Lecture 01
History and the Evolution of the Computer Systems
Components of a Computer System

IT1020 Introduction to Computer Systems

Part 01 – Computer Fundamentals

Pre-lecture activities

- History of computer systems
  - Major Milestones
- Diagram of Von Neumann
- Components of computer systems
- Storage devices

Lecture Outline

1. Generations of Computers
   - Characteristics of each Generation
2. Components of Computers
   - Different types of components

1. Generations of the Computers

1st Generation computers(1944-1955)

2nd Generation computers (1955-1964)

3rd Generation computers (1964-1971)

4th Generation Computers (1971- Present)

5th Generation Computers (Present and Beyond)

1. Generations of the Computers
1st Generation Computers (1944-55)

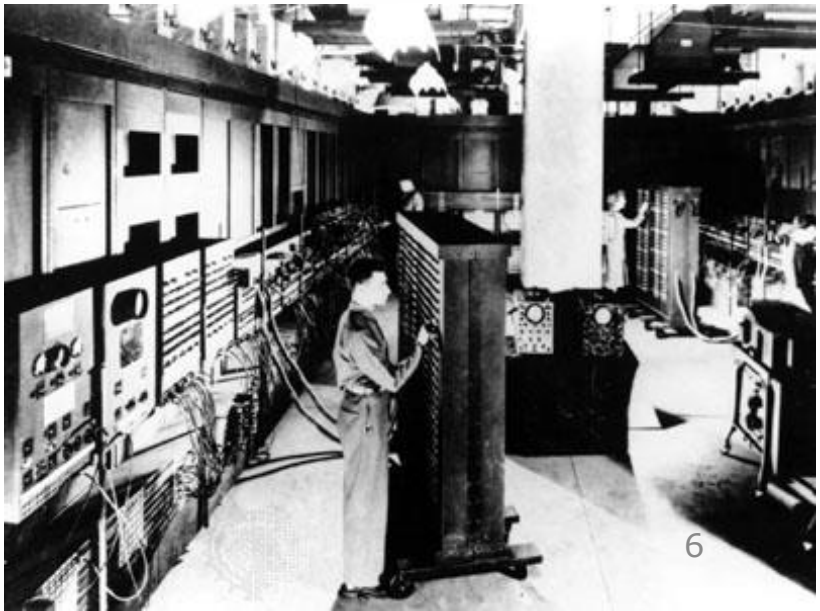
Main characteristics of this generation

- Used Thermion valves
- Large in size and very heavy in weight
- Power consumption was very high
- First Generation Computers relied on Machin Language
- Writing program on them was difficult or quite slow
- They were very expensive to operate, using a great deal of electricity, generated a lot of heat, which was often the cause of malfunctions

1. Generations of the Computers
1st Generation Computers

Electronic Numerical Integrator and Calculator (ENIAC)

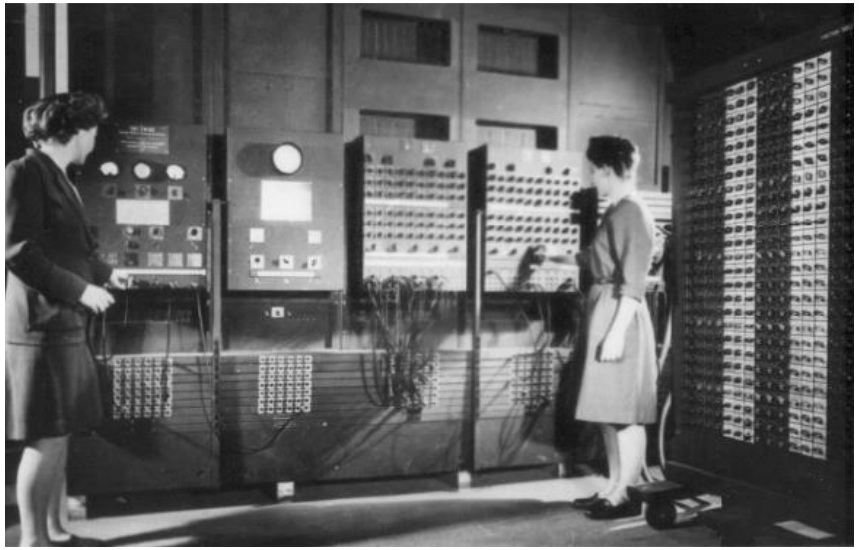
- 1946 : First electronic general purpose calculator, ENIAC was built in U.S, weighs 33 tons, consumes 150kw, and averages 5000 operations per second



1. Generations of the Computers
1st Generation Computers

Hard wired programming

- Early computers were programmed, using large number of switches in the console panel and plugging/unplugging cables
- It is called hardwired programming



Two women working with ENIAC computer

United States Army Photo.

1. Generations of the Computers
1st to 2nd generation

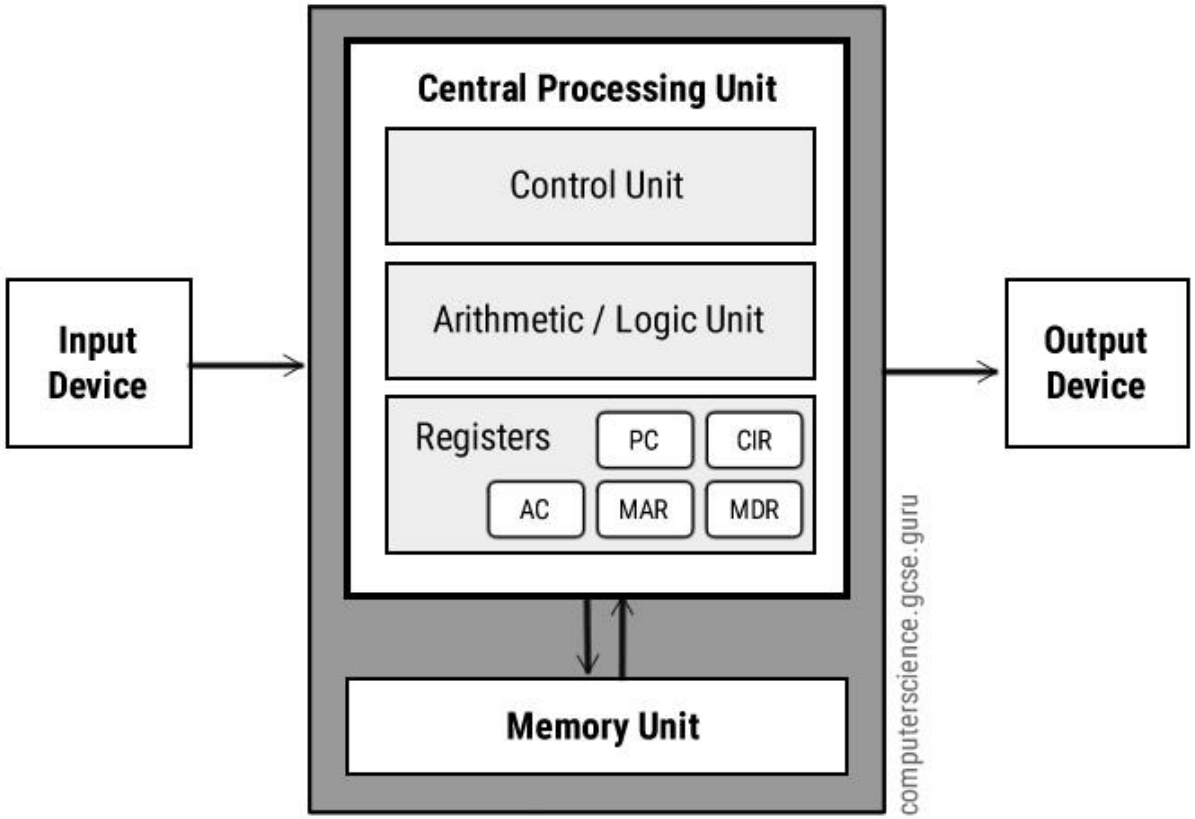
Von Neumann Architecture

- It was required re-wire and re-design the machine to run a different program. It was a manual and very tedious task
- Von Neumann proposed that programs and data can be stored in a **memory** device and instead of rewiring the machine we can change the program easily.

John Von Neumann

1. Generations of the Computers
1st to 2nd generation

Von Neumann Architecture



- All computers share the same basic architecture, whether it be a multi-million dollar mainframe or a Palm Pilot.
- All have memory, an I/O system, and arithmetic/logic unit, and a control unit.

1. Generations of the Computers
1st to 2nd generation

Von Neumann Architecture

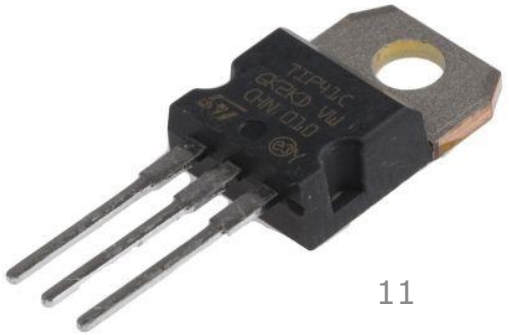
- The use of the binary number system
- A single sequentially addressed memory
- A separate arithmetic/logic unit for performing arithmetic and logical computations
- The stored program concept in which both the programs and its data are stored in memory.
- A controller that fetches instructions from memory and executes them.

1. Generations of the Computers
1st to 2nd generation

Invention of Transistor

- 1947 : Transistor, essential storage device for computers invented at Bell Labs by American engineers William Shockley, John Bardeen and Walter Bartain .

- Transistors were much smaller, more rugged, cheaper to make and far more reliable than valves.



1. Generations of the Computers
2nd Generation Computers (1955-64)

- Used transistors instead of Thermion valves.
- Comparatively higher operating speed.
- Size and weight of the computers decreased
- Manufacturing cost reduced
- The concepts of Central Processing Unit (CPU), memory, programming language and input and output unites were developed.
- High-level programming languages introduced
- Development of software for computers
- Computer industry experienced rapid growth.

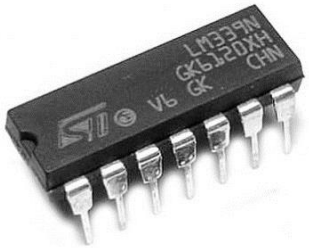
1. Generations of the Computers
2nd Generation Computers (1955-64)

IMB 1620

IMB 1401

1. Generations of the Computers
3rd Generation Computers(1964-71)

- Integrated Circuits (ICs) were used (A single IC has many transistors, resisters and capacitors built on a single thin slice of silicon.)
- The size of the computer got further reduced
- High Level Languages were developed in this generation



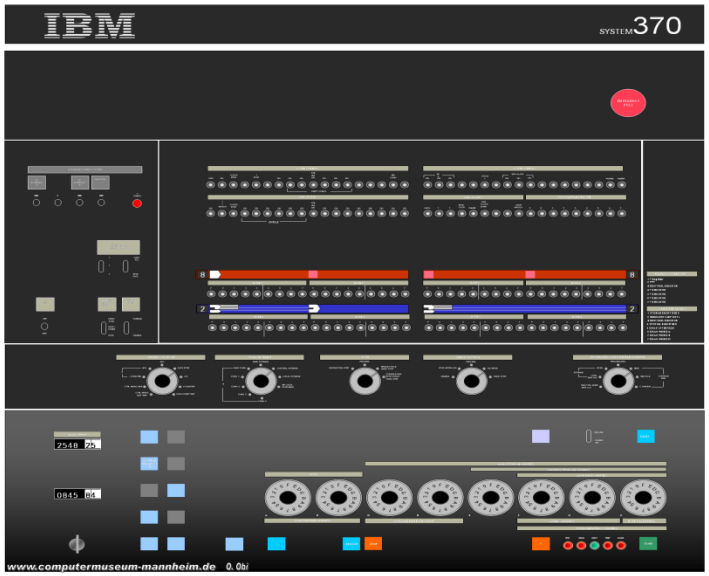
- Large IC companies were started. (INTEL started in 1968, AMD started in 1969)
- The computers were low cost, large memory and processing speed was very high.

1. Generations of the Computers
3rd Generation Computers(1964-71)

- Substantial **operating systems** were developed to manage and share the computing resources and time sharing operating systems were developed. These greatly improved the efficiency of computers.
- Computers had by now pervaded most areas of business and administration.
- Allowed the device to run many different applications at one time.

1. Generations of the Computers
3rd Generation Computers(1964-71)

IBM System/360



IBM System/370

1. Generations of the Computers
4th Generation Computers (1971-)

- Personal computers were developed and IBM launched
- the Power PC and Pentium introduced the 8088 and 8086 microprocessors. (Most of the computers at present are belong to this generation)
- It uses large scale Integrated Circuits (LSIC) built on a single silicon **chip** called **microprocessors**.
- Memory chips are in megabit range

1. Generations of the Computers
4th Generation Computers (1971-)

- On the software side, more powerful operating systems are available such as Unix.
- Fourth generation languages (4GLs) make the development process much easier and faster.
- Applications software has become cheaper and easier to use.
- Software development techniques have vastly improved.

1. Generations of the Computers
4th Generation Computers (1971-)

Microprocessor

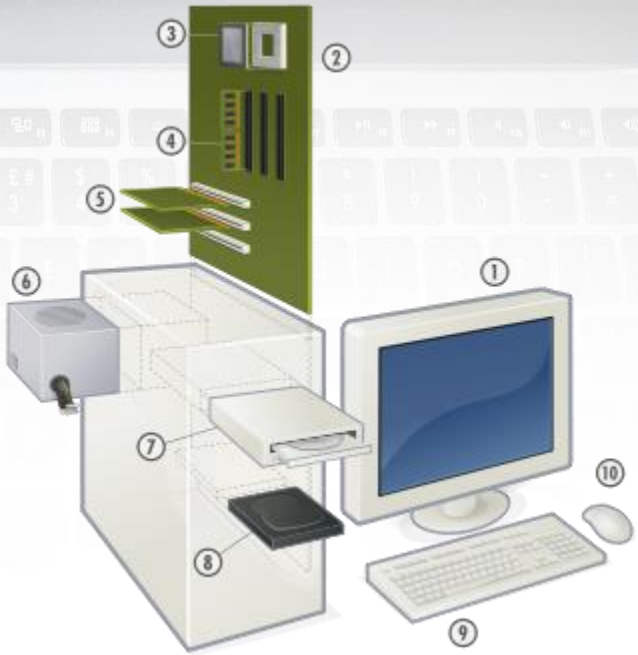
1. Generations of the Computers
5th Generation Computers

Present and beyond

- Fifth generation computing devices, based on Artificial Intelligence (AI).
- Are still in development, though there are some applications, such as voice recognition.
- The use of parallel processing and superconductors is helping to make artificial intelligence a reality.
- The goal of fifth-generation computing is to develop devices that respond to natural language input and are capable of learning and self-organization.

2. Components of the Computer

1. Input devices
2. Output devices
3. Processing devices
4. Storage devices
5. Other devices
   - Motherboard, Expansion cards, Power supply



2. Components of the Computer
Basic layout

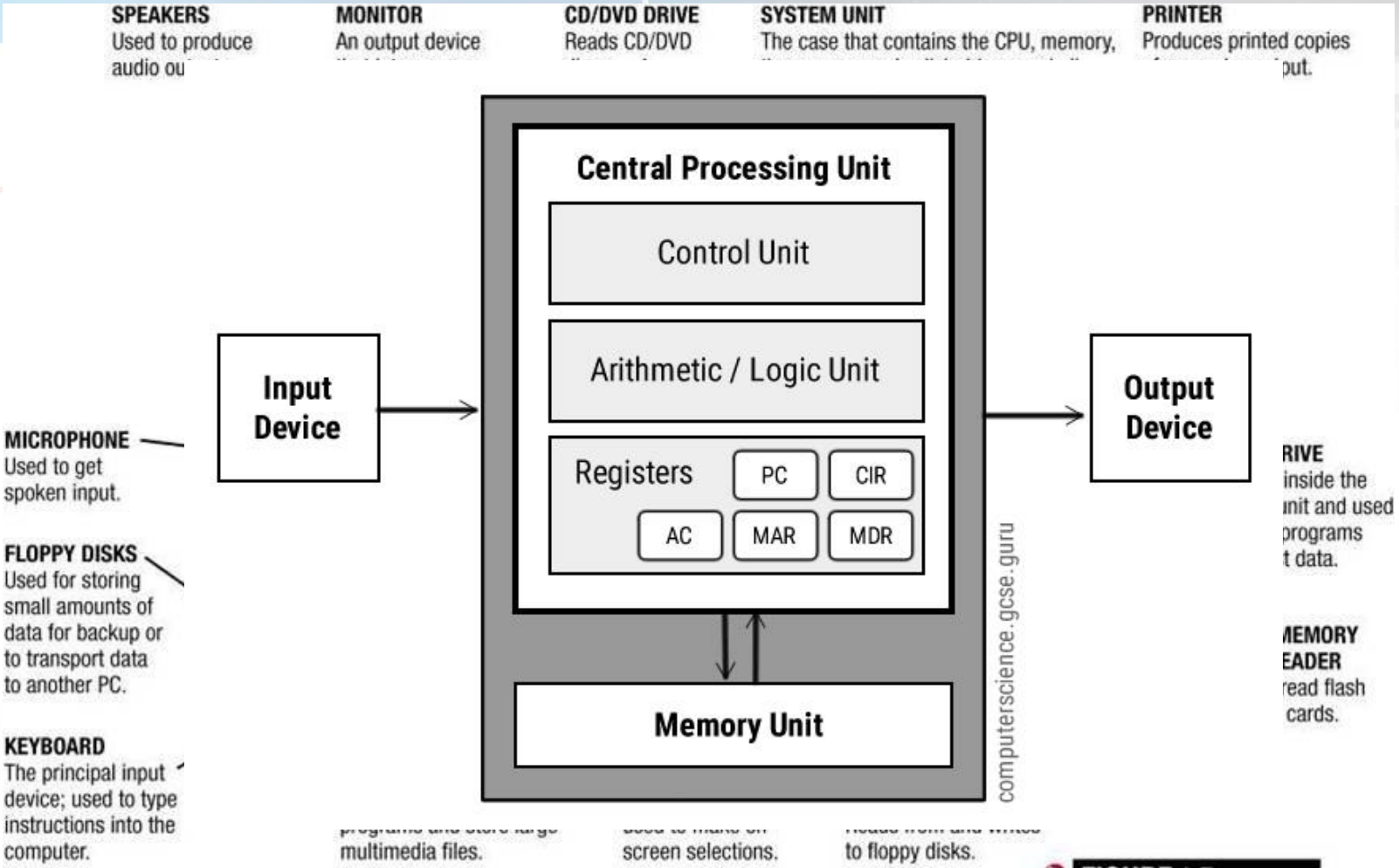
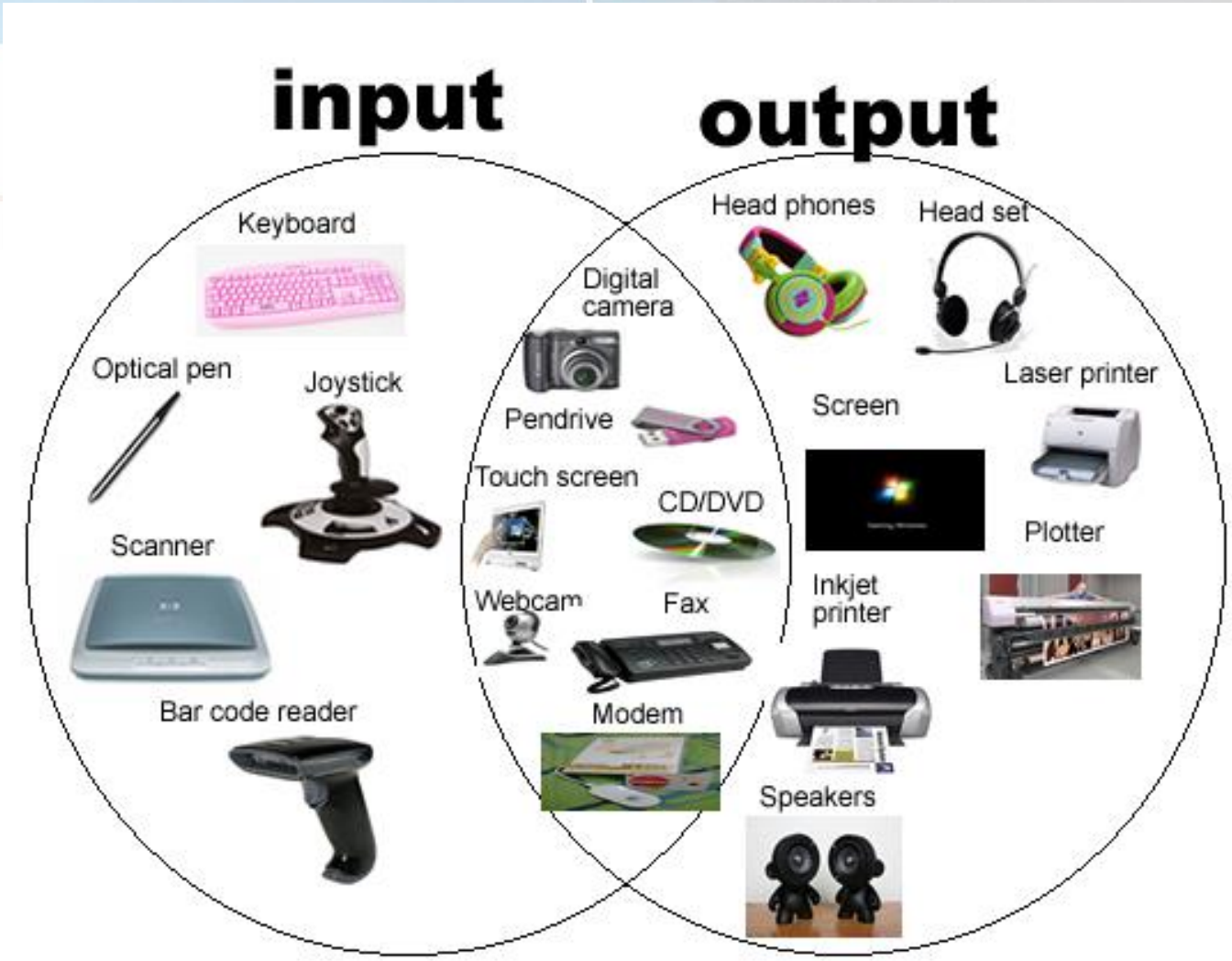


FIGURE 1-7
Typical computer hardware.

2. Components of the Computer
Input and output devices



2. Components of the Computer
Processing devices

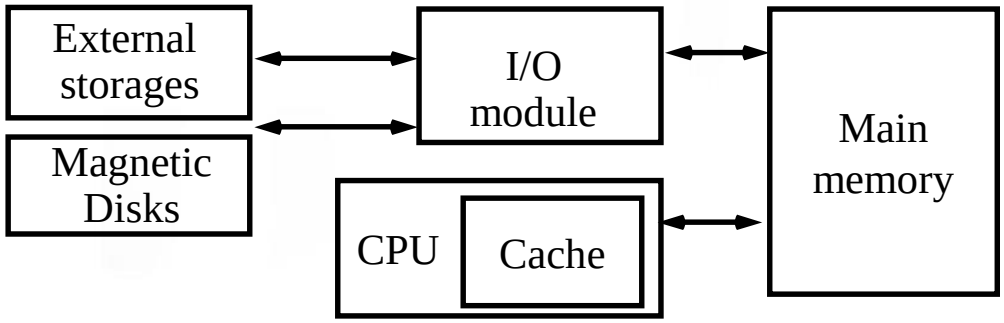
2. Components of the Computer Storage Devices – Features

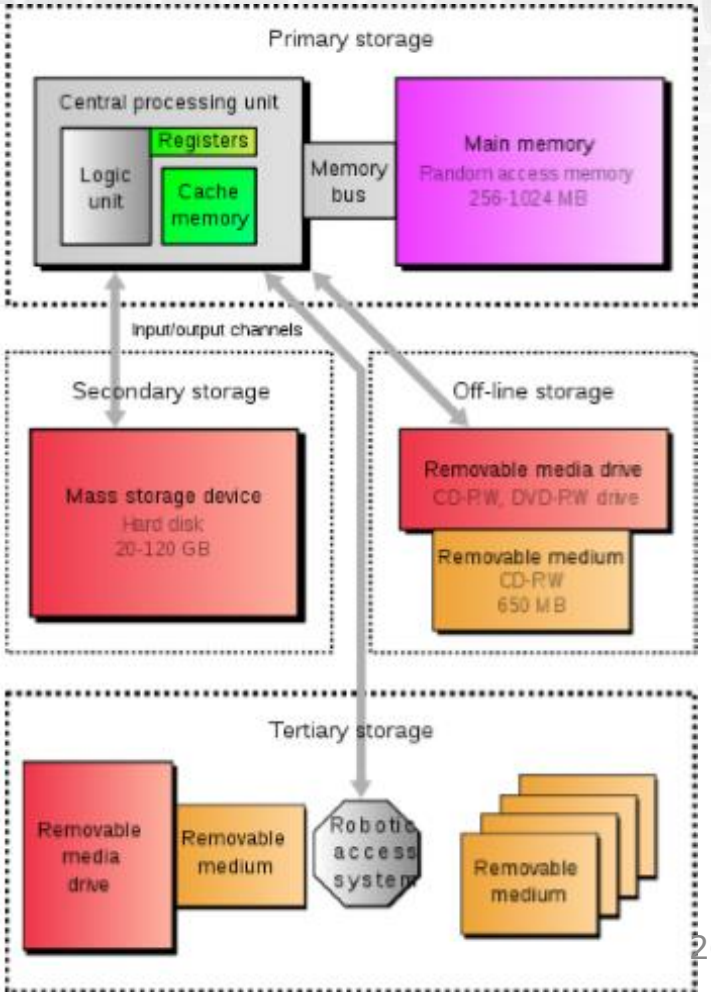
1. Volatility
   - Volatile storage
   - Non-Volatile storage
2. Accessibility
   - Random access
   - Sequential access
3. Mutability
   - Read/write storage or mutable storage
   - Read only storage
   - Slow write, fast read storage
4. Addressability
   - Location addressable
   - File addressable
   - Content addressable

2. Components of the Computer Storage Devices – Types

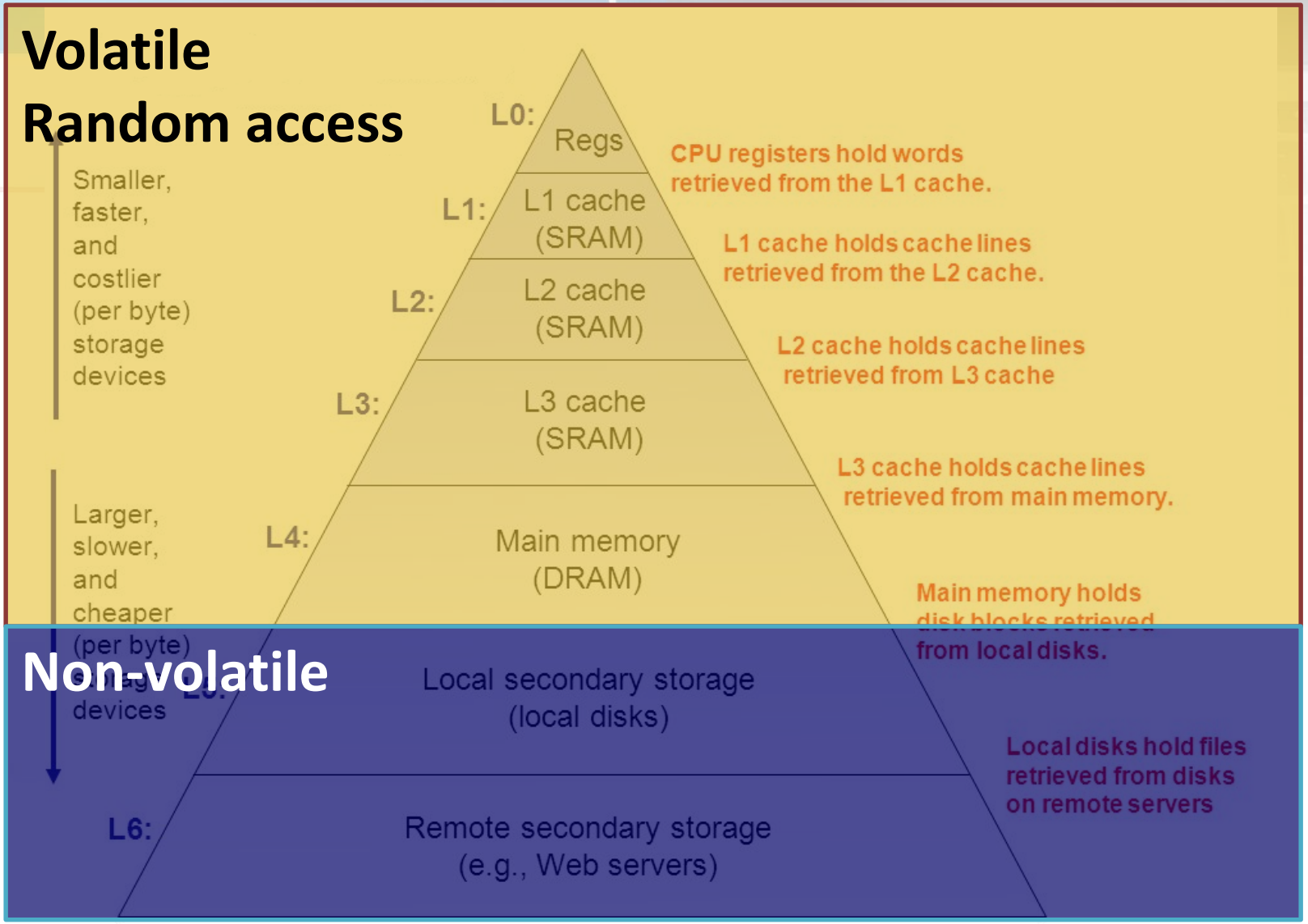
There are four type of storage:

1. Primary Storage
2. Secondary Storage
3. Tertiary Storage
4. Off-line Storage





2. Components of the Computer Storage Devices – Hierarchy



2. Components of the Computer Storage Devices – Primary storage

Primary storage types

1. Registers
2. Main Memory
3. Cache

2. Components of the Computer Storage Devices – Primary storage

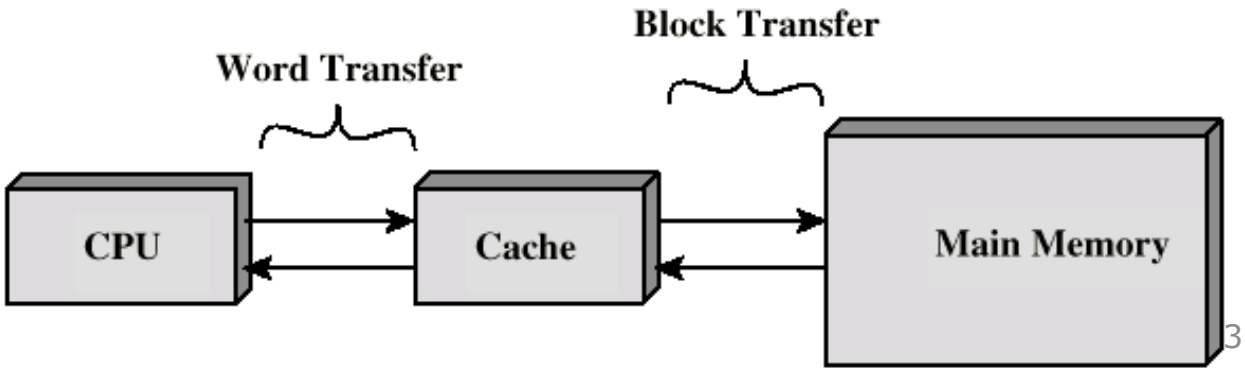
Registers

- very small amount of very fast memory that is built into the CPU
- This is to speed up its operations by providing quick access to commonly used values.
- Fastest memory in computer.
- Registers are normally measured by the number of bits they can hold, for example, an 8-bit register or a 32-bit register.
- Registers can also be classified into
  1. general purpose
  2. special purpose.

2. Components of the Computer Storage Devices – Primary storage

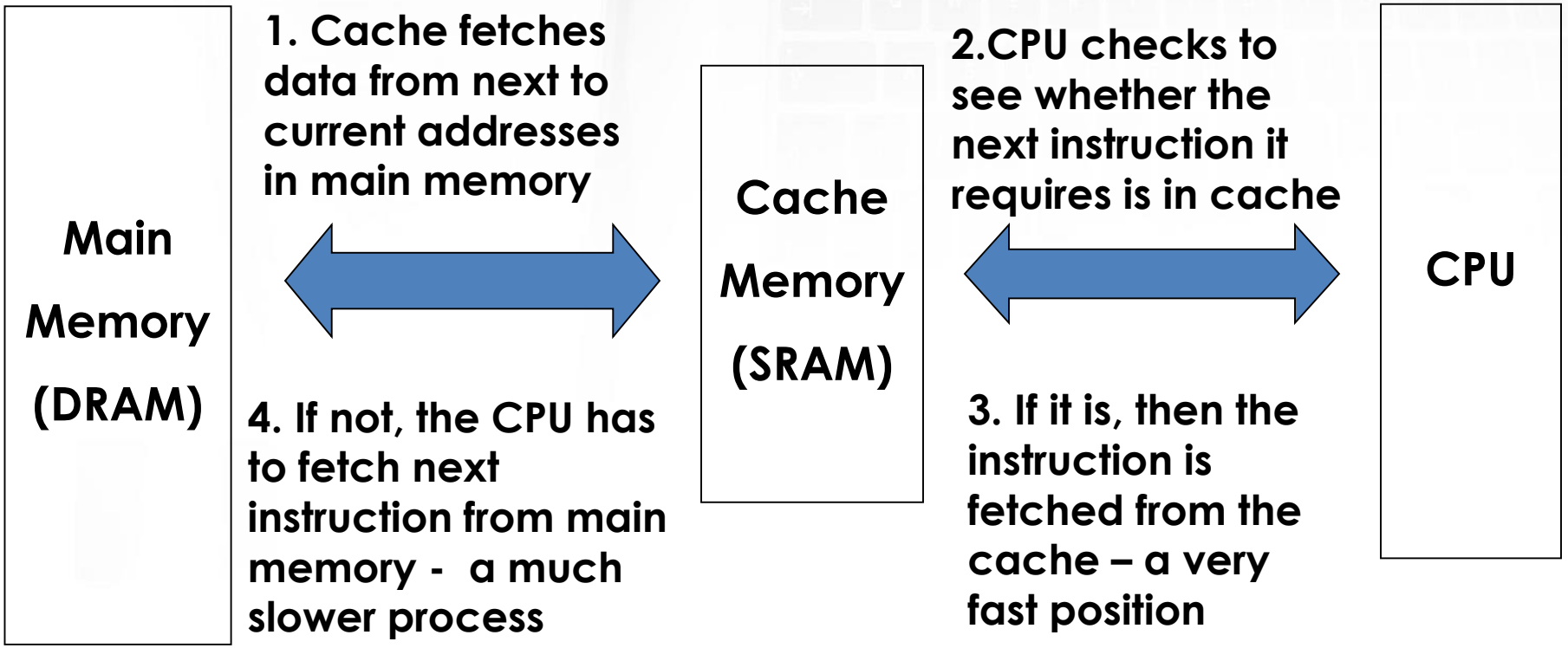
Cache

- Small amount of fast memory (Faster than RAM, static memory)
- Sits between normal main memory and CPU
- May be located on CPU chip or module.
- Cache works on the principal of locality of reference.



2. Components of the Computer Storage Devices – Primary storage

Cache – a The operation of cache memory



2. Components of the Computer Storage Devices – Primary storage

Memory – technologies

Static memory	Dynamic memory
▪ No charges to leak ▪ No refreshing needed when powered ▪ More complex construction ▪ Larger implementation per bit ▪ More expensive ▪ Faster that dynamic memory	▪ Bits stored as charge in capacitors ▪ Level of charge determines value - charges leak ▪ Need refreshing even when powered (need refresh circuits) ▪ Simpler construction, smaller per bit ▪ Less expensive ▪ Slower than Static memory

2. Components of the Computer Storage Devices – Primary storage

Memory – types

Read Only Memory (ROM)	Random Access Memory (RAM) Also called read/write memory.

▪Non-volatile in nature
▪These cannot be accidentally changed
▪Use static memory
▪Faster than dynamic memory
▪E.g. **BIOS chip**
  ▪Masked ROM
  ▪Programmable ROM (PROM)
  ▪Erasable PROM (EPROM)
  ▪EEPROM
	▪Volatile
	▪This is a semi conductor memory (dynamic memory)
	▪E.g. **Main memory**
	  ▪Main memory is usually called RAM. (misnamed because all semiconductor memory is random access)
	  ▪Main Memory can be made faster by using static memory. Then why don't we do that?
	  ▪Main memory is directly or indirectly connected to the central processing unit via a bus.
	  ▪The CPU continuously reads instructions stored in the main memory and executes them as required.

2. Components of the Computer Storage Devices – Primary storage

Memory – Main memory

- Main memory consists of a number of storage locations, each of which is identified by a unique address
- The ability of the CPU to identify each location is known as its addressability
- Each location stores a word i.e. the number of bits that can be processed by the CPU in a single operation. Word length may be typically 16, 24, 32 or as many as 64 bits.

