manufacture. A ROM stores such instructions that are required to start a computer. This operation is referred to as bootstrap

7.5 Types of Read Only Memory



1. MROM (Masked ROM)

The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kind of ROMs are known as masked ROMs, which are inexpensive.

2. PROM(Programmed Read Only Memory)

- It is a programmable device. User can programmed, but once programmed, the contents can not be deleted.
- A special equipment PROM programmer is used to store programs in a PROM.

3. EPROM(Erasable Programmed Read Only Memory)

- User can program on EPROM chip and can delete the contents also.
- The content of EPROM can be erased by using ultraviolet rays. For this the chip has to be removed from system and putting inside EPROM eraser equipment to be exposed to UV light.
- Deleting one or more locations is not possible but entire contents are erased.

4. EEPROM(Electrically Erase PROM)

- The contents are deleted electrically.
- The contents of any location can be altered without erasing the complete contents and need not be removed chip from the computer.
- It can be reprogrammed and erased more than 10,000 times.

7.6 Summary

In this chapter you have learned about:

- Memory Unit
- Internal Structure of Memory Unit
- Random Access Memory
- Read only Memory
- Types of Read Only Memory

Unit 8: Bus and Memory Transfer

Unit Structure

8.1 Concept of BUS

A bus is a unit use to exchange data between the one part of computer to the other part

Bus is a collection of physical connections like cables, circuits or wires etc. through which the data is transmitted.

Bus can be shared by multiple hardware components.

Bus carries signals which can be unidirectional or bidirectional.

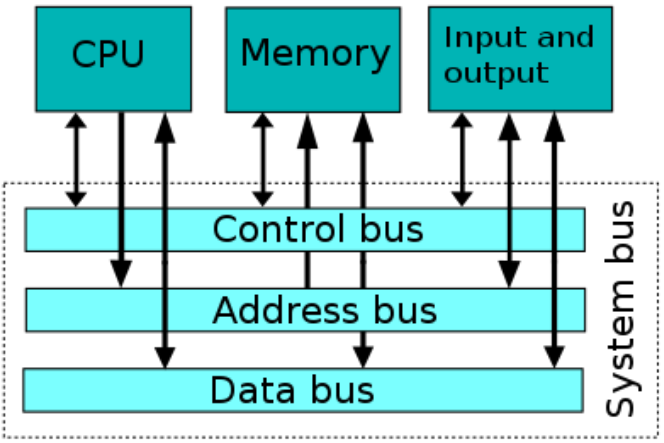
Bus Width

The bus size is determined by amount of information that can be transferred at a time known as its width.

This amount is in bits, corresponds to the number of physical lines over which data is sent simultaneously.

Ex. 32 bit bus can transmit 32 bits , 64 bit bus can transmit 64 bits

All the computer use three 3 types of bus
1. Control bus
2. Data bus
3. Address bus



Control bus

It is used by CPU to monitor the other active part of computer.

It is used to transmit various signals like read, write, interrupt etc.

Data bus

It is used to transfer all data and instruction between different parts of computer.

It is bidirectional which can only transmit in one direction at a time.

It transfers data from memory to CPU and also between memory and I/O.

Address bus

Used to transports memory address which the CPU wants to access for read or write data.It is unidirectional.

Before data or instruction can be written into or write from memory by the CPU or I/O, an address must be transmitted to memory over an address bus.

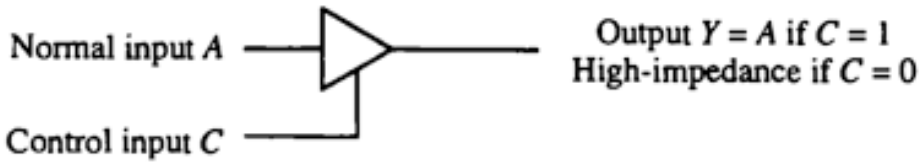
8.2 Three-State Buffer

A three-state gate is a digital circuit that exhibits three states. Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate. The third state is a high-impedance state.

The high-impedance state (Disable) behaves like an open circuit which means that the output is disconnected and does not have logic significance.

The graphic symbol of a three-state buffer gate is shown below.

Figure 4-4 Graphic symbols for three-state buffer.



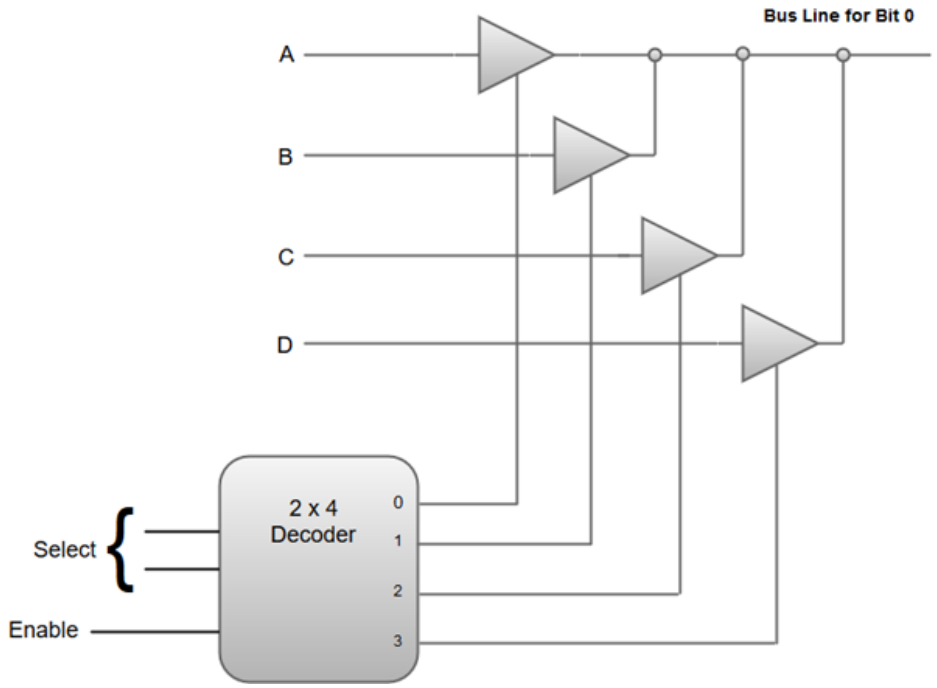
It is distinguished from a normal buffer by having both a normal input and a control input. The control input determines the output state.

When the control input is equal to 1, the output is enabled and the gate behaves like any conventional buffer, with the output equal to the normal input.

When the control input is 0, the output is disabled and the gate gives to a high-impedance state.

The construction of a bus system with three-state buffers is shown in following diagram.

Bus line with three state buffer:



8.3 Memory Transfer

One way of constructing a common bus system is with multiplexers. The multiplexers select the source register whose binary information is then placed on the bus.

The construction of a bus system for four registers is shown in diagram.

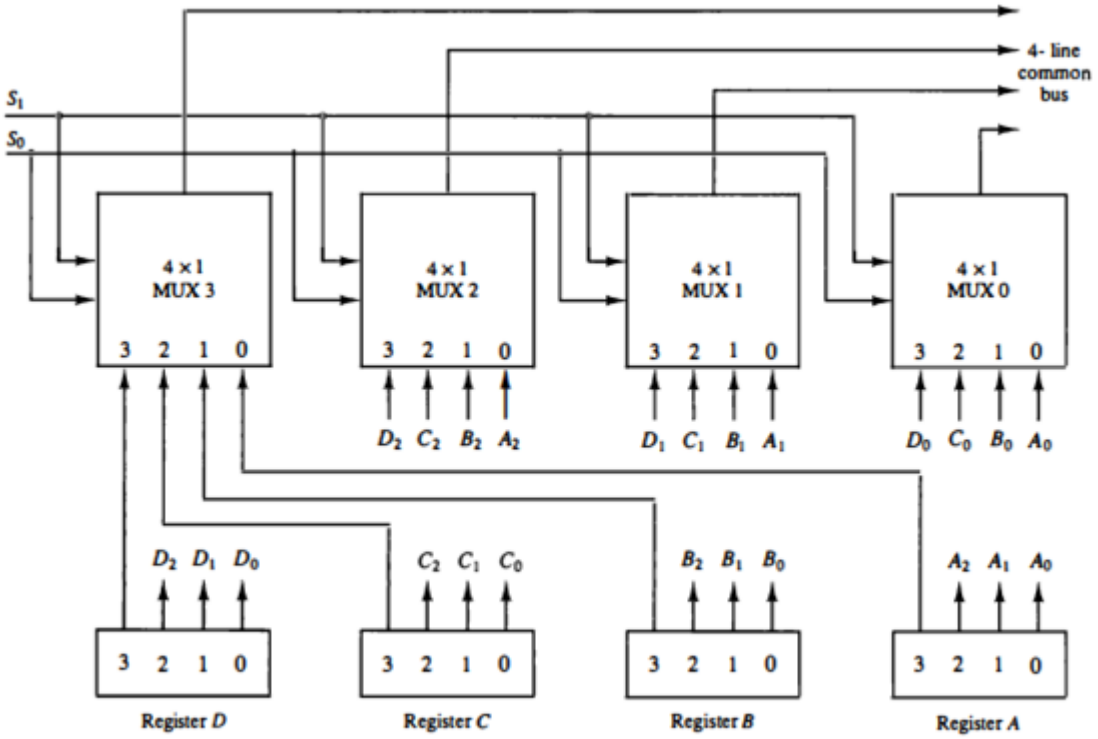


Figure: A bus system for transfer of contents of four registers

Each register has four bits, numbered 0 through 3. The bus consists of four 4 ×1 multiplexers each having four data inputs, 0 through 3, and two selection inputs, S1and S0.

Thus MUX 0 multiplexes the four 0 bits of the registers,MUX 1 multiplexes the four 1 bits of the registers, and similarly for the other two bits.

The two selection lines S1and S0 are connected to the selection inputs of all four multiplexers.

Truth Table

TABLE Function Table for Bus of Fig.

S_1	S_0	Register selected
0	0	*A*
0	1	*B*
1	0	*C*
1	1	*D*

The selection lines choose the four bits of one registerand transfer them into the four-line common bus. When S1S0= 00, the 0 data inputs of all four multiplexers are selected and applied to the outputs that form the bus.

This causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers.

Similarly, register B is selected if S1S0= 01, and so on.The transfer of information from a bus into one of manydestination registers can be accomplished by connecting the bus lines to the inputs of all destination registers and activating the load control of the particular destination register selected.

The symbolic statement for a bus transfer may mention the bus or its presence may be implied in the statement.When the bus is includes in the statement, the register transfer is symbolized as follows:

BUS ←C, R1 ←BUS

The content of register C is placed on the bus, and the content of the bus is loaded into register R1 by activating its load control input. If the bus is known to exist in the system, it may be convenient just to show the direct transfer.

R1 ←C

8.4 Instruction Execution

Whenever CPU needs to read/write a byte of data to/from the memory, it specifies the address on address bus since address bus is unidirectional only CPU specifies address.Data is placed on data bus since it is bidirectional both CPU and the memory place information on it.

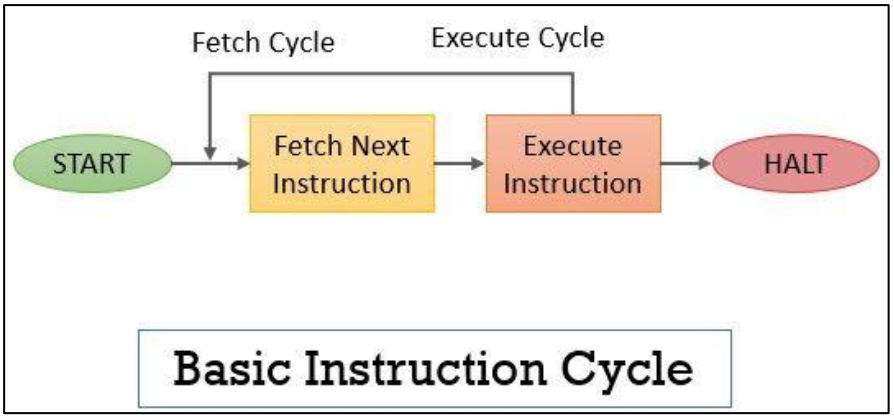
The read/write wire in the control bus is set to “1” to indicate that the memory is to be read. It is set to “0” to indicate that the memory is to be written.The ready line in the address bus is set to “1” by the CPU whenever there is a valid address on the address bus.

