COMPUTER SCIENCE

Subject Code 9608

PAPER - 3

A Level

COMPUTER SCIENCE

Paper-3

Article # 253

A LEVEL NOTES SERIES

CIE SYLLABUS CODE 9608

2017 – 18 Edition

PREFACE

This is a comprehensive and carefully balanced compilation of all the relevant topics that need to be essentially covered and understood by any O Level candidate who wishes to ace his Chemistry paper. We have managed to adopt a purely focused and goal oriented approach in this context that would enable students not only to grasp the content but also assist them in analysing and evaluating individual components.

Table of Contents

Section 3 - Advance Theory

Data Representation – Real Number and Normalised Floating Point

Representing Real Numbers

- ✓ Real numbers contain a decimal or an exponent or both to store the result of a division or an average.
  - For Example, 35, $4.8 * 10^{-2}$, $5 * 10^{7}$
- ✓ There are two main types of real numbers:
  - Fixed point -> Contains a decimal but no exponent
  - Floating Point contains an exponent. It may or may not have a decimal.
  - E.g. $5.1 * 10^{3}$, $10 * 10^{-3}$

Binary to Fixed Point

TIP

✓ 2^{-1}	= 0.5	2^{-2}	= 0.25		
✓ 2^{-3}	= 0.125	2^{-4}	= 0.0625	2^{-5}	= 0.03125

- Simply put the weight of base 2 over the numbers after the power 0.

- E.g. $(1101.101)_2 = (13.625)_{10}$

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	1	0	1	1	0	1

$= 2^3 + 2^2 + 2^{-1} + 2^{-3}$

$= 8 + 4 + 0.5 + 0.125$

$= (13.625)_{10}$

Fixed Point to binary

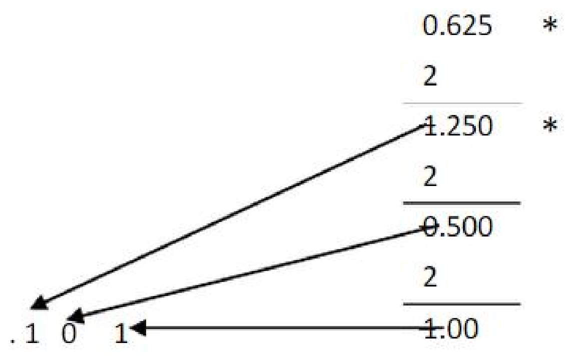
To convert base 10 to fixed point:

1. Convert integer part to binary
2. Convert decimal part to binary (repetitive multiplication by 2 and cut digit before decimal place till number is 0)

e.g. $(22.625)_{10} = (10110.101)_2$

Step 1: 22 -> 10110

Step 2:



Hence, $(22.625)_{10} = (10110.101)_2$

Floating Point Number

Floating point numbers are used to represent very large numbers or very small numbers. This is why they are often used for scientific calculations. But the problems of floating point numbers are that they are not **accurate** enough.

A floating point number essentially consist of 3 parts:

1. A Sign
2. A Mantissa
3. An Exponent

$$+/- m * r^e$$

Where:

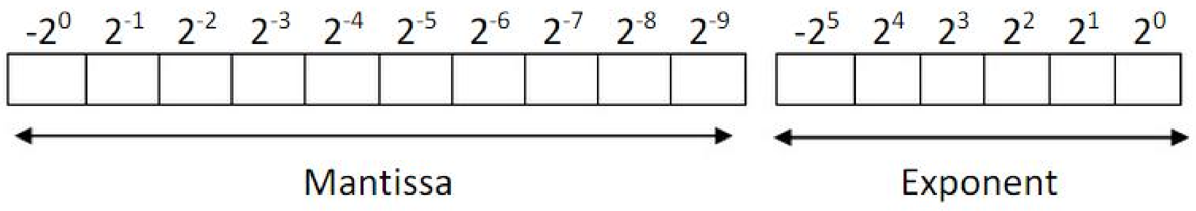
+/ -	-> sign
M	-> Mantissa
e	-> Exponent
r	-> Radix (Base)

Ex1: A floating point number in base 10: $4.75 * 10^3$	**Ex2 : A floating point number in base 2 :** $0.1101 * 2^3$
▪ Sign = +	▪ Sign = +
▪ Mantissa = 4.75	▪ Mantissa = 0.1101
▪ Radix = 10	▪ Radix = 2
▪ Exponent = 3	▪ Exponent = 3

Consider the following Representation for a floating point number

16 bits to represent a floating point number where 10 bits is reserved for 2's complement mantissa and 6 bits for 2's complement exponent

Positive Mantissa and Positive Exponent



E.g. Let us consider a floating point: 0.111 010 000 000 101

Mantissa										Exponent					
-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	-2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	1	0	1	0	0	0	0	0	0	0	1	0	1

Mantissa = 0.11101 $= (2^{-1} + 2^{-2} + 2^{-3} + 2^{-5})$

Exponent = 000 101 $= 2^2 + 2^0 = 5$

Floating Point = Mantissa * 2^{exponent}

$= (2^{-1} + 2^{-2} + 2^{-3} + 2^{-5}) * 2^5$

= 29

Positive Mantissa and Negative Exponent

E.g. Let us consider a floating point: **0**.110 110 000 **1**11 111

Floating point > 0 because mantissa is positive. However, the number will be a fraction because the exponent is negative

Mantissa										Exponent					
-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	-2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	0	1	1	0	0	0	0	1	1	1	1	1	1

Mantissa = 0.11011 = $(2^{-1} + 2^{-2} + 2^{-4} + 2^{-5})$

Exponent = 111 111 = $-2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -1$

Floating Point = Mantissa * 2^{exponent}

$= (2^{-1} + 2^{-2} + 2^{-4} + 2^{-5}) * 2^{-1}$

$= (2^{-2} + 2^{-3} + 2^{-5} + 2^{-6}) = (1/4 + 1/8 + 1/32 + 1/64) = (16 + 8 + 2 + 1)/64 =$
$27/64$

Negative Mantissa and Positive Exponent

E.g. Let us consider a floating point: 1.011 000 000 000 011

Floating Point is negative because mantissa is negative

Mantissa										Exponent					
-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	-2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Mantissa = 1.011 = $(-2^0 + 2^{-2} + 2^{-3})$

Exponent = 000 011 = $2^1 + 2^0 = 3$

Floating Point = Mantissa * 2^{exponent}

$= (-2^0 + 2^{-2} + 2^{-3}) * 2^3$

$= (-2^3 + 2^1 + 2^0) = -5$

Consider the following example

A particular computer uses two 8-bit bytes to store floating-point values. One byte is used to store the mantissa and the other is used to store the exponent.

- Largest Positive value = 0111 1111 0111 1111
- Smallest Magnitude Negative number = 1111 1111 1000 0000

Normalisation

- ✓ The reason for normalising the mantissa is in order to hold numbers to as high a degree of accuracy as possible. The exponent is always an integer and is represented in two's complement.

Note:

- ✓ When a negative number is normalised, care needs to be taken.

Procedure

- ➢ The normalised mantissa of positive number is written first.
- ➢ The two's complement of the mantissa is then written

Let us consider the following three floating point number

A. 0.100 000 000 | 000 010
B. 0.010 000 000 | 000 011
C. 0.001 000 000 | 000 100

I. F.P = $(2^{-1}) * 2^{2} = 2$
II. F.P = $(2^{-2}) * 2^{3} = 2$
III. F.P = $(2^{-3}) * 2^{4} = 2$

Note: For the same value, there are different floating point number representation available.

Normalising Floating Point Number

Normalise Positive Floating Point Number

- ➢ It must have no leading 0's after the decimal

Normalise Negative Floating Point Number

- ➢ It must have no leading 1's after the decimal

E.g. State whether the following FP are normalised or not.

I. 0.100 100 000 | 000 011 – Normalised
II. 0.100 100 000 | 111 111 – Normalised
III. 0.010 000 000 | 111 111 – Not Normalised

Range V/S Accuracy

There are always a finite number of bits that can be used to represent numbers in a computer. This means that if we use more bits for the mantissa we will have to use fewer bits for the exponent.

Range

- Range is the difference between the largest and smallest number that can be represented.

Range $\propto$ Number of bits in **Exponent**

Accuracy

Accuracy is the number of significant figures gives the accuracy of a number in 3 S.F is 10 times more accurate than 2.S.F. Hence, a measurement of 3.51 cm is 10 times more accurate than 3.5 cm.

Accuracy $\propto$ Number of bits in **Mantissa**

Advantage of using Floating Point compared to integers

- ✓ The range of number is larger. Very large and very small number can be represented

Disadvantage

- ✓ Not Accurate Enough

Communication and Internet Technologies - Protocols

Protocols

A protocol is the set of rules that govern the exchange of *data*. A protocol ensures which party is able to transmit and when. The protocol deals with the sequencing of messages and ensures an orderly recovery in the event of failure.

At the very least, a communications protocol must define the following:

- ✓ Rate of transmission (in baud or bps).
- ✓ Whether transmission is to be synchronous or asynchronous.
- ✓ Whether data is to be transmitted in half-duplex or full-duplex mode.

Network Reference Model

A network reference model serves as a blueprint, dictating how network communication should occur. Programmers and engineers design products that adhere to these models, allowing products from multiple manufacturers to interoperate.

The two most widely recognized network reference models are:

- ✓ The **Open Systems Interconnection (OSI)** model
- ✓ The **Department of Defence (DoD)** model

The OSI model was the first true network model, and consisted of seven layers. However, the OSI model has become deprecated over time, replaced with more practical models like the TCP/IP (or DOD) reference model.

Why a Layered Network Model?

- Reduces complexity
- Standardises interfaces
- Facilitates modular engineering
- Ensures interoperable technology
- Accelerates evolution
- Simplifies teaching and learning

OSI Reference Model (Open Systems Interconnection)

This model was developed by ISO (International Organisation for Standardisation) to provide a framework governing how information is sent across a network. Various mnemonics have been devised to help people remember the order of the OSI model's layers.

7	Application	**Devices**	**A**ll	↓	**A**way	↑
6	Presentation		**P**eople		**P**izza	
5	Session		**S**eem		**S**ausage	
4	Transport		**T**o		**T**hrow	
3	Network	Routers	**N**eed		**N**ot	
2	Data Link (LLC/MAC)	Bridge, Switch	**D**ata		**D**o	
1	Physical	Hubs, Repeaters	**P**rocessing		**P**lease	

The top 3 layers are known as Upper Layers.

Layer	Layer Name	Responsible For	Description
1	Physical Layer	Binary Transmission	▪ Defines the electrical, mechanical, procedural and functional specifications for activating, maintaining and deactivating the physical link.
2	Data Link Layer	Access to Media	▪ Defines how data is formatted for transmission and how access to network is controlled. ▪ It also provides error detection.
3	Network Layer	Data Delivery	▪ It routes data packets, selects best path to deliver data and provides logical addressing and path selection.
4	Transport Layer	End to End Connections	▪ Handles transportation issues between hosts. ▪ Ensures data transport reliability , establishes, maintains and terminates virtual circuits. ▪ Provides reliability through fault detection and recovery information flow control.
5	**Session Layer**	Interhost Communication	▪ Establishes, manages and terminates sessions between applications.
6	**Presentation Layer**	Data Representation	▪ Ensures that data is readable by recipient system. ▪ The layer formats data, structures data and negotiates data transfer syntax for application layer. ▪ It also provides encryption.

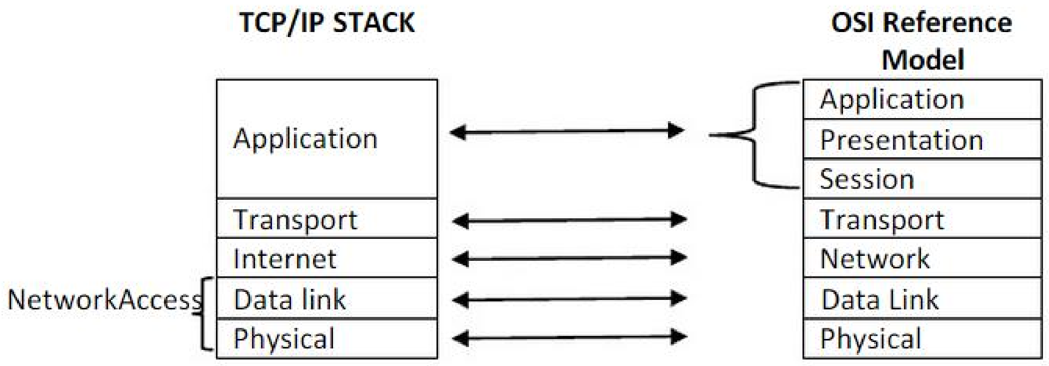
7	**Application Layer**	Network Processes to Application	▪ Provides network services to application processes. ▪ Provides user authentication.

A protocol model provides a model that closely matches the structure of a particular protocol suite.

TCP/IP STACK

TCP/IP is now the most widely used protocol due to its flexible addressing scheme , its usability by most operating systems and platforms, its many tools and utilities , and the need to use it to connect to the Internet.

The components of the TCP/IP stack are the network access ,Internet , transport and application layers. It is defined in 4 layers (**NITA**). It uses different names for Layers 1 through 3. It combines layers 5 through 7 into single application layer.



Role and Function of Each Layer

1. **Application** - provides applications for network troubleshooting, file transfer, Internet activities and support network application programming interfaces (APIs) that allows programs that have been created for a particular OS to access the network. It also represents data to the user plus encoding and dialog control

2. **Transport** - provides communication services directly to the application processes running on network hosts.

3. **Internet** - provides routing of data from source to destination by defining packet and the addressing scheme to move data between the data link and transport layers. It also

carries out routing packets of data to remote hosts and performing fragmentation and reassembly of data packets. It determines the best path through the network.

4. **Network Access** - controls the hardware devices and media that make up the network.

Common Protocols

HTTP

It is the short form for Hypertext Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an Http command to the **Web server** directing it to fetch and transmit the requested **Web page.**

FTP

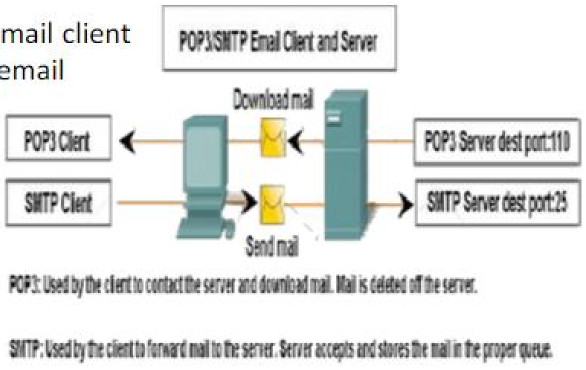
FTP is the short form for File Transfer Protocol, the protocol for exchanging files over the internet. FTP works in the same way as HTTP for transferring Web pages from a server to a user's browser. FTP uses the internet's TCP/IP protocols to enable data transfer. FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server (e.g., uploading a Web page file to a server).

SMTP

SMTP is the short form for Simple Mail Transfer Protocol, a protocol for sending email messages between servers. Most email systems that send mail over the internet use SMTP to send messages from one server to another; the messages can then be retrieved with an email client. In addition, SMTP is generally used to send messages from a mail client to a mail server.

POP3 (Post Office Protocol Version 3)

It is an application layer internet protocol used by email client software such as Outlook, Thunderbird to retrieve email from a remote server using TCP/IP connection. There is the possibility to download the email locally and delete them from server and disconnect but it also allows the user to leave the message on server itself. It encrypts the communication channel using TLS and SSL protocols between the servers.


POP3/SMTP Email Client and Server
Download mail
POP3 Client
POP3 Server dest port:110
SMTP Client
SMTP Server dest port:25
Send mail
POP3: Used by the client to contact the server and download mail. Mail is deleted off the server.
SMTP: Used by the client to forward mail to the server. Server accepts and stores the mail in the proper queue.

Bit Torrent

It is a protocol makes use of peer to peer file sharing is used to distribute large amount of data over the Internet. Bit Torrent protocol reduces the load of server and network since user do not have to download a file from a single server but rather users join a swarm and download chunk of files from each other (peers). Since users download from each other computer, the bandwidth usage becomes low and help prevent large spikes in traffic.

1. BitTorrent protocol has a central server called a tracker which keeps tracks of the users connected in the swarm. The tracker keeps track of user sharing files (Seeders) and those not sharing files with the one not sharing (Leechers). The more the user share, the faster their download is.
2. Using a peer to peer file sharing software a request is sent for the file you require to the tracker. BitTorrent search for the complete file (Seed computers) and peers downloading portion of the file. The tracker then identifies a swarm that has all the files and which are in the process of sending or receiving it.
3. The tracker then gives the client software pieces of file they want in the swarm and the computer receives several pieces of the file simultaneously.
4. After the user file download is completed, the client seeds to other users and future download rates improved as seeder uploads and share the files.

BitTorrent improves the peer to peer download technology by allowing download of pieces of the file at the same time. Peers normally upload at slower rate than download rate and by multiple pieces download concurrently, the overall speed is greatly enhanced. The more peers in the swarm, the faster the file transfer due to more sources of each file.

Uses of BitTorrent in Real Life

- ✓ Facebook uses BitTorrent to distribute updates to Facebook servers
- ✓ Twitter uses BitTorrent to distribute updates to twitter servers.
- ✓ Academics use Bit Torrents to allow scientist to share large scientific data sets.

Technical terms in BitTorrent

1. **Leeches** – Users who like to download file but do not share file on their own PC with others.
2. **Seed or Seeders** – PC with complete copy of file
3. **Swarm** – A group of computers simultaneously uploading or downloading the same file
4. **.torrent** – A pointer file that directs the computer to the file user wishes to download
5. **Tracker** – A server that manages the BitTorrent file transfer process

LAN

A computer network that spans over a relatively small area. Most LANS are confined to a single building or group of buildings where the latter connects workstations and personal computers. Each node (individual computer) in a LAN has its own CPU with which it executes programs, but it also is able to access data and devices anywhere on the LAN.

This means that many users can share expensive devices, such as laser printers, as well as data. Users can also use the LAN to communicate with each other, by sending e-mail or engaging in chat sessions.

Ex: A LAN in a school

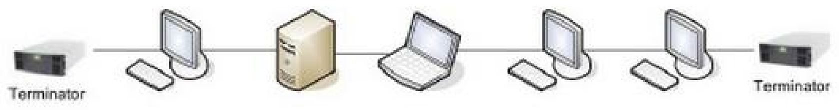
Network Topology

A topology is defined as the logical and physical layout of a network. The logical topology normally implies the path that data takes to travel between nodes while the physical topology means the layout of networking wiring materials and location of nodes.

The 5 main types of topologies are:

1. **BUS** (can be both logical and physical)
2. **STAR** (physical only)
3. **RING** (can be both logical and physical)
4. **MESH** (can be both logical and physical)
5. **HYBRID** (usually physical)

BUS Topology

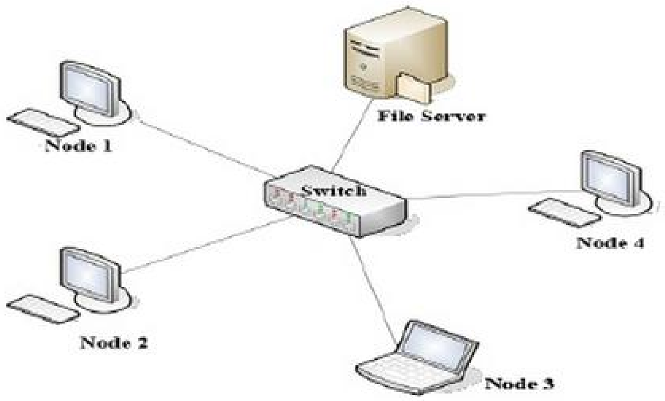


There is a continuous line which is formed whereby the nodes are connect adjacent to each other in the line. Coaxial cable is used along with T-connectors to connect the devices. At the end of each line a terminating device is used and its role is to absorb the data signal. Information passes from each node one and latter determines if the data is addressed to it.

Advantages	Disadvantages
▪ It is easy to install ▪ It uses less cable than STAR network ▪ It is the best choice for temporary networks	▪ If there is a problem with the central cable, the entire network stops working ▪ There is low security since every workstation can see all of the data in the network ▪ Difficult to add a workstation

STAR Topology

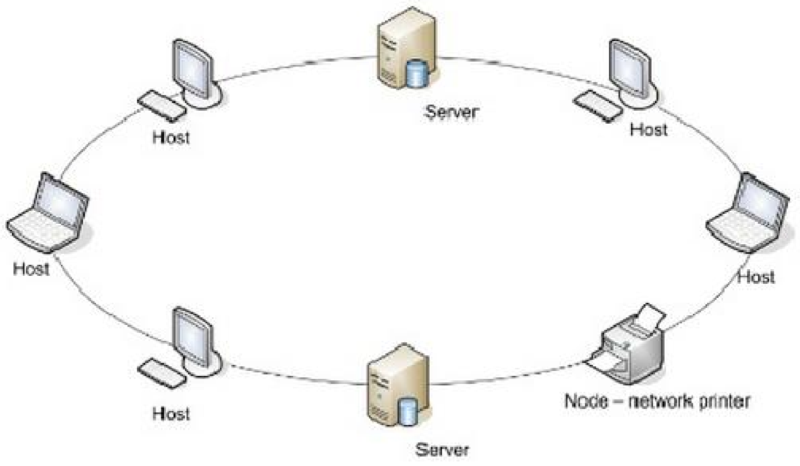
The nodes are connected to the central network using a switch or a hub. The switch normally distributes the information packet to its required destination by analysing the packet frames. Each workstation is connected directly by its own cable to the switch.



Advantage	Disadvantage
▪ It is very reliable. i.e. if one station fails , others can still use the network ▪ It is easy to identify faults using this type of topology ▪ It is easy to expand this type of network	▪ Installing such networks require experts ▪ If central device i.e. switch breaks down, all communication fail. ▪ It is one of the expensive network layout as it requires a large amount of cables

RING Topology

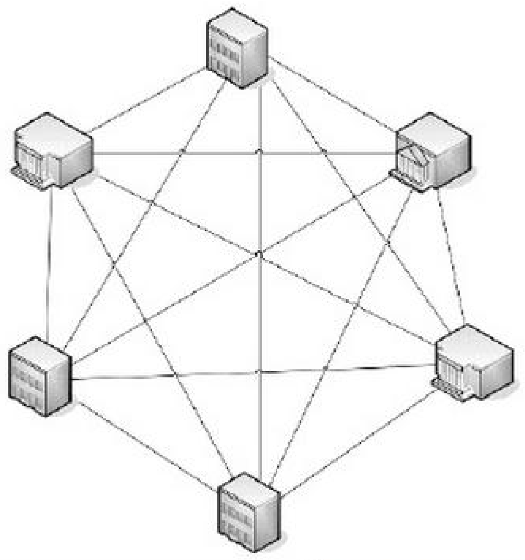
In this topology, each node is connected to two nodes on either side of it. All nodes form a continuous loop and da token is passed around the ring. The node can only transmit data if it has the token.