2. Components of the Computer Storage Devices – Primary storage

Memory – Main memory

- Program and data are stored in memory prior to execution.(This is called Stored Program Concept proposed by Von Neumann).
- Memory is a semiconductor device in modern computers (Magnetic core memories were used earlier)
- Main memory, primary storage are synonyms to memory. (RAM also denotes the same)

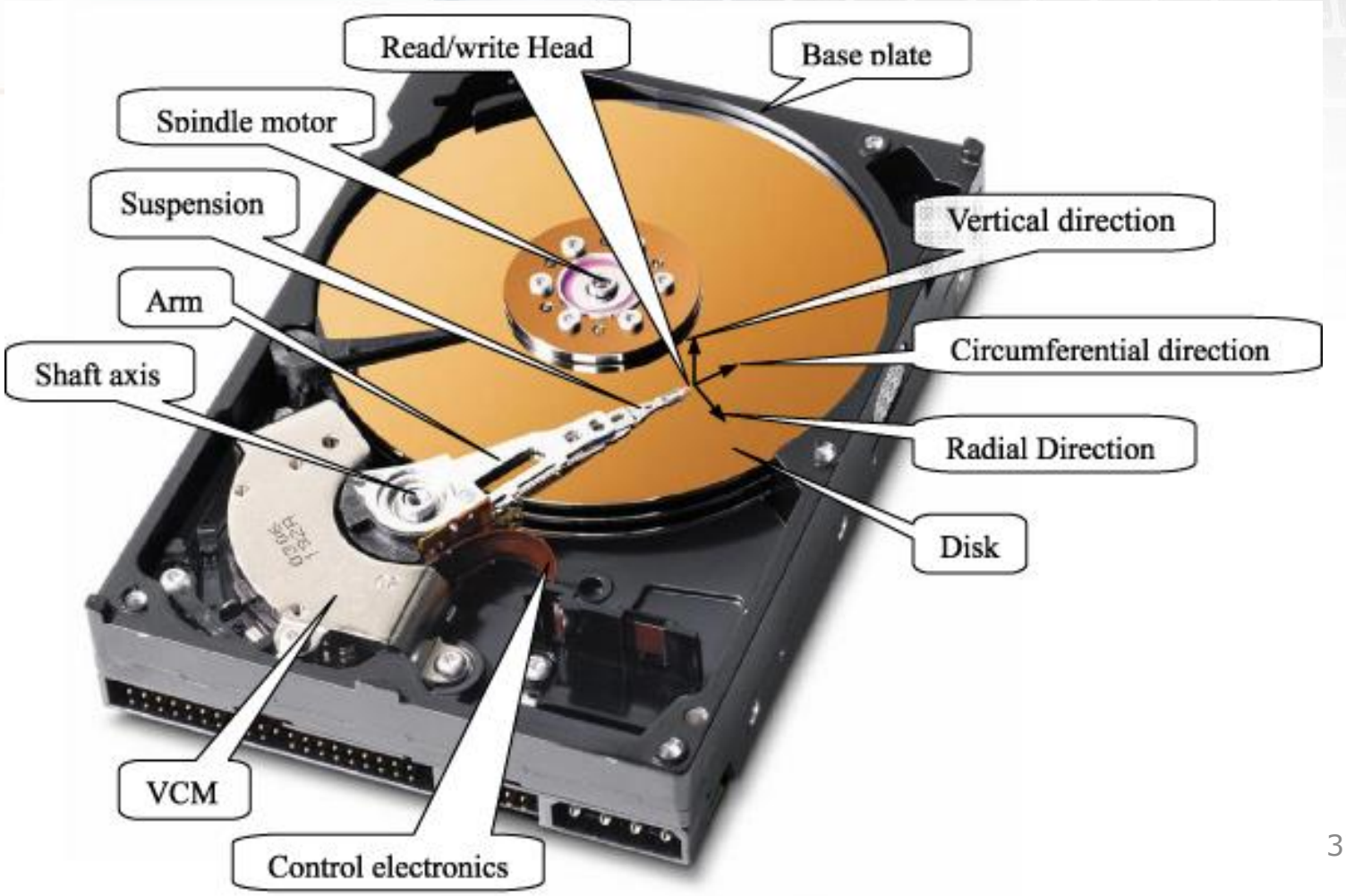
2. Components of the Computer Storage Devices – Primary storage

Memory – Main memory

- Memory is byte addressable
- Each byte has a unique address
- Addresses start from zero and increment sequentially.
- Memory Refresh – Memory refresh is the process of periodically read data from an area of computer memory and immediately writing the read information to the same area with no modifications.

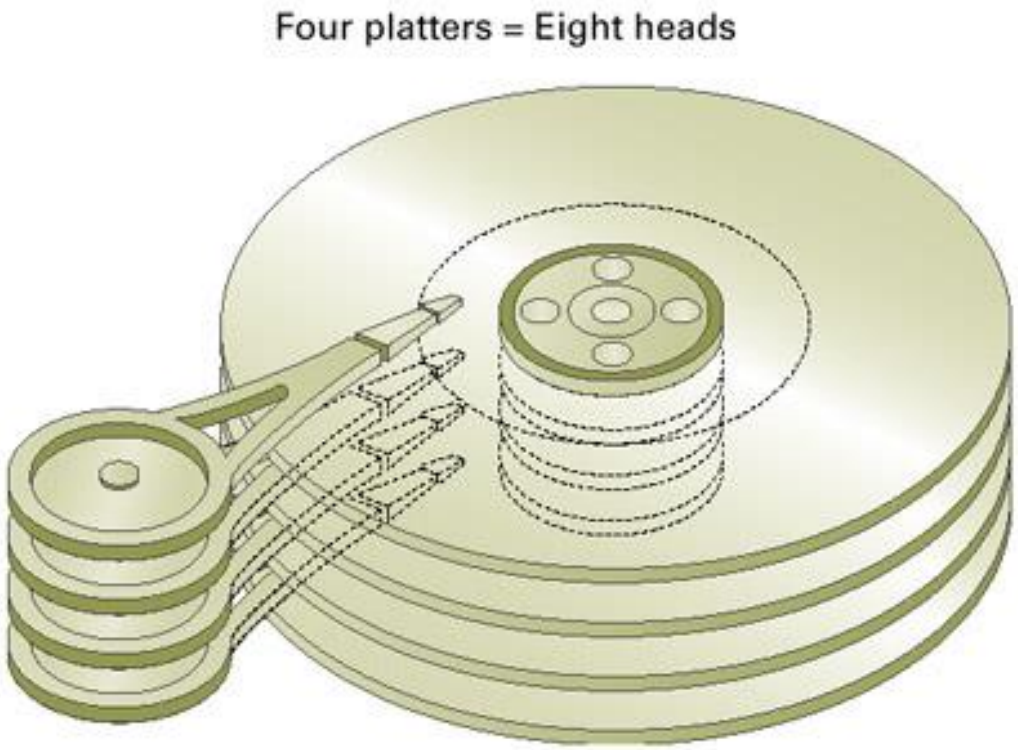
2. Components of the Computer Storage Devices – Secondary storage

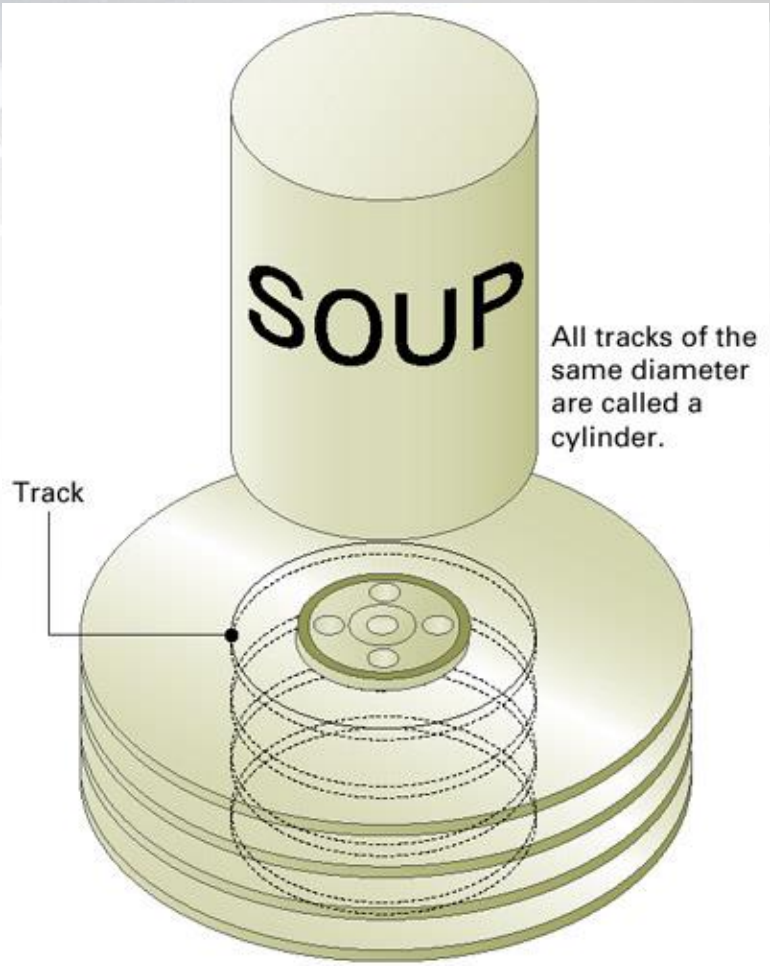
Hard disk



2. Components of the Computer Storage Devices – Secondary storage

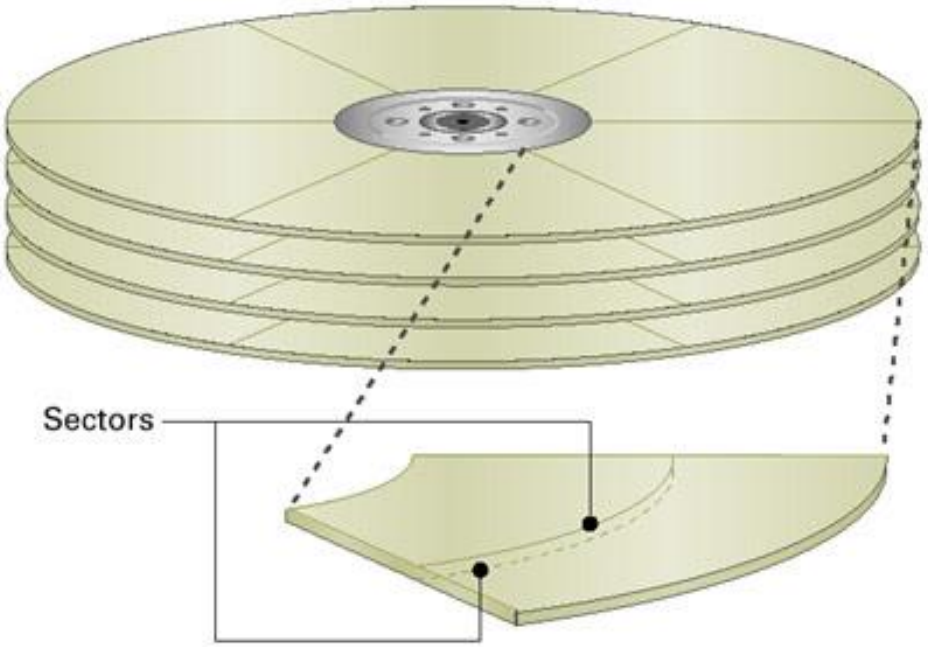
Hard disk

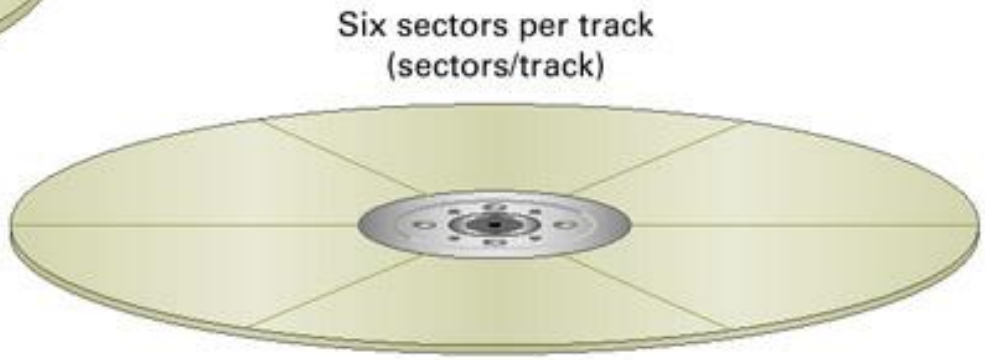




2. Components of the Computer Storage Devices – Secondary storage

Hard disk





2. Components of the Computer Storage Devices – Secondary storage

Hard disk – performance parameters

- **Access time** – seek time + rotational delay+ transfer time
- **Seek time** – track selection time (moving the head on the desired sector on the track)
- **Rotational delay** – the time it takes for the head to reach the beginning of the sector
- **Transfer time** – the time required to transfer data

2. Components of the Computer Storage Devices – Tertiary Storage

- Typically it involves a robotic mechanism which will mount (insert) and dismount removable mass storage media into a storage device.
- It is a comprehensive computer storage system that is usually very slow, so it is usually used to archive data that is not accessed frequently.
- This is primarily useful for extraordinarily large data stores, accessed without human operators.

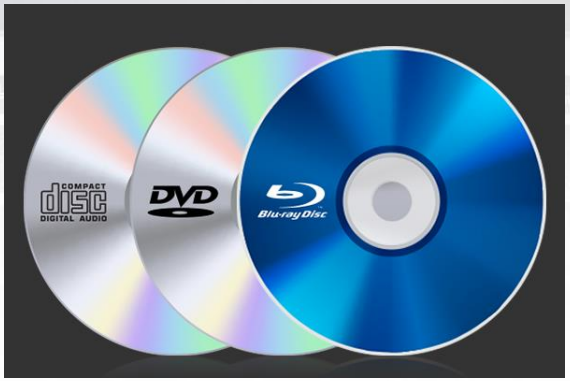
2. Components of the Computer Storage Devices – Tertiary Storage

2. Components of the Computer Storage Devices – Offline Storage

- Also known as Disconnected storage
- Is a computer data storage on a medium or a device that is not under the control of a processing unit
- It must be inserted or connected by a human operator before a computer can access it again
- Examples
  - Floppy Disk
  - CD/DVD/Blue-ray
  - USB Flash Drive
  - Memory Cards

2. Components of the Computer Storage Devices – Offline Storage

- A soft magnetic disk.
- Floppy disks are portable.
- Floppy disks are slower to access than hard disks and have less storage capacity, but they are much less expensive.
- Can store data up to 1.44MB.
- Two common sizes: 5 ¼" and 3 ½".



Memory Card

- An electronic flash memory storage disk commonly used in consumer electronic devices such as digital cameras, MP3 players, mobile phones, and other small portable devices.
- Memory cards are usually read by connecting the device containing the card to your computer, or by using a USB card reader.


Secure Digital card (SD)

MiniSD

Compact Flash


Memory Stick

- A small, portable flash memory card that plugs into a computer's USB port and functions as a portable hard drive.
- Flash drives are available in sizes such as 256MB, 512MB, 1GB, 5GB, and 16GB and are an easy way to transfer and store information.

2. Components of the Computer Storage Devices – Offline Storage

	CD	DVD
Stands for	Compact Disc	Digital Versatile Disc
Purpose	CDs are made with the purpose of holding audio files as well as program files.	DVDs are made with the purpose of holding video files, movies, substantial amount of programs, etc.
Media type	Optical disc	Optical disc
Capacity	Typically up to 700 MiB (up to 80 minutes audio)	DVD can range from 4.7 GB to 17.08 GB.
Types	CD-R, CD-RW, CD-Text ETC.	DVD-RW, DVD+RW, DVD-RAM and Blu-Ray.

2. Components of the Computer Storage Devices

Other Storage techniques

Cloud Storage

- Cloud storage means "the storage of data online in the cloud," wherein a data is stored in and accessible from multiple distributed and connected resources that comprise a cloud.
- Cloud storage can provide the benefits of greater accessibility and reliability; rapid deployment; strong protection for data backup, archival and disaster recovery purposes.

2. Components of the Computer Storage Devices

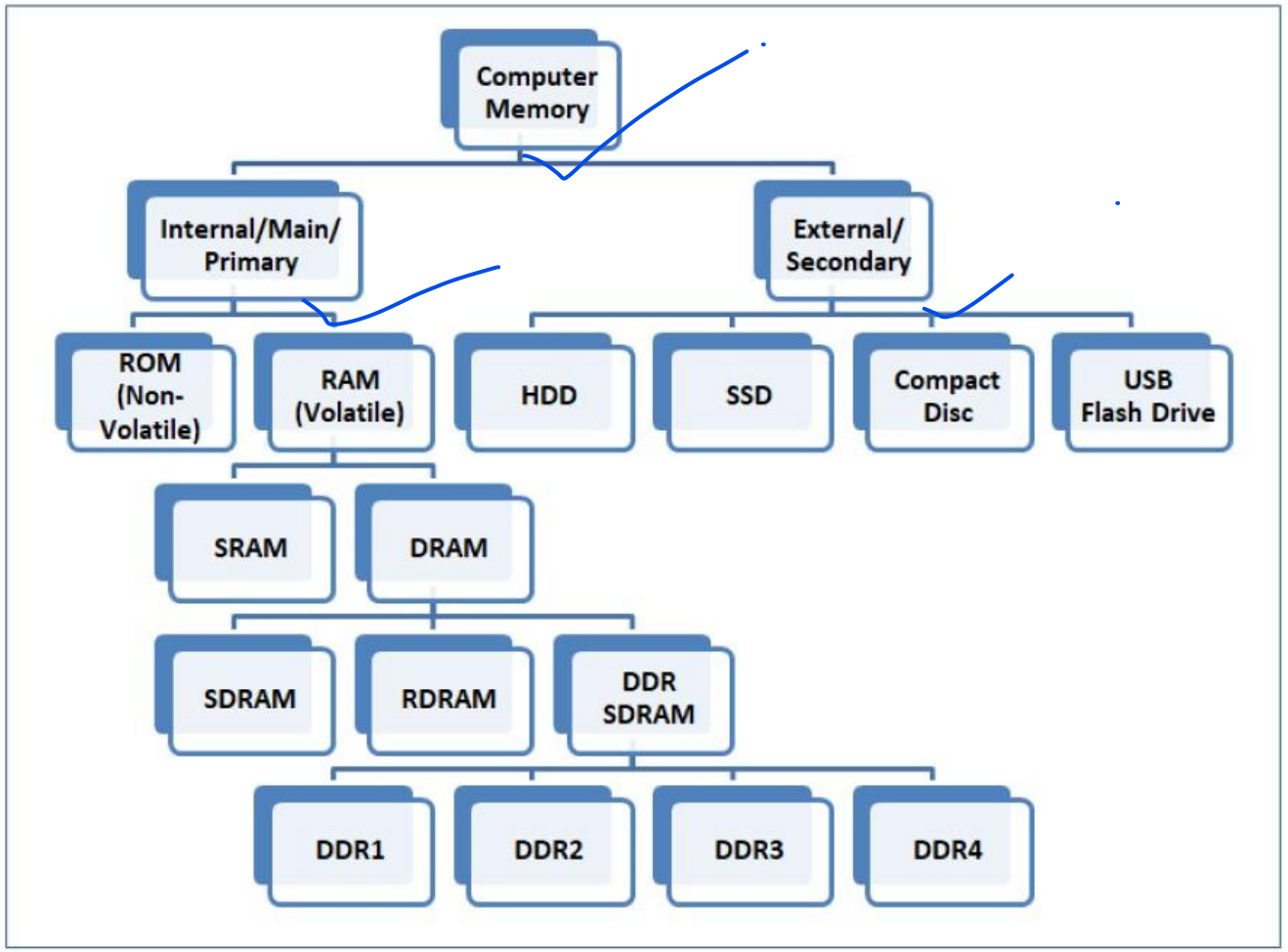
Other Storage techniques

Cloud Storage

- Examples:
  - Google Drive
  - Flickr
  - Microsoft Sky Drive

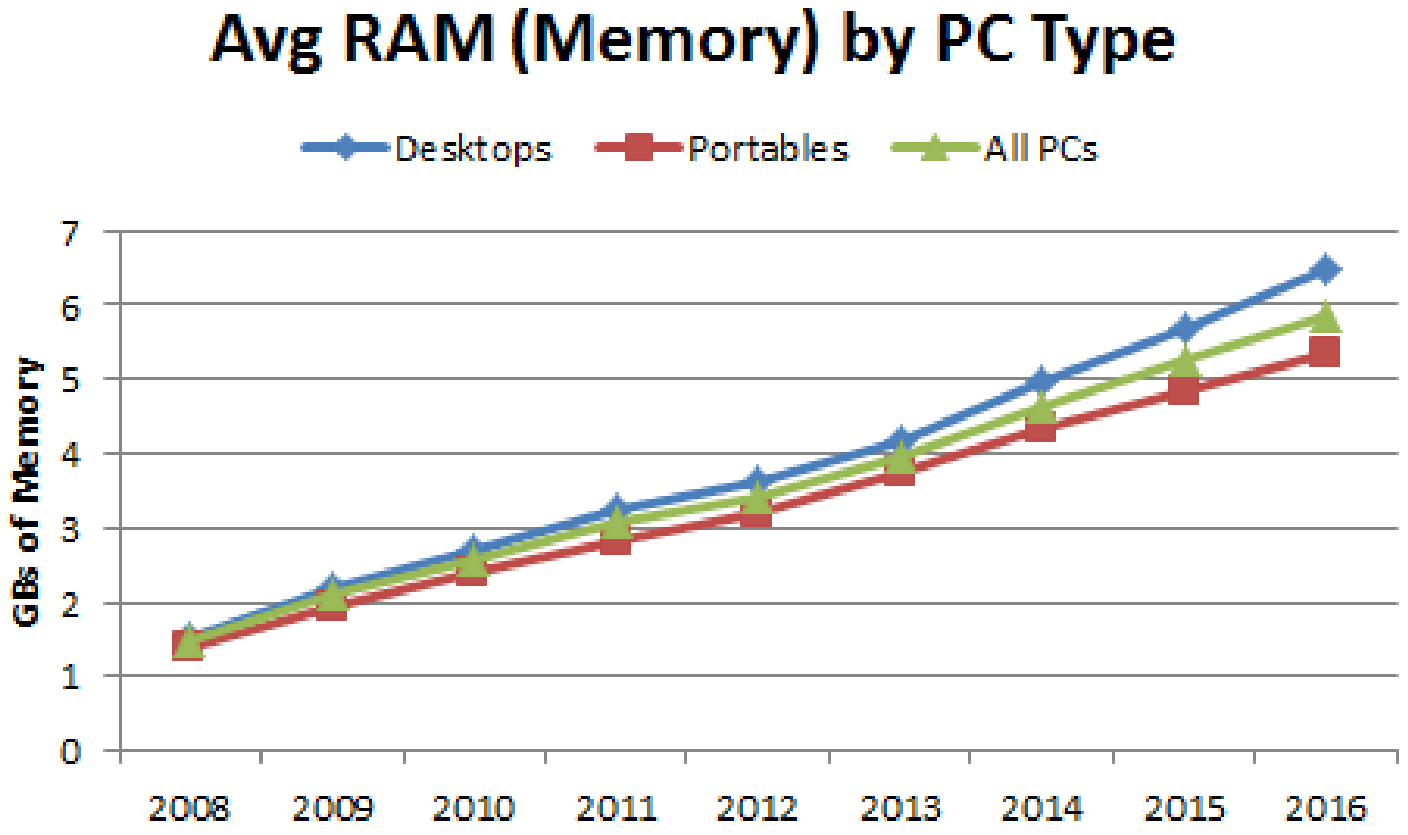
2. Components of the Computer Storage Devices

Evolution of storages



2. Components of the Computer Storage Devices

Evolution of storages



Thank You

Lecture01 - Post-lecture activities

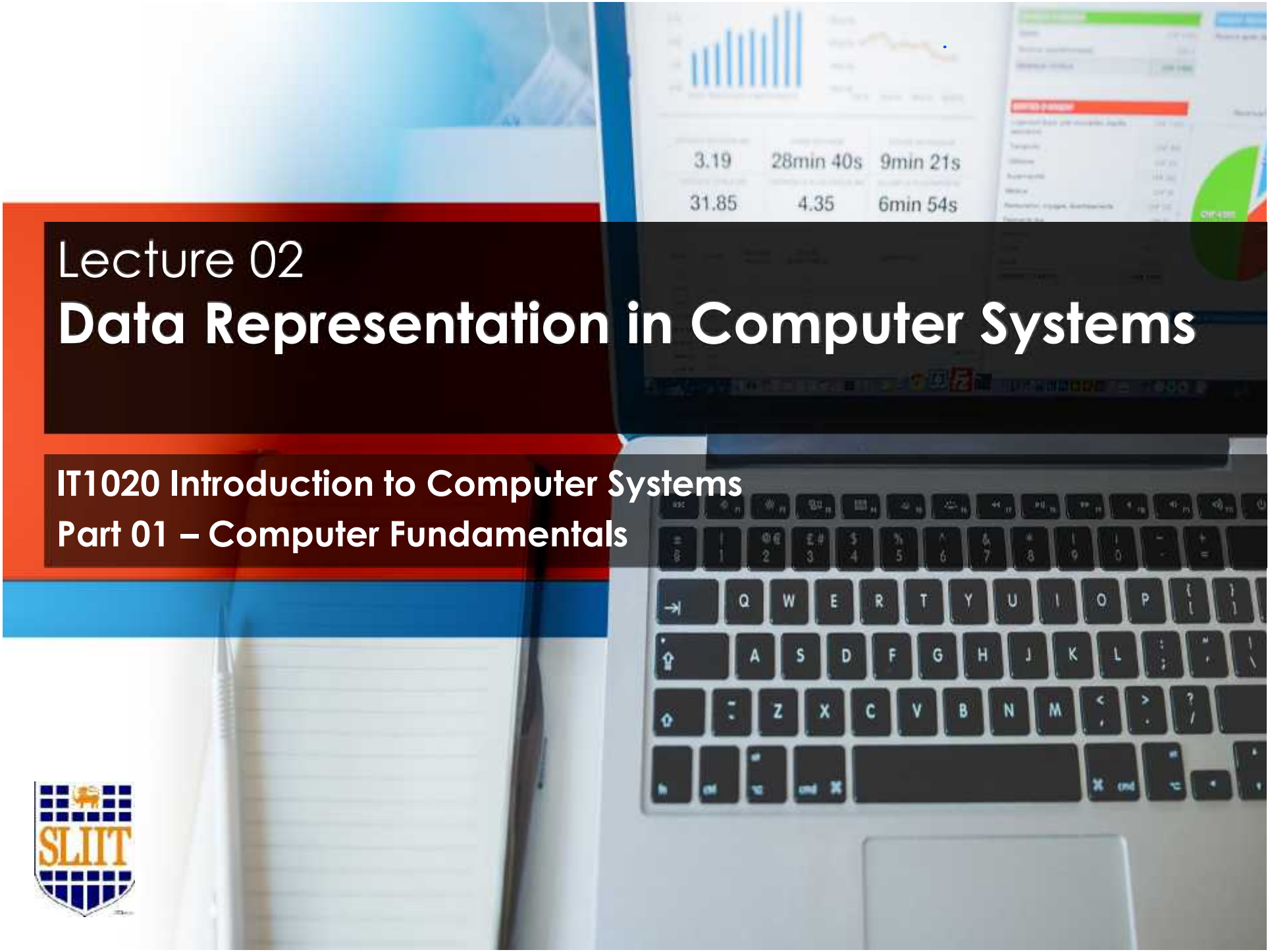
Find facts and create graph of
CPU Ops, Speed, Cores, Disk space, RAM speed

Lecture 02

Data representations

Lecture02 - Pre-lecture activities

Data & Information, Different Types of Data, ASCII/Unicode



Lecture 02

Data Representation in Computer Systems

IT1020 Introduction to Computer Systems

Part 01 – Computer Fundamentals

Pre-lecture activities

- Data & Information
- Different Types of Data
- ASCII/UNICODE

Content

1. Data and Information
2. Data types
3. Data Representation in Computers
4. Error checking
5. Data Compression

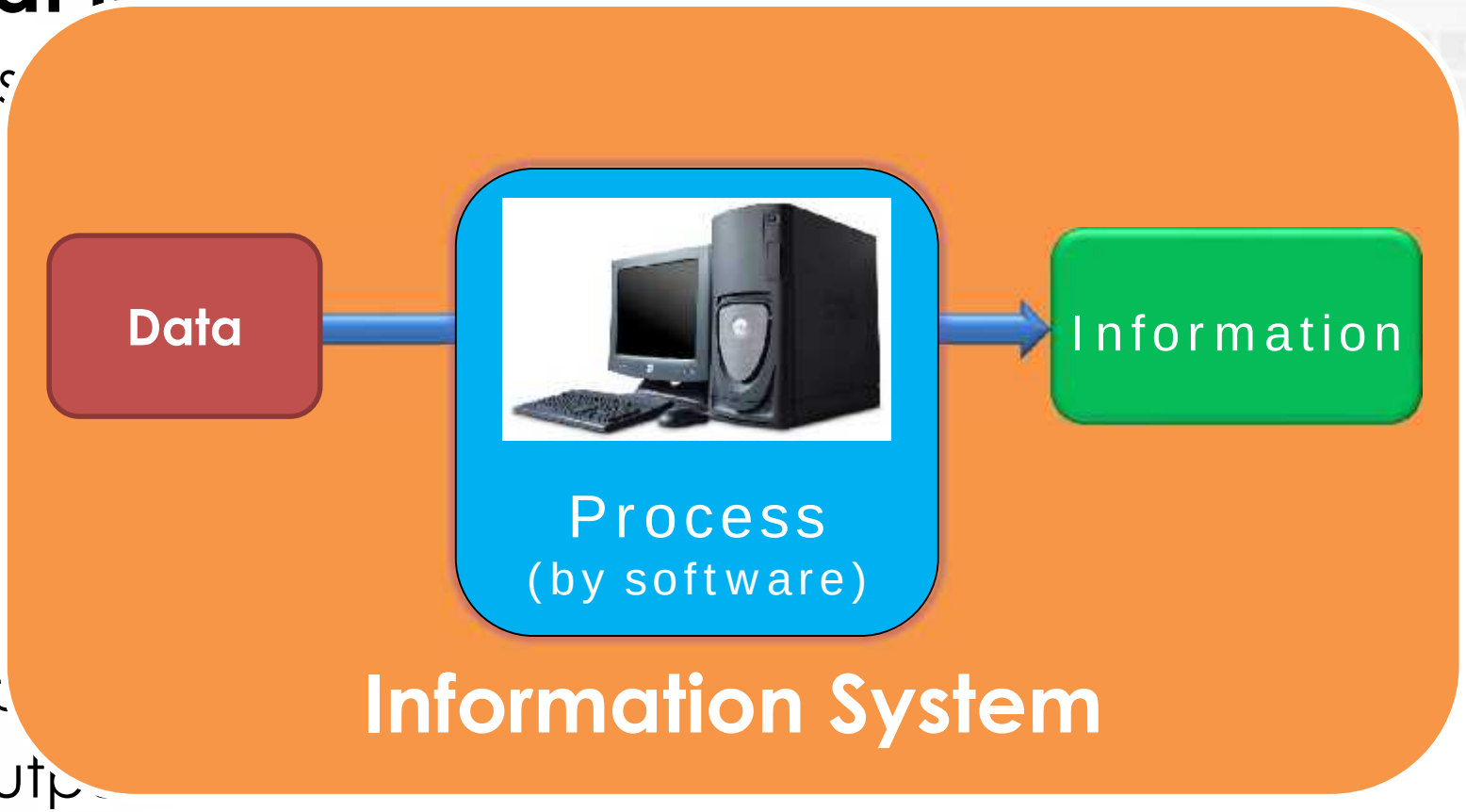
1. Data and Information Related terms

What is a System?

- A s
  pu
- A
  pr

Wha

- IS
- Inp
- Outp



1. Data and Information Related terms

Software Engineering

- The engineering discipline of constructing Information Systems

Information and Communication Technology (ICT)

- The technology used to engineer ISs

1. Data and Information

- Data is a collection of raw facts
  - Numbers
  - Words
  - Measurements
  - Observations
  - Description of things

- Information is the processed outcome of data.
  - It is derived from data

Saman		Kamal
	Nuwan	
Maths		English
	50	
60		70 80
90	40	100

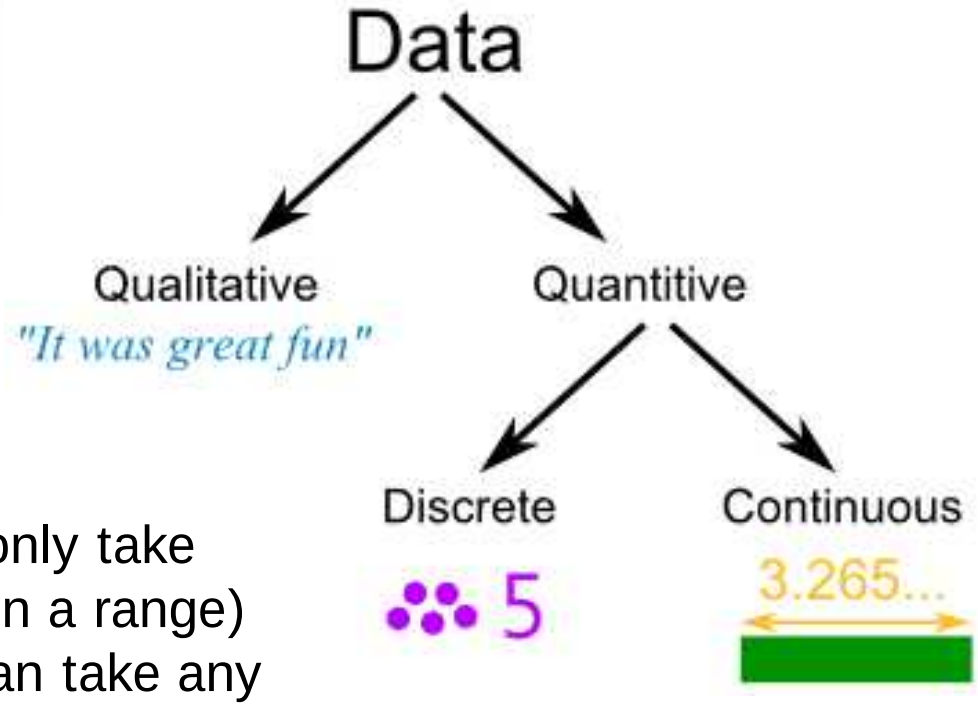
Saman	Maths	60
Kamal	Maths	100
Nuwan	Maths	40
Saman	English	50
Kamal	English	70
Nuwan	English	80

1. Data and Information
What is Information?

- Information provide meaningful values to the receiver
  - ✓ **Timely -** Information should be available when required
  - ✓ **Accuracy -** Information should be correct
  - ✓ **Completeness -** Information should be complete

2. Data Types
Qualitative vs. Quantitative Data

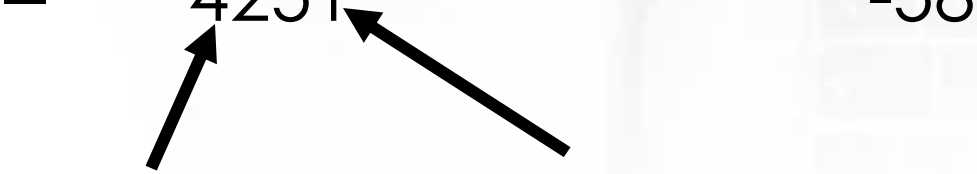
- Quantitative Data : Numerical Information
- Qualitative Data : Descriptive data



Discrete data: Can only take certain values (within a range)
Continuous Data: Can take any value (within a range)

2. Data Types
Numeric data

- **Integer numbers**
  - Whole numbers, + or -
  - 4251 -582



- **Real (Decimal) numbers**
  - All numbers including everything between integers
  - 0.23, 0, 5½, -2.3,

2. Data Types
Character Data

Single character

Numeric

0, 1, 2, ……

Alphabetic

A, B, C, a, b, c, ……

Special

#, @, %, (, $, &, …..

Multiple characters (String)

Numeric

349, 53.781, ……

Alphabetic

Cat, Software, ……

Special

(#%$), …..

Alpha-numerical

DIT451789#, ……..

2. Data Types Summary

What are the other data types?