The major types of bus cycles are

No.	Name	Description
1	I/O read Cycle	CPU reads data from an input device.
2	I/O write Cycle	CPU writes into an output device.

3	Memory read Cycle	CPU reads data from a memory location.
4	Memory write Cycle	CPU writes into a memory location.

Instruction Fetch Execute Cycle



Steps of fetch execute cycle

1. CPU examines PC (Program Counter) register. It gives address of the instruction to be brought in from memory.

2. The processor then sets the read/write line to “1” to indicate that the memory value is to be read into the CPU. After that it sets Address bus enable line to indicate that the address bus holds a valid address.

3. The memory fetches the byte from the specified address and places it on the data bus.

4. The memory then sets the Data Bus Enable line to indicate to places it on the data bus.

5. The CPU reads the byte from the data bus is called “fetch”.

6. As the instruction has been executed, CPU increases program counter(PC)

Summarized Steps

- CPU fetch instruction from main memory
- Increment program counter
- Decode Instruction

- Fetch operands from memory
- Execute Instruction
- Write Results to memory

8.5 Interrupt

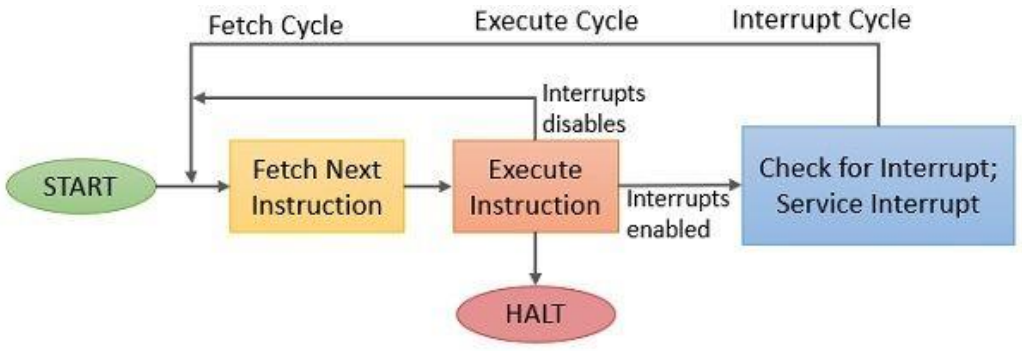
An interrupt is an event that occurs while the processor is executing an instruction.

CPU suspends the current program execution & a branch to ISR (Interrupt service routine).ISR is a program that services the interrupt by taking appropriate actions. After the execution of ISR, CPU returns back to interrupted program & resume its work.

The condition of CPU should be saved before taking up ISR before returning from ISR this saved status should be loaded into CPU.

Interrupt which are generated by other devices such as printer is called hardware interrupt. The processor is capable of responding 15 hardware interrupt.

Interrupt signal initiated by program is called software interrupt. The processor is capable of responding 256 software interrupt.



Instruction Cycle with Interrupts

Types of Interrupt

1. Maskable Interrupt
A Maskable interrupt is an interrupt whose trigger event is not always important.

When CPU needs continuous work for a program CPU's interrupt reorganization can be suspended by masking it.

Masking interrupt at CPU level is done by resting the IE (Interrupt enable) flag. If it is "1" the CPU sense the interrupt,if it is "0" CPU ignores interrupt.

2. Non-Maskable Interrupt
A non-Maskable interrupt is so important that it should never be ignored. Ex. reset button.

This type of interrupt has to be serviced without delay otherwise damage may be caused to data or program.

3. Interrupt priority
When two interrupt happen at the same time, the higher priority interrupt will take precedence over the lower priority one.

Examples of Interrupt

- Divide by 0
- Virtual memory page fault
- Illegal instruction
- Reset
- Power failure
- External hardware device

8.6 Summary

In this chapter you have learned about:

- Concept of Bus
- Three-State Buffer
- Memory Transfer
- Instruction Execution
- Interrupt

BLOCK 3: CPU DESIGN AND INSTRUCTION SET

Block Introduction

In this block-3 of digital electronics, I have tried to emphasis on: language used for defining micro operations, various micro operations in computer, basic computer organization and about central processing unit. Basically, I introduced concept of RTL used in computer to define any operations which happen.

I hace also emphasis on working of central processing unit , basic computer instruction format and various addressing format used by various types of instructions.

Block Objective

The objective of the block is to explain concept of register transfer language used in computer for any operation and basic instruction format and its types for better understanding of working of instruction cycle.

By learning this block of digital electronics student will also learn about more details of microoperations . Reader of this block, will know instruction formats and how transfer is happen between CPU and memory can possible.

Unit 9: Register Transfer And Microoperation

Unit Structure

7.1 Register Transfer Language

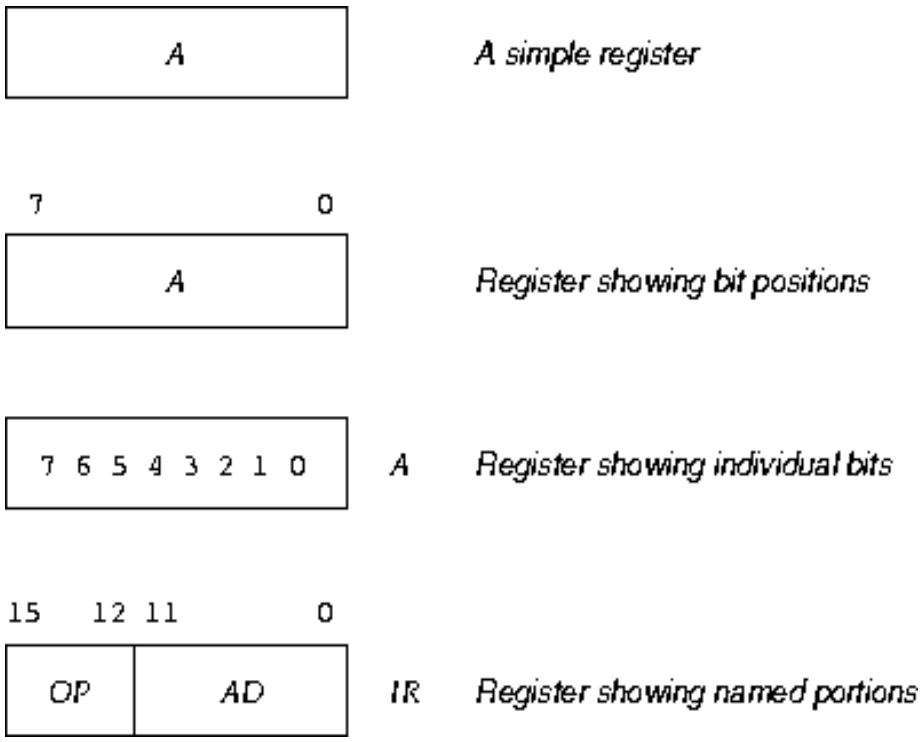
A register transfer language is a symbolic notation used to describe the microperation transfers between registers.

Registers are denoted by capital letters and are sometimes followed by numerals, e.g.

1. MAR – Memory Address Register (holds addresses for the memory unit)
2. PC – Program Counter (holds the next instruction's address)
3. IR – Instruction Register (holds the instruction being executed)
4. R1 – Register 1 (a CPU register)

We can indicate individual bits by placing them in parentheses. e.g., PC (8-15), R2 (5), etc.

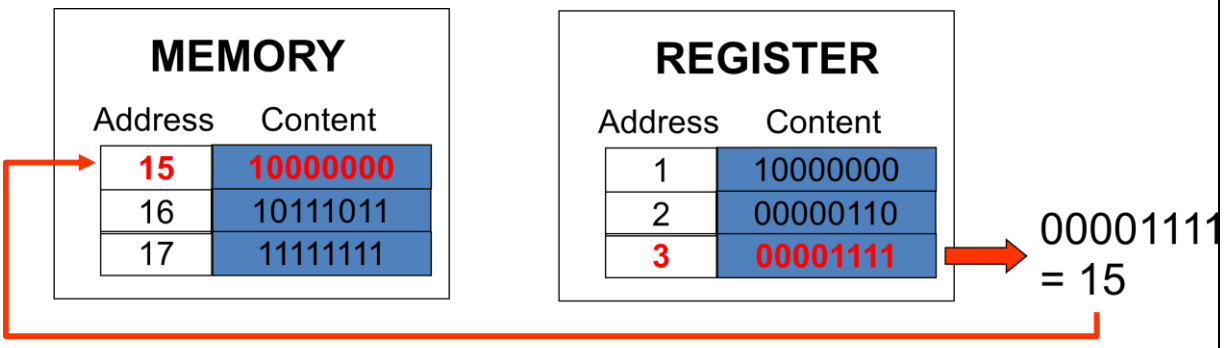
Block Diagrams of Registers



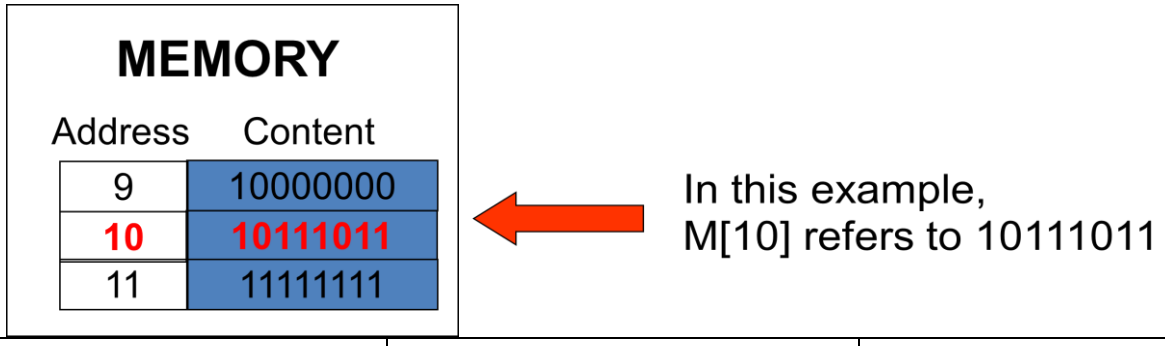
Basic Symbols for Register Transfer

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	MAR, R2
R followed by a number is referring to a register: **R2** = second register/register no 2 **R2**		
Parentheses ( )	Denotes a part of a register	R2(0-7), R2(L)
Arrow →	Denotes Transfer of information	R2 ←R1
R2←R1 = Stores the value of R1 into R2		
Comma ,	Separates 2 microoperations	R1 ←R2, R6 ←R7
R1← R2, R6← R7 ➔ Stores R2 into R1 and at the same time stores R7 into R6.		
Square brackets	Specifies an address for memory	M[R3]

M[R3] = content of the memory address in R3



M[10] = contents of memory address 10



Colon	Denotes termination of control function	K:

K: R1←R2

➔If K=1, then stores R2 into R1.

K1K2': R3←R2

- If K=1 and K2=0, then stores R2 into R3.

K: a control signal generated in the control unit, 0 or 1

MATHEMATICAL AND LOGICAL SYMBOLS

- Addition is indicated by the + sign:

  R1← R2+R3

  Add R2 and R3, stores in R1

  R2← R4+R1

  Add R4 and R1, stores in R2

- Subtraction is handled not with the minus sign but with complementing.

