Data & Information

Lecture 1

Introduction

What is data?

Data is a raw and unorganized fact that is required to be processed to make it meaningful. It can be considered as facts and statistics collected together for reference or analysis. Data are individual units of information

What is information?

Processed data is considered as information. Information is the knowledge that is remodeled and classified into an intelligible type, which may be utilized in the method of deciding.

Difference between data and information

Home Task: Write down the differences between data and information.

Let's Guess

$$5+5=10$$

1. **What are the data in this equation?**

2. **What is the information in this equation?**

Computer

The word computer comes from the word “compute”, which means, “to calculate”

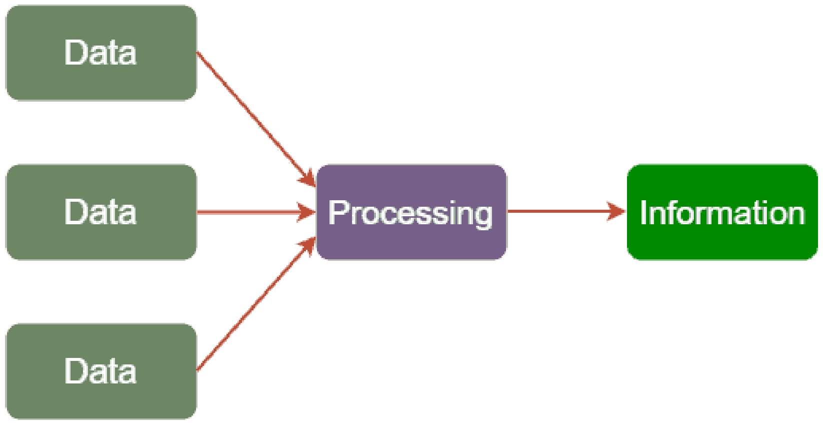
Thereby, a computer is an electronic device that can perform arithmetic operations at high speed

A computer is also called a data processor because it can store, process, and retrieve data whenever desired

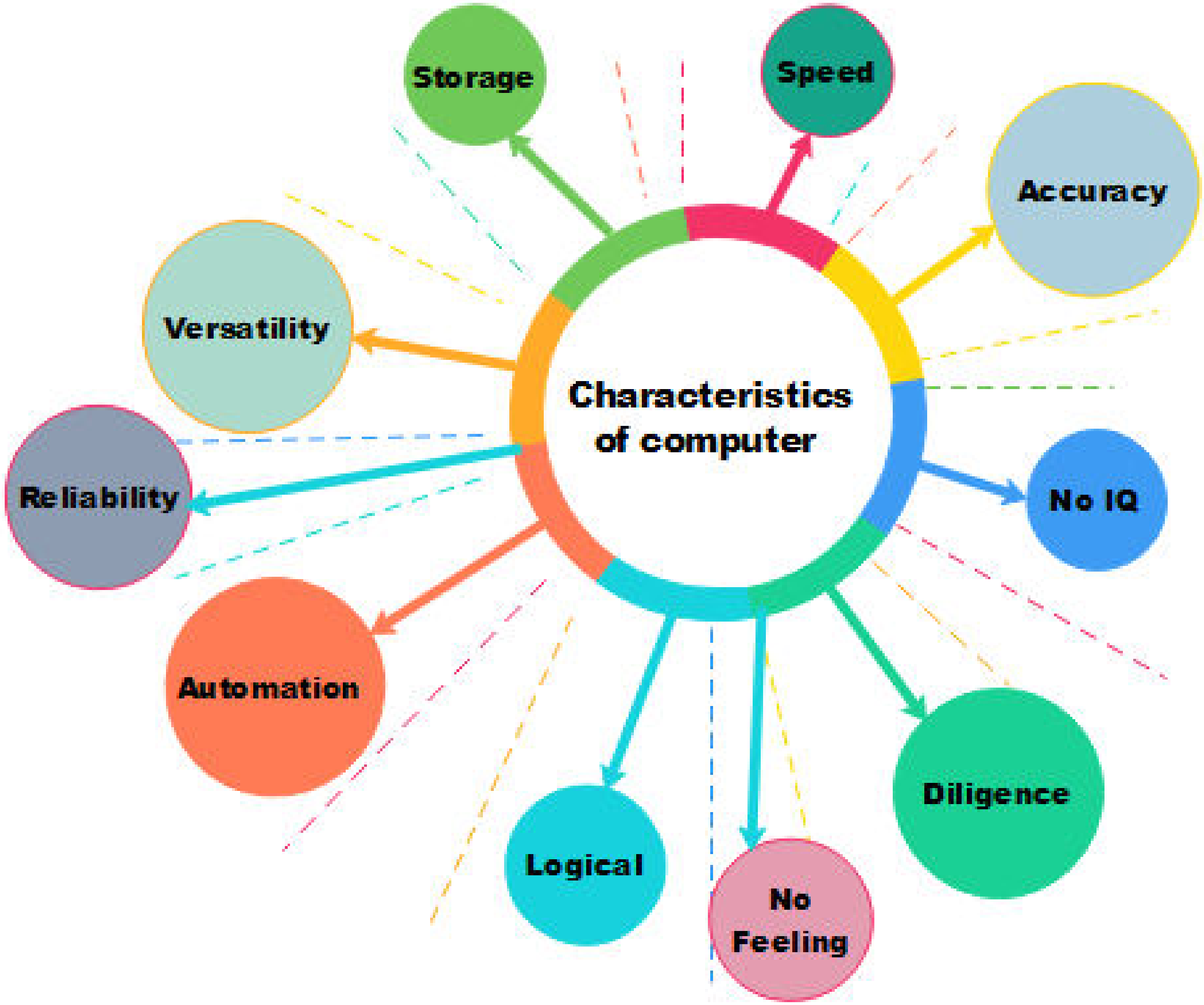
Data Processing

The activity of processing data using a computer is called data processing

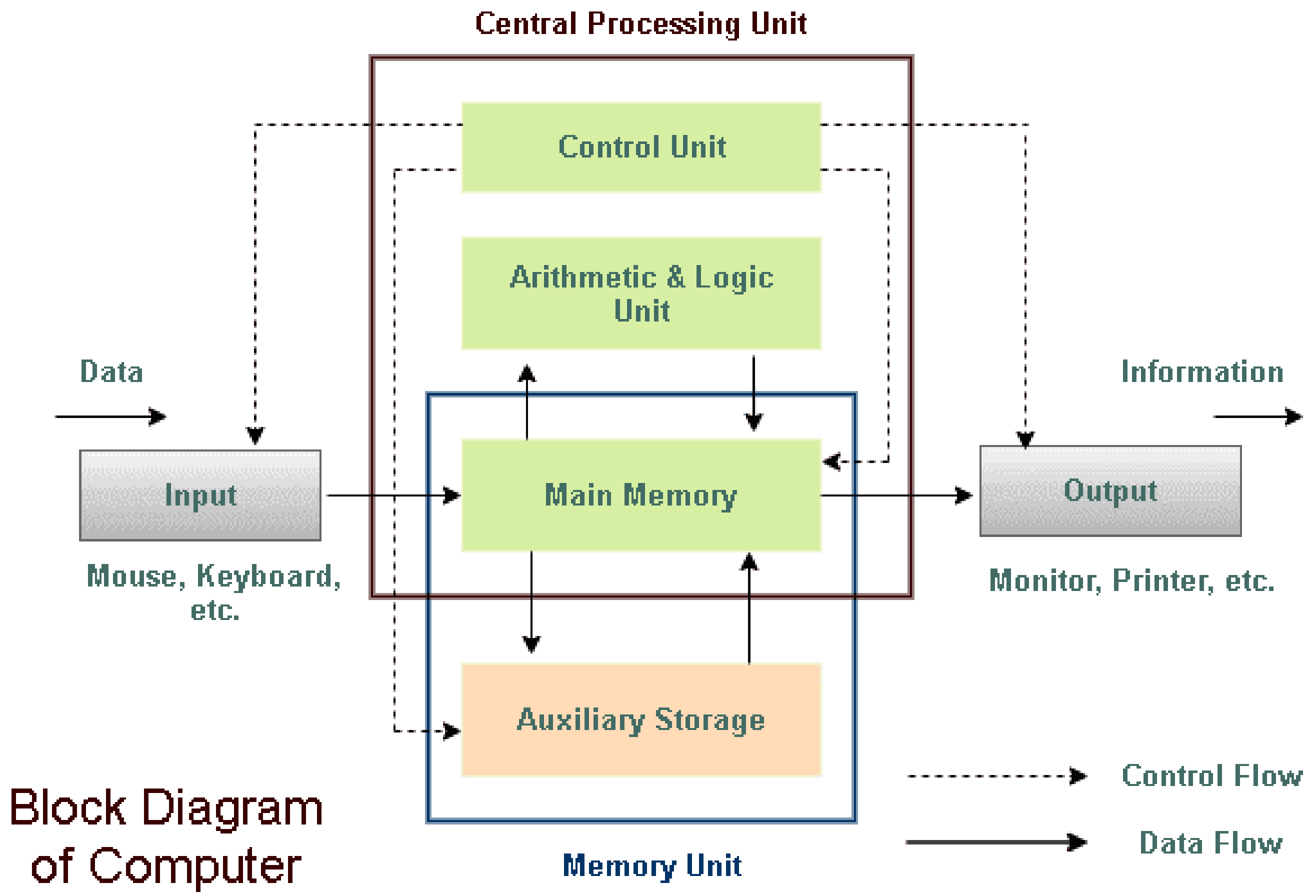
Data is raw material used as input and information is processed data obtained as output of data processing



Characteristics of Computers



Working of a computer



Working of a computer

The working of a computer can be broadly categorized into following four functions or steps.

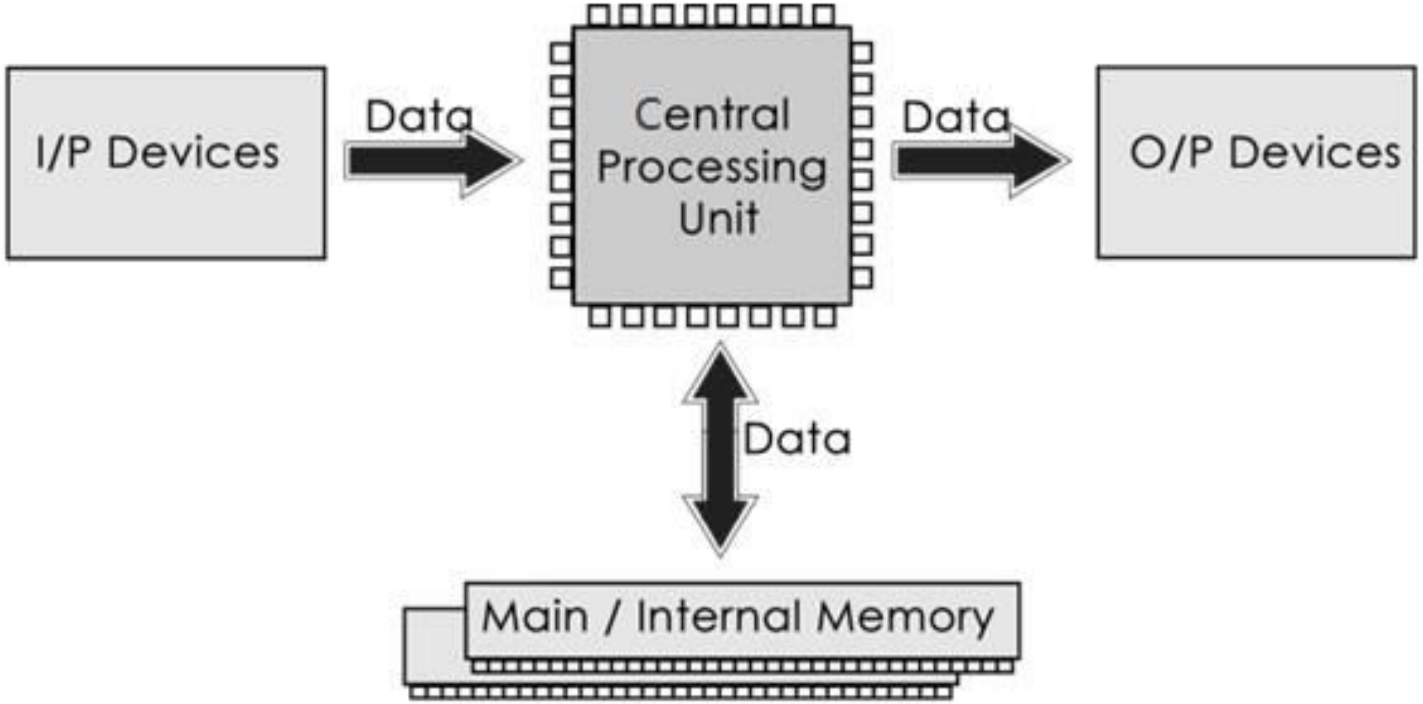
(i) Receive input – Accept data/information from user through various input devices like the keyboard, mouse, scanner, etc.

(ii) Process information–Perform arithmetic or logical operations on data/information.

(iii) Store information—Store the information in storage devices like hard disk, CD, pen drive etc.

(iv) Produce output–Communicate information to the user through any of the available output devices like monitor, printer, etc.

Components of a computer



That’s all for today...........

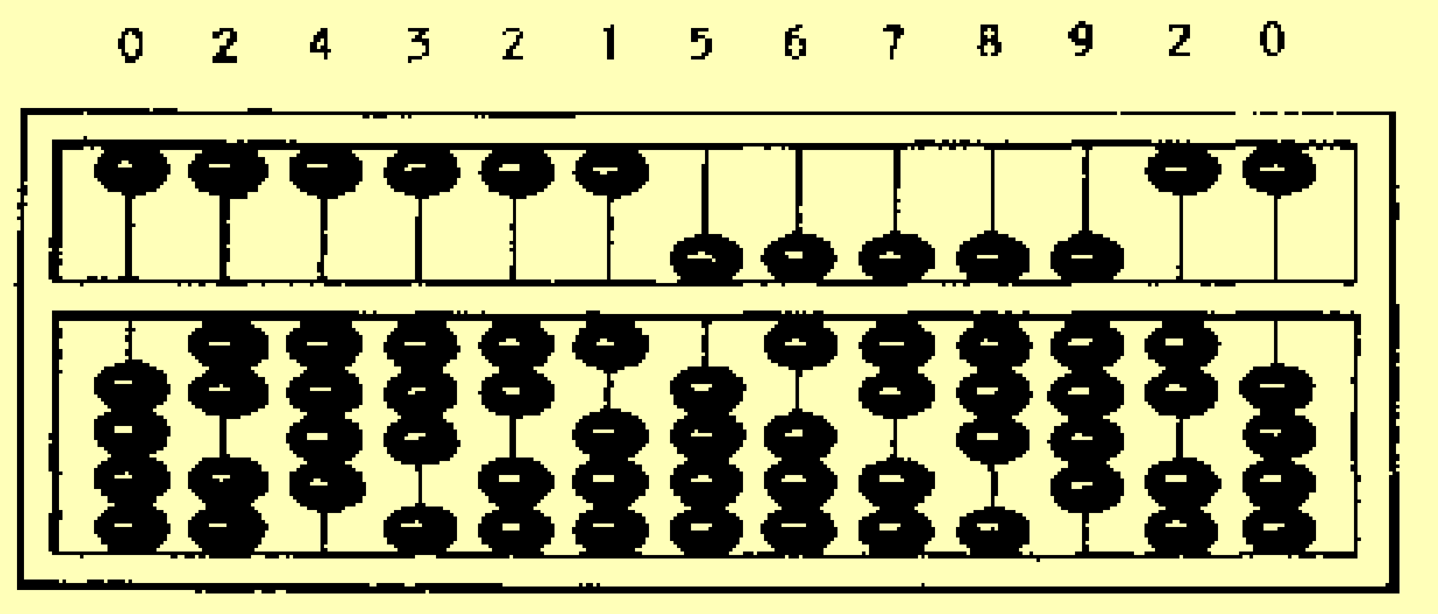
History and Evolution of Computer

Lecture 4

Evolution of Computers

The abacus is a manually operated digital computer used in ancient civilizations

The use of the word abacus dates before 1387 AD



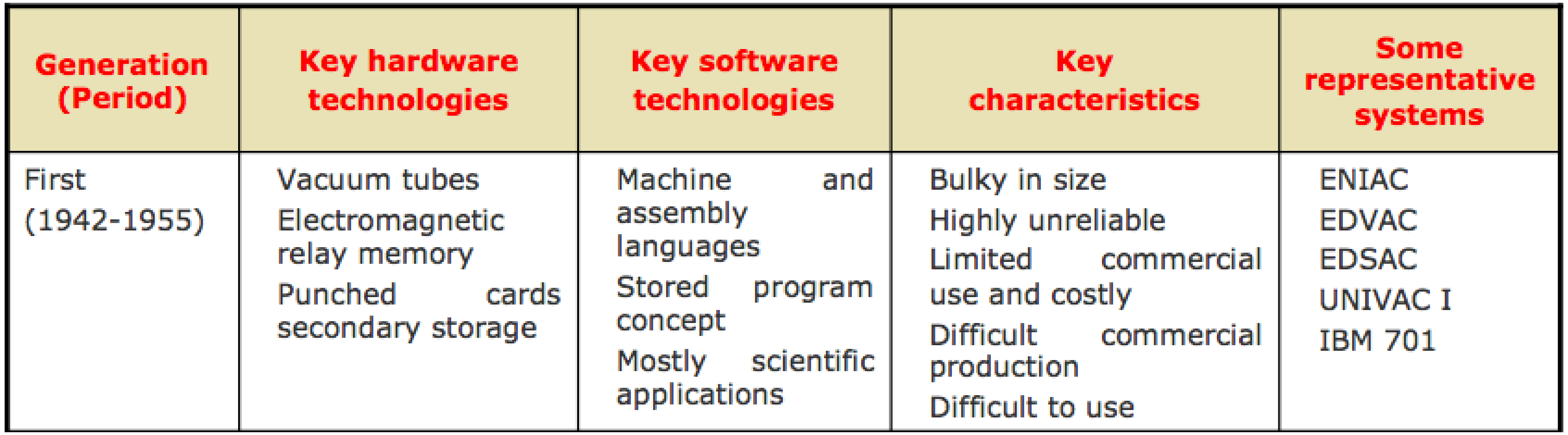
Computer Generations

“Generation” in computer talk is a step in technology. It provides a framework for the growth of computer industry

Originally it was used to distinguish between various hardware technologies, but now it has been extended to include both hardware and software

