
EW
@ENGINEERINGWALLAH

UNIT - III

Number System and Logic Gates

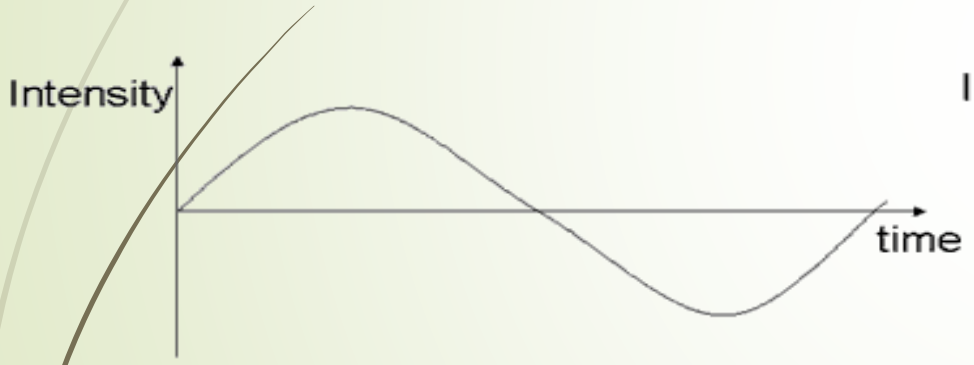
Syllabus

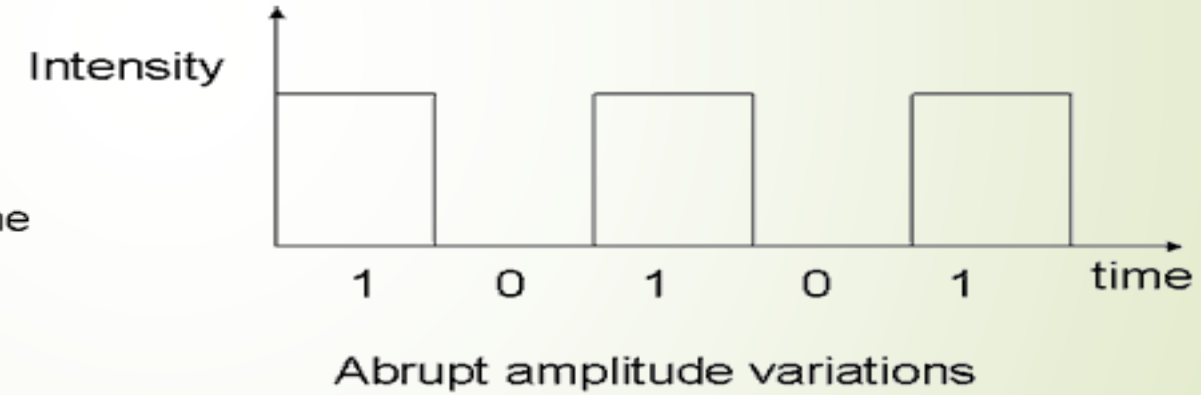
Unit III

Number System and Logic Gates [7L]

- Number System:- Binary, BCD, Octal, Decimal, Hexadecimal their conversion and arithmetic, De-Morgan's theorem.
- Basic Gates:- AND, OR, NOT, Universal Gate- XOR, XNOR, Half adder, Full adder Flip Flop's SR, JK, T and D
- Introduction to Microprocessor and Microcontroller (Only block diagram and explanation)

- Analog circuits processes analog signals and digital circuits processes digital signals





Analog Vs Digital

Analog Electronics	Digital Electronics
It has larger circuits and occupies more space.	It has smaller integrated circuits and occupies smaller space.
Less accurate .	More accurate.
Less noise immunity.	More noise immunity.
Processing speed is less.	Processing speed is high.
Cannot store analog signals.	Can be stored and processed the data.

Type of Digital Signal	Number of Distinct Values
Binary	2
octal	8
Hexadecimal	16

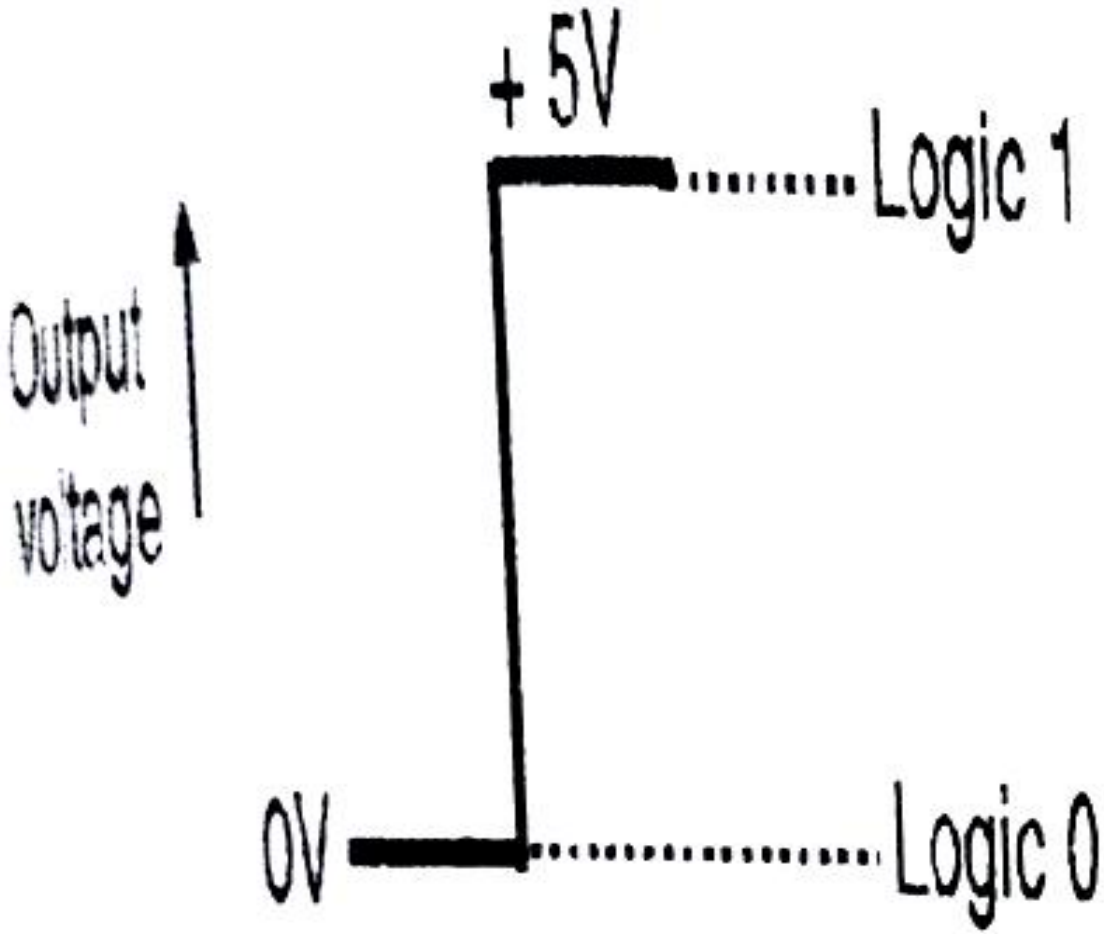
Binary as a Voltage

Voltages are used to represent logic values:

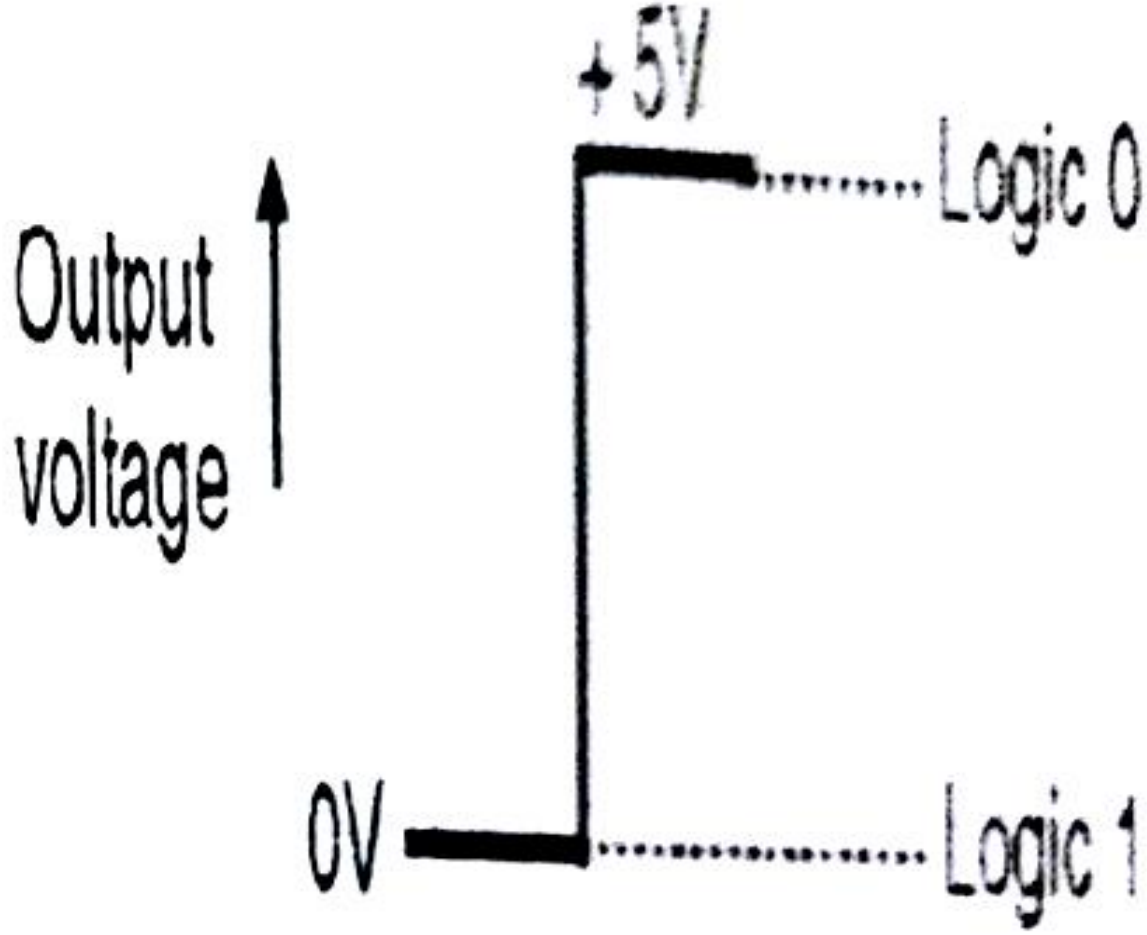
A voltage present (called Vcc or Vdd) = 1

Zero Volts or ground (called gnd or Vss) = 0

A simple switch can provide a logic high or a logic low.