Advantage	Disadvantage
▪ There is no data collision ▪ It is possible to create large networks using this topology ▪ Transmission of data is fairly simple as it only travels in one direction	▪ If a single node is switched off, the network doesn't work ▪ It is difficult to add a new station as it has to come between 2 nodes ▪ If there are any problems with the network, they can be difficult to identify the cause

MESH Topology

In this topology, all the nodes have independent connections to every all other nodes. Such topology is very fault tolerant and scalable in design. In fact the Internet is based on the MESH topology. Routers are used to search multiple routes through the mesh.



Fawad Khan 0321-6386013

Advantage	Disadvantage
▪ It is one of the most reliable and high fault tolerant network ▪ Simplest for data flow	▪ Nodes required lots of cables ▪ Requires multiple network cards ▪ Most expensive layout to set up ▪ Require complex wiring scheme

HYBRID Topology

It uses two or more types of network topologies combined into one network. Most networks today are not only hybrid, but they are heterogeneous.

Advantage	Disadvantage
▪ Exploits the best features of all other topologies	▪ It may be one of the most expensive network to set up ▪ Complex to set up

Wireless Network

Wireless Local Area Network (WLAN) standards are guided by IEEE 802.11 standards documents which is often known as 'Wi-Fi'. Wireless devices communicate across specific range of Radio Frequencies (RF) usually known as channel using an antenna off a radio card.

Wi-Fi Alliance - is a global non-profit industry trade association and promotes wireless growth through interoperability certification

All 802.11 connections are half duplex and to achieve full duplex, normally devices transmit on a channel and receive on another one. But 802.11 devices have no method of detecting collision except an acknowledgement message. Hence, collision in wireless networks are avoided using CSMA/CA contention based media access.

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

Each device listen before attempting to transmit and only transmit if no other device is currently transmitting. If transmission is occurring by other devices, all other remaining one must wait until the transmission is finished. The current transmitting device includes a duration value within the header which informs others of the estimated time length of its transmission. Others will not only wait for that duration but also wait for an additional random amount of time before transmitting.

Major Wireless Standards

Standard	Speeds	Frequency	Indoor Distance	Outdoor distance	Interference
802.11b	<= 11 Mbps	2.4 GHz	45 m	90 m	yes
802.11a	<= 54 Mbps	5 GHz	15 m	30 m	Not compatible with 802.11b
802.11g	>20 Mbps, 54 Mbps	2.4 GHz	45 m	90 m	Yes , compatible with 802.11b
802.11n	<=600 Mbps	2.4 or 5 GHz	70 m	250 m	No Compatible with 802.11 b,g,a

Advantages	Disadvantages
▪ No cost of installing cables ▪ Hosts free to move around	▪ Interference from other wireless communications, fluorescent lights, microwave ovens, cordless phones etc. ▪ Building materials can block signals ▪ Security is a major issue as it can easy be hacked or eave drop compared to wired networks

Ethernet

It is the default standard technology used in LAN networking. There are several classification of Ethernet based on bandwidth requirements:

i.	802.3	- Ethernet (10 Mbps)
ii.	802.3u	- Fast Ethernet (100 Mbps)
iii.	802.3z or 802.3ab	- Gigabit Ethernet (1000 Mbps)

There are various categories of Ethernet with each one operating at various speeds, distance and cable types that are used: However they use the same addressing scheme and basic frame format.

Standard	Cable Type	Speed	Max Distance
10base2	Coaxial (Thinnet)	10 Mbps	185 meters
10base5	Coaxial (Thicknet)	10 Mbps	500 meters
10baseT	Twisted Pair	10 Mbps	100 meters
100baseT	Twisted Pair	100 Mbps	100 meters
1000baseSX	Fibre Optic (multi-mode)	1 Gbps	>500 meters
1000baseLX	Fibre Optic (single mode)	1Gbps	>3 Km

Ethernet devices normally using switch operate at full duplex whereby they can transmit and receive data simultaneously. Devices using hub use half duplex hence can either transmit or receive data but not simultaneously. Thus Half-duplex Ethernet uses Career Sense Multiple Access with Collision Detection (CSMA/CD) to control access to the cable. This is also known as controlling media access.

The Data link layer of OSI is divided into two sub layers which are LLC and MAC. Logical Link Control (LLC) links to upper layers and is independent of equipment while Media Access Control (MAC) provides addressing, frame format, error detection and CSMA/CD.

Advantages of Ethernet

- Reliable
- Simple and easy to maintain
- Ability to incorporate new technologies
- Low cost of installation and upgrade

Data Transmission

Ethernet used shared coaxial cable originally and hence if hosts transmit at the same time, collisions occurred. The wire is used as a shared transmission system. All the nodes detect the data transmission on the network but only to which it was addressed receives it.

Channel Access methods

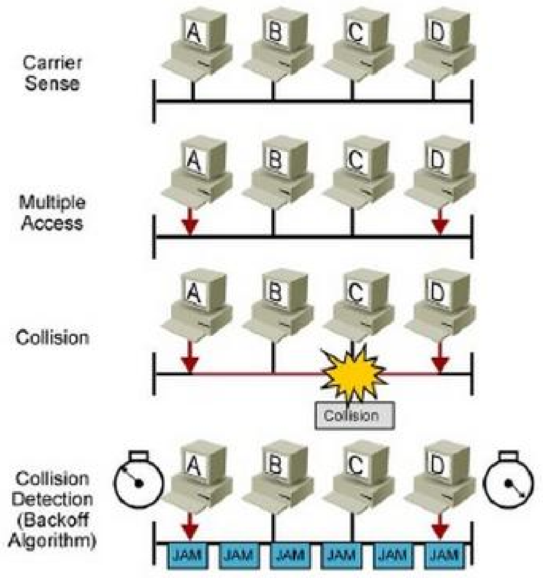
This determines the physical method by which data is send across the transmitting media. There are two main types which are:

i. CSMA/CD
ii. CSMA/CA

How CSMA/CD Works?

- **Carrier Sense** : 'Listen' to see if there are signals on the cable
- **Multiple Access** : Hosts share the same cable and all have access to it
- **Collision Detection** : Detect and manage any collisions of signals when they occur
- This is the 'first come, first served' method of letting hosts put signals on the medium

The diagram shows how CSMA/CD works
Collisions happen if a host transmits when there is already a signal on the cable and the latter does not yet know about it. This is due to latency, i.e., the time a signal takes to travel to the far end of the cable.



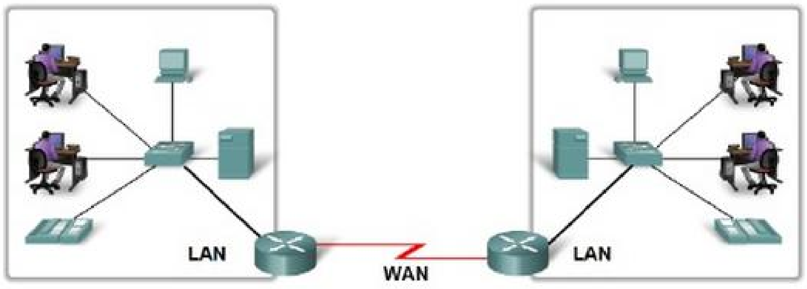
If a host detects a collision while it is sending the first 64 bits of a frame then CSMA/CD works and the frame will get resent later. However, if the host has send the 64 bits and then detects a collision, it is too late and thus it will not resend the frame. Hence, latency must be small so that all collision are detected in time which limits cable length and number of intermediate devices.

Solution to get rid of collisions

Use switches instead of hubs .This ensures that each device has a private cable and gets the full bandwidth. Hence full duplex mode is used on each link which obviously means there will be no collisions. Even higher bandwidth can be used to solve the problem of collisions.

WAN

A computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more LANs. Computers connected to a WAN are often connected through public networks, such as the telephone system. They can also be connected through leased lines or satellites. E.g. Connection of Branch Offices located in different countries



MAN

- A MAN is defined as a network that spans several LAN's across a city-wide geographic area

Internet

- It is defined as a global mesh of interconnected networks that spans across the globe.

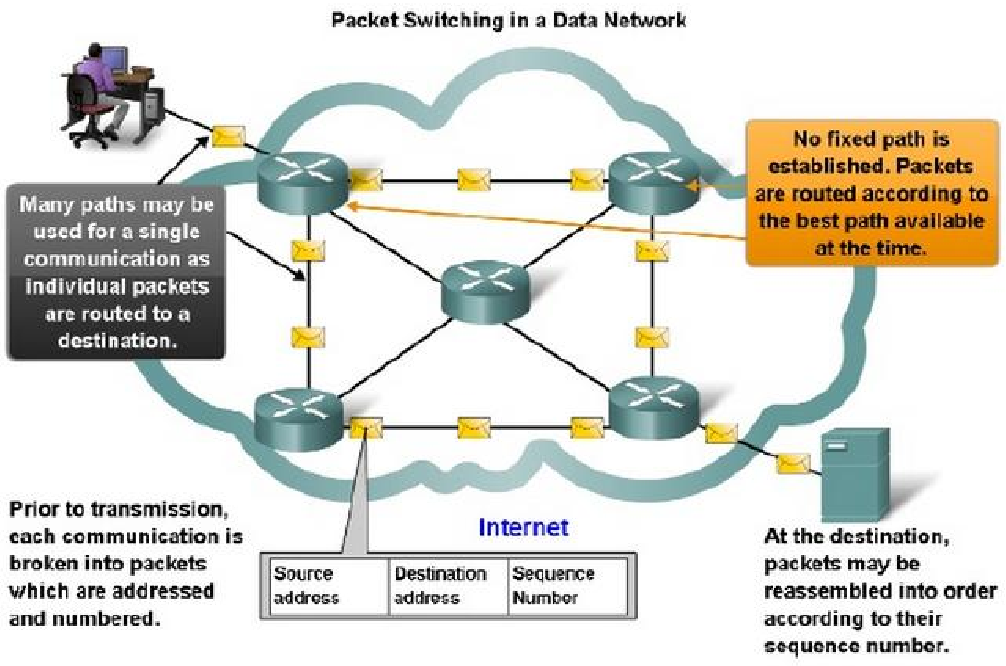
Packet switching

Data is grouped into packets before being sent over shared network. The packets can contain a variety of data types and they are routed through different parts. The data packets can be buffered and queued and then transmitted over the network. This can results in transmission delays.

Examples include: LAN and Internet

Packet Switching Steps

1. Data is split into packets.
2. Each packet has a source, destination address and a payload (data)
3. If data is split into multiple packet, each packet is assigned a number.
4. Packets are sent over the network moving from one router to another taking distinct path. Each packet can take a different travelling time.
5. Once the packet reach its destination, they are ordered in the required sequence.
6. An acknowledgement message is sent from the recipient to sender for message received.
7. If the sender receives no acknowledgement message, the sender re-transmit the data again

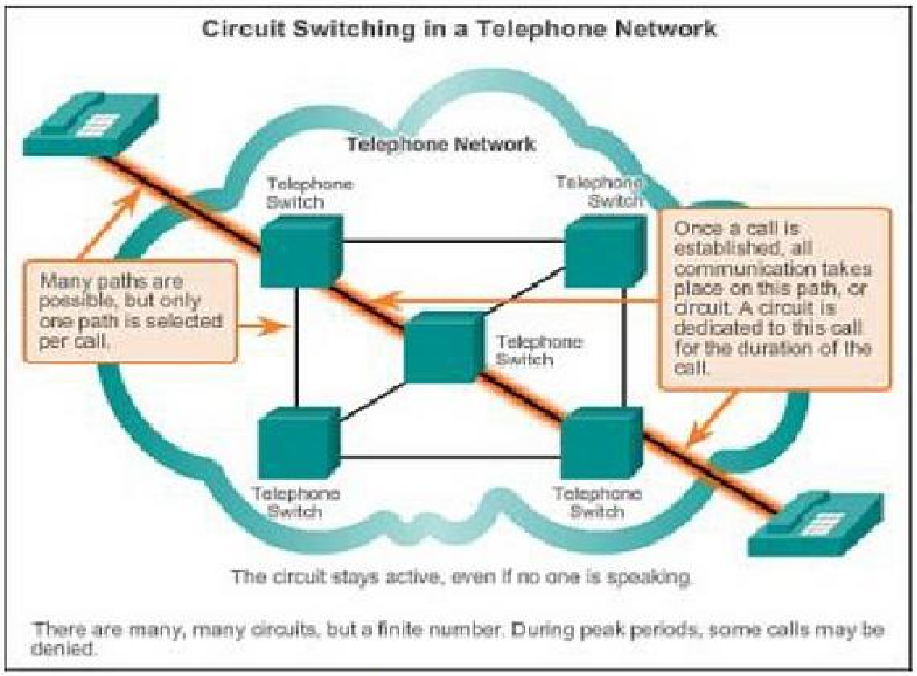

Packet Switching in a Data Network
Many paths may be used for a single communication as individual packets are routed to a destination.
No fixed path is established. Packets are routed according to the best path available at the time.
Prior to transmission, each communication is broken into packets which are addressed and numbered.
Internet
Source address
Destination address
Sequence Number
At the destination, packets may be reassembled into order according to their sequence number.

(Reference: cisco.com)

Circuit Switching

There is a virtual dedicated path for data transmission that occurs between the two nodes. Then transfer of data moves at non-stop rate along the path. The current path remains unavailable for other traffic until it's released.

Examples: PSTN and ISDN


Circuit Switching in a Telephone Network
Telephone Network
Telephone Switch
Telephone Switch
Telephone Switch
Telephone Switch
Telephone Switch
Many paths are possible, but only one path is selected per call.
Once a call is established, all communication takes place on this path, or circuit. A circuit is dedicated to this call for the duration of the call.
The circuit stays active, even if no one is speaking.
There are many, many circuits, but a finite number. During peak periods, some calls may be denied.

(reference : cisco.com)

Hardware – Logic Gate and Circuit Design

Combinational Logic Circuits (Paper 1 – Logic gates covered first before this one)

Electronic computer components are made from both sequential logic and combinational circuits. Combinational circuits always give the same output for a given set of input. Hence, it does not store any information. E.g. include

Adder, Decoder, multiplexer and Shifter among others. Combination of these components form larger unit such as ALU.

Consider the following binary addition in single bit:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

We can rewrite the above in two bits

0 + 0 = 00

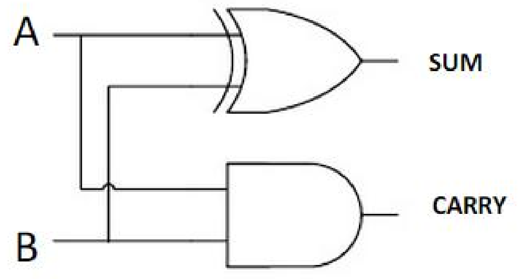
0 + 1 = 01

1 + 0 = 01
1 + 1 = 10

Half Adder

The output of '1' of 10 becomes the CARRY out while the normal output becomes the SUM. Hence,

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



SUM $= \overline{A} \bullet B + A \bullet \overline{B}$, which is equal to a EXOR gate output.

CARRY $= A \bullet B$, which is equal to an AND gate output.

This circuit is known as half adder and does 1 bit addition

Drawback: it does not account for carry in input

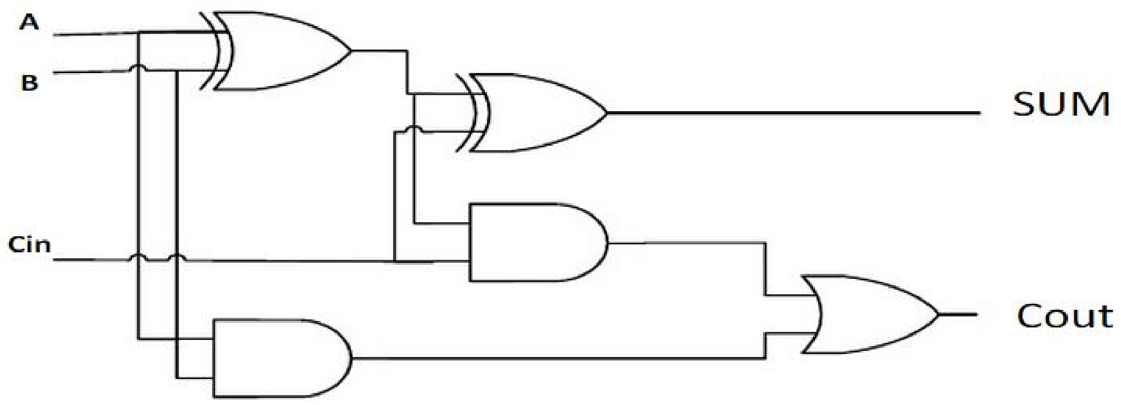
Full Adder

A full adder has three inputs and two outputs. The two first inputs are A and B and the third one is the Carry known as CIN. The output is known as COUT and the SUM.

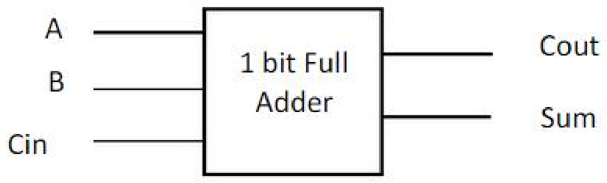
INPUTS			OUTPUTS	
A	**B**	**CIN**	**SUM**	**Cout**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

SUM is an EXOR between the input A and half adder SUM output with B and Cin inputs.

Cout is true if any two inputs out of the three are 1.

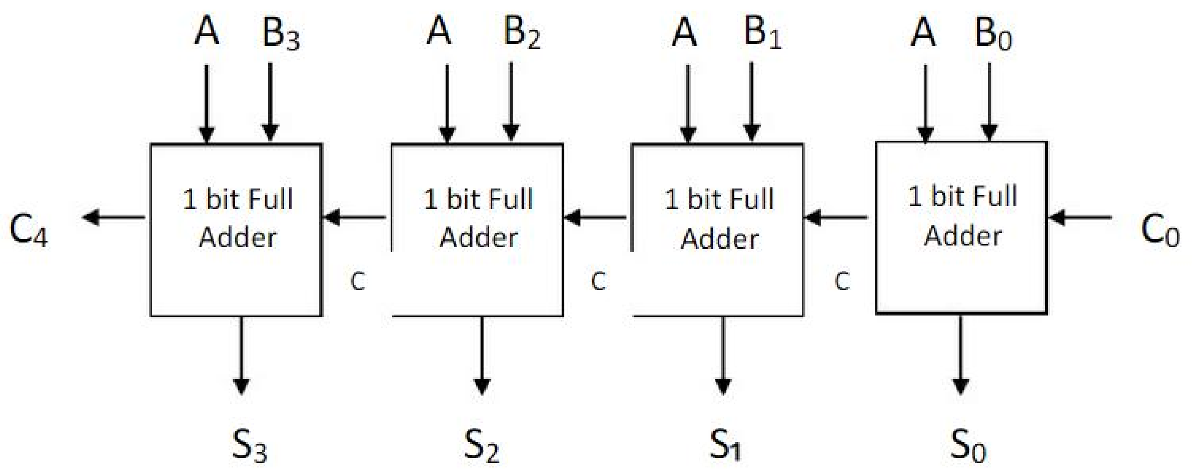


Though the implementation of larger logic diagrams is possible with the full adder circuit above, a simpler diagrammatic representation is given for a one bit full adder



N - Bit Adder

Full adders can be connected in series where N is the number of bits required to add. The carry bit ripples from one full adder to the next, hence this configuration is called a ripple carry adder.



Hardware – Boolean Algebra

Boolean Algebra

It is a mathematical system for the manipulation of variables that can take two values only. Either true or false, 1 or 0, high or low etc. Another way of representing gate logic is through Boolean algebra, a way of algebraically representing logic gates.

Description	**Bitwise Operator**
NOT GATE	NOT ($\overline{A}$)
AND GATE	AND (•)
OR GATE	OR (+)
XOR GATE	XOR (⊕)
NAND GATE	NAND ($\overline{A \bullet B}$)
NOR GATE	NOR ($\overline{A + B}$)

Boolean expressions are created by performing operations on Boolean variables. The common Boolean operators include AND, OR and NOT.

Characteristics of a Boolean Function

It has:

- At least one Boolean variable
- At least one Boolean operator
- At least one input from the set {0,1}
- It produces an output that is also a member of the set {0,1}

E.g. F (a, b, c) = a AND NOT c OR b

Boolean Operator

Boolean operator can be completely described using a truth table.

The **AND** operator is also known as "Boolean **Product**". It is represented with the dot symbol. E.g. $A \bullet B$

The **OR** operator is also known as "Boolean **Sum**". It is represented with the "+" symbol. E.g. A+B

The **NOT** operator is denoted by an over bar. It is sometimes designated by a prime mark (') or tilde (~). E.g. **$\bar{A}$**

A	B	F = $A \bullet B$	F = $A + B$
0	0	1	0
0	1	1	1
1	0	1	1
1	1	0	1

Evaluating Boolean Function

Consider the truth table for the following Boolean Function:

$$F(a, b, c) = a\bar{c} + b$$

To evaluate the Boolean function easier, the truth table contains extra columns to hold evaluations of subparts of the function.

F (a, b, c) = $a\bar{c} + b$					
a	b	c	$\bar{c}$	$a\bar{c}$	$a\bar{c} + b$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Boolean Operator Precedence

1. NOT (Highest)
2. AND
3. OR (Lowest)

Boolean Function Reduction

All digital computers circuitry implements Boolean functions. The simpler we construct a Boolean function, the smaller the circuit will result.

Benefits of Function Reduction

1. Simpler circuits imply that they are cheaper to build.
2. Simpler circuits consume less power
3. Simpler circuits run faster than complex circuits.

Hence, bearing this is mind, we always want to reduce our Boolean functions to their simplest form. There are two common ways to reduce Boolean function which are:

a) Using Boolean Identities (similar to proof of identities in maths)
b) Using Karnaugh Map

Boolean Identities

Boolean identities are rules that allows you to simplify Boolean expressions. Most Boolean identities have an AND (product) form as well as an OR (sum) form.