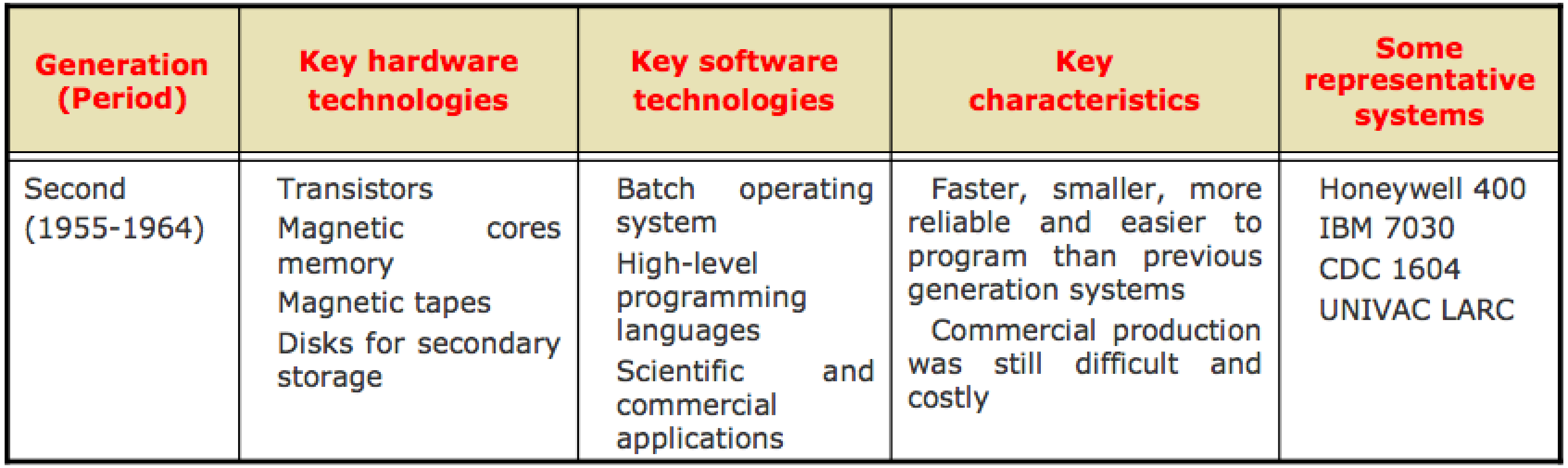
Till today, there are five computer generations

1st Generation Computer

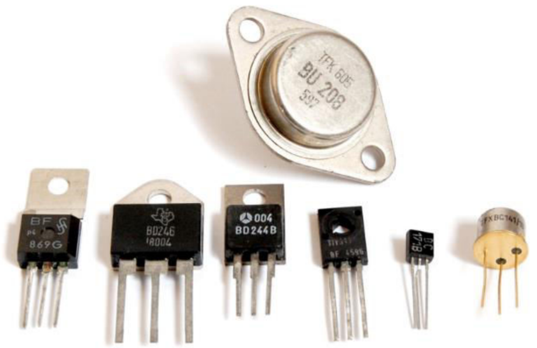


Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some representative systems
First (1942-1955)	Vacuum tubes Electromagnetic relay memory Punched cards secondary storage	Machine and assembly languages Stored program concept Mostly scientific applications	Bulky in size Highly unreliable Limited commercial use and costly Difficult commercial production Difficult to use	ENIAC EDVAC EDSAC UNIVAC I IBM 701

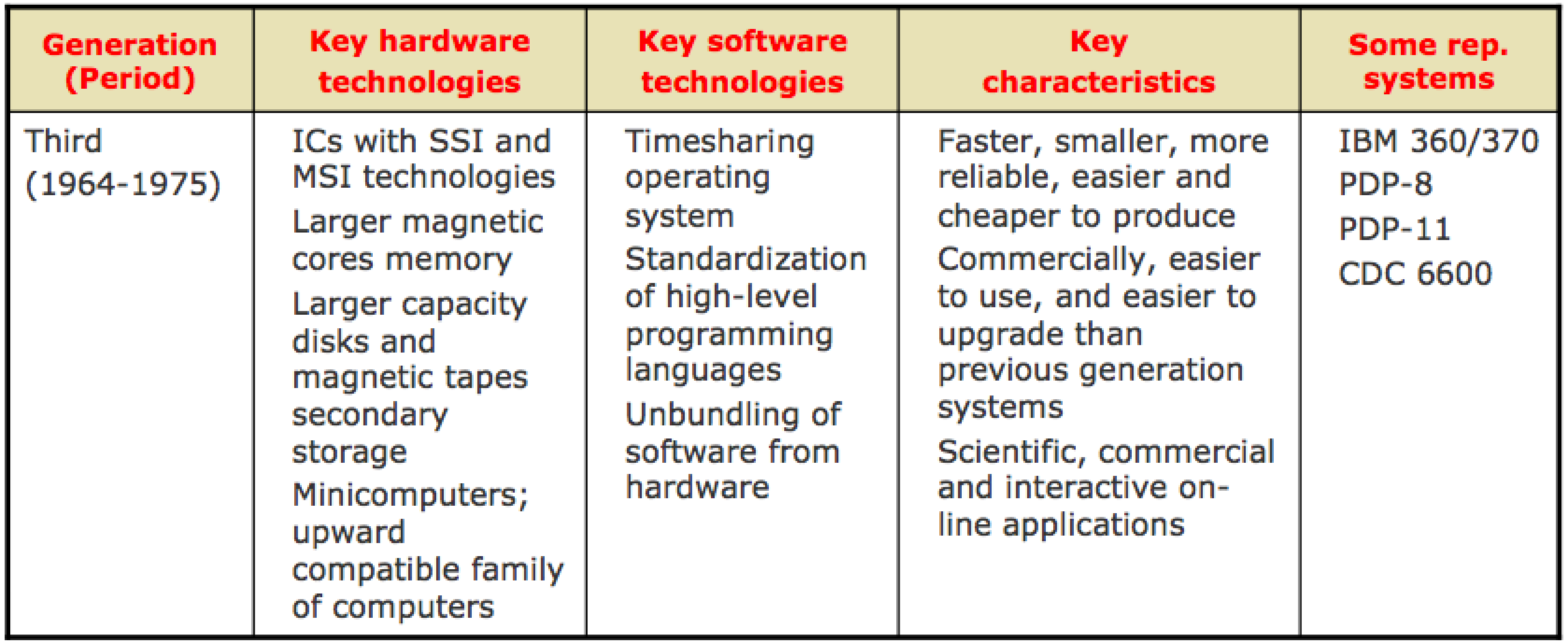
2nd Generation Computer



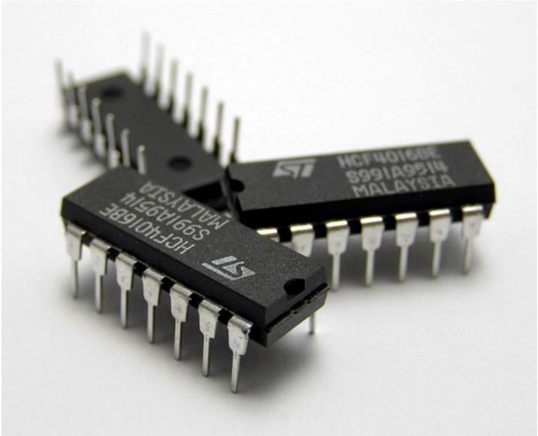
Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some representative systems
Second (1955-1964)	Transistors Magnetic cores memory Magnetic tapes Disks for secondary storage	Batch operating system High-level programming languages Scientific and commercial applications	Faster, smaller, more reliable and easier to program than previous generation systems Commercial production was still difficult and costly	Honeywell 400 IBM 7030 CDC 1604 UNIVAC LARC



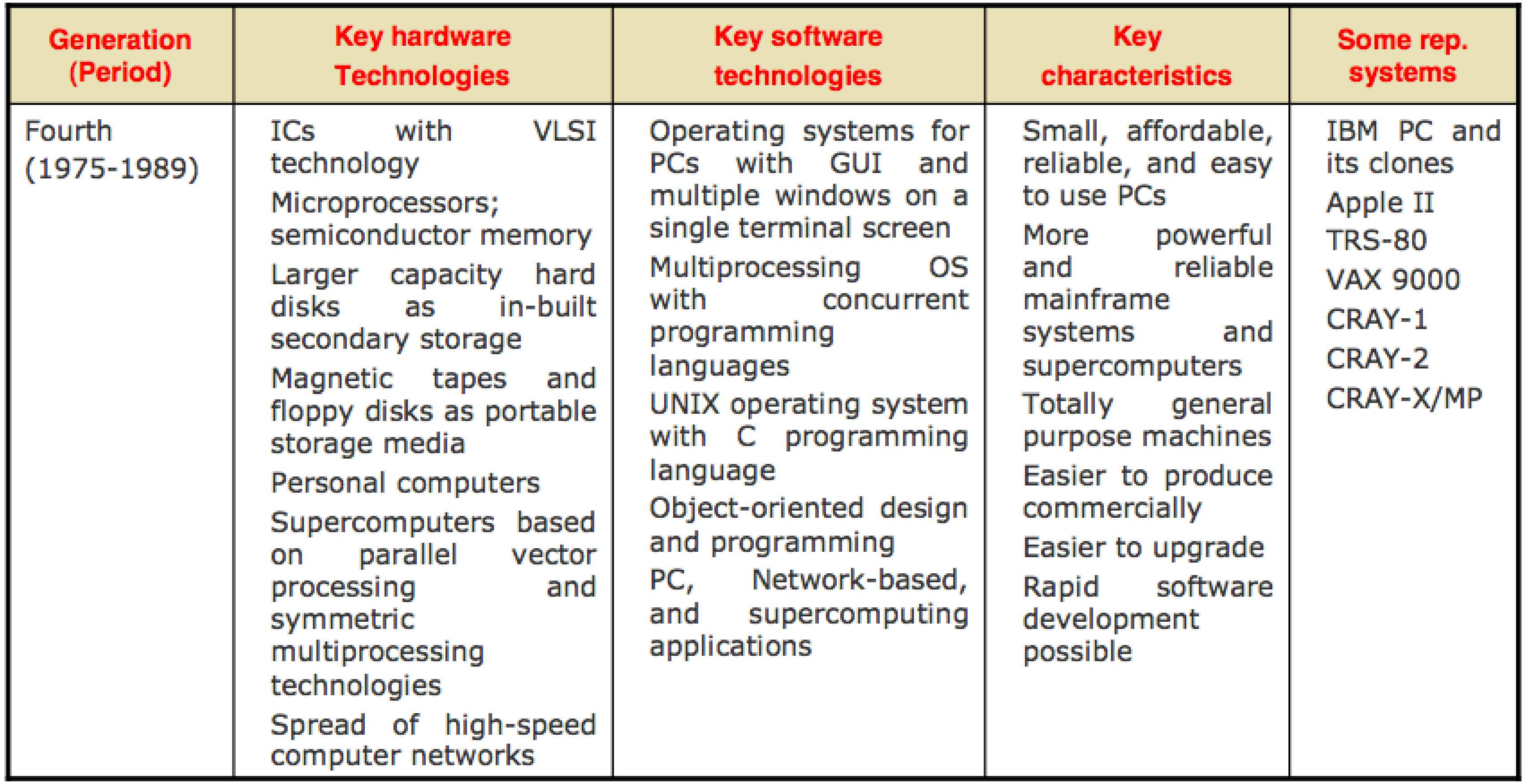
3rd Generation Computer



Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some rep. systems
Third (1964-1975)	ICs with SSI and MSI technologies Larger magnetic cores memory Larger capacity disks and magnetic tapes secondary storage Minicomputers; upward compatible family of computers	Timesharing operating system Standardization of high-level programming languages Unbundling of software from hardware	Faster, smaller, more reliable, easier and cheaper to produce Commercially, easier to use, and easier to upgrade than previous generation systems Scientific, commercial and interactive on-line applications	IBM 360/370 PDP-8 PDP-11 CDC 6600

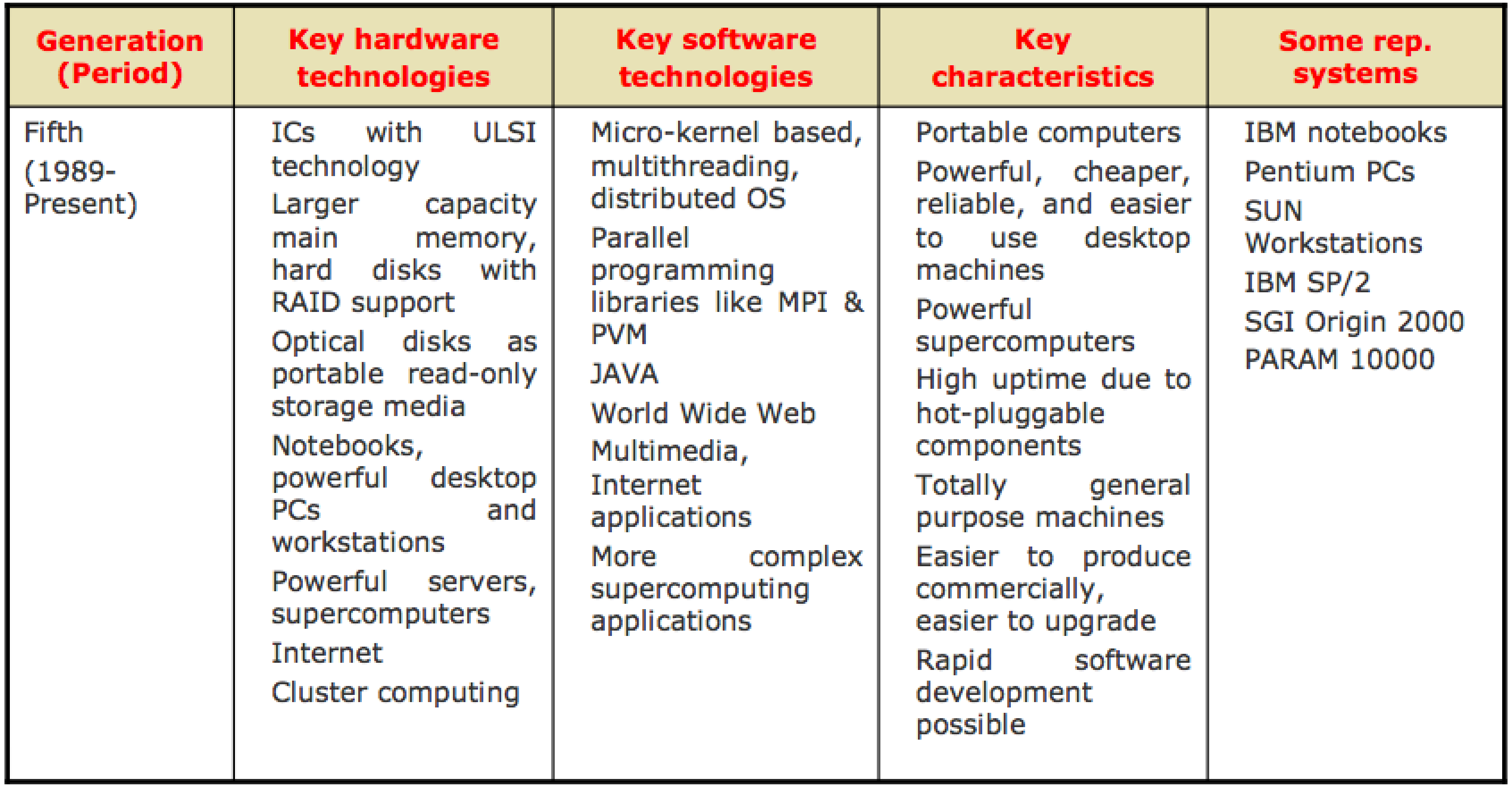


4th Generation Computer



Generation (Period)	Key hardware Technologies	Key software technologies	Key characteristics	Some rep. systems
Fourth (1975-1989)	ICs with VLSI technology Microprocessors; semiconductor memory Larger capacity hard disks as in-built secondary storage Magnetic tapes and floppy disks as portable storage media Personal computers Supercomputers based on parallel vector processing and symmetric multiprocessing technologies Spread of high-speed computer networks	Operating systems for PCs with GUI and multiple windows on a single terminal screen Multiprocessing OS with concurrent programming languages UNIX operating system with C programming language Object-oriented design and programming PC, Network-based, and supercomputing applications	Small, affordable, reliable, and easy to use PCs More powerful and reliable mainframe systems and supercomputers Totally general purpose machines Easier to produce commercially Easier to upgrade Rapid software development possible	IBM PC and its clones Apple II TRS-80 VAX 9000 CRAY-1 CRAY-2 CRAY-X/MP

5th Generation Computer



Generation (Period)	Key hardware technologies	Key software technologies	Key characteristics	Some rep. systems
Fifth (1989-Present)	ICs with ULSI technology Larger capacity main memory, hard disks with RAID support Optical disks as portable read-only storage media Notebooks, powerful desktop PCs and workstations Powerful servers, supercomputers Internet Cluster computing	Micro-kernel based, multithreading, distributed OS Parallel programming libraries like MPI & PVM JAVA World Wide Web Multimedia, Internet applications More complex supercomputing applications	Portable computers Powerful, cheaper, reliable, and easier to use desktop machines Powerful supercomputers High uptime due to hot-pluggable components Totally general purpose machines Easier to produce commercially, easier to upgrade Rapid software development possible	IBM notebooks Pentium PCs SUN Workstations IBM SP/2 SGI Origin 2000 PARAM 10000

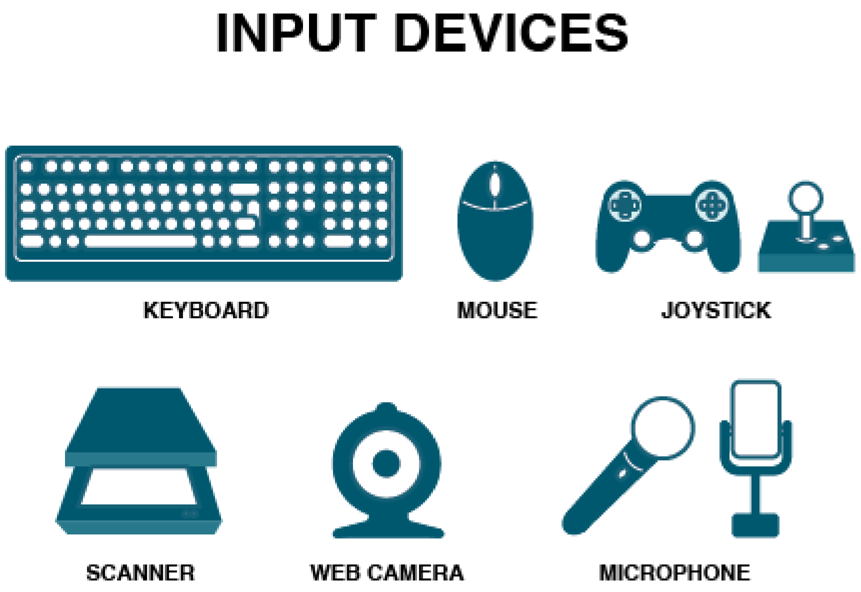
I/O Devices & Components of a computer

Lecture 2

INPUT DEVICES

An input device is used to get data or instructions from the user. This data is then

passed on to CPU for processing so as to produce the desired result.