  1's complement :

  $R5 \leftarrow R3+\overline{R4}$

  R3 minus R4 in 1's complement

  2's complement :

  $R5 \leftarrow R3+\overline{R4}+1$

  R3 minus R4 in 2's complement

ARITHMETIC MICROOPERATIONS

Symbolic Designation	Description
$R0 \leftarrow R1 + R2$	Addition
$R0 \leftarrow \overline{R1}$	Ones Complement
$R0 \leftarrow \overline{R1} + 1$	Two's Complement
$R0 \leftarrow R2 + \overline{R1} + 1$	R2 minus R1 (2's Comp)
$R1 \leftarrow R1 + 1$	Increment (count up)
$R1 \leftarrow R1 - 1$	Decrement (count down)

COMPLEX LOGICAL SYMBOLS

- Content of R5 will be stored in R4 only **IF** both condition K1 **and** condition K2 are true:

If K1=1 **and** K2=1, then stores R5 into R4

Content of R5 will be stored in R4 **IF**

either condition K1 **or**condition K2 were

true, a **+** sign would be used:

In (K1 + K2),"+" means "**OR**"

In R1 ← R1 + R3, "+" means "**plus**"

- If **– then – else** is implemented with **commas** and **multiple colons**.
- If K1 true, then stores R4 into R6, else if K2 true, then stores R5 into R6, else store R7 into R6.

7.2 Memory Read Operation

The read signal specifies a transfer-out operation.

The steps that must be taken for the purpose of transferring a stored word out of memory are as follows:

1. Apply the binary address of the desired word into the address lines.
2. Activate the read input.

The memory unit will then take the bits from the word that has been selected by the address and apply them into the output data lines. The content of the selected word does not change after reading.

7.3 Memory Write Operation

The two operations that a random access memory can perform are **the write and read operations**.

The write signal specifies a transfer-in operation

On accepting one of these control signals. The internal circuits inside the memory provide the desired function. The steps that must be taken for the purpose of transferring a new word to be stored into memory are as follows:

1. Apply the binary address of the desired word into the address lines.
2. Apply the data bits that must be stored in memory into the data input lines.
3. Activate the write input.

The memory unit will then take the bits presently available in the input data lines and store them in the specified by the address lines.

7.4 Micro operations

The micro operation is inside the computer and for executing any instruction CPU has to perform some micro operation.

Types of micro operation are

- Increment counter
- Resetting register
- Perform arithmetic and logical operation
- Data transfer between the register
- Reading from memory
- Writing into memory

This micro operation is executed when the corresponding control signal is issued by CPU.

Example

No.	Control Signal	Micro Operation
1	MAR ⟵ PC	Content of PC is transfer to MAR
2	PC ⟵ PC +1	Content of PC are incremented
3	IR⟵MBR	Content of MBR are copied to IR
4	MBR⟵ AC	Content of AC are copied to MBR

7.5 Summary

In this chapter you have learned about:

- Register Transfer Language
- Memory Read Operation
- Memory Write Operation
- Micro operations

Unit 10: Arithmetic Micro Operations

Unit Structure

10.1 Introduction

The basic arithmetic micro operations are addition,subtraction, increment, decrement, and shift.

Arithmetic shifts are explained later in conjunction with the shift micro operations. The arithmetic micro operation defined by the statement specifies an add micro operation.
R3 ←R1 + R2

It states that the contents of register R1 are added to the contents of register R2 and the sum transferred to register R3.

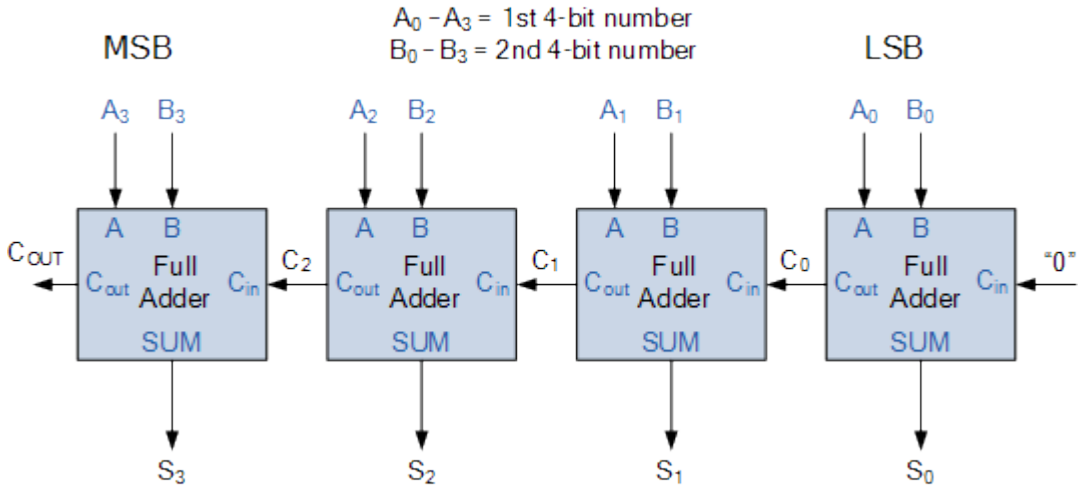
Subtraction is most often implemented through complementation and addition. Instead of using the minus operator, we can specify the subtraction by the following statement:
R3 ←R1 + R2+ 1

R2 is the symbol for the 1's complement of R2. Adding 1 to the 1's complement produces the 2's complement. Adding the contents of R1 to the 2's complement of R2 is equivalent to R1 – R2.

The multiplication operation is implemented with a sequence of add and shift micro operations. **Division** is implemented with a sequence of subtract and shift micro operations.

10.2 Binary Adder

The digital circuit that generates the arithmetic sum of two binary numbers of any lengths is called a binary adder. Thebinary adder is constructed with full-adder circuits connected in cascade.



The output carry from one full-adder connected to the input carry of the next full-adder.

The bits of A and the bits of B are designated by subscript numbers from right to left.(from registers)

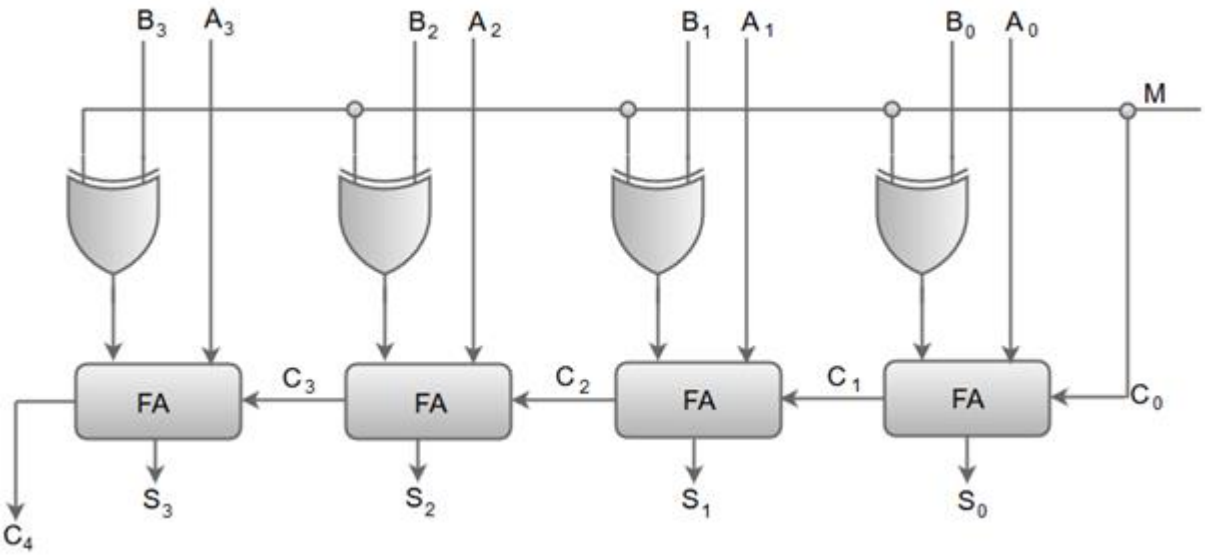
With subscript 0 denoting the low-order bit. The carries are connected in a chain through the full-adders. The input carry to the binary adder is C0 and the output carry is C4.

The S outputs of the full-adders generate the required sum bits.

10.3 Binary Adder-Subtractor

The addition and subtraction operations can be combined into one common circuit by including an exclusive-OR gate with each full-adder.

4 bit adder-subtractor:



The mode input M controls the operation. **When M = 0** the circuit is an adder and **when M = 1** the circuit becomesa subtractor.

Each exclusive-OR gate receives input M and one of the inputs of B.

When M = 0, we have B $\oplus$0 = B. the full-adders receive the value of B, the input carry is 0, and the circuit performs A plus B. **When M = 1**, we have B $\oplus$1 = B' and C0=1.

The B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation Aplus the 2's complement of B

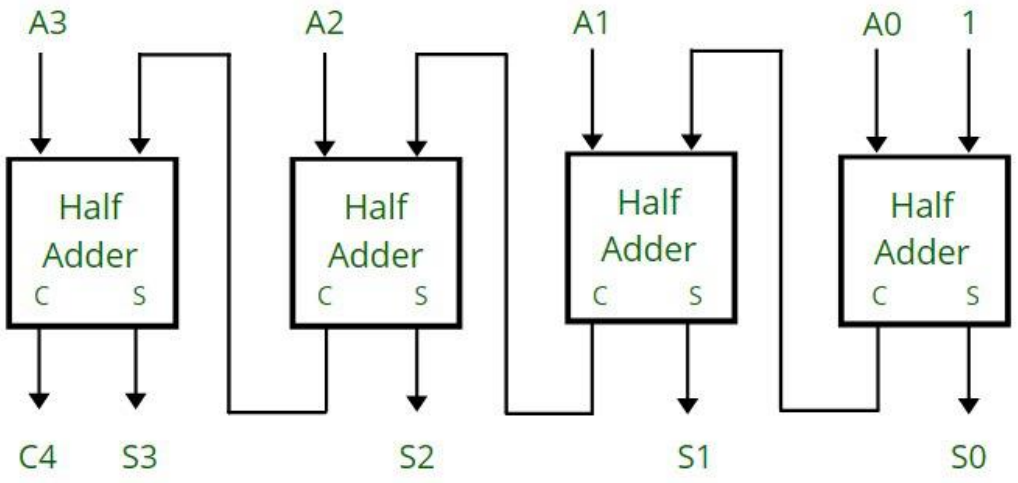
10.4 Binary Incrementor

The increment micro operation adds one to a number in a register.

For example, if a 4-bit register has a binary value 0110, it will go to 0111 after it is incremented.

This micro operation is easily implemented with a binary counter. Every time the count enable is active, the clock pulse transition increments the content of the register by one.

This can be accomplished by means of half-adders connected in cascade.



4- Bit Binary Incrementer

The least significant bit must have one input connected to logic-. The other inputs receive the number to be incremented or the carry from the previous stage.

The output carry from one half-adder is connected to one of the inputs of the next-higher-order half-adder. The circuit receives the four bits from A0through A3, adds one to it, and generates the incremented output in S0through S3.

The output carry C4will be 1 only after incrementing binary 1111. This also causes outputs S0 through S3 to go to 0.

10.5 Summary

- Introduction
- Binary Adder
- Binary Adder-Subtractor
- Binary Incrementor

Unit 11: Basic Computer Organization

Unit Structure

- Instruction Code
- Basic Computer Instructions
- Addressing
- Instruction Cycle
- Summary

11.1 Instruction Code

An instruction code is a group of bits that instruct the computer to perform a specific operation.

It is usually divided into parts

1. operation code
2. number of bits

The operation code of an instruction is a group of bits that define such operations as add, subtract, multiply, shift, and complement.

The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer.