Identity Name	**AND Form**	**OR Form**
Identity Law	$1 \bullet A = A$ $\begin{array}{ccc} 1 & A & 1 \bullet A = A \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$	$0 + A = A$ $\begin{array}{ccc} 0 & A & 0 + A = A \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{array}$
Null Law	$0 \bullet A = 0$ $\begin{array}{ccc} 0 & A & 0 \bullet A = 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{array}$	$1 + A = 1$ $\begin{array}{ccc} 1 & A & 1 + A = 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}$
Idempotent Law	$A \bullet A = A$ $\begin{array}{ccc} A & A & A \bullet A = A \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{array}$	$A + A = A$ $\begin{array}{ccc} A & A & A + A = A \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{array}$

Inverse Law	$A \bullet \overline{A} = 0$	$A + \overline{A} = 1$

A	$\overline{A}$	$A \bullet \overline{A} = 0$
0	1	0
1	0	0

A	$\overline{A}$	$A + \overline{A} = 1$
0	1	1
1	0	1

Commutative Law	$A \bullet B = B \bullet A$	$A + B = B + A$

A	B	$A \bullet B$	$B \bullet A$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

A	B	$A + B$	$B + A$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Associative Law	$(A \bullet B) \bullet C = A \bullet (B \bullet C)$	$(A + B) + C = A + (B + C)$
Distributive Law	$A + (B \bullet C) = (A + B) \bullet (A + C)$	$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$

Absorption Law	$A \bullet (A + B) = A$	$A + (A \bullet B) = A$
DeMorgan's Law	$\overline{(A \bullet B)} = \overline{A} + \overline{B}$	$\overline{(A + B)} = \overline{A} \bullet \overline{B}$

$A \bullet (A + B) = A$

A	B	$A + B$	$A \bullet (A + B)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

$A + (A \bullet B) = A$

A	B	$A \bullet B$	$A + (A \bullet B)$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

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

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

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

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

Double Compleme nt Law	$\overline{\overline{A}} = A$

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
1	0	1

Simplification of function using Boolean Identities

e.g. 1 F(x, y) = $(x + y) \bullet (\overline{x} + y)$

1. F(x, y) = $x\overline{x} + xy + y\overline{x} + yy$ (Distributive Law)
2. F(x, y) = $0 + xy + y\overline{x} + y$ (Inverse + Idempotent Law)
3. F(x, y) = $xy + y\overline{x} + y$ (Identity Law)
4. F(x, y) = $y(x + \overline{x}) + y$ (Distributive Law)
5. F(x, y) = $y(1) + y$ (Inverse Law)
6. F(x, y) = $y + y$ (Identity Law)
7. F(x, y) = y (Idempotent Law)

e.g. 2 F(x, y) = $(A \bullet \overline{B}) + (A \bullet B)$

1. F(x, y) = $A \bullet (\overline{B} + B)$ Take Common Factor out
2. F(x, y) = $A \bullet (1)$ Using Inverse Law
3. F(x, y) = A Using Identity Law

DeMorgan's Law

DeMorgan's law provides an easy way of finding the complement of a Boolean function.

It is used to simplify Boolean equations so that a circuit could be build using one sort of gate usually NAND or NOR gate. This is done because it leads to cheaper hardware design.

The two laws of DeMorgan's are:

1. Law 1: $\overline{(A \bullet B)} = \overline{A} + \overline{B}$
2. Law 2: $\overline{(A + B)} = \overline{A} \bullet \overline{B}$

The above two law have been proven in their respective truth table. The law can be extended to any number of variables.

To get a function complement using DeMorgan's Law , simply replace each variable by its complement and change all ANDs to ORs and all ORs to ANDs .

E.g. F (A, B, C) = $(AB) + (\overline{A}C) + (B\overline{C})$

1. $\overline{F}$ (A, B, C) = $\overline{(A \bullet B) + (\overline{A} \bullet C) + (B \bullet \overline{C})}$ Using DeMorgan's Law
2. $\overline{F}$ (A, B, C) = $\overline{(A \bullet B)} \bullet \overline{(\overline{A} \bullet C)} \bullet \overline{(B \bullet \overline{C})}$ Replace all OR By AND
3. $\overline{F}$ (A, B, C) = $(\overline{A} + \overline{B}) \bullet (A + \overline{C}) \bullet (\overline{B} + C)$ Simplify using Double Complement Law

Boolean Function Standardisation

It has to be noted that there are several ways of stating the same Boolean expression which are logically equivalent and logically equivalent expressions have identical truth tables.

Hence, to eliminate as much as confusion as possible, circuit designers express Boolean functions in standardised or canonical form.

Furthermore, the benefits of standardised form is that Boolean function expression can be made from the output column of the truth table.

There are two canonicals forms for Boolean expressions:

1. Sum of Products (SOP)
2. Product of sums (POS)

In the sum of product form, ANDed variables are ORed together. E.g. F (A, B, C) = AB +AC +BC

In the product of sum form, ORed variables are ANDed together. E.g. F (A, B, C) = (A+B) (A+C) (B+C)

Sum of Product Form

We need to use the truth table to convert the output to SOP form. Only the output of the function where value are true are considered (i.e. 1)

Using the truth table, list the values of variables that result in a true value

a) The variables corresponding to row with output 1 are "ANDed"
b) If the variable's input value is 1, then it is written as it is else the complement of that variable is written.

Each group of variable is then "ORed" together

E.g. consider the following Function truth Table

F = $A\overline{C} + B$				
A	B	C	$A\overline{C} + B$	SOP where output =1
0	0	0	0	
0	0	1	0	
0	1	0	**1**	$\overline{A}B\overline{C}$
0	1	1	**1**	$\overline{A}BC$
1	0	0	**1**	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	**1**	$AB\overline{C}$
1	1	1	**1**	ABC

SOP Form:

F(A,B, C) = $\overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$ (SOP Form)

F = $A\overline{C} + B$ (reduced form using Boolean Identities)

Alternative Approach to Boolean Function Simplification

Normally simplification of Boolean functions leads to simpler (and usually faster) digital circuits. Simplifying Boolean functions using identities is time consuming and/or error prone.

Karnaugh Map (Kmap)

It is a graphical representation of visualising and simplifying Boolean expressions. The Kmap is a matric consisting of rows and columns which represent the output values of Boolean functions. It can be of two forms:

i. SOP Form
ii. POS Form

The output values are placed in each cell of the matrix are derived from the "minterms" of a Boolean function.

A minterm is a product term that contains all of the function's variables exactly one, either complemented or not complemented.

We reduce our complicated expression to its simplest terms by finding adjacent 1's in the Kmap that can be collected into groups that are "Powers of two", group of 1, 2, 4, 8 and 16. The smaller the grouping the larger will be the number of variables required to describe it.

A variable may be found in more than one group for reducing a Boolean expression.

E.g. consider the following Function truth Table

	F = $A\overline{C} + B$				**MINTERMS**
Decimal Value	A	B	C	$A\overline{C} + B$	SOP where all output
m0	0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$
m1	0	0	1	0	$\overline{A}\,\overline{B}C$
m2	0	1	0	**1**	$\overline{A}B\overline{C}$
m3	0	1	1	**1**	$\overline{A}BC$
m4	1	0	0	**1**	$A\overline{B}\,\overline{C}$
m5	1	0	1	0	$A\overline{B}C$
m6	1	1	0	**1**	$AB\overline{C}$
m7	1	1	1	**1**	ABC

Minterms of the above functions are: $\overline{A}\,\overline{B}\,\overline{C}$, $\overline{A}\,\overline{B}C$, $\overline{A}B\overline{C}$, $\overline{A}BC$, $A\overline{B}\,\overline{C}$, $A\overline{B}C$, $AB\overline{C}$ and ABC

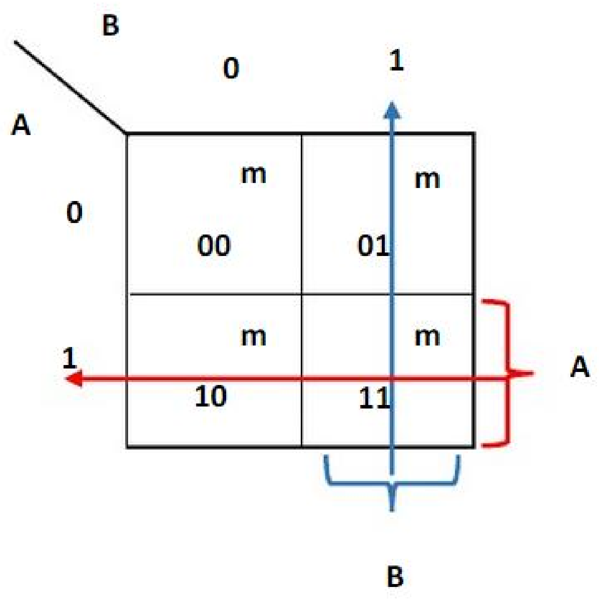
Note: if the variable input is 1, then it is written as it is else the complement of the variable is written

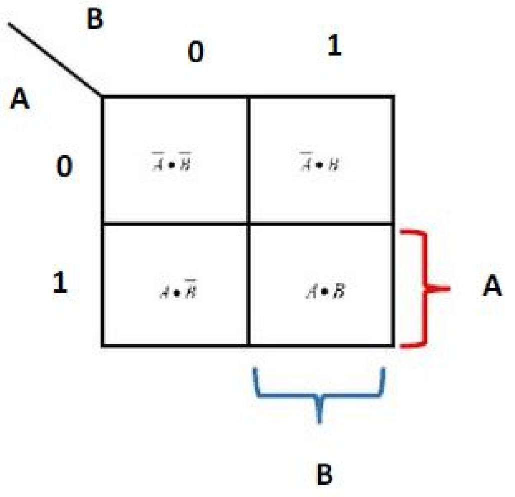
Kmap for 2 input

Minterms Value	A	B	**SOP FORM of AB**
m0	0	0	$\overline{A} \bullet \overline{B}$
m1	0	1	$\overline{A} \bullet B$
m2	1	0	$A \bullet \overline{B}$
m3	1	1	$A \bullet B$

Imagine the boxes as plot of land where A is the owner of plot m2, m3 while B is the owner of m1, m3 and there is a plot that both own in common i.e. m3 and a plot that both do not own , m0 ($\overline{A} \bullet \overline{B}$).

This analogy shall help you identify the correct SOP of function reduction later. Note the arrows indicate the plot each owner has, the region covered by A and B. the region plot number is where respective owner have value 1 in their input.





Minterms Value	A	B	SOP FORM of AB
m0	0	0	$\overline{A} \bullet \overline{B}$
m1	0	1	$\overline{A} \bullet B$
m2	1	0	$A \bullet \overline{B}$
m3	1	1	$A \bullet B$

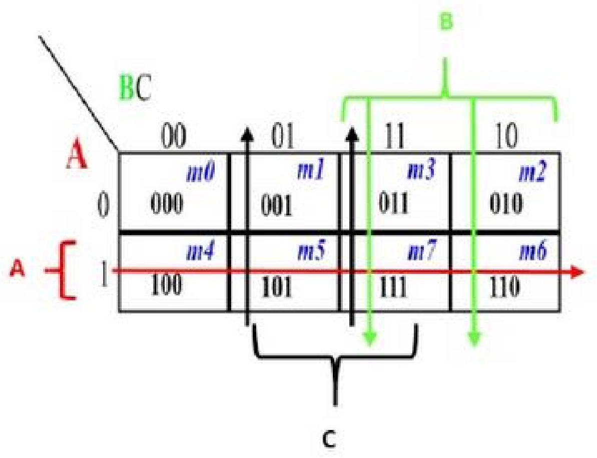
Minterms Value	A	B	SOP FORM of AB
m0	0	0	$\overline{A} \bullet \overline{B}$
m1	0	1	$\overline{A} \bullet B$
m2	1	0	$A \bullet \overline{B}$
m3	1	1	$A \bullet B$

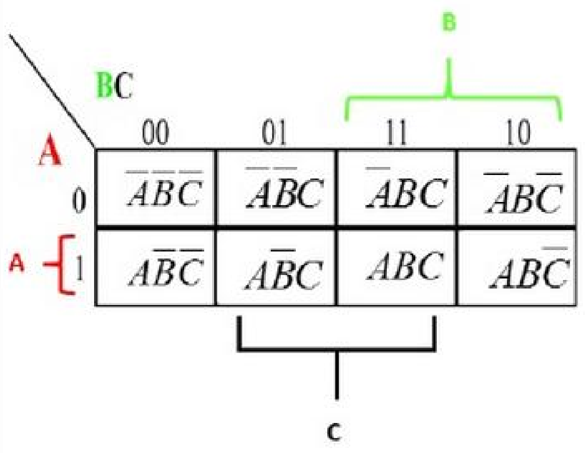
Kmap for 3 input variables

A Kmap must be ordered so that each minterm differs only in one variable from each neighbouring cell hence 11 appears before 10 – Rule!! (Will help simplification)

We have placed each minterm in the cell that will hold its value. Thus, the first row of the Kmap contains all minterms where *A* has a value of zero. The first column contains all minterms where *B* and *C* both have a value of zero

Note: values for the *BC* combination at the top of the matrix form a pattern that is not a normal binary sequence





Note : we have chosen simply not to put the • since we already know by writing for instance ABC implies A **AND** B **AND** C

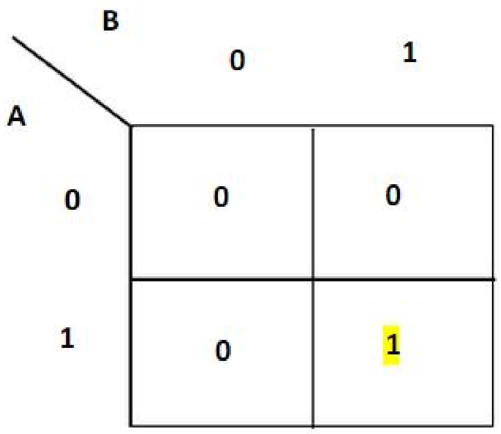
				MINTERMS
Decimal Value	A	B	C	**SOP**
m0	0	0	0	$\bar{A}\,\bar{B}\,\bar{C}$
m1	0	0	**1**	$\bar{A}\,\bar{B}\,C$
m2	0	1	0	$\bar{A}\,B\,\bar{C}$
m3	0	1	**1**	$\bar{A}\,BC$
m4	1	0	0	$A\,\bar{B}\,\bar{C}$
m5	1	0	**1**	$A\,\bar{B}\,C$
m6	1	1	0	$AB\,\bar{C}$
m7	1	1	**1**	ABC

Kmap Cell using SOP Form and Simplification for two variables

E.g. 1, consider the following truth table for the given function

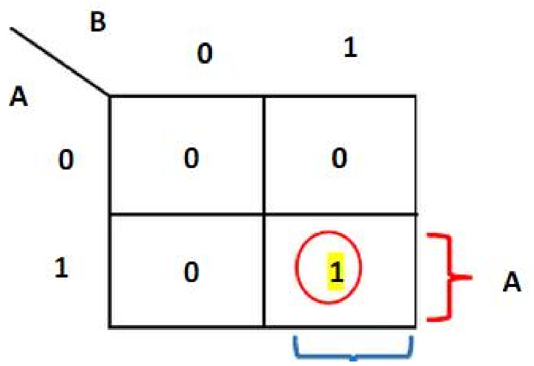
	F (A,B) = AB			
Minterms	A	B	**F= AB**	
m0	0	0	0	$\overline{A}\,\overline{B}$
m1	0	1	0	$\overline{A}B$
m2	1	0	0	$A\overline{B}$
m3	1	1	1	AB

Corresponding Kmap



F = AB

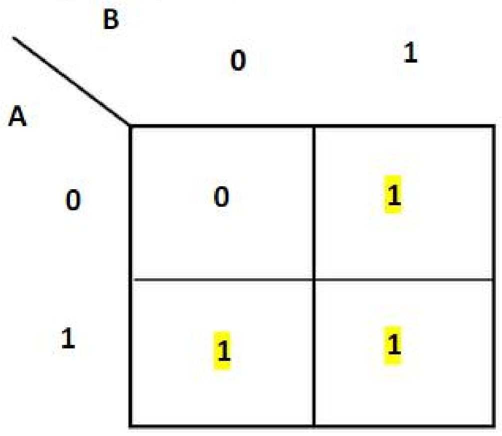
It is in the simplest form since the grouping available is shown below



E.g. 2, consider the following truth table for the given function

	F (A,B) = A + B			
Minterms	A	B	**F= A + B**	
m0	0	0	0	$\overline{A}\,\overline{B}$
m1	0	1	1	$\overline{A}B$
m2	1	0	1	$A\overline{B}$
m3	1	1	1	AB

Corresponding Kmap

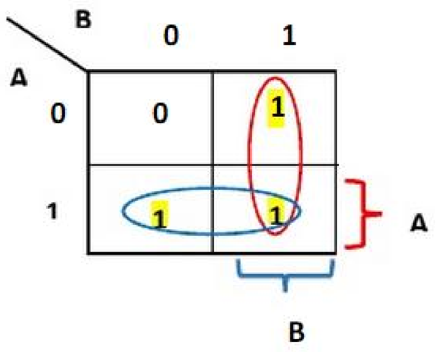


F (A,B) = A+B = $\overline{A}B + A\overline{B} + AB$

Of course from out Kmap, the minterms derived above are not in its simplest term.

We can reduce our complicated expression to its simplest terms by finding adjacent 1s in the Kmap that can collected into groups that are “powers of two” , i.e. 1,2,4,8,16

From the above Kmap , we can have the following 2 groups , one vertical (red) and one horizontal (blue)

Hence, group red, is found in the plot of B and A. we would direct someone to say that the plot they want to buy belongs to both A and B , hence : **F(A,B) = AB**	 Hence, group red, is found in the plot of B as we could describe to someone while group blue, we can describe as being found in the plot of A. Therefore , F (A,B) = **A + B** which is the simplest form of the expression which has been reduced using Kmap

Rules for Kmap Simplification using SOP

The rules of Kmap simplification are:

1. Groupings can contain only 1's : no 0s
2. Groups can be formed only at right angles ; diagonal groups are not allowed
3. The number of 1s in a group must be a power of 2 even if it contains a single 1
4. The groups must be made as large as possible
5. Groups can overlap and wrap around the sides of the Kmap

Kmap with three variables

E.g. 1. Consider the Following Function

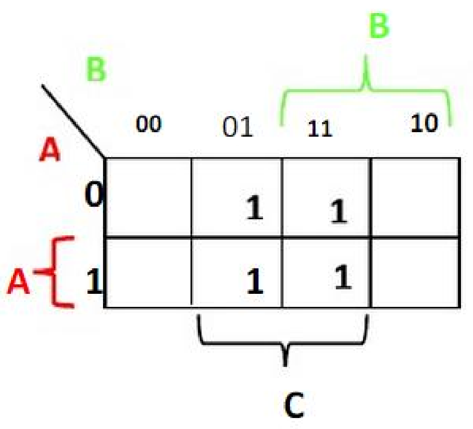
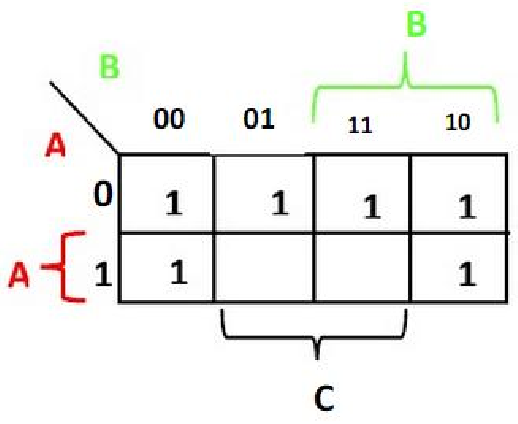
$F(A,B,C) = \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}C + ABC$

					MINTERMS
Decimal Value	A	B	C	F	SOP
m0	0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$
m1	0	0	1	1	$\overline{A}\,\overline{B}C$
m2	0	1	0	0	$\overline{A}B\overline{C}$
m3	0	1	1	1	$\overline{A}BC$
m4	1	0	0	0	$A\overline{B}\,\overline{C}$
m5	1	0	1	1	$A\overline{B}C$
m6	1	1	0	0	$AB\overline{C}$
m7	1	1	1	1	ABC

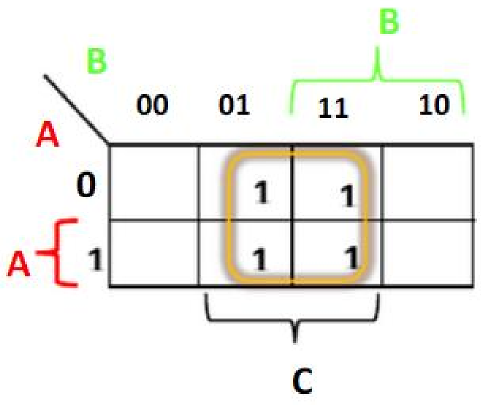
E.g. 2. Consider the Following Function

$F(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C}$

					MINTERMS
Decimal Value	A	B	C	F	SOP
m0	0	0	0	1	$\overline{A}\,\overline{B}\,\overline{C}$
m1	0	0	1	1	$\overline{A}\,\overline{B}C$
m2	0	1	0	1	$\overline{A}B\overline{C}$
m3	0	1	1	1	$\overline{A}BC$
m4	1	0	0	1	$A\overline{B}\,\overline{C}$
m5	1	0	1	0	$A\overline{B}C$
m6	1	1	0	1	$AB\overline{C}$
m7	1	1	1	0	ABC

Looking at the output of function F, the following Kmap is constructed as shown below for values where output is 1 and not put the output where 0 :  B B A A 00 01 11 10 0 1 1 1 1 1 C F(A,B,C) = $\overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}C + ABC$	Looking at the output of function F, the following Kmap is constructed as shown below for values where output is 1 and not put the output where 0 :  B B A A 00 01 11 10 0 1 1 1 1 1 1 1 1 C F(A,B,C) = $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\,\overline{C}$ $+ AB\overline{C}$

Using Kmap grouping, we can group the values in a group of 4.

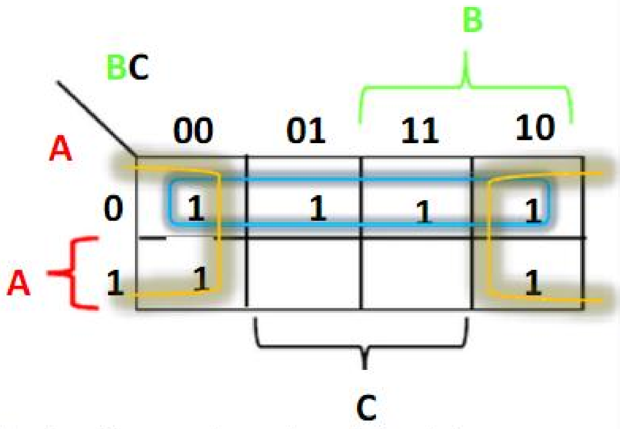


The function can be reduced simply to :

F(A,B,C) = C

Which represents the group of 4. Notice the larger the group, the simpler is the expression

Using Kmap ,we can group the values in two groups of 4 as shown : group 1 : Blue , group 2 : Yellow



The function can be reduced simply to :

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

Where $\overline{A}$ is representing the blue group and $\overline{C}$ is the yellow group.