Positive logic



Negative logic

Digital level Logic

	Positive Logic	**Negative Logic**
1 is represented by	High Voltage (5V or Vcc)	Low Voltage (0V or Ground)
0 is represented by	Low Voltage (0V or Ground)	High Voltage (5V or Vcc)

Units of information

- Bit= either ‘1’ or ‘0’
- Nibble=Group of 4-bits

  = e.g. ‘ 0010’
- Byte = Group of 8-bits

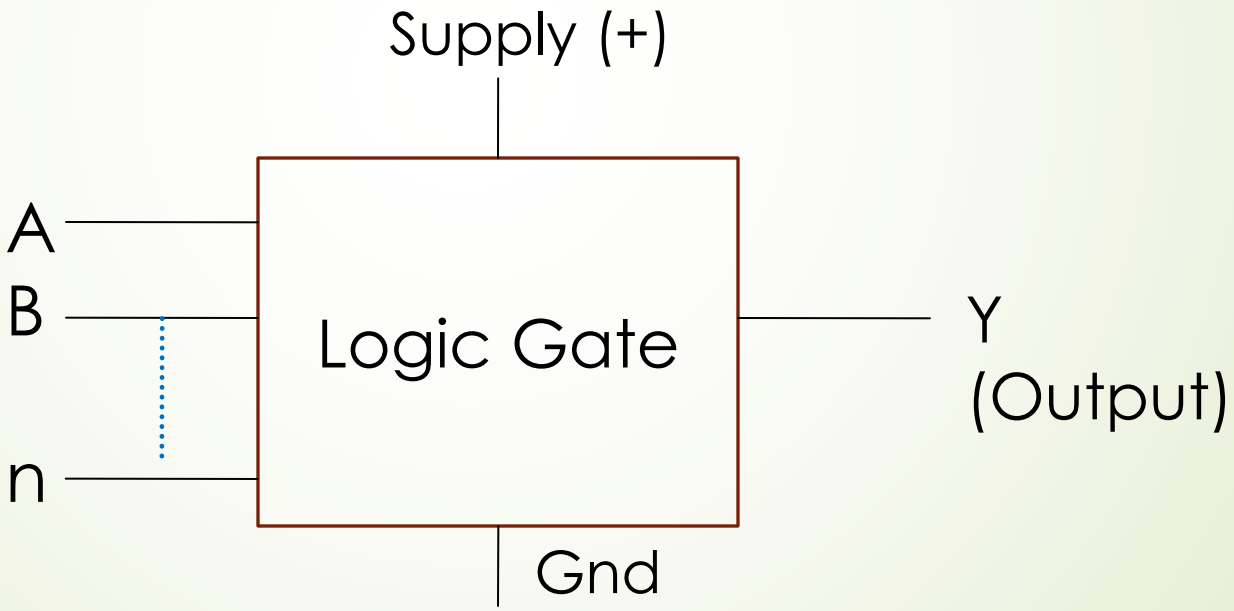
  1 Byte = e.g. ‘1111 0010’
- Word = Group of 16-bits(2 bytes)

Integrated circuit (IC) Generation

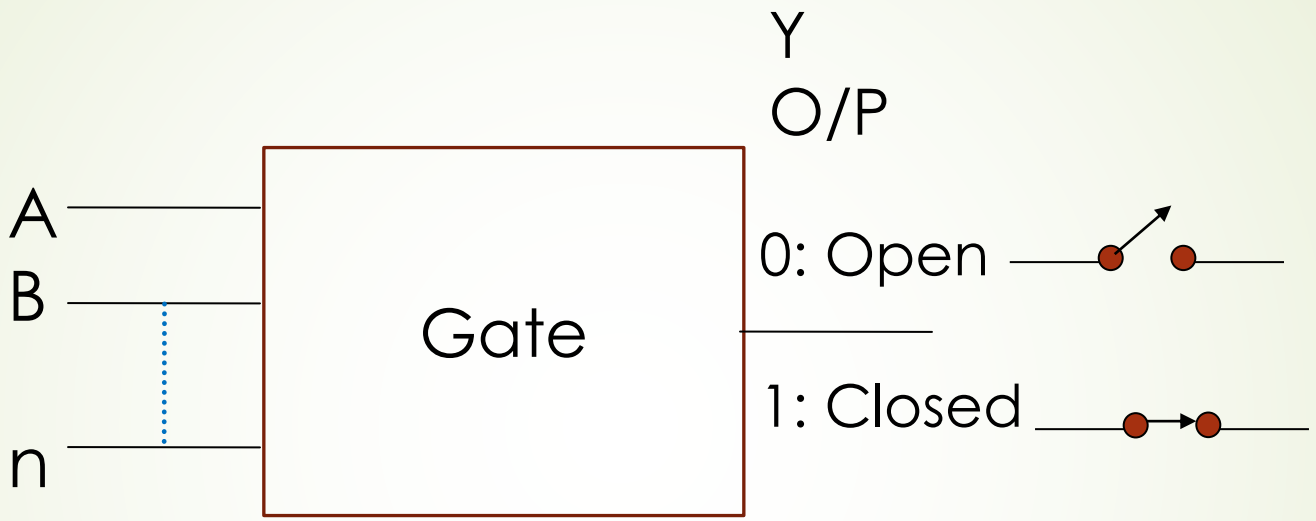
Name	*Signification*	*Year*	*Number of Components*	*Application*
SSI	**Small-scale Integration**	**1964**	**< 10**	**Logic Gates or Static Switches**
MSI	**Medium-scale Integration**	**1968**	**< 100**	**Adders. Sub tractors, MUX, DMUX, Encoder, Decoder, Counters, Registers**
LSI	**Large-scale Integration**	**1971**	**>100**	**Microcontrollers and Microprocessors IC's**
VLSI	**Very Large-scale Integration**	**1980**	**>1000**	
ULSI	**Ultra-large-scale Integration**	**1984**	**>1,000,00 and more**	

Logic Gates

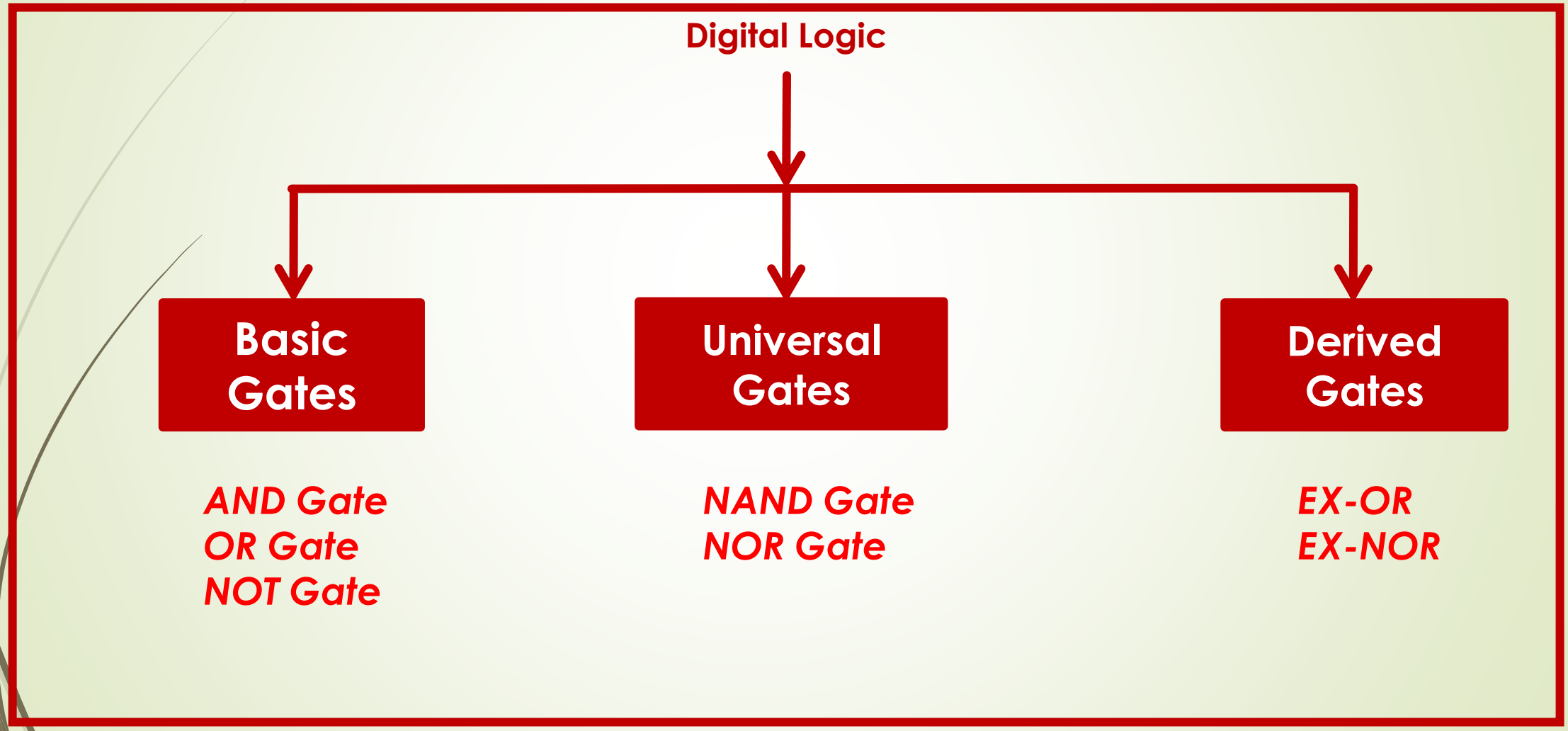
- Definition: It is an electronic circuit which can have many input but only one output.
- Symbol:



Logic Gates



Digital Logic

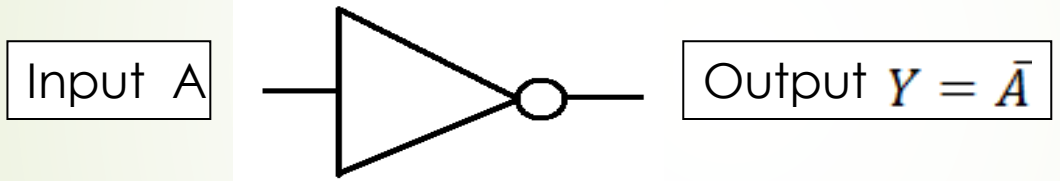


Logic Gates

- **Boolean Equation**: It is a **mathematical expression** representing a particular gate or logic. Scientist Boolean invented this algebra.
- **Symbol**: Each gate is represented by a **logic symbol**.
- **Logic diagrams:** A graphical representation of a circuit; each gate has its own symbol
- **Truth tables:** A table showing all possible input value and the associated output values

Basic Gates : The **NOT** Function

- **"The output is the opposite state of the input."**
- **The NOT function is often called INVERTER.**
- **Logical Symbol:**



If the input is 1, the output is 0

If the input is 0, the output is 1

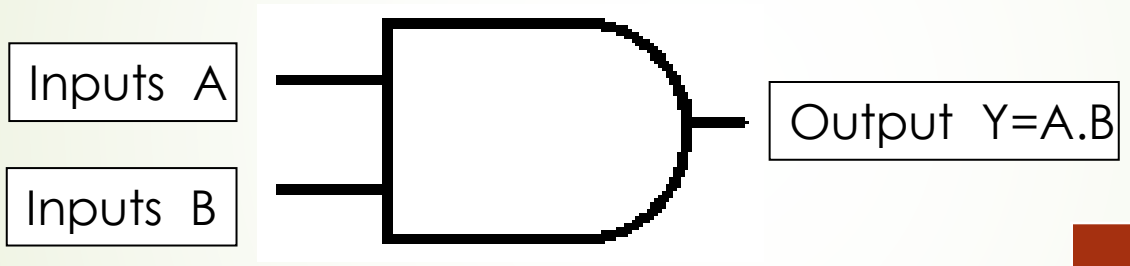
Input	Output
0	1
1	0

Logical Equation $Y = \bar{A}$

IC Number- 7404

Basic Gates : The **AND** Function

- **The output is high if and only if all the inputs are high.**
- **Logical Symbol:**



If both inputs are 1, the output is 1

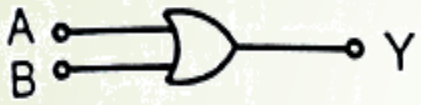
If any input is 0, the output is 0

A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

- **Logical Equation :** $Y = A \cdot B$
- **IC Number- 7408**

Basic Gates : The **OR** Function

Symbol



Definition:

OR gate: The output is high when any of the inputs is high

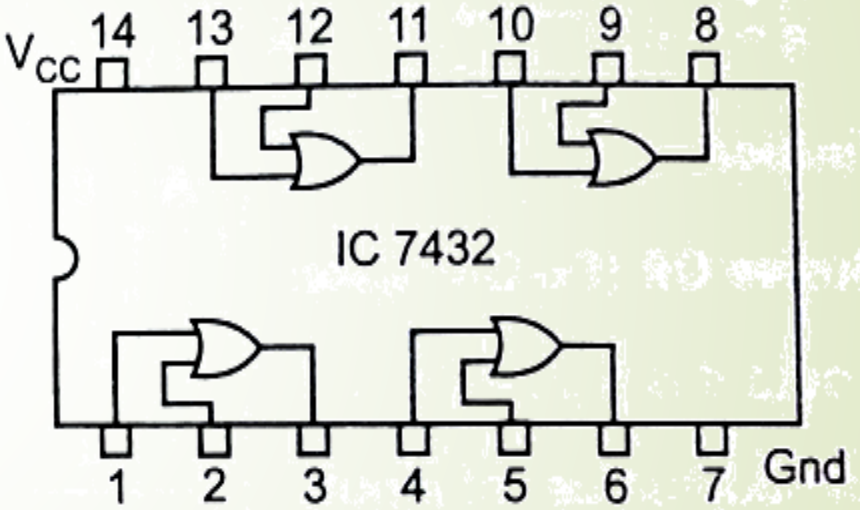
Boolean Expression:

$Y = A + B$

Truth Table

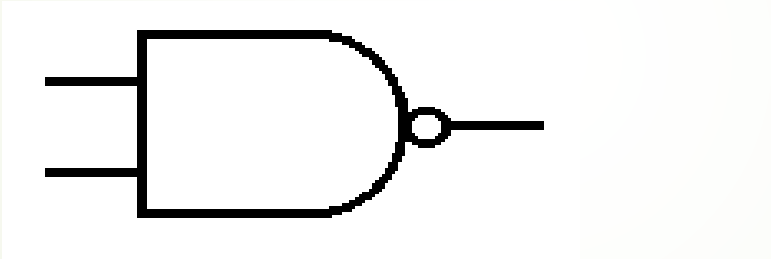
Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Pin Diagram



Universal Gates : The **NAND** Function

- **The NAND gate is a combination of AND with NOT gate.**
- **The output is low when all the inputs are high.**
- **Logical Symbol :**



The Bubble in front of the gate is an inverter.

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

- **Logical Equation :**

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

- **IC Number- 7400**

Universal Gates : The **NAND** Function

Symbol

Definition:

NAND gate: The output is high only when one of the inputs is Low.

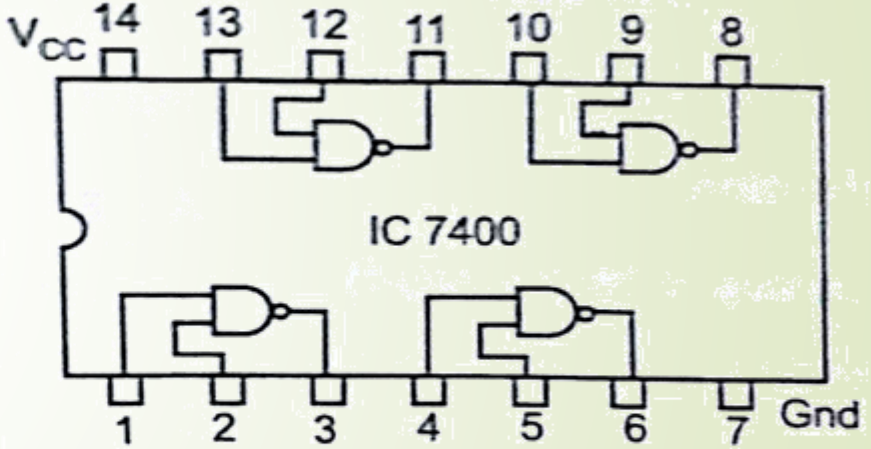
Boolean Expression:

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

Truth Table

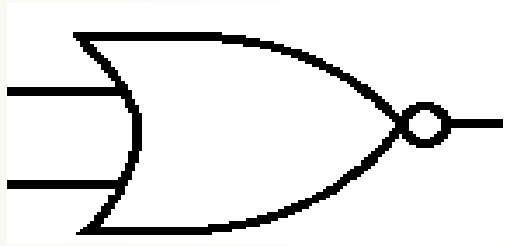
Input		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Pin Diagram



Universal Gates : The **NOR** Function

- **The NOR gate is a combination of OR with NOT gate.**
- **The output of NOR gate is high when all the inputs are low.**
- **Logical symbol :**



The Bubble in front of the gate is an inverter.