INPUT DEVICES

Keyboard:

The keyboard is very much like a standard typewriter with a few additional keys.

Generally, we find a QWERTY keyboard with 104 keys on it. The additional keys may be included in modern multimedia keyboards.

Mouse:

It is basically a pointing device that controls the movement of the cursor or pointer on a display screen. It is a small object that you can roll along a hard and flat surface. As you move the mouse, the pointer on the display screen moves in the same direction.

INPUT DEVICES

Scanner:

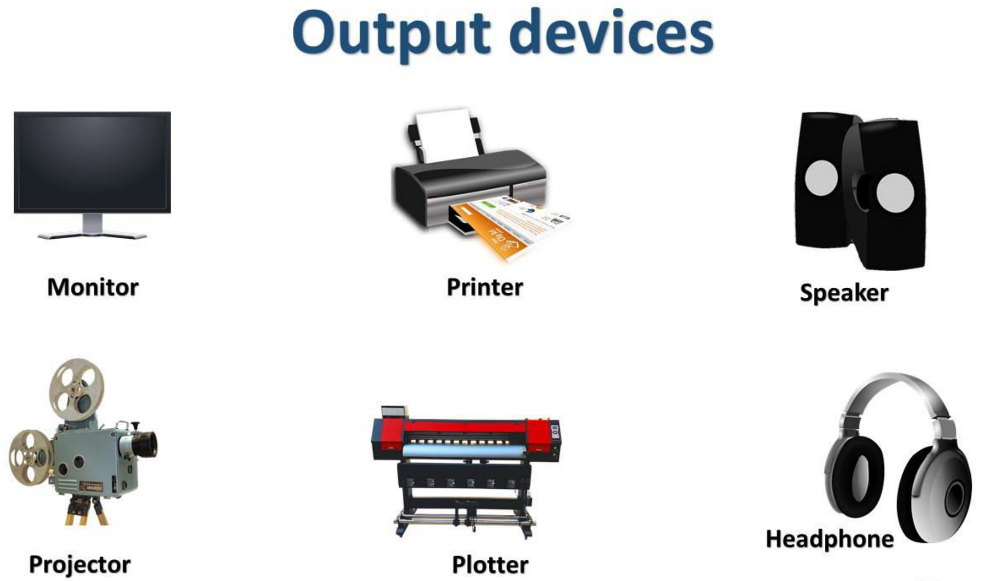
Scanner is an input device that can read text or an illustration printed on paper and translates the information into a form that the computer can use. A scanner works by digitizing an image - dividing it into a grid of boxes and representing each box with either a zero or a one, depending on whether the box is filled in.

Optical Character Recognition (OCR):

An Optical Character Recognition (OCR) is a device that is used for reading text from paper and translating the images into a form that the computer can understand. An OCR is used to convert books, magazines and other such printed information into digital form.

OUTPUT DEVICES

Output devices receive information from the CPU and present it to the user in the desired form. Some of the output devices are monitor, printers, plotters, etc.



www.xpartinfo.com

OUTPUT DEVICES

Monitor:

The monitor is just like a television screen and it is used to display data and information. When some data or instruction is being keyed in, the monitor displays the characters being typed..

Printer:

You must have used printer for taking printouts. Printer is a device that produces the output on paper. Such an output is also known as hard copy and it may be in the form of text or graphics.

Home Task:

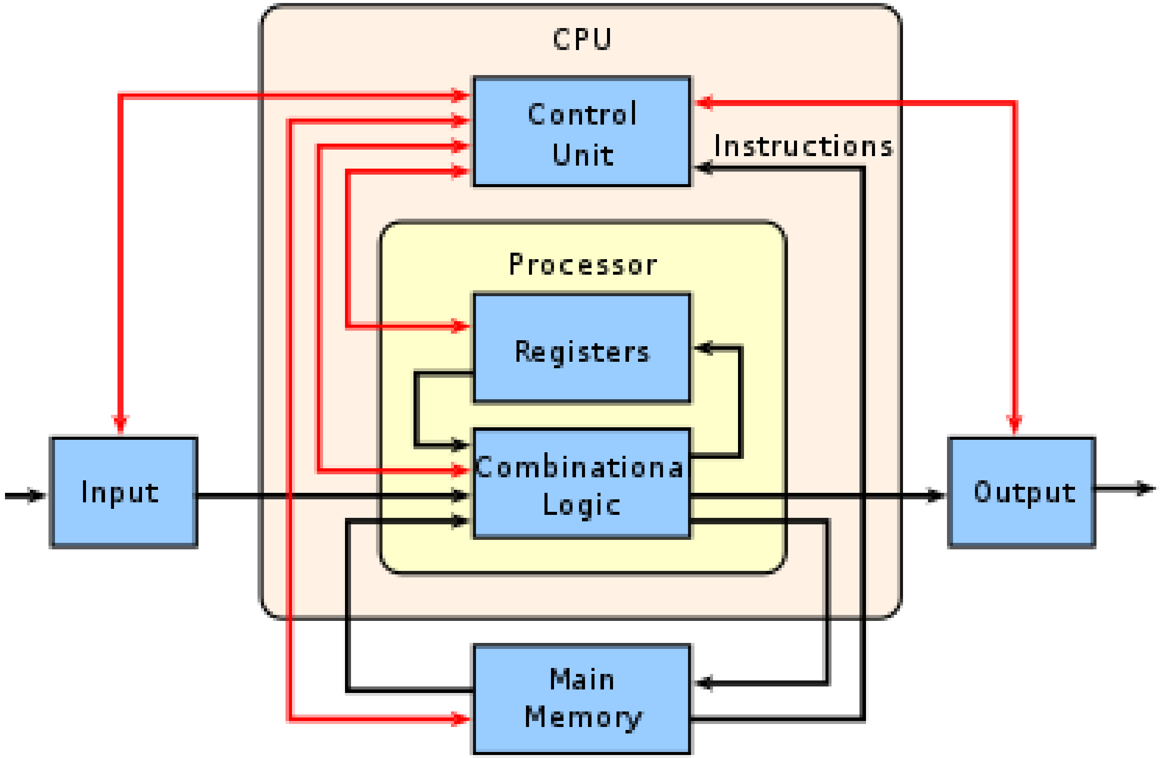
List out different types of I/O devices and write down about them

CENTRAL PROCESSING UNIT (CPU)

CPU is the ‘Brain of your computer’. This is because it processes or executes the instructions given to the computer. In a microcomputer, the CPU is based on a single chip called the microprocessor.

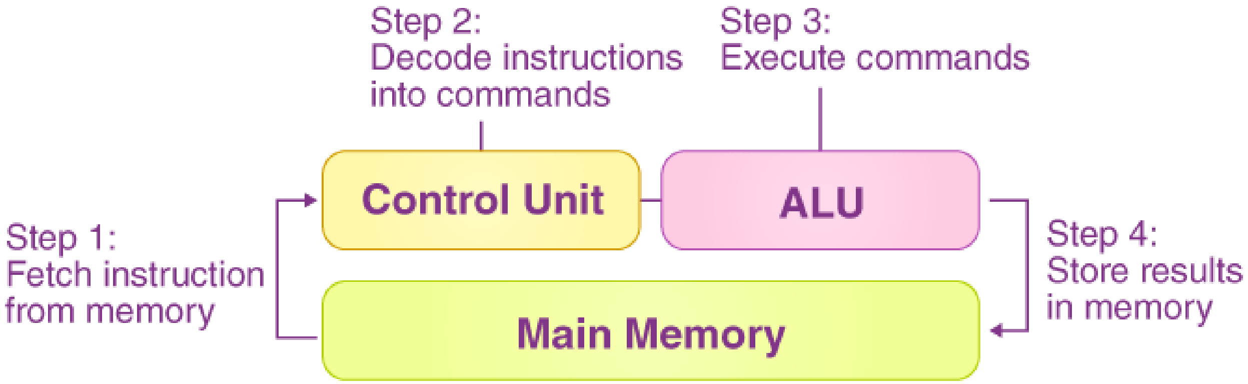
A typical CPU has the following components:

- ✓ Control Unit (CU)
- ✓ Arithmetic Logic Unit (ALU)
- ✓ Memory Registers



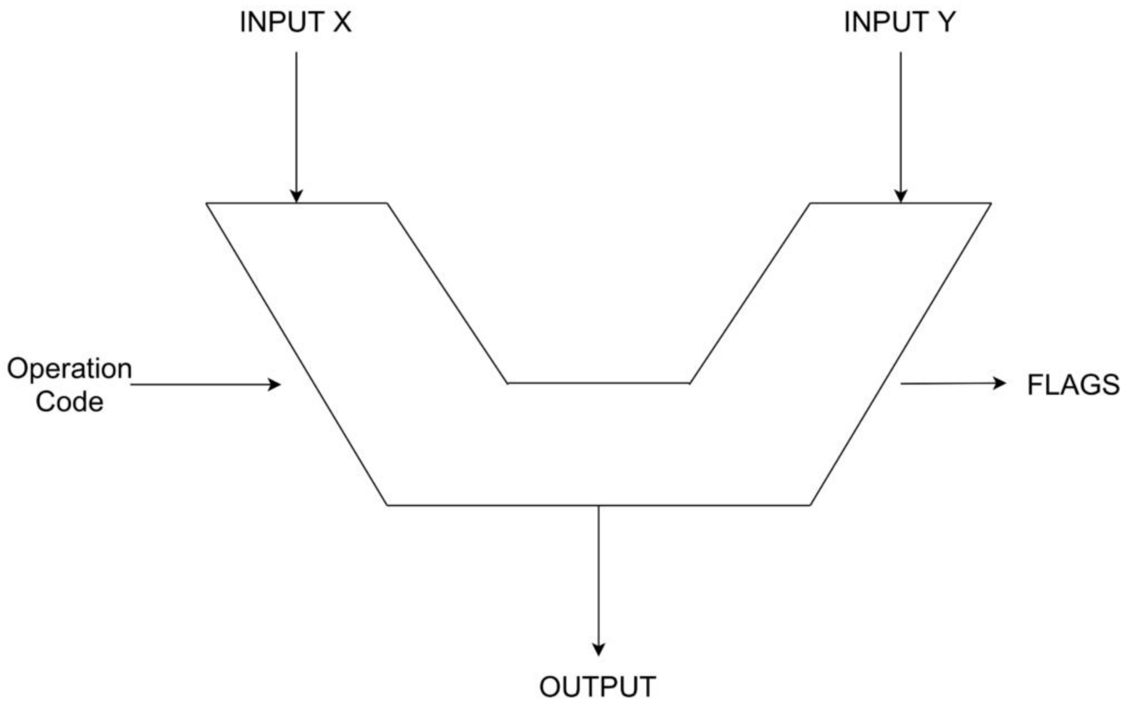
Control Unit (CU)

The Control unit manages the instructions given to the computer. It coordinates the activities of all the other units in the system by instructing rest of the components of the computer about how to carry out a program's instructions. It reads and interprets instructions from memory and transforms them in to series of signals to be executed or stored. It also directs the movement of these electronic signals between memory and ALU or between CPU and input/output devices.



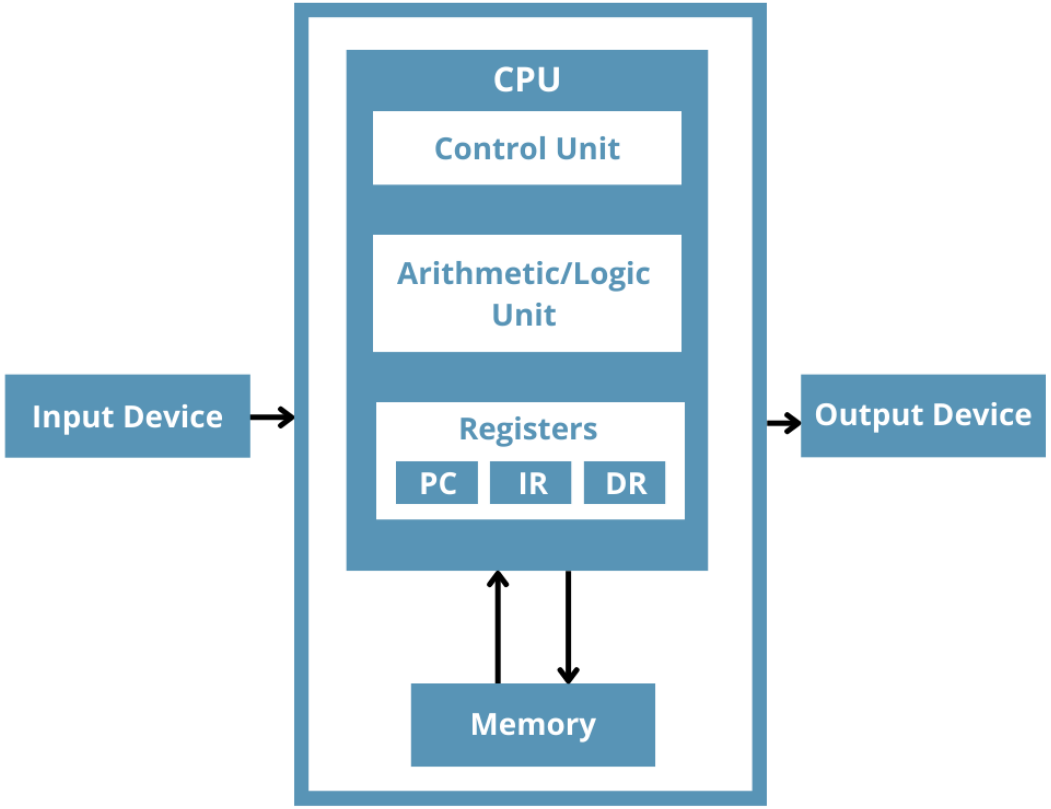
Arithmetic Logic Unit (ALU)

Arithmetic Logic Unit or ALU performs two types of operations - arithmetic and logical. Arithmetic operations are the fundamental mathematical operations consisting of addition, subtraction, multiplication and shifting operations. Logical operations consist of boolean comparisons such as AND, OR and NOT.

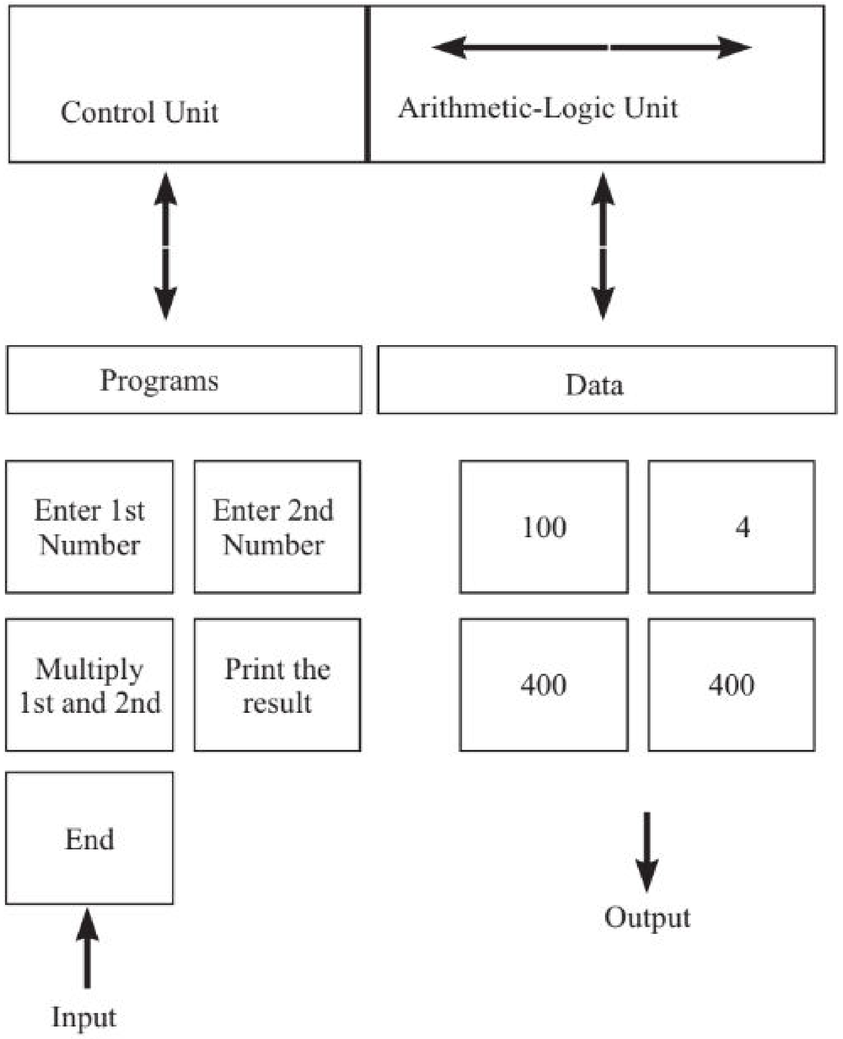


Memory Registers

A memory register is a sort of special storage area that holds the data and instructions temporarily during processing. Since this is internally located in the CPU, the processing time is very less. They often hold data for less than a millisecond. This high speed storage area make processing more efficient.



How the CPU and Memory work together?