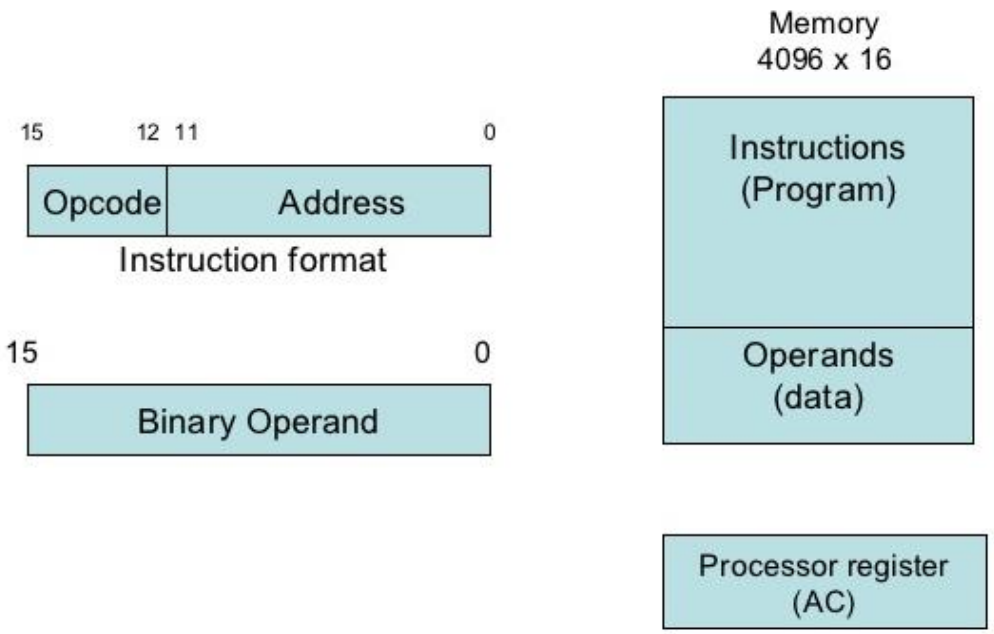
The operation code must consist of at least n bits for a given 2^n operations.

The operation part of an instruction code specifies the operation to be performed.This operation must be performed on some data stored in processor registers or in memory.

Stored Program Organization

Instruction code format is divided in two parts.

The first part specifies the operation to be performed and the second specifies an address.



Instructions are stored in one section of memory and data in another. For a memory unit with 4096 words we need 12 bits to specify an address since $2^{12} = 4096$.

If we store each instruction code in one 16-bit memory word, we have available four bits for the operation code to specify one out of 16 possible operations, and 12 bits to specify the address of an operand.

The operation is performed with the memory operand and the content of AC. For example, operations such as clear AC, complement AC, and increment AC operate on data stored in the AC register. They do not need an operand from memory.

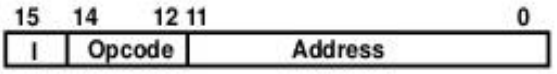
11.2 Basic Computer Instructions

The basic computer has three instruction code formats, as shown in following diagram. Each format has 16 bits.

BASIC COMPUTER INSTRUCTIONS

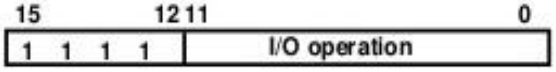
- **Basic Computer Instruction Format**

Memory-Reference Instructions (OP-code = 000 ~ 110)



Register-Reference Instructions (OP-code = 111, I = 0)

Input-Output Instructions (OP-code =111, I = 1)



The operation code (opcode) part of the instruction contains three bits and the meaning of the remaining 13 bits depends on the operation code encountered.

A memory reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I. I is equal to 0 for direct address and to 1 for indirect address.

The register-reference instructions are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction. A register-reference instruction specifies an operation on or a test of the AC register. An operand from memory is not needed; therefore, the other 12 bits are used to specify the operation.

An input-output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits are used to specify the type of input-output operation or test performed.

11.3 Addressing

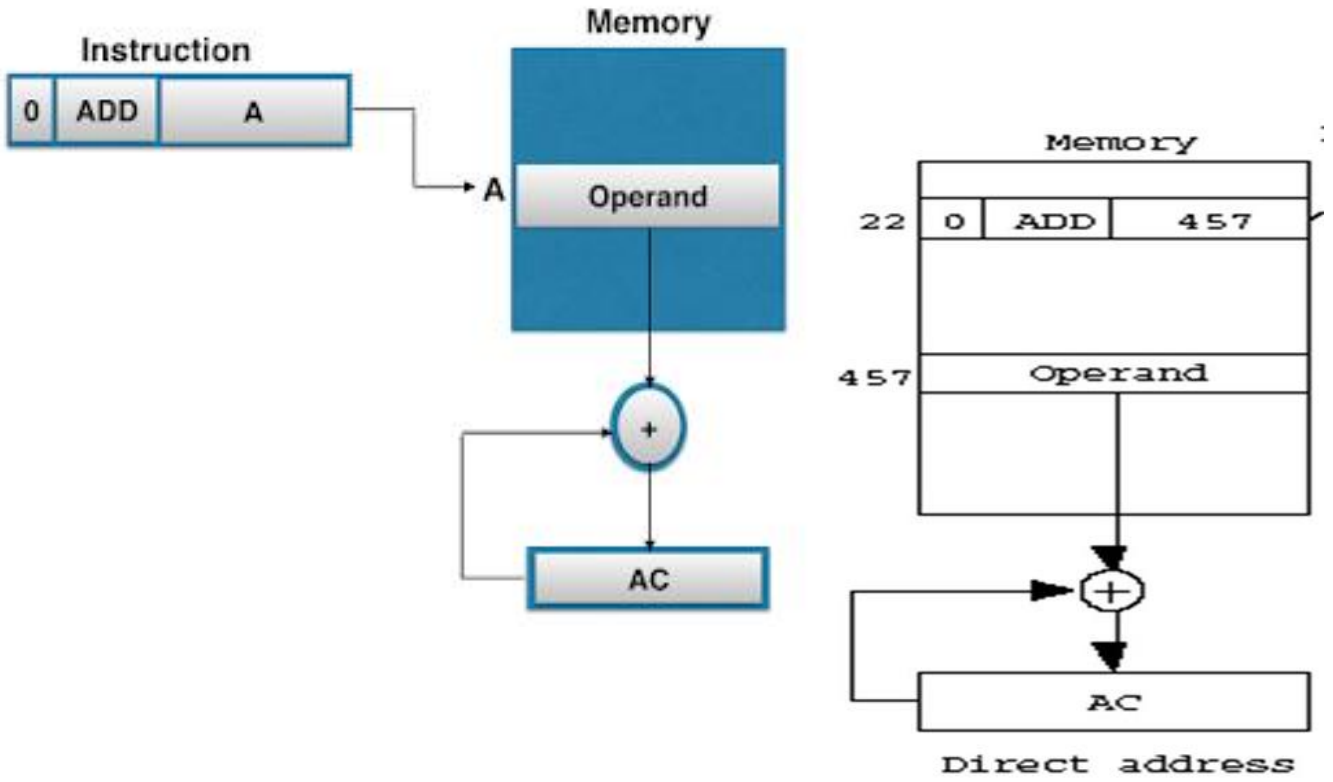
Direct Addressing Mode

It is also known as “Absolute Addressing Mode”.In this mode the address of data(operand) is specified in the instruction itself.That is, in this type of mode, the operand resides in memory and its address is given directly by the address field of the instruction.

For example: Consider the instruction:

ADD A - Means add contents of cell A to accumulator .

It Would look like as shown below:



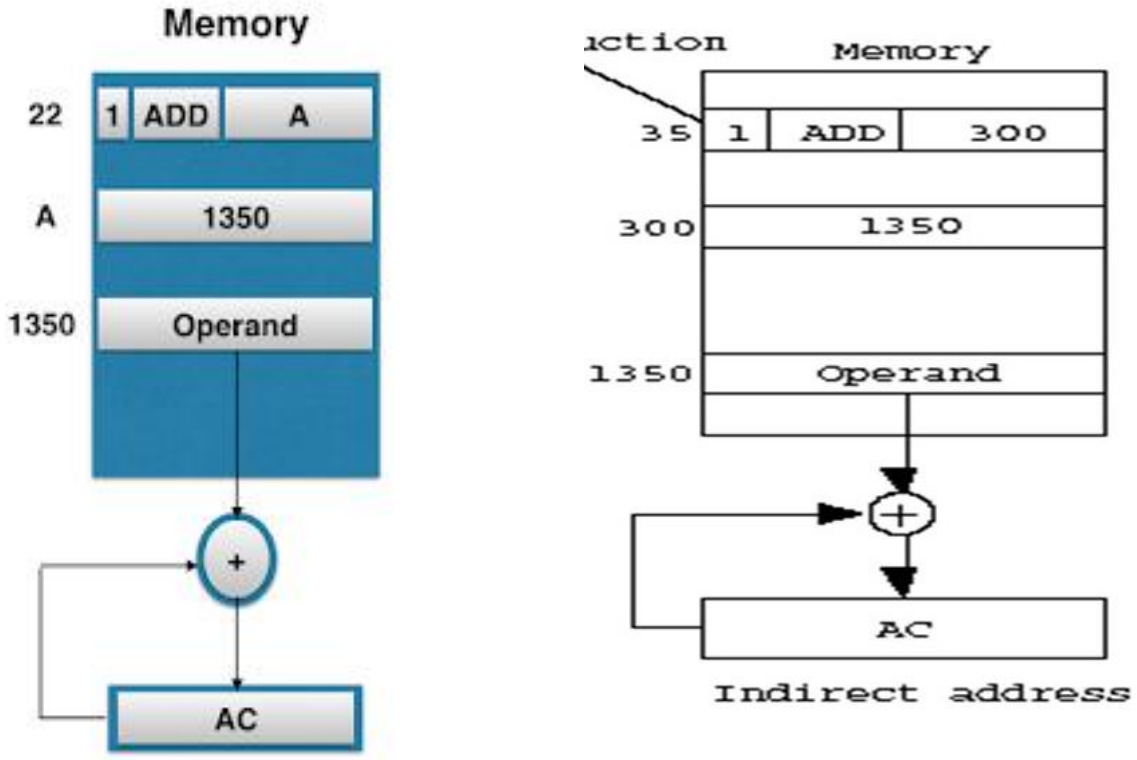
Indirect Addressing

In this mode, the address field of instruction gives the memory address where on, the operand is stored in memory.

For example: Consider the instruction:

ADD (A) Means adds the content of cell pointed to contents of A to Accumulator.

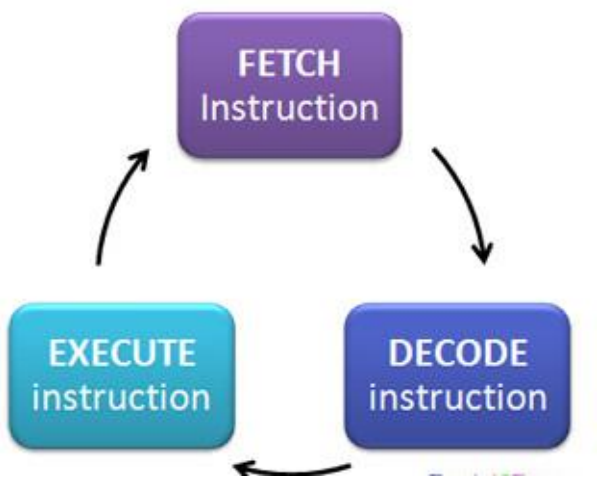
It look like as shown in diagram below:



11.4 Instruction Cycle

The program is executed in the computer by going through a cycle for each instruction.

Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases.



In the basic computer each instruction cycle consists or the following phases:

1. Fetch an instruction from memory.
2. Decode the instruction
3. Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered.