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

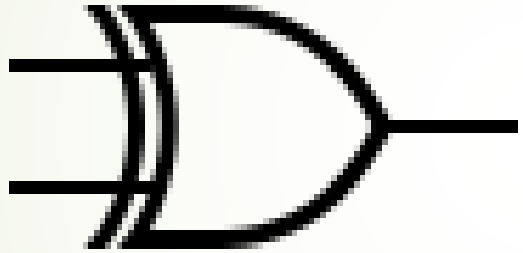
Logical Equation :

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

- **IC Number- 7402**

Derived Gates : The **EX-OR** Function

- **The output of EX-OR gate is high when both the inputs are either 0 or 1.**
- **Logical symbol :**



Determine the output

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

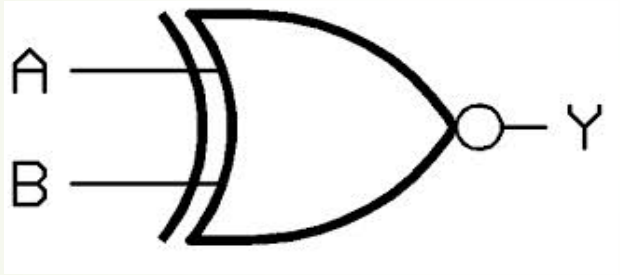
- **Logical Equation :**

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

IC Number: 74266

Derived Gates : The **EX-NOR** Function

- **The output of EX-NOR gate is high when both the inputs are either 0 or 1.**
- **Logical symbol :**



Determine the output

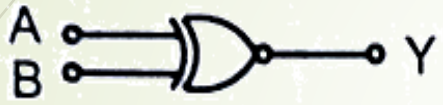
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

- **Logical Equation :** $Y = \overline{A \oplus B} = AB + \overline{A}\,\overline{B}$

IC Number: 74266

Derived Gates : The **EX-NOR** Function

Symbol



Definition:

Exclusive NOR (Ex-NOR) gate: The output is high if both the inputs are equal.

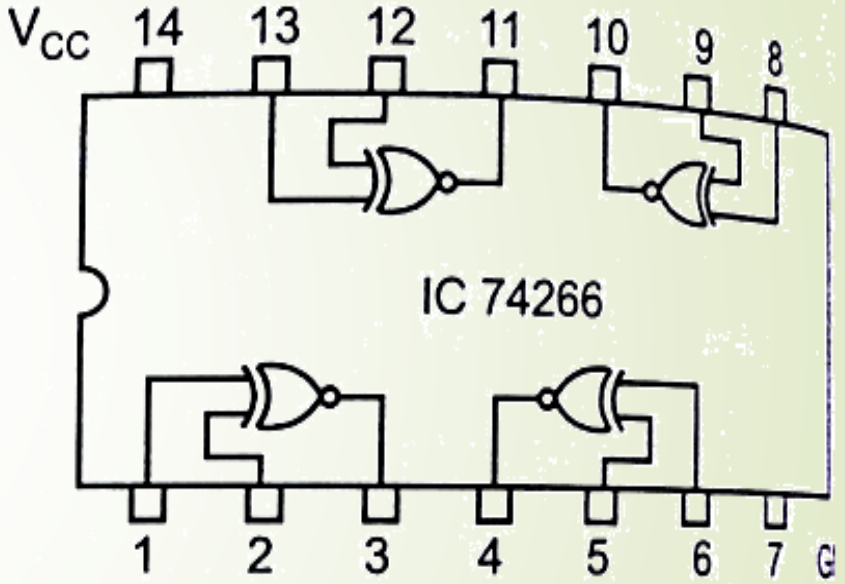
Boolean Expression:

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

Truth Table

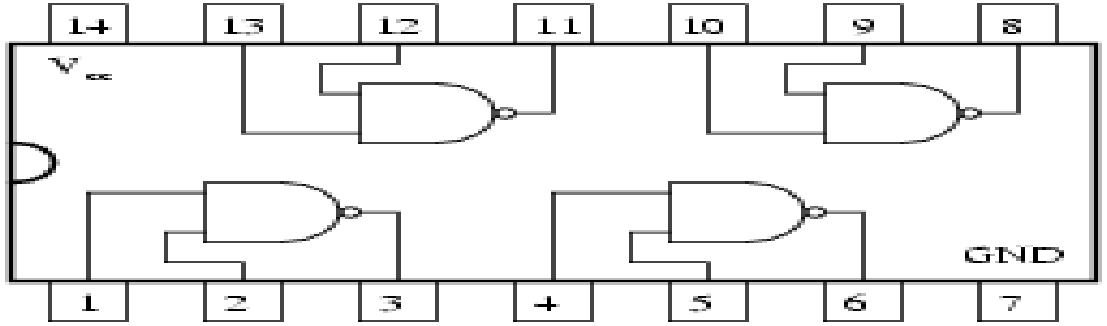
Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Pin Diagram