That’s all for today….. See you in next class

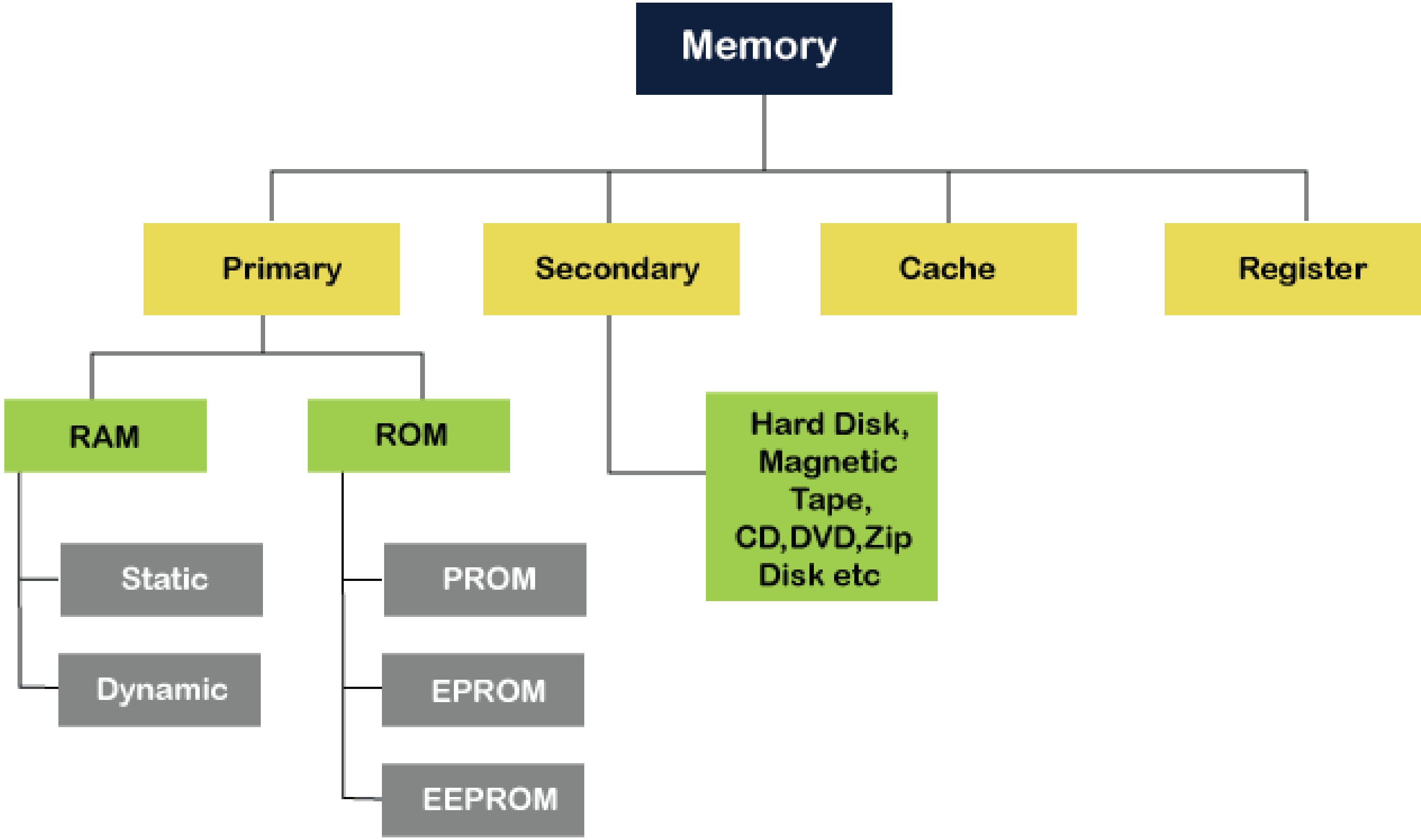
Computer Memory Unit

Lecture 3

Memory

Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored. It is used to store data and instructions. The memory is divided into large number of small parts called cells. Each location or cell has a unique address. For example, if the computer has 64k words, then this memory unit has 64 * 1024 = 65536 memory locations. The address of these locations varies from 0 to 65535.

Classification



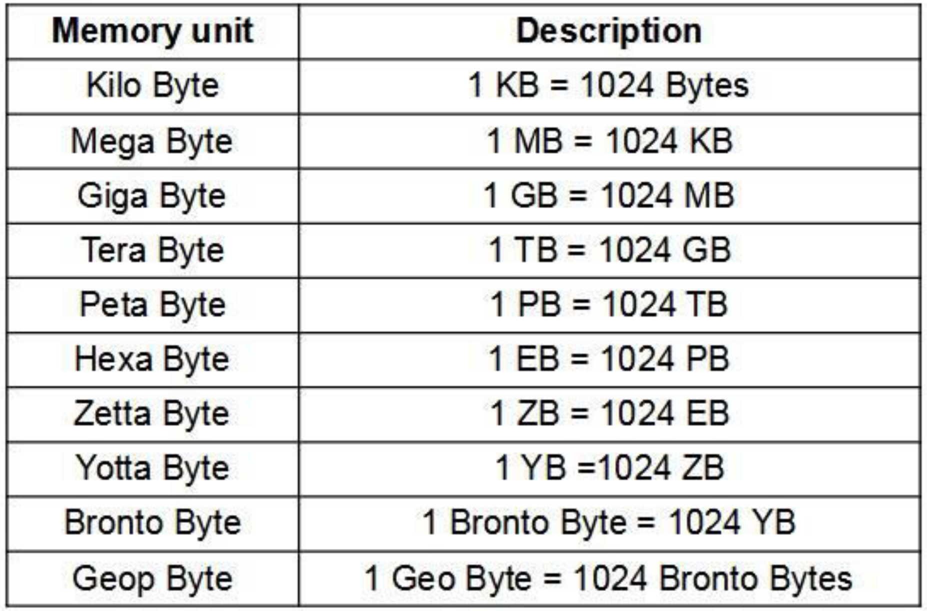
Home Task

Find out the differences between **primary memory** and **secondary memory**

Measuring Memory

The primary or internal storage unit is made up of several small storage locations called cells. Each of these cells can store a fixed number of bits called word length. Each cell has a unique number assigned to it called the address of the cell and it is used to identify the cells.

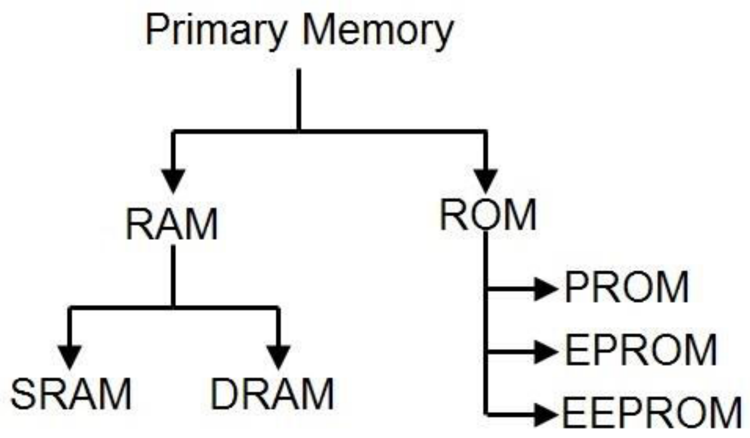
You know that data in computer is stored in the form of 0s and 1s. Each of these digits is known as a bit. A collection of 8 bits constitutes a byte. Each cell of memory contains one character or 1 byte of data. So the capacity is defined in terms of bytes or words. However higher units of memory are Kilobytes, Megabytes, Gigabytes etc. 1 Kilobyte is equal to 1024 bytes. Thus 64 Kilobyte (KB) memory is capable of storing 64 × 1024 = 32,768 bytes.



Memory unit	Description
Kilo Byte	1 KB = 1024 Bytes
Mega Byte	1 MB = 1024 KB
Giga Byte	1 GB = 1024 MB
Tera Byte	1 TB = 1024 GB
Peta Byte	1 PB = 1024 TB
Hexa Byte	1 EB = 1024 PB
Zetta Byte	1 ZB = 1024 EB
Yotta Byte	1 YB =1024 ZB
Bronto Byte	1 Bronto Byte = 1024 YB
Geop Byte	1 Geo Byte = 1024 Bronto Bytes

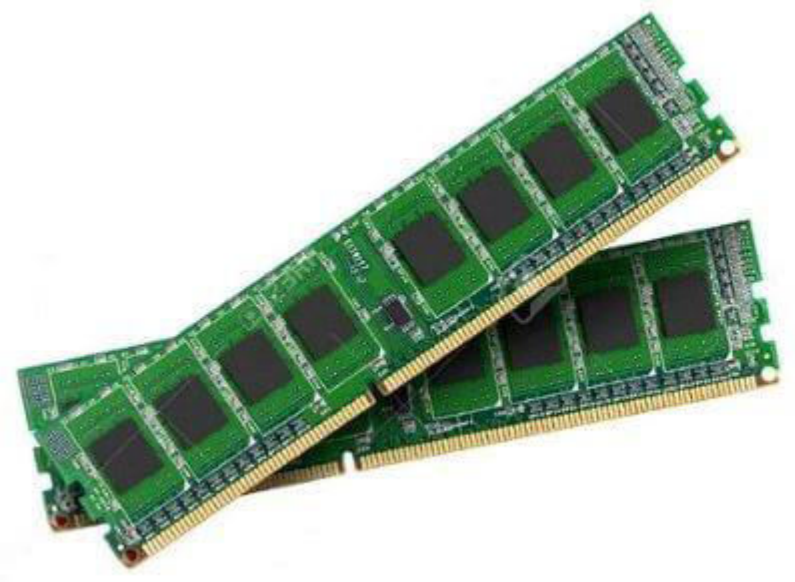
Primary Memory

Primary memory is the memory that is accessed by the processor directly. It is also known as main memory or internal memory. It helps in executing applications that are temporarily stored in a specific memory location. Primary memory is of two types – **RAM and ROM**.



Random Access Memory (RAM)

Random Access Memory (RAM) is the type of memory in which it is possible to randomly select and use any location of the memory directly to store and retrieve data. It is also called as read/write memory. Since it is volatile, the data from RAM is lost as soon as the power to the computer is switched off.



Random Access Memory (RAM)

Static RAM

- SRAM is faster compared to DRAM.
- It does not have a refreshing unit.
- These are expensive.
- In this bit are stored in voltage form.
- SRAM has higher data transfer rate
- SRAM is used in high performance applications

Dynamic RAM

- DRAM provides slow access speeds.
- It has a refreshing unit.
- These are cheaper.
- In this bit is stored in the form of electric energy.
- DRAM has lower data transfer rate
- DRAM is used in general purpose applications

Read Only Memory (ROM)

This is another type of primary memory from which data can only be read. We cannot write or modify data once written on to the ROM. Also this type of primary memory is not volatile. The storage of program and data in the ROM is permanent. The ROM stores some standard processing programs supplied by the manufacturers to operate our computer. The Basic Input Output System (BIOS) is stored in the ROM. It examines and initializes the start up process of the computer and also checks various peripheral devices attached to the PC when the computer is turned ON.

Read Only Memory (ROM)

Programmable Read Only Memory (PROM):

it is not possible to modify or erase programs stored in ROM, but it is possible for you to store your program in PROM chip. Once the programs are written it cannot be changed. Also the program is not lost even if power is switched off.

Erasable Programmable Read Only Memory (EPROM) :

This type of ROM overcomes the problem of PROM and ROM. EPROM chip can be programmed time and again by erasing the information stored earlier in it. Information stored in EPROM can be erased by exposing it to ultraviolet light. This memory can be reprogrammed using a special programming facility.

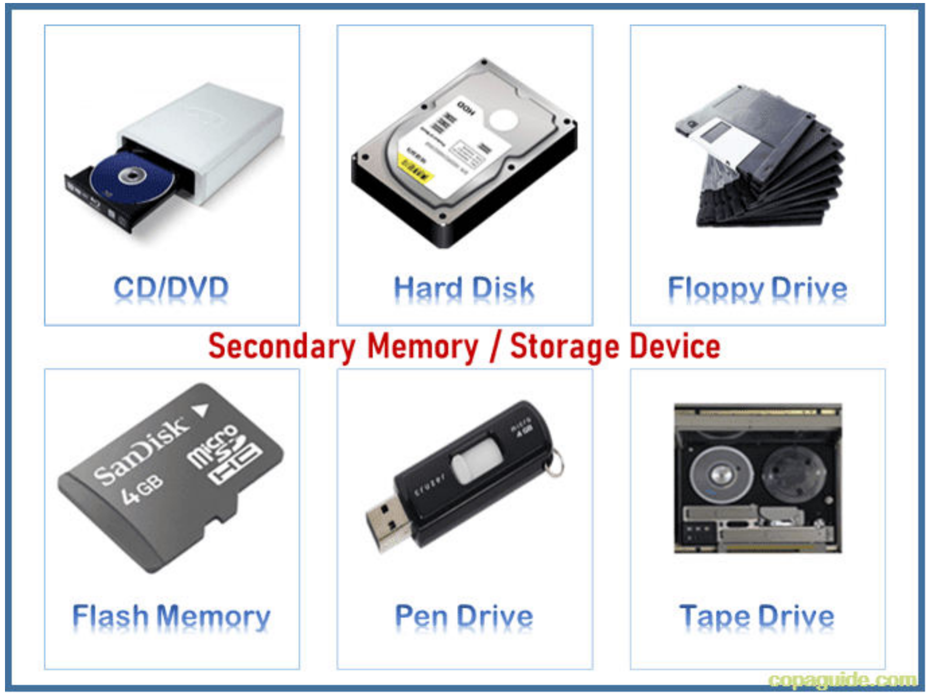
Electrically Erasable Programmable Read Only Memory(EEPROM):

The only difference is that unlike EPROM, electrical signals are used to erase the contents of EEPROM. Also, this type of ROM need not be completely erased. Partial modification of ROM is possible.

Secondary Memory

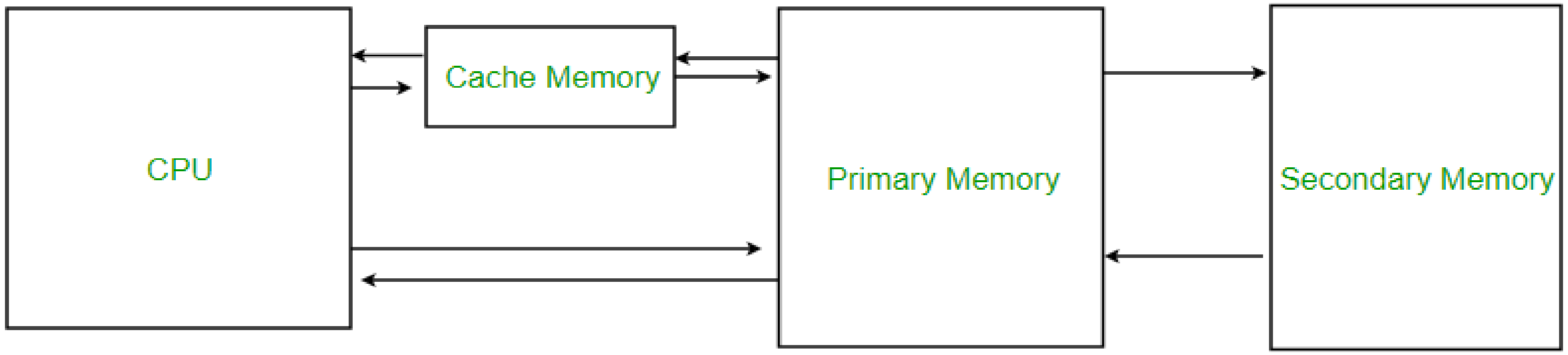
This type of memory is also known as external memory or non-volatile. It is slower than the main memory. These are used for storing data/information permanently. CPU directly does not access these memories, instead they are accessed via input-output routines. The contents of secondary memories are first transferred to the main memory, and then the CPU can access it. For example, disk, CD-ROM, DVD, etc.

Find out more info about **Magnetic Tape, Magnetic Disk, Floopy Disk, HDD.**



Cache Memory

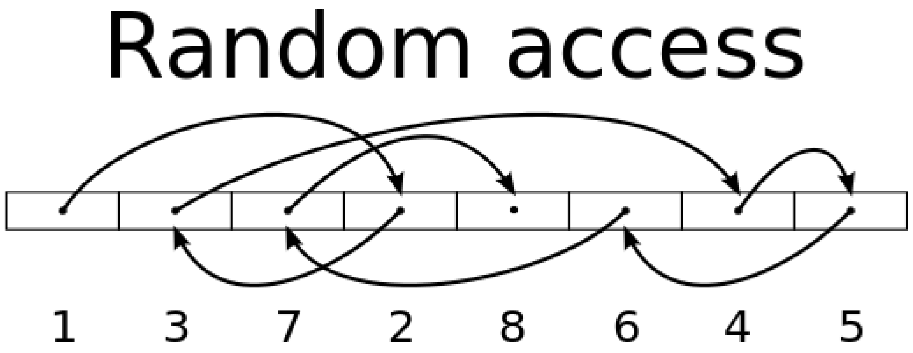
To increase the performance of CPU, a small memory chip is attached between CPU and main memory whose access time is very close to the processing speed of CPU. This memory is called as Cache memory. It is used to store programs or data currently being executed or data that is being frequently used by the CPU. Fast access of these data and instructions increases the overall execution speed of the software. It is very expensive memory and so has to be used in a limited amount. Cache memory is also known as CPU memory. It is either integrated directly with the CPU chip or is placed separately on the motherboard. As the microprocessor processes data, it first looks in the cache memory. If the required data or instructions are found there, it does not have to spend more time in reading data from other storage devices that have slower access time. Hence the processing speed increases.



MEMORY ACCESSING MODES

Direct Access or Random Access:

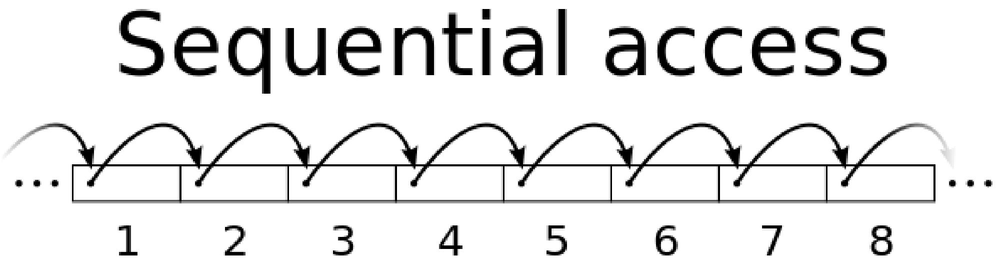
It is the type of accessing mode in which the value to be stored in a particular memory location is obtained directly. Since the data can be accessed in any order that is why this type of memory access is also known as random access of memory. This type of memory access is generally fast and more flexible. The type to be stored in memory is obtained directly by retrieving it from another memory location. This type of memory access is based on the principle that any piece of data can be returned in a constant time, regardless of its physical location and previous access. The web uses direct memory access mode.



MEMORY ACCESSING MODES

Sequential Access memory:

It is the type of memory in which the stored data is read in sequence. That means if you want to display the fourth record, the reading will start from the first record and then move to second, third and then fourth record. This type of memory access can be time consuming and hence slow, especially when the data to be read is towards the end. Sequential access devices are generally a form of magnetic storage. Hard disks, CD-ROMs and magnetic tapes use sequential access of memory.