11.5 Summary

In this chapter you have learned about:

- Instruction Code
- Basic Computer Instructions
- Addressing
- Instruction Cycle

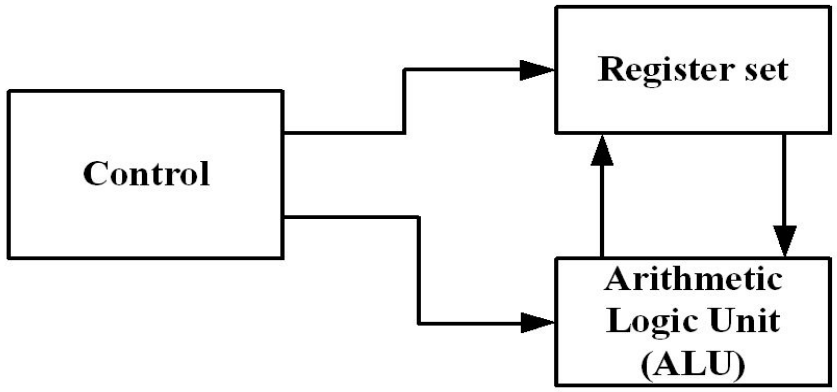
Unit 12: Central Processing Unit

Unit Structure

12.1 Introduction

The part of the computer that performs the bulk of data-processing operations is called the central processing unit and is referred to as the CPU.

The CPU is made up of three major parts, as shown in diagram.



Major components of CPU

The register set stores intermediate data used during the execution of the instructions.

The arithmetic logic unit (ALU) performs the required micro operations for executing the instructions.

The control unit supervises the transfer of information among the registers and instructs the ALU as to which operation to perform.

12.2 Stack Organization

A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved.

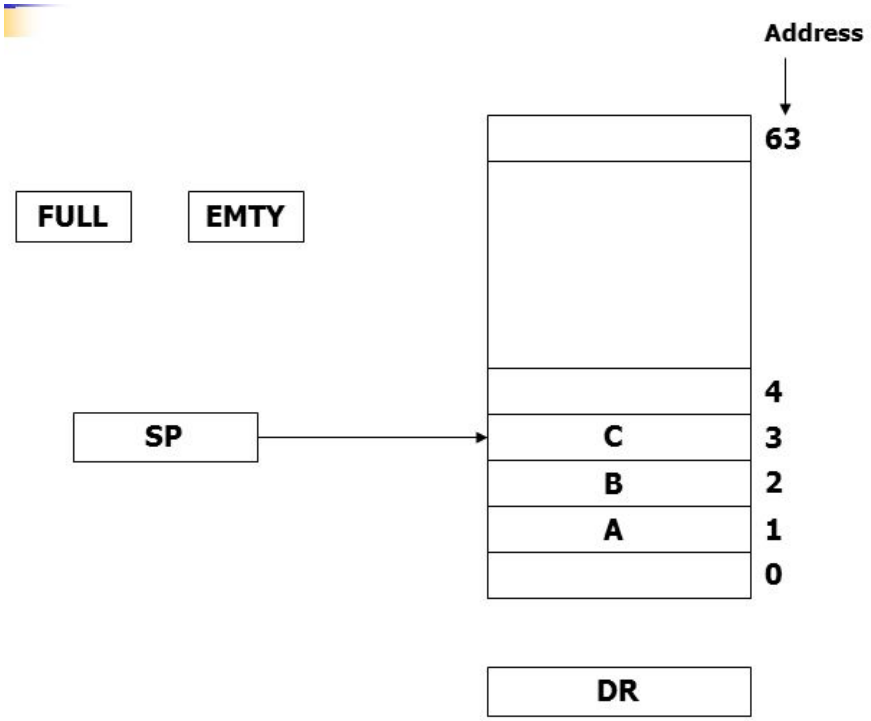
The register that holds the address for the stack is called a stack pointer (SP) because its value always points at the top item in the stack.

The two operations of a stack are the insertion and deletion of items. The operation of insertion is called push (or push-down) because it can be through of as the result of pushing a new item on top.

The operation of deletion is called pop (or pop-up) because it can be thought of as the result of removing one item so that the stack pops up.

Register Stack

A stack can be placed in portion of a large memory or it can be organized as a collection of a finite number of memory words or registers. Diagram shows the organization of a 64-word register stack.



Three items are placed in the stack: A, B, and C, in that order. Item C is on top of the stack so that the content of SP is now 3.

To remove the top item, the stack is popped by reading the memory word at address 3 and decrementing the content of SP. Item B is now on top the stack since SP holds address 2.

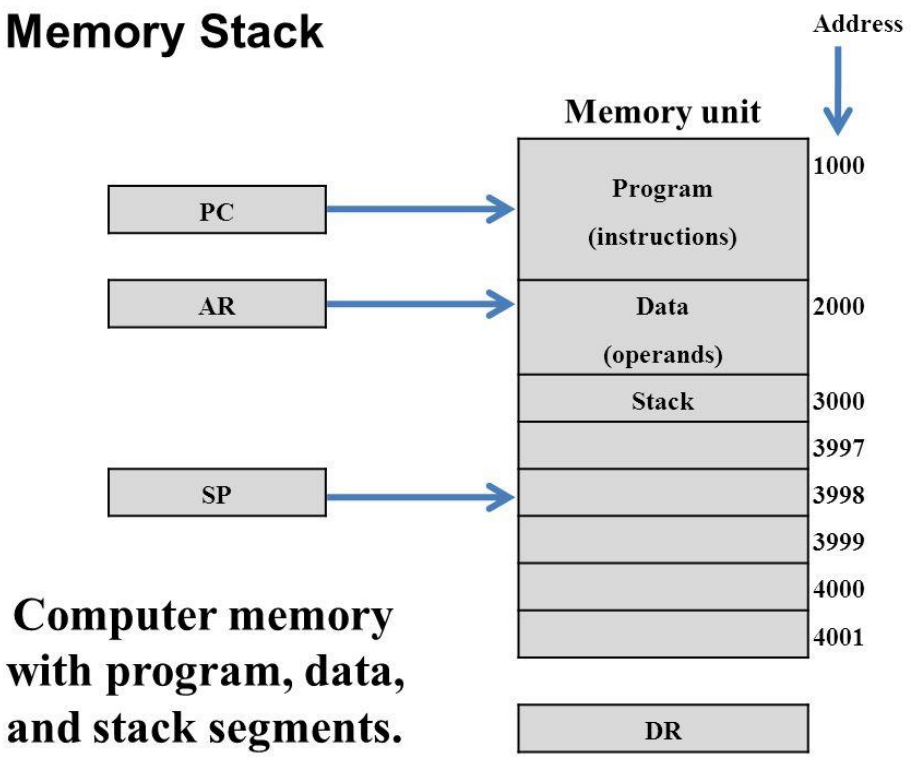
To insert a new item, the stack is pushed by incrementing SP and writing a word in the next-higher location in the stack.

Memory Stack

A stack can exist as a stand-alone unit or can be implemented in a random-access memory attached to a CPU.

The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer.

Diagram shows a portion of computer memory partitioned into three segments: program, data, and stack.



The program counter PC points at the address of the next instruction in the program.

The address register AR points at an array of data. The stack pointerSP points at the top of the stack.

PC is used during the fetch phase to read an instruction. AR is used during the execute phase to read an operand. SP is used to push or pop items into or from the stack.

The initial value of SP is 4001 and the stack grows with decreasing addresses. Thus the first item stored in the stack is at address 4000, the second item is stored at address 3999, and the last address that can be used for the stack is 3000.

We assume that the items in the stack communicate with a data register DR. A new item is inserted with the push operation as follow:

SP ← SP – 1
M [SP] ← DR

The stack pointer is decremented so that it points at the address of the next word. A memory write operation inserts the word form DR into the top of the stack. A new item is deleted with a pop operation as follows:

DR ←M[SP]
SP ← SP + 1

The top is read form the stack into DR. The stack pointer is then incremented to point at the next item in the stack.

A stack pointer is loaded with an initial value. This initial value must be the bottom address of an assigned stack in memory. So, SP is automatically decremented or incremented with every push or pop operation.

The advantage of a memory stack is that the CPU can refer to it without having to specify an address, since the address is always available and automatically updated in the stack pointer.

12.3 Addressing Modes

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Following are various addressing modes.

1. Implied Mode

In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction “complement accumulator” is an implied-mode instruction.

All register reference instructions that use an accumulator are implied-mode instructions.

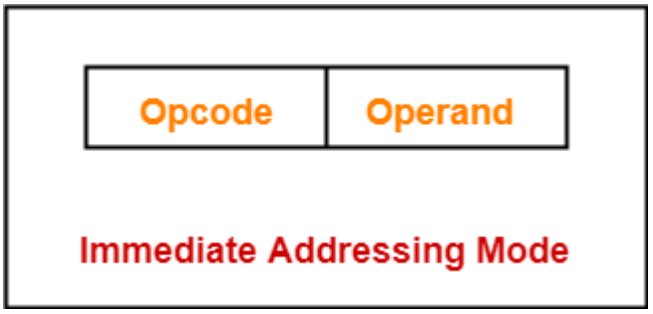
Zero-addressinstructions in a stack-organized computer are implied-mode instructions since the operands are implied to be on top of the stack.

2. Immediate Mode

In this mode the operand is specified in the instruction itself.

An immediate-mode instruction has an operand field rather than an address field.

The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.



Example:

ADD 10 will increment the value stored in the accumulator by 10.

MOV R #20 initializes register R to a constant value 20.

3. Register Mode

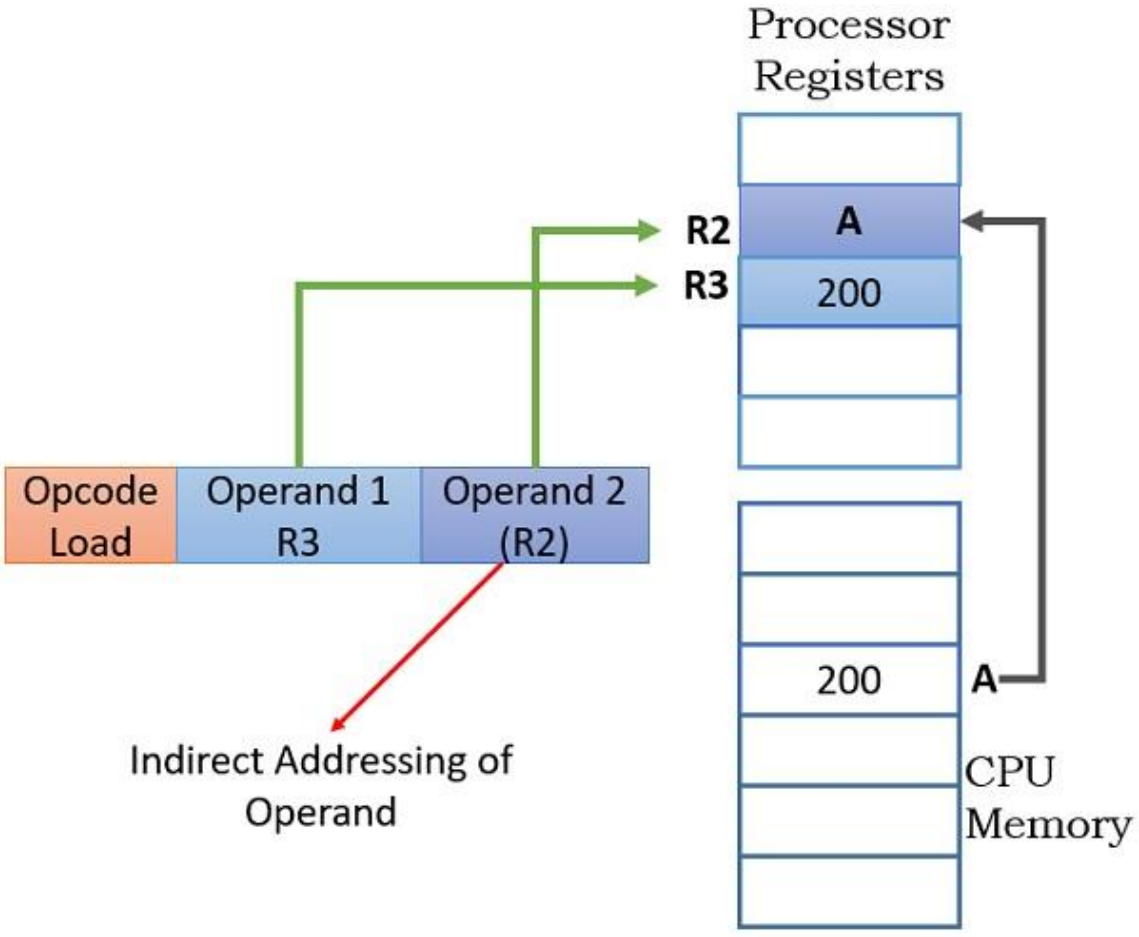
In this mode the operands are in registers that reside within the CPU.