2. Data Representation

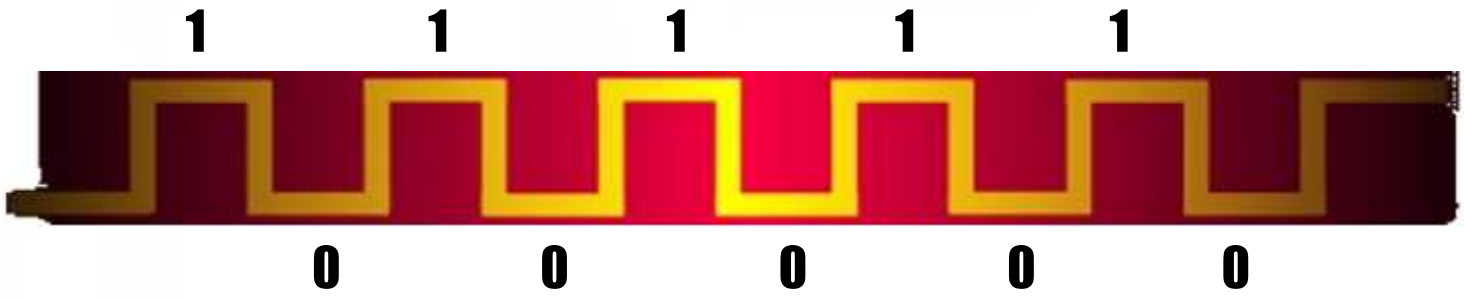
- Data representation refers to the form in which data is stored, processed, and transmitted
- Devices such as smartphones, iPods, and computers store data in digital formats that can be handled by *electronic circuitry*.
- Digitization is the process of converting information, such as text, numbers, photo, or music, into digital data that can be manipulated by electronic devices.

2. Data Representation

- The **Digital Revolution** has evolved through four phases, beginning with big, expensive, standalone computers, and progressing to today's digital world in which small, inexpensive digital devices are everywhere.

3. Data Representation in Computers
How do computers represent data?

- Computers are digital
  - Recognize only **two discrete states**: **on** or **off**
  - Computers are electronic devices powered by electricity, which has only two states, on or off
  - Binary number system (0,1) is used for processing

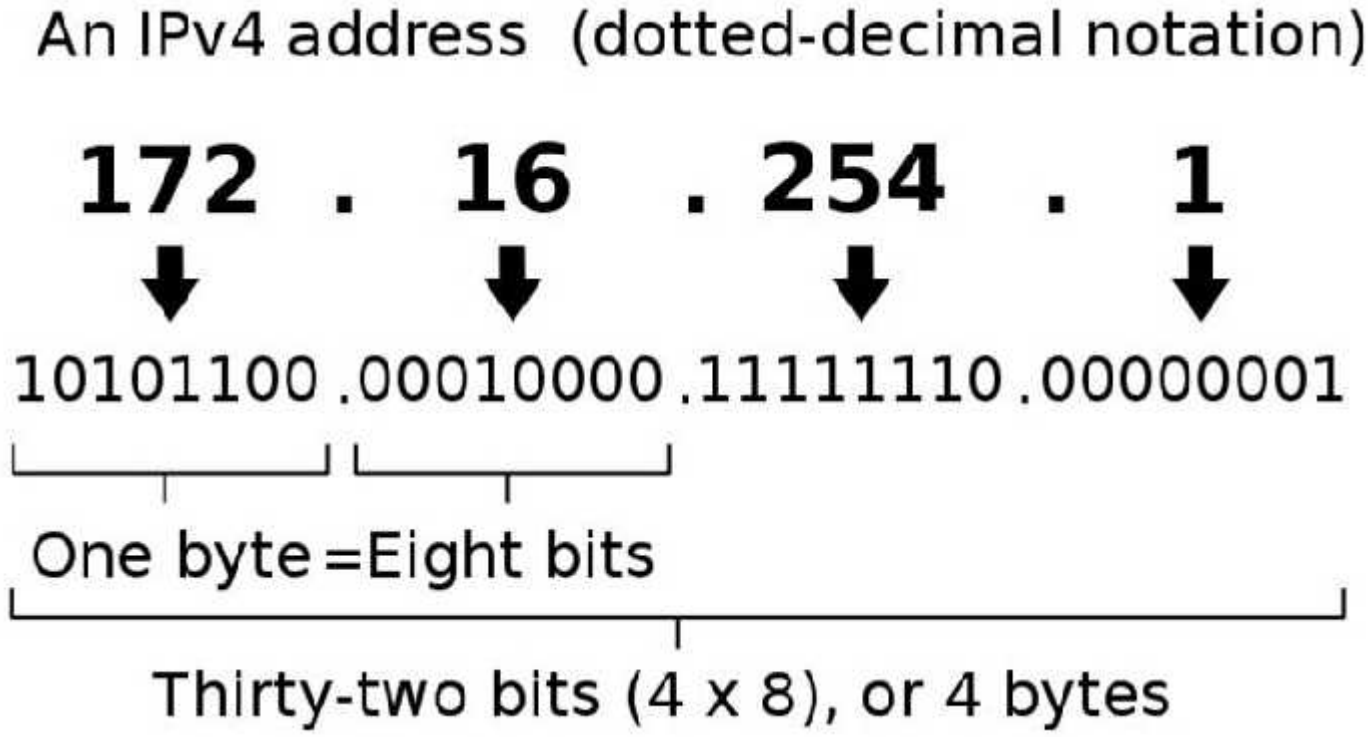


What is the meaning of “Digital”?
What are the advantages of using binary number system?

3. Data Representation in Computers
How do computers represent data?

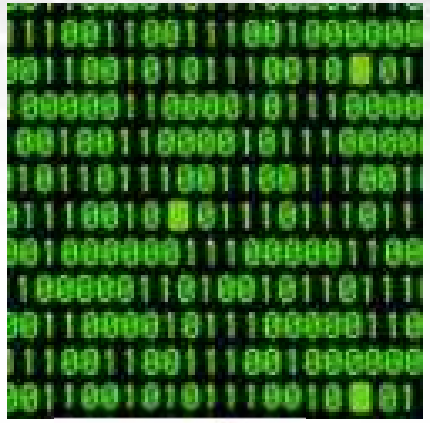
Example of the use of binary numbers in data communication

- IP addressing



3. Data Representation in Computers
How do we store data in a computer?

honda	789	234	baldwin	112	1545
toyota	1470	366	oscar	189	2141
dallas	2932	990	love	415	2989
miami	211	230	American	111	1096
coupe	599	334	history	272	2716
sedan	1693	524	white	284	3058
2000	896	641	black	221	2327
2004	3711	1124	time	145	907
black	2391	972	john 2007	391	4545
gold	709	508	action 2007	272	2601



3. Data Representation in Computers Binary number system

- The memory is made up of BITS and BYTES
- Single bit can hold a binary digit (0 or 1)
  - 8 Bits = 1 Byte
  - 1024 Bytes = 1KB
  - 1024 KB = 1 MB
  - 1024 MB = 1 GB
  - 1024 GB = 1 TB

$$2^{10} = 1024$$

104 KB: Kilobyte (KB or Kbyte)

50 Mbps: Megabits/sec

Terminology related to bits and bytes is extensively used to describe storage capacity and network access speed.

3. Data Representation in Computers
Binary number system

- Each BYTE can be addressed uniquely
- When the address is expressed in Binary, the number of maximum BITs used to write the address specifies the total number of locations available
- If **n** number of BITs are available then the total number of locations available is $\mathbf{2^n}$
- If we have 32 BITs then we can have 4GB of Memory ($2^{32} = 4$ GB)

What is the maximum amount of memory for 64bit computer?

128 GB

3. Data Representation in Computers BCD (Binary Coded Decimal)

- 4 bit code (for numeric values only)

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001
12	0001 0010

3. Data Representation in Computers

ASCII (American Standard Code for Information Interchange)

- ASCII
  - 7 bit code for all 128 characters
  - A=1000001
  - Fundamentally, computers just deal with numbers. The letters and other characters are stored by assigning a number for each one
- Extended ASCII
  - This system is an 8-bit system and allows the system to store up to 256 different characters

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011

Symbol	Decimal	Binary
a	97	01100001
b	98	01100010
c	99	01100011

3. Data Representation in Computers ASCII

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Symbol	Decimal	Binary
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110
g	103	01100111
h	104	01101000
i	105	01101001
j	106	01101010
k	107	01101011
l	108	01101100
m	109	01101101
n	110	01101110
o	111	01101111
p	112	01110000
q	113	01110001
r	114	01110010
s	115	01110011
t	116	01110100
u	117	01110101
v	118	01110110
w	119	01110111
x	120	01111000
y	121	01111001
z	122	01111010

3. Data Representation in Computers Unicode

- **Unicode** provides a unique number for every character
  - no matter what the platform
  - no matter what the program
  - no matter what the language

0D85	අ	SINHALA LETTER AYANNA = sinhala letter a	0B85	அ	TAMIL LETTER A
0D86	ආ	SINHALA LETTER AAYANNA = sinhala letter aa	0B86	ஆ	TAMIL LETTER AA
0D87	ඇ	SINHALA LETTER AEYANNA = sinhala letter ae	0B87	இ	TAMIL LETTER I
0D88	ඈ	SINHALA LETTER AEEYANNA = sinhala letter aae	0B88	ஈ	TAMIL LETTER II
0D89	ඉ	SINHALA LETTER IYANNA = sinhala letter i	0B89	உ	TAMIL LETTER U
0D8A	ඊ	SINHALA LETTER IIYANNA	0B8A	ஊ	TAMIL LETTER UU
			0B8B		<reserved>
			0B8C		<reserved>
			0B8D		<reserved>
			0B8E	எ	TAMIL LETTER E
			0B8F	ஏ	TAMIL LETTER EE

How many bits are used to represent a character? 8

How many characters can be represented? 255

3. Data Representation in Computers Unicode

Independent vowels

0D85 අ SINHALA LETTER AYANNA
= sinhala letter a

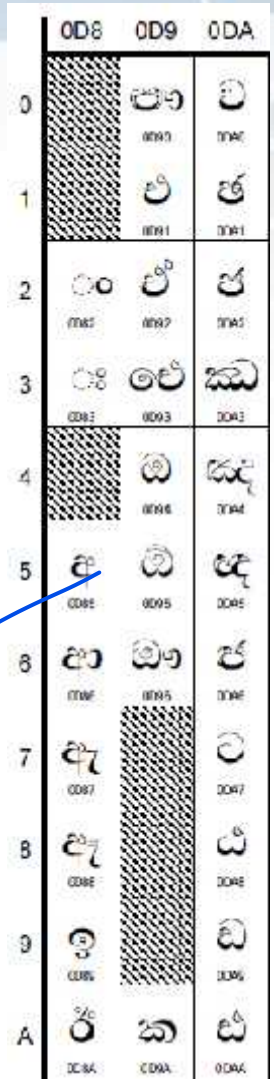
0D86 ආ SINHALA LETTER AAYANNA
= sinhala letter aa

0D87 ඇ SINHALA LETTER AEYANNA
= sinhala letter ae

0D88 ඈ SINHALA LETTER AEEYANNA
= sinhala letter aae

0D89 ඉ SINHALA LETTER IYANNA
= sinhala letter i

0D8A ඊ SINHALA LETTER IIYANNA



Independent vowels

0B85 அ TAMIL LETTER A
0B86 ஆ TAMIL LETTER AA
0B87 இ TAMIL LETTER I
0B88 ஈ TAMIL LETTER II
0B89 உ TAMIL LETTER U
0B8A ஊ TAMIL LETTER UU
0B8B <reserved>
0B8C <reserved>
0B8D <reserved>
0B8E எ TAMIL LETTER E
0B8F ஏ TAMIL LETTER EE

4. Error checking Parity

- The parity is used in error checking, to find errors that may occur during data storing/transmission
- A parity bit is a single bit added to a binary data transmission used to indicate if whether the 0's and 1's within that data transmission is an even or odd number

4. Error checking Parity

- There are two modes of parity
  - Odd parity:- The number of 1-bits (including the parity bit) must be an odd number
  - Even parity:- The number of 1-bits (including the parity bit) must be an even number
- A single bit is appended to each data chunk (either as the least or most significant bit)
  - Makes the number of 1 bits even/odd

Example: even parity	Example: odd parity
1000000(1)	1000000(0)
1111101(0)	1111101(1)
1001001(1)	1001001(0)

4. Error checking
Parity – Example

- You receive a binary word "11000101"
  - **odd parity** is used
  - Most significant bit is used for parity

- Is the binary word correct?
- The answer is no:
  - There are 4 1-bits, which is an even number
  - We are using an odd parity
  - So there must be an error.

4. Error checking Parity – Example2

- Assume we are using **even parity** with 7-bit ASCII.
- The letter V in 7-bit ASCII is encoded as 0110101.

- How will the letter V be transmitted with parity?
  - Because there are four 1s (an even number), parity is set to zero.
  - This would be transmitted as: 0110101(0).

4. Error checking Parity – Exercises

- Add the odd parity bit as the least significant bit for the following data
  - 0100101 (0)
  - 1101101 (0)
  - 1011001 (1)
  - 0011010 (0)
  - 1111011 (1)

4. Error checking Parity – Exercises

- Verify the correctness of the following data with the even parity bit on the most significant bit
  - 01010001 Wrong
  - 10100101 Correct
  - 01101101 Wrong
  - 01111011 Correct
  - 10011010 Correct

5. Data Compression

- To reduce file size and transmission times, digital data can be compressed.
- Data compression refers to any technique that recodes the data in a file so that it contains fewer bits.
- Compression is commonly referred to as "zipping"
- The process of reconstituting zipped files is called extracting or unzipping.
- Compressed files may end with a .zip, .gz, .pkg, or .tar.gz

Summary

1. Data and Information
2. Data types
3. Data Representation in Computers
4. Error checking
5. Data Compression

Thank You

Lecture 02 – Post-lecture activities

Parity & Binary Exercises

Lecture 03

Computer Architecture

Lecture 03 – Pre-lecture activities

Diagram of 8086, Bus Architecture with 3 bus types

COMPUTER ARCHITECTURE

IT1020 Introduction to Computer Systems

Part 01 – Computer Fundamentals

CONTENT

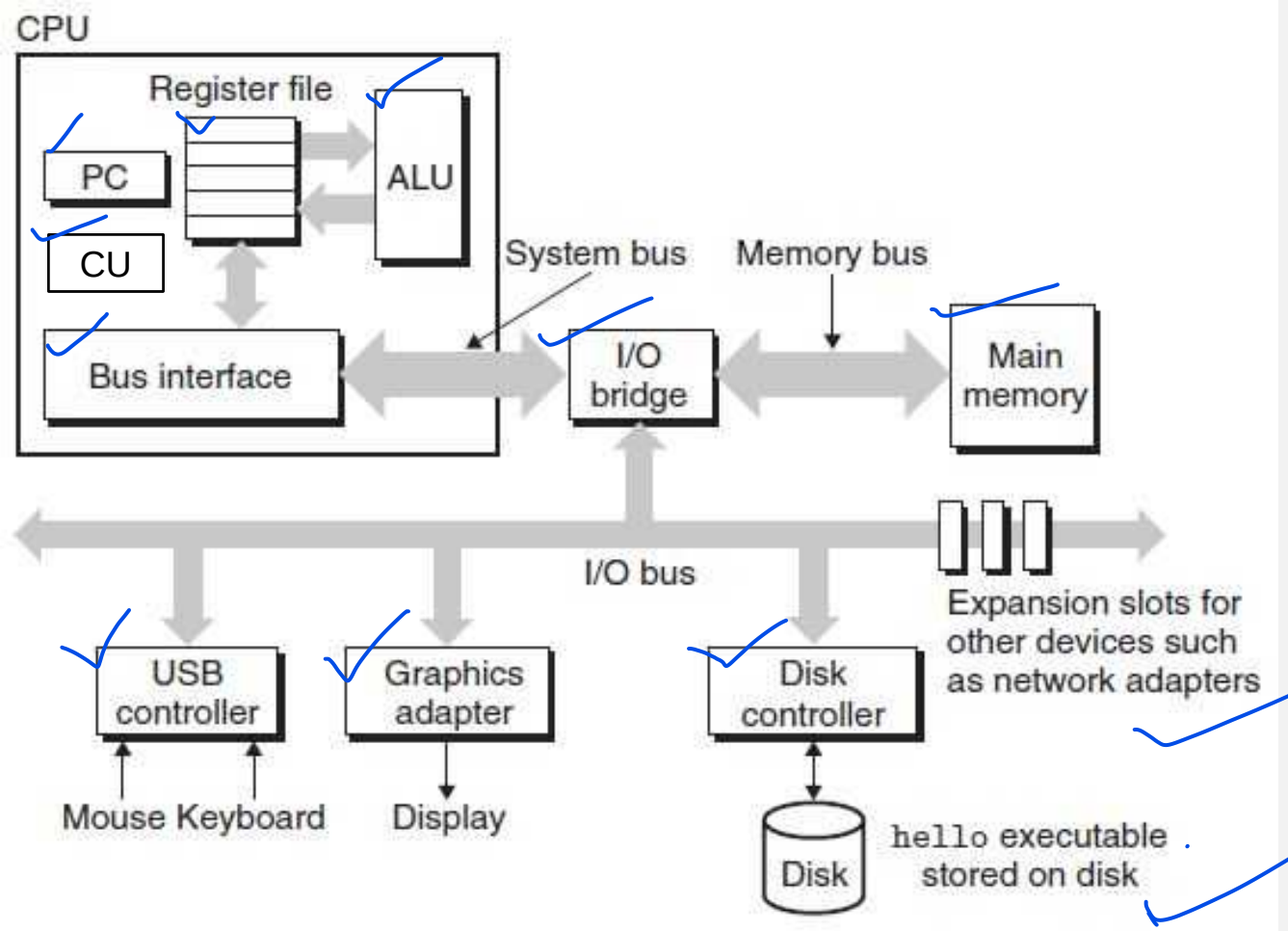
- This lecture covers:
  - fundamentals of computer organization,
  - the heart of a computer system, CPU, its structure and functionality,
  - how CPU is seen by programmers and CPU design engineers
  - instruction set architecture of a processor.

COMPUTER ORGANIZATION

- Different groups of engineers see computer different manner
- Hardware Engineers see how different components work, their configuration, how they are organized – Computer Organization
- Software Engineers see how computer can be used for different applications, how their high level language codes can be executed in machine, How easily it can be used for programs – Instruction Set Architecture

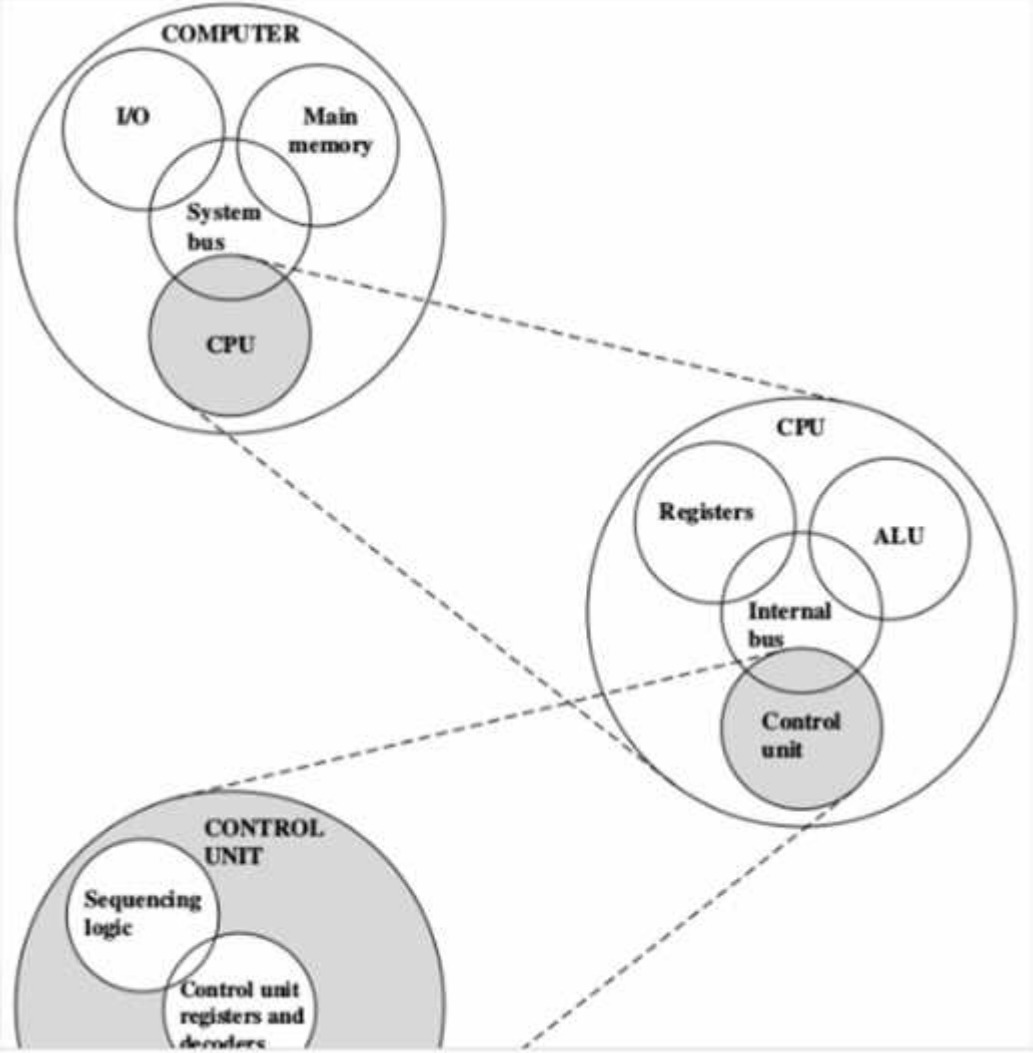
COMPUTER ORGANIZATION

Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system.



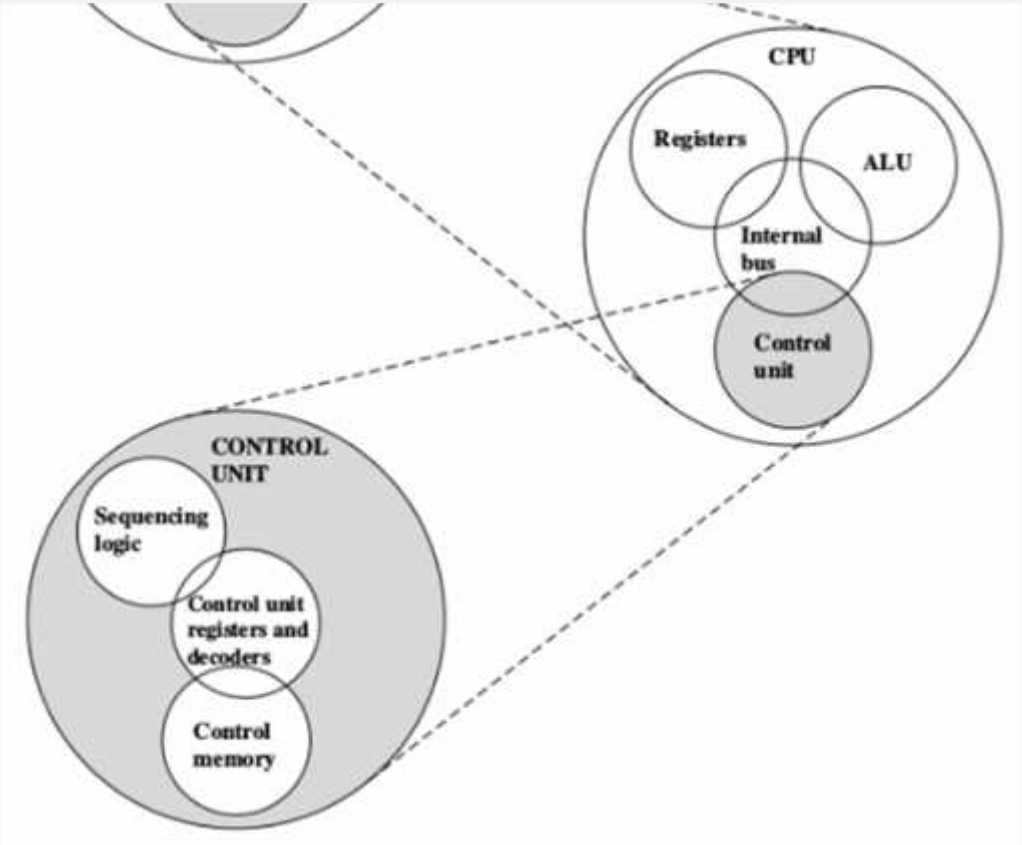
COMPUTER ORGANIZATION

- CPU: Controls the operation of the computer and performs its data processing functions.
- Main Memory: Stores data and Instructions (Programs)
- I/O: Moves the data between the computer and its external environment.
- System Interconnections (System Bus): Mechanism that provides for communication among CPU, MM, and I/O.

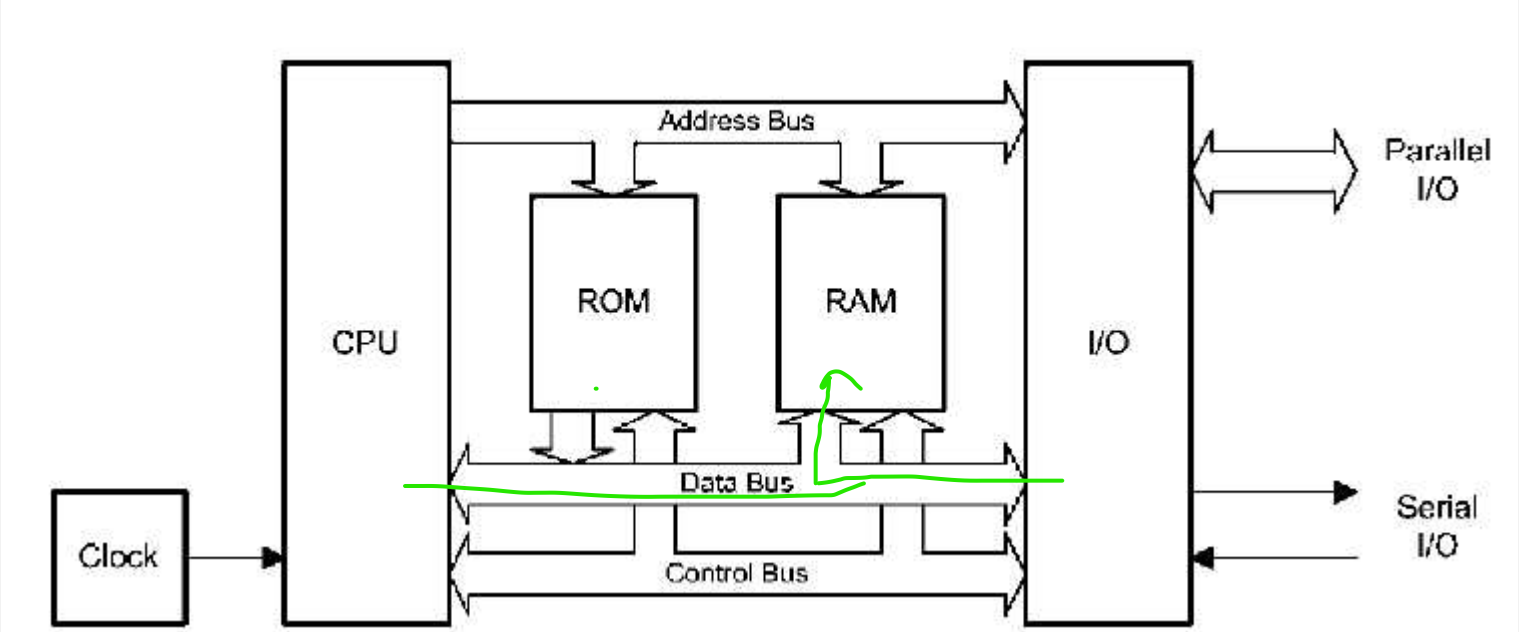


COMPUTER ORGANIZATION

- ALU: Performs the computer's data processing functions
- Registers: Provides storage internal to the CPU
- Control Unit: A digital circuitry within the processor that coordinates the sequence of data movements into, out of, and between a processor's sub-units

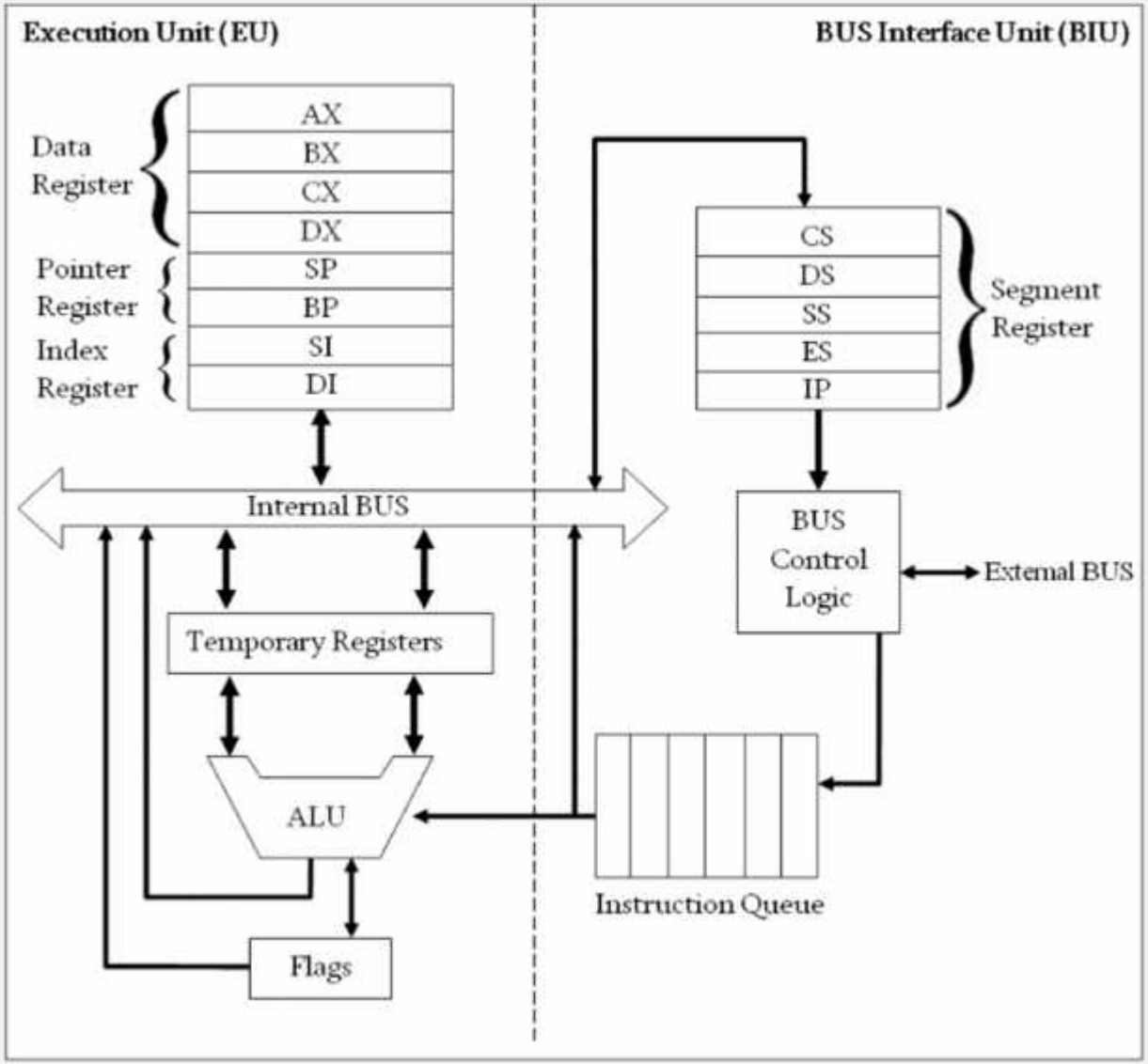


MODERN COMPUTER



- Address Bus: CPU specify the address of MM or I/O, Uni-directional
- Data Bus: Transfer data/instruction to and from CPU
- Control Bus: Carries commands from CPU and status signals to CPU

8086 ARCHITECTURE



8086 REGISTERS

They are grouped into several categories as follows:

- Four general-**purpose** registers, AX, BX, CX, and DX.
- Four special-**purpose** registers, SP, BP, SI, and DI.
- Four segment registers, CS, DS, ES, and SS.
- The Program Counter/Instruction Pointer
- The status flag register, FLAGS

8086 REGISTERS

General Purpose	Special Purpose (Addressing registers)
• AX is the **accumulator**. It is used for all input/output operations and some arithmetic operations. • BX is the **base register**. It can be used as an address register. • CX is the **count register**. It is used by instructions which require to count (Ex: Looping) • DX is the **data register**. It is used for some input/output and also when multiplying and dividing.	• The **addressing registers** are used in memory addressing operations, such as holding the source address of the memory and the destination address. • SI is the **source index** and is used with extended addressing commands. • DI is the **destination index** and is used in some addressing modes. • BP is the **base pointer**. • SP is the **stack pointer**.

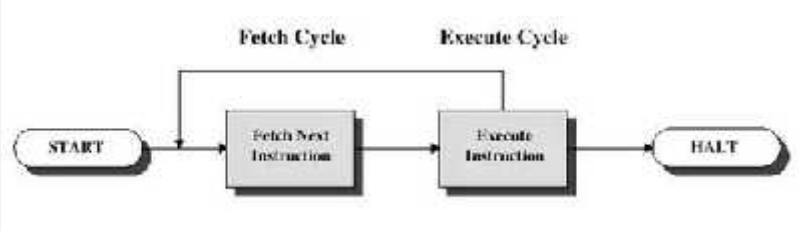
8086 REGISTERS

- The two status registers have 16 bits and are called the Instruction Pointer (IP) or Program Counter (PC) and the flag register (F)
- IP is the instruction pointer and contains the address of the next instruction of the program. This is called as Program Counter (PC)
- Flag register holds a collection of 16 different conditions. (Ex: result is zero or not, there is a "carry" etc)

8086 REGISTERS

- Segments registers: There are four areas of memory called segments, each of which are 16 bits and can thus address up to 64KB (from 0000h to FFFFh).
  - Code segment (cs register), where the program code is stored.
  - Data segment (ds register), where data from the program is stored
  - Stack segment (ss register), where the stack is stored.
  - Extra segment (es register), a spare segment
- All addresses are with reference to the segment registers.