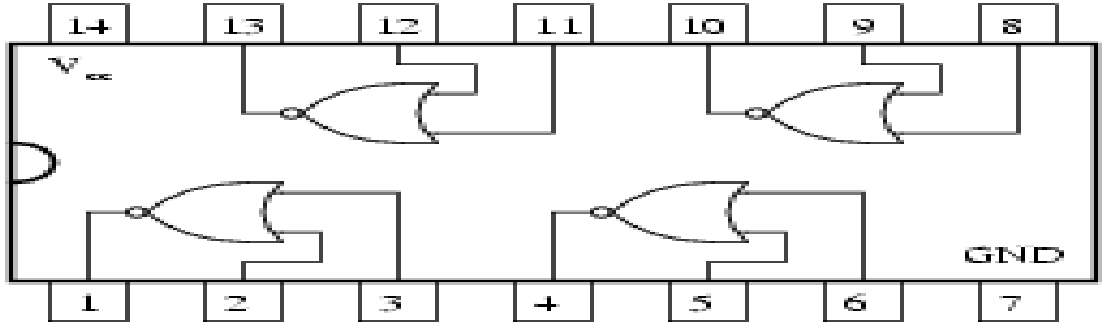


GATE NAME	IC Number
NAND	7400
NOR	7402
NOT	7404
AND	7408
OR	7432
EX-OR	7486
EX-NOR	74266

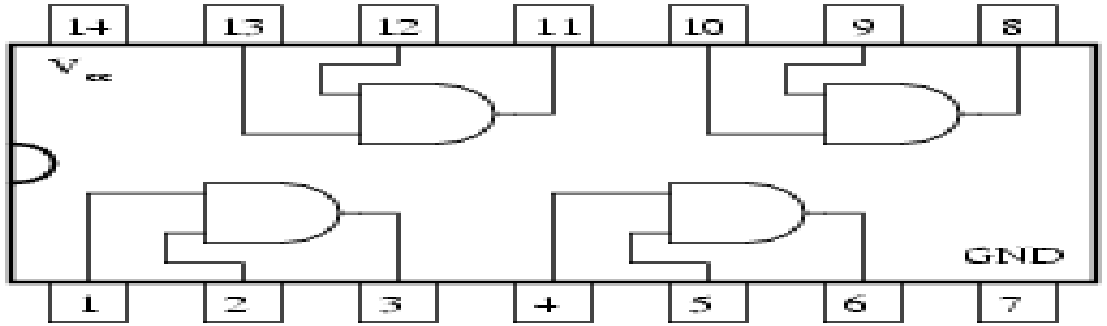
5400/7400
Quad NAND gate



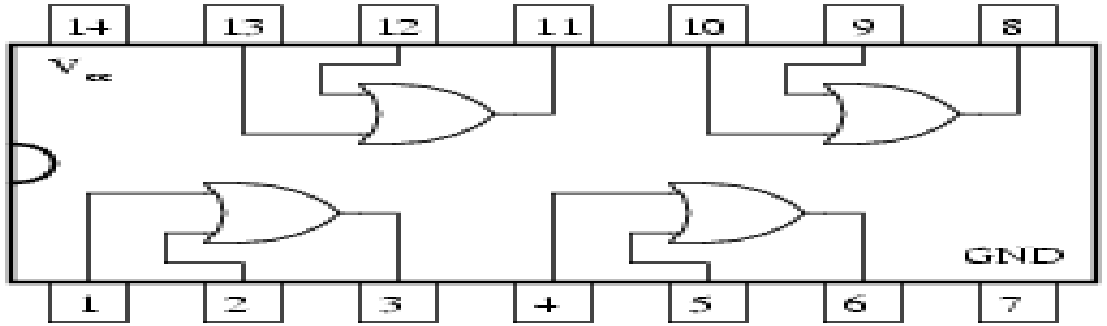
5402/7402
Quad NOR gate



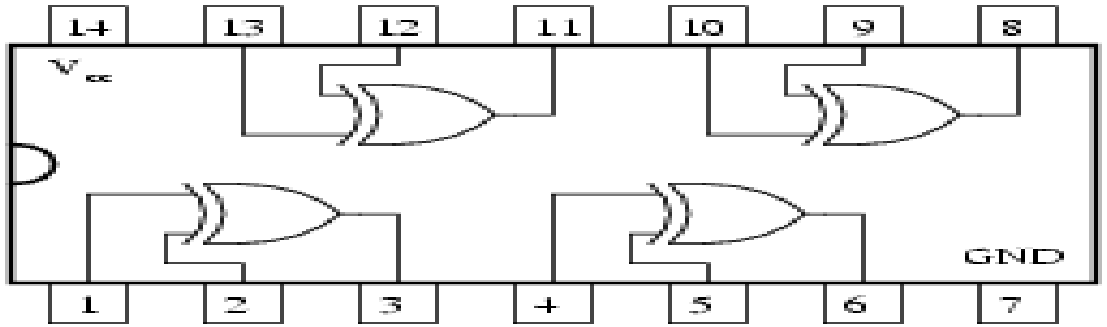
5408/7408
Quad AND gate



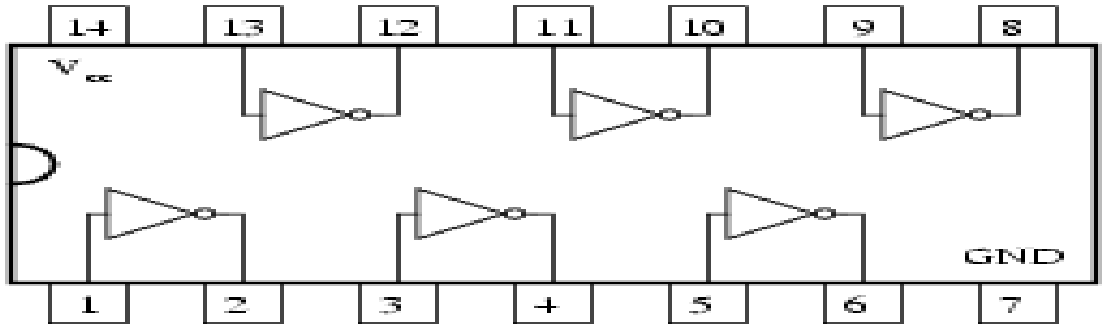
5432/7432
Quad OR gate



5486/7486
Quad XOR gate



5404/7404
Hex inverter



Gate Name	Symbol	IC Number	Boolean Equation	Truth Table		
NOT				A		Y
AND				A	B	Y
OR				A	B	Y
NAND				A	B	Y
NOR				A	B	Y
EX-OR.				A	B	Y
EX- NOR				A	B	Y

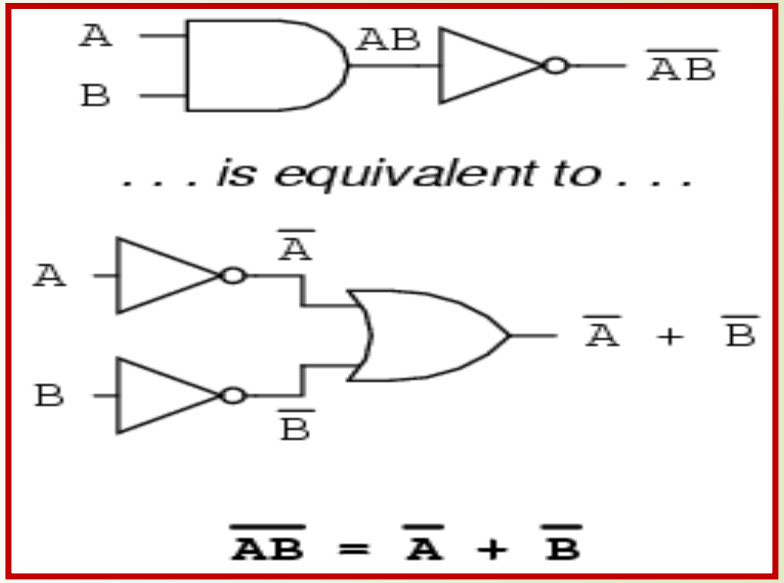
Boolean Laws

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

De-Morgan's Theorems

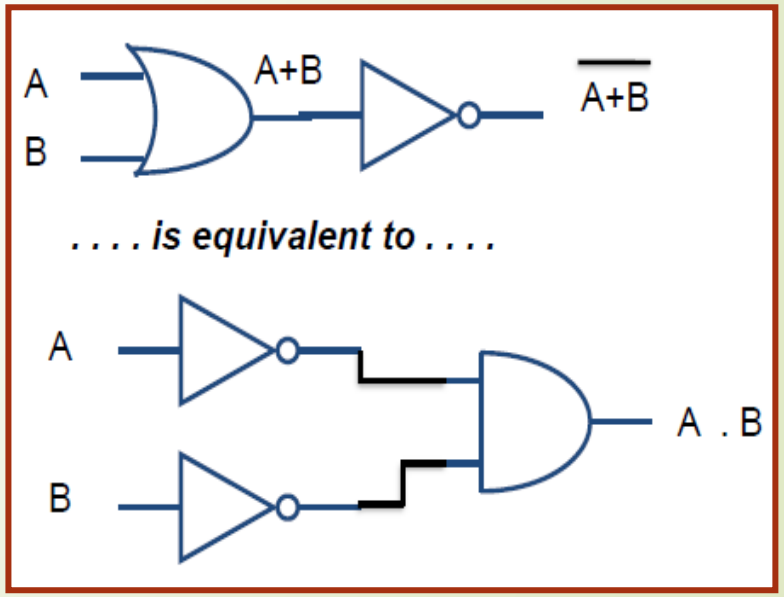
- **Theorem 1: $\overline{AB} = \overline{A} + \overline{B}$ (NAND = Bubbled OR)**

It states that the, complement of a product is equal to addition of the complements.



- **Theorem 2: $\overline{A+B} = \overline{A} \cdot \overline{B}$ (NOR = Bubbled AND)**

It states that the, complement of a sum is equal to product of the complements.

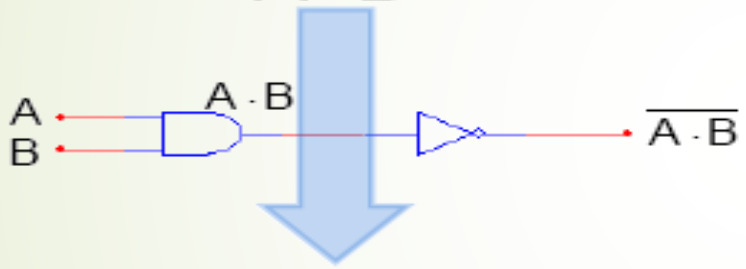


DeMorgan's Theorem #1

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

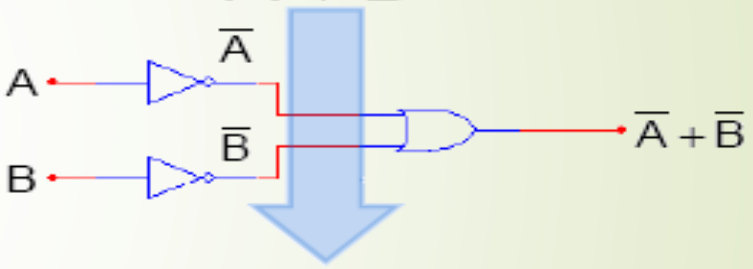
Proof

$\overline{A \cdot B}$



A	B	A · B	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

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



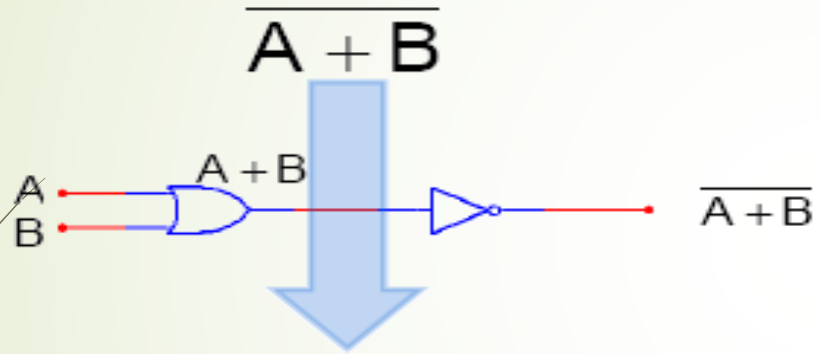
A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

The truth-tables are equal; therefore, the Boolean equations must be equal.