The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2k registers.

4. Register Indirect Mode

In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

In other words, the selected register contains the address of the operand rather than the operand itself.



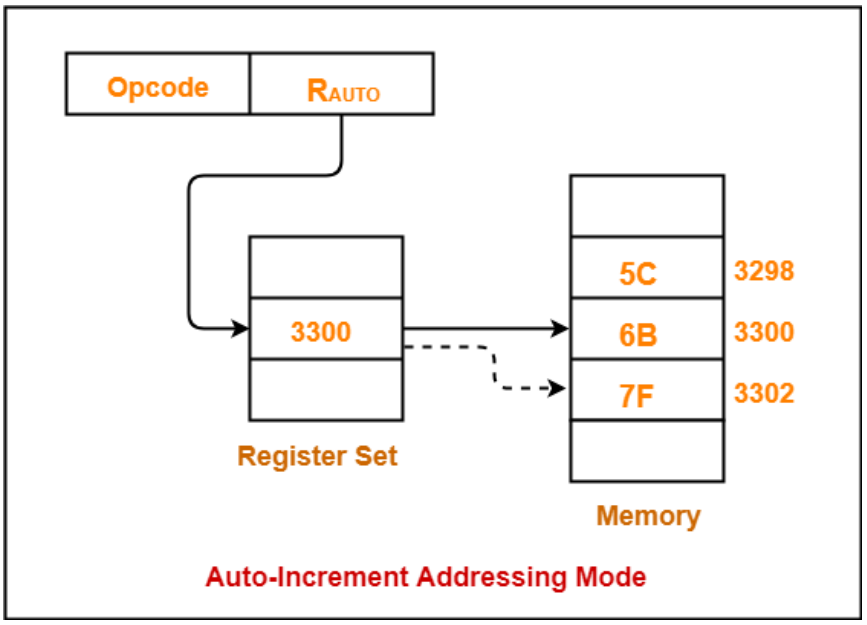
Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.

5. Auto increment Mode:

- The register is incremented or decremented after (or before) its value is used to access memory. When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table.

  Effective Address of the Operand
  = Content of Register

- This can be achieved by using the increment or decrement instruction or sometimes it is auto incremented.



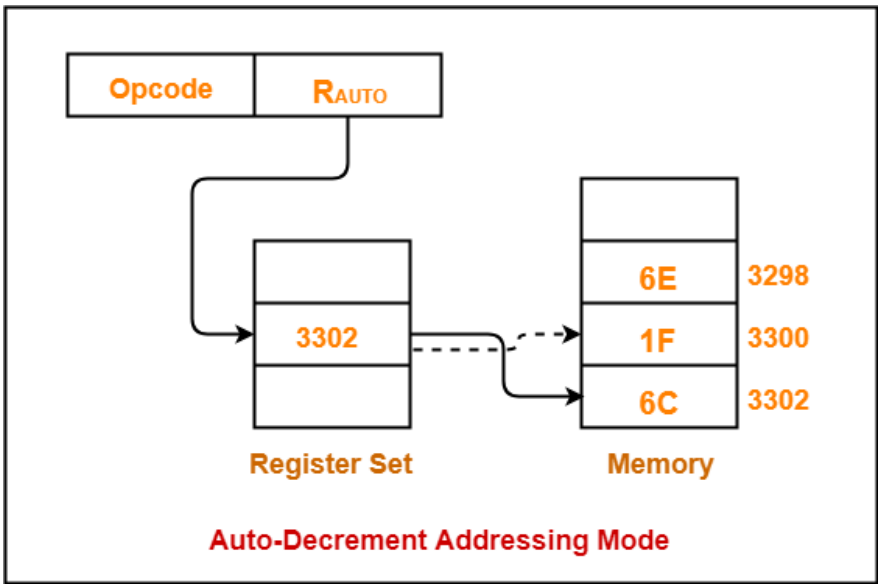
Assume operand size = 2 bytes.
Here,

- After fetching the operand 6B, the instruction register RAUTO will be automatically incremented by 2.
- Then, updated value of RAUTO will be 3300 + 2 = 3302.
- At memory address 3302, the next operand will be found.

6. Auto decrement

This addressing mode is again a special case of Register Indirect Addressing Mode where-

Effective Address of the Operand = Content of Register – Step Size



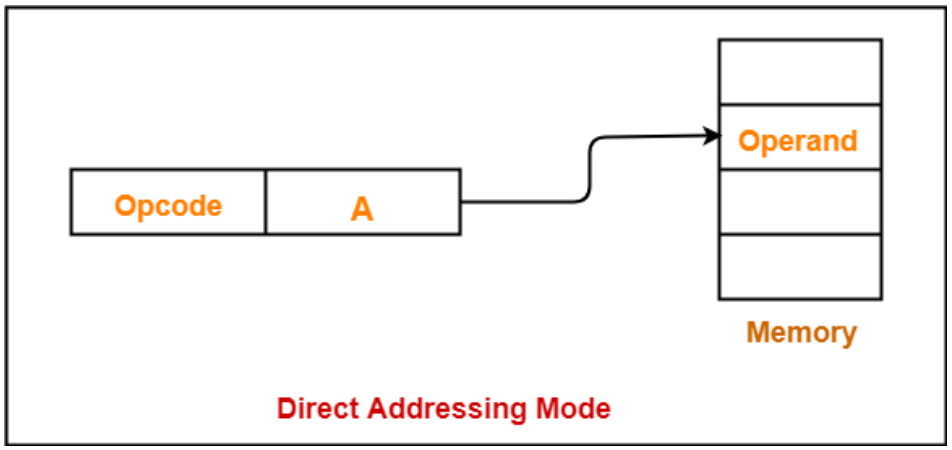
Assume operand size = 2 bytes.

Here,

- First, the instruction register RAUTO will be decremented by 2.
- Then, updated value of RAUTO will be 3302 – 2 = 3300.
- At memory address 3300, the operand will be found.

7. Direct Address Mode

In this mode the effective address is equal to the address part of the instruction.



The operand resides in memory and its address is given directly by the address field of the instruction. In a branch-type instruction the address field specifies the actual branch address.

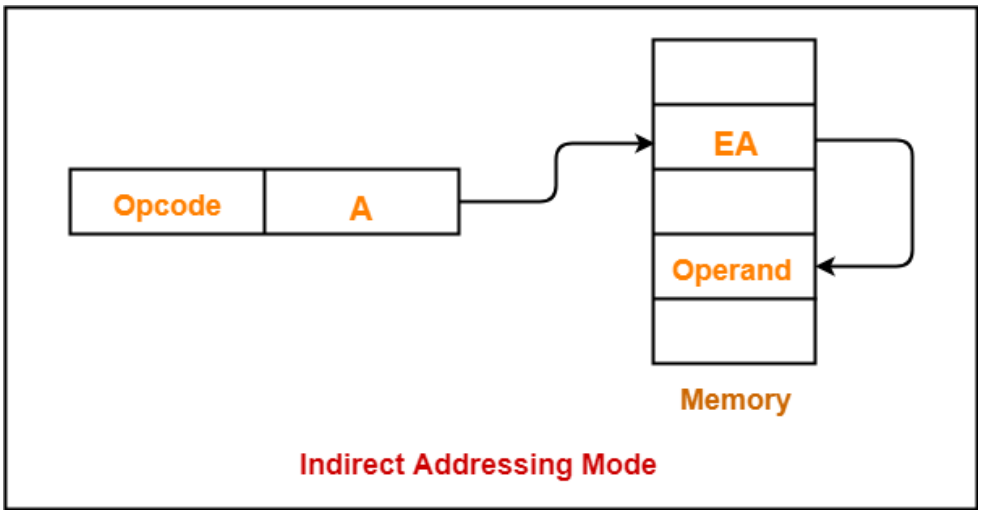
Example

ADD X will increment the value stored in the accumulator by the value stored at memory location X.

$$AC \leftarrow AC + [X]$$

8. Indirect Address Mode

In this mode the address field of the instruction gives the address where the effective address is stored in memory.



Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

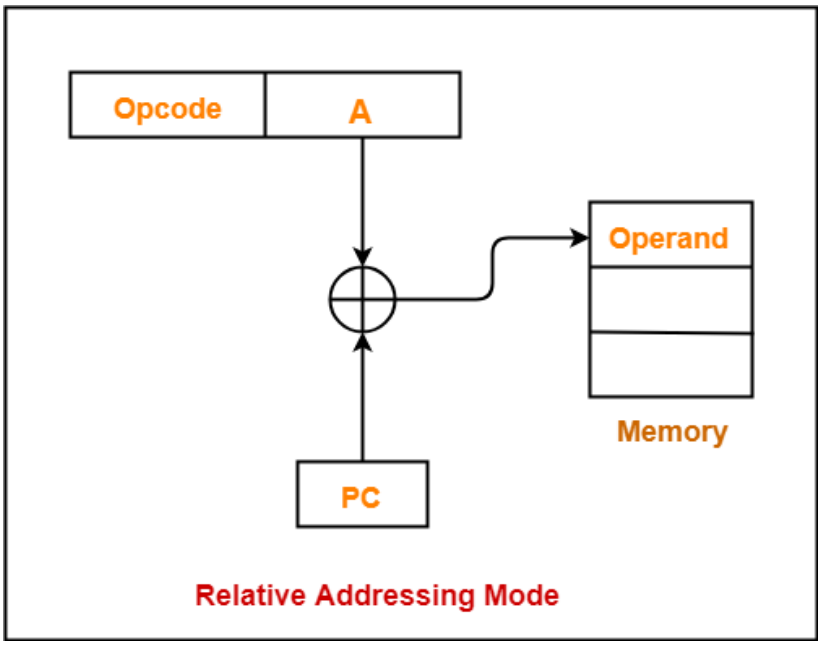
9. Relative Address Mode

In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction is usually a signed number (in 2's complement representation) which can be either positive or negative.

Effective Address

= Content of Program Counter + Address part of the instruction

When this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction.



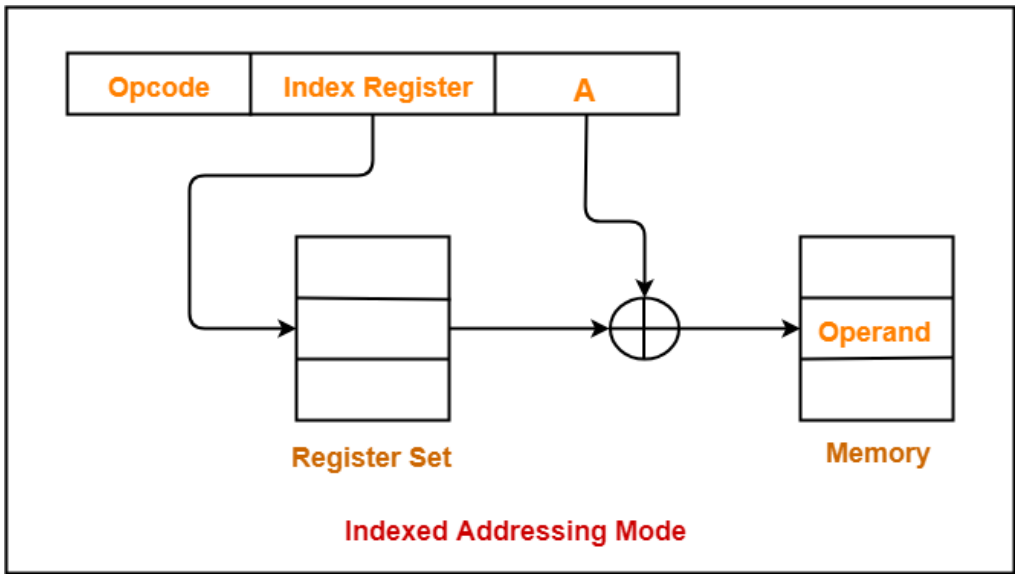
To clarify **with an example**, assume that the program counter contains the number 825 and the address part of the instruction contains the number 24. The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826 + 24 = 850. This is 24 memory locations forward from the address of the next instruction.