INSTRUCTION CYCLE



- The basic function performed by a computer is execution of a program
- Program is a well defined set of Instructions
- Processing a single instruction is called instruction cycle
- It has two steps referred as fetch cycle and execute cycle

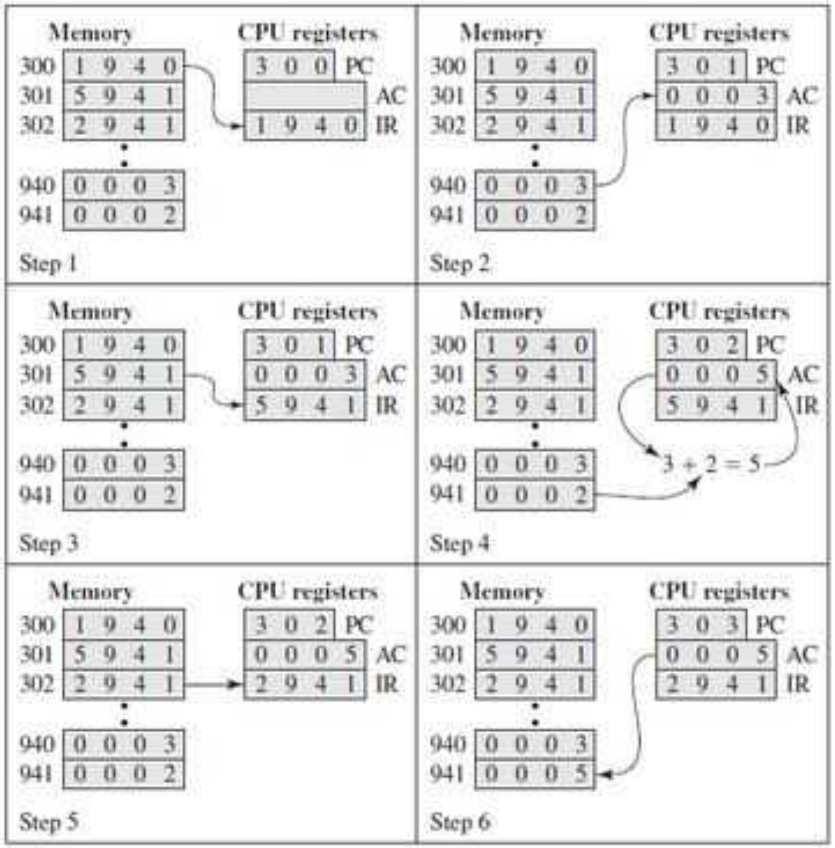
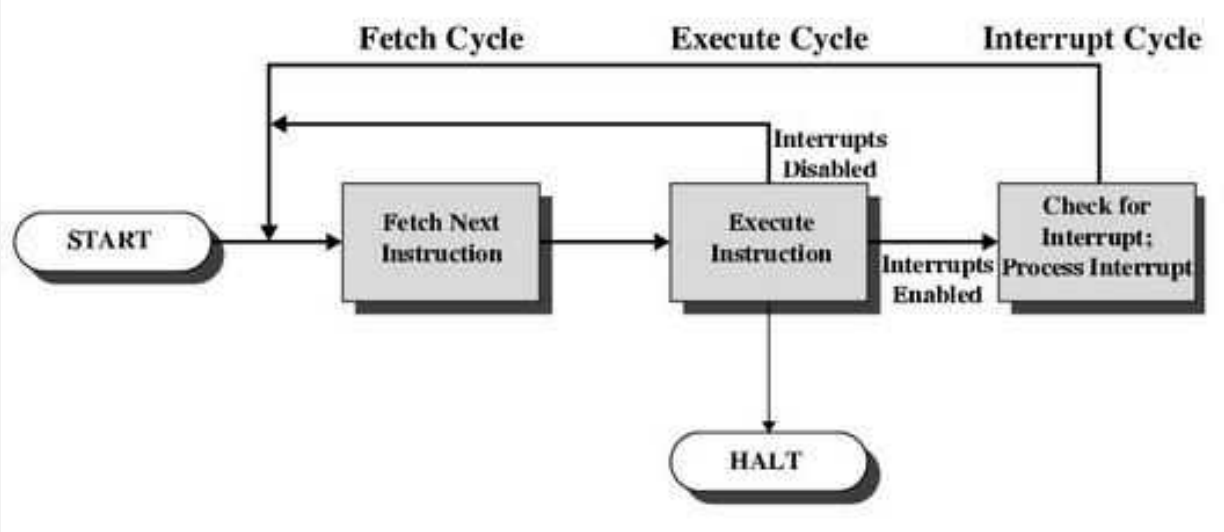


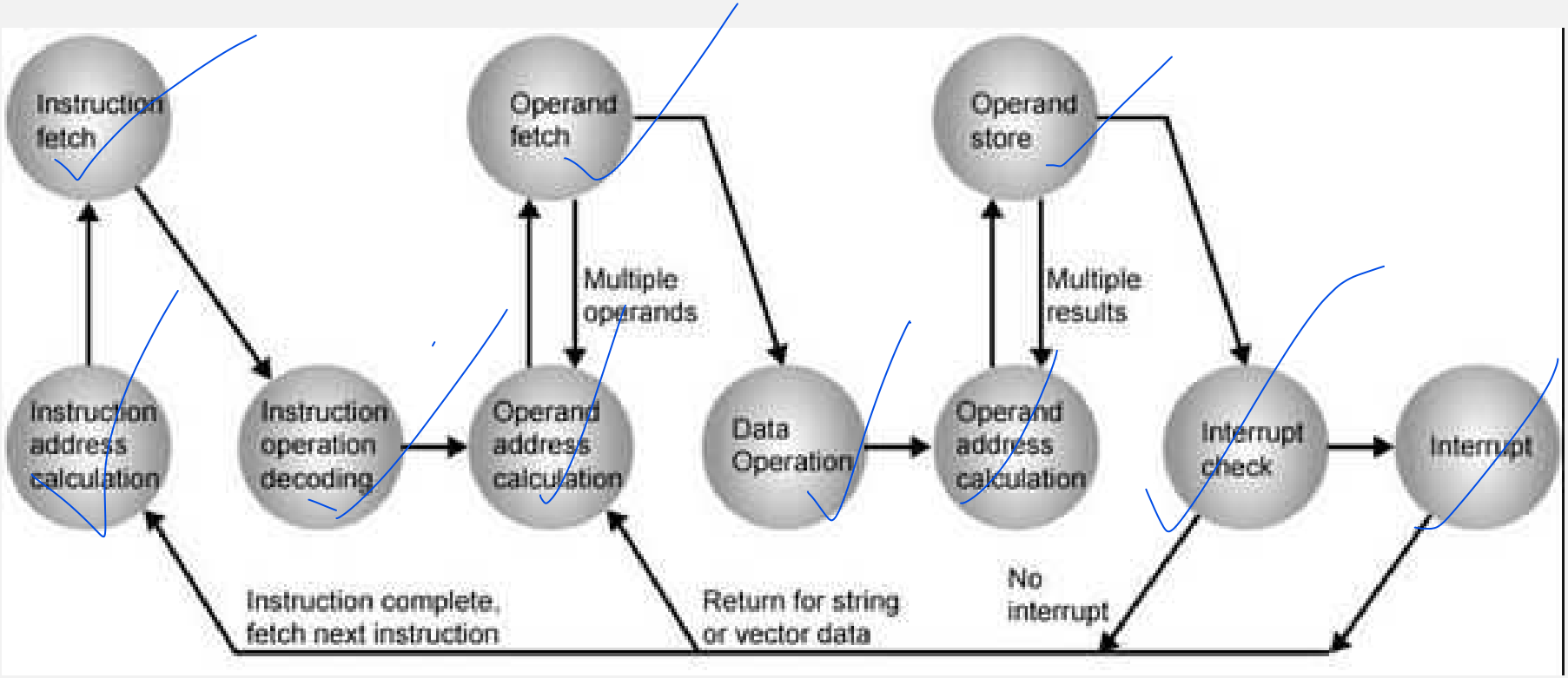
Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

INSTRUCTION CYCLE WITH INTERRUPT

- All computers provide a mechanism by which other modules (I/O, Memory) may **interrupt** the normal processing of the processor
- Processor engaged in executing other instructions while slow external devices (ex. I/O module) become ready and send interrupt request to processor
- Processor responds to such interrupt request by running interrupt handler



STATE DIAGRAM OF INSTRUCTION CYCLE WITH INTERRUPT

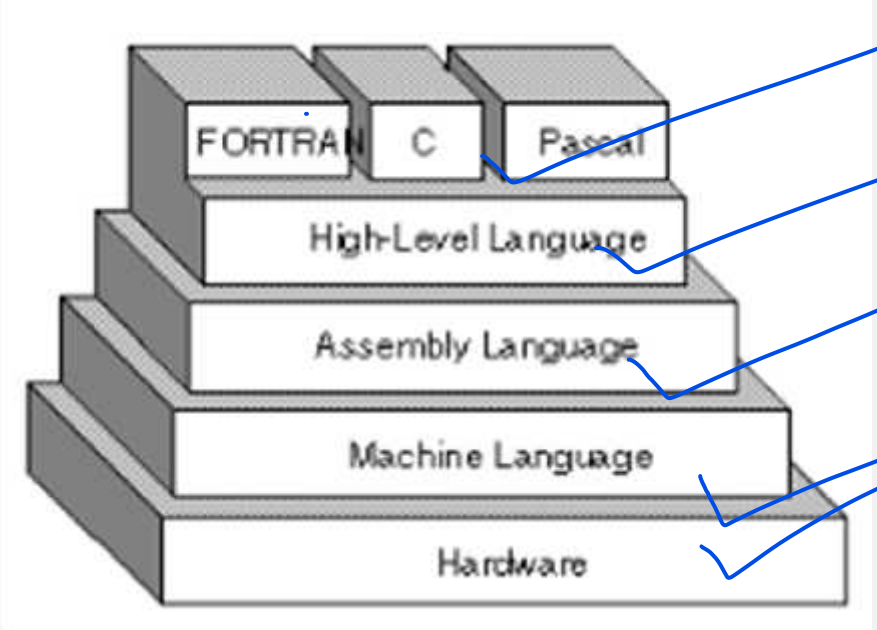


INSTRUCTION SET ARCHITECTURE (ISA)

- Instruction Set is a part of the computer that pertains to internal (or low level) programming, which is basically machine language.
- The instruction set provides commands to the processor.
- Instructions instruct (give orders to) the processor what it needs to do. Consider as "words" in processor's language.
- User codes, normally written in High Level Languages (closer to Natural Language) must convert to Machine Language to be run on the processor

INSTRUCTION SET ARCHITECTURE (ISA)

- Assembly Language has rich set of mnemonics to represent Machine Language Instructions
- Assembly Language commands are in a human readable format.



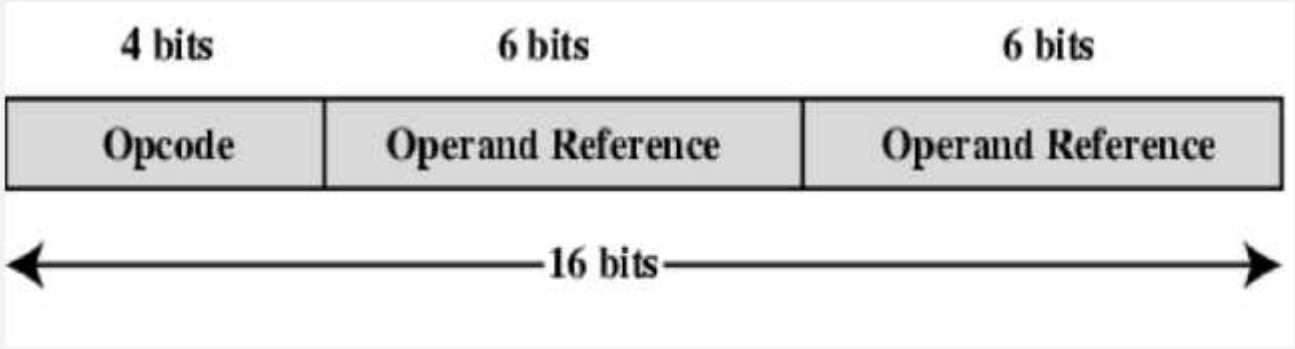

```
MOV  AX,DATA
MOV  DS,AX

MOV  AL,NUM1
ADD  AL,NUM2
```

INSTRUCTION SET ARCHITECTURE (ISA)

- Typical ISA defines:
  - How to access data in Registers, Memory, and other I/O devices
  - Mechanisms to transfer data and instruction to and from processor
  - Operations such as additions, subtractions which processor can execute
  - Control mechanisms such as branch, jump
- To be effective as a programmer or processor designer, one should know how ISA works.

INSTRUCTION FORMAT

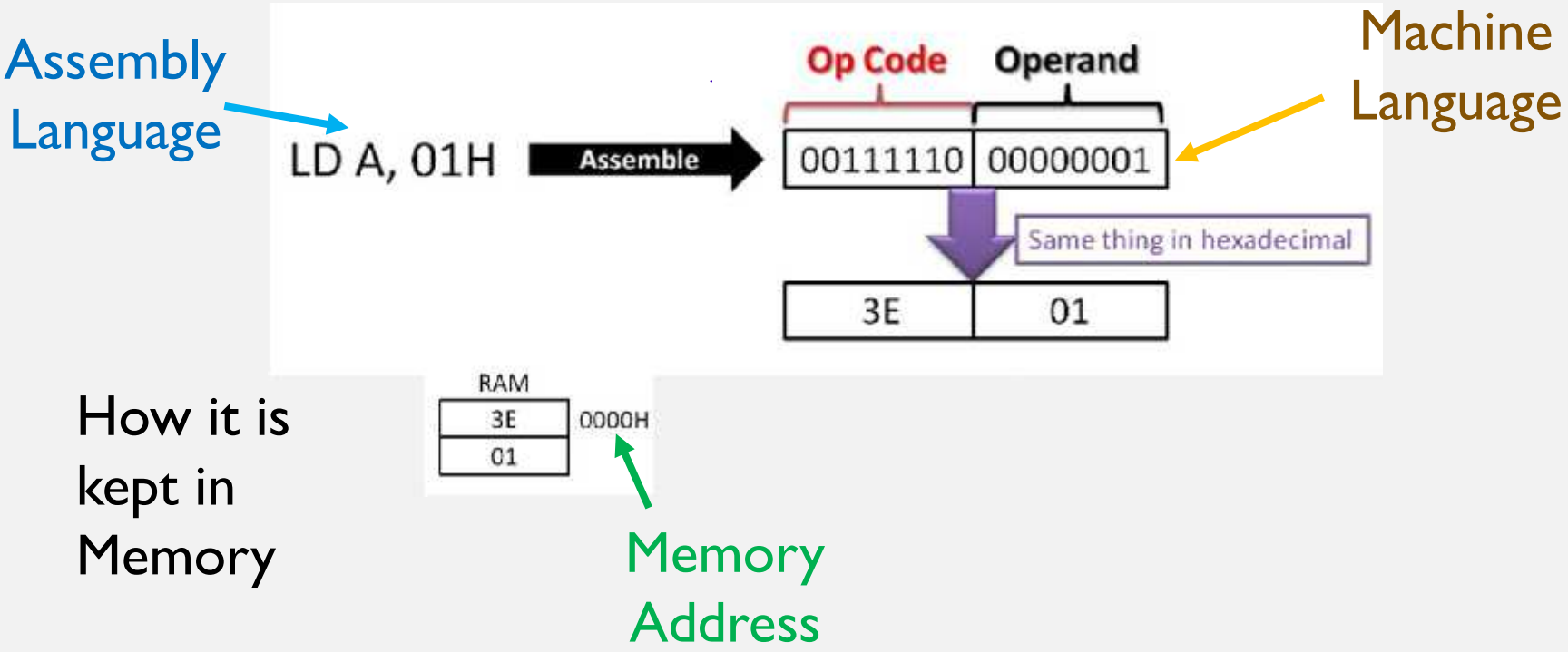


Opcode: Specifies the operation to be performed (Ex- ADD, SUB).

Operand: One or more inputs/outputs, or their source or destination

Next instruction reference: Tells the processor where to fetch the next instruction once the execution of this instruction is completed.

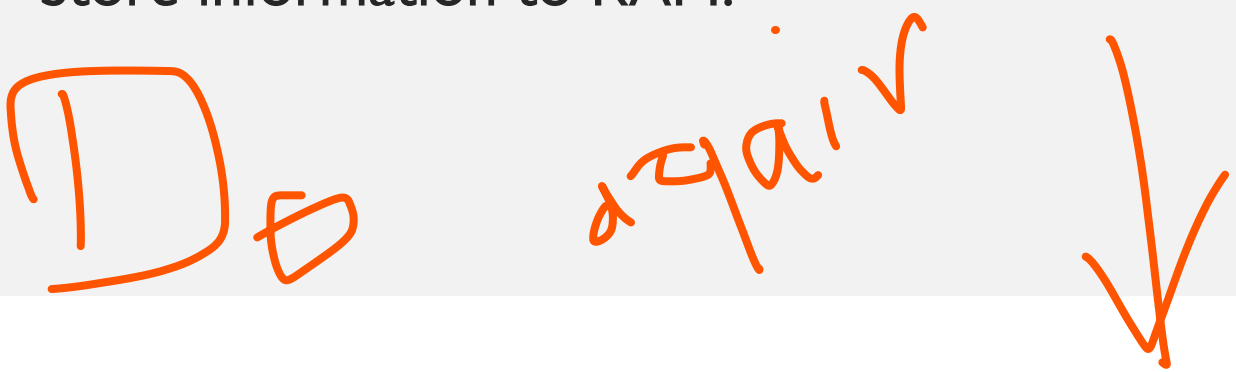
INSTRUCTION FORMAT



INSTRUCTION SET

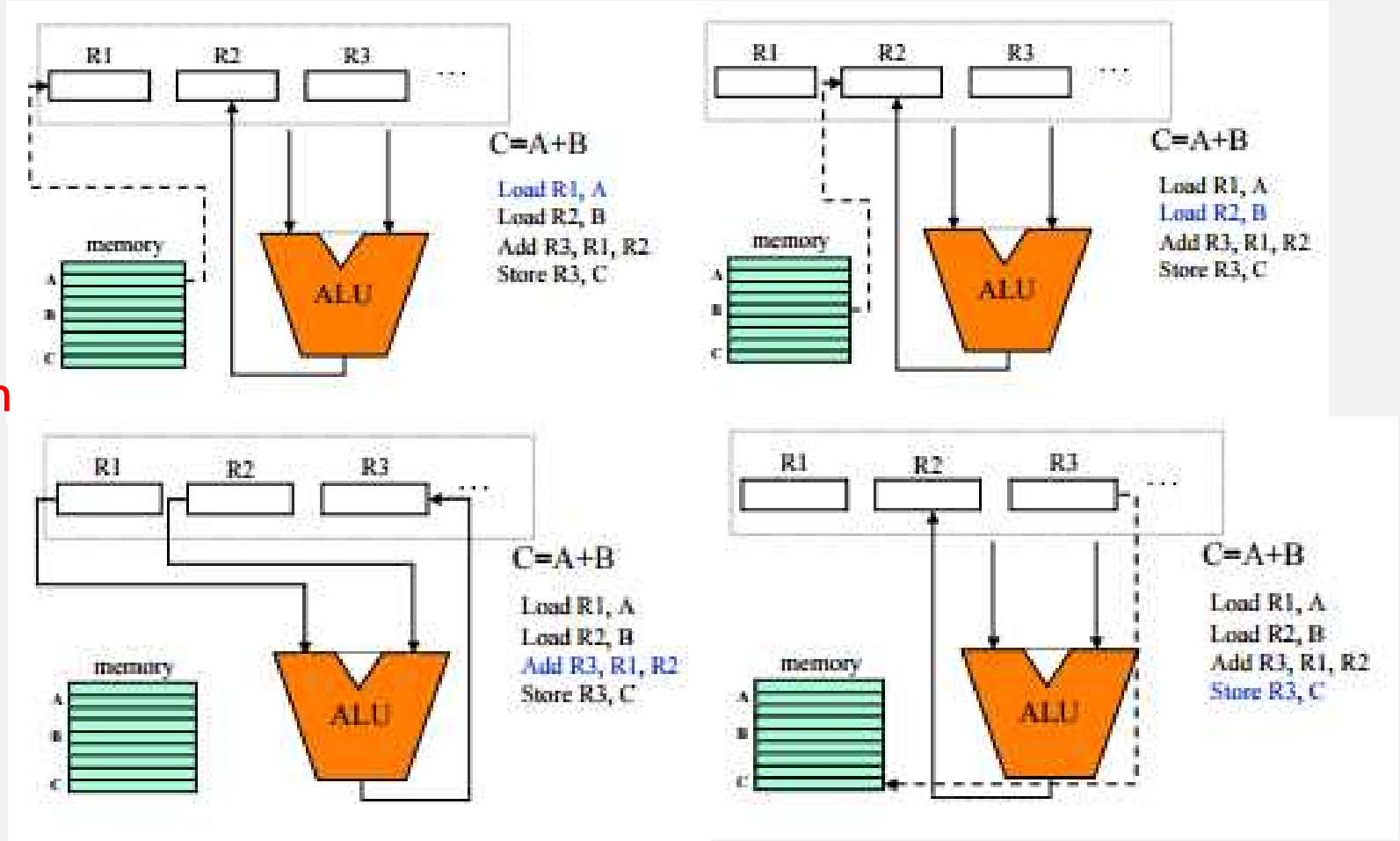
Sample Instructions

- **ADD** - Add two numbers.
- **CMP** - Compare numbers.
- **IN** - Input from port into AL or AX. Second operand is a port number.
- **JMP** - Unconditional Jump. Transfers control to another part of the program.
- **JNE** - Short Jump if first operand is Not Equal to second operand.
- **LOAD** - Load information from RAM to the CPU.
- **OUT** - Output information to device, e.g. monitor.
- **STORE** - Store information to RAM.

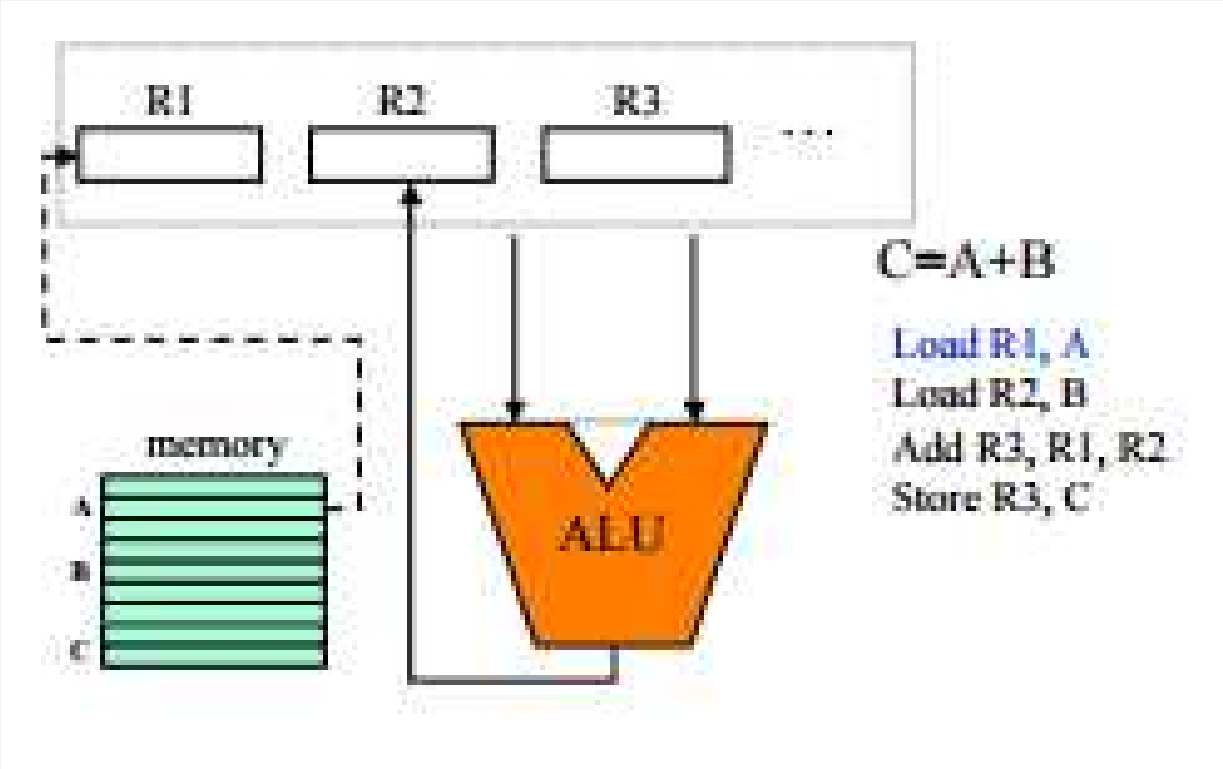


EXECUTION OF INSTRUCTIONS

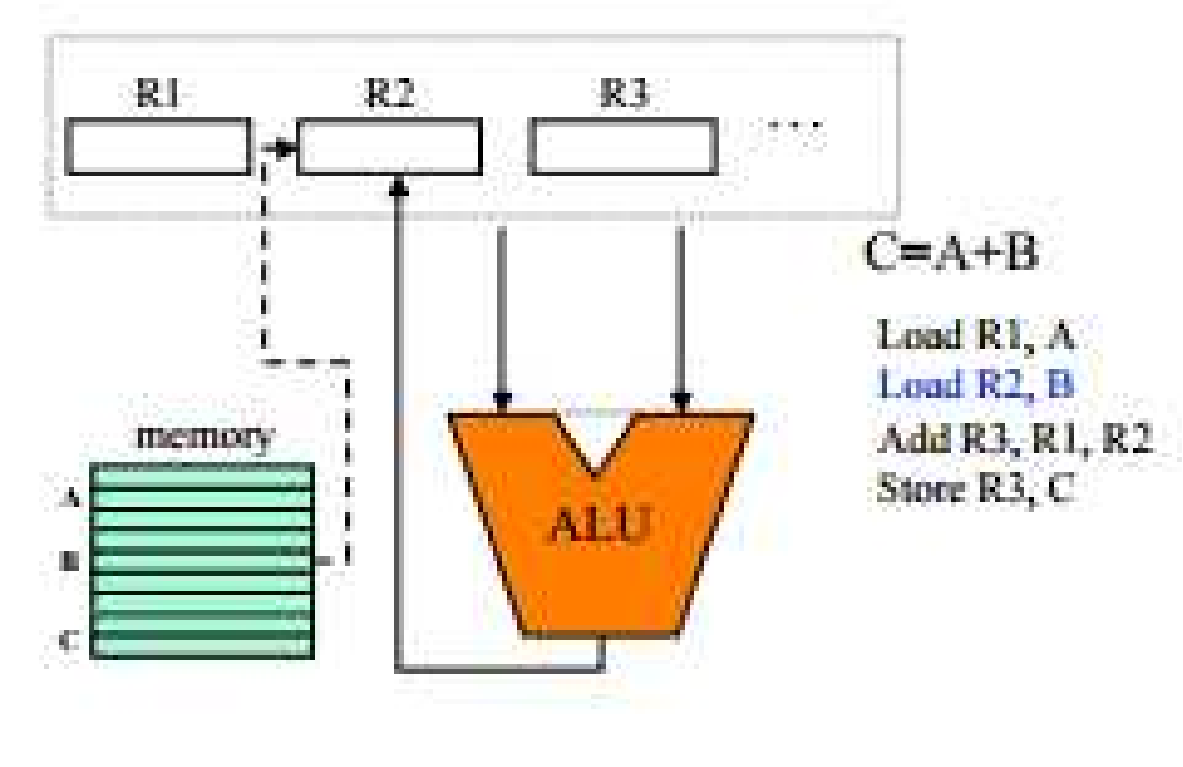
Processor Executes one instruction at a time.



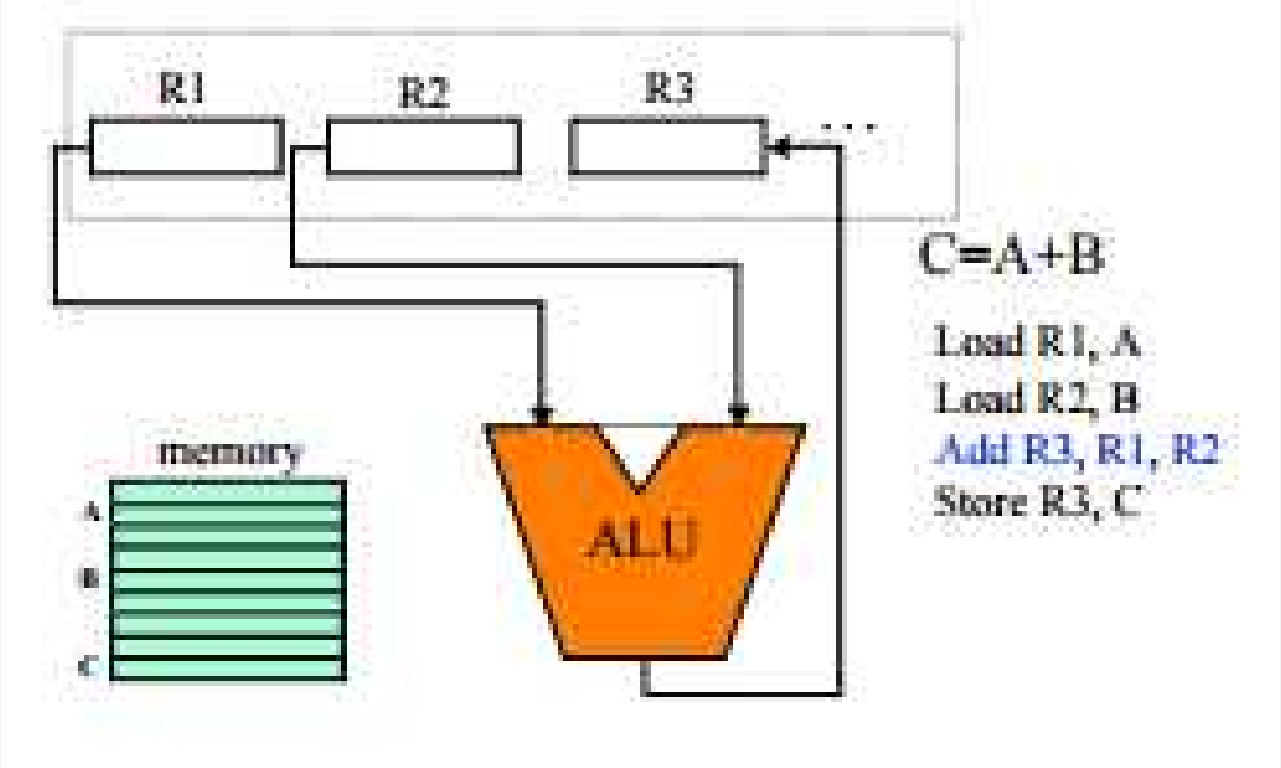
EXECUTION OF INSTRUCTIONS



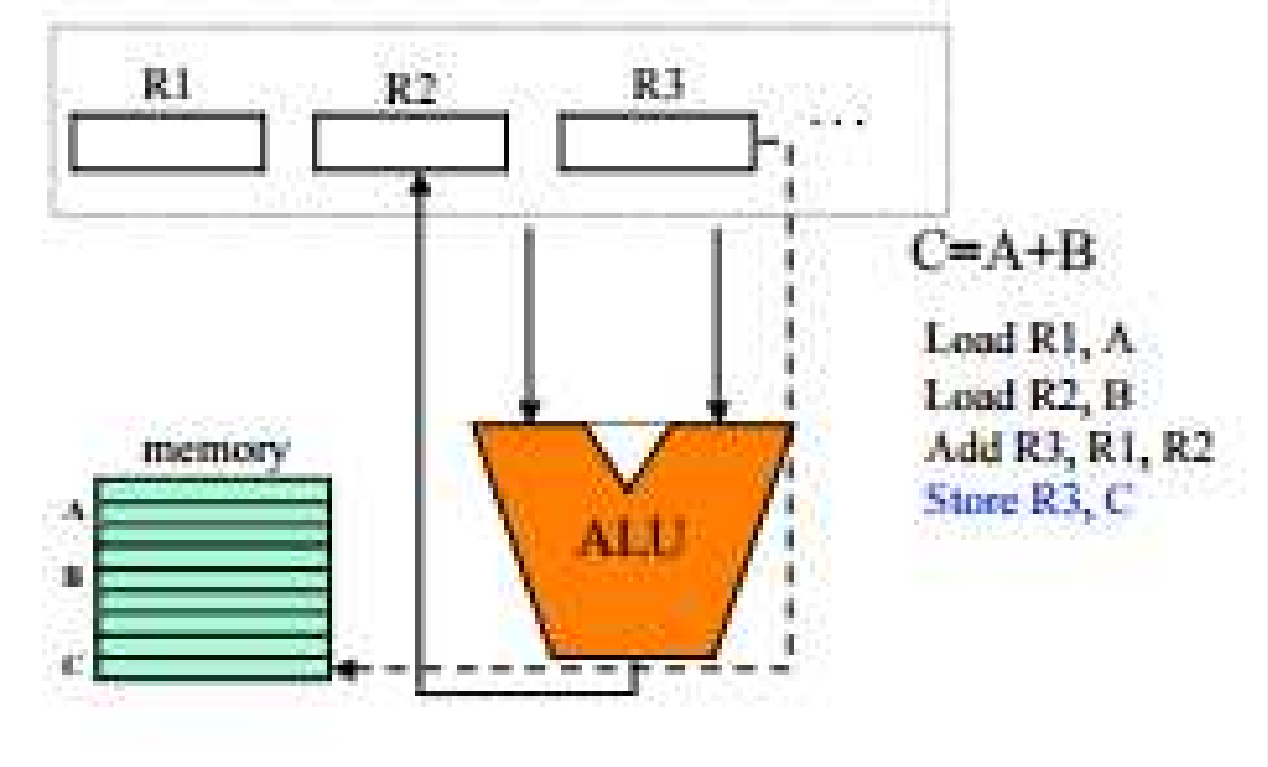
EXECUTION OF INSTRUCTIONS



EXECUTION OF INSTRUCTIONS



EXECUTION OF INSTRUCTIONS



CISC & RISC

- CISC:
  - Primary goal is to complete a task in as few lines of assembly as possible
  - Processor hardware complex; Needs less RAM to store the code; Instruction set is high-level, hence Compiler workload is low
  - Ex: MULT M1,M2

CISC & RISC

- RISC:
  - Simple instructions that can be executed within one clock cycle
  - Processor hardware simple; Need more RAM; Instruction set Low-Level, hence Compiler workload high
  - Ex: LOAD A, M1
    - LOAD B, M2
    - PROD A, B
    - STORE M1,A

CISC & RISC

CISC	RISC
• Includes multi-clock complex instructions • Emphasis on hardware • Memory-to-memory: "LOAD" and "STORE" incorporated in instructions • Small code sizes, high cycles per second • Transistors used for storing complex instructions	• Single-clock, reduced instruction only • Emphasis on hardware • Register to register: "LOAD" and "STORE" are independent instructions • Low cycles per second, large code sizes • Spends more transistors on memory registers

CISC & RISC

CISC	RISC
• Intel X86 family, AMD processors are heavily used in desktop, laptop and server computers	• SPARC and Power PC are used in desktop computers and game consoles • RISC processors are heavily used in real-time embedded systems such as mobile phones, washing machines, Routers • Raspberry pi and Arduino • IoT drives by RISC processors

THANK YOU

Post-lecture activities

Instruction Set of Core i5, ARM processor

Lecture 04

Operating system

Lecture 04 – Pre-lecture activities

Types of OSs, Windows, Linux, MacOS descriptions

IT1020 – Introduction to Computer Systems

Lecture 4

Operating Systems

Agenda

- What Operating Systems Do
- Computer-System Organization
- Operating-System Structure
- Operating-System Operations
- Process, Memory & Storage Management
- Protection and Security
- Computing Environments
- Open-Source Operating Systems

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments

What is an Operating System

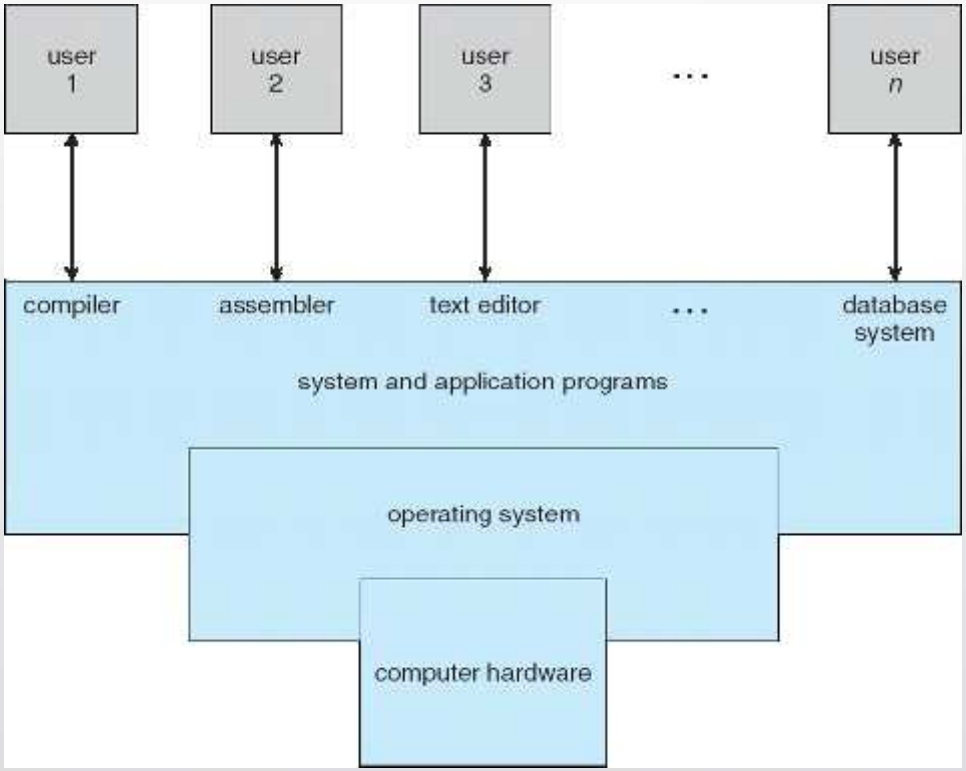
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Manage the computer hardware in an efficient manner

Operating-System Structure

Computer system can be divided into four components:

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- Users
  - People, machines, other computers

Operating-System Structure



Operating System Definition

- OS as a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
- OS as a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

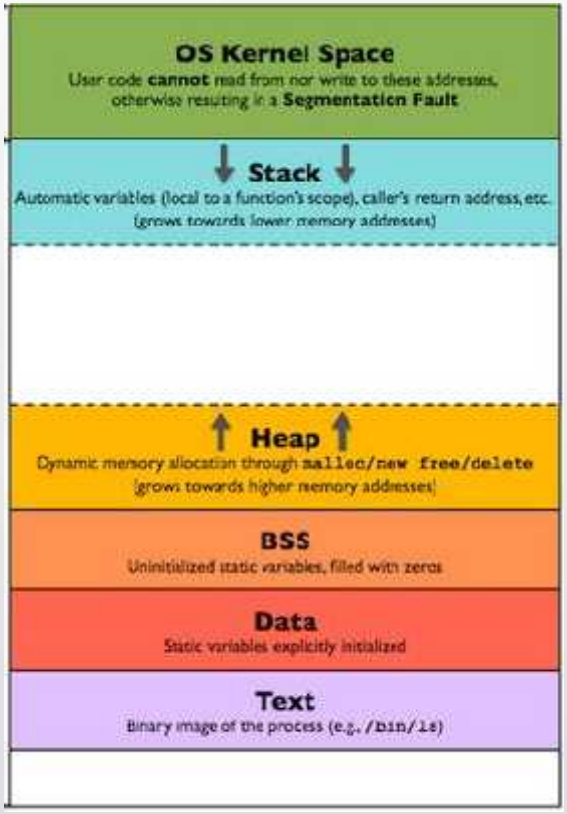
Operating System Definition – Common Beliefs

- "Everything a vendor ships when you order an operating system"
- "The one program running at all times on the computer"
  - kernel.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

Computer-Start up – How OS comes to work?