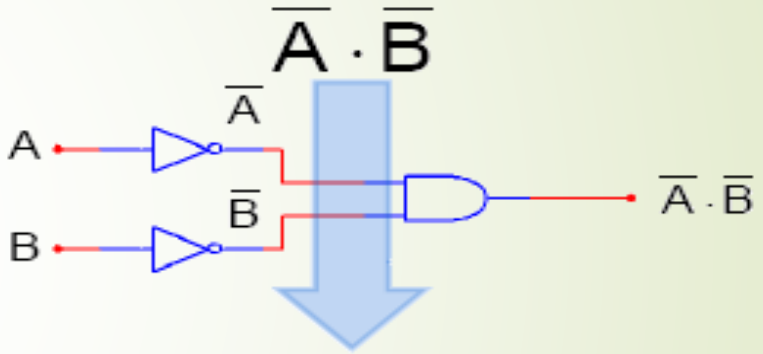
DeMorgan's Theorem #2

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

Proof



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

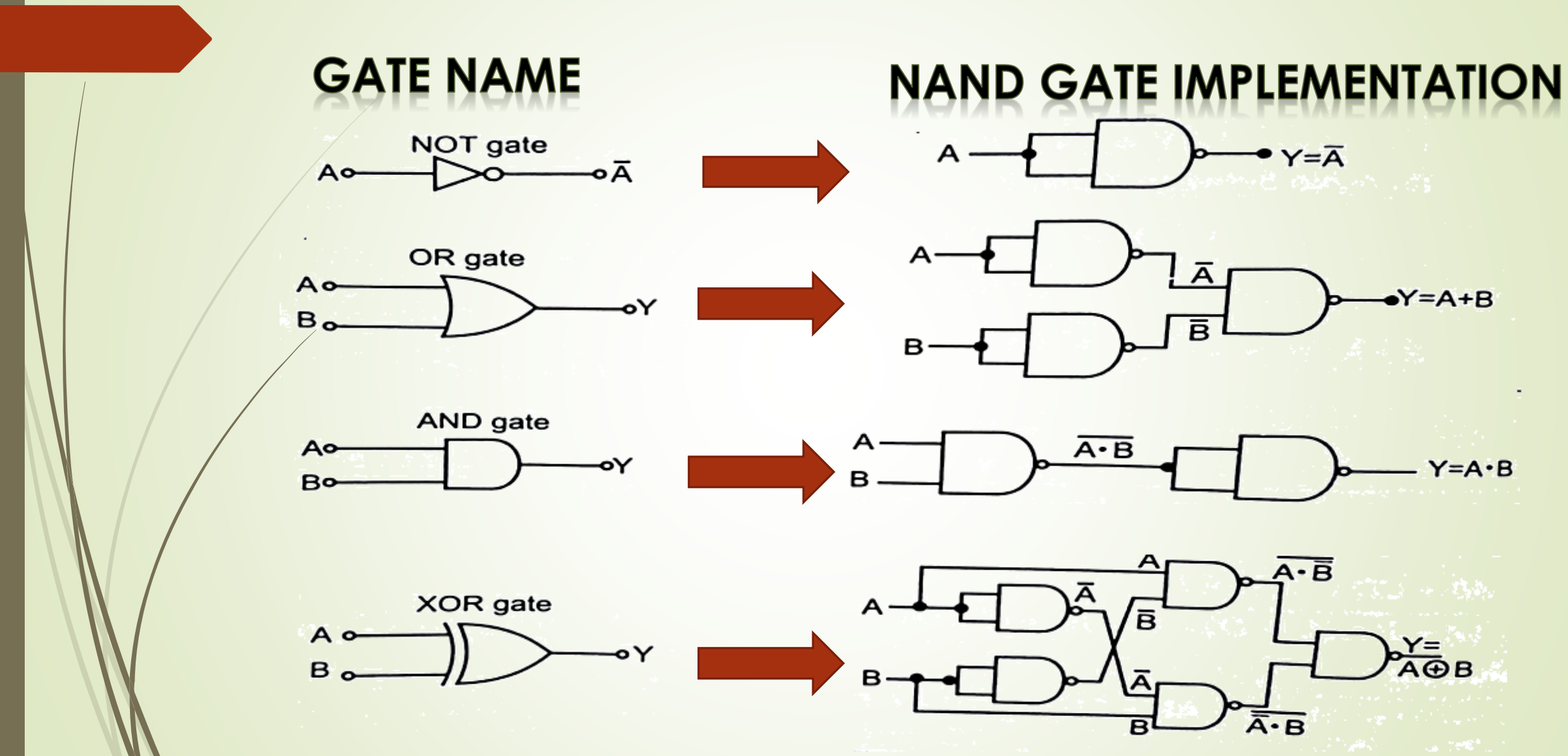


A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

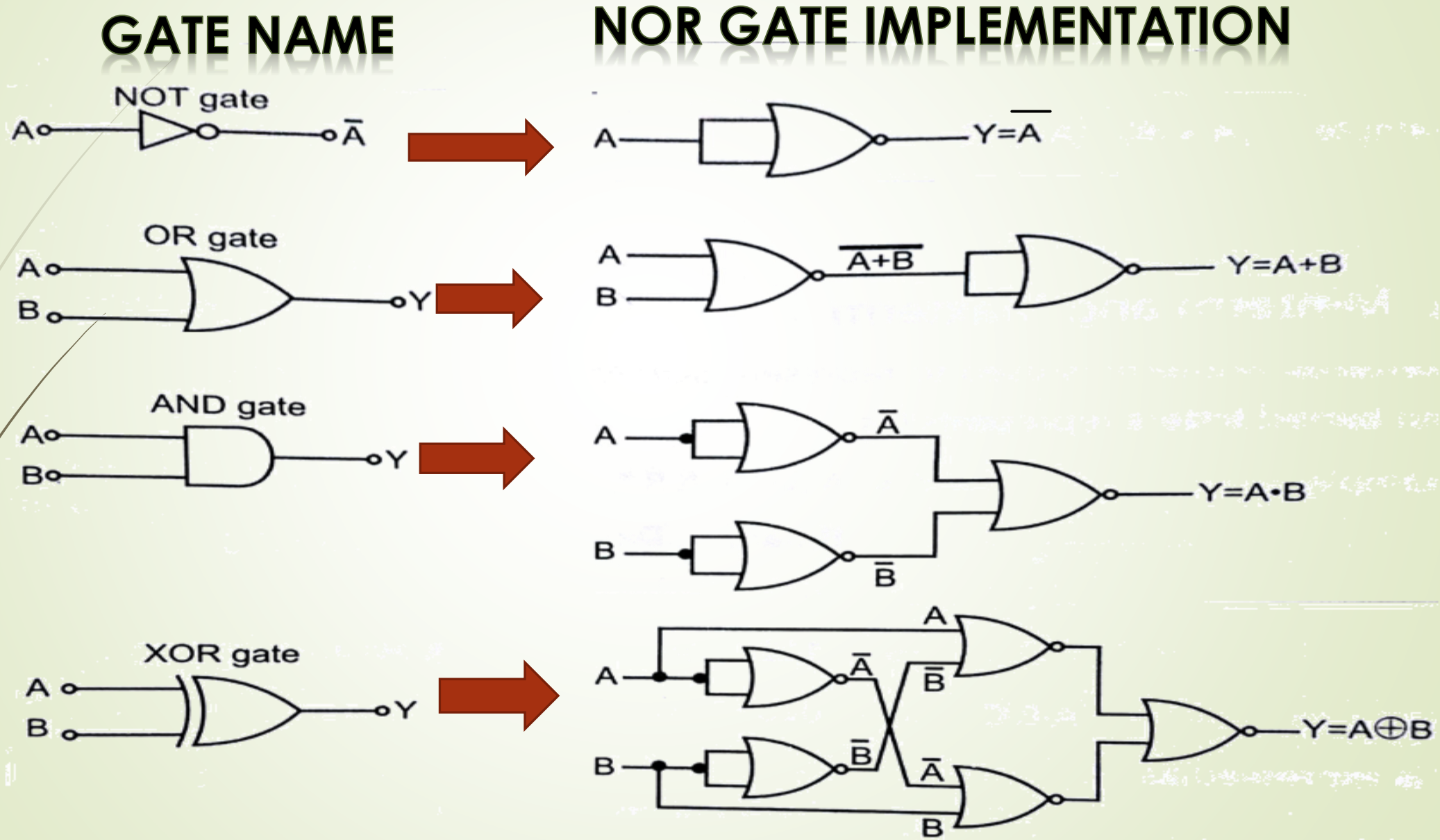
The truth-tables are equal; therefore, the Boolean equations must be equal.

Q. Why NAND and NOR gate are called universal gates? Realize AND, OR, NOT, EX-OR, EX-NOR using NAND and NOR.

NAND Gate as Universal Gate



NOR Gate as Universal Gate



Rules:

- **Binary Addition Rules :**

0 + 0 = 00 (0 with a 0 carry)

0 + 1 = 01 (1 with a 0 carry)

1 + 0 = 01 (1 with a 0 carry)

1 + 1 = 10 (0 with a 1 carry)

- **Addition Rules with Carries :**

0+0+0 = 00 (0 WITH 0 CARRY)

0+0+1 = 01 (1 WITH 0 CARRY)

0+1+1 = 10 (0 WITH 1 CARRY)

1+1+1 = 11 (1 WITH 1 CARRY)

- **Addition Rules :**

0	1	0	1
+ 0	+ 0	+ 1	+ 1
0 0	0 1	0 1	1 0

- **Adding Binary Nos. :**

		0 1 1 1 0 0 0
28	⟶	00011100
+ 43	⟶	+ 00101011
71		01000111

Binary Addition

- **The addition of two binary numbers is performed in exactly the same manner as the addition of decimal numbers.**
- **Least-significant-digit first.**
- **“Carry” of 1 into the next position may be needed.**
- **4 different cases for binary addition**

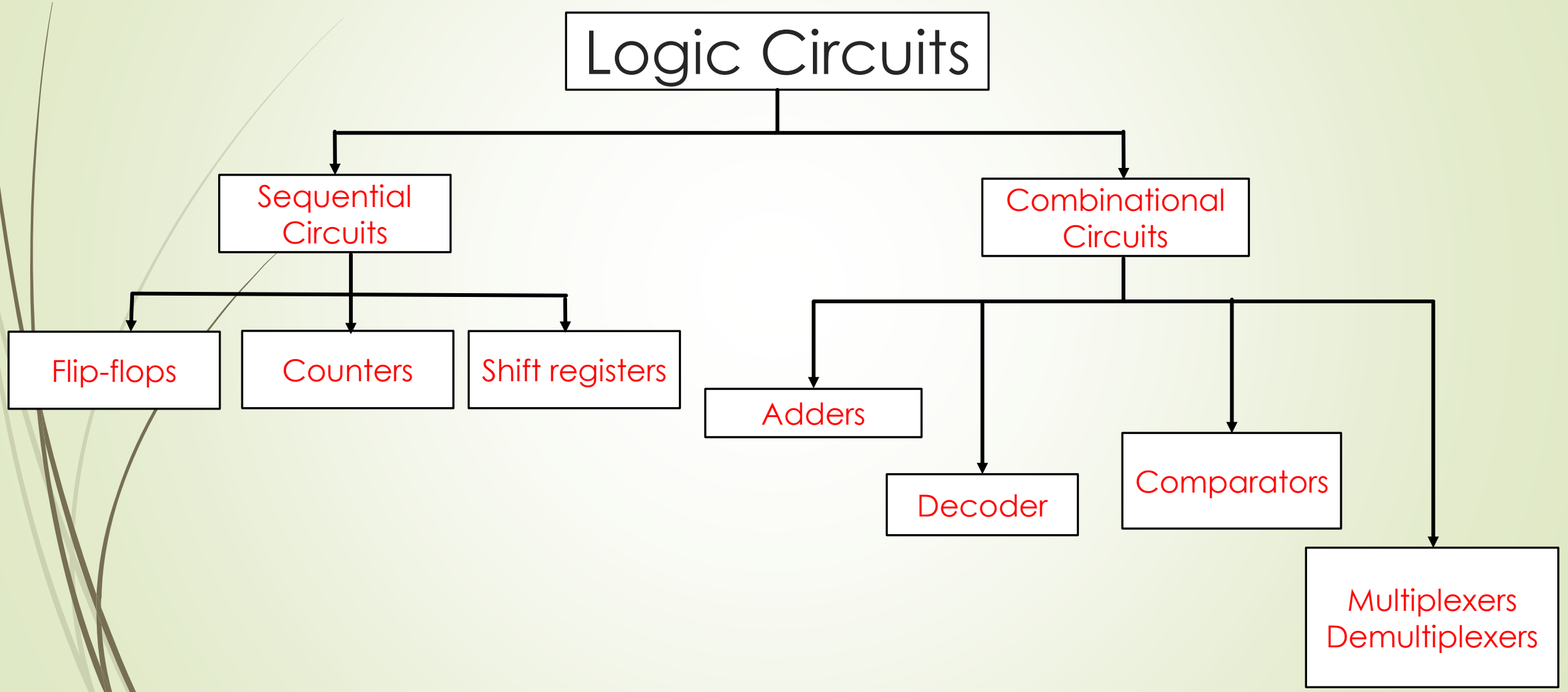
A	B	Addition	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- **The operations of subtraction, multiplication, and division actually use only addition as their basic operation**