Choosing Kmap Groups

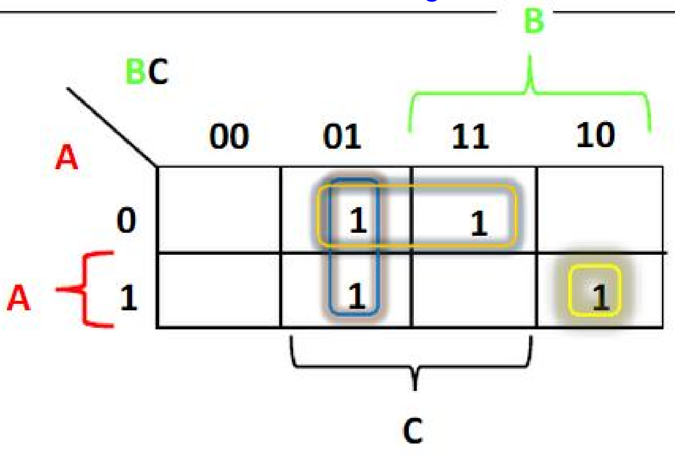
It is possible to have a choice as to how to pick groups within a Kmap while keeping the groups as large as possible.

Don't Care Conditions

Real circuits don't always need to have an output defined for every possible input. If a circuit is designed so that a particular set of inputs can never happen, we call this sets of inputs a ***don't care*** condition. They are very helpful to us in Kmap circuit simplification.

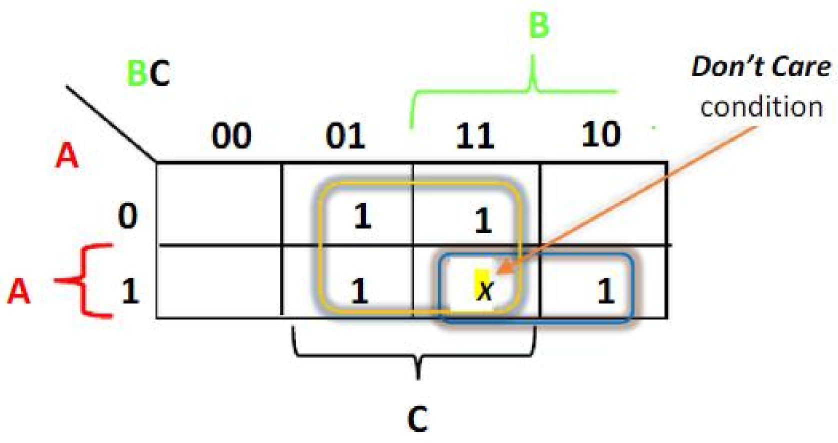
Don't Care Example.

In a Kmap, a don't care condition is identified by an *X* in the cell of the minterm(s) for the ***don't care*** inputs as shown below. In performing the simplification, we are free to include or ignore the X's when creating our groups



Consider the above map , if we perform the grouping above , **F(A,BC) =** $C\bar{B} + \bar{A}C + BA\bar{C}$

Suppose now we add a don't care condition to the above map as shown below :



Now our grouping is easier and we **have F(A,B,C) =** $C + AB$

Hardware – Flip Flops

Combinational Logic

We have seen combinational logic whereby the output(s) depends only on the current values of the input variables.

Sequential Logic

The output depend on the present and also the past values of the input and the output variables. Sequential circuits exist in one of a defined number of states at any one time i.e. they move through a defined sequence of transitions between states and the output variables describe the state of a sequential circuit directly or by deriving state variables from them

Synchronous and Asynchronous Sequential Logic

- **Synchronous** – the timing of all state transitions is controlled by a common clock and the changes in all variables occur simultaneously.
- **Asynchronous** – state transitions occur independently of any clock and normally depends on the timing of transitions in the input variables. The changes in more than one output do not necessarily occur simultaneously.
- **A clock signal** is a square wave of fixed frequency. Normally, transitions will occur on one of the edges of clock pulses, i.e. the rising edge or the falling edge.

Flip Flops

Flip-flops are fundamental elements of sequential circuits. They are bi-stable. They are essentially 1 bit storage devices where their output can be set to store either 0 or 1 depending on the inputs. Even if the inputs are de-asserted, the outputs retain their set value.
A sequential circuit may use many flip-flops to store as many bits as required.

Flip-flops have 2 complimentary outputs normally usually denoted by Q and $\overline{Q}$.

The 4 main types of flip flops are:

1. S – R (known as basic Flip Flop)
2. J - K
3. **D – Flop (known as Delay Flip Flop) – not in syllabus**
4. **T Flip-flop – not in syllabus**

The major difference among various types of flip-flops are in the number of inputs they possess and in the way in which the inputs affect the binary state. Flip flops are in fact an application of logic gate. We can create memory with them and hence, they can be considered as the most basic idea of Random Access Memory.

Use

- Implementation of feedback circuits.
- Helpful in designing better electronic circuits.

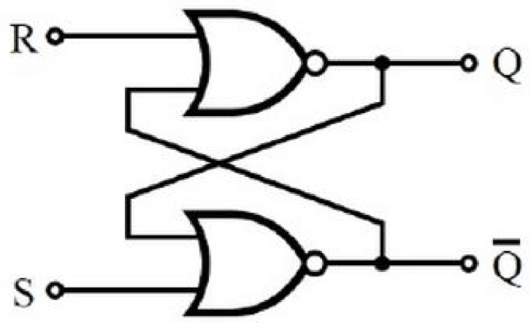
The most basic flip-flops operate with signal levels and are referred to as latches.

R-S Flip Flop (SET –RESET)

The Set-Reset flip flop can be designed using two NOR gates or two NAND gates. Sometime. These flip flops are also called S-R Latch.

R -> Reset, S -> Set

Logic Diagram



R-S Truth Table

S	R	Q	$\overline{Q}$	
1	0	1	0	
0	0	1	0	(after S =1 and R = 0)
0	1	0	1	
0	0	0	1	(after S = 0 , R = 1)
1	1	0	0	

We can deduce from the diagram that the flip flop has 4 main states.

1. S = 1, R= 0, Q =1 , $\overline{Q}$ = 0 , this state is known as SET state
2. S = 0, R= 1, Q = 0 , $\overline{Q}$ = 1 , this state is known as RESET state

Note: in both states, the outputs simply are complements of each other and the value of Q follows that of S

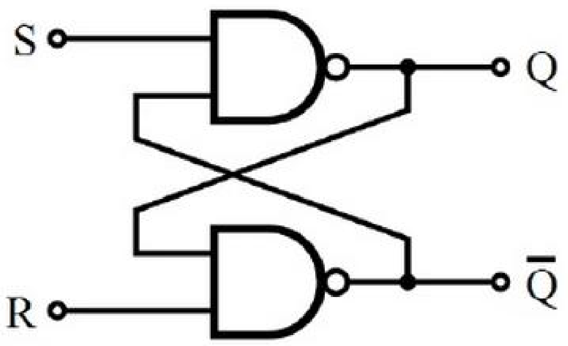
3. S = 0, R= 0, Q & $\overline{Q}$, Remember State

If both S = 0, R = 0, then the circuit remembers the previous value of S and R , Invalid State

4. S = 1, R= 1, Q =0, $\overline{Q}$ = 0 , Invalid state – because Q & $\overline{Q}$ are supposed to be the complement of each other. Hence, this state should be avoided.

S-R Flip-flop using NAND gate

Logic Diagram



S-R Truth Table

S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	(after S =1 and R = 0)
0	1	1	0	
1	1	1	0	(after S = 0 , R = 1)
0	0	1	1	

The 4 states are:

1. S = 1, R= 0, Q =0 , $\overline{Q}$ = 1 , this state is known as SET state
2. S = 0, R= 1, Q = 1 , $\overline{Q}$ = 0 , this state is known as RESET state

Note: in both states, the outputs simply are complements of each other and the value of Q follows that of S

3. S =0, R= 0, Q =1 & $\overline{Q}$ =1 , Invalid state

In case both S and R = 0, it is an invalid state since Q and $\overline{Q}$ =1 where supposed to be complement of each other but both are 1 in this case.

4. S = 1, R= 1, Q = & $\overline{Q}$, Remember state

J- K Flip-flop

It can be defined as a modification of the S-R flip-flop. The only difference is that the intermediate state is more refined and precise than that of a S-R flip flop.

J and K behaviour are similar to that of S and R where J stands for SET and K stands for CLEAR. The two NOR gates form the memory; at any one time one is high, the other low and feedback maintains that state.

or

When both J and K = 1. The flip flop switch to the complement state. So, for a value of Q = 1, it switches to Q=0 and for a value of Q = 0, it switches to Q=1.

A clock pulse (CP) is used such that when CP =1, the flip-flop is cleared.

Q	J	K	Q (t +1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Hardware – RISC Processors

Instruction Set Architecture

It is defined as the instruction set of the microprocessor, i.e. the set of all of the assembly language instructions that the CPU can execute. It normally specifies the following:

i. Registers available, their size and the set of instructions that can use each register
ii. Information necessary to interact with the memory (e.g. start a specific memory location)
iii. The way the processor will react to interrupts

CISC (Complex Instruction Set Computers)

CISC development was triggered on that basis that better performance would be obtained by reducing the number of instruction required to implement a program. However, as compiler technologies began to evolve, soon researchers wondered if CISC architectures really delivered better performance than RISC

Common processor based on that architecture: Intel 80x86

Characteristic of CISC

- ✓ There were fewer instructions to execute a given task than RISC
- ✓ The programs for CISC take less storage space than programs for RISC
- ✓ The arithmetic or other instructions may read their operand from memory and could write the result in memory
- ✓ Extensive addressing capabilities for memory operations.
- ✓ Relatively few registers

CISC Problems

- ✓ Performance tuning unsuccessful (rarely used in high level instructions , sometimes slower than equivalent sequence)
- ✓ High Complexity (Pipelining bottle necks implies lower clock rates , interrupt handling can complicate more)

RISC (Reduced Instruction Set Computers)

RISC is a CPU architecture that uses a small highly optimised set of instructions instead of a more specialised set of instructions.

Characteristics of RISC

- ✓ Reduced instruction set.
- ✓ Less complex, simple instructions.
- ✓ Hardwired control unit and machine instructions.
- ✓ Few addressing schemes for memory operands with only two basic instructions, LOAD and STORE
- ✓ Many symmetric registers which are organised into a register file.

RISC Problems

- ✓ Compiler Issues (The compilers themselves , the compiler writer)

E.g. of RISC V/S CISC instruction

RISC	CISC
LD R4, (R4) LD R5, (R5) ADD R6, R4, R5 ST (R3), R6	ADD (R3), (R2), (R1)

Addition of two operands from memory, with result written in memory, in RISC and CISC architectures

Having an operation broken into small instructions (RISC) allows the compiler to optimize the code. I.e. between the two LD instructions (memory is slow) the compiler can add some instructions that don't need memory access

The CISC instruction has no option but to wait for its operands to come from the memory, potentially delaying other instructions

Comparison between CISC V/S RISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes

Transistors used for storing complex instructions	Spends more transistors on memory registers

Pipelining

Consider the following.

A Von Neumann Architecture instruction can be in one of three phases:

1. Fetched instruction From memory
2. Decode instruction by control unit
3. Execute the instruction by the control unit

Problems with Von Neumann architecture

- ✓ It executes instructions serially.
- ✓ There is a time limitation on how much it takes to process each instruction.
- ✓ There are some registers which are idle during the Fetch - Decode - Execute cycle.

Pipelining is a technique whereby multiple instructions are overlapped in execution. The pipeline is divided into stages whereby each stage completes a part of an instruction in parallel.
Note: Pipelining does not decrease the time for individual instruction execution but rather it increases the instruction throughput. The latter is determined by how often an instruction exits the pipeline.

Pipelining is widely used in modern processors and it improves system performance in terms of throughput. However, pipelined organisation requires sophisticated compilation techniques.

Pipelining—CPU takes at least four steps (called stages):

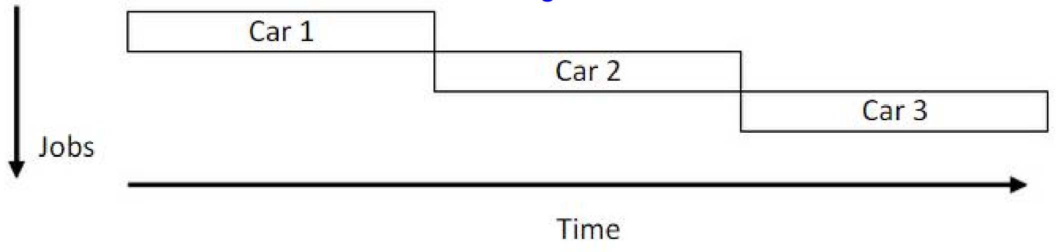
1. Fetch : Get the data from the EDB (External Data Bus)
2. Decode : Figure out what type of command needs to be executed
3. Execute: Perform the calculation
4. Write : Send the data back onto the EDB

Some newer CPUs have many stages in pipeline. This makes a CPU run more efficiently without increasing the clock speed. Moreover, some processors use multiple decode stages to reduce pipeline stalls. There is also some CPUs which offer multiple pipelines, allowing the arithmetic logic unit (ALU) and the floating point unit to work at the same time

Analogy to Understand Pipeline

Let us consider the production of a car in a factory which is Un-pipelined.

Unpipelined – Start and Finish a job before moving to the next.



One car is produced at a time where all workers are busy working on different aspects of the car all at the same time. Let's say it takes 24 hours to build a whole car, then everybody get to work on the next car.

This is an un-optimised design because there is no parallelism since 1 car at a time.

Parallelism – 1 Car, **Throughput Rate** -> 1 Car /24 Hr

Pipelined Optimised Design

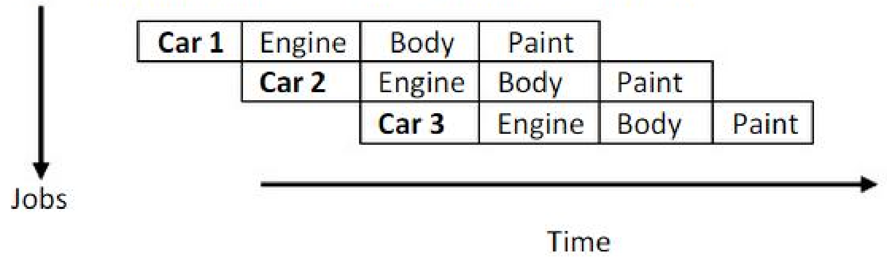
Now let's take the building process and breaking the job into 3 smaller stages of 8 hours each.

- Stage 1 - Engine of Car
- Stage 2 - Body of Car
- Stage 3 - Paint of Car

At first there is a car and there is a specialised group of workers that works on engine and spend 8 hours to produce engine for car 1, after 8 hours car1 goes into stage 2 where there is a different group of workers specialised in the body. Now the group of workers in stage A are free to build a second car2 Engine. At this point of time there is 2 car being built and this continues on as shown in the diagram and 3 cars is produced in 40 hrs.

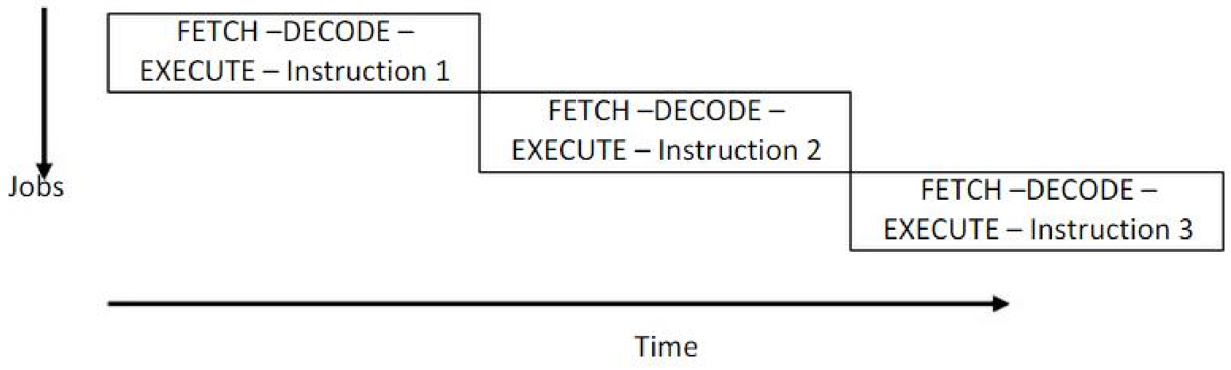
Always 3 car in production at any point in time.

Parallelism – 3 Cars, **Throughput** - > 1 Car / 8hrs (increased by 3 X)

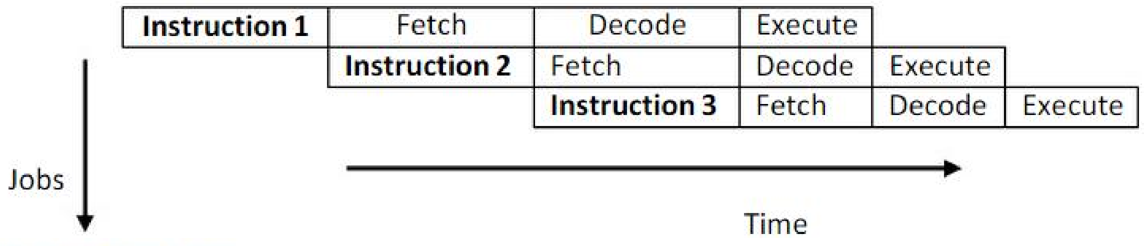


In the Context of Computer Pipelining,

Unpipelined


FETCH –DECODE – EXECUTE – Instruction 1
FETCH –DECODE – EXECUTE – Instruction 2
FETCH –DECODE – EXECUTE – Instruction 3
Jobs
Time

Pipelined Version



Effects of Pipelining

1. The registers are being efficiently used since they are no longer idle
2. At least three instructions are being dealt with concurrently. This will consequently lead to a reduction in the execution time especially if it's a linear program.
3. If there is jump instruction, several issues crop up since the incorrect instructions are in the pipe line waiting to be executed. Hence, each time the sequence of instructions alters, the pipe line has to be cleared and the process started all over again.

RISC Pipelines

A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

i. fetch instructions from memory
ii. read registers and decode the instruction
iii. execute the instruction or calculate an address
iv. access an operand in data memory
v. write the result into a register

RISC Disadvantages

- ✓ RISC architecture put greater burden on the software by making the hardware simpler

RISC and CISC Interrupt Handling

There are 3 main components related to the performance of processing interrupts.

1. **Interrupt Latency** -Time elapsed between the interrupt request is received by processor and time the latter takes action to process the interrupt service routine.
2. **Interrupt Processing Time** – amount of time the CPU spends on saving the machine state of the current interrupted job and diverting execution to the interrupt service routine
3. **State saving overhead** – time taken to save machine registers which were not automatically saved by the interrupt processing logic but which must be saved in order for the interrupt service routine to function.

Note: The cost of restoring all machine state and interrupted routines are considered in determining overall system performance.

Interrupt Response Latency

The response latency is degraded by CISC machines since they may have instructions which take a very long time to execute.

RISC can have a very quick response to interrupt response latency since being only a single cycle long instructions , few clock cycles elapse before interrupt request is acknowledged and processed.

Hardware – Parallel Processing

Parallel Processing

It is an alternative approach to the classical Von Neumann Architecture. It simply involves the use of many independent processors working in parallel on the same program or multiple computational threads. It has to be noted that programs running on such systems need to be specifically written to be used in parallel environment. Ideally, parallel processing makes programs run faster because there are more engines (CPUs or cores) running it.

Advantage	Disadvantage
▪ Multicore Chips present new opportunities ▪ Faster execution time , so higher throughput ▪ Solve larger / complex problems (Wen Search engines)	▪ It is not suitable for low power and mobile devices ▪ It requires more hardware along with more power

Key Elements of Parallel Processing

- Synchronisation (Coordination of concurrent tasks)
- Locking (ways of synchronising tasks)
- Speedup (Extent to which more hardware can perform same task in less time)and Scale Up (factor by which more work can be done in same amount of time period by a system n times larger)
- Messaging (communication between nodes)

Factors advocating the use of parallel processing applications

- Applications which are time sensitive or which requires fast response time.
- Applications which stores large amount of data such as databases (e.g. Oracle database)
- Complex real world phenomena modelling and simulation. (Climate change, plate tectonics, vehicle assembly, rush hour traffic etc., galaxy formation...)

E.g. Include

1. **Science and Engineering** (Computer Science, Bioscience, Physics ,Mechanical Engineering)
2. **Industrial and Commercial** (Databases, Medical Imaging and diagnosis , collaborative work environment)
3. **Weather Forecasting**

Parallel Processing Approach

There are different ways to classify parallel computers but the most widely used is called Flynn's Taxonomy. It depends on parallelism it exhibits with its instruction stream and data stream.

As we know, a sequence of instructions (instruction stream) manipulate a sequence of operands (the data stream).The instruction stream (I) and the data stream (D) are classified as either single (S) or multiple (M).

Hence, there are four classifications according to Flynn:

1. **SISD** (Single Instruction Stream Single Data Stream)
2. **SIMD** (Single Instruction Stream Multiple Data Stream)
3. **MISD** (Multiple Instruction Stream Single Data Stream)
4. **MIMD** (Multiple Instruction Stream Multiple Data Stream)

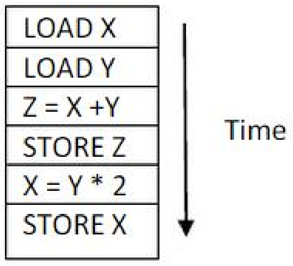
SISD (Single Instruction Single Data)

The computer tackles and processed each task in order and is often coined as sequential. Hence, they aren't capable of performing parallel processing on their own.

Characteristics

✓ It uses a Single CPU system (Uniprocessor)

- ✓ Concurrent processing is allowed (instruction prefetching, pipelined execution of instructions)
- ✓ Concurrent execution is allowed (independent concurrent tasks can execute different sequences of operations .E.g. I/O controllers are independent of CPU)
- ✓ E.g. Older generation mainframes, minicomputers, workstations and single processor PCs.