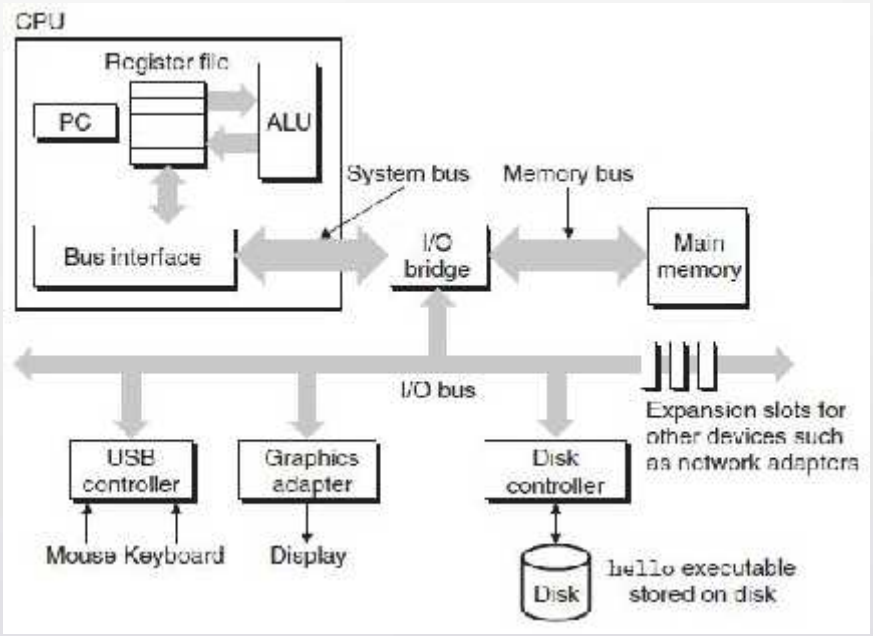
- **BIOS program** is loaded at power-up or reboot
  - Typically stored in EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

System Memory Map



Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt



Computer System Operation: Interrupts

- Interrupt transfers control to the interrupt service routine
- Interrupt architecture must save the address of the interrupted instruction
- A trap (or an exception) is a software-generated interrupt caused either by an error or a user request
- An operating system is interrupt driven

Operating-System Operations

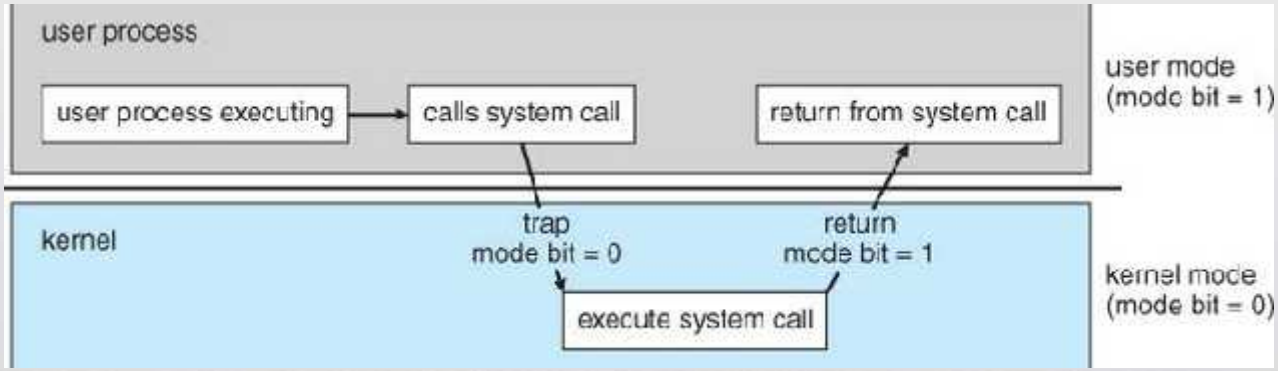
- **Multiprogramming** (**Batch system**) needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

Operating-System Operations

- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

Operating-System Operations

- Dual-mode operation allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code



Re do

Operating-System Operations

- Some instructions designated as privileged, only executable in kernel mode
- System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. virtual machine manager (VMM) mode for guest VMs

A system call is the way that a computer program requests a service from the kernel. This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling.

Operating System - Services

- Process Management
- Memory Management
- Storage Management
  - File system management
  - Storage space management
- Protection and Security

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion

Process Management

- The operating system is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, **logical view** of information storage
- Abstracts physical properties to logical storage unit - file
- Each medium is controlled by a device (i.e., disk drive, tape drive)
- Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

Storage Management

- File-System management
- Files usually organized into directories
- Access control on most systems to determine who can access what
- OS activities include
  - Creating and deleting files and directories
  - Primitives to manipulate files and directories
  - Mapping files onto secondary storage
  - Backup files onto stable (non-volatile) storage media

Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file

Computing Environments – Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Computing Environments – Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a "traditional" laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

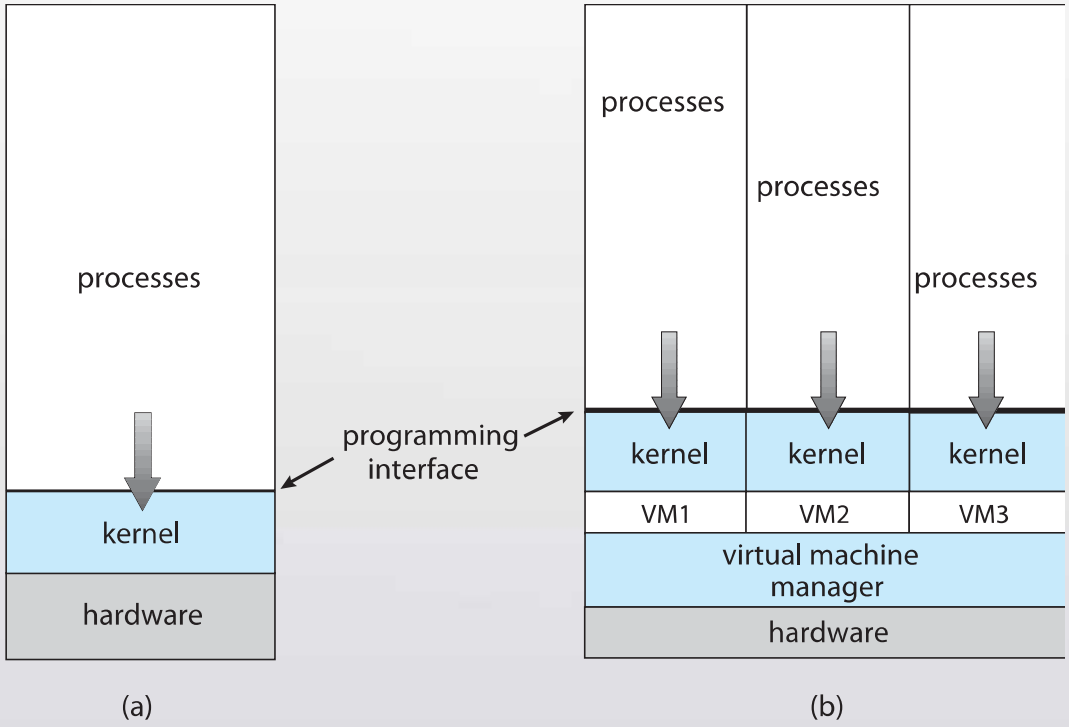
Computing Environments – Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Computing Environments – Virtualization

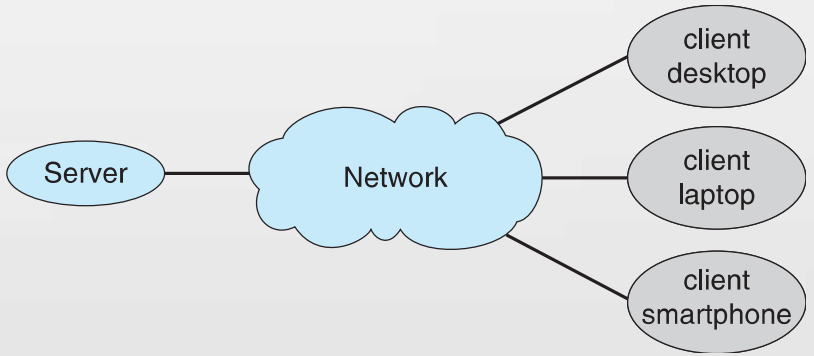
- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing computer environments within data centers

Computing Environments – Virtualization



Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now servers, responding to requests generated by clients
- Compute-server system provides an interface to client to request services (i.e., database)
- File-server system provides interface for clients to store and retrieve files



Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components

Computing Environments – Cloud Computing

- Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
- Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
- Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **Real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

Thank you

Introduction to Computer Systems

LECTURE 5.1: LOGIC CIRCUITS & SIMPLIFICATION

Objectives

After completing this module you will be able to:

- Understand what a logic circuit is.
- Understand how to simplify the digital logic circuits
- Understand how to simplify the circuits using Boolean Algebra
- Understand how to simplify the circuits using K-Map

Semiconductors to Computers

Increasing level of complexity:

- Transistors built from semiconductors
- Logic gates built from transistors
- Logic functions built from gates
- Flip-flops built from logic gates
- Counters and sequencers from flip-flops
- Microprocessors from sequencers
- Computers from microprocessors

Logic Gates

Logic Gates are the building blocks of Digital Circuits

Logic Gate is an electronic circuit having one or more than one input and only one output

The relationship between the input and the output is based on a certain logic

Inputs and Outputs can be represented as binary variables (logic variables, Boolean variables)

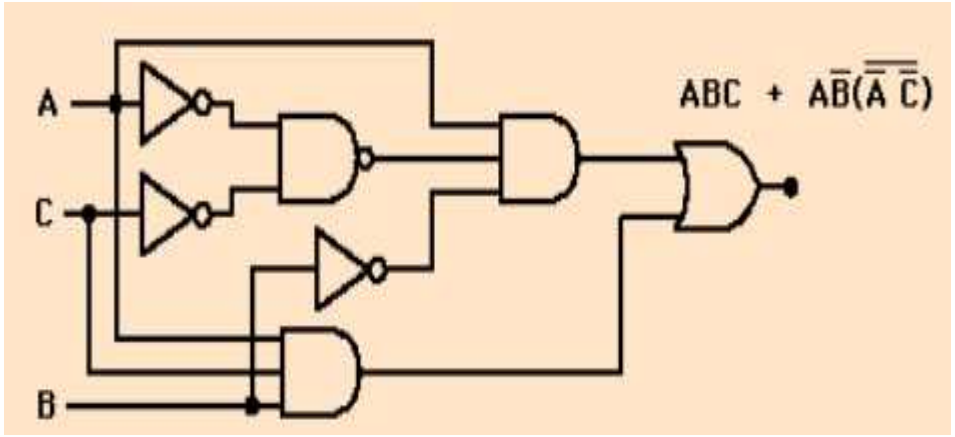
Logic Gates

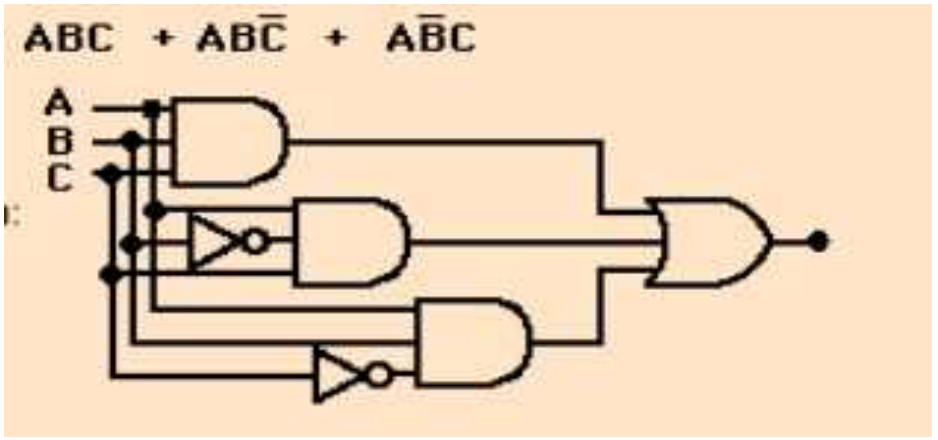
Used algebraic or tabular forms to describe the manipulation and processing of binary information.

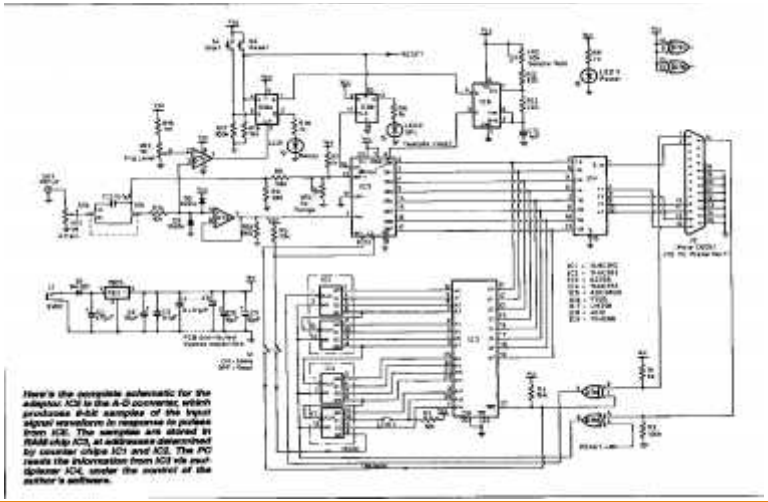
- Important advantages for two-valued Digital Circuit:
  - Mathematical Model – Boolean Algebra
  - Can help *design*, *analyse*, *simplify* Digital Circuits.

Digital Logic Circuits

Digital Logic Circuits

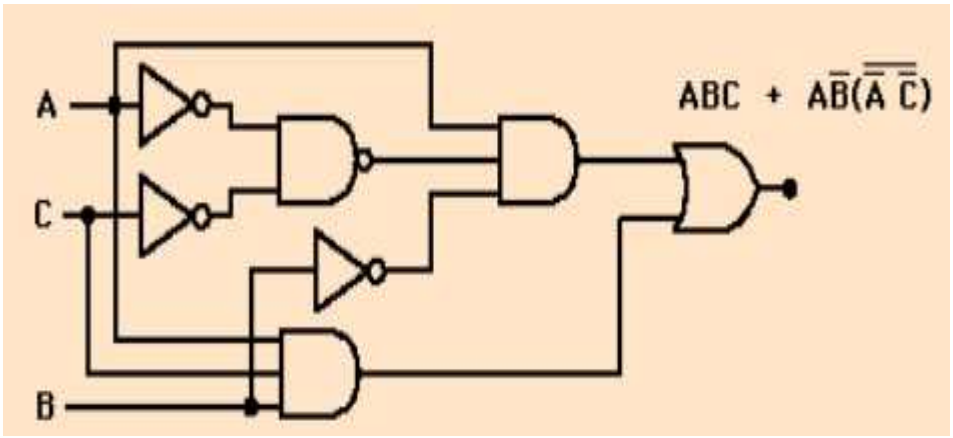


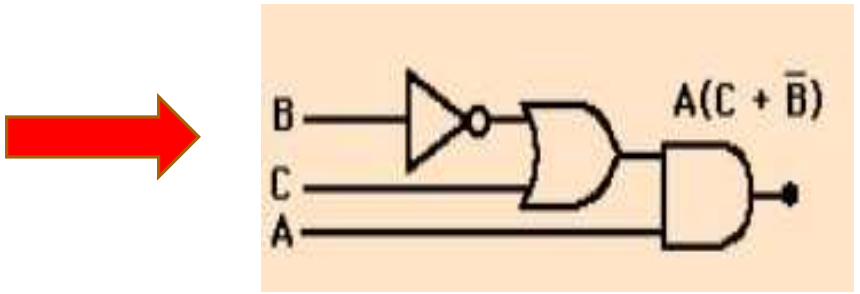


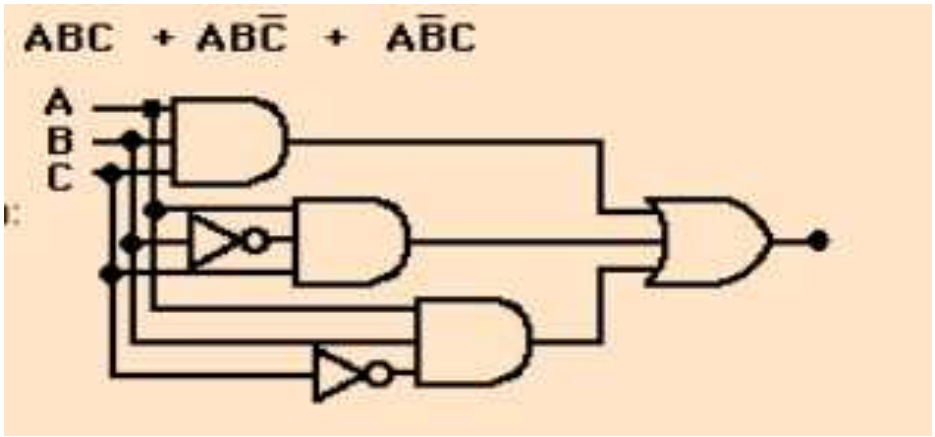


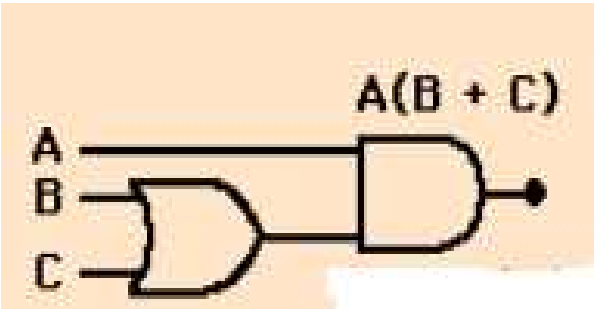
Simplification of the Digital Logic Circuits

Digital Logic Circuits









Simplification of the Digital Logic Circuits

Why do we need simplification?

- To reduce the number of gates required to build the circuit and hence:
  - Reduce the cost
  - Reduce the power consumption
  - Reduce the space required
  - Reduce the propagation delay

Simplification of the Digital Logic Circuits (Contd.)

Methods to Simplify the Digital Logic Circuits

- Using Truth Table
- Using Mathematical Method

Using Truth Table

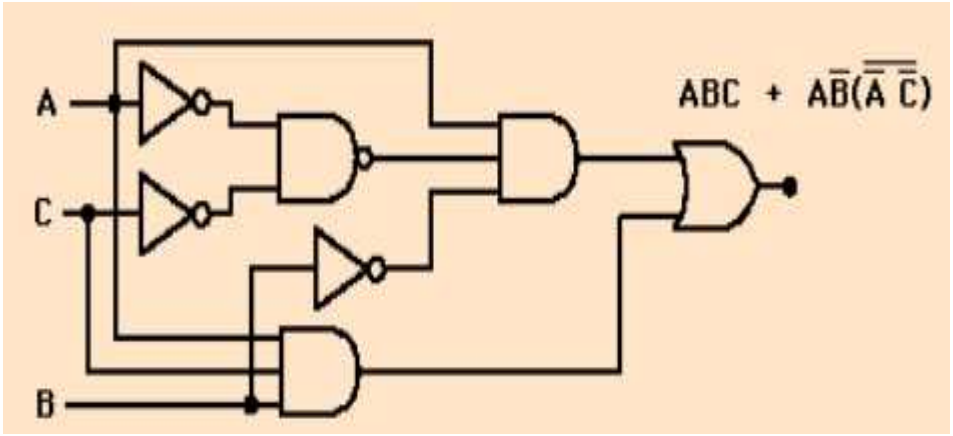
- The problem is analyzed
- The truth table is developed
- Using the truth table the circuit is built.

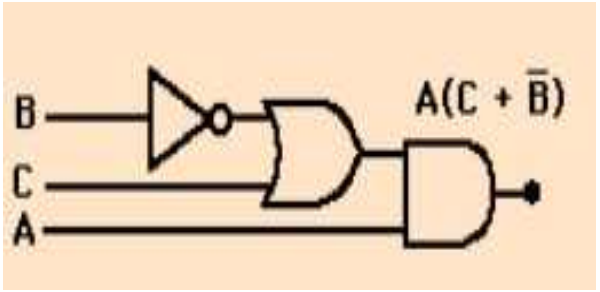
Simplification of Combinational Circuits Using Mathematical Methods

There are two methods to simplify the combinational Circuits

- Boolean Algebraic method
- K-map

Simplification of Combinational Circuits





$ABC + A\bar{B}(\bar{A} + \bar{C})$ DeMorgan's theorem

$ABC + A\bar{B}A + A\bar{B}C$ sum of products form

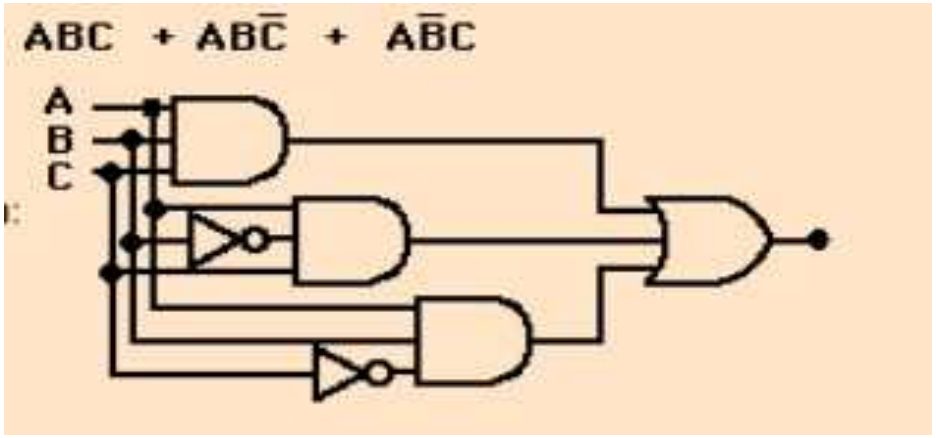
$ABC + A\bar{B} + A\bar{B}C$ BA=AB and AA = A

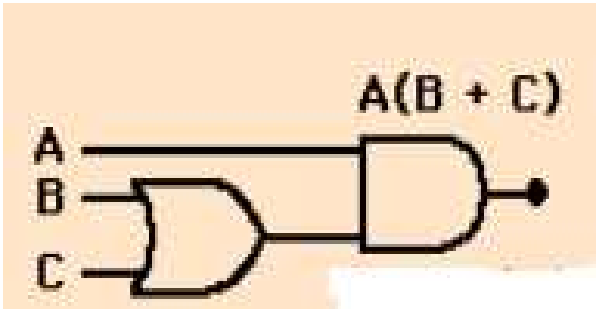
$AC(B + \bar{B}) + A\bar{B}$

$AC + A\bar{B}$ $B + \bar{B} = 1$

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

Simplification of Combinational Circuits





$ABC + AB\bar{C} + A\bar{B}C + ABC$

$AB(C + \bar{C}) + AC(\bar{B} + B)$

$AB + AC = A(B + C)$

$ABC = ABC + ABC$

$\bar{C} + C = 1$

K-Map to Simplify the Digital Logic Circuits

Introduction : K-Map

- The Karnaugh map was invented in 1952 by Edward W. Veitch.
- It was further developed in 1953 by Maurice Karnaugh, a physicist at Bell Labs, to help simplify digital electronic circuits.
- While exploring the new field of digital logic and its application to the design of telephone circuits, he invented a graphical way of visualizing and then simplifying Boolean expressions.
- This graphical representation, now known as a Karnaugh map, or Kmap, is named in his honor.

Description of K-maps and Terminology

A Kmap is a matrix consisting of rows and columns that represent the output values of a Boolean function.

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

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

Description of Kmaps and Terminology

- For example, the minterms for a function having the inputs *x* and *y* are:
- Consider the Boolean function,
- Its minterms are:

$$F(x,y) = XY + \bar{X}Y$$

Minterm	X	Y
$\bar{X}\bar{Y}$	0	0
$\bar{X}Y$	0	1
$X\bar{Y}$	1	0
XY	1	1

A two-variable function, such as f(x,y), has $2^2 = 4$ minterms:

Description of Kmaps and Terminology

- Similarly, a function having three inputs, has the minterms that are shown in this diagram.
- $2^3 = 8$ miniterms

Minterm	X	Y	Z
$\bar{X}\bar{Y}\bar{Z}$	0	0	0
$\bar{X}\bar{Y}Z$	0	0	1
$\bar{X}Y\bar{Z}$	0	1	0
$\bar{X}YZ$	0	1	1
$X\bar{Y}\bar{Z}$	1	0	0
$X\bar{Y}Z$	1	0	1
$XY\bar{Z}$	1	1	0
XYZ	1	1	1

Description of Kmaps and Terminology

- A Kmap has a cell for each minterm.
- This means that it has a cell for each line for the truth table of a function.
- The truth table for the function $F(x,y) = xy$ is shown at the right along with its corresponding Kmap.

F(X,Y) = XY

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X \ Y	0	1
0	0	0
1	0	1

Description of Kmaps and Terminology

- As another example, we give the truth table and KMap for the function, $F(x,y) = x + y$ at the right.
- This function is equivalent to the OR of all of the minterms that have a value of 1. Thus:

$$F(x,y) = X+Y = \bar{X}Y + X\bar{Y} + XY$$

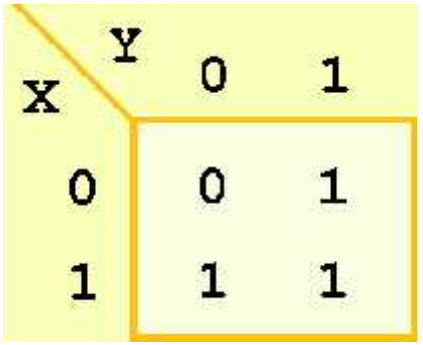
F(X,Y) = X+Y

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

X \ Y	0	1
0	0	1
1	1	1

Kmap Simplification for Two Variables

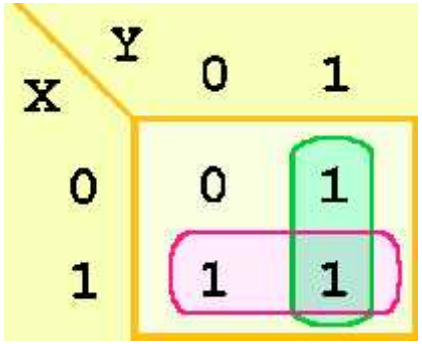
- Of course, the minterm function that we derived from our Kmap was not in simplest terms.
  - That's what we started with in this example.
- We can, however, reduce our complicated expression to its simplest terms by finding adjacent 1s in the Kmap that can be collected into groups that are powers of two.



- In our example, we have two such groups.
  - Can you find them?

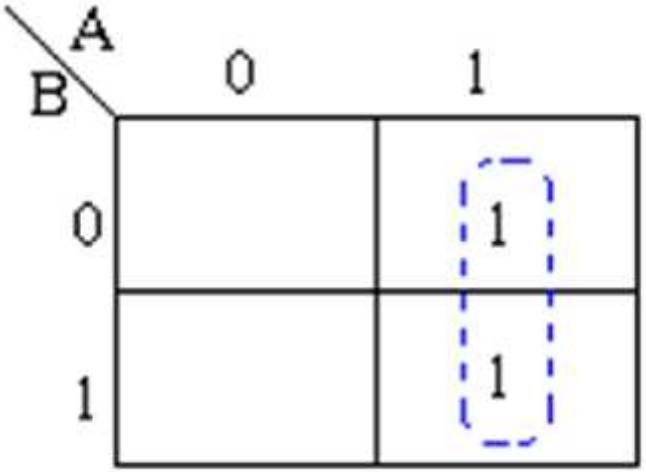
Kmap Simplification for Two Variables

- The best way of selecting two groups of 1s form our simple Kmap is shown below.
- We see that both groups are powers of two and that the groups overlap.



Example -1

Consider the following map.



The function plotted is:

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

Using algebraic simplification,

$Z = AB + A\overline{B}$

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

$\overline{A} + A = 1$

$Z = A$

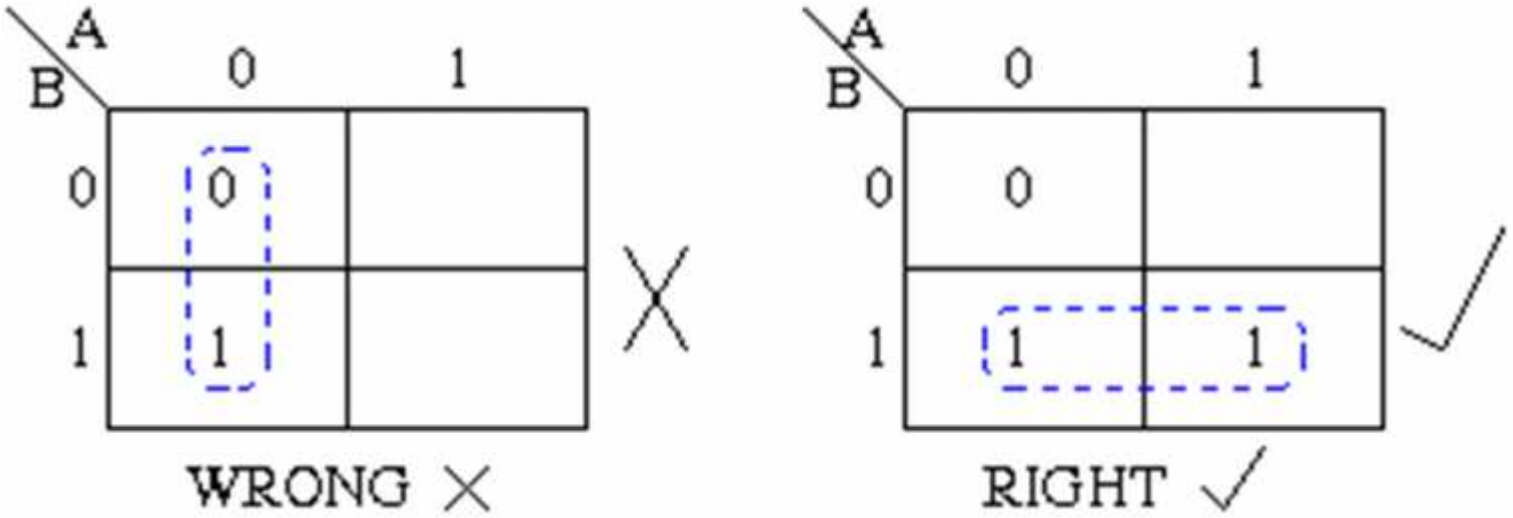
B becomes redundant

Referring to the map, the two adjacent 1's are grouped together. Through inspection it can be seen that variable B has its true and false form within the group. This eliminates variable B leaving only variable A which only has its true form.
The minimized answer therefore is Z = A.

Kmap Simplification for Two Variables

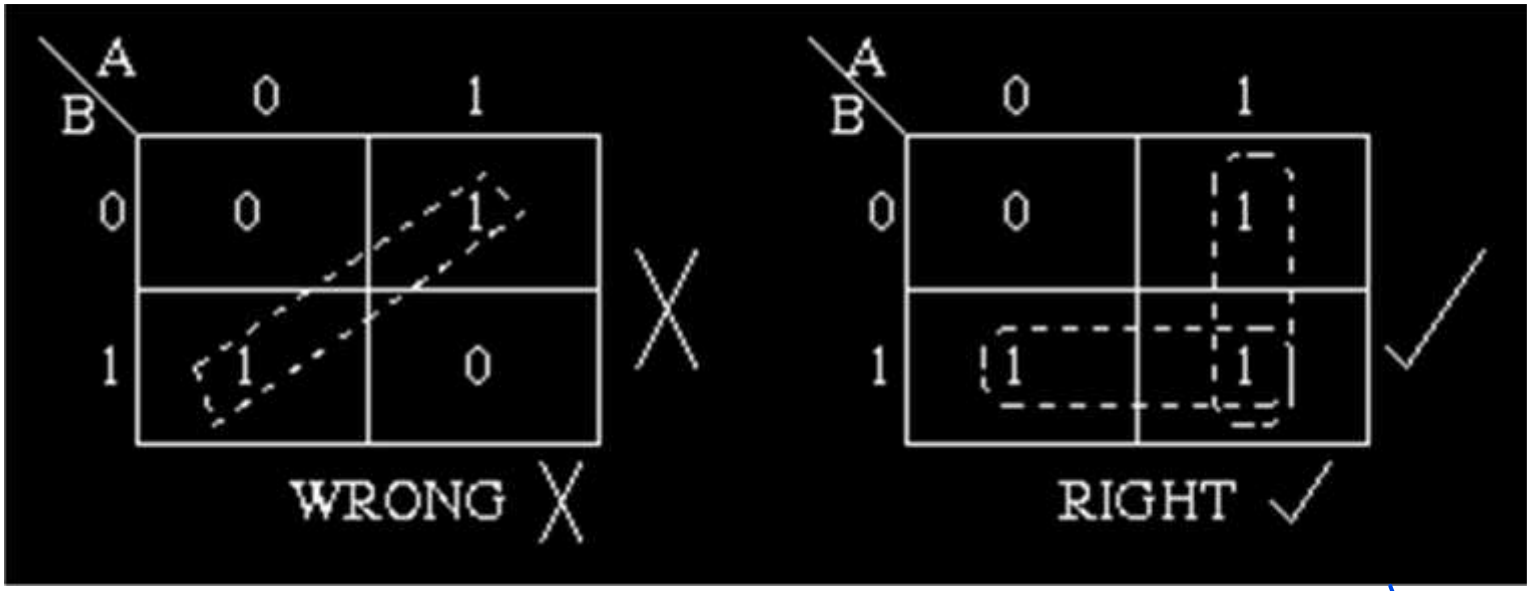
- The rules of K-map simplification are:
  - ✓ Groupings can contain only 1s; no 0s.
  - ✓ Groups can be formed only at right angles; diagonal groups are not allowed.
  - ✓ The number of 1s in a group must be a power of 2 – even if it contains a single 1.
  - ✓ The groups must be made as large as possible.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