10. Indexed Addressing Mode

In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.

Effective Address

= Content of Index Register + Address part of the instruction



The index register is a special CPU register that contains an index value.

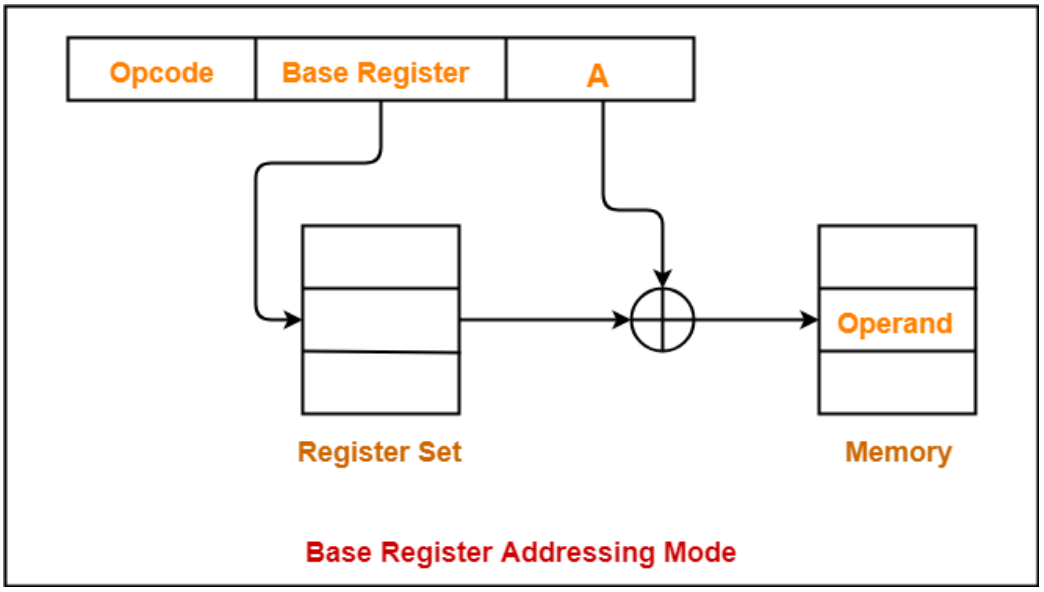
11. Base Register Addressing Mode:

In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

Effective Address

= Content of Base Register + Address part of the instruction

This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register.



The difference between the two modes is, an index register is assumed to hold an index number that is relative to the address part of the instruction. A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.

Applications of Addressing Modes

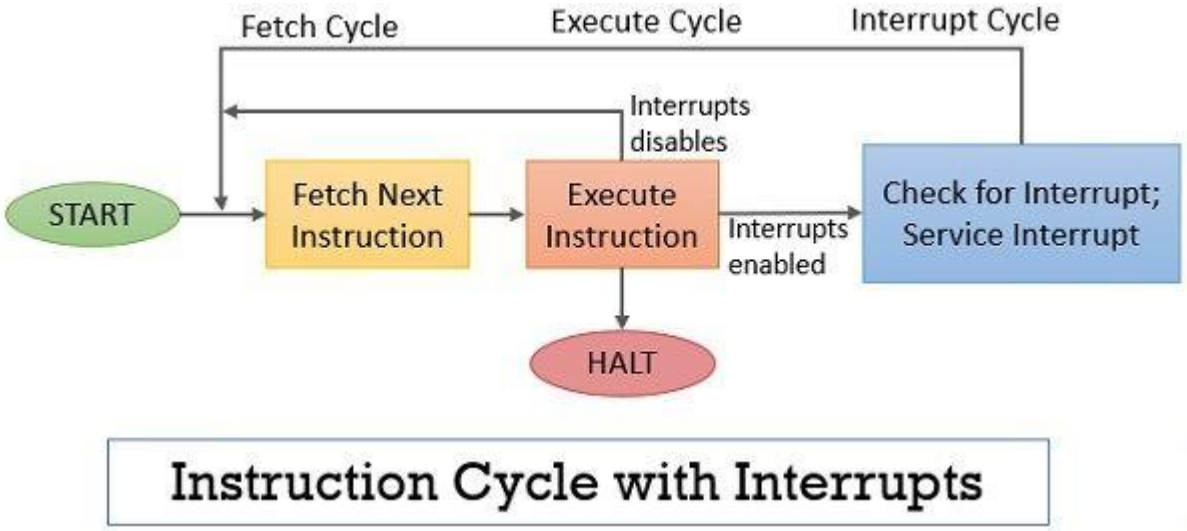
	Addressing Modes	Applications
1	**Immediate Addressing Mode**	• To initialize registers to a constant value
2	**Direct Addressing Mode and Register Direct Addressing Mode**	• To access static data • To implement variables
3	**Indirect Addressing Mode and Register Indirect Addressing Mode**	• To implement pointers because pointers are memory locations that store the address of another variable • To pass array as a parameter because array name is the base address and pointer is needed to point the address
4	**Relative Addressing Mode**	• For program relocation at run time i.e. for position independent code

		• To change the normal sequence of execution of instructions • For branch type instructions since it directly updates the program counter
5	**Index Addressing Mode**	• For array implementation or array addressing • For records implementation
6	**Base Register Addressing Mode**	• For writing relocatable code i.e. for relocation of program in memory even at run time • For handling recursive procedures
7	**Auto-increment Addressing Mode and Auto-decrement Addressing Mode**	• For implementing loops • For stepping through arrays in a loop • For implementing a stack as push and pop

12.4 Program Interrupt

An event external to the currently executing process that causes a change in the normal flow of instruction execution; usually generated by hardware devices external to the CPU.

After a program has been interrupted and the service routine been executed, the CPU must return to exactly the same state that it was when the interrupt occurred.



Instruction Cycle with Interrupts

The state of the CPU at the end of the execute cycle is determined from

1. The content of the program counter
2. The content of all processor register
3. The content of status conditions

The collection of all status bit conditions in the CPU is called **program status word or PSW.** The PSW is stored in separate register, it includes the status bits from the last ALU operation and it specifies the interrupts that are allowed to occur.

When CPU is executing a program that is part of the operating system, it is said to be a **supervisor or system mode.**

Types of Interrupt

Generally there are three types o Interrupts those are Occurred.

1. Internal Interrupt
2. Software Interrupt
3. External Interrupt

External Interrupt

These types of interrupts generally come from external input / output devices which are connected externally to the processor.

For Example When a Program is executed and when we move the Mouse on the Screen then the CPU will handle this External interrupt first and after that he will resume with his Operation.

Internal Interrupt

The Internal Interrupts are those which are occurred due to some Problem in the Execution They are also known as traps and their causes could be due to some illegal operation or the erroneous use of data.

For Example When a user performing any Operation which contains any type of Error.

Software Interrupt

The interrupts which are caused by the software instructions are called software interrupts

12.5 Summary

In this chapter you have learned about:

- Introduction
- Stack Organization
- Addressing Modes
- Program Interrupt

Dr. Babasaheb Ambedkar Open University

BCAR-203

Digital Electronics and Computer Organisation

BLOCK4: ORGANIZATION OF INPUT- OUTPUT AND MEMORY

BLOCK 4: ORGANIZATION OF INPUT-OUTPUT AND MEMORY

Block Introduction

In this block-4 of digital electronics, I have tried to emphasis on: input-output organization and memory organization. Basically, I introduced concept various peripheral devices which are external to CPU and how communication with the CPU can be possible and various types of memory used in computer for various purpose.

I hace also emphasis on direct memory access concept in which CPU will be idle during the data transfer between peripheral device and memory.

Block Objective

The objective of the block is to explain concept of data transfer between CPU and peripheral devices and also transfer of data directly with memory.I have also explain concept about various other memories which are having different speed and storage capacity.

By learning this block of digital electronics student will also learn about more details of memory and peripheral devices . Reader of this block, will know how data transfer happen between CPU and memory and which are the flags used for storing thr between conditions .

Unit 13: Input-Output Organization

Unit Structure

13.1Peripheral devices

Input or output devices attached to the computer are also called peripherals. Among the most common peripherals are keyboards, display units, and printers. Peripherals that provide auxiliary storage for the system are magnetic disks and tapes.

Video monitors are the most commonly used peripherals. They consist of a keyboard as the input device and a display unit as the output device. There are different types of video monitors, but the most popular use a cathode ray tube (CRT).

Printers provide a permanent record on paper of computer output data or text. There are three basic types of character printers: daisywheel, dot matrix, and laser printers.

The daisywheel printer contains a wheel with the characters placed along the circumference. To print a character, the wheel rotates to the proper position and an energized magnet then presses the letter against the ribbon.

The dot matrix printer contains a set of dots along the printing mechanism.

The laser printer uses a rotating photographic drum that is used to imprint the character images. The pattern is then transferred onto paper in the same manner as a copying machine.

ASCII Alphanumeric Characters

Input and output devices that communicate with people and the computer are usually involved in the transfer of alphanumeric information to and from the device and the computer is ASCII (American Standard Code for Information Interchange) .It uses seven bits to code 128 characters.

13.2 Input-Output Interface

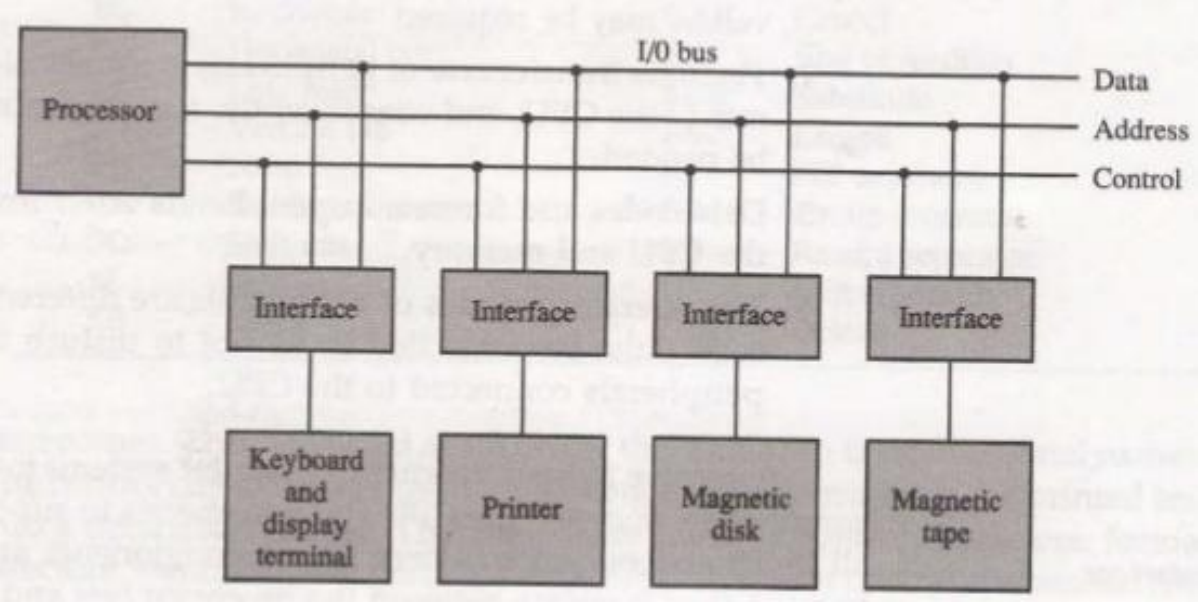
Input-output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.

Computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.

In addition, each device may have its own controller that supervises the operations of the particular mechanism in the peripheral.

I/O Bus and Interface Modules

Communication link between the processor and several peripherals is shown in following diagram.



The I/O bus consists of data lines, address lines, and control lines. The magnetic disk, printer, and terminal are employed in practically any general-purpose computer. The magnetic tape is used in some computers for backup storage. Each peripheral device has associated with it an interface unit.

Each interface decodes the address and control received from the I/O bus, interprets them for the peripheral, and provides signals for the peripheral controller. It also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller that operates the particular electromechanical device.

At the same time that the address is made available in the address lines, the processor provides a function code in the control lines. The interface selected responds to the function code and proceeds to execute it. The function code is referred to as an I/O command

13.3 DMA

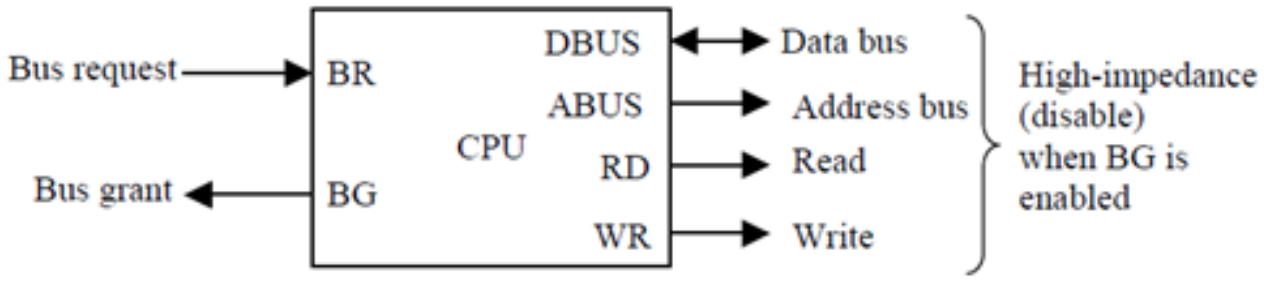
Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA).

During DMA transfer, the CPU is idle and has no control of the memory buses. A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

The CPU may be placed in an idle state in a variety of ways. One common method extensively used in microprocessors is to disable the buses through special control signals.

Diagram shows two control signals in the CPU that facilitate the DMA transfer.

The bus request (BR) input is used by the DMA controller to request the CPU to



relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state. behaves like an open circuit, which means that the output is disconnected and does not have a logic significance.

The CPU activates the Bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

13.4Summary

In this chapter you have learned about:

- Peripheral devices
- Input-Output Interface
- DMA

Unit 14: MEMORY ORGANIZATION

Unit Structure

14.1 Memory Hierarchy

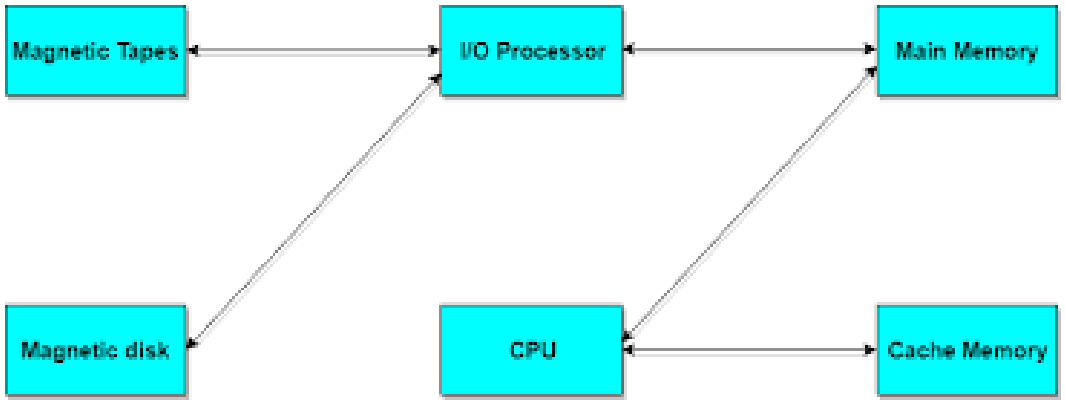
The memory hierarchy consists of all the storage devices in the computer from the slow but high capacity auxiliary memory to a fast but low capacity cache memory.

The **magnetictape** used to store removable files. The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files, and other backup information. The capacity is larger but speed is very slow.

The **main memory** interacts with CPU. The data in the auxiliary memory is brought in the main memory.

Only programs and data currently needed by the processor reside in main memory. All other information is stored in auxiliary memory and transferred to main memory when needed.

A special very-high speed memory called a cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate



Many operating systems are designed to enable the CPU to process a number of independent programs concurrently. This concept, **called multiprogramming.**

It refers to the existence of two or more programs indifferent parts of the memory hierarchy at the same time. In this way it is possible to keep all parts of the computer busy by working with several programs in sequence

11.2 Main Memory

The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation.

The principal technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.

The static RAM consists essentially of internal flip-flops that store the binary information. The stored information remains valid as long as power is applied to unit.

The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.

Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips.

RAM is used for storing the bulk of the programs and data that are subject to change. ROM is used for storing programs that are permanently resident.

The ROM portion of main memory is needed for storing an initial program called a bootstrap loader. The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.

The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the operating system, which prepares the computer for general use.

11.3 Auxiliary Memory

The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. Other components used, but not as frequently, are magnetic drums, magnetic bubble memory, and optical disks.

Magnetic Disks

A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material. Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.

All disks rotate together at high speed and are not stopped or started from access purposes. Bits are stored in the magnetized surface in spots along concentric circles called tracks.

The tracks are commonly divided into sections called sectors. In most systems, the minimum quantity of information which can be transferred is a sector. The sub division of tone disk surface into tracks and sectors.

Magnetic Tape

A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit.

The tape itself is a strip of plastic coated with a magnetic recording medium. Bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.

Read/write heads are mounted one in each track so that data can be recorded and read as a sequence of characters.

Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewound. However, they cannot be started or stopped fast enough between individual characters.

For this reason, information is recorded in blocks referred to as records. Gaps of unrecorded tape are inserted between records where the tape can be stopped.

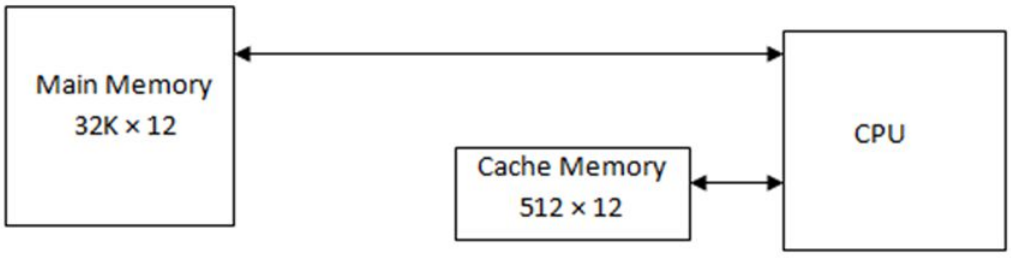
14.4 Cache Memory

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory.

Cache memory is an intermediate buffer between CPU main memory. The objective is to reduce CPU waiting time during main memory access.

Example

A system with 512 x 12 cache and 32 K x 12 of main memory.



The cache memory access time is less than the access time of main memory The cache is the fastest component in the memory hierarchy The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory.

Basic operation of the cache

When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word. A block of words containing the one just accessed is then transferred from main memory to cache memory.

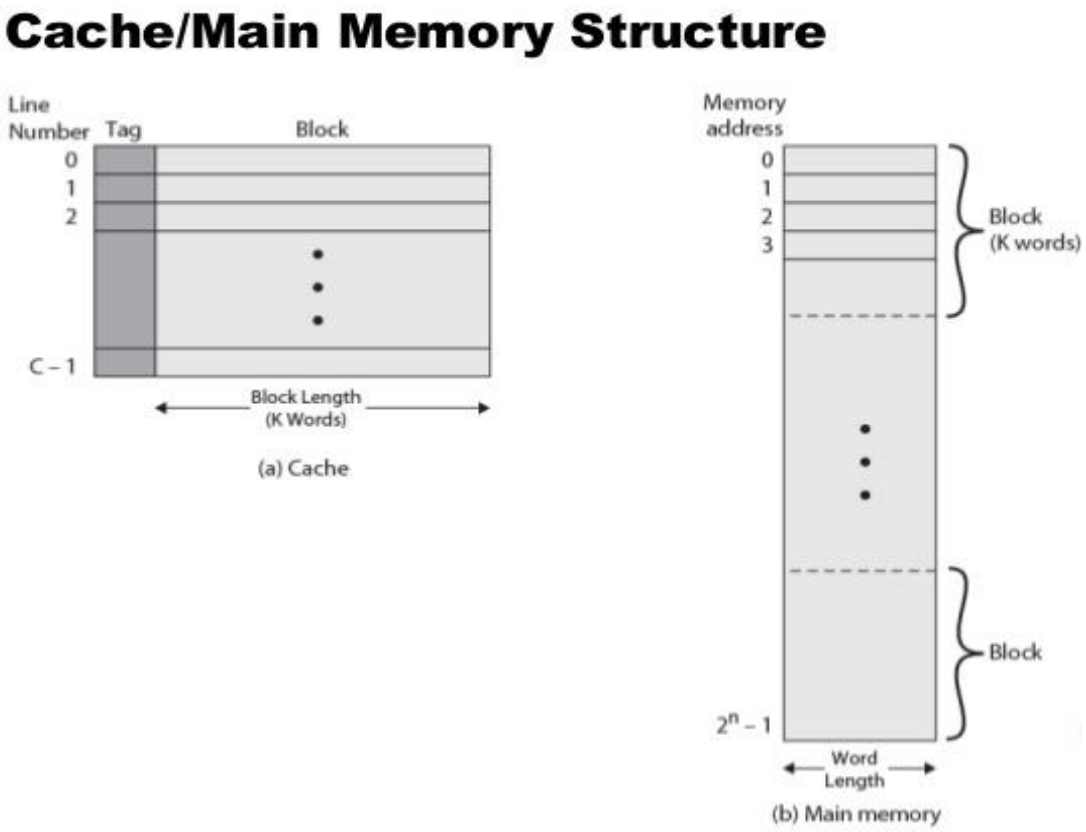
Hit Ratio

The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When the CPU refers to memory and finds the word in cache, it is said to produce a hit.

If the word is not found in cache, it is in main memory and it counts as a miss.

The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio

Structure of cache



The main memory is conceptually divided into many blocks. Each containing a fixed number of consecutive locations.

While reading a location from main memory the content of entire block is transferred & stored in cache memory.

The cache memory is organized as number of lines or blocks & size of each line is same as the capacity of main memory.

14.5 Virtual Memory

It is a concept that helps on run a large program in a small physical memory.

Each address that is referenced by the CPU goes through an address mapping from the so called virtual address to a physical address in main memory.

Virtual memory is used to give programmers the illusion that they have a very large memory, even though the computer actually has a relatively small main memory.

This is done dynamically, while programs are being executed in the CPU. The translation or mapping is handled automatically by the hardware by means of a mapping table.

14.6 Summary

In this chapter you have learned about:

- Memory Hierarchy
- Main Memory
- Auxiliary Memory
- Cache Memory
- Virtual Memory