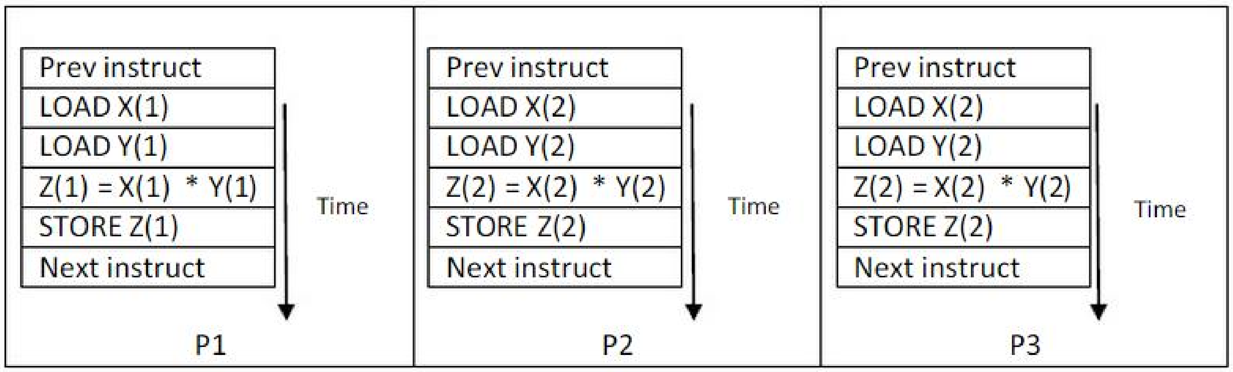
SIMD (Single Instruction Multiple Data)

These computers have multiple processors which follow the same set of instruction but each processor input distinct data into those instructions. This can be useful in analysing large piece of data based on same criteria. Thus, it cannot be used for complex computational problems.

Most modern computers with graphics processor units (GPU) use SIMD instructions and execution units.

Characteristics

- ✓ One instruction stream is broadcast to all processors
- ✓ Each processor, also called a *processing element* (or PE), is usually simplistic and logically is essentially an ALU;
- ✓ PEs do not store a copy of the program nor have a program control unit.
- ✓ Individual processors can remain idle during execution of segments of the program (based on a data test).
- ✓ All active processor executes the same instruction synchronously, but on different data
- ✓ Technically, on a memory access, all active processors must access the *same location* in their local memory.
- ✓ This requirement is sometimes relaxed a bit.
- ✓ The data items form an array (or vector) and an instruction can act on the complete array in one cycle.
- ✓ This architecture is also caller processor array by Quinn. E.g. ILLIAC IV (1974) , The STARAN and MPP, MasPar , Connection Machine CM2 and others.



Tip to view a SIMD machine

1. Think of soldiers all in a unit
2. The commander selects some solders as active , let's say first row
3. The commander issues out an order to all the active soldiers who execute the order synchronously
4. The remaining soldiers do not execute orders until they are re-activated.

Advantage	Disadvantage
▪ Less hardware than MIMDs as they have only one control unit ▪ Less memory needed than MIMD ▪ Much less time required for communication between Pes and data movement. ▪ Easier to program, understand and debug SIMD application	▪ SIMD seem to have a data parallel orientation though not all problems are data parallel ▪ Speed drops for conditionally executed branches ▪ Don't adapt to multiple users well

MISD (Multiple Instruction Single Data)

These computers have multiple processors with each one using a distinct algorithm but same shared input data. Thus, MISD computers can analyse the same set of data using several different operations at the same time. The number of operations depends upon the number of processors.

- ✓ Only few such computers might have ever existed.
- ✓ Quinn argues that a systolic array is an example of a MISD structure

Example of use: Multiple cryptography algorithms attempting to rack a single coded message

MIMD (Multiple Instruction Multiple Data)

Such computers have multiple processors where each one is capable of accepting its own instruction stream independently from the others. Each processor also pulls data from a separate data stream.

E.g. Current Supercomputers, network parallel computer clusters, multi-processor SMP computers, Multi-core PCs

Characteristics

- ✓ Processors are asynchronous, since they can independently execute different programs on different data sets.
- ✓ Communications are handled either through shared memory (**multiprocessors**) or by use of message passing (**multicomputer**)
- ✓ MIMD's have been considered by most researchers to include the most powerful and least restricted computers.

System Software – Purpose of an Operating System

Types of User Interface

There are several ways a user can interact with a program for use in different situations and for different types of users. The common types of interface are:

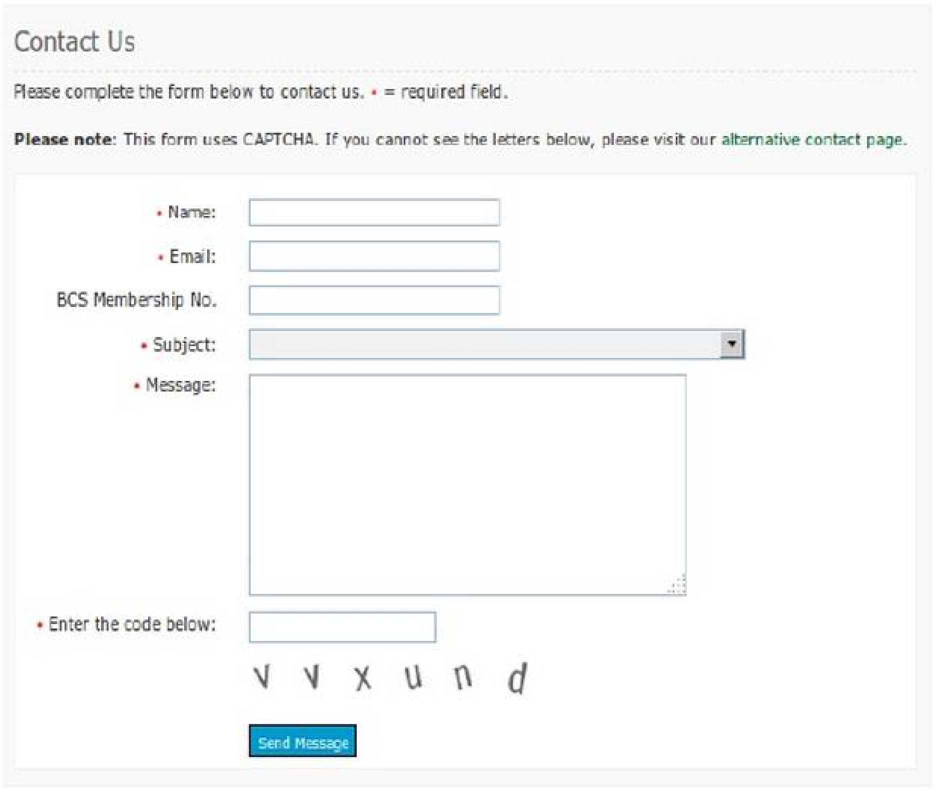
i. Form based
ii. Menu based
iii. Graphical User Interface (GUI)
iv. Natural Language
v. Command Line

Form Based Interface

When a user is required to enter data such, as for instance, data about customer records in a hotel management system, it is common to have a form displayed for data capture. It has the following feature:

- Specified area for the data , data entered in order or in format specified
- **Use:** operator taking information over phone
- It does not allow information to be missed out and is simple to use

An example of a form is shown.


Contact Us

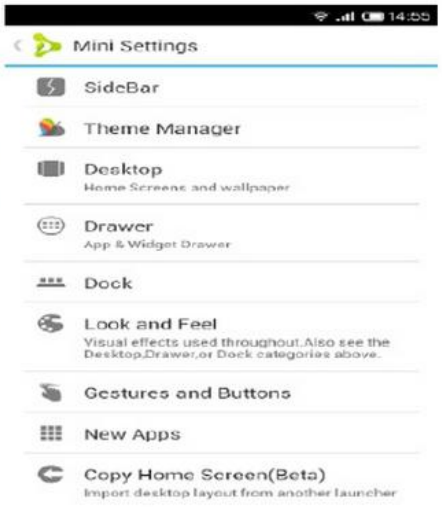
Please complete the form below to contact us. • = required field.

Please note: This form uses CAPTCHA. If you cannot see the letters below, please visit our alternative contact page.

• Name:
• Email:
BCS Membership No.
• Subject:
• Message:
• Enter the code below:

Menu Based Interface

Users are presented with a choice of programs that they can run. A list of choices is made available followed by a further set of choices based on the first choice and so on until the result is obtained. The choices are restricted in numbers. It restricts access to the computer system and usually suits a touch screen device.



14:55

Mini Settings

SideBar

Theme Manager

Desktop
Home Screens and wallpaper

Drawer
App & Widget Drawer

Dock

Look and Feel
Visual effects used throughout.Also see the Desktop,Drawer,or Dock categories above.

Gestures and Buttons

New Apps

Copy Home Screen(Beta)
Import desktop layout from another launcher

Use :

- Menu based interfaces are used in situations where the operator tends not to know what the options are that are available.

Graphical User Interface

A graphical user interface (GUI) presents the user commands and programs in the form of icons. Another feature of GUI offers user a view of the program on the screen surrounded by a border window. It provides wimp (windows, icon, menu, pointer controlled) which is usually meant for inexperience users. Choices are selected by the user using some sort of pointing device to indicate choice, typically this would be a mouse or touch screen.

Graphical User Interface

A graphical user interface (GUI) presents the user commands and programs in the form of icons. Another feature of GUI offers user a view of the program on the screen surrounded by a border window. It provides wimp (windows, icon, menu, pointer controlled) which is usually meant for inexperience users. Choices are selected by the user using some sort of pointing device to indicate choice, typically this would be a mouse or touch screen.

Advantage	Disadvantage
▪ Suitable for novice to learn and use a system	▪ May restrict some program or options ▪ Some GUI program may contain bugs and not perform appropriately compared to their command line equivalent specially when program path changes.

Natural Language

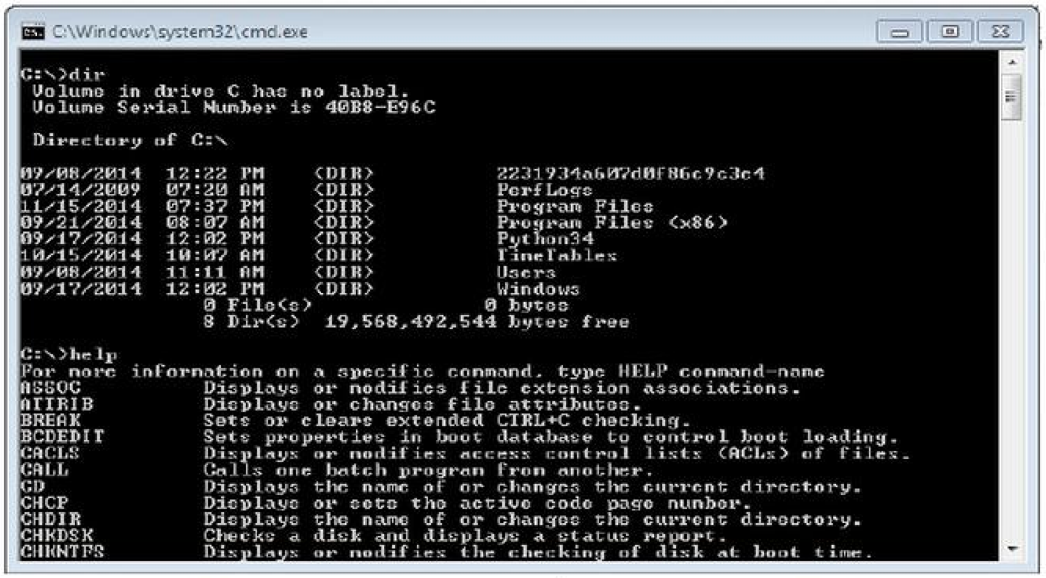
This type of interface provides user a series of question to which the user will respond. Normally such system restrict itself to questions to which only sensible answers are the ones that it knows. In case a user respond with an unexpected response, a message is produced to force the user to make another attempt. E.g. Search Engine can natural language queries.

Advantage

- Can be used for disabled person

Command Line

Command driven interface uses words or sentences to tell the computer what to do. It is used by technical users to run certain programs, which is not on the menu icon. Besides it also allows access to the whole system if one knows what commands are available. E.g. Microsoft command line, Linux terminal mode, Mac Terminal



```
C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is 40B8-E96C

 Directory of C:\

09/08/2014  12:22 PM    <DIR>          2231934a607d0f86c9c3c4
07/14/2009  07:20 AM    <DIR>          PerfLogs
11/15/2014  07:37 PM    <DIR>          Program Files
09/21/2014  08:07 AM    <DIR>          Program Files (x86)
09/17/2014  12:02 PM    <DIR>          Python34
10/15/2014  10:07 AM    <DIR>          TimeTables
09/08/2014  11:11 AM    <DIR>          Users
09/17/2014  12:02 PM    <DIR>          Windows
               0 File(s)              0 bytes
               8 Dir(s)  19,568,492,544 bytes free

C:\>help
For more information on a specific command, type HELP command-name
ASSOC          Displays or modifies file extension associations.
ATTRIB         Displays or changes file attributes.
BREAK          Sets or clears extended CTRL+C checking.
BCDEDIT        Sets properties in boot database to control boot loading.
CACLS          Displays or modifies access control lists (ACLs) of files.
CALL           Calls one batch program from another.
CD             Displays the name of or changes the current directory.
CHCP           Displays or sets the active code page number.
CHDIR          Displays the name of or changes the current directory.
CHKDSK         Checks a disk and displays a status report.
CHKNTFS        Displays or modifies the checking of disk at boot time.
```

Advantage	Disadvantage
▪ All programs available can be run and hence is more open compared to other interfaces. Example menu driven only allow programs that are on offer to be run. ▪ Does not consume lot of computer resources in terms of memory and processor ▪ Reduce the risk of security beach especially on graphical user interface bugs. (That is why server OS offer command Line mode for Advance administrators)	▪ It takes longer to type commands ▪ Requires the user to know what commands are available ▪ Often we get typing errors, if commands are long

Multitasking

It is defined as the running of multiple tasks simultaneously. Example a user might be browsing the web while listening to music and printing a document in the background. The main function of a processor is to execute program instructions.

Process

A process is defined as an instance of a program running on a computer. It can also be considered as a unit of resource allocation. A process consists of the following: Executable program, Program counter, Program data and stack, Stack pointer and other registers , number of resources and other information required for the program to run.

Process Model

Each runnable program on the computer is organised into processes and each user view the processor as executing his process sequentially. In fact the CPU switches back and forth from process to process with resource allocation to promote multiprogramming.

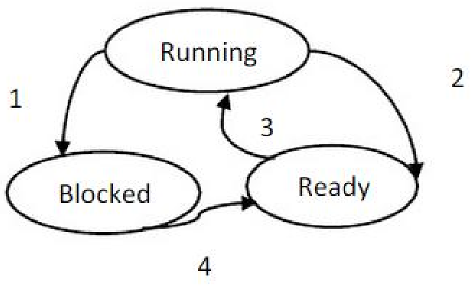
Multiprogramming

It is defined as the running of several processes concurrently loaded into main memory that give the illusion that more than one program is executing at the same time.

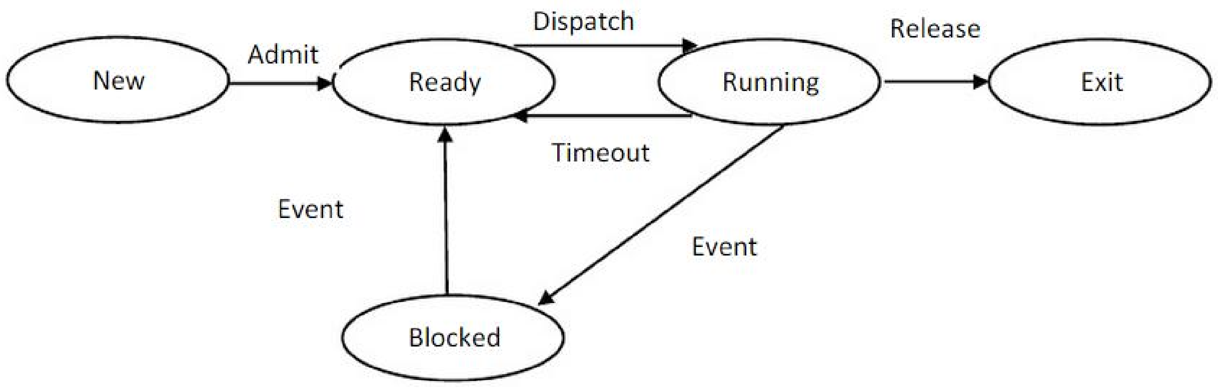
Process States

3 states process model

The possible process states are: running, blocked and ready. The diagram above shows the transitions between states.



5 process state model



Appropriate order of tasks

One of the main tasks of the OS is to organise process or jobs in an appropriate manner so to maximise the processor usage. Sometimes even priorities of certain job are considered,

Classification of Process or Jobs

A process or job may be classified as:

i. Input or Output bound jobs
ii. Processor bound jobs (CPU bound Jobs)

Input /Output Bound Jobs

These are jobs that need little processing but do need to use the peripheral devices considerably. Example: counting the number of lines in a file

Processor Bound Jobs

These are jobs that need a considerate amount of processor time and but little use of the various peripheral devices. Example: multiplying small matrices and printing the result or 3D-graphics calculation

Priority

If a CPU bound job is allocated priority over an I/O bound job the latter may have to wait for a long period of time before being serviced.
It is implemented by having multiple ready queues to represent each level of priority. The scheduler will always choose a process of higher priority over one of lower priority. Hence, lower priority jobs may suffer from starvation. Thus, there should be a mechanism for a process to change its priority based on its age or execution history.

Processor Scheduling

Objectives of Scheduling

- Be fair to all users to allocate processor time and ensure no one starves
- Maximise utilisation : that is keep I/O devices busy
- Maximise throughput : Maximise job or time
- Minimize Latency : that is provide acceptable response time or job completion time
- Prevent deadlock
- Prevent the system from failing if it becomes overloaded

High Level Scheduler (HLS)

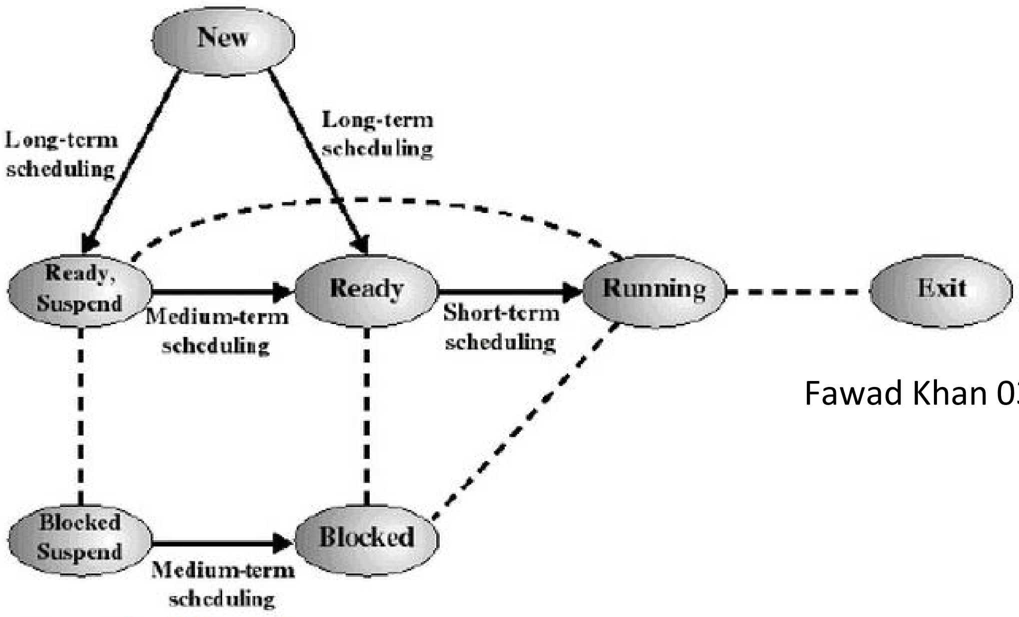
- Its role consist of deciding which process to admit and placing new jobs in the ready queue.

Medium Level Scheduler (MLS)

- Its role consist of deciding which process to swap in or out of the main memory and backing store.

Low Level Scheduler (LLS)

- Its role consists of deciding which ready process to execute next and also cater for the movement jobs in and out of the ready state.



Characteristic Scheduling Policies

A. **Selection function** : the latter determines which process in the ready queue is chosen next for execution
B. **Decision mode** : specifies the instant in time at which the selection function is used

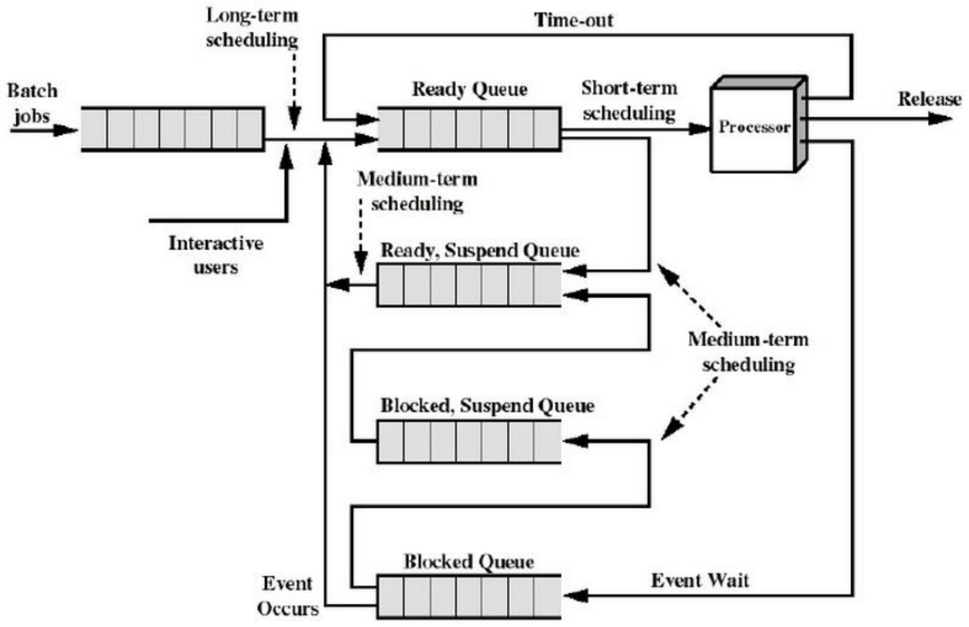
Non Pre-emptive Scheduling

Once a procession is in the running state, it will continue until it terminates or blocks itself for I/O. This might be the case specially when a job has terminated or needs an I/0 device. Hence, it simply follows the queue to be processed.