Karnaugh Maps – Rules



Group only elements containing 1

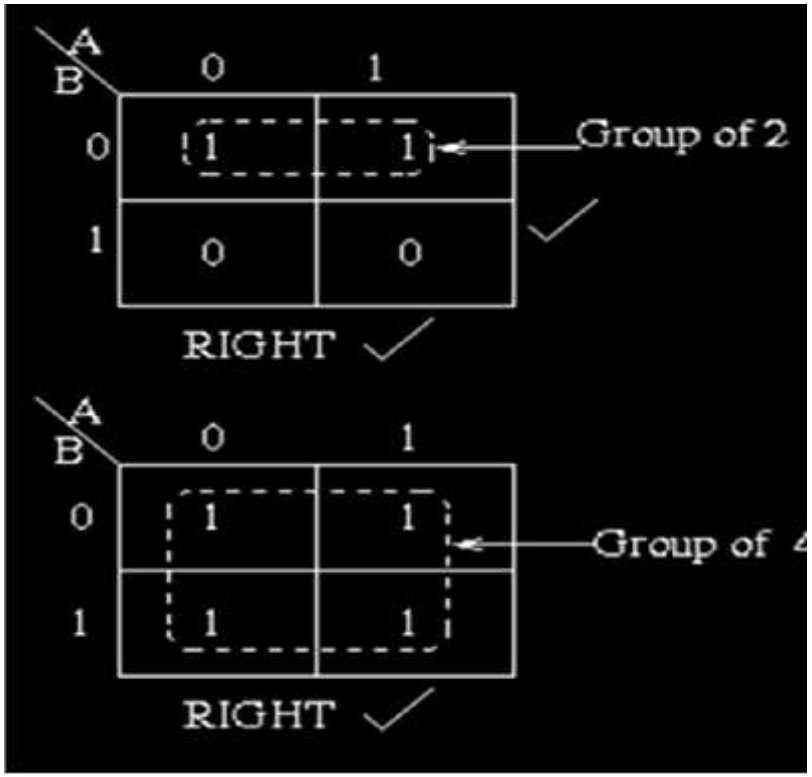
Karnaugh Maps – Rules



Only Horizontal and Vertical Grouping

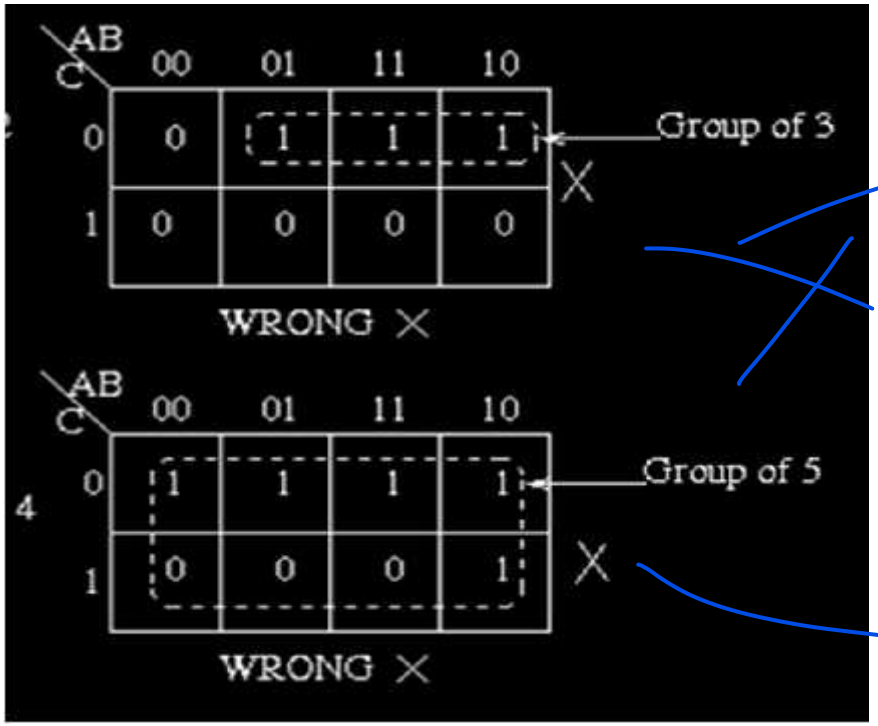
Diagonal not allowed

Karnaugh Maps – Rules



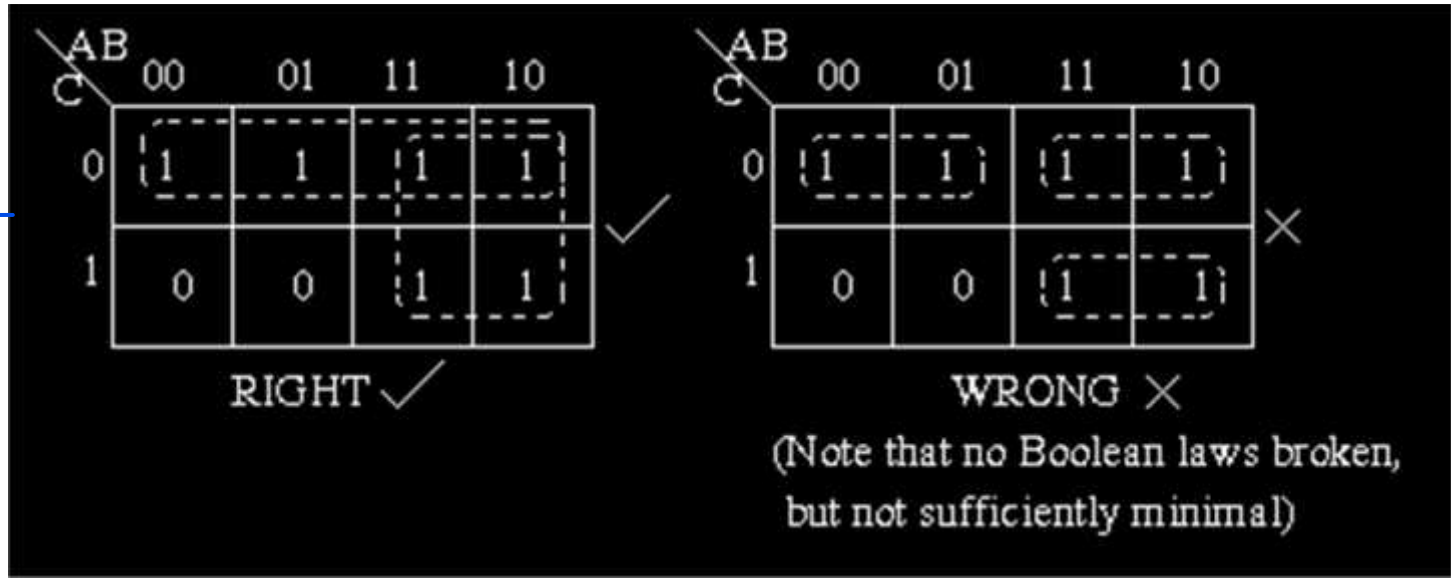
Groups Powers of 2

Karnaugh Maps – Rules



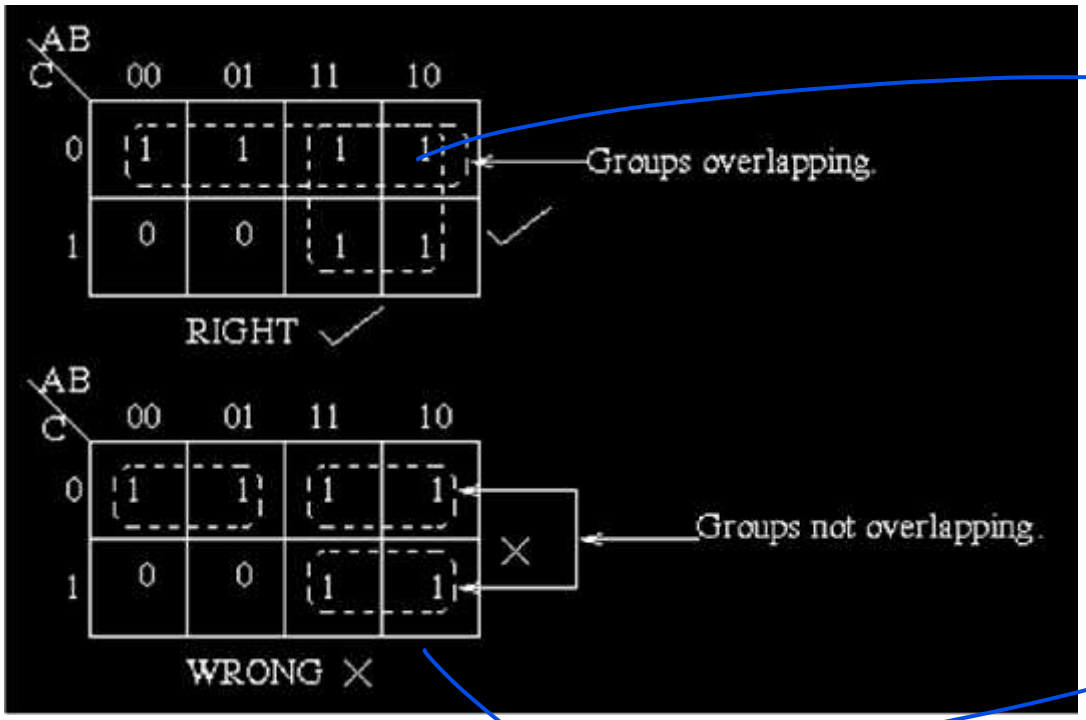
Groups Power of 2

Karnaugh Maps – Rules



Each group should be large as possible

Karnaugh Maps – Rules

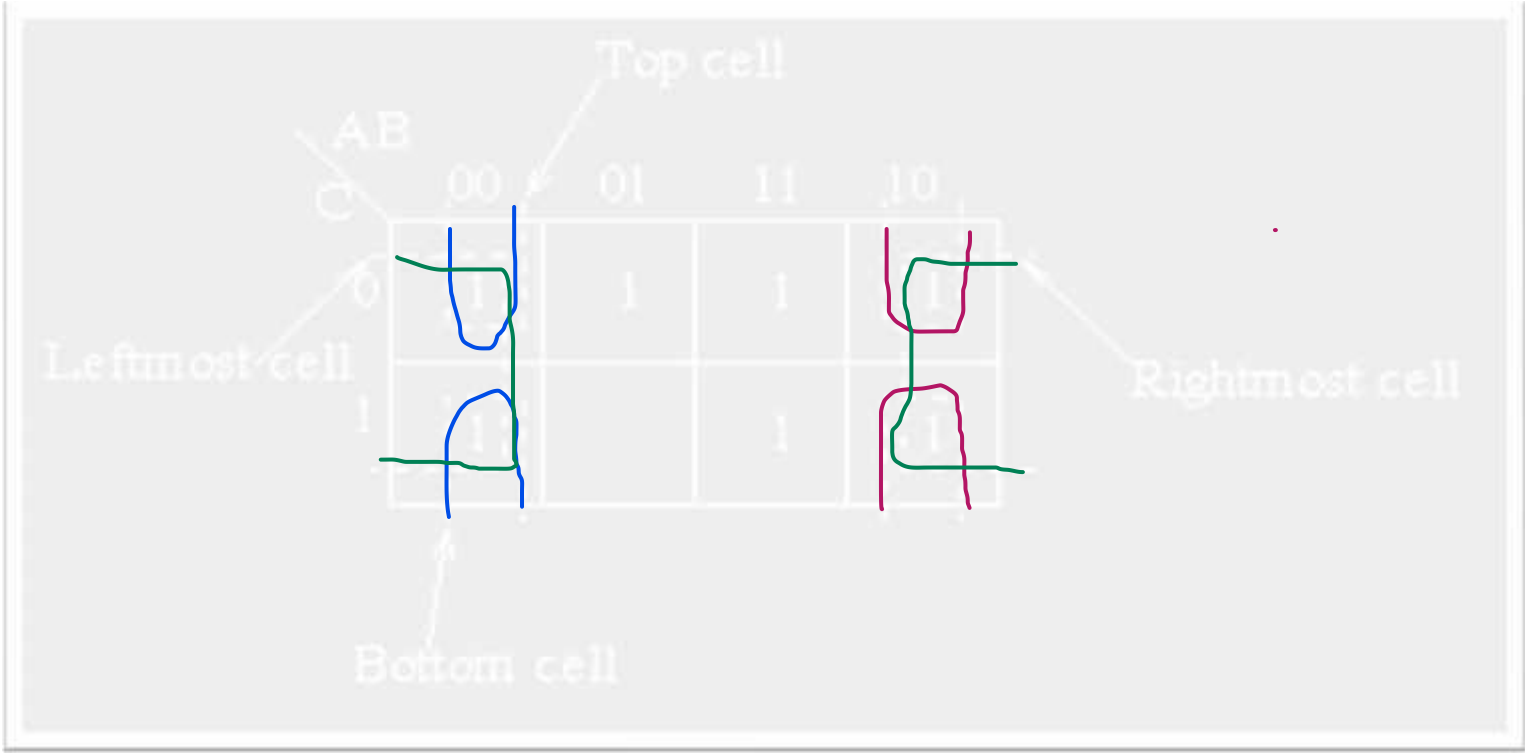


4

Groups may overlap

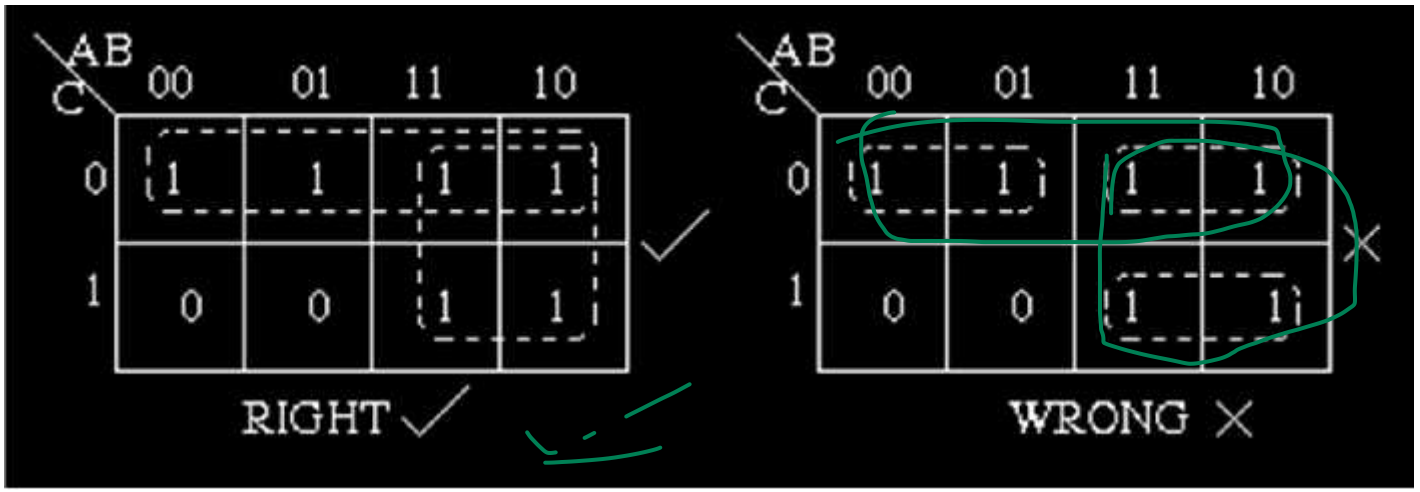
not ALL

Karnaugh Maps – Rules



Groups may wrap around the table

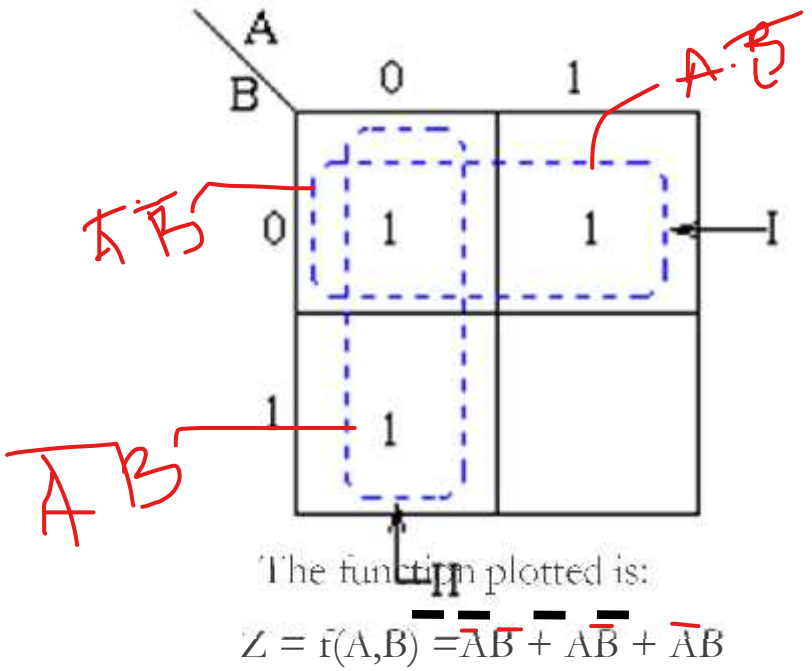
Karnaugh Maps – Rules



Should have few groups as possible

Example-2

Consider the following map.



The function plotted is:

$Z = f(A,B) = \bar{A}\bar{B} + A\bar{B} + \bar{A}B$

Pairs of 1's are *grouped* as shown above, and the simplified answer is obtained by using the following steps:

- Note that two groups can be formed for the example given above, bearing in mind that the largest rectangular clusters that can be made consist of two 1s.
- Notice that a 1 can belong to more than one group.
- The first group labelled I, consists of two 1s which correspond to A = 0, B = 0 and A = 1, B = 0.
- Put in another way, all squares in this example that correspond to the area of the map where B = 0 contains 1s, independent of the value of A.
- So when B = 0 the output is 1.
- The expression of the output will contain the term **B**

For group labeled II corresponds to the area of the map where $\bar{A} = 0$.

The group can therefore be defined as **A**

This implies that when A = 0 the output is 1.

The output is therefore 1 whenever $\bar{B} = 0$ and $\bar{A} = 0$

Hence the simplified answer is $Z = \bar{A} + \bar{B}$

Kmap Simplification for Three Variables

- A Kmap for three variables is constructed as shown in the diagram below.
- We have placed each minterm in the cell that will hold its value.
  - Notice that the values for the *yz* combination at the top of the matrix form a pattern that is not a normal binary sequence.

x \ yz	00	01	11	10
0	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
1	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

Kmap Simplification for Three Variables

- Thus, the first row of the Kmap contains all minterms where *x* has a value of zero.
- The first column contains all minterms where *y* and *z* both have a value of zero.

x \ YZ	00	01	11	10
0	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
1	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

Kmap Simplification for Three Variables

- Consider the function:

$$F(X,Y,Z) = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}Z + XYZ$$

- Its Kmap is given below.
  - What is the largest group of 1s that is a power of 2?

X \ YZ	00	01	11	10
0	0	1	1	0
1	0	1	1	0

Kmap Simplification for Three Variables

- This grouping tells us that changes in the variables *x* and *y* have no influence upon the value of the function: They are irrelevant.
- This means that the function,

$$F(X,Y,Z) = \bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}Z + XYZ$$

reduces to $F(x) = z$.

You could verify this reduction with identities or a truth table.

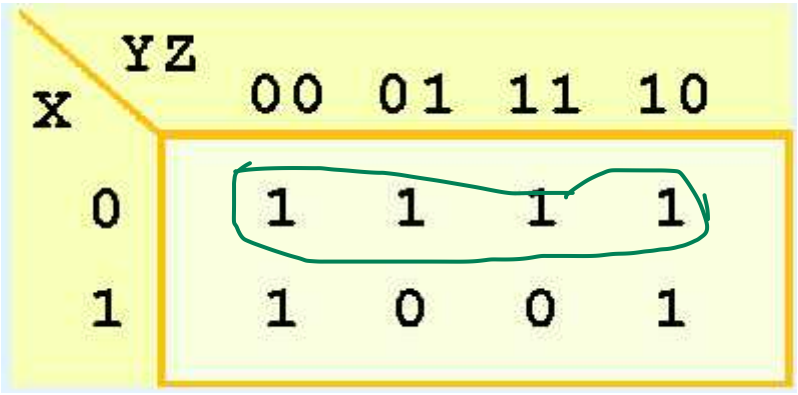
X \ YZ	00	01	11	10
0	0	1	1	0
1	0	1	1	0

Kmap Simplification for Three Variables

- Now for a more complicated Kmap. Consider the function:

$$F(X,Y,Z) = \bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + \bar{X}YZ + X\bar{Y}\bar{Z} + XY\bar{Z}$$

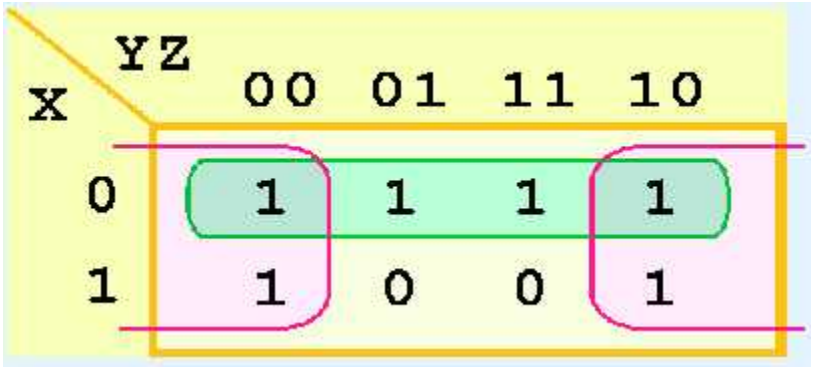
- Its Kmap is shown below. There are (only) two groupings of 1s.
  - ✓ Can you find them?



Kmap Simplification for Three Variables

- In this Kmap, we see an example of a group that wraps around the sides of a Kmap.
- This group tells us that the values of x and y are not relevant to the term of the function that is encompassed by the group.
  - What does this tell us about this term of the function?

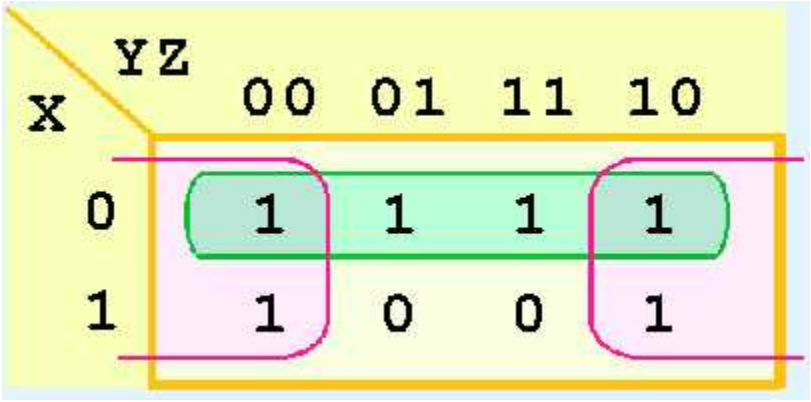
What about the green group in the top row?



Kmap Simplification for Three Variables

- The green group in the top row tells us that only the value of x is significant in that group.
- We see that it is complemented in that row, so the other term of the reduced function is .
- Our reduced function is: $F(x,y,z) = \overline{X} + \overline{Z}$

- **Recall that we had six minterms in our original function!**



Kmap Simplification for Four Variables

- Our model can be extended to accommodate the 16 minterms that are produced by a four-input function.
- This is the format for a 16-minterm Kmap.

WX \ YZ	00	01	11	10
00	$\bar{W}\bar{X}\bar{Y}\bar{Z}$	$\bar{W}\bar{X}\bar{Y}Z$	$\bar{W}\bar{X}YZ$	$\bar{W}\bar{X}Y\bar{Z}$
01	$\bar{W}X\bar{Y}\bar{Z}$	$\bar{W}X\bar{Y}Z$	$\bar{W}XYZ$	$\bar{W}XY\bar{Z}$
11	$WX\bar{Y}\bar{Z}$	$WX\bar{Y}Z$	$WXYZ$	$WXY\bar{Z}$
10	$W\bar{X}\bar{Y}\bar{Z}$	$W\bar{X}\bar{Y}Z$	$W\bar{X}YZ$	$W\bar{X}Y\bar{Z}$

Kmap Simplification for Four Variables

- We have populated the Kmap shown below with the nonzero minterms from the function:

$$F(W,X,Y,Z) = \bar{W}\bar{X}\bar{Y}\bar{Z} + \bar{W}\bar{X}\bar{Y}Z + \bar{W}\bar{X}Y\bar{Z} + \bar{W}X\bar{Y}\bar{Z} + W\bar{X}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}Z + W\bar{X}Y\bar{Z}$$

  - Can you identify (only) three groups in this Kmap?

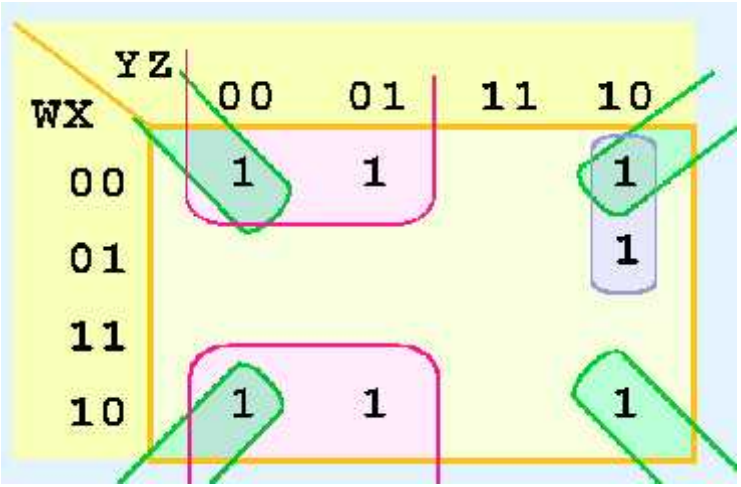
Recall that groups can overlap.

WX \ YZ	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

Kmap Simplification for Four Variables

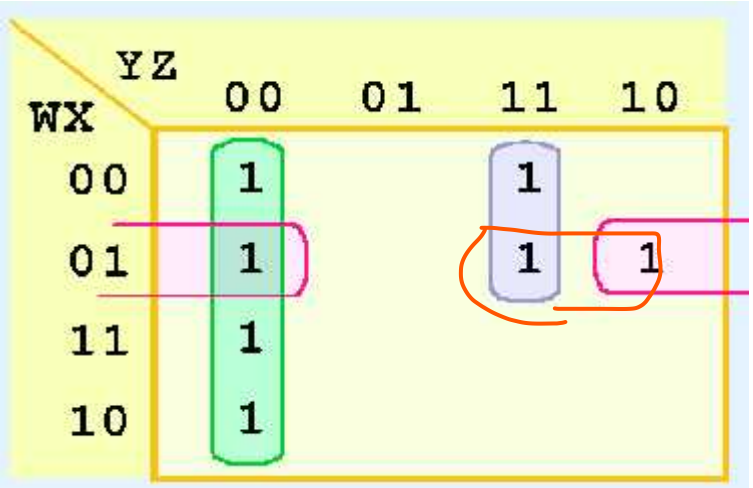
- Our three groups consist of:
  - ✓ A purple group entirely within the Kmap at the right.
  - ✓ A pink group that wraps the top and bottom.
  - ✓ A green group that spans the corners.
- Thus we have three terms in our final function:

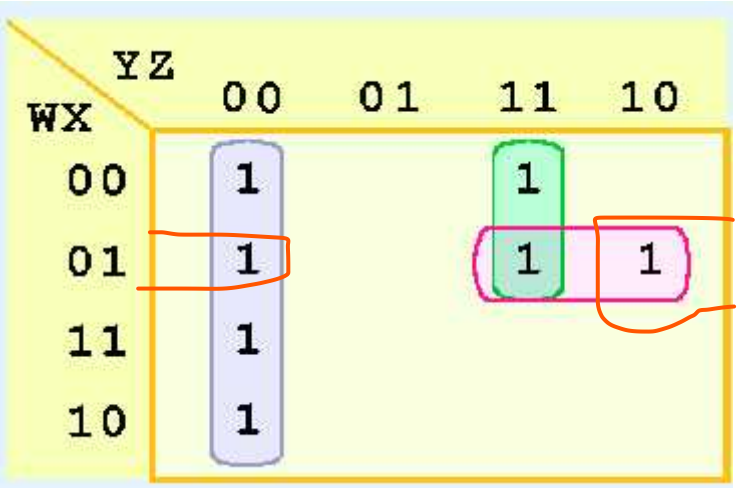
$$F(W,X,Y,Z) = \bar{X}\bar{Y} + \bar{X}\bar{Z} + \bar{W}Y\bar{Z}$$



Kmap Simplification for Four Variables

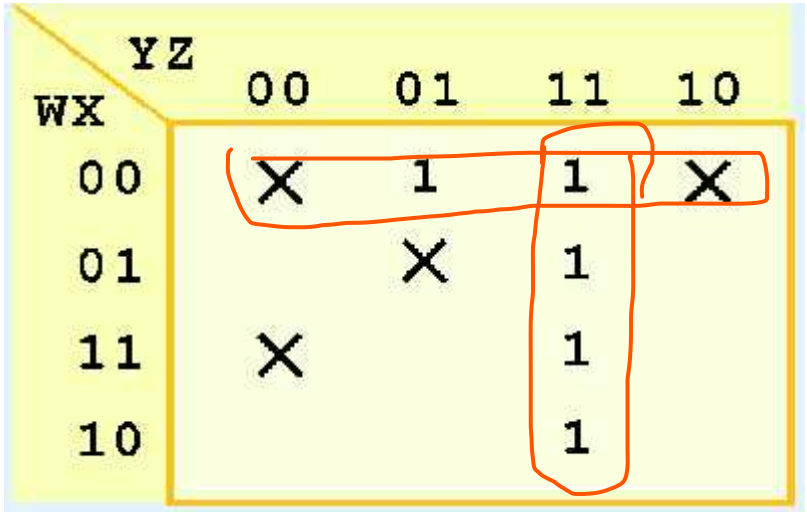
- It is possible to have a choice as to how to pick groups within a Kmap, while keeping the groups as large as possible.
- The (different) functions that result from the groupings below are logically equivalent.





Don't Care Conditions

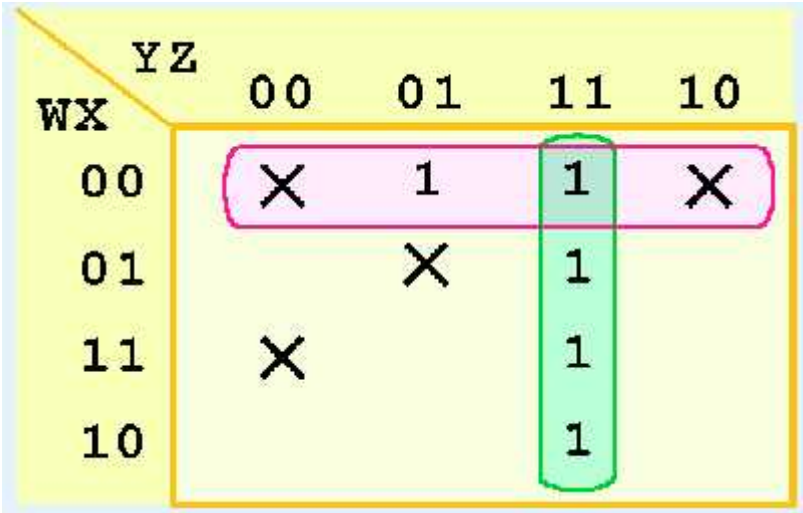
- In a Kmap, a don't care condition is identified by an *X* in the cell of the minterm(s) for the don't care inputs, as shown below.
- In performing the simplification, we are free to include or ignore the *X*'s when creating our groups.



Don't Care Conditions

- In one grouping in the Kmap below, we have the function:

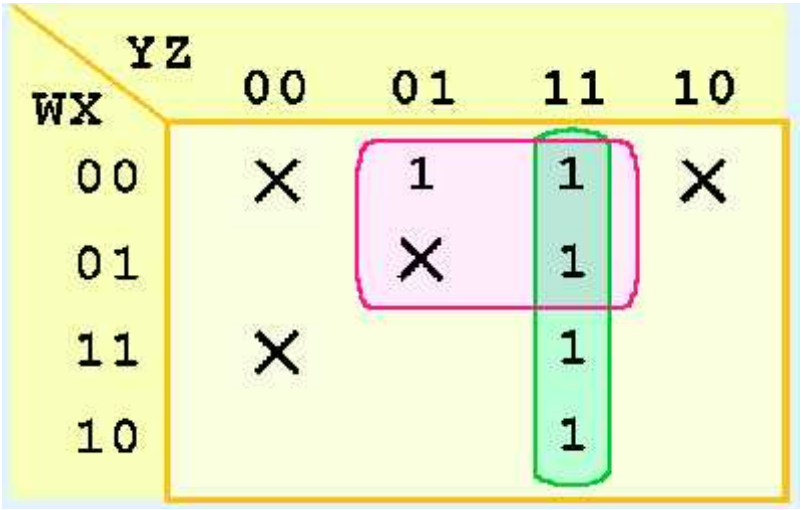
$$F(W,X,Y,Z) = \bar{W}\bar{X} + YZ$$



Don't Care Conditions

- A different grouping gives us the function:

$$F(W,X,Y,Z) = \bar{W}Z + YZ$$



Don't Care Conditions

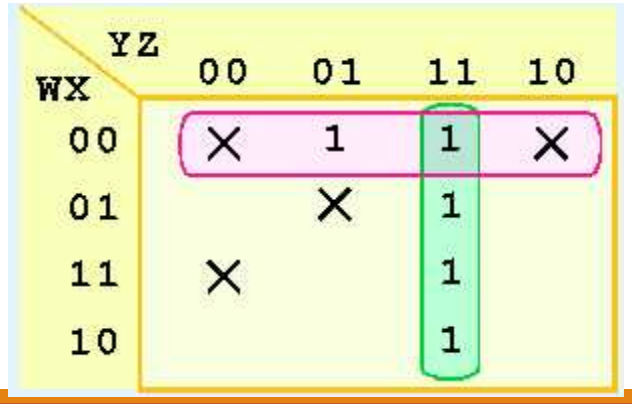
- The truth table of:

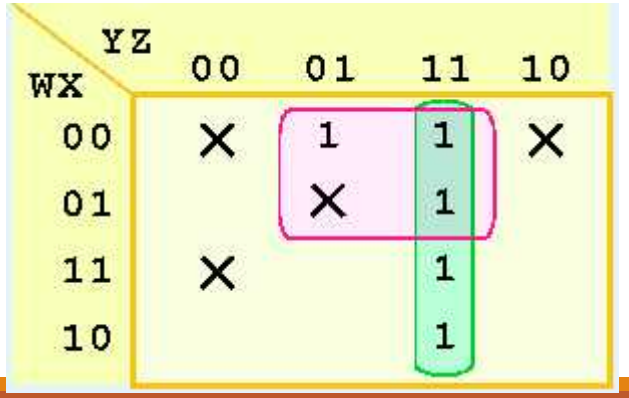
$$F(W,X,Y,Z) = \bar{W}\bar{X} + YZ$$

- is different from the truth table of:

$$F(W,X,Y,Z) = \bar{W}Z + YZ$$

- However, the values for which they differ, are the inputs for which we have don't care conditions.





Summery

Recapping the rules of Kmap simplification:

- Groupings can contain only 1s; no 0s.
- Groups can be formed only at right angles; diagonal groups are not allowed.
- The number of 1s in a group must be a power of 2 – even if it contains a single 1.
- The groups must be made as large as possible.
- Groups can overlap and wrap around the sides of the Kmap.
- Use don't care conditions when you can.

Thank You

End of Lecture 5.1

Lecture 5.2 : Combinational and Sequential Circuits

Introduction to Computer Systems

LECTURE 6: COMBINATIONAL AND SEQUENTIAL LOGIC CIRCUITS

Objectives

After completing this lesion you will be able to:

- Understand different types of Digital Logic Circuits.
- Understand the characteristics of the DLC.
- Understand the usage of each type.

Digital Logic Circuits

Digital Logic Circuits

Logic circuits can be categorized as

- Combinational Circuits
- Sequential circuits.

A combinational circuit consists of input variables, logic gates and output variables.

- Output depends only on current input
- Has no memory

Combinational Circuits

A combinational circuit consists of input variables, logic gates, and output variables.