That’s all for today….see you in next class!!

Number Systems

Learning Objectives

In this lecture you will learn about:

- ✓ Non-positional number system
- ✓ Positional number system
- ✓ Decimal number system
- ✓ Binary number system
- ✓ Octal number system
- ✓ Hexadecimal number system

- Convert a number’s base
  - ✓ Another base to decimal base
  - ✓ Decimal base to another base
  - ✓ Some base to another base
- Shortcut methods for converting
  - ✓ Binary to octal number
  - ✓ Octal to binary number
  - ✓ Binary to hexadecimal number
  - ✓ Hexadecimal to binary number
- Fractional numbers in binary number system

Number Systems

Two types of number systems are:

① Non-positional number systems

② Positional number systems

Non-positional Number Systems

- **Characteristics**
  - Use symbols such as I for 1, II for 2, III for 3, IIII for 4, IIIII for 5, etc.
  - Each symbol represents the same value regardless of its position in the number
  - The symbols are simply added to find out the value of a particular number
- **Difficulty**
  - It is difficult to perform arithmetic with such a number system

Positional Number Systems

- **Characteristics**
  - Use only a few symbols called digits
  - These symbols represent different values depending on the position they occupy in the number
- **The value of each digit is determined by**
  1. The digit itself
  2. The position of the digit in the number
  3. The base of the number system (**base** = total number of digits in the number system)
- The maximum value of a single digit is always equal to one less than the value of the base

Decimal Number System

- **Characteristics**
  - A positional number system
  - Has 10 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Hence, its base = 10
  - The maximum value of a single digit is 9 (one less than the value of the base)
  - Each position of a digit represents a specific power of the base (10)
  - We use this number system in our day-to-day life
- **Example**

$$\begin{aligned} 2586_{10} &= (2 \text{ x } 10^3) + (5 \text{ x } 10^2) + (8 \text{ x } 10^1) + (6 \text{ x } 10^0) \\ &= 2000 + 500 + 80 + 6 \end{aligned}$$

Binary Number System

Characteristics

- A positional number system
- Has only 2 symbols or digits (0 and 1). Hence its base = 2
- The maximum value of a single digit is 1 (one less than the value of the base)
- Each position of a digit represents a specific power of the base (2)
- This number system is used in computers

Example

$$10101_2 = (1 \text{ x } 2^4) + (0 \text{ x } 2^3) + (1 \text{ x } 2^2) + (0 \text{ x } 2^1) \text{ x } (1 \text{ x } 2^0)$$

$$= 16 + 0 + 4 + 0 + 1$$

$$= 21_{10}$$

Bit

- Bit stands for binary digit
- A bit in computer terminology means either a **0** or a **1**
- A binary number consisting of *n* bits is called an n-bit number

Representing Numbers in Different Number Systems

- In order to be specific about which number system we are referring to, it is a common practice to indicate the base as a subscript.
- Thus, we write: $10101_2 = 21_{10}$

Octal Number System

- **Characteristics**
  - A positional number system
  - Has total 8 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7).
  - Hence, its base = 8
  - The maximum value of a single digit is 7 (one less than the value of the base)
  - Each position of a digit represents a specific power of the base (8)
  - Since there are only 8 digits, 3 bits ($2^3 = 8$) are sufficient to represent any octal number in binary
- **Example**

$$\begin{aligned} 2057_8 &= (2 \times 8^3) + (0 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) \\ &= 1024 + 0 + 40 + 7 \\ &= 1071_{10} \end{aligned}$$

Hexadecimal Number System

- **Characteristics**
  - A positional number system
  - Has total 16 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Hence its base = 16
  - The symbols A, B, C, D, E and F represent the decimal values 10, 11, 12, 13, 14 and 15 respectively
  - The maximum value of a single digit is 15 (one less than the value of the base)
  - Each position of a digit represents a specific power of the base (16)
  - Since there are only 16 digits, 4 bits ($2^4 = 16$) are sufficient to represent any hexadecimal number in binary

- **Example**

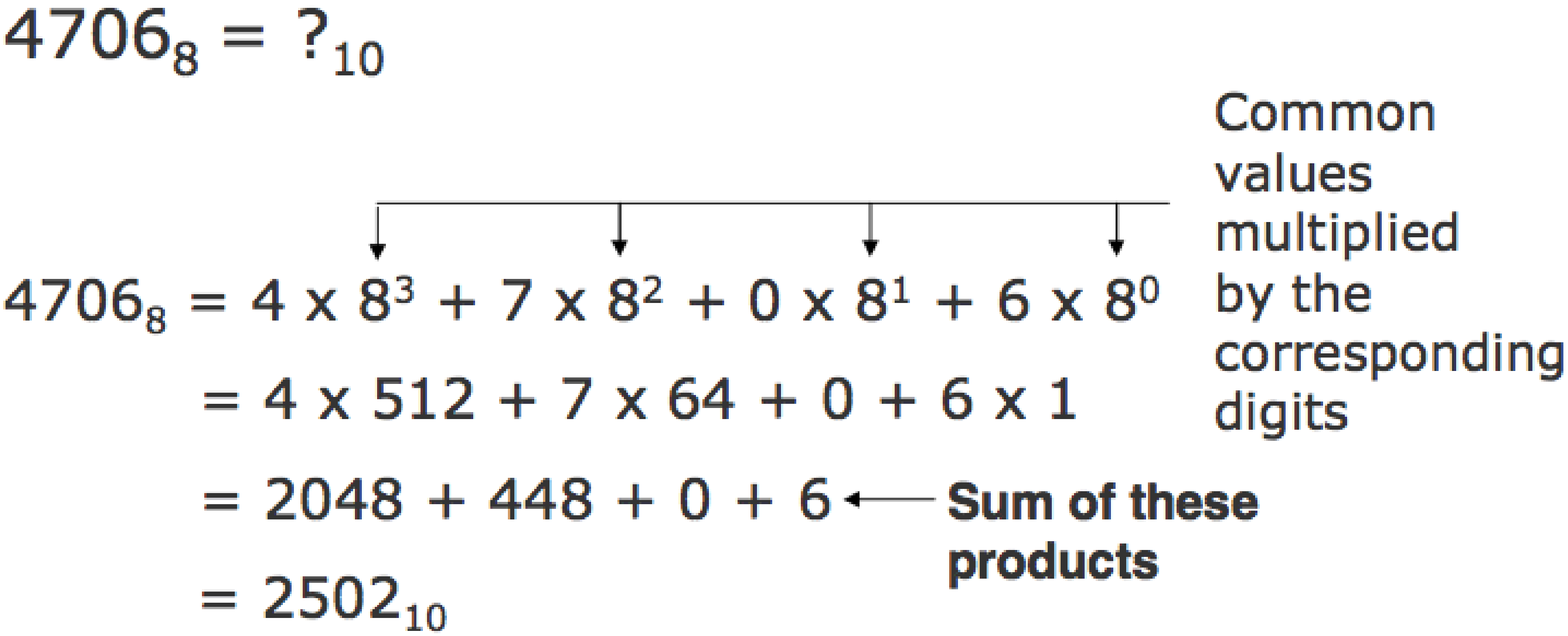
$$
\begin{aligned}
1AF_{16} &= (1 \text{ x } 16^2) + (A \text{ x } 16^1) + (F \text{ x } 16^0) \\
&= 1 \text{ x } 256 + 10 \text{ x } 16 + 15 \text{ x } 1 \\
&= 256 + 160 + 15 \\
&= 431_{10}
\end{aligned}
$$

Converting a Number of Another Base to a Decimal Number

- **Method**
  - Step 1: Determine the column (positional) value of each digit
  - Step 2: Multiply the obtained column values by the digits in the corresponding columns
  - Step 3: Calculate the sum of these products

Converting a Number of Another Base to a Decimal Number

- Example



$4706_8 = ?_{10}$

Common values multiplied by the corresponding digits

$$\begin{aligned} 4706_8 &= 4 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 6 \times 8^0 \\ &= 4 \times 512 + 7 \times 64 + 0 + 6 \times 1 \\ &= 2048 + 448 + 0 + 6 \\ &= 2502_{10} \end{aligned}$$

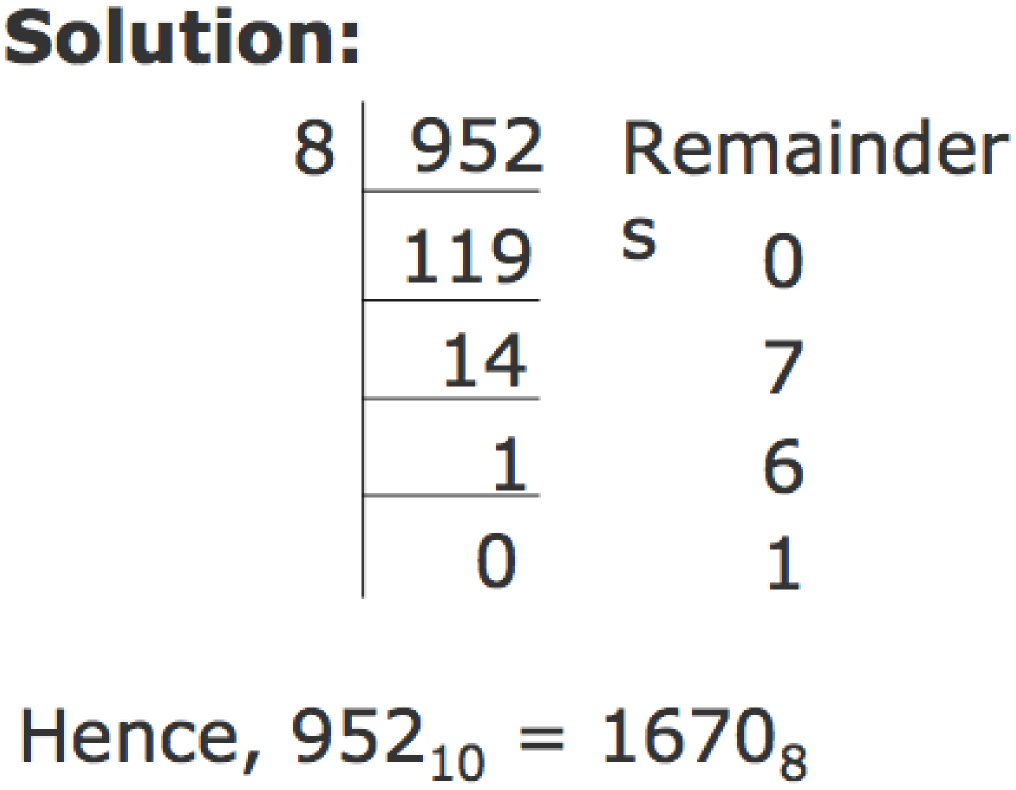
Sum of these products

Converting a Decimal Number to a Number of Another Base

- **Division-Remainder Method**
  - **Step 1:** Divide the decimal number to be converted by the value of the new base
  - **Step 2:** Record the remainder from Step 1 as the rightmost digit (least significant digit) of the new base number
  - **Step 3:** Divide the quotient of the previous divide by the new base
  - **Step 4:** Record the remainder from Step 3 as the next digit (to the left) of the new base number
  - Repeat Steps 3 and 4, recording remainders from right to left, until the quotient becomes zero in Step 3
- Note that the last remainder thus obtained will be the most significant digit (MSD) of the new base number

Converting a Decimal Number to a Number of Another Base

- **Example:** $952_{10} = ?_8$



Solution:

8	952	Remainder	
	119	s	0
	14		7
	1		6
	0		1

Hence, $952_{10} = 1670_8$

Converting a Number of Some Base to a Number of Another Base

- **Method**
  - **Step 1:** Convert the original number to a decimal number (base 10)
  - **Step 2:** Convert the decimal number so obtained to the new base number

Converting a Number of Some Base to a Number of Another Base

- Example:

$545_6 = ?_4$

Solution:

Step 1: Convert from base 6 to base 10

$$\begin{aligned} 545_6 &= 5 \text{ x } 6^2 + 4 \text{ x } 6^1 + 5 \text{ x } 6^0 \\ &= 5 \text{ x } 36 + 4 \text{ x } 6 + 5 \text{ x } 1 \\ &= 180 + 24 + 5 \\ &= 209_{10} \end{aligned}$$

Converting a Number of Some Base to a Number of Another Base

Step 2: Convert 209_{10} to base 4

4	209	Remainders
	52	1
	13	0
	3	1
	0	3

Hence, $209_{10} = 3101_4$

So, $545_6 = 209_{10} = 3101_4$

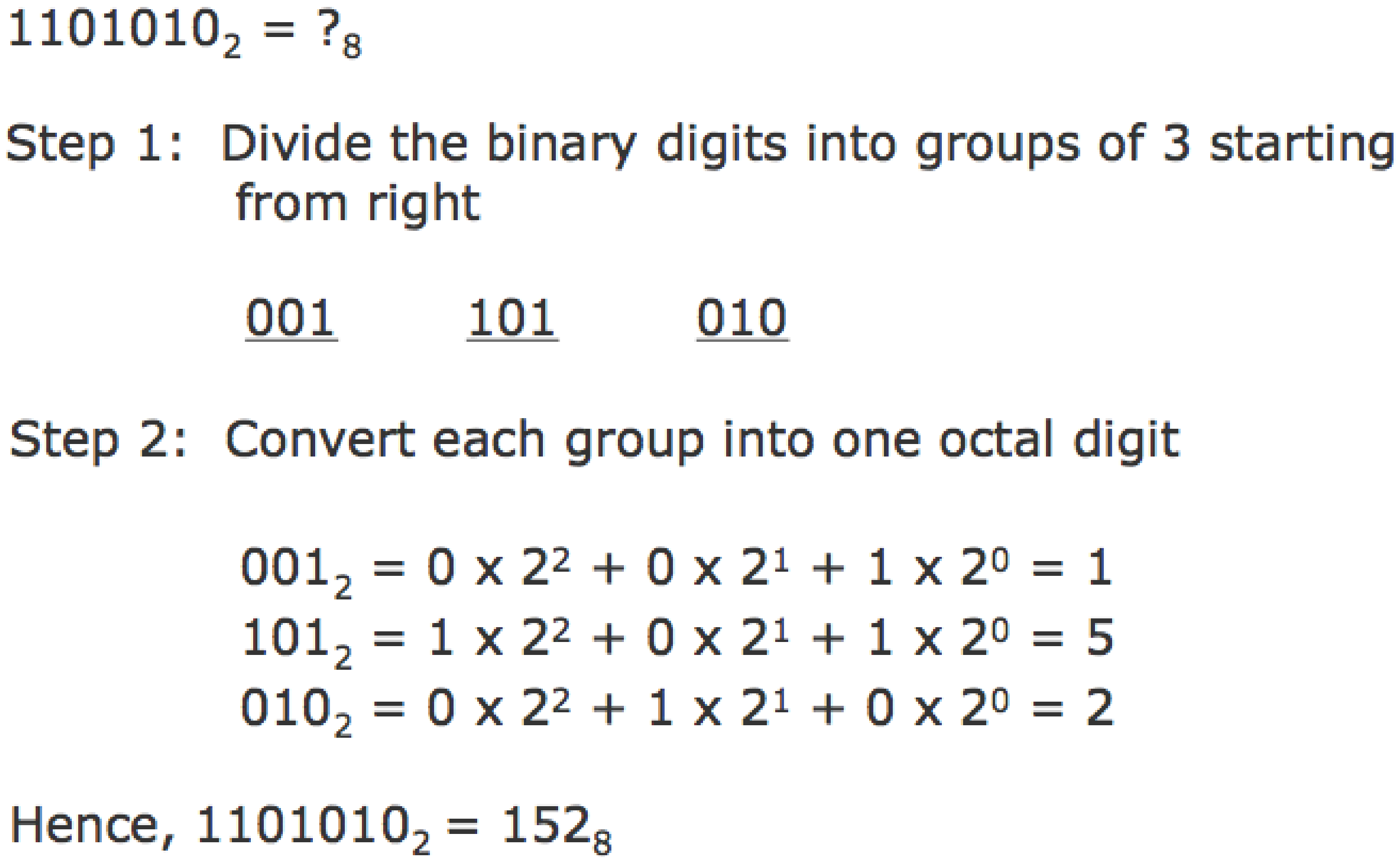
Thus, $545_6 = 3101_4$

Shortcut Method for Converting a Binary Number to its Equivalent Octal Number

- **Method**
  - Step 1: Divide the digits into groups of three starting from the right
  - Step 2: Convert each group of three binary digits to one octal digit using the method of binary to decimal conversion

Shortcut Method for Converting a Binary Number to its Equivalent Octal Number

- **Example**:



$1101010_2 = ?_8$

Step 1: Divide the binary digits into groups of 3 starting from right

$\underline{001}$ $\underline{101}$ $\underline{010}$

Step 2: Convert each group into one octal digit

$001_2 = 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1$

$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$

$010_2 = 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2$

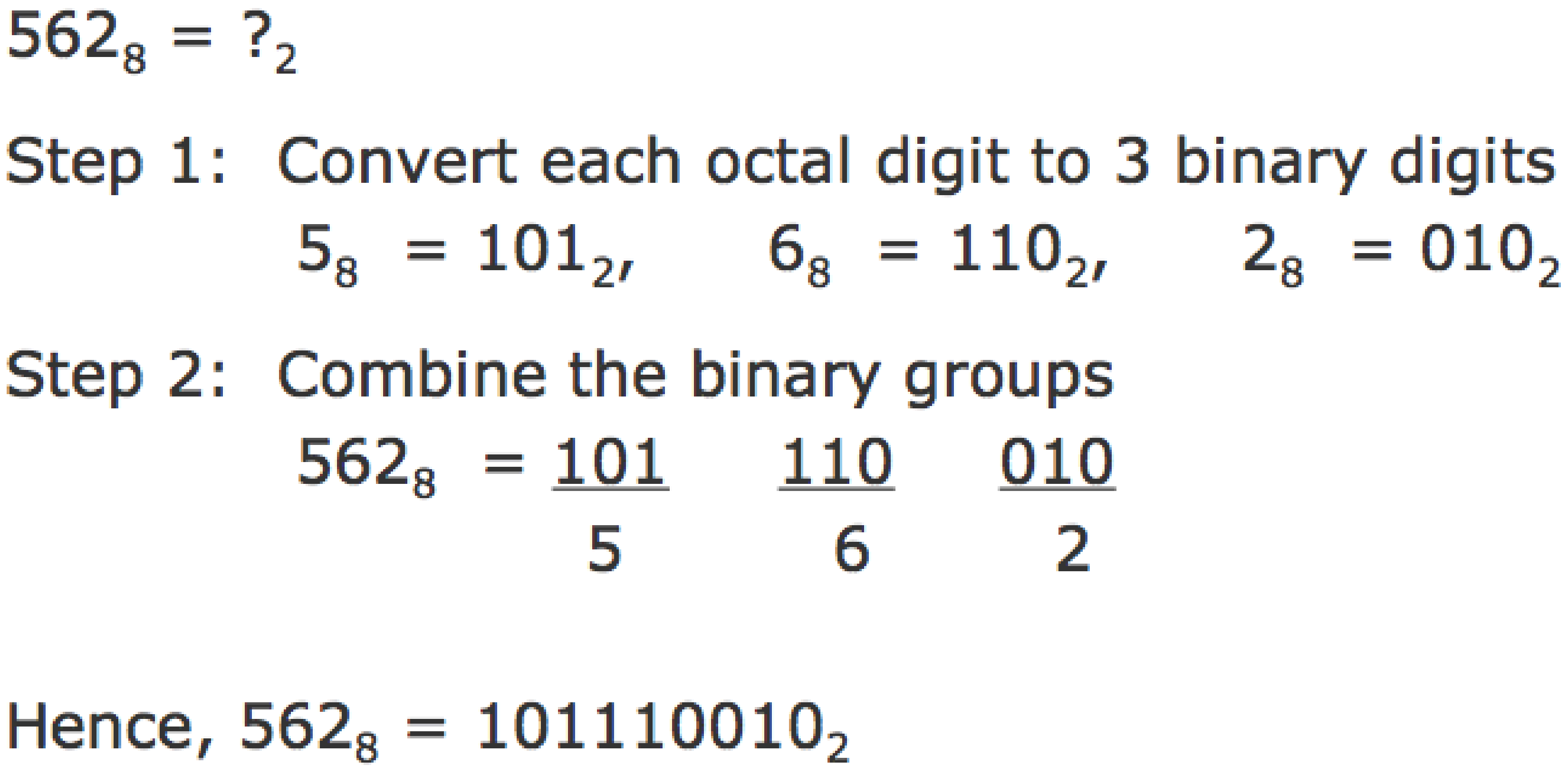
Hence, $1101010_2 = 152_8$

Shortcut Method for Converting an Octal Number to Its Equivalent Binary Number

- **Method**
  - Step 1: Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal for this conversion)
  - Step 2: Combine all the resulting binary groups (of 3 digits each) into a single binary number

Shortcut Method for Converting an Octal Number to Its Equivalent Binary Number

- **Example:**



$562_8 = ?_2$

Step 1: Convert each octal digit to 3 binary digits

$5_8 = 101_2, \quad 6_8 = 110_2, \quad 2_8 = 010_2$

Step 2: Combine the binary groups

$$562_8 = \underset{5}{\underline{101}} \quad \underset{6}{\underline{110}} \quad \underset{2}{\underline{010}}$$

Hence, $562_8 = 101110010_2$

Shortcut Method for Converting a Binary Number to its Equivalent Hexadecimal Number

- **Method**

  - Step 1: Divide the binary digits into groups of four starting from the right
  - Step 2: Combine each group of four binary digits to one hexadecimal digit

Shortcut Method for Converting a Binary Number to its Equivalent Hexadecimal Number

- Example:

$111101_2 = ?_{16}$

Step 1: Divide the binary digits into groups of four starting from the right

$\underline{0011}$ $\underline{1101}$

Step 2: Convert each group into a hexadecimal digit

$0011_2 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3_{10} = 3_{16}$

$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 3_{10} = D_{16}$

Hence, $111101_2 = 3D_{16}$

Shortcut Method for Converting an Octal Number to Its Equivalent Binary Number

- **Method**
  - Step 1: Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal for this conversion)
  - Step 2: Combine all the resulting binary groups (of 3 digits each) into a single binary number

Shortcut Method for Converting an Octal Number to Its Equivalent Binary Number

- **Example:**

$562_8 = ?_2$

Step 1: Convert each octal digit to 3 binary digits

$5_8 = 101_2, \quad 6_8 = 110_2, \quad 2_8 = 010_2$

Step 2: Combine the binary groups

$$562_8 = \underset{5}{\underline{101}} \quad \underset{6}{\underline{110}} \quad \underset{2}{\underline{010}}$$

Hence, $562_8 = 101110010_2$

Shortcut Method for Converting a Binary Number to its Equivalent Hexadecimal Number

- **Example:**

$111101_2 = ?_{16}$

Step 1: Divide the binary digits into groups of four starting from the right

0011 1101

Step 2: Convert each group into a hexadecimal digit

$0011_2 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3_{10} = 3_{16}$

$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 3_{10} = D_{16}$

Hence, $111101_2 = 3D_{16}$

Shortcut Method for Converting a Hexadecimal Number to its Equivalent Binary Number

- **Method**
  - Step 1: Convert the decimal equivalent of each hexadecimal digit to a 4 digit binary number
  - Step 2: Combine all the resulting binary groups (of 4 digits each) in a single binary number

Shortcut Method for Converting a Hexadecimal Number to its Equivalent Binary Number

- **Example:** $2AB_{16} = ?_2$

Step 1: Convert each hexadecimal digit to a 4 digit binary number

$$2_{16} = 210 = 0010_2$$

$$A_{16} = 1010 = 1010_2$$

$$B_{16} = 1110 = 1011_2$$

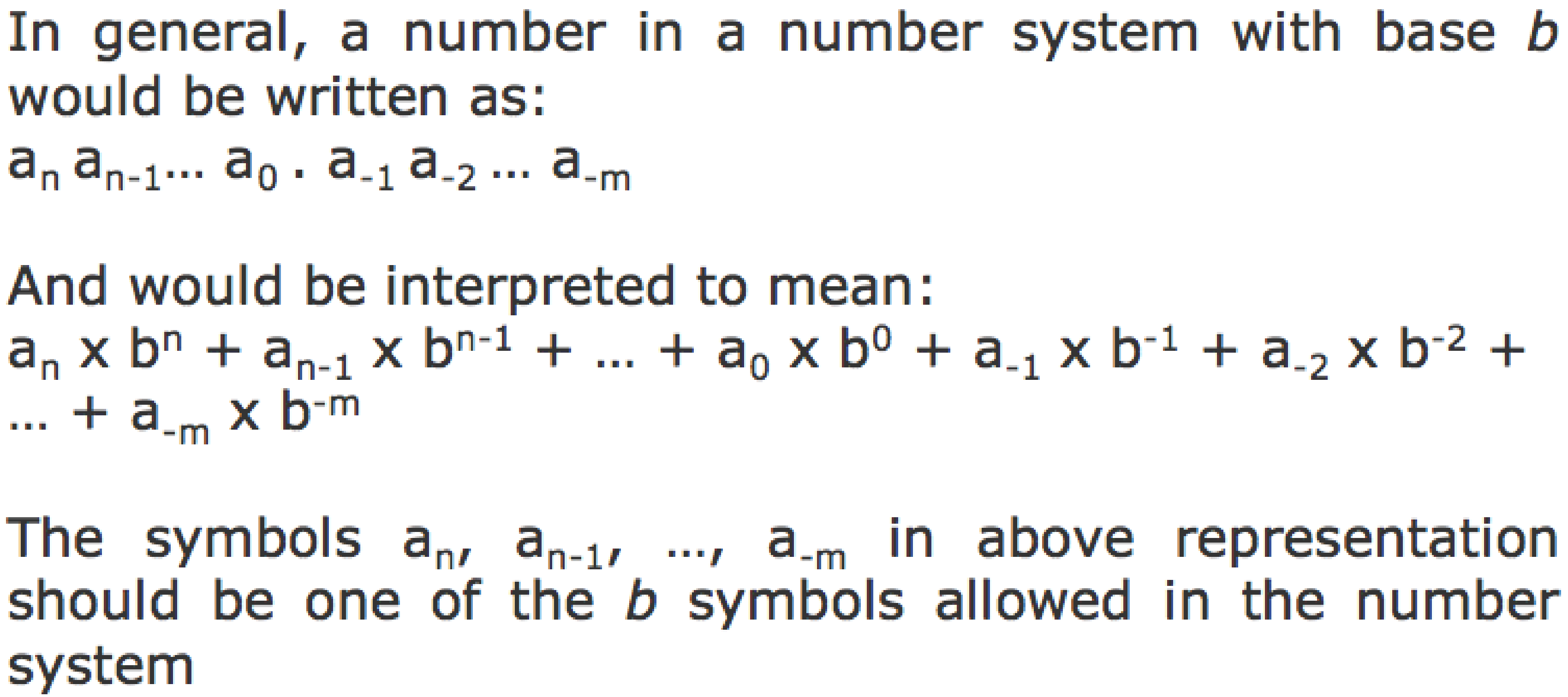
Step 2: Combine the binary groups

$$2AB_{16} = \underset{2}{\underline{0010}} \quad \underset{A}{\underline{1010}} \quad \underset{B}{\underline{1011}}$$

Hence, $2AB_{16} = 001010101011_2$

Fractional Numbers

- **Fractional numbers** are formed same way as decimal number system



In general, a number in a number system with base *b* would be written as:
$a_n\, a_{n-1} \dots\, a_0\, .\, a_{-1}\, a_{-2} \dots\, a_{-m}$

And would be interpreted to mean:
$a_n \times b^n + a_{n-1} \times b^{n-1} + \dots + a_0 \times b^0 + a_{-1} \times b^{-1} + a_{-2} \times b^{-2} + \dots + a_{-m} \times b^{-m}$

The symbols a_n, a_{n-1}, …, a_{-m} in above representation should be one of the *b* symbols allowed in the number system

Formation of Fractional Numbers in Binary Number System

Binary Point ↓

Position	4	3	2	1	0	.	-1	-2	-3	-4
Position Value	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}
Quantity Represented	16	8	4	2	1		$^1/_2$	$^1/_4$	$^1/_8$	$^1/_{16}$

- **Example:**

$$
\begin{aligned}
110.101_2 &= 1 \text{ x } 2^2 + 1 \text{ x } 2^1 + 0 \text{ x } 2^0 + 1 \text{ x } 2^{-1} + 0 \text{ x } 2^{-2} + 1 \text{ x } 2^{-3} \\
&= 4 + 2 + 0 + 0.5 + 0 + 0.125 \\
&= 6.625_{10}
\end{aligned}
$$

Formation of Fractional Numbers in Octal Number System

Octal Point
↓

Position	3	2	1	0	.	-1	-2	-3
Position Value	8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
Quantity Represented	512	64	8	1		$^1/_8$	$^1/_{64}$	$^1/_{512}$

- **Example**:

$$
\begin{aligned}
127.54_8 &= 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 5 \times 8^{-1} + 4 \times 8^{-2} \\
&= 64 + 16 + 7 + {}^5/_8 + {}^4/_{64} \\
&= 87 + 0.625 + 0.0625 \\
&= 87.6875_{10}
\end{aligned}
$$

Key Words/Phrases

- Base
- Binary number system
- Binary point
- Bit
- Decimal number system
- Division-Remainder technique
- Fractional numbers
- Hexadecimal number system

Least Significant Digit (LSD)

Memory dump

Most Significant Digit (MSD)

Non-positional number system

Number system

Octal number system

Positional number system

Logic Gates, Logic Circuits & Boolean Algebra

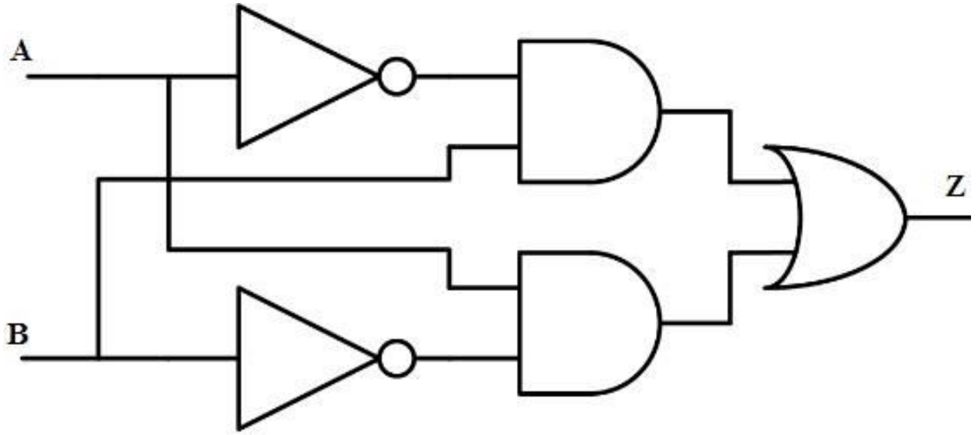
Lecture 9

Logic Gates

Logic gates are electronic circuits that operate on one or more input signals to produce standard output signal.

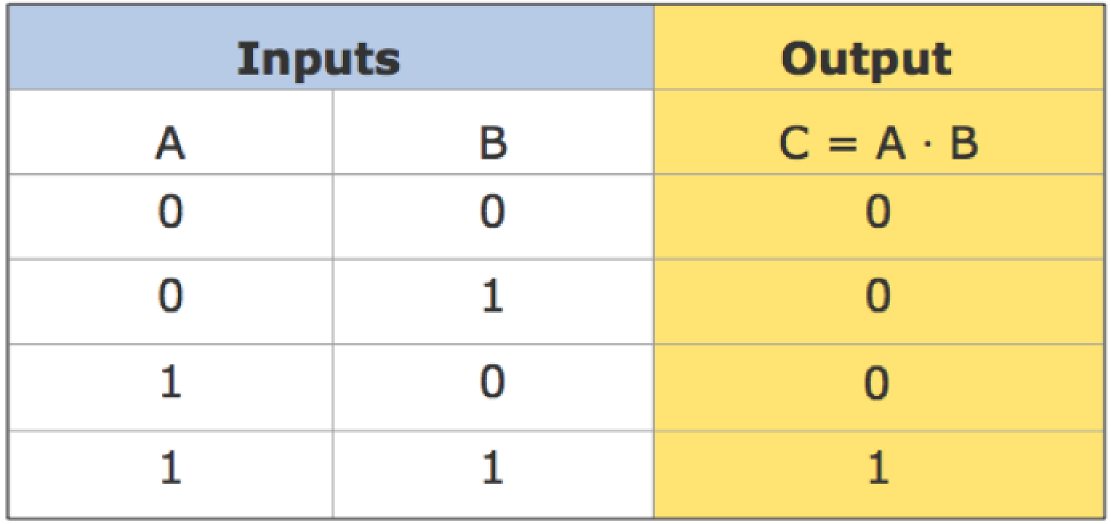
Some of the most basic and useful logic gates are -