Pre-Emptive Scheduling

Currently running process may be interrupted and moved to the Ready state by the OS via the Low Level Scheduler. It allows for better service since one process cannot monopolise the process for a long period of time.

Queuing Diagram for Scheduling



Common Scheduling Policies

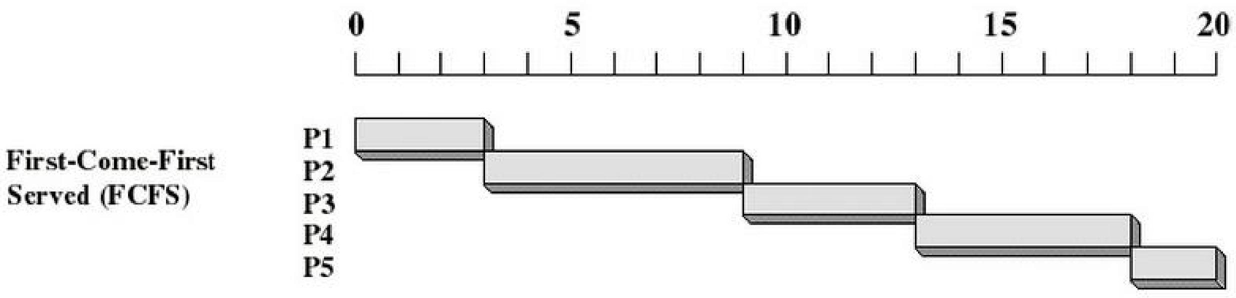
Let us use the following to understand the various scheduling policies

Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

Service Time = total processor time needed in one (CPU-I/O Cycle)

Jobs with long service time are CPU-bound jobs and are referred to as “long jobs”

First Come First Served (FCFS)



The first process having joined the queue is the first one to actually enter the running state. Normally this scheme favours long jobs. Average waiting time depends on arrival order.

Selection Function : The process that has been waiting the longest in the ready queue

Decision Mode : Non Pre-emptive – a process run until it blocks itself

Advantage

- ✓ Really simple

Drawbacks

- ✓ A process that does not perform any input or output will monopolise the processor
- ✓ It favours CPU bound processes (I/O bound processed will have to wait for processor bound job to terminate and even in case where their I/O are completed)

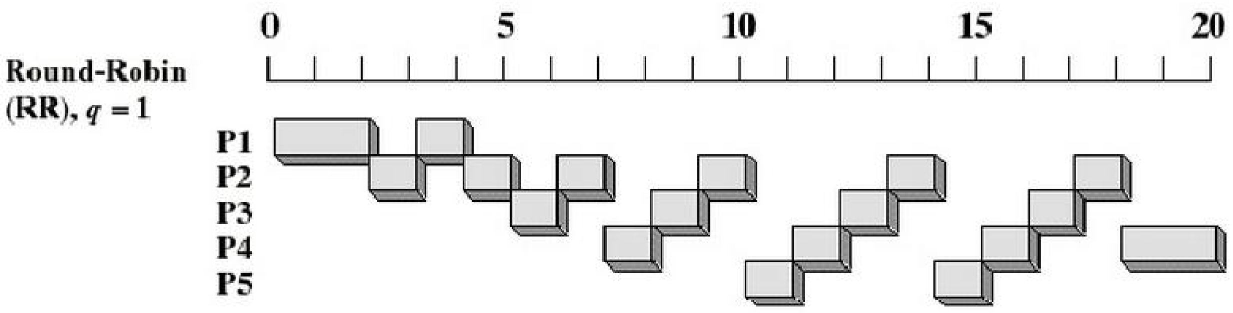
LIFO (Last in First Out)

New jobs arriving are placed at the head of ready queue. This improves response time for newly created threads.

Drawback

- ✓ It may lead to starvation – early processes may never get the CPU

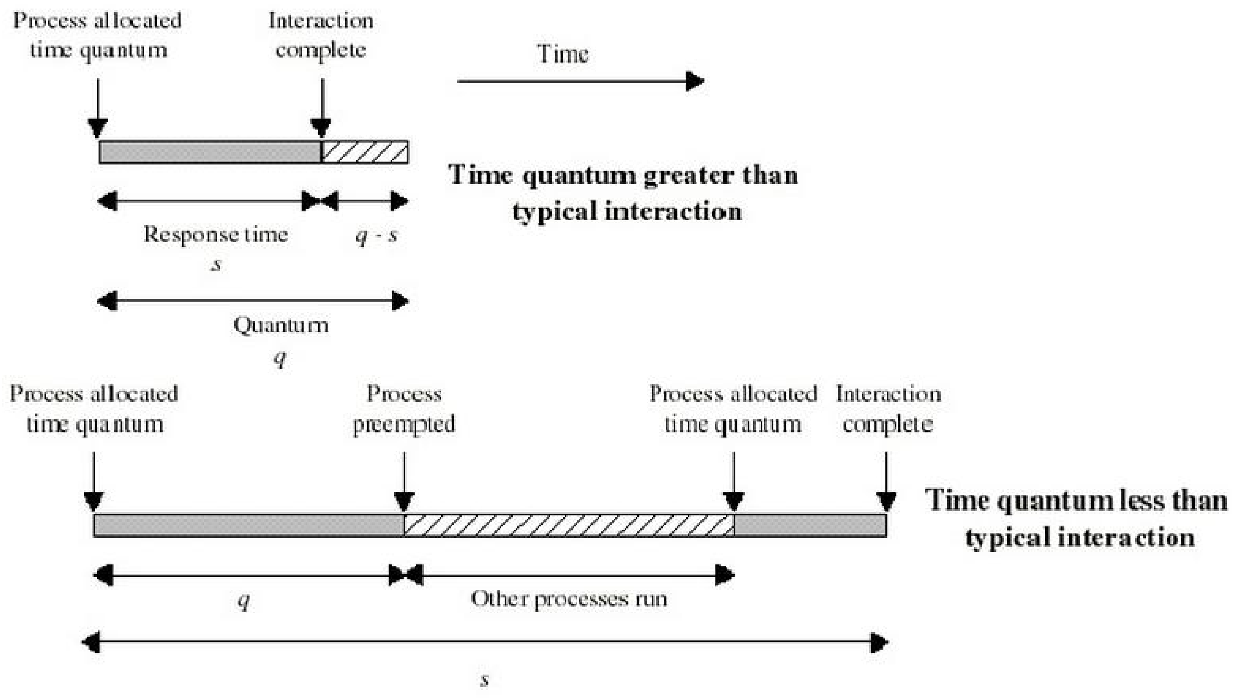
Round Robin (RR)



Selection Function : same as FCFS

Decision mode : pre-emptive – a process is allowed to run until the time slide (quantum – 10/100 ms) period has expired and then a clock interrupt occurs and the running process is put on the ready queue.

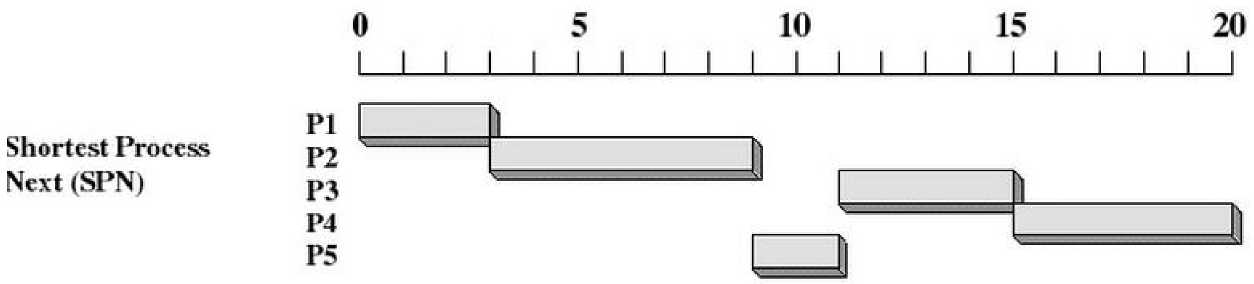
The time quantum for Round Robin must be substantially larger than the time required to handle the clock interrupt and dispatching. Moreover, the slice should be larger than typical interaction (but not much more to avoid penalising I/O bound jobs)


Process allocated time quantum
Interaction complete
Time
Time quantum greater than typical interaction
Response time s
q - s
Quantum q
Process allocated time quantum
Process preempted
Process allocated time quantum
Interaction complete
Time quantum less than typical interaction
q
Other processes run
s

Drawback

✓ Still favours CPU-bound process

Shortest Process Next (SPN)



Selection Function : the process / job with the shortest expected CPU burst time is chosen

Decision mode : non-pre-emptive

I/O bound processes will be picked first. However we need to estimate the required processing time (CPU burst time) for each process.

Drawback

- ✓ Possibility of starvation for longer processes as long as there is a steady supply of shorter processes
- ✓ Lack of pre-emption is not suited in a time sharing environment
- ✓ SPN implicitly incorporates priorities , i.e. shortest jobs are given preferences
- ✓ The next algorithm penalises directly longer jobs

Multilevel Feedback Scheduling

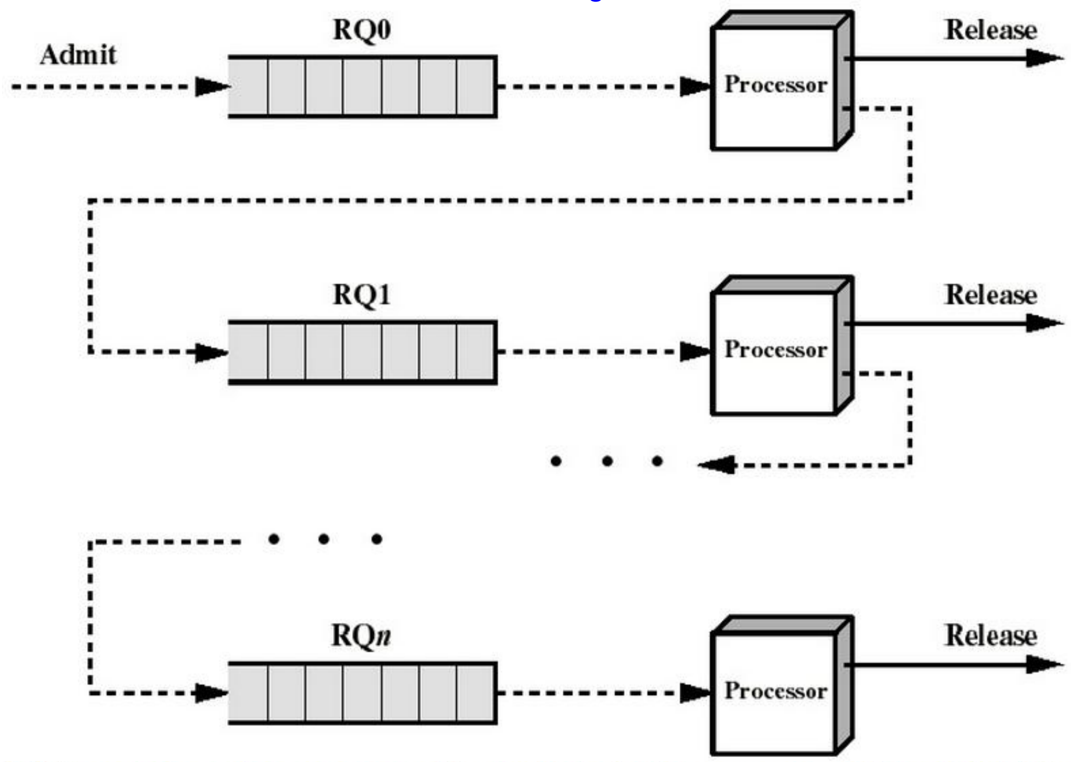
It uses pre-emptive scheduling with dynamic priorities. There are several ready to execute queues with decreasing priorities.

P(RQ0) > P(RQ1) >.... >P(RQn)

New processes are placed in RQ0 and when they reach the time quantum, they are placed in RQ1. If again they reach it, they are placed in RQ2.. until they reach RQn.

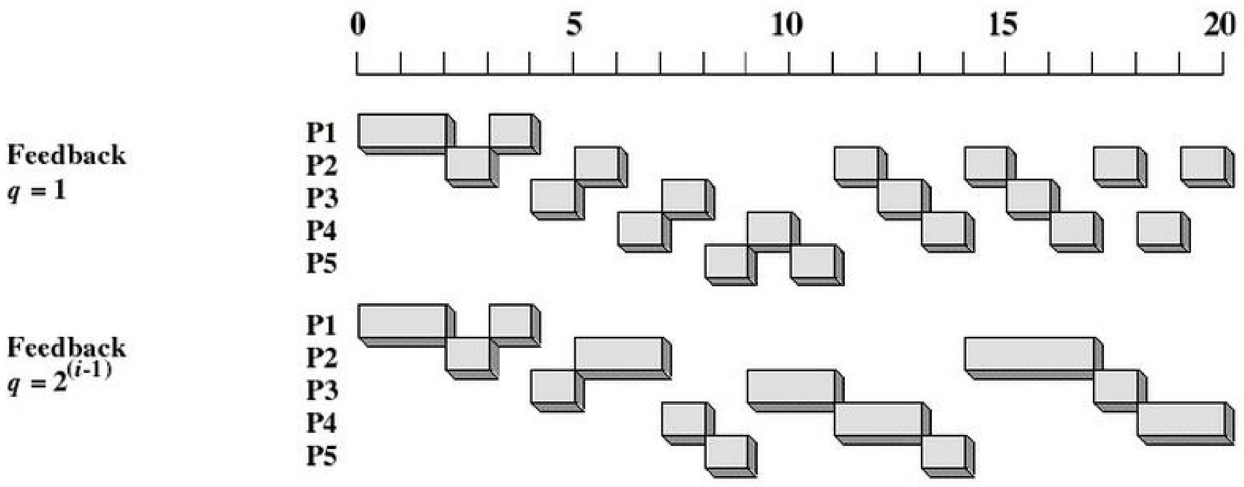
I/O bound jobs will stay in higher priority queues while CPU bound jobs will drift downward. The dispatcher chooses a process for execution in RQi only if RQi-1 to RQ0 are empty.

Consequently, long jobs may starve



FCFS is used in each queue except for lowest priority queue where Round Robin in used.

Feedback scheduling time slice



If a fixed quantum time is allocated , the turnaround time of longer jobs becomes alarming. To compensate we can increase the time quantum according to the depth of the queue.

Note : longer processes may still suffer starvation.

Solution: promote a process to higher priority after some time

Which Algorithm to choose?

Depends on:

- ✓ System workload (extremely variable)
- ✓ Hardware support for the dispatcher
- ✓ Evaluation method used
- ✓ Relative Weight of performance criteria (CPU utilisation , response time , throughput)

Importance of Process / Type of Process:

Usually safety critical jobs, online and real time applications are given high priorities

Allocating Priorities

- Importance of Job /Type of Job
- Amount of time already waited
- Size of job
- Amount of peripheral time
- Input / Output high priority
- Amount of processor time already given
- Necessary response time

Interrupts

The simplest way of obeying instructions. An interrupt is a signal sent to the processor by devices or programs indicating that they require the attention of the processor.

Types of Interrupts

A. **Hardware Generated Interrupts**

- Printer to inform the processor that it is out of paper or paper jam
- Reset button actioned by the user
- Keyboard to indicate that data has been entered and requires saving
- Mouse – mouse click to refresh current screen

System Clock – timer signal

B. **Software or Program Interrupt** – Division by zero, a file not being found

C. **Clock interrupt** – must complete the current fetch execute cycle

Managing Interrupts

1. Sequence of steps the processor would carry out after receiving an interrupt.
2. Source of interrupt is determined
3. Interrupts of lower priority are masked out
4. Program Counter contents are saved
5. Contents of all other registers are saved on the stack
6. Interrupt Service Routine (ISR) is loaded
7. ISR code is run

8. Contents of registers are restored
9. PC contents are restored
10. All lower priority interrupts are restored
11. Next process is resumed

Memory Management

It is essential to understand that for a process or job to use the CPU, the job must be store in the computer's RAM. There may be multiple jobs stored simultaneously in the main memory with the space available but only one of the job can be processed.

The main memory can be viewed as a continuous array as shown below

OS
JOB1
JOB 4
JOB 3
Free Space

Fragmentation of Memory

There are several strategies used to allocate memory to processes and jobs. When a job terminates look at the available gaps and load next job, when a job terminates, moves all other jobs to create one large hole. Look for the most suitable job from the 'wait list'

Segmentation

- The program is divided into segments of variable size or Logical units
- Hence , not all the program needs to be loaded at start up

Virtual Memory

- It is a technique that uses secondary storage space to behave as main memory.

Partitioning

Using this technique, the memory is divided into fixed areas. Hence , each partition is used for a particular job.

Paging

The program or job is divided into a number of pages. Each page is of a fixed size. The main memory also is divided into a number of page frames of the same size.Hence, pages are swapped continually in and out of memory as they are required. Thus, not all the program needs to be loaded at start up.

Consider the Following example:

0	OS		**JOB**	**Memory**	**CPU Burst**
40K			J1	60K	10
			J2	100K	5
			J3	30K	20
			J4	70K	8
			J5	60K	15
			J6	50K	9
260K					

Assume each page of size 10k. Assume Memory Available is 260 K for simplicity.

Job1(J1)	**Job2(J2)**	**Job3(J3)**	**Job4(J4)**	**Job 5(J5)**	**Job 6(J6)**	Memory
PAGE 6	PAGE 10	PAGE 3	PAGE 7	PAGE 6	PAGE 5	OS 1
PAGE 5	PAGE 9	PAGE 2	PAGE 6	PAGE 5	PAGE 4	OS 2
PAGE 4	PAGE 8	PAGE 1	PAGE 5	PAGE 4	PAGE 3	OS 3
PAGE 3	PAGE 7		PAGE 4	PAGE 3	PAGE 2	OS 4
PAGE 2	PAGE 6		PAGE 3	PAGE 2	PAGE 1	
PAGE 1	PAGE 5		PAGE 2	PAGE 1		
	PAGE 4		PAGE 1			
	PAGE 3					
	PAGE 2					
	PAGE 1					

After Job 1, Job 2 AND Job 3 have joined memory the status will be as follows

Job1(J1)	Job2(J2)	Job3(J3)	Job4(J4)	Job 5(J5)	Job 6(J6)	Memory
PAGE 6	PAGE 10	PAGE 3	PAGE 7	PAGE 6	PAGE 5	OS 1
PAGE 5	PAGE 9	PAGE 2	PAGE 6	PAGE 5	PAGE 4	OS 2
PAGE 4	PAGE 8	PAGE 1	PAGE 5	PAGE 4	PAGE 3	OS 3
PAGE 3	PAGE 7		PAGE 4	PAGE 3	PAGE 2	OS 4
PAGE 2	PAGE 6		PAGE 3	PAGE 2	PAGE 1	J1 - PAGE 6
PAGE 1	PAGE 5		PAGE 2	PAGE 1		J1 - PAGE 5
	PAGE 4		PAGE 1			J1 - PAGE 4
	PAGE 3					J1 - PAGE 3
	PAGE 2					J1 - PAGE 2
	PAGE 1					J1 - PAGE 1
						J2 - PAGE 10
						J2 - PAGE 9
						J2 - PAGE 8
						J2 - PAGE 7
						J2 - PAGE 6
						J2 - PAGE 5
						J2 - PAGE 4
						J2 - PAGE 3
						J2 - PAGE 2
						J2 - PAGE 1
						J3 - PAGE 3
						J3 - PAGE 2
						J3 - PAGE 1
						Free Space

Let us say, Job 2 terminates and releases 100K (10 Pages) of free space and Job 4 is ready and is given then space to load in memory

Job1(J1)	Job2(J2)	Job3(J3)	Job4(J4)	Job 5(J5)	Job 6(J6)	Memory
PAGE 6	PAGE 10	PAGE 3	PAGE 7	PAGE 6	PAGE 5	OS 1
PAGE 5	PAGE 9	PAGE 2	PAGE 6	PAGE 5	PAGE 4	OS 2
PAGE 4	PAGE 8	PAGE 1	PAGE 5	PAGE 4	PAGE 3	OS 3
PAGE 3	PAGE 7		PAGE 4	PAGE 3	PAGE 2	OS 4
PAGE 2	PAGE 6		PAGE 3	PAGE 2	PAGE 1	J1 - PAGE 6
PAGE 1	PAGE 5		PAGE 2	PAGE 1		J1 - PAGE 5
	PAGE 4		PAGE 1			J1 - PAGE 4
	PAGE 3					J1 - PAGE 3
	PAGE 2					J1 - PAGE 2
	PAGE 1					J1 - PAGE 1
						J4 - PAGE 7
						J4 - PAGE 6
						J4 - PAGE 5

J4 - PAGE 4
J4 - PAGE 3
J4 - PAGE 2
J4 - PAGE 1
Free Space
Free Space
Free Space
J3 - PAGE 3
J3 - PAGE 2
J3 - PAGE 1
Free Space
Free Space

The gap consists of pages into which JOB 4 (7 pages) will fit leaving 3 pages + 2 original page free.

Now suppose Job 5 is ready and Job 1 Terminates.