Binary Subtraction

- The subtraction of two binary numbers is performed in exactly the same manner as the addition of decimal numbers.
- Least-significant-digit first.
- “Borrow” of 1 into the next position may be needed.
- 4 different cases for binary subtraction

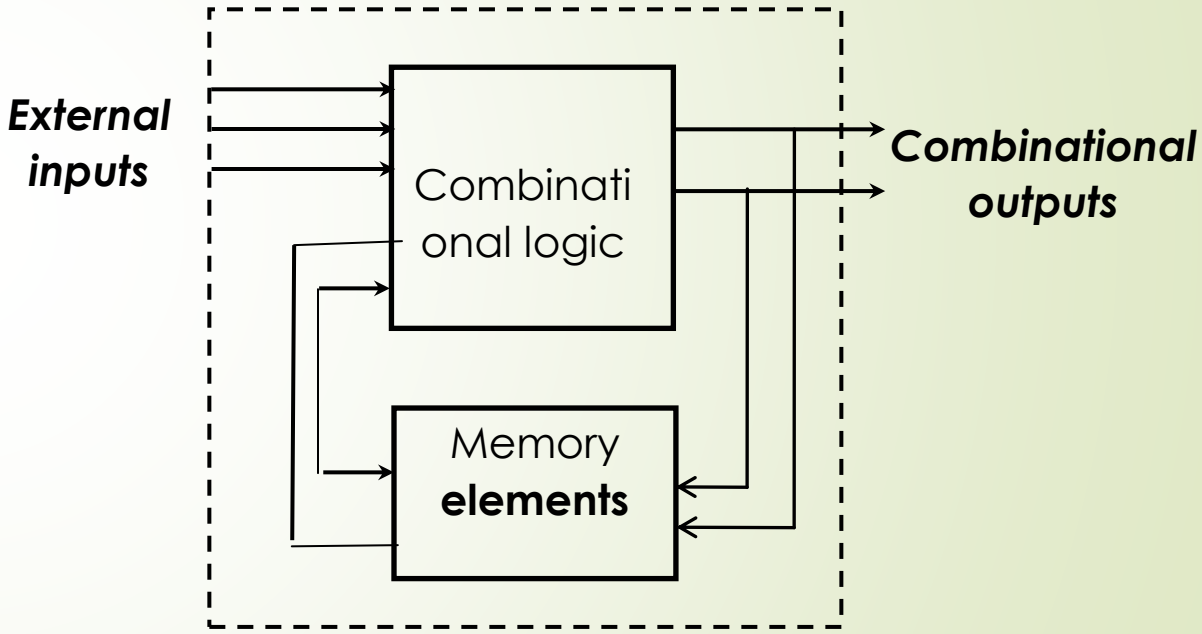
A	B	Subtraction	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0


Logic Circuits
Sequential Circuits
Combinational Circuits
Flip-flops
Counters
Shift registers
Adders
Decoder
Comparators
Multiplexers
Demultiplexers

Parameter	Combinational Circuits	Sequential Circuits
Output Depend on	Input present at instant of time	Present inputs and past inputs/outputs
Memory Element	Not Necessary	Necessary
Clock Input	Not Necessary	Necessary
Example	Adders, Subtractors, Half Adders, Full Adders, MUX, DMUX	Flip flops, counters , shift registers
Block Diagram		

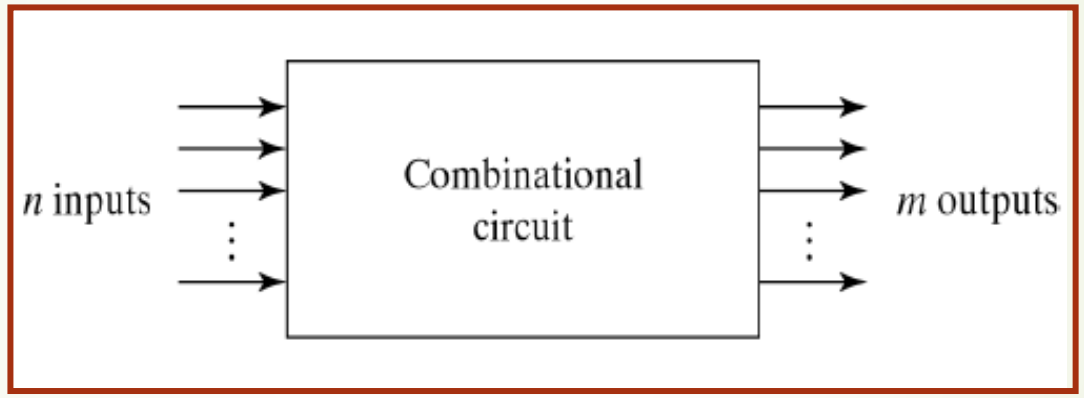
Combinational Circuits

Sequential Circuits



Combinational circuits – Adder

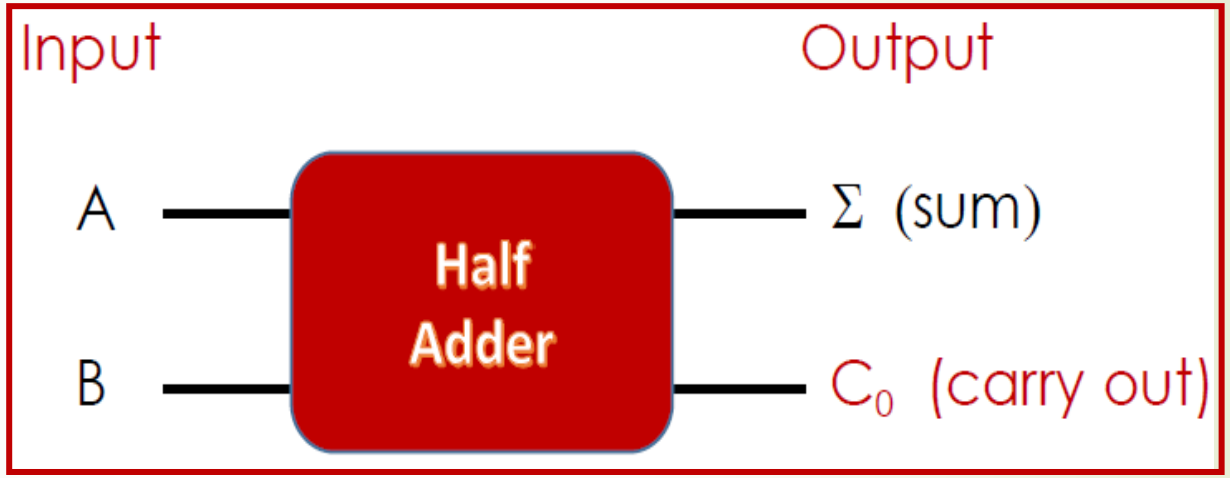
- A **combinational circuit** consists of input variables, logic gates, and output variables.