- AND,
- OR,
- NOT,
- NAND and
- NOR gate

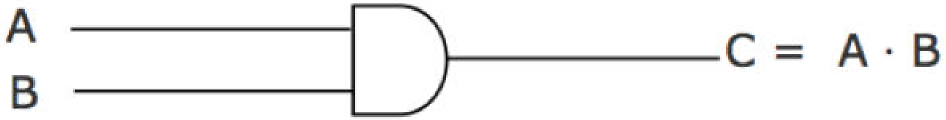


AND Gate

Generates an output signal of 1 only if all input signals are also 1

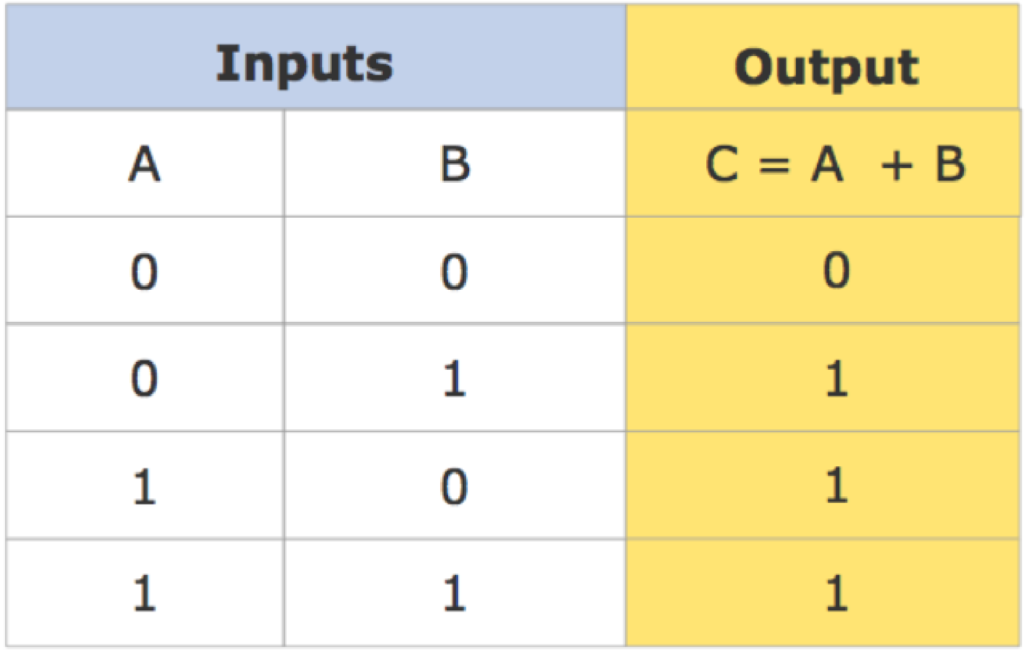


Inputs		Output
A	B	C = A · B
0	0	0
0	1	0
1	0	0
1	1	1

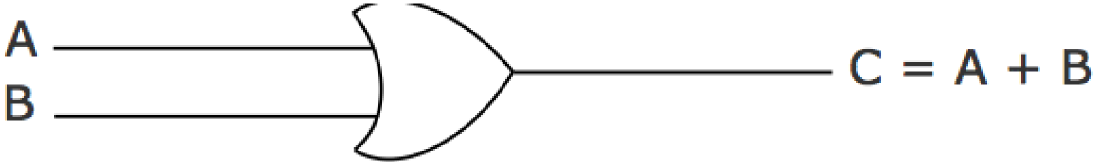


OR Gate

Generates an output signal of 1 if at least one of the input signals is also 1

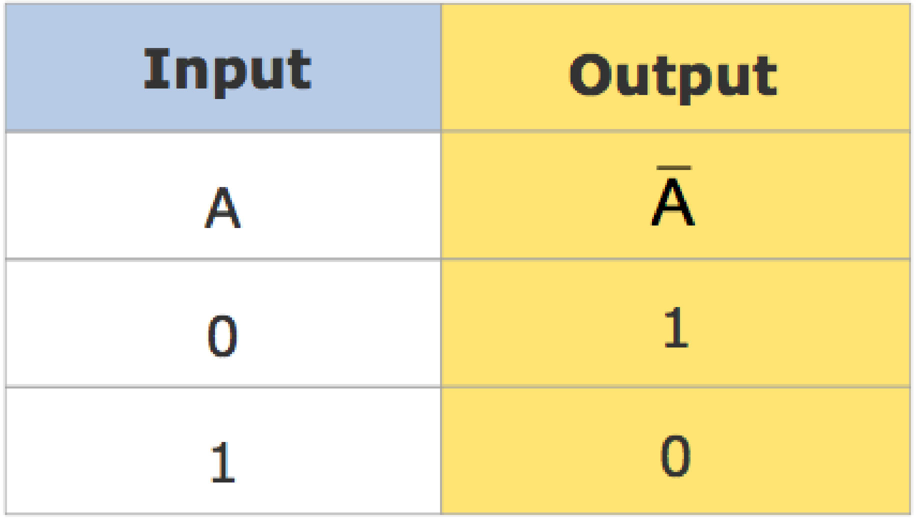


Inputs		Output
A	B	C = A + B
0	0	0
0	1	1
1	0	1
1	1	1

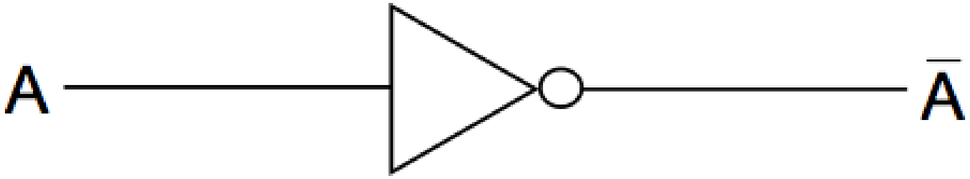


NOT Gate

Generates an output signal, which is the reverse of the input signal



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

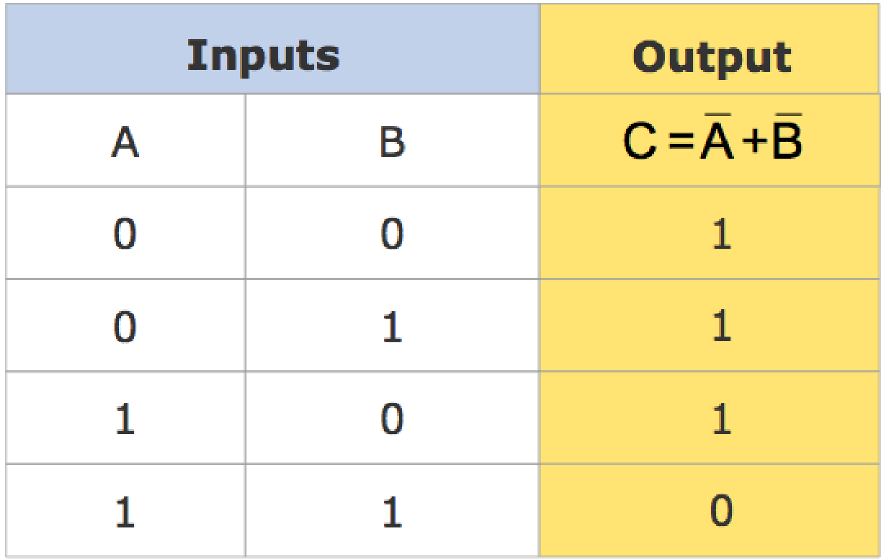


NAND Gate

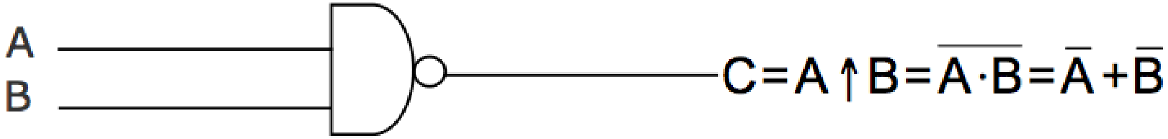
Complemented AND gate. Generates an output signal of:

① 1 if any one of the inputs is a 0

② 0 when all the inputs are 1



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



$C=A\uparrow B=\overline{A\cdot B}=\overline{A}+\overline{B}$

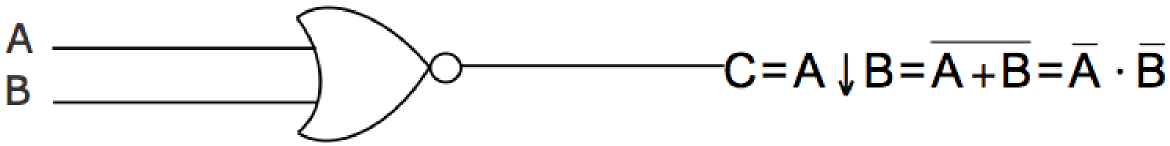
NOR Gate

Complemented OR gate. Generates an output signal of:

① 1 only when all inputs are 0

② 0 if any one of inputs is a 1

Inputs		Output
A	B	$C = \overline{A} \cdot \overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0



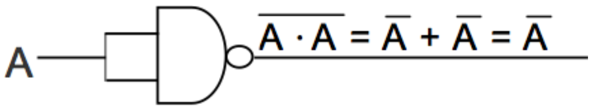
$C = A \downarrow B = \overline{A+B} = \overline{A} \cdot \overline{B}$

Universal NAND Gate

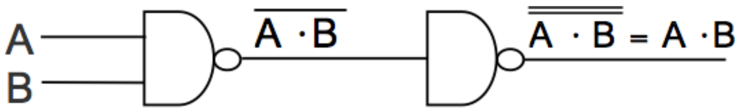
NAND gate is an universal gate, it is alone sufficient to implement any Boolean expression

To understand this, consider:

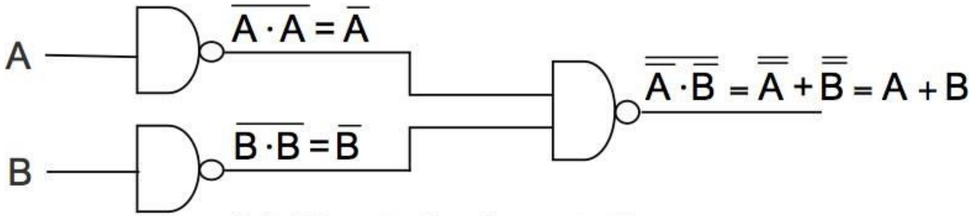
Basic logic gates (AND, OR, and NOT) are logically complete. Sufficient to show that AND, OR, and NOT gates can be implemented with NAND gates


A
$\overline{A \cdot A} = \overline{A} + \overline{A} = \overline{A}$

(a) NOT gate implementation.



(b) AND gate implementation.


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

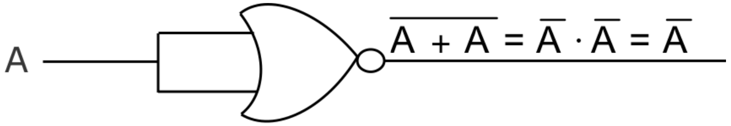
(c) OR gate implementation.

Universal NOR Gate

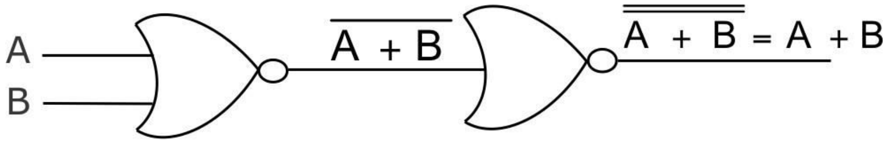
NOR gate is an universal gate, it is alone sufficient to implement any Boolean expression

To understand this, consider:

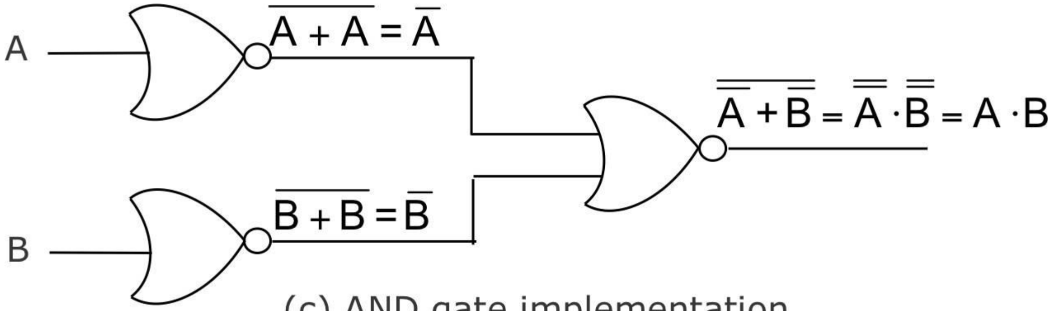
Basic logic gates (AND, OR, and NOT) are logically complete. Sufficient to show that AND, OR, and NOT gates can be implemented with NOR gates


A
$\overline{A+A} = \overline{A}\cdot\overline{A} = \overline{A}$

(a) NOT gate implementation



(b) OR gate implementation.


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

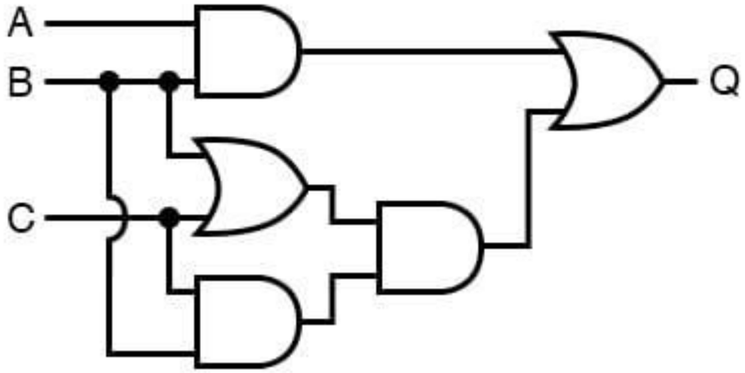
(c) AND gate implementation.

Logic Circuits

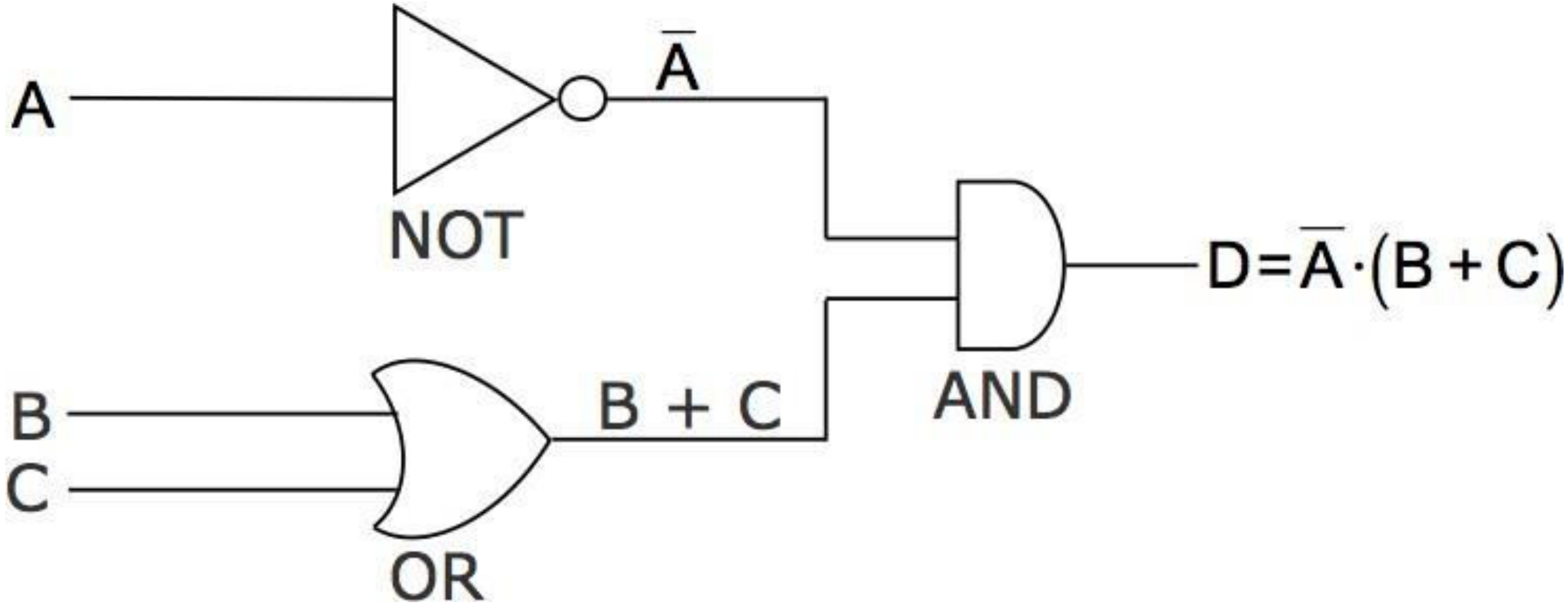
When logic gates are interconnected to form a gating / logic network, it is known as a combinational logic circuit

The Boolean algebra expression for a given logic circuit can be derived by systematically progressing from input to output on the gates

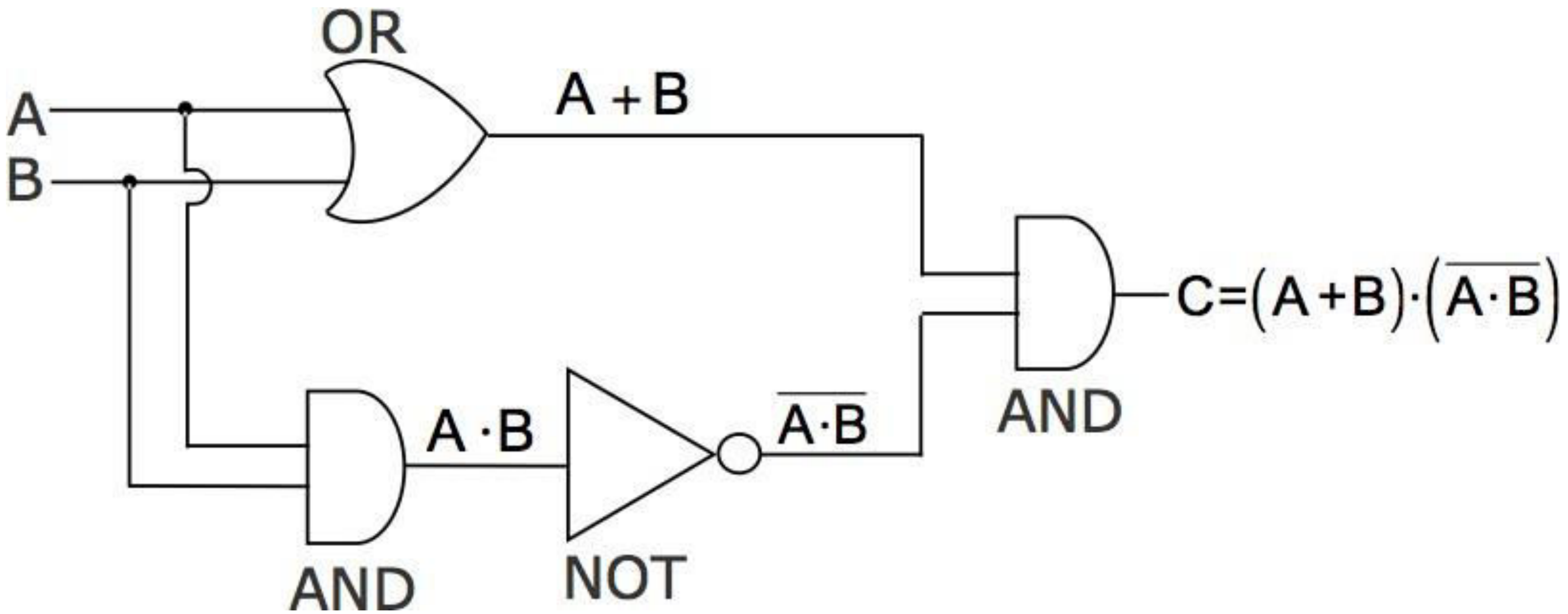
The three logic gates (AND, OR, and NOT) are logically complete because any Boolean expression can be realized as a logic circuit using only these three gates



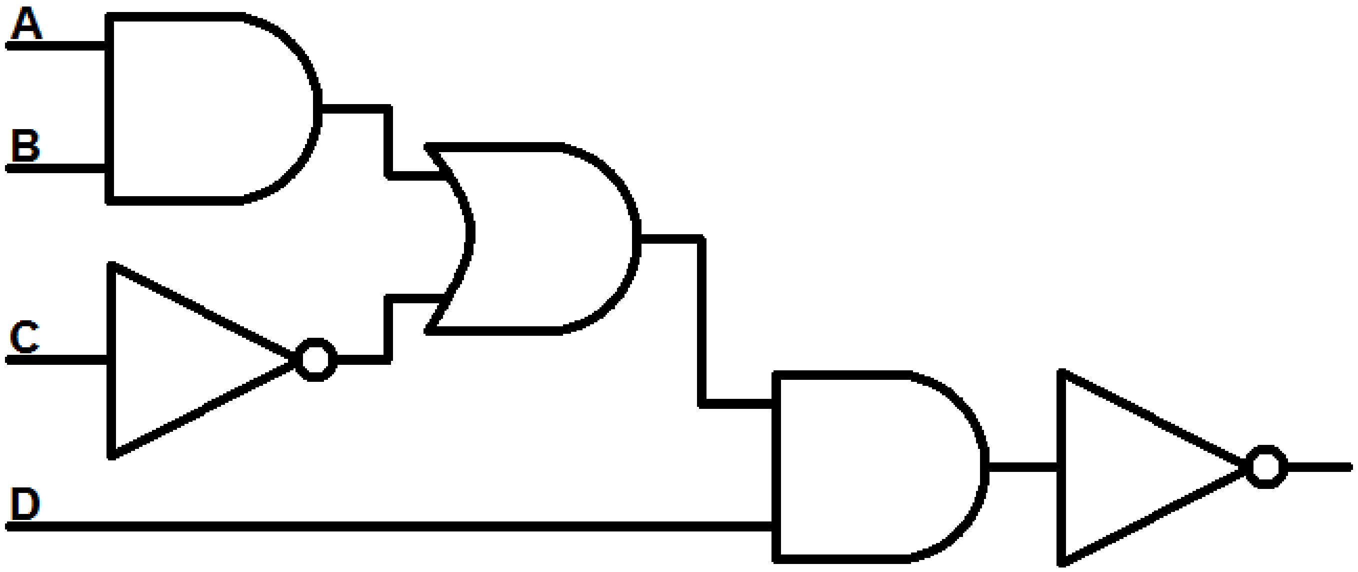
Finding Boolean Expression of a Logic Circuit (Example 1)