Job1(J1)
PAGE 6
PAGE 5
PAGE 4
PAGE 3
PAGE 2
PAGE 1

Job2(J2)
PAGE 10
PAGE 9
PAGE 8
PAGE 7
PAGE 6
PAGE 5
PAGE 4
PAGE 3
PAGE 2
PAGE 1

Job3(J3)
PAGE 3
PAGE 2
PAGE 1

Job4(J4)
PAGE 7
PAGE 6
PAGE 5
PAGE 4
PAGE 3
PAGE 2
PAGE 1

Job 5(J5)
PAGE 6
PAGE 5
PAGE 4
PAGE 3
PAGE 2
PAGE 1

Job 6(J6)
PAGE 5
PAGE 4
PAGE 3
PAGE 2
PAGE 1

Memory
OS 1
OS 2
OS 3
OS 4
Free Space
Free Space
Free Space
Free Space
Free Space
Free Space
J4 - PAGE 7
J4 - PAGE 6
J4 - PAGE 5
J4 - PAGE 4
J4 - PAGE 3
J4 - PAGE 2
J4 - PAGE 1
Free Space
Free Space
Free Space
J3 - PAGE 3
J3 - PAGE 2
J3 - PAGE 1
Free Space

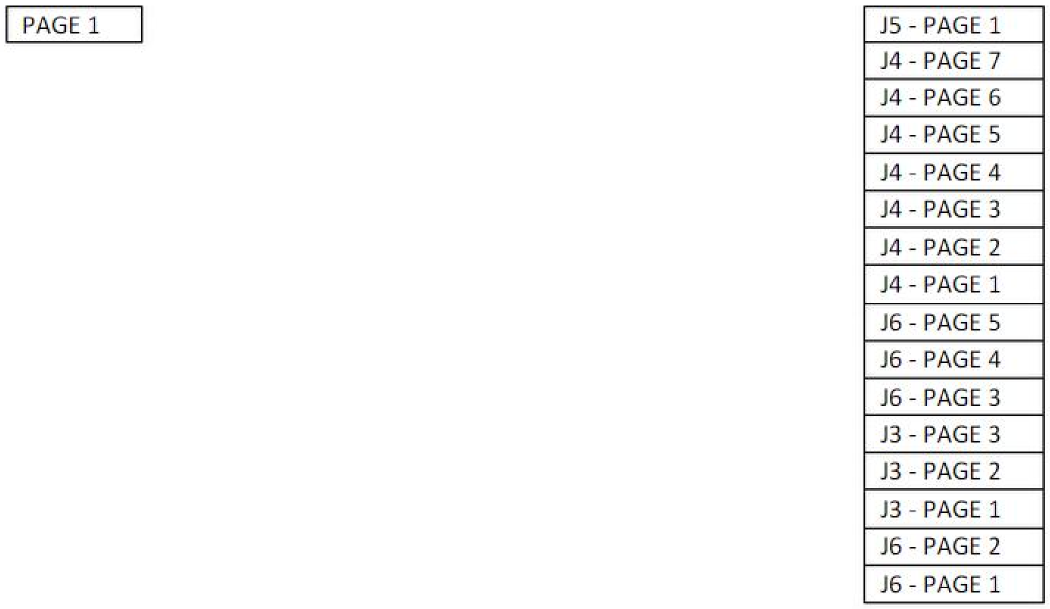
Free Space

Let us say Job 5 and 6 are ready to be put in memory. Job 5 access the free space as shown:

Job1(J1)	**Job2(J2)**	**Job3(J3)**	**Job4(J4)**	**Job 5(J5)**	**Job 6(J6)**	Memory
PAGE 6	PAGE 10	PAGE 3	PAGE 7	PAGE 6	PAGE 5	OS 1
PAGE 5	PAGE 9	PAGE 2	PAGE 6	PAGE 5	PAGE 4	OS 2
PAGE 4	PAGE 8	PAGE 1	PAGE 5	PAGE 4	PAGE 3	OS 3
PAGE 3	PAGE 7		PAGE 4	PAGE 3	PAGE 2	OS 4
PAGE 2	PAGE 6		PAGE 3	PAGE 2	PAGE 1	J5 - PAGE 6
PAGE 1	PAGE 5		PAGE 2	PAGE 1		J5 - PAGE 5
	PAGE 4		PAGE 1			J5 - PAGE 4
	PAGE 3					J5 - PAGE 3
	PAGE 2					J5 - PAGE 2
	PAGE 1					J5 - PAGE 1
						J4 - PAGE 7
						J4 - PAGE 6
						J4 - PAGE 5
						J4 - PAGE 4
						J4 - PAGE 3
						J4 - PAGE 2
						J4 - PAGE 1
						Free Space
						Free Space
						Free Space
						J3 - PAGE 3
						J3 - PAGE 2
						J3 - PAGE 1
						Free Space
						Free Space

Job 6 is ready, there are enough available space but they are not contiguous. Hence, using paging, Job 6 is split in five fixed pages size and put in memory. It should be noted that jobs do not have to occupy continuous pages in memory using the paging technique.

Job1(J1)	**Job2(J2)**	**Job3(J3)**	**Job4(J4)**	**Job 5(J5)**	**Job 6(J6)**	Memory
PAGE 6	PAGE 10	PAGE 3	PAGE 7	PAGE 6	PAGE 5	OS 1
PAGE 5	PAGE 9	PAGE 2	PAGE 6	PAGE 5	PAGE 4	OS 2
PAGE 4	PAGE 8	PAGE 1	PAGE 5	PAGE 4	PAGE 3	OS 3
PAGE 3	PAGE 7		PAGE 4	PAGE 3	PAGE 2	OS 4
PAGE 2	PAGE 6		PAGE 3	PAGE 2	PAGE 1	J5 - PAGE 6
PAGE 1	PAGE 5		PAGE 2	PAGE 1		J5 - PAGE 5
	PAGE 4		PAGE 1			J5 - PAGE 4
	PAGE 3					J5 - PAGE 3
	PAGE 2					J5 - PAGE 2



Consequently, it is essential to keep a table showing which memory pages are used for the job pages. Let us say that each address used in a job or process consists of a page number and distance required location is from the start of the page, an appropriate algorithm might be used.

Virtual Memory

Definition: *It is a technique used when there is not enough main memory available to store the pages requiring the CPU. Hence, allows a piece of software that is so big that it will not fit into memory to be used. Part of the hard disk is used to act as virtual main memory. Eventually, the pages are transferred to main memory as and when they are required.*

Virtual memory serves two purposes. Firstly, it allows us to extend the use of physical memory by using disk. Secondly, it allows us to have memory protection, because each virtual address is translated to a physical address.

Advantage of Virtual Memory

- ✓ The logical address space can be much larger than physical address space
- ✓ More process can be in physical memory partially at any point in time

Use

- ✓ It is useful for large datasets and for compilation models since today we do have large memory.

Disk Thrashing

This is a discrete phenomenon occurs when the physical memory is too small to hold the working sets of all the processes running. The outcome of thrashing is that pages always spend their time waiting for the backing store to fetch their pages.

In order to eliminate thrashing, we must drop degree of multiprogramming. Thus, we swap all of a process out to backing store and then suspend the process. We can also come up with a Page Fault Frequency based solution whereby if a page faults too often, then we need to give it more physical page frames

Thrashing causes a severe drop in performance of the processor especially if the memory is very full, this process will end up using up a lot of the processor's time.

Usually some of the factors that causes thrashing are page faults, low memory space, process requiring large memory space and OS monitoring utilisation.

System Software – Virtual Machine

Virtual Machine (VM)

Normally virtual machine emulate a particular computer system. They are classified according to the degree to which they implement the functionality of the targeted real machine. Process virtual machine in OS are designed to execute a single computer program by providing abstract and platform independent program execution environment with the view to promote portability and flexibility. The software running inside a virtual machine is limited to the resources and abstraction provided by the VM.

The concept of virtual machine resides with the OS whereby the user is hidden from the complexity of the hardware layers behind the OS software. The lower layers which are user interface and Application programming interface calls the services of the OS. The following is a list of layers that hide most complexities:

- Hardware
- Kernel – (OS bridge between applications and hardware resources)
- Device drivers
- Memory Management
- Processor Management
- File Management
- Input / Output Management
- User Interface / API

Note: API makes it possible to run same piece of software on different computers with the layer of software responsible for calling the services of the OS.

Translation Process

A compiler usually translate a high level language by applying the following 3 stages:

- ✓ Lexical Analysis
- ✓ Syntax Analysis
- ✓ Code Generation

The process of breaking down the source program into parts is called parsing.

Lexical Analysis

During this stage, the lexical analyser uses the source program as input. Unnecessary characters, comments and whitespaces are removed. The strings of characters are grouped to form keywords or reserved words. The reserved words are checked for validity against a dictionary or table of keywords. The Identifiers are placed in symbol table. Errors for either invalid keywords or identifiers are reported. Finally keywords, reserved words and identifiers are replaced by tokens. Thus, the final output is a token string.

Syntax Analysis

The syntax analyser used the output produced from the lexical analysis stage. The compiler will make reference to meta-language statements (e.g. BNF- Backus Naur Form) by scanning the tokenized version of the program. The statements describe all possible forms of construction for each keyword. They are checked against similar meta-language rules which exist for permitted identifier names and grammar of statements. Finally, Errors in either the statement composition or identifier names are reported. Then it produces code ready for the code generation stage
How syntax errors are identified during compilation?

In lexical analysis stage keywords are identified by comparing to list of accepted words. Here, the format of instruction or token string is compared to forms for acceptable expressions and statements as defined by the Meta language used. -example of a syntax error e.g. IF THEN Y=32

Code Generation

During this stage, the compiler transform the tokenised form into code that can be understood by the computer's processor using lookup tables. The source code is converted to an object code (Executable code).

Code Optimisation

Code produced by the code generation process may not be the most efficient code. Code optimisation produce code which executes faster than that produced by the translator software. This may entail producing code which takes up less memory when executed.

Linker and loader

Linker software

It links segments of code which have been compiled independently. This is usually needed when the programmer has developed program libraries (which can then be used by many applications).

Loader software

It loads object or executable code into main memory. It should be noted that many are re-locatable loaders.

How a Variable is managed throughout compilation?

During lexical analysis identifiers and keywords are differentiated and checked against rules (e.g. length) for identifiers. Error messages are produced if identifier does not match the expected rules. Variable identifiers will be tokenized and variable identifiers are entered into symbol table. The Data type will be added to the entry in the symbol table and addresses in memory allocated to variables during syntax/semantic analysis stage and assignment of illegal types of data to variables is reported.

MetaLanguages

Meta-Languages are used to represent the syntax of programming languages. It governs the way that the elements of the language can be joined together to form valid statements. The possible way to represent meta-languages are:

- Backus-Naur Form (BNF)
- Extended Backus-Naur Form (EBNF)
- Syntax diagram (or Railroad diagram)

BNF notation (Backus-Naur Form)

It is one of the notation used to describe the syntax of languages. Since all programming languages needs to be translated to machine code, they must be clearly defined and BNF offers the way to describe them.

BNF Symbols

Components	Symbol	Meaning
Production Rule	::=	Is read as "is defined as"
Non-terminal symbol	<>	Any characters written within these symbols make up a non-terminal language element.
Alternatives	|	Is read as "or"
Terminal symbol		Any set of characters not enclosed by symbols is a Terminal. E.G B

Note: BNF does not have symbols for optional or repetition. It uses recursion to represent repetition.

<letter>::= a|b|c|d|e|f

<word>::=<letter>|<letter><word>

The above is read as:

- ✓ Letter is defined as one symbol a, b, c, d, e, or f.
- ✓ A word is defined as either letter or letter followed by words.

Let us consider an example using the above rule:

Example	Legal/Illegal	Reason
b	Legal	b is defined as a letter
g	Illegal	g is not defined as a letter
bed	Legal	b,e,d are all legally defined to make a word
Bed	Illegal	Upper case B not defined as a letter
5	Illegal	5 not defined as letter
bad fab	Illegal	Space character not defined as letter

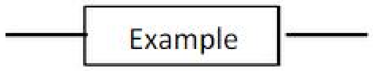
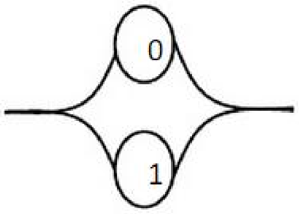
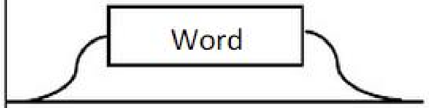
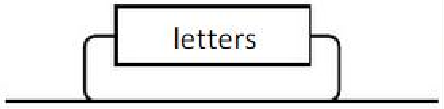
e.g. of BNF rules

<digit> ::=0|1|2|3|4|5|6|7|8|9
<letter>::=
a|b||c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|
W|X|Y|Z

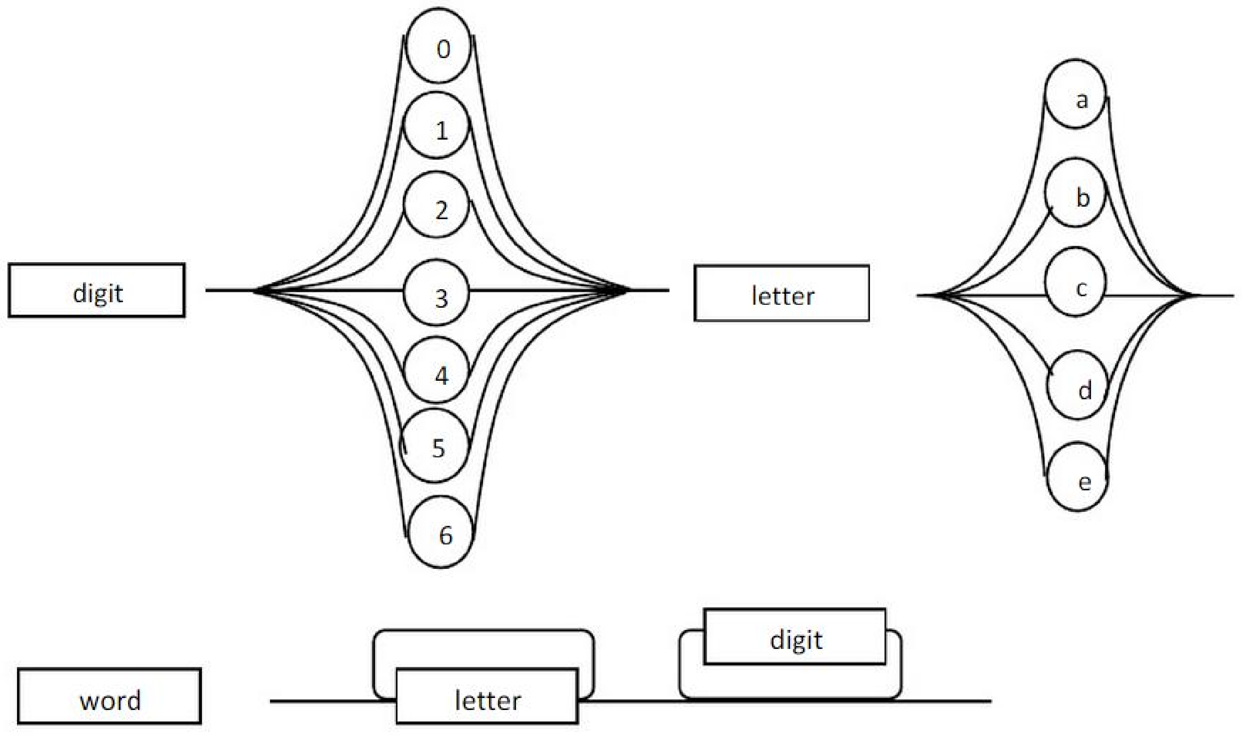
<integer> ::= <digit>|<digit><integer>

Syntax Diagrams

It can also be used to describe a language's syntax in a graphical form. The definitions starts from the left and follows a "railway" line which can even loop but reverse direction is not allowed. Elements which repeats or are optional are easily represented by branching.

Component	Example	Meaning
Non-terminal Symbol		A non-terminal language element that is defined elsewhere
Alternatives		Either 0 or 1
Optional		Either word or nothing
Repetition		0 or more letters

Syntax Diagram Examples



The diagrams above show that a digit is defined as 0,1,2,3,4,5 or 6 and a letter as a,b,c,d or e. A word may have one or more letters followed by zero or more digits.

Example	Element	Legal/Illegal	Reason
b	letter	Legal	b is defined as a letter
cab	word	legal	All characters are defined letters
Bed	word	Illegal	Upper case B not defined as a letter
5	digit	legal	5 is defined as a digit
8	digit	illegal	8 is not defined as digit
ad40	word	legal	One or more letters followed by one or more digits
d3c5	word	Illegal	Cannot reverse direction. One a digit is in the word, only digits may follow

RPN (Reverse Polish Notation)

Reverse polish Notation (also known as Postfix Notation) is an unambiguous way of writing mathematical expressions without using brackets. Traditionally, we are used to write expression using the Infix Notation i.e. A * B where the (*) operator is found between the two operands (A, B). The problem with this notation is that when we have several operators, brackets must be used so as we know which part goes together. E.g. 8 * (A+B)

To solve the above issue, we use postfix notation whereby brackets are avoided by putting the operator before the operands. E.g. + A B * 8

Using RPN, we have A B + 8 *

What are the advantages?

1. They can be solved via stacks
2. Brackets are not needed
3. Important in computing industry as expressions written in it are unambiguous

How can we convert from RPN and Infix Expressions?

We can use a binary tree or even a stack.

- ✓ Using InOrder Traversal of the binary tree we can get the infix notation
- ✓ Using PostOrder Traversal of the binary tree we can get the reverse polish notation
- ✓ By highlighting each major part of an expression, in the order it is calculated, a binary tree can be constructed.

Converting Infix to Postfix Notation (BODMASS)

Hierarchy of Operators

1. () brackets first
2. / or * that comes first from the left hand side is done first , precedence level is same
3. + or – that comes first from the left hand side is done first , precedence level is same

e.g. F = 3 * (6 + 2) – 4 / (3 + 7) which equals to 2

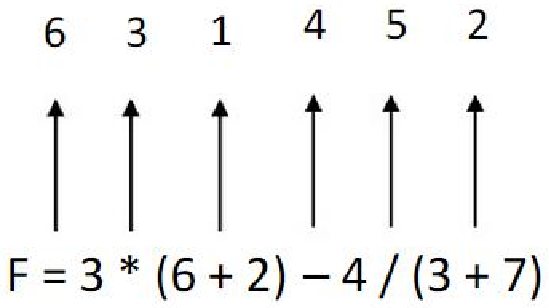
Steps:

Number the operators according to the precedence of the equation

1. +
2. +
3. *
4. -
5. /
6. =

The above is the hierarchy of the operators for this equation.

e.g.

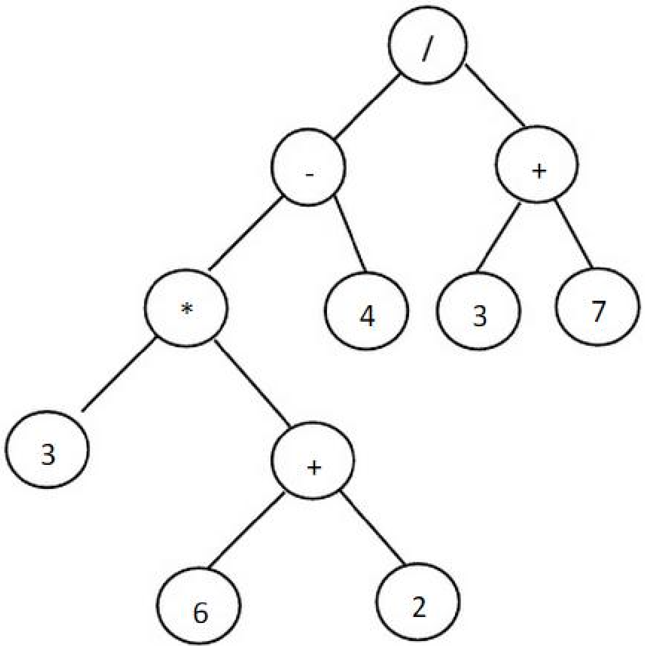


Using a stack to evaluate RPN

Stacks can be used to evaluate expressions.

Steps:

1. Each operand is pushed onto the stack in turn until it meets an operator.
2. Read the operator.
3. The top two values in the stack are popped.
4. The operator is carried out on them.
5. The result is placed on the stack.
6. The same above steps (1-5) are repeated until the final infix expression is the only item on the stack or the expression has been evaluated.



Infix Notation : 3 * (6 +2) -4 / (3+7)

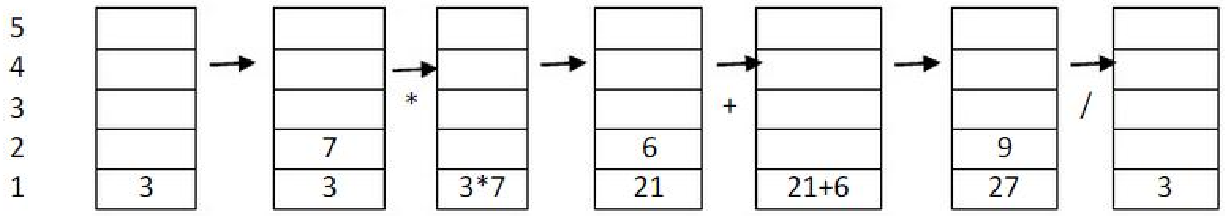
Postfix Notation : 3 6 2 + * 4 – 3 7 + /

Postfix to Infix Notation

Step

1. PUSH the operands in the stack from the postfix notation
2. When an operator comes then POP out the previous two operands and perform the operation.
3. PUSH the result back into the stack.

Consider how the stack will be used to evaluate the RPN expression given: 3 7 * 6 + 9 /



Security - Asymmetric keys and Encryption Methods

Cryptography

It is defined as the conversion of data into a scrambled code that is decrypted and send across a private or public network. It is normally is to protect data such as email messages, chat sessions, web transactions and others. Its main objective to ensure Confidentiality, Authentication, Integrity and Non repudiation.

Encryption Terminologies

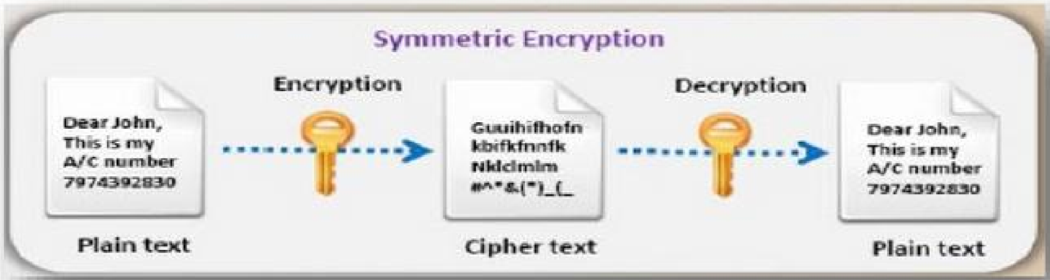
Plain text - It is defined as the message text or data before encryption. That is the unaltered text or original text

Cipher text - It is defined as the message text after encryption has occurred.