- An adder or summer is a **digital circuit** that performs **addition** of numbers.
- **Types** of Adder:
  1. Half Adder
  2. Full Adder

Half adder

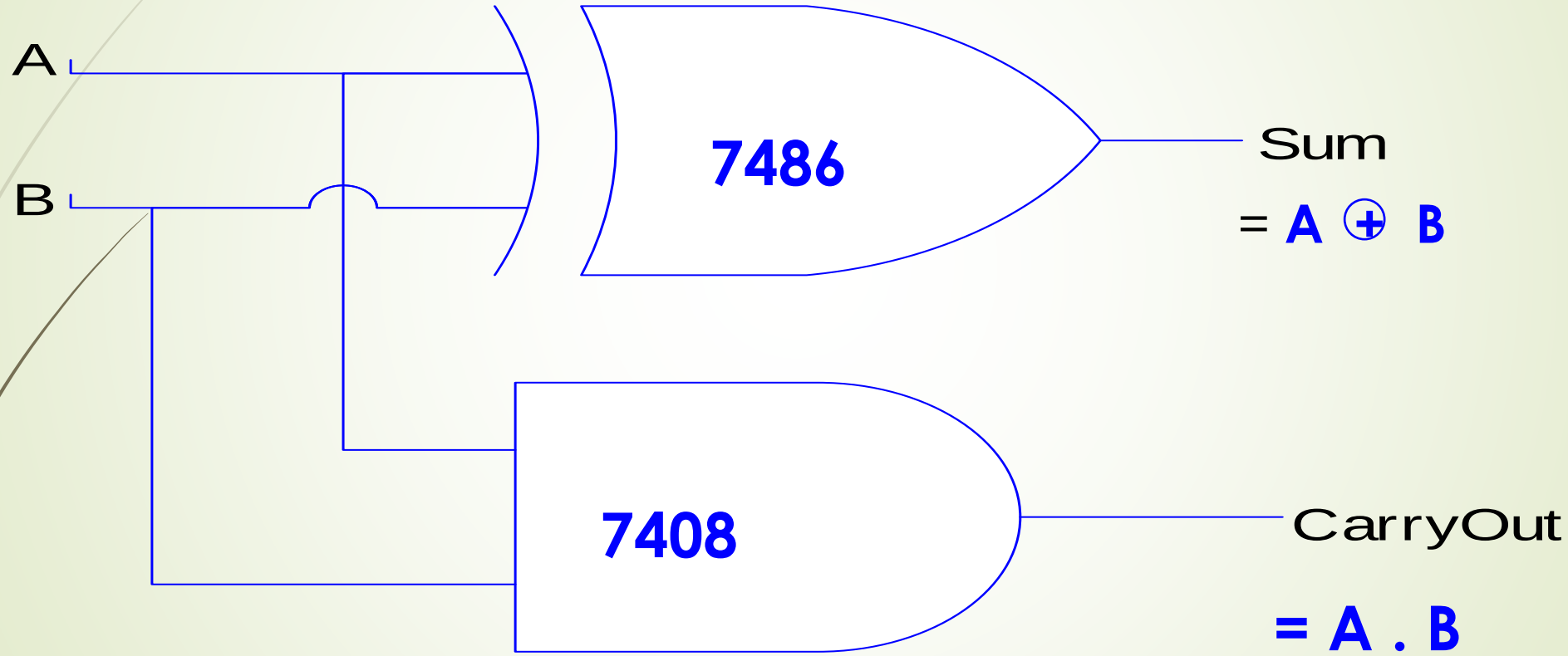
- **A combinational circuit that performs the addition of two bits is called a half adder.**
- **Logic symbol :**



- **Truth Table :**

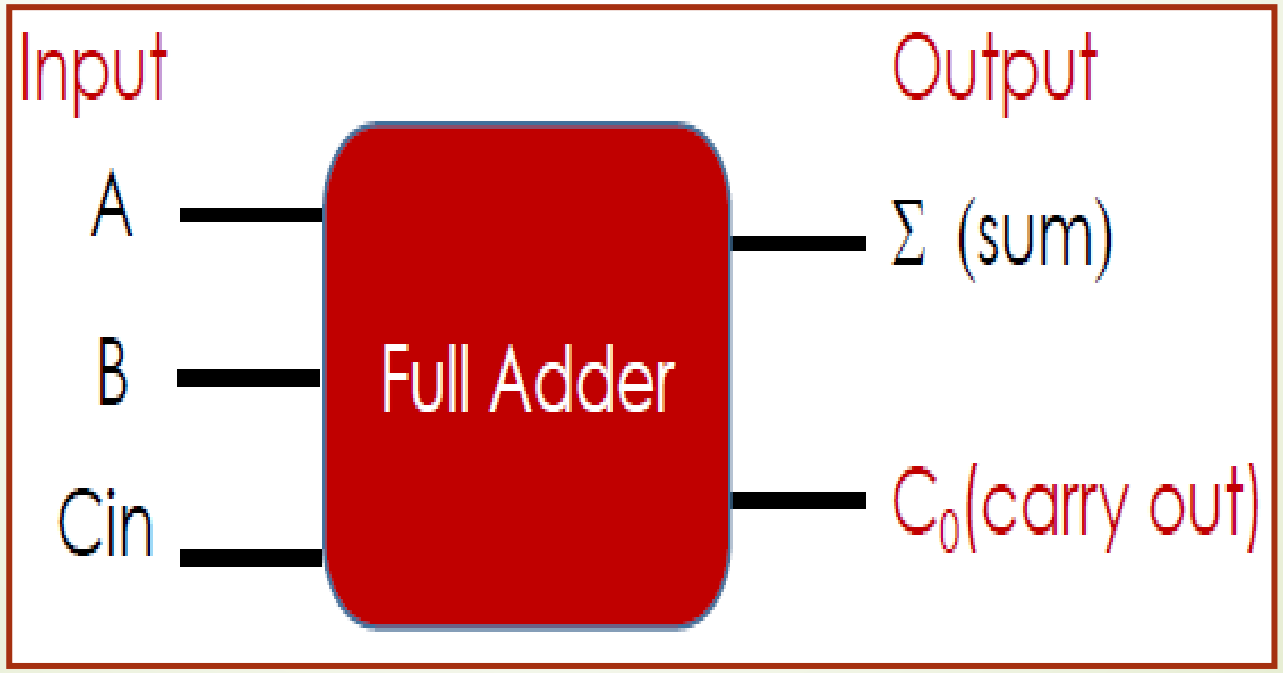
INPUT		Output	
A	B	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic Diagram for the Half Adder



Full Adder

- One that performs the addition of three bits(two significant bits and a previous carry) is a **full adder**.
- In a full adder, three bits can be added at a time. The third bit is a carry from a less significant column.
- Logic symbol :



Full Adder

- **Truth Table :**

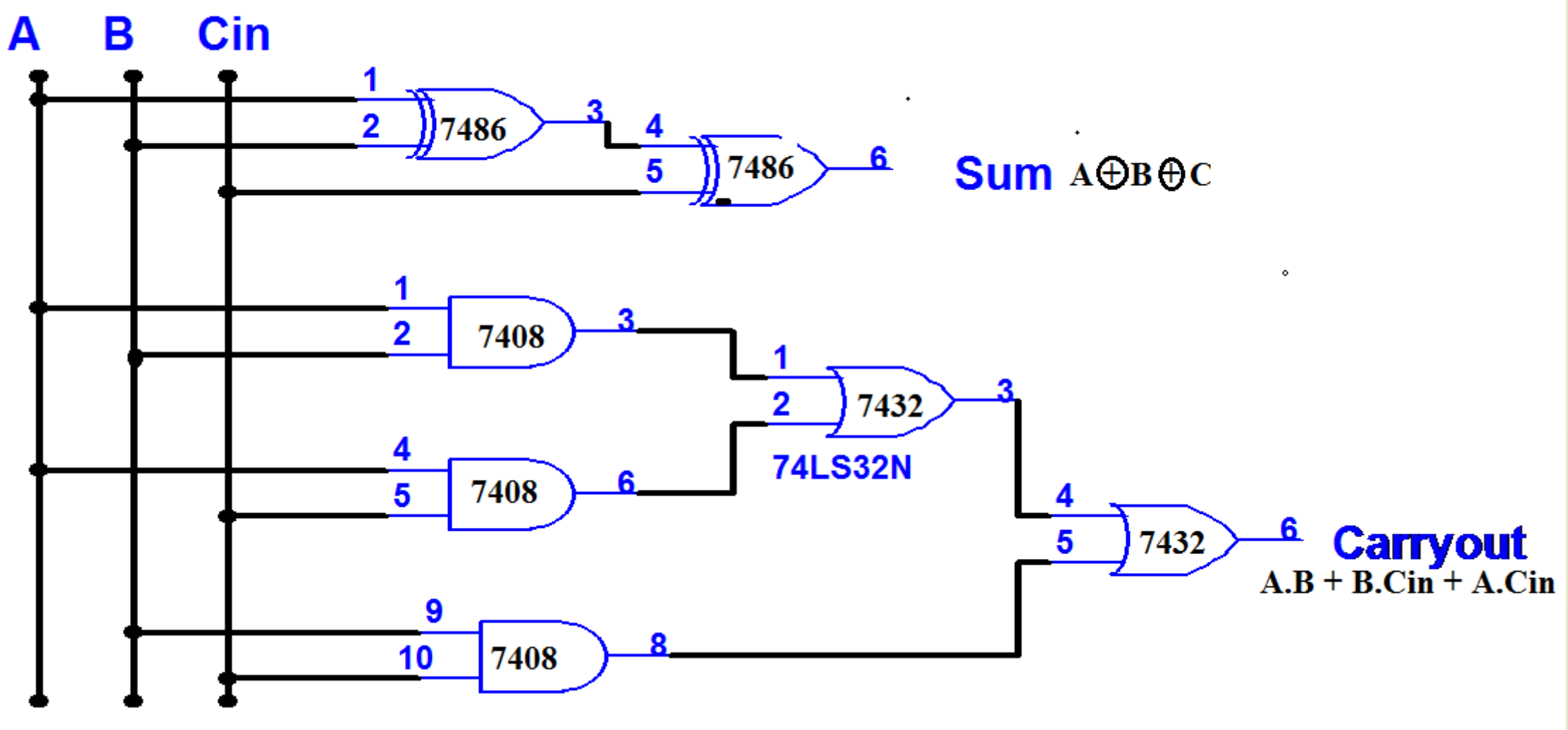
INPUT			Output	
A	B	Cin	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- **Boolean expressions :**