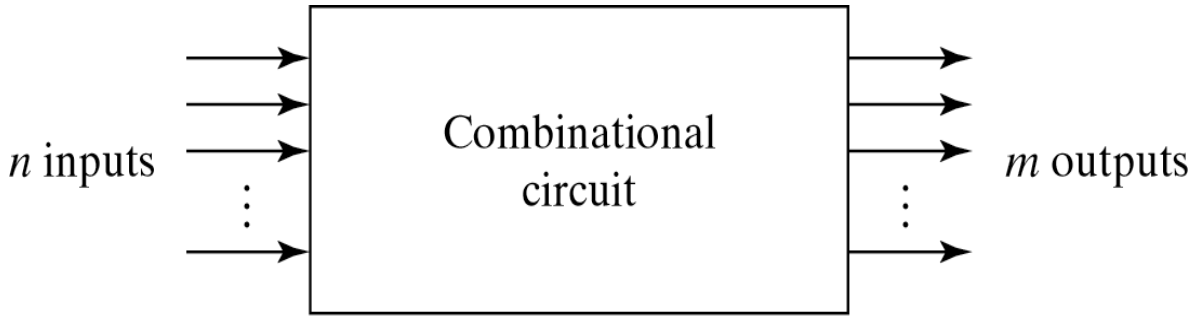
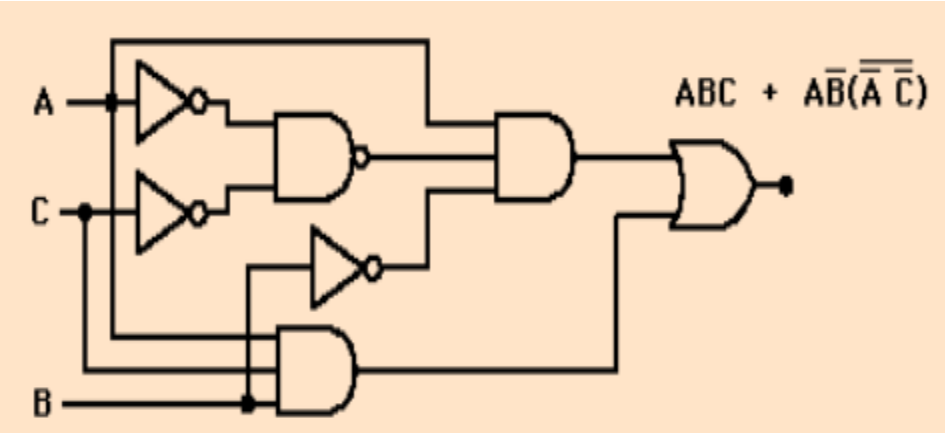
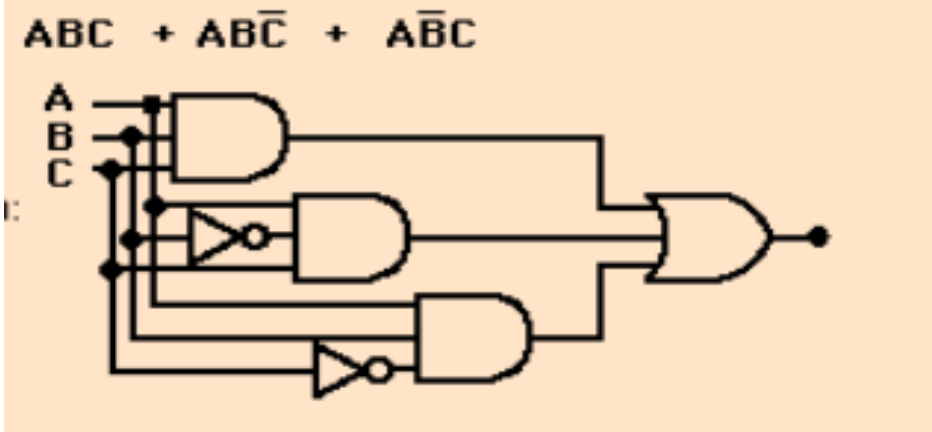


Fig. 4-1 Block Diagram of Combinational Circuit

Combinational Circuits (Examples)



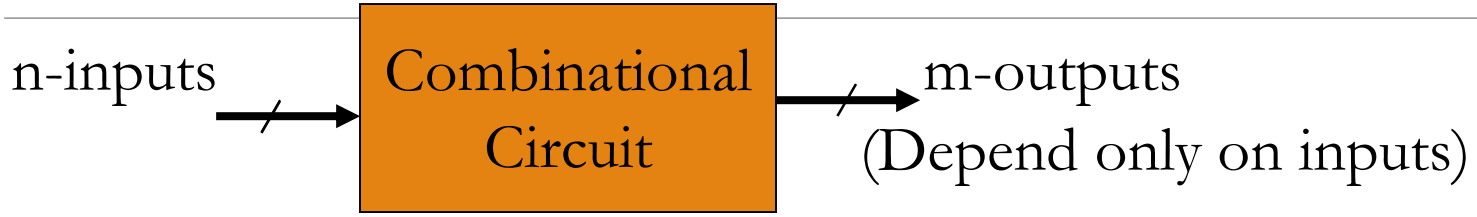


Sequential Logic

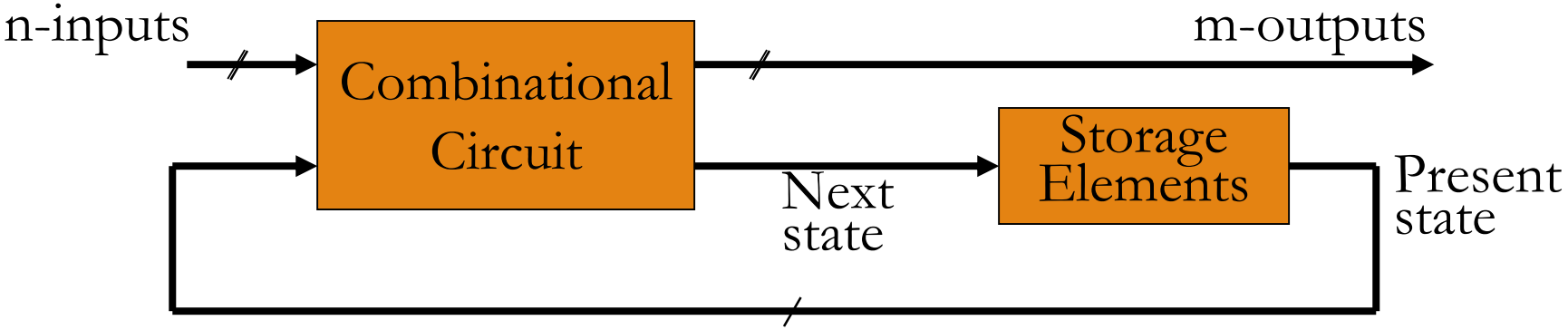
Sequential Logic:

- Output depends not only on current input but also on past input values, e.g., design a counter
- Need some type of memory to remember the past input values

Combinational vs. Sequential Circuits



Combinational Circuit



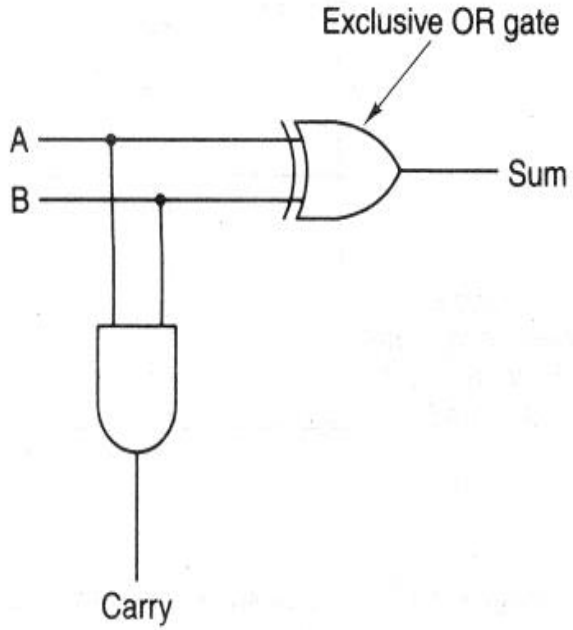
Sequential Circuit

Combinational Circuits

Combinational Circuits
Half Adder

Half Adder : The sum is **XOR** operation and the carry an **AND**

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Sum= A'B+AB'

$A \oplus B$

Carry=AB

Figure 4-23 (a) Truth table for 1 bit addition. (b) A circuit for a half adder

Combinational Circuits – Full Adder

5. Draw a diagram

Input			Output	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

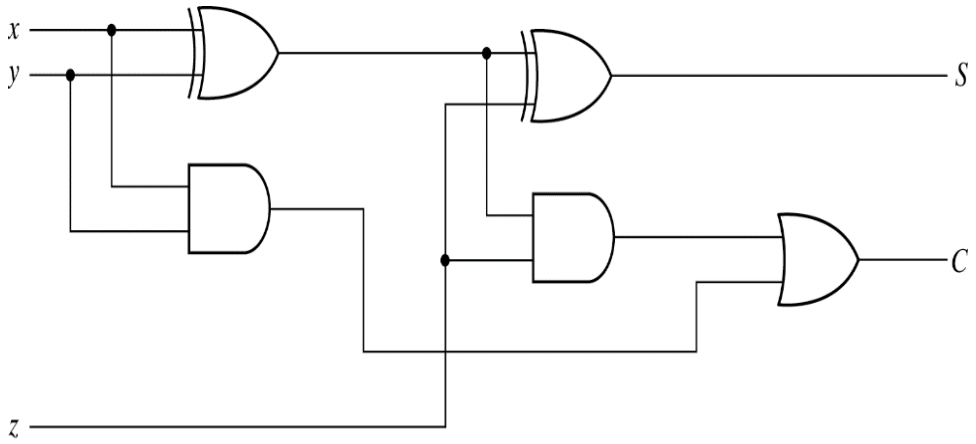
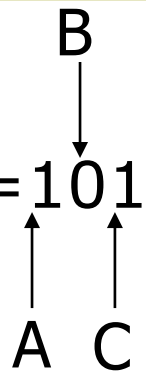


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

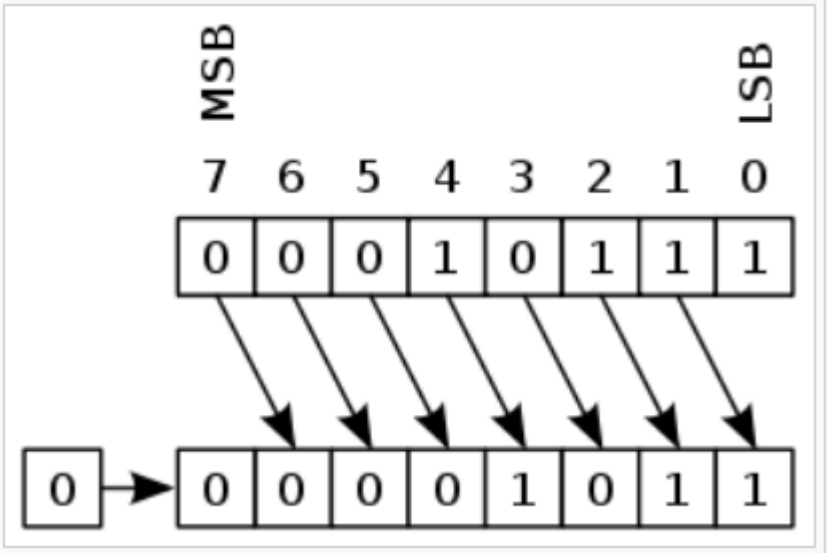
Binary Multiplication

- E.g: 15 * 5 = 75, 15=1111,5=101<-multiplier

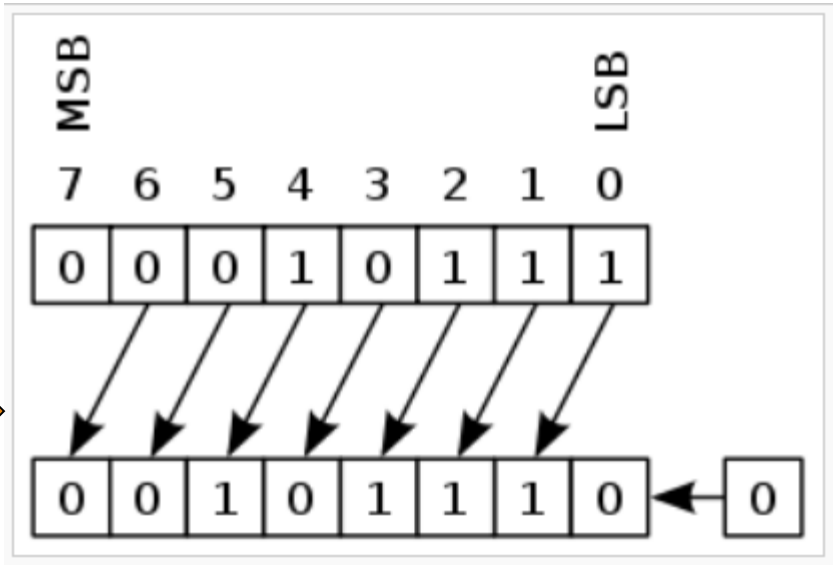


0000 0000	initialize to 0
0000 1111	A =1
0000 1111	sum
0001 1110	shift ← (left)
0000 0000	B = 0
0001 1110	sum
0011 1100	shift ← (left)
0000 1111	C = 1
0100 1011	sum, no shift

Shifter



Right shift 1-bit



Left shift 1-bit

Shifter

Figure shows an eight-input, eight-output, 1-bit left/right shifter.

- An eight-bit input is applied onto D_0, D1…D7, and the shifted output is taken out of S_0, S1…S7.
- The shift direction is controlled with the control signal C.

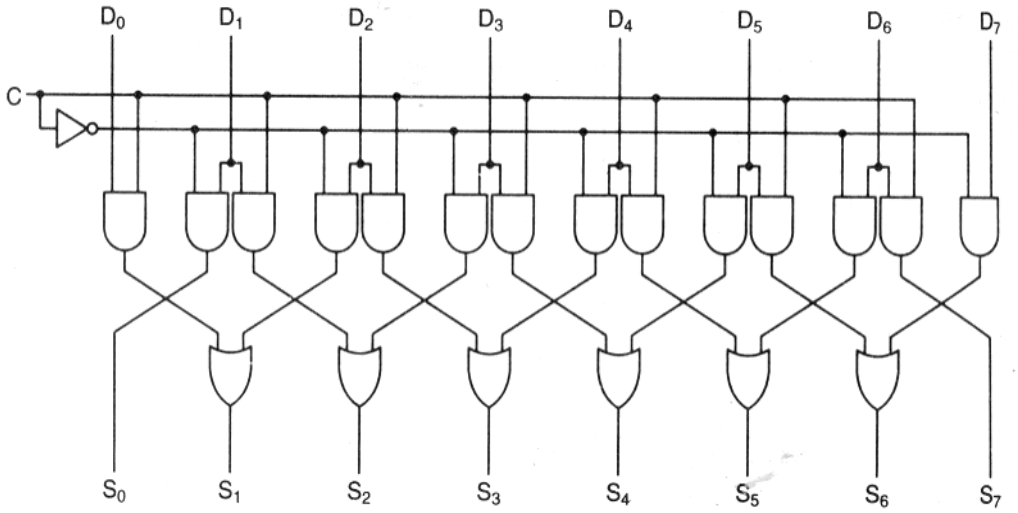


Figure 4-22. A 1-bit left/right shifter.

- A logic one on C enables the right hand side AND gates and pass each input bit to the OR gate to its right and hence, a one-bit right shift is performed.
- Similarly, a logic zero on C performs a left shift.

Comparator

A comparator circuit compares two input words.

The simplest comparator circuit is an exclusive NOR gate, which compares two individual bits:

- produces a zero if the two bits are unequal
- produces a logic one if they are equal

Comparator

When two words of length *n* are to be compared,

- *n* XOR gates and
- a NOR gate can be combined

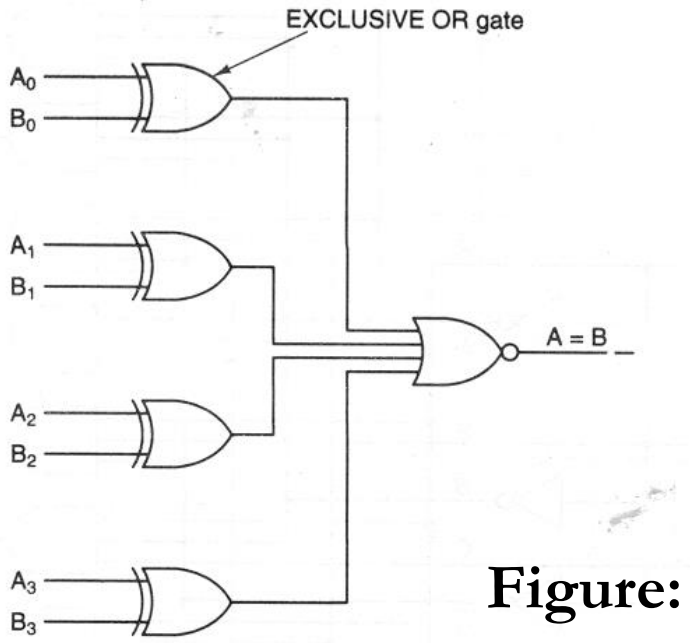


Figure: A simple 4-bit comparator

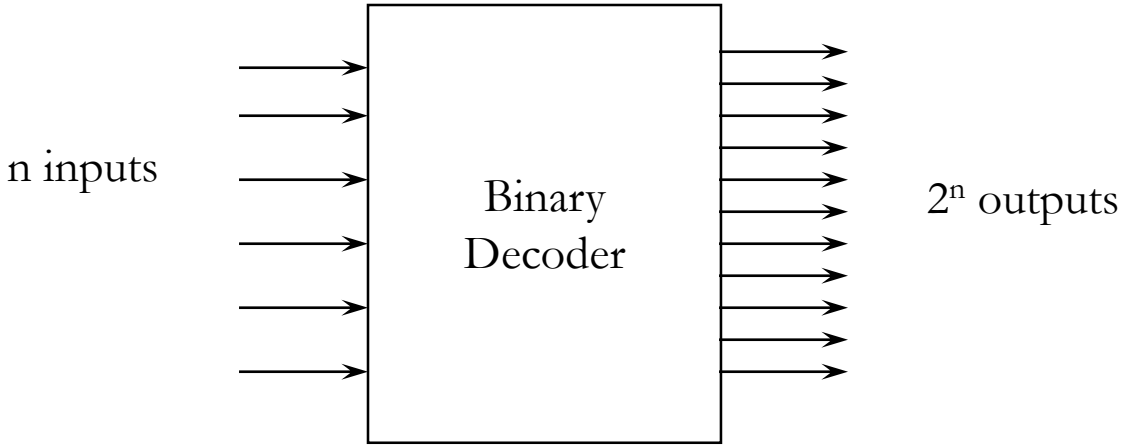
Decoder/Encoder

Decoder

- A combinational circuit that converts binary information from the ***n*** coded inputs to a maximum of 2^n unique outputs
- ***n***-to-***m*** line decoder = ***n x m*** decoder Ex: Octal to Binary line decoder
  - ***n*** inputs, ***m*** outputs (3 inputs, 8 outputs)
- If the n-bit coded information has unused bit combinations, the decoder may have less than 2^n outputs
  - $m \leq 2^n$

Binary Decoder

- Black box with n input lines and 2^n output lines

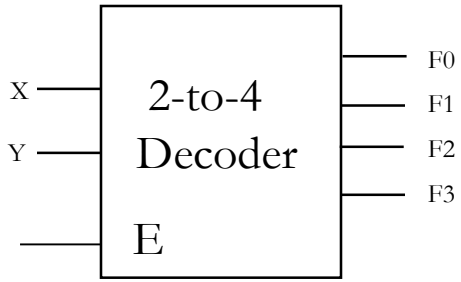


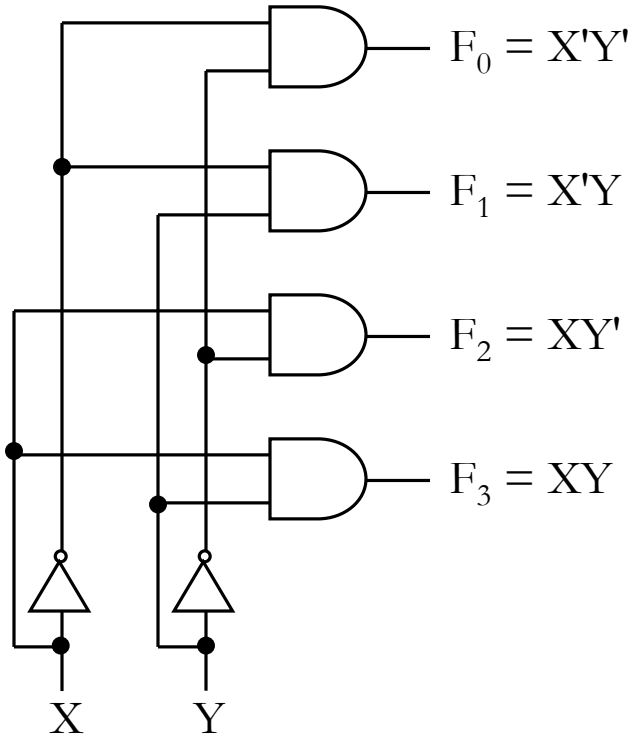
2-to-4 Binary Decoders

Truth Table:

X	Y	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

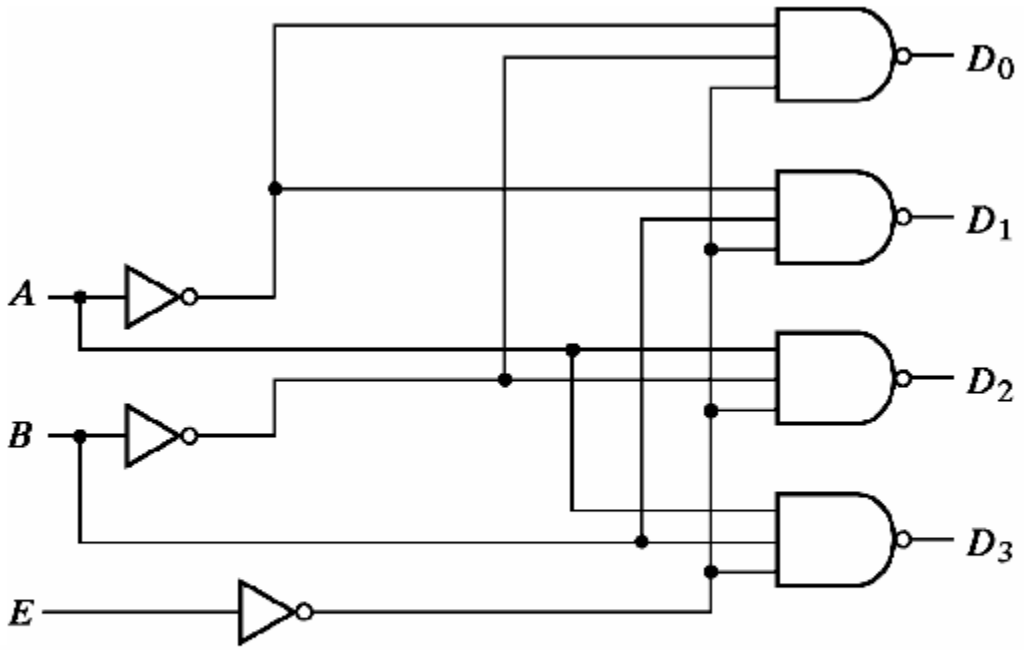
- From truth table, circuit for 2x4 decoder is:
- Note: Each output is a 2-variable minterm (X'Y', X'Y, XY' or XY)





2-to-4 Decoders : NAND Implementation

Decoder is enabled when E=0 and an output is active if it is 0



(a) Logic diagram

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

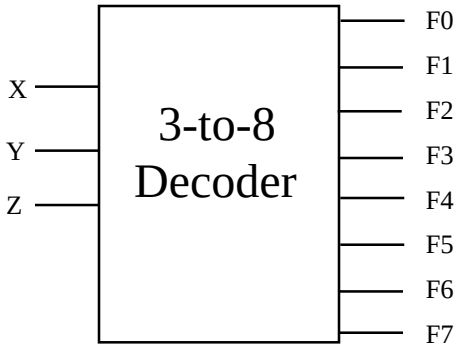
(b) Truth table

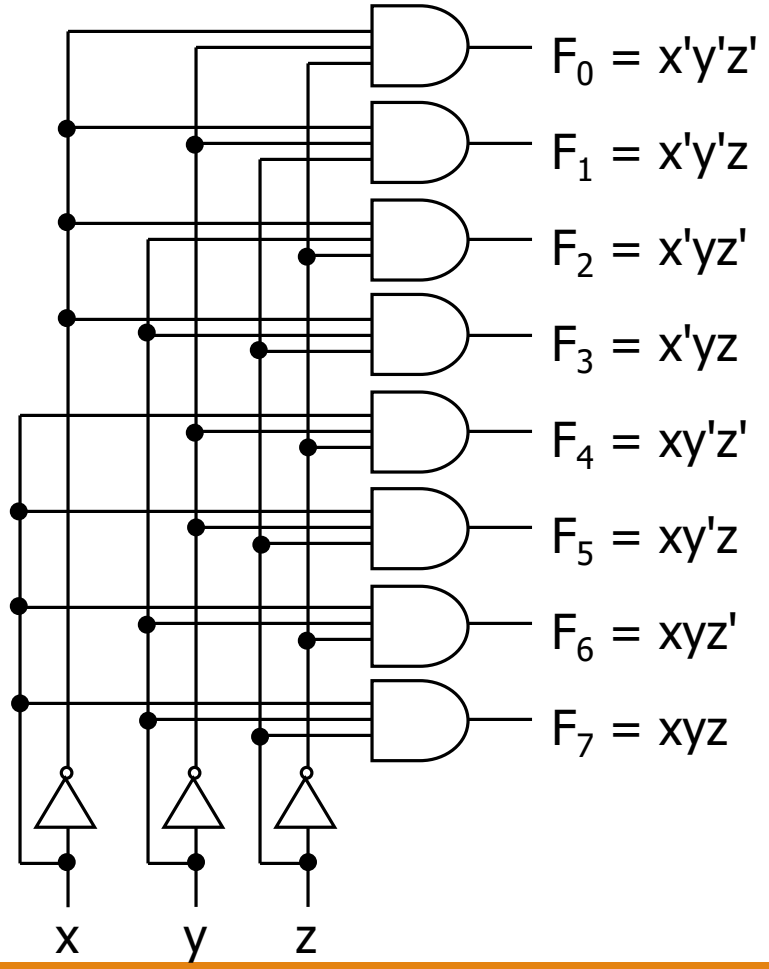
2-to-4-Line Decoder with Enable Input

3-to-8 Binary Decoder

Truth Table:

x	y	z	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1





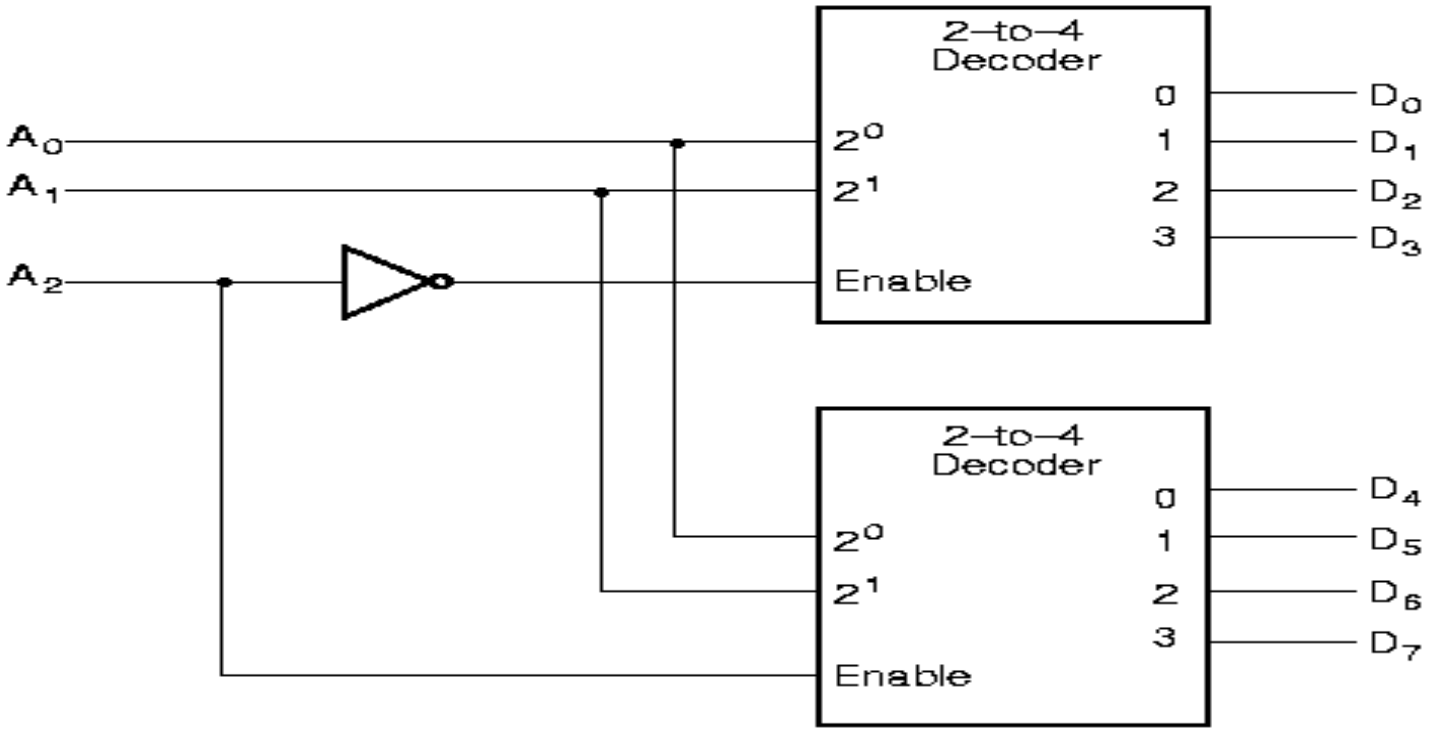
Decoder Expansion

Decoder expansion

- Combine two or more small decoders with enable inputs to form a larger decoder

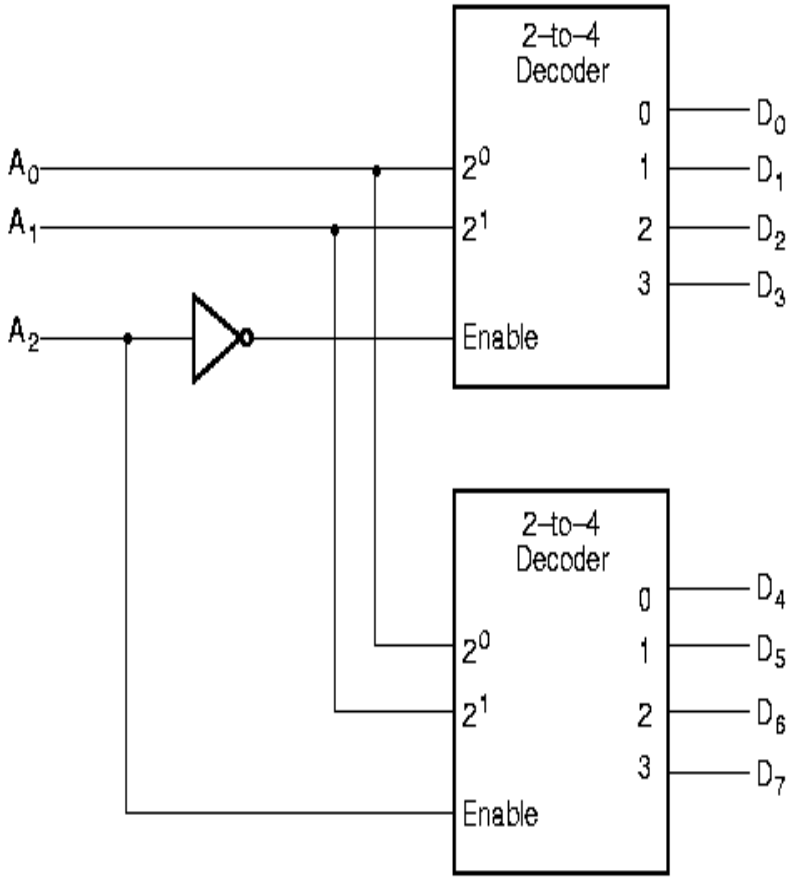
- 3-to-8-line decoder constructed from two 2-to-4-line decoders
  - **The MSB is connected to the enable inputs**
  - **if A_2=0, upper is enabled; if A_2=1, lower is enabled.**

Decoder Expansion(Contd.)



Combining two 2-4 decoders to form one 3-8 decoder using enable switch

A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



The highest bit is used for the enables

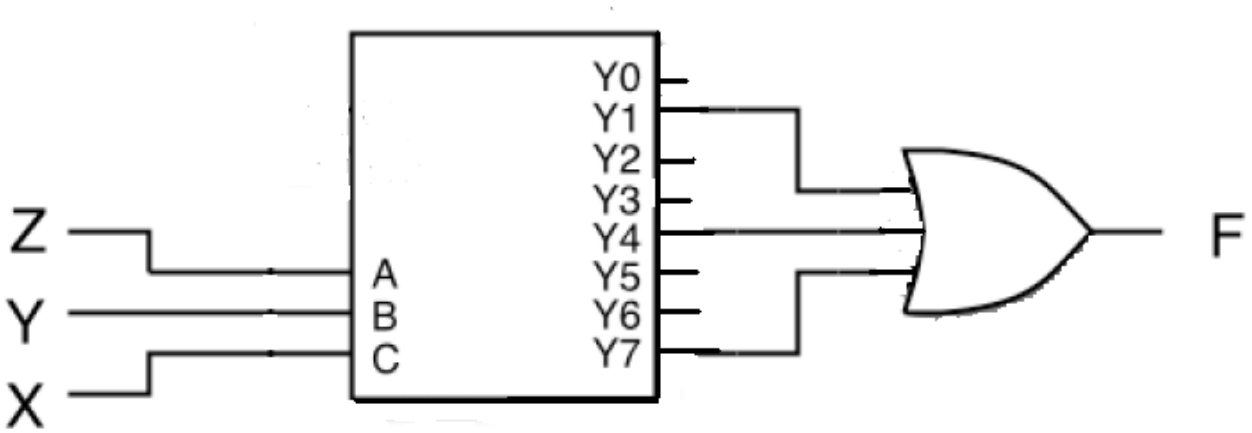
Combinational Circuit Design with Decoders

Combinational circuit implementation with decoders

- A decoder provide 2^n minterms of n input variables
- Since any Boolean function can be expressed as a sum of minterms, one can use a decoder and external OR gates to implement any combinational function.

Combinational Circuit Design with Decoders

Example Realize F (X,Y,Z) = Σ (1, 4, 7) with a decoder:



Encoder

Inverse Operation of a decoder

2^n input, n output

Truth Table

3 OR Gates Implementation

- A0 = D1 + D3 + D5 + D7
- A1 = D2 + D3 + D6 + D7
- A2 = D4 + D5 + D6 + D7

Inputs								Outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Truth Table for Encoder

Multiplexer (MUX)

- A combinational circuit that receives binary information from one of 2^n input data lines and directs it to a single output line
- A 2^n -to 1 multiplexer has *2^n input data lines* and *n input selection lines*
- 4-to-1 multiplexer Diagram
- 4-to-1 multiplexer Function Table

Function Table for 4-to-1 line Multiplexer

Select		Output
S1	S0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

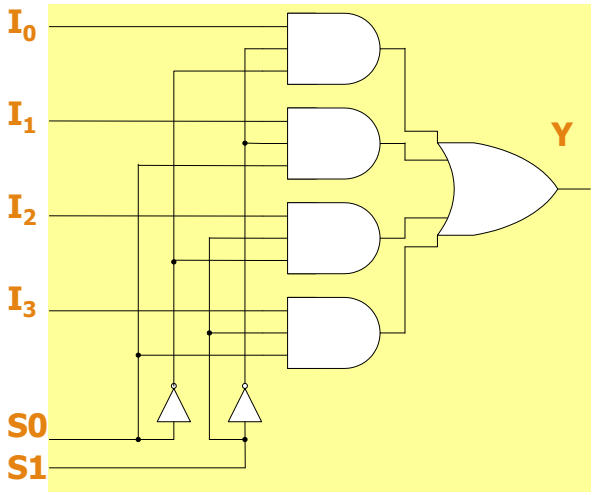
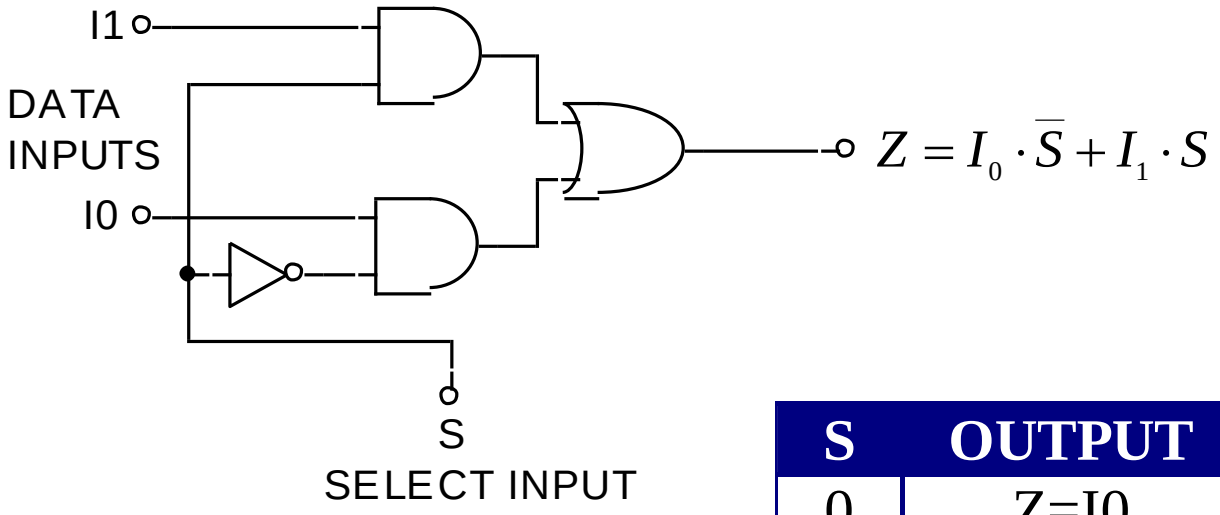


Fig. 4-to-1 Line Multiplexer

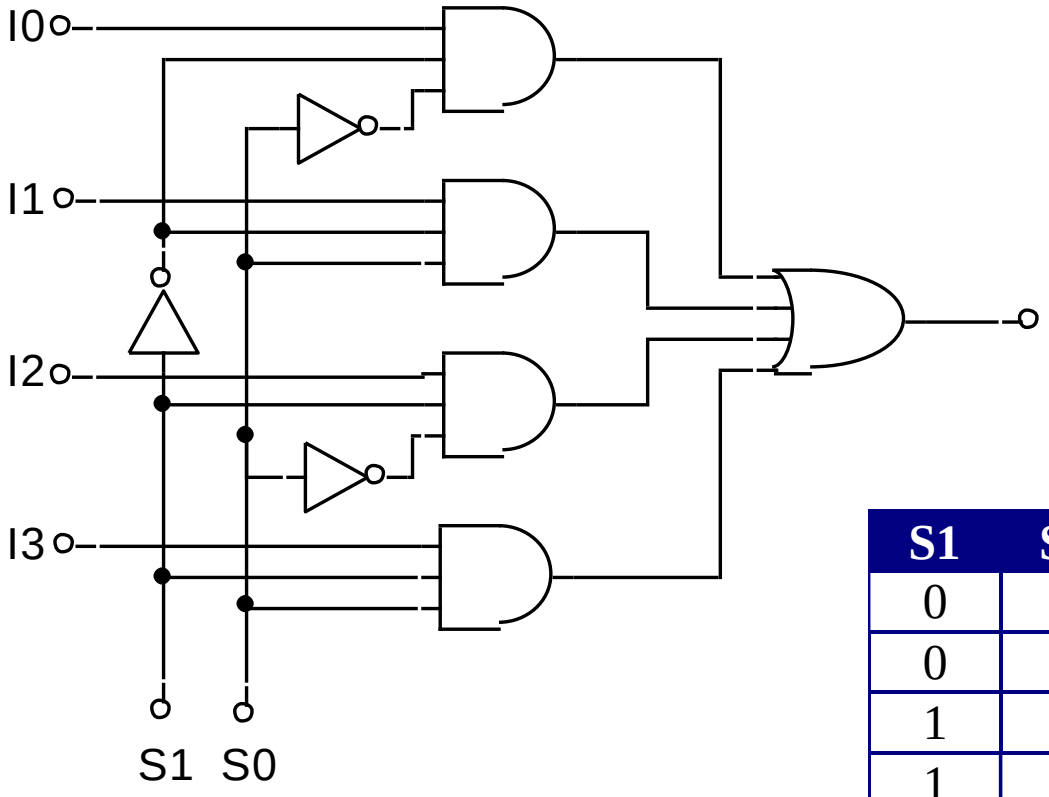
Usage: Data routing, Parallel-to-serial conversion, Operation sequencing, Implement logic function of a truth table