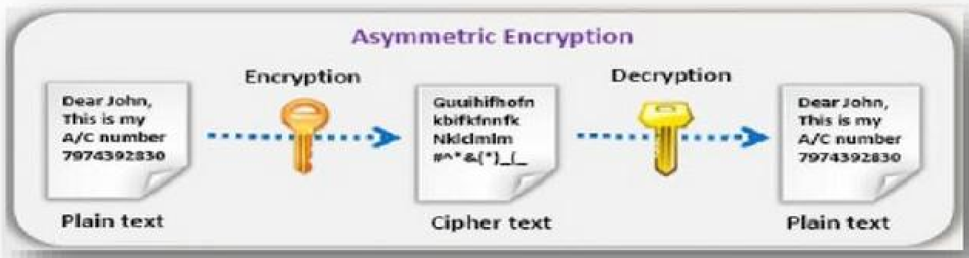
Types of Cryptography

If we increase the key length, the stronger is the encryption

1. Symmetric Encryption (Secret Key ,Shared Key , Private Key)
   - ✓ use the same key for encrytpion as they do for decryption
   - ✓ e.g AES – **Advance Encryption Standard** cipher uses (128,192 ,256 bit) and is used by US government
   - ✓ e.g DEA – **Data Encryption Algorithm** – was designed for hardware implementation whereby it is often used for single user encryption of their files on hard disk


Symmetric Encryption
Encryption
Decryption
Dear John, This is my A/C number 7974392830
Guuihifhofn kbifkfnnfk Nkldmlm #^*&(*)_(_
Dear John, This is my A/C number 7974392830
Plain text
Cipher text
Plain text

2. Asymmetric Encryption (public key)
   - ✓ uses different encryption keys for encryption and decryption. These keys are known as public and private keys.


Asymmetric Encryption
Encryption
Decryption
Dear John, This is my A/C number 7974392830
Guuihifhofn kbifkfnnfk Nkldmlm #^*&(*)_(_
Dear John, This is my A/C number 7974392830
Plain text
Cipher text
Plain text

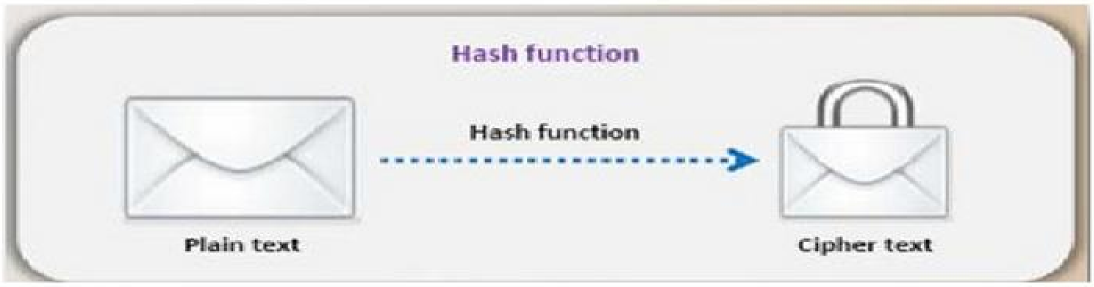
Private Key

Private Key is known only to the owner. It is used by the owner to encrypt the content.

Public Key

Public key is known by both parties and is used to decrypt the content received from a sender.
Note: Public and private keys are obtained from the purchase of a digital certificate

3. **Hash Function** (is a message digest or one- way encryption)
   - ✓ it uses no key for encryption and decryption



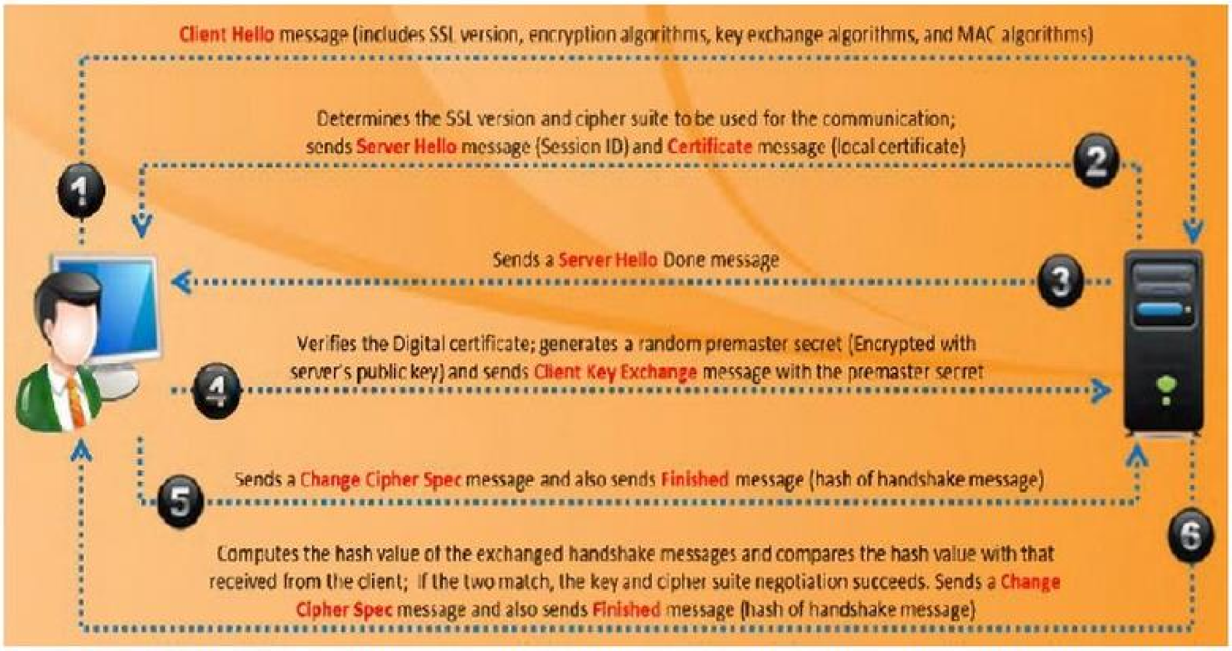
SSL

It is the short form for Secure Sockets Layer, a protocol developed by Netscape for transmitting private documents via the Internet. SSL uses a cryptographic system that uses two keys to encrypt data – a public key known to everyone and a private or secret key known only to the recipient of the message.

Its main aim is to provide privacy between two communicating applications such as a client and a server. Moreover, the protocol also authenticate the server and the client. It usually required a reliable transport protocol such as TCP for data transmission and reception. Both Netscape Navigator and Internet Explorer support SSL and many web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with https: instead of http. SSL is used often in online shopping sites, online banking sites

It offers 3 basic properties:

- A private channel where the messages are encrypted after the simple handshake that defines the secret key.
- The channel is authenticated. The server endpoints are always authenticated but the client endpoints are optionally authenticated.
- The channel is reliable. The transmission has an integrity check.


Client Hello message (includes SSL version, encryption algorithms, key exchange algorithms, and MAC algorithms)

Determines the SSL version and cipher suite to be used for the communication; sends Server Hello message (Session ID) and Certificate message (local certificate)

Sends a Server Hello Done message

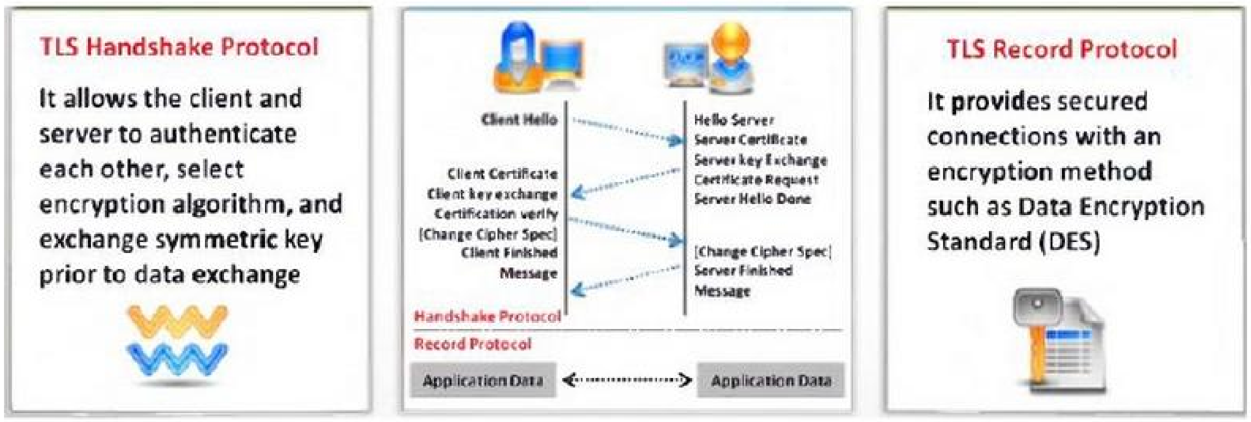
Verifies the Digital certificate; generates a random premaster secret (Encrypted with server's public key) and sends Client Key Exchange message with the premaster secret

Sends a Change Cipher Spec message and also sends Finished message (hash of handshake message)

Computes the hash value of the exchanged handshake messages and compares the hash value with that received from the client; If the two match, the key and cipher suite negotiation succeeds. Sends a Change Cipher Spec message and also sends Finished message (hash of handshake message)

TLS (Transport Layer Security)

TLS is a protocol to establish a secure connection between a client and a server and ensures privacy and integrity of information during transmission. TLS protocol is used to provide information security over the Internet. It usually uses asymmetric cryptography for key exchange, symmetric encryption for confidentiality, and message authentication codes for message integrity. It uses the RSA algorithm with 1024 and 2048 bit strength. By using TLS, risks can be reduced such as tampering, message forgery mail communications, and eavesdropping during transmission of Email or information.


TLS Handshake Protocol

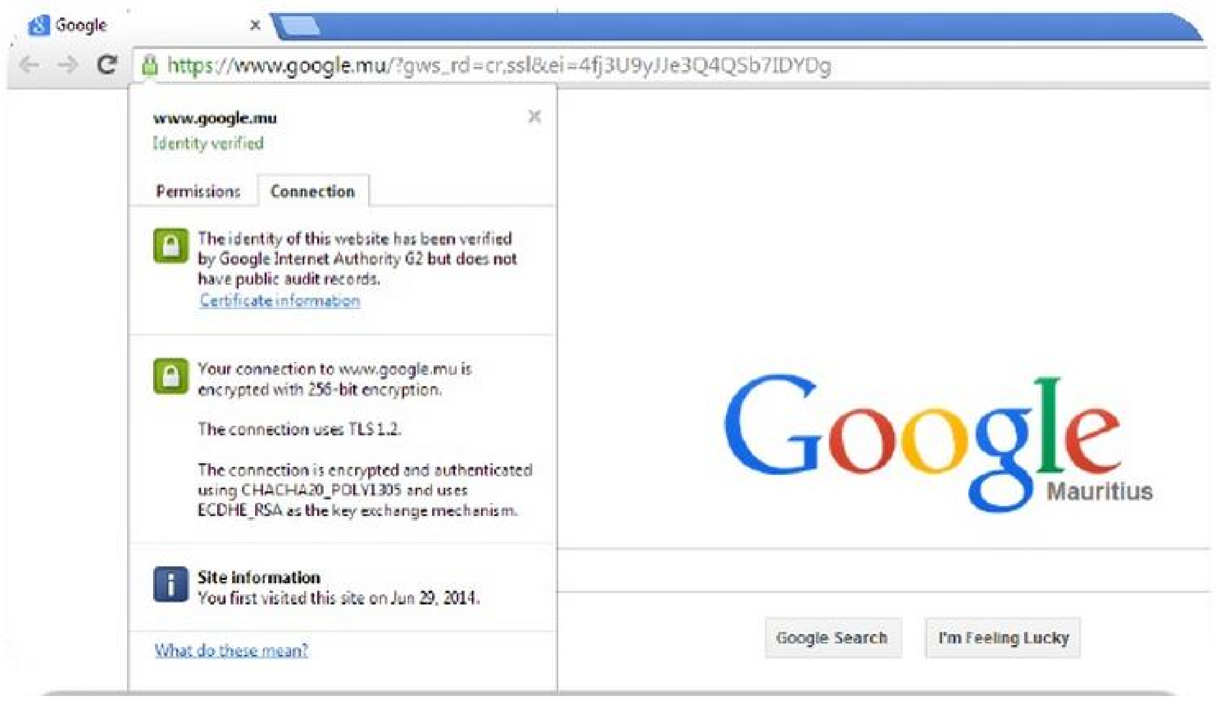
It allows the client and server to authenticate each other, select encryption algorithm, and exchange symmetric key prior to data exchange

TLS Record Protocol

It provides secured connections with an encryption method such as Data Encryption Standard (DES)

Digital Certificate

A digital certificate is an attachment to an electronic message used for security purposes. The most common use of a digital certificate is to verify that a user sending a message is who he or she claims to be, and to provide the receiver with the means to encode a reply. It is used often in online shopping sites, online banking sites.


Google

https://www.google.mu/?gws_rd=cr,ssl&ei=4fj3U9yJJe3Q4QSb7IDYDg

www.google.mu
Identity verified

Permissions | Connection

The identity of this website has been verified by Google Internet Authority G2 but does not have public audit records.
Certificate information

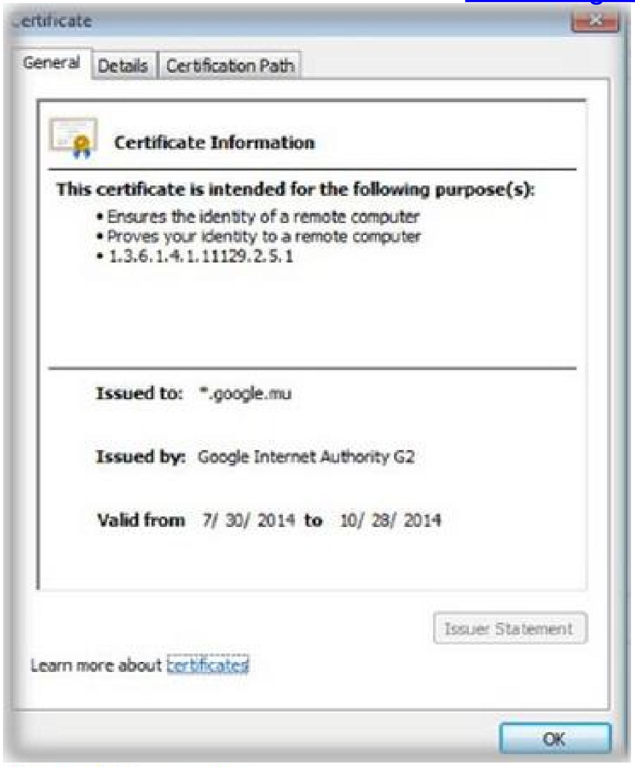
Your connection to www.google.mu is encrypted with 256-bit encryption.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using CHACHA20_POLY1305 and uses ECDHE_RSA as the key exchange mechanism.

Site information
You first visited this site on Jun 29, 2014.

What do these mean?



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- 1.3.6.1.4.1.11129.2.5.1

Issued to: *.google.mu

Issued by: Google Internet Authority G2

Valid from 7/ 30/ 2014 to 10/ 28/ 2014

Learn more about certificates

Digital Signature

It is defined as a mathematical scheme used for authentication of a digital message computed using a set of rules and a set of parameters such that the identity of the signatory and integrity of the data can be verified. It is used in asymmetric encryption to simulate the security properties of a signature in digital, rather than written form. A hash function is used to create and verify a digital signature.

DSA (Digital Signature Algorithm) are used by most government.

A digital signature algorithm includes a signature generation process and a signature verification process as well.

i. Signature Generation Process – the private key is used to know who has signed it.
ii. Signature Verification Process – the public key is used to verify whether the given digital signature is genuine or not.

Today many online shopping sites, e-payments sit and other electronic payment modes rely on various systems like DSA. E.g. RSA (Rivest Shamir Adleman), MD5.

Definition and Security Problems

Cookies

These are small files sent to user's computer when visiting a website. It stores information about user which is accessed every time user visits that website and lets website know who you are a past visitor.

Pharming

It is defined as malicious code installed on a user's computer or web server which misdirecting the user(s) to a bogus/fake fraudulent website without their knowledge. Technically, they can use domain name poisoning.

Virus

Viruses are programs that replicate themselves and are designed to disrupt computer system by deleting, altering or corrupting files. They can be grouped in several categories such as Logic Bomb, Time Bomb, and Polymorphic and so on.

Hacking

Unauthorised access to files or programs either locally or remotely in view to gather information or for disrupting the system in an effort to use data illegally (e.g. fraud)

Phishing

It is an attack which involves sending emails to recipients claiming to be a legitimate company. When the email is opened, recipient is directed to bogus website/gets details about customer. (Usually done to credit card using websites such as ecommerce or banks)

Key logging/spyware

It is a Program installed on a computer to monitor all key presses. Each key press is relayed back to the program writer.

Spyware

It scans files on hard drive and 'snoop' applications.

Shoulder Surfing

It is defined as the act of watching a person key in secure data (e.g. PIN, password, etc.).It involves stealing security data by using binoculars, CCTV near ATMs etc. to watch key presses etc.

War Driving

It means locating a wireless network by touring round an area in a vehicle and accessing wireless networks for malicious purpose. It usually requires a laptop, special software (such as backtrack, kali etc.) and an antenna which allow for packet injection.

Problem	Safeguard
Hacking	▪ Use an IDS (Intrusion Detection System) to prevent illegal access to files ▪ Update Software Regularly, activate service packs and patches ▪ Check Logs regularly and do regular penetration testing

	▪ Use of passwords for authentication purpose ▪ Use of Firewalls to restrict access ▪ Also locking the computer itself or locking the computer room can help here. ▪ Encryption stops the information being read even if access has been gained to a file but won't stop hacking!!
Viruses	▪ Use of updated anti-virus software to prevent viruses entering a computer. ▪ It is also sensible not to open emails or attachments from "unknown" sources; ▪ It also helps to only load software from disks or CDs which are originals rather than using pirated software sources
Corruption/Loss of Data can occur in a number of ways: - viruses - hackers - accidental damage - hardware faults - software faults - incorrect computer operation	▪ viruses can be protected against as described above ▪ accidental damage is best guarded against by keeping back up files or use the Grandfather-Father-Son (GFS) method; also use of passwords and ids can help by restricting access in the first place ▪ protection against hardware faults could be to keep backups or use GFS; use of UPS (in case of power loss) ▪ parallel system also help ▪ - keeping back up files or using GFS would help here backing up files would guard against problems caused by incorrect power down of the system
Spyware / Malware Pharming	Use of an anti-spyware or anti-malware Check for website address properly and certificates

Monitoring and Control System

Monitoring Systems

Monitoring systems make measurements for display or analysis .These include:

- ✓ Monitoring system for the body functions of hospital patients (Vital signs)
- ✓ Burglar Alarm System(Monitoring for Intruders)
- ✓ Weather monitoring system
- ✓ Environmental monitoring.
- ✓ Chemical and Nuclear Plants (Monitoring of Key Parameters such as temperature, airflow etc.)

Control Systems

It uses measurement to provide feedback in order to control a process. These include the control of chemical plant, nuclear power stations and road traffic among others. One or more computers are used to regulate the operation of other devices.

Monitoring and Control applications

These include:

- ✓ Monitoring and control systems for industrial plants
- ✓ Monitoring and control systems for traffic. (controlling the sequence of lights to maintain optimum traffic flow)

Steps in Monitoring and Control

1. Data are gathered via sensors in analogue form
2. An ADC (Analogue to Digital Convertor) is used to transform the data before transmission to a computer /data logger / microprocessor.

   If **monitoring** system is being used **then**
3. Microprocessor compares received data with pre stored parameters or settings for the environment to be monitored.
4. The system will warn the user in terms of a screen output or a siren to notify the user.

   End if

   IF a Control System is being used **then**
5. A signal (converted via a DAC before transmission) is sent to the actuator (valve, heaters etc.) which control the motor.
6. Environment settings and parameters are adjusted accordingly

   End if

7. Monitoring and control start again.

Note: Stage 1- 3 -> Monitoring Phase, Stage 4-6 Control Phase

Why is Monitoring and Control Done?

Monitoring and control using computers/microprocessors is often done for the following reasons:

- ✓ It is safer (faster response to non-standard conditions and they don't get tired and miss key data)
- ✓ Computers work 24/7 (even though humans can work in shifts there is always the danger of missing information at shift handover etc.)
- ✓ Data can be automatically displayed and analysed without the need to enter data manually (which in itself could introduce errors into the system)
- ✓ Computers are more accurate and can take more frequent readings (e.g. if readings need to be taken every 30 seconds, humans can make mistakes or miss readings or even find it impossible to take readings at such short time intervals)

Bit Manipulation

Instruction		**Explanation**
Opcode	**Operand**	
AND	#n	Bitwise AND operation of the contents of ACC with the operand
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>
OR	#n	Bitwise OR operation of the contents of ACC with the operand
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address> <address> can be an absolute address or symbolic address
Label>: <opcode > <label>:	<operand> <data>	Labels an instruction Gives a symbolic address <label > to the memory location

Logical Operation

- AND : outputs 1 if only if both inputs are 1
- OR : outputs 1 if at least one input is 1
- XOR : outputs 1 if exactly one input is 1

Truth Table

A	B	A AND B	A OR B	A XOR B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

ANDING

ANDing a bit with 0 produces a 0 at the output while ANDing a bit with 1 produces the original bit. This can be used to create a mask.

Consider the following:

Data : 1101 1001 1010

Mask : 1111 1111 1111

ANDing : 1101 1001 1010

ORING

ORing a bit with 1 produces a 1 at the output while ORing a bit with 0 produces the original bit. This can be used to force certain bits of a string to 1s.

Consider the following:

Data : 1101 1001 1010 **Mask** : 1111 1111 1111 **ORing** : 1111 1111 1111	**Data** : 1101 1001 1010 **Mask** : 0000 1111 0000 **ORing** : 1101 1111 1010

XORing

XORing a bit with 1 flips the bit (0->1, 1->0) at the output while XORing a bit with 0 produces the original bit. It can be used to tell whether two bits are unequal. It can also be used as an optional bit-flipper.

Example

Consider a controlled environment in a chemical where a pump for air has to be turned on. The actuator will control the motor with 1 for on and 0 for off. Suppose that the pump motor bit is located in memory location 0904. Bit 3 of this memory location controls the pump.

7	6	5	4	3	2	1	0	Bit number
0	0	1	1	**1**	1	0	1	value

Using the assembly instruction below, the actuator will set the location 0904 to 1.

Instruction		Explanation
Op Code	Operand	
LDM	#n	Immediate addressing. Load the number n to ACC
LDD	<address>	Direct addressing. Load the contents of the given address to ACC
STO	<address>	Store the contents of ACC at the given address
OUT		Output to the screen the character whose ASCII value is stored in ACC
AND	#n	Bitwise AND operation of the contents of ACC with the operand
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand
OR	#n	Bitwise OR operation of the contents of ACC with the operand

To set the bit correctly the following assembly language code is used.

```
LDD 0904

OR # 8   // OR #B 0000 1000

STO 0904
```