Finding Boolean Expression of a Logic Circuit (Example 2)



Finding Boolean Expression of a Logic Circuit (Example 3)

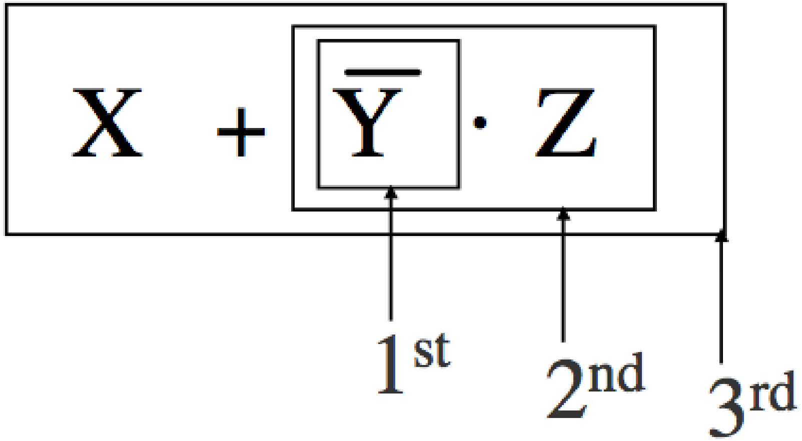


Boolean Algebra

- **Use of Binary Digit:** Boolean equations can have either of two possible values, 0 and 1
- **Logical Addition:** Symbol '+', also known as 'OR' operator, used for logical addition. Follows law of binary addition
- **Logical Multiplication:** Symbol '. ', also known as 'AND' operator, used for logical multiplication. Follows law of binary multiplication
- **Complementation:** Symbol '-', also known as 'NOT' operator, used for complementation. Follows law of binary compliment

Operator Precedence

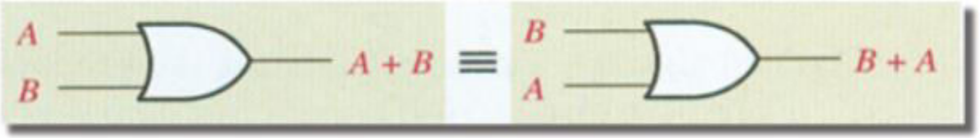
- Each operator has a precedence level
- Higher the operator's precedence level, earlier it is evaluated
- Expression is scanned from **left to right**
- First, expressions enclosed within **parentheses are evaluated**
- Then, all complement **(NOT)** operations are performed
- Then, all '·' **(AND)** operations are performed
- Finally, all '+' **(OR)** operations are performed



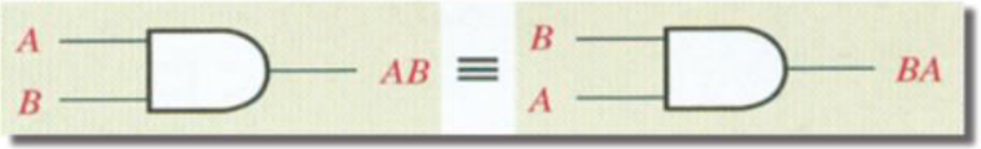
Laws of Boolean Algebra

- Commutative Laws

A + B = B + A

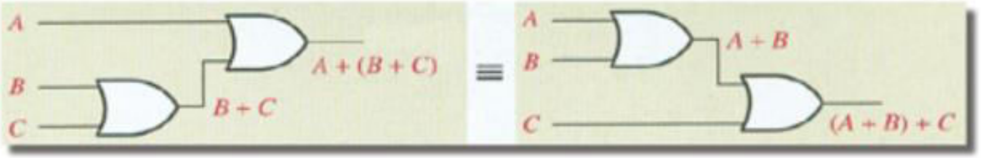


A • B = B • A

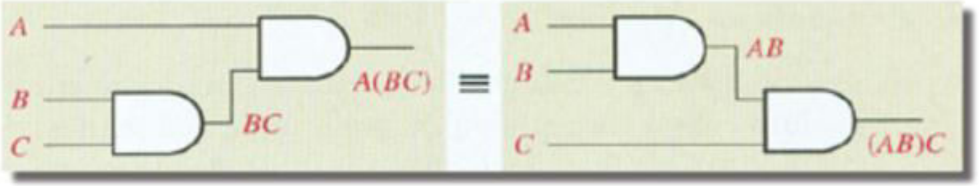


- Associative Laws

A + (B + C) = (A + B) + C

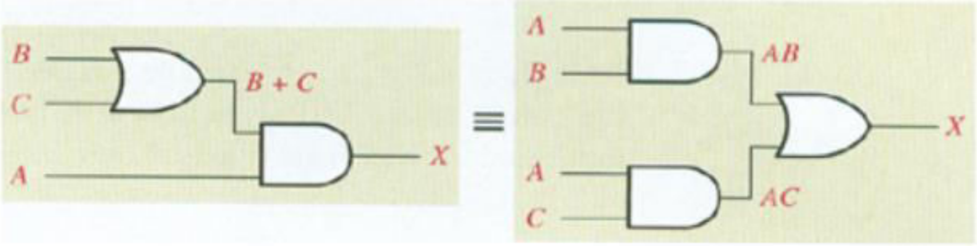


A • (B • C) = (A • B) • C

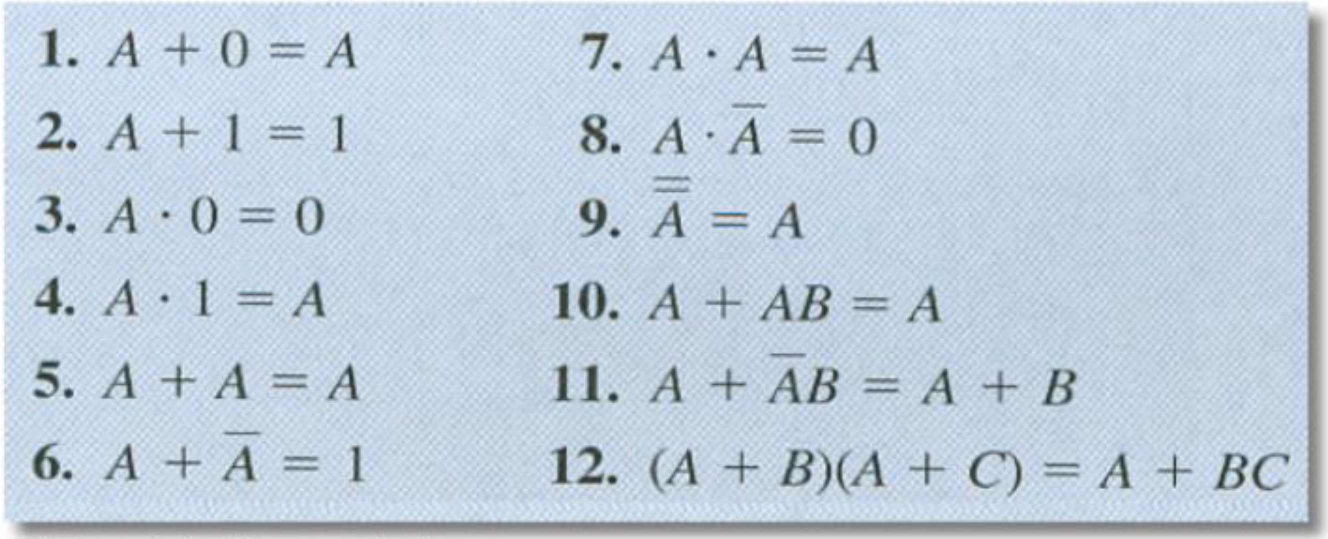


- Distributive Law

A • (B + C) = A • B + A • C
A (B + C) = A B + A C

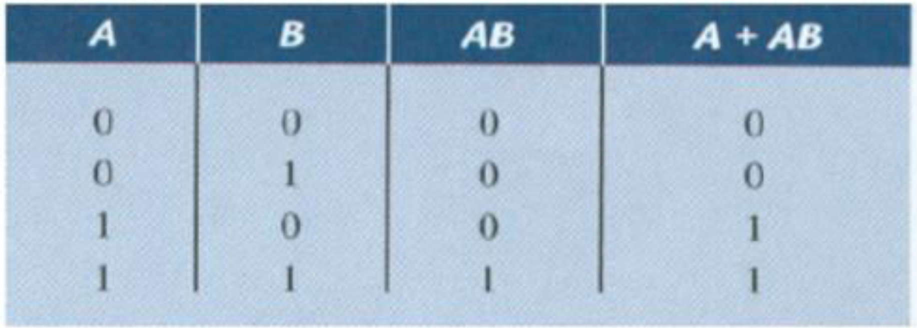


Rules

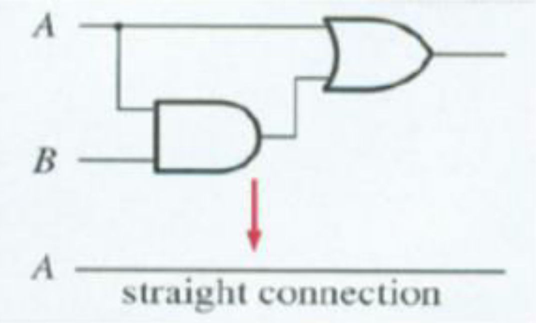


1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

- Rules 1 to 9 are obvious.
- Rule 10: A + AB = A

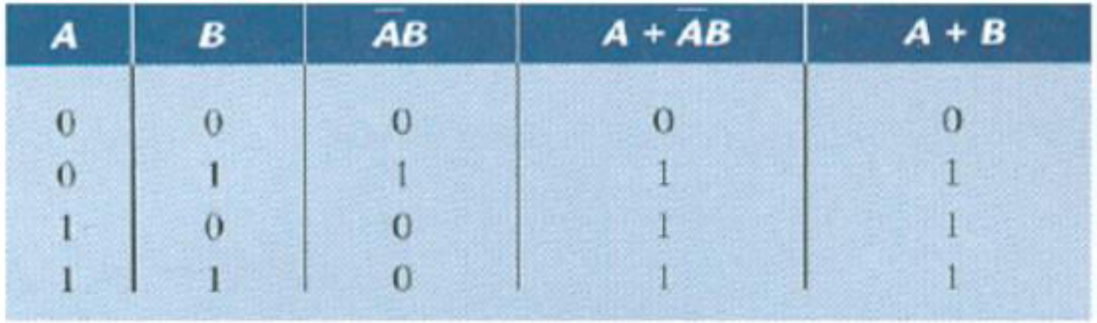


A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

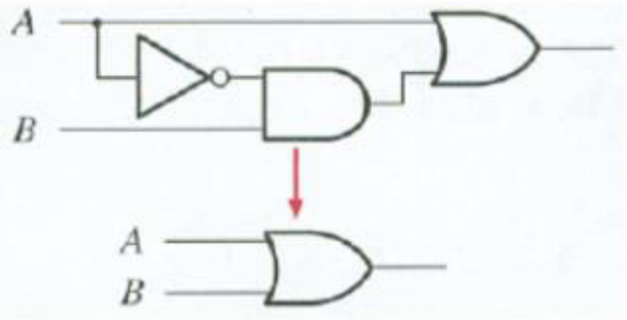


Rules

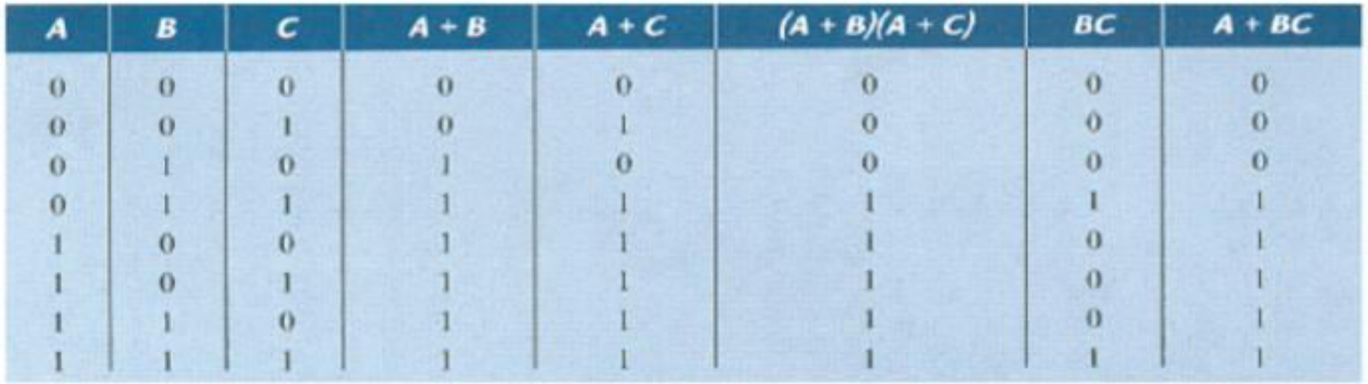
- Rule 11: $A + \overline{A}B = A + B$



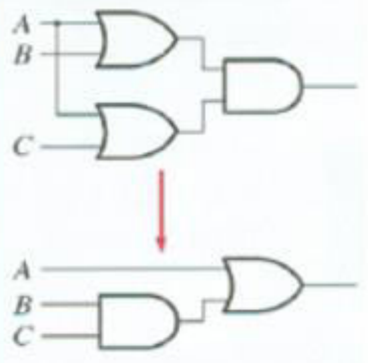
A	B	$\overline{A}B$	$A + \overline{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1



- Rule 12: (A + B)(A + C) = A + BC



A	B	C	$A + B$	$A + C$	$(A + B)(A + C)$	BC	$A + BC$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

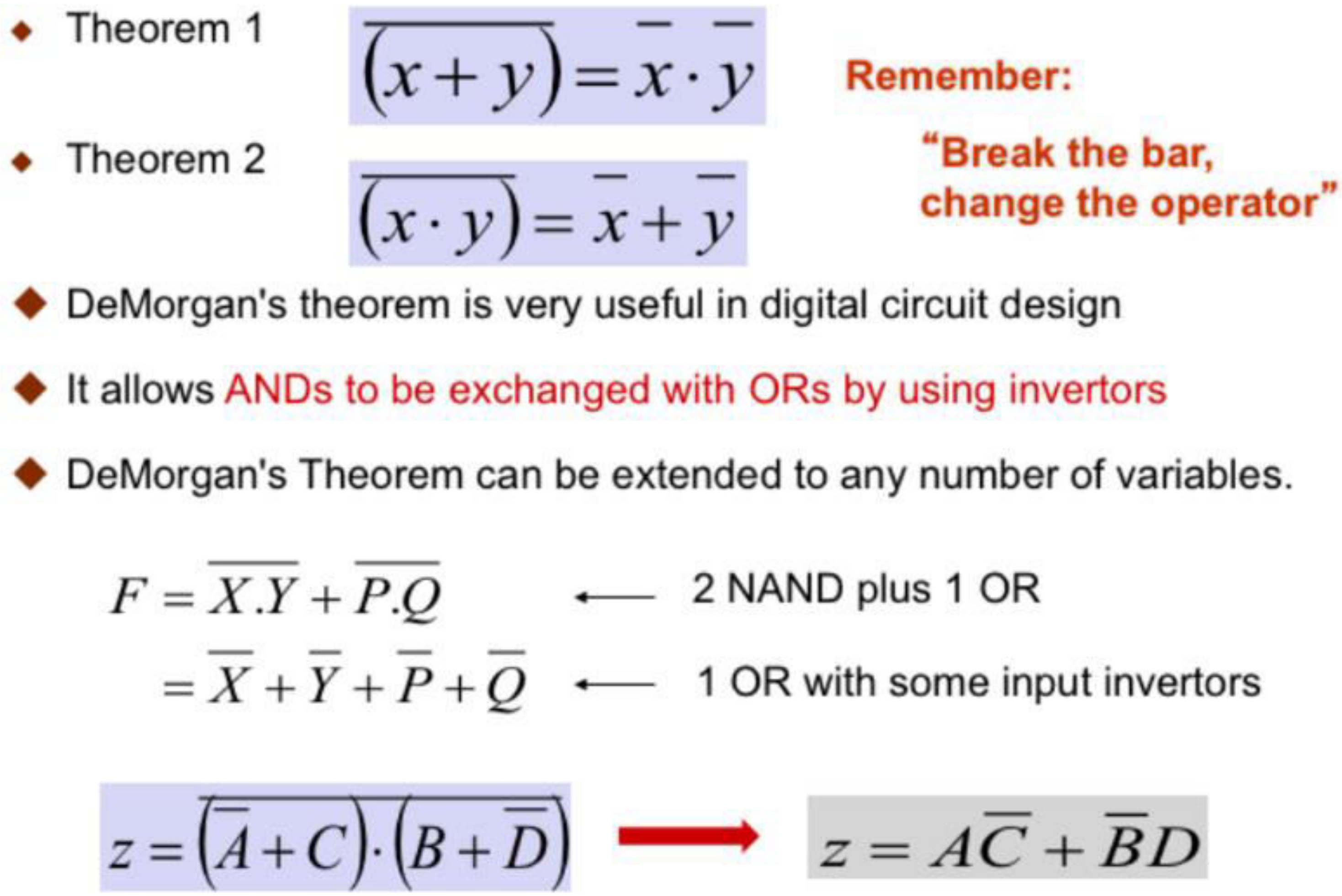


Simplify Boolean Equation

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

$$
\begin{aligned}
&= A + AC + AB + BC \quad (AA = A) \\
&= A(1 + C) + AB + BC \qquad (1 + C = 1) \\
&= A.\,1 + AB + BC \\
&= A(1 + B) + BC \qquad (1 + B = 1) \\
&= A.\,1 + BC \qquad (A\,.\,1 = A) \\
&= A + BC
\end{aligned}
$$

DeMorgan’s Theorem



- Theorem 1 $\overline{(x+y)}=\overline{x}\cdot\overline{y}$
- Theorem 2 $\overline{(x\cdot y)}=\overline{x}+\overline{y}$

Remember:

“Break the bar, change the operator”

- DeMorgan's theorem is very useful in digital circuit design
- It allows ANDs to be exchanged with ORs by using invertors
- DeMorgan's Theorem can be extended to any number of variables.

$F=\overline{X.Y}+\overline{P.Q}$ ← 2 NAND plus 1 OR

$=\overline{X}+\overline{Y}+\overline{P}+\overline{Q}$ ← 1 OR with some input invertors

$z=\overline{\left(\overline{A}+C\right)\cdot\left(B+\overline{D}\right)}$ → $z=A\overline{C}+\overline{B}D$

That’s all for today………