Basic 2-Input Multiplexer



S	OUTPUT
0	Z=I0
1	Z=I1

4-Input Multiplexer

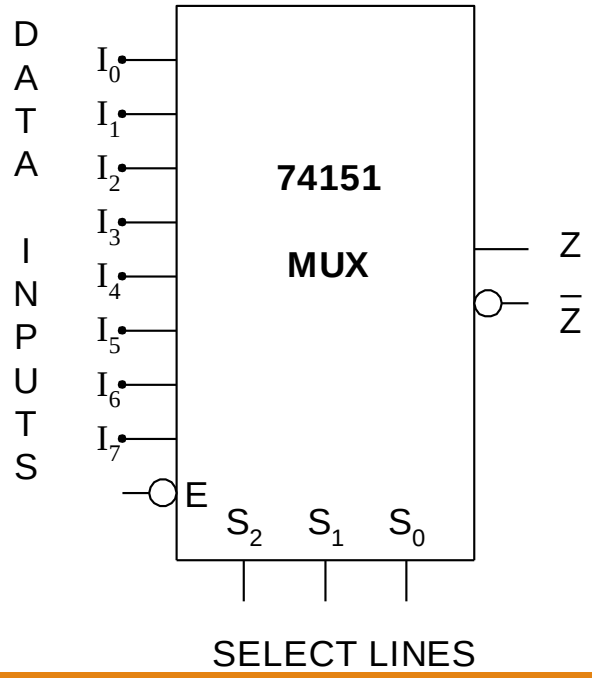


Select Inputs

S1	S0	OUTPUT
0	0	Z=I0
0	1	Z=I1
1	0	Z=I2
1	1	Z=I3

Multiplexer Logic Diagram

- Takes one of many inputs and funnels it to an output Z.
- Take the selector lines convert to a decimal number and this is the input funneled to the output.
- Strobe is active low enable



S2	S1	S0	E	Z
0	0	0	0	I0
0	0	1	0	I1
0	1	0	0	I2
0	1	1	0	I3
1	0	0	0	I4
1	0	1	0	I5
1	1	0	0	I6
1	1	1	0	I7

Multiplexer

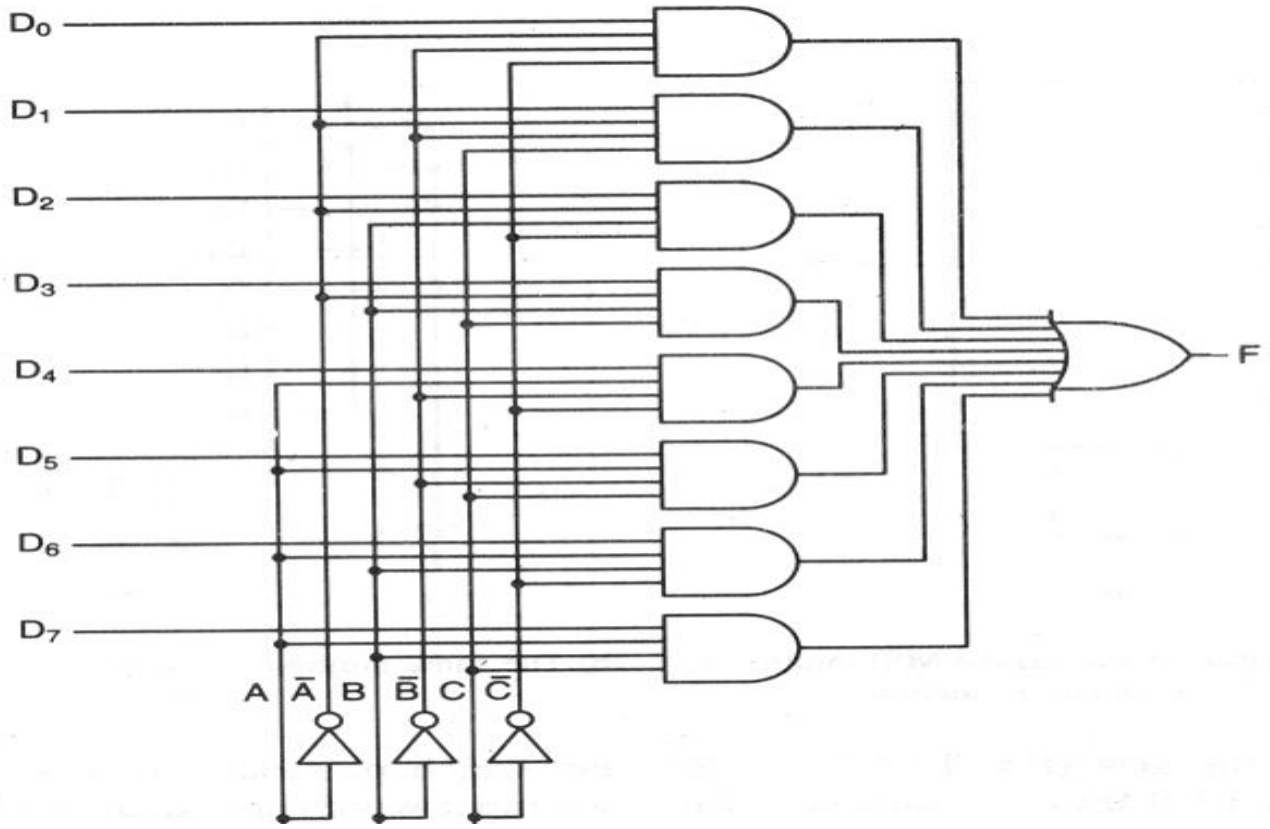


Figure :A 8-line to 1-line multiplexer

Programmable Logic Arrays (PLA)

Programmable logic arrays (PLA) are very general chips that provide a quick and easy way to implement Boolean functions expressed in the sum-of-products form.

- A PLA comprises of an array of AND gates and another array of OR gates.
  - Each one of the AND gate outputs is tapped through tiny fuses to the inputs of each OR gate.
  - Similarly, the input lines of the PLA chip are tapped to the inputs of each AND gate.
- Applying a high voltage to the chip a designer can blow selected fuses leaving only the desired circuit connections intact.
- Therefore, by choosing the proper set of fuses to blow, a designer can implement or “program” the desired Boolean functions rapidly.

Programmable Logic Arrays (PLA)

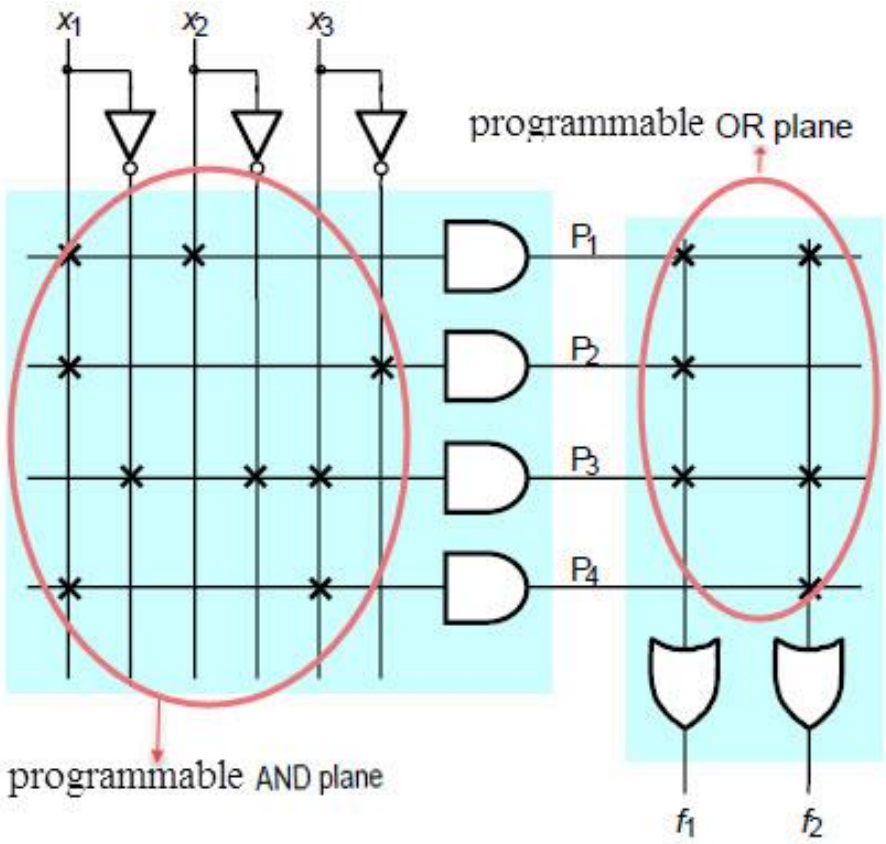


Figure: Customary schematic for the PLA

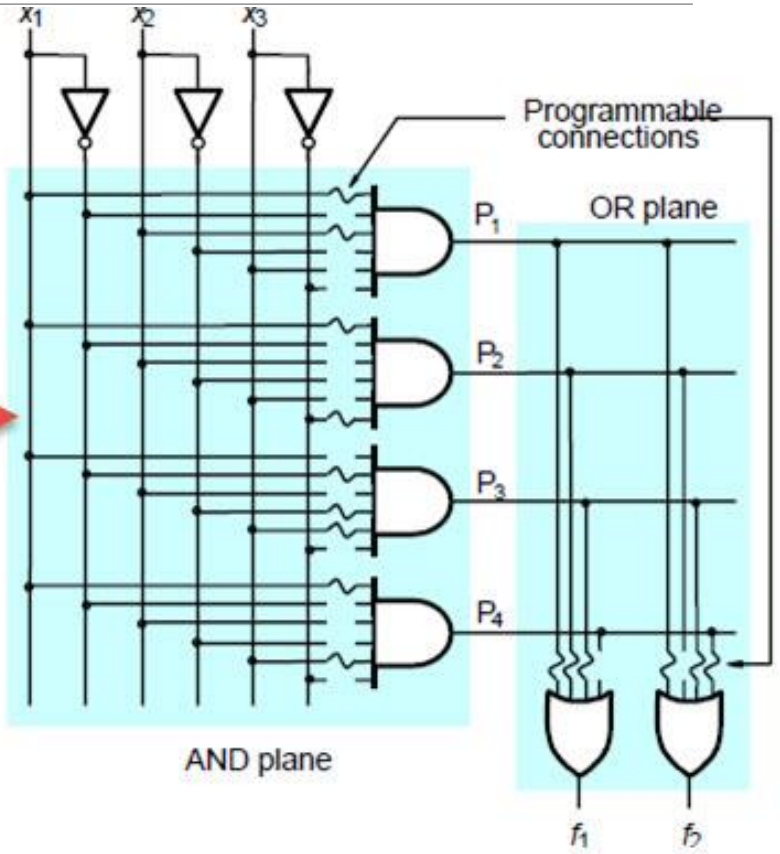


Figure: Gate-level diagram of a PLA

Arithmetic Logic Unit (ALU)

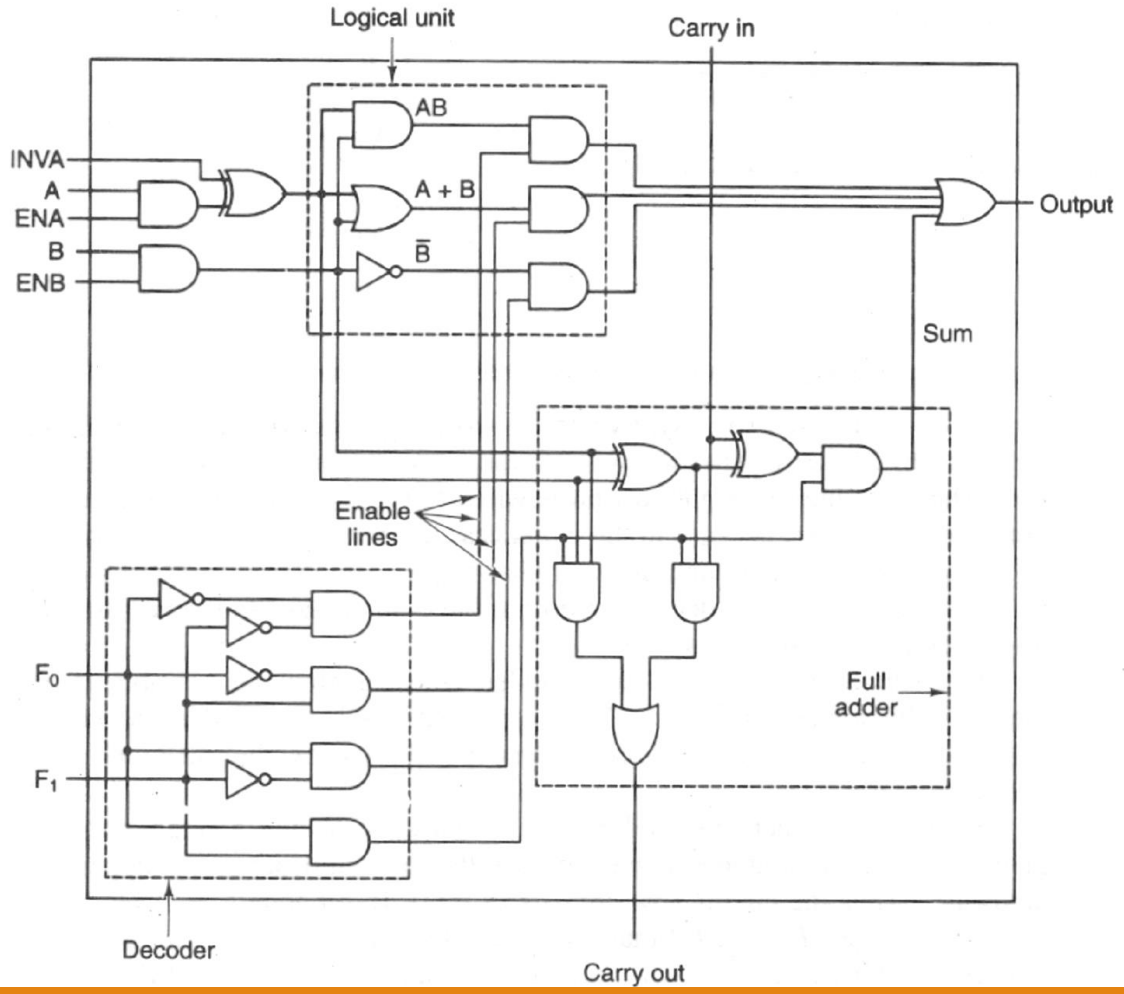
At the heart of any computer CPU is the arithmetic logic unit or the ALU.

It is a circuit that performs AND, OR, and sum operations of two input words.

AND and OR logical operations are easily implemented using the corresponding logic gates.

The sum operation is achieved using the full adder circuit described earlier.

Arithmetic Logic Unit (ALU)



Arithmetic Logic Unit (ALU)

If the CPU uses *n*-bit machine words, then *n* such ALU blocks must be connected together.

While the AND and OR operations only need the *n* ALUs to be placed in the *n* bit positions, the SUM operation requires the Carry-in and Carry-out signals of adjacent units cascaded

Arithmetic Logic Unit (ALU)

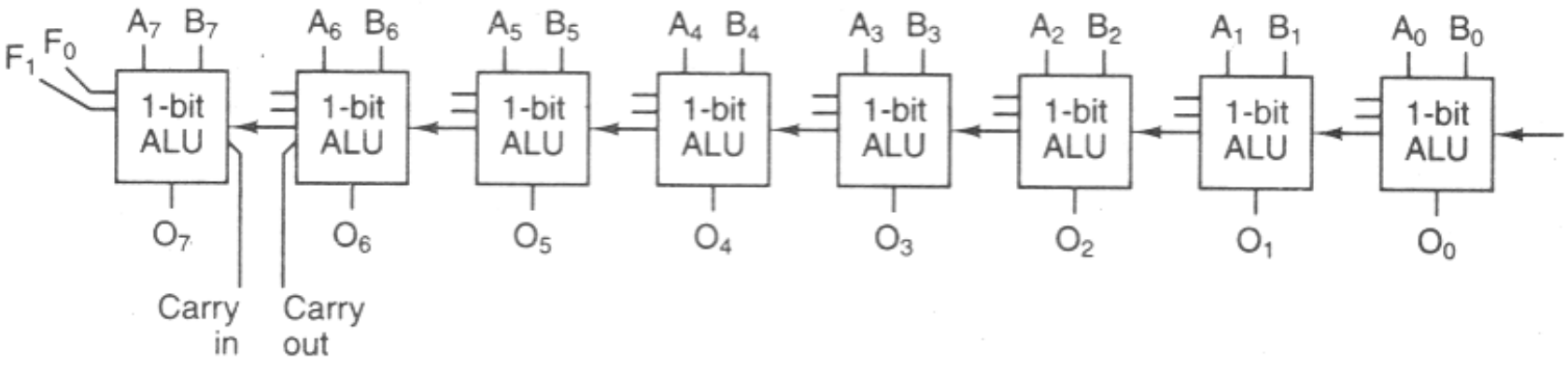


Figure 4-27. Cascading eight 1-bit ALUs to form an 8-bit ALU

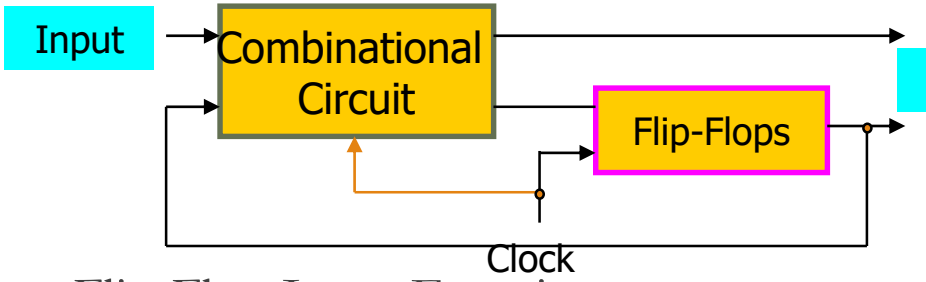
Sequential Circuits

Sequential Circuits

A sequential circuit is an interconnection of F/F and Gate

Clocked synchronous sequential circuit

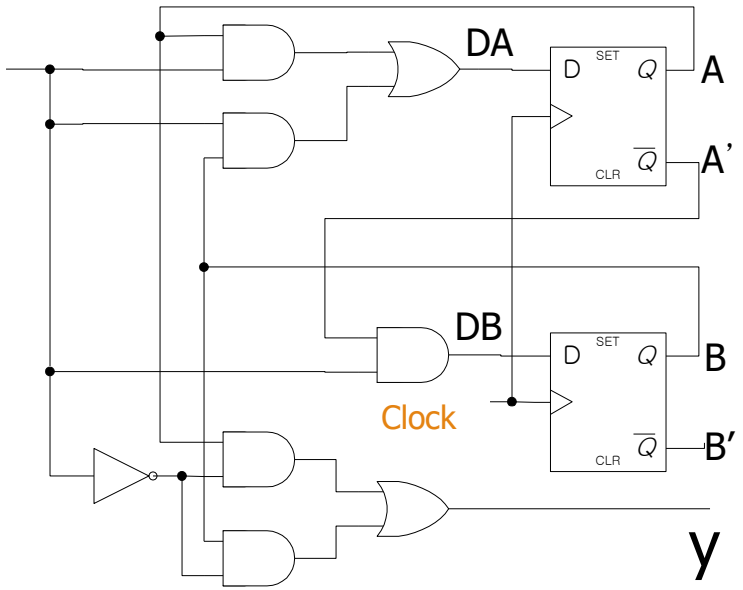
Combinational Circuit = Gate
Sequential Circuit = Gate + F/F



Flip-Flop Input Equation

- Boolean expression for F/F input
- Input Equation
  - $D_A = Ax + Bx$, $D_B = A'x$
- Output Equation
  - $y = Ax' + Bx'$
- *Example of a sequential circuit*

x



Sequential Circuits

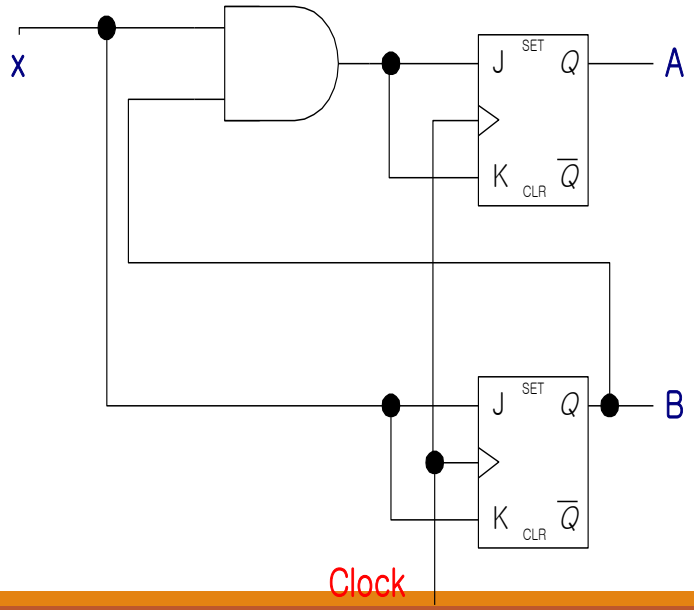
Sequential Circuit Design Procedure

1. The Problem is stated
2. I/O variables are assigned
3. Truth table(I/O relation)
4. Simplified Boolean Functio
5. Logic circuit diagram

Recall Combinational Circuit Design

Sequential Circuit: State diagram, State table

F/F : 2^{m+n} (m - State , n - Input)

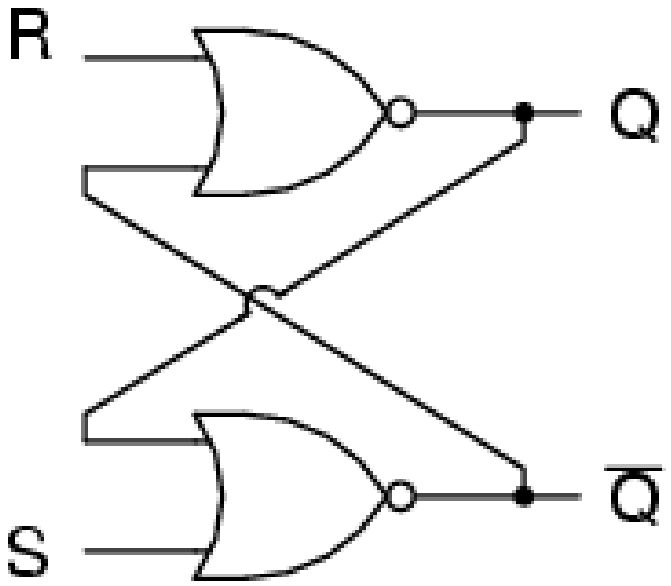


- Logic Diagram

Latches

SR Latch

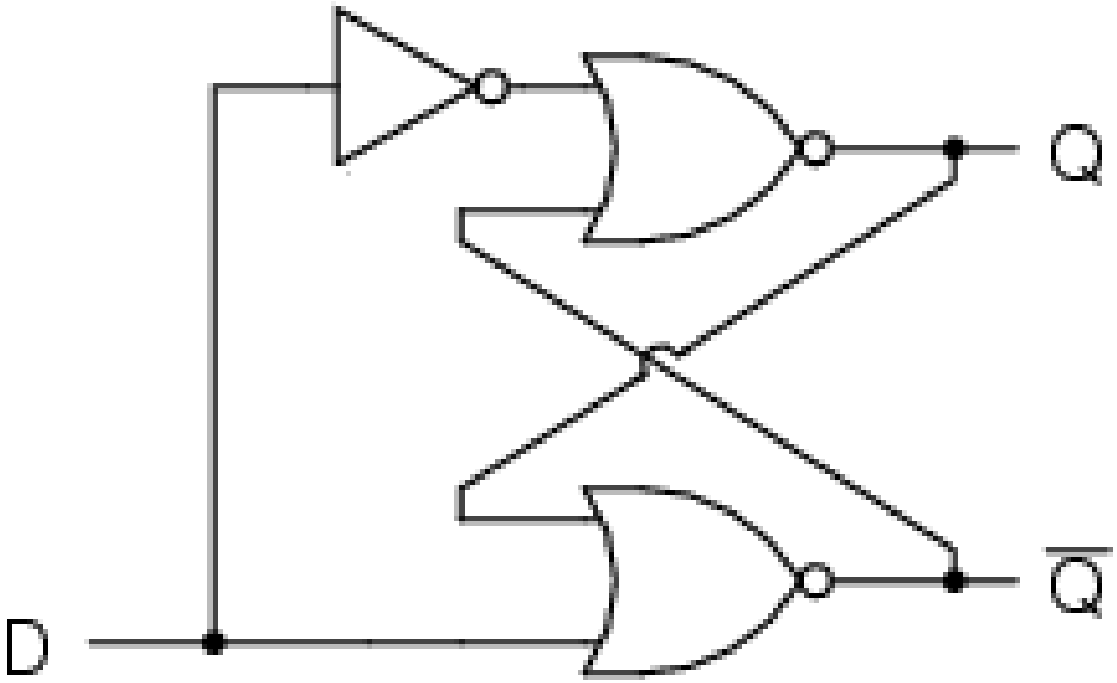
S	R	Action
0	0	Keep state
0	1	Q=0
1	0	Q=1
1	1	Restricted Combination



Characteristic Table

Latches

D Latch



Clock Signals

Clock signals are used to maintain the desired timing in the circuits.

- Clock circuits emit pulse trains of precise repetition interval and width.
- Sometimes it is necessary to have one clock pulse train trail another by a fixed time.
- A circuit with the appropriate delay may be inserted to achieve the desired phase shift

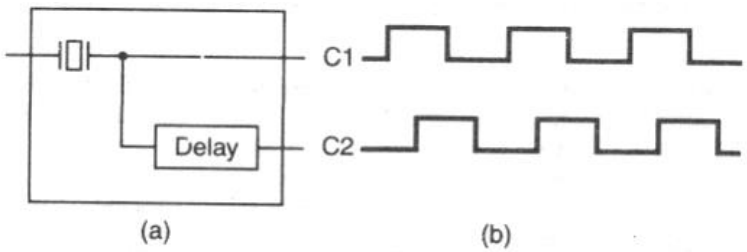
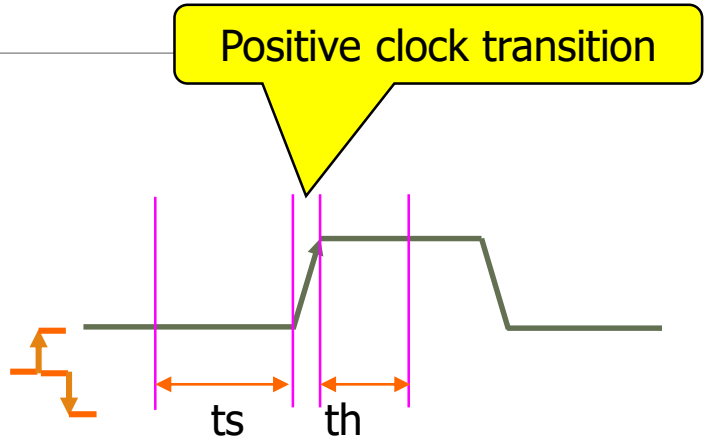


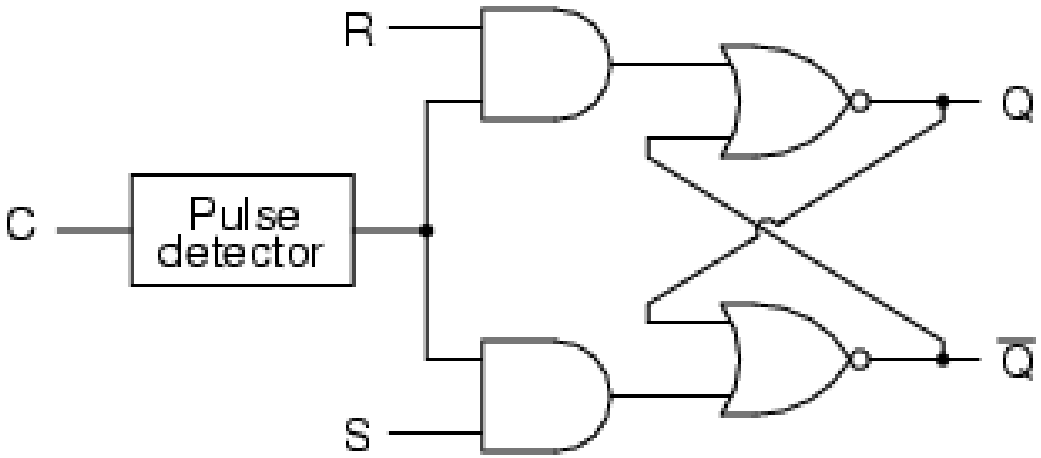
Figure 4-28. Clock circuit and the clock waveforms

Edge-Triggered Flip Flops



- State Change : *Clock Pulse*
  - Rising Edge(positive-edge transition)
  - Falling Edge(negative-edge transition)
- Setup time(20ns)
  - Minimum time that D input must remain at constant value before the transition.
- Hold time(5ns)
  - Minimum time that D input must not change after the positive transition.
- Propagation delay(max 50ns)
  - time between the clock input and the response in Q

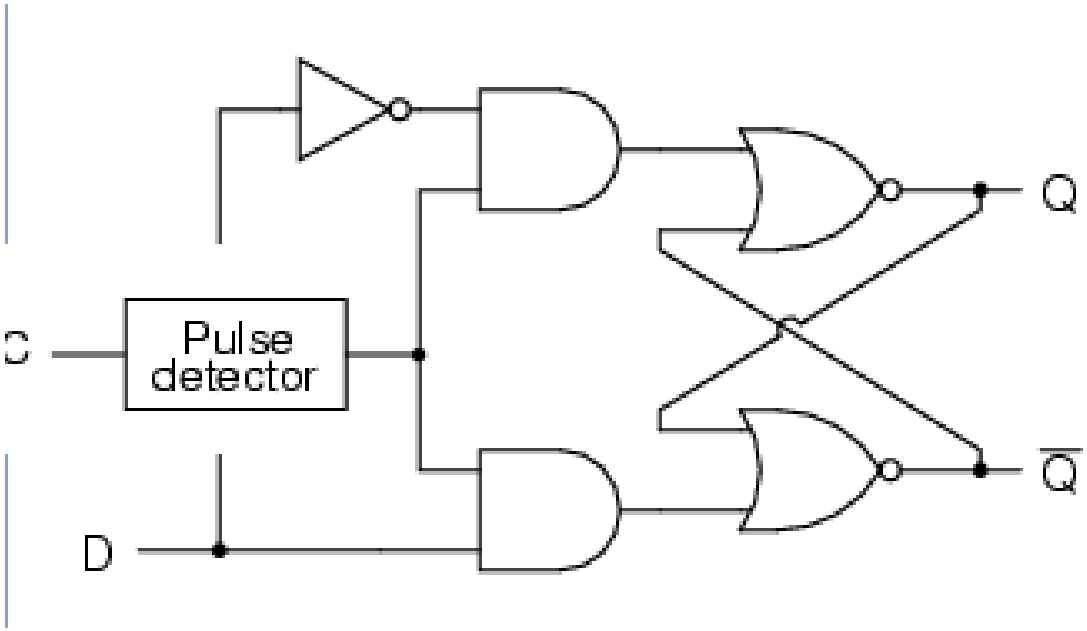
SR Flip-flop



C	S	R	Q	$\overline{Q}$
⌐	0	0	latch	latch
⌐	0	1	0	1
⌐	1	0	1	0
⌐	1	1	0	0
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

Characteristic Table

D Flip-flop



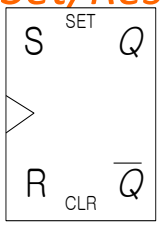
E	D	Q	$\overline{Q}$
0	**0**	latch	latch
0	1	latch	latch
1	**0**	**0**	1
1	1	1	**0**

Flip-flops

The *storage elements* employed in clocked *sequential circuit*

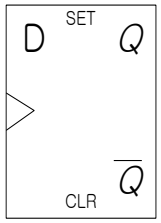
A binary cell capable of storing one bit of information

- SR *(Set/Reset)* F/F



S	R	Q(t+1)	
0	0	Q(t)	no change
0	1	0	clear to 0
1	0	1	set to 1
1	1	?	Indeterminate

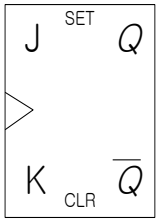
- D *(Data)* F/F



D	Q(t+1)	
0	0	clear to 0
1	1	set to 1

"no change" condition: Q(t+1)=D

- JK *(Jack/King)* F/F

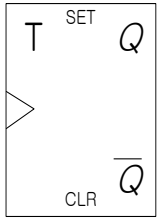


J	K	Q(t+1)	
0	0	Q(t)	no change
0	1	0	clear to 0
1	0	1	set to 1
1	1	Q(t)'	Complement

- JK F/F is a refinement of the SR F/F
- The indeterminate condition of the SR type is defined in complement

1) Disable Clock 2) Feedback output into input

- T *(Toggle)* F/F



T	Q(t+1)	
0	Q(t)	no change
1	Q'(t)	Complement

- T=1(J=K=1), T=0(J=K=0) JK F/F
- $Q(t+1)= Q(t) \oplus T$

Flip-flops

Excitation Table

- Required input combinations for a given change of state
- Present State Next State

SR F/F			
Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	0	1
1	0	1	0
1	1	X	1

JK F/F			
Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

D F/F		
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

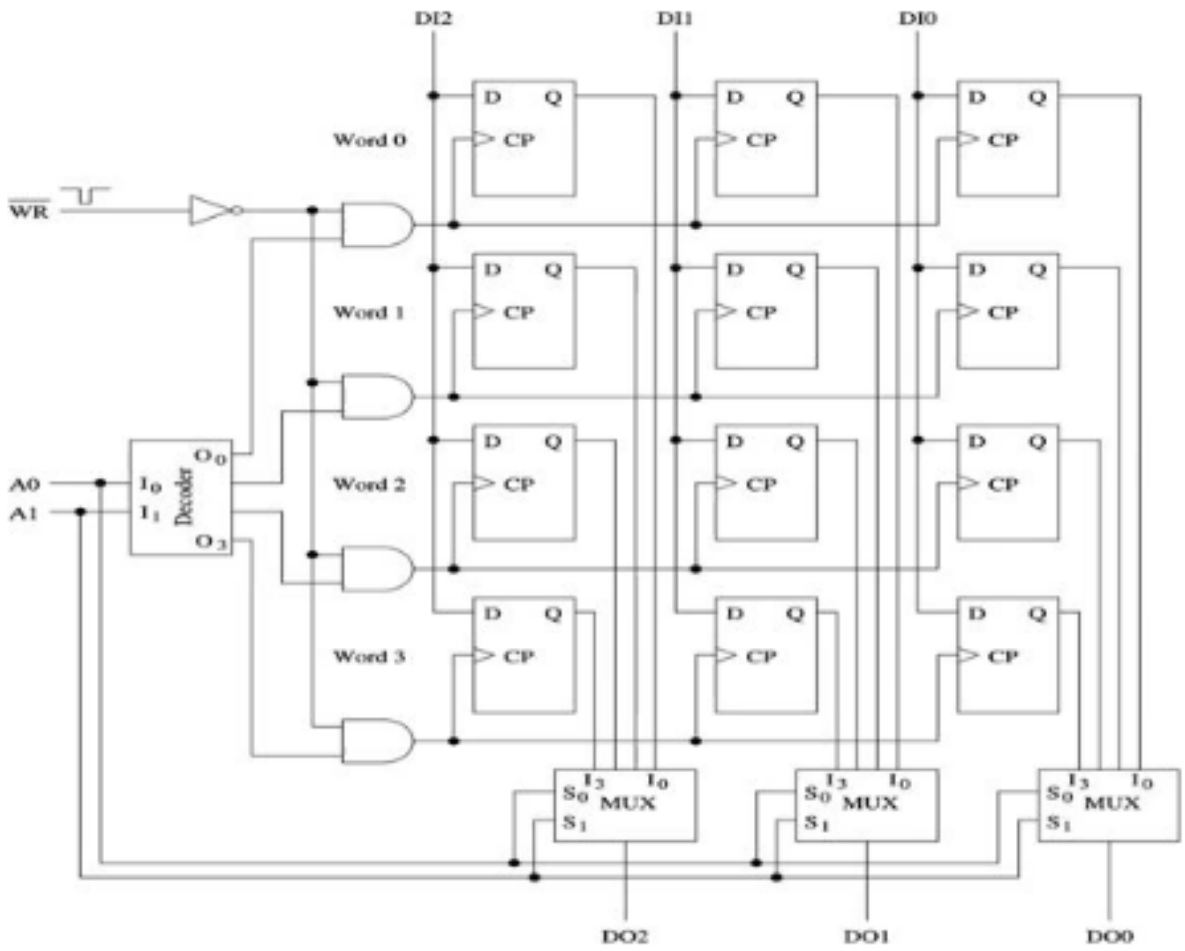
T F/F		
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Don't Care

1 : Set to 1
0 : Complement

1 : Clear to 0
0 : No change

Use of Circuits: Memory



Thank You
End of CF from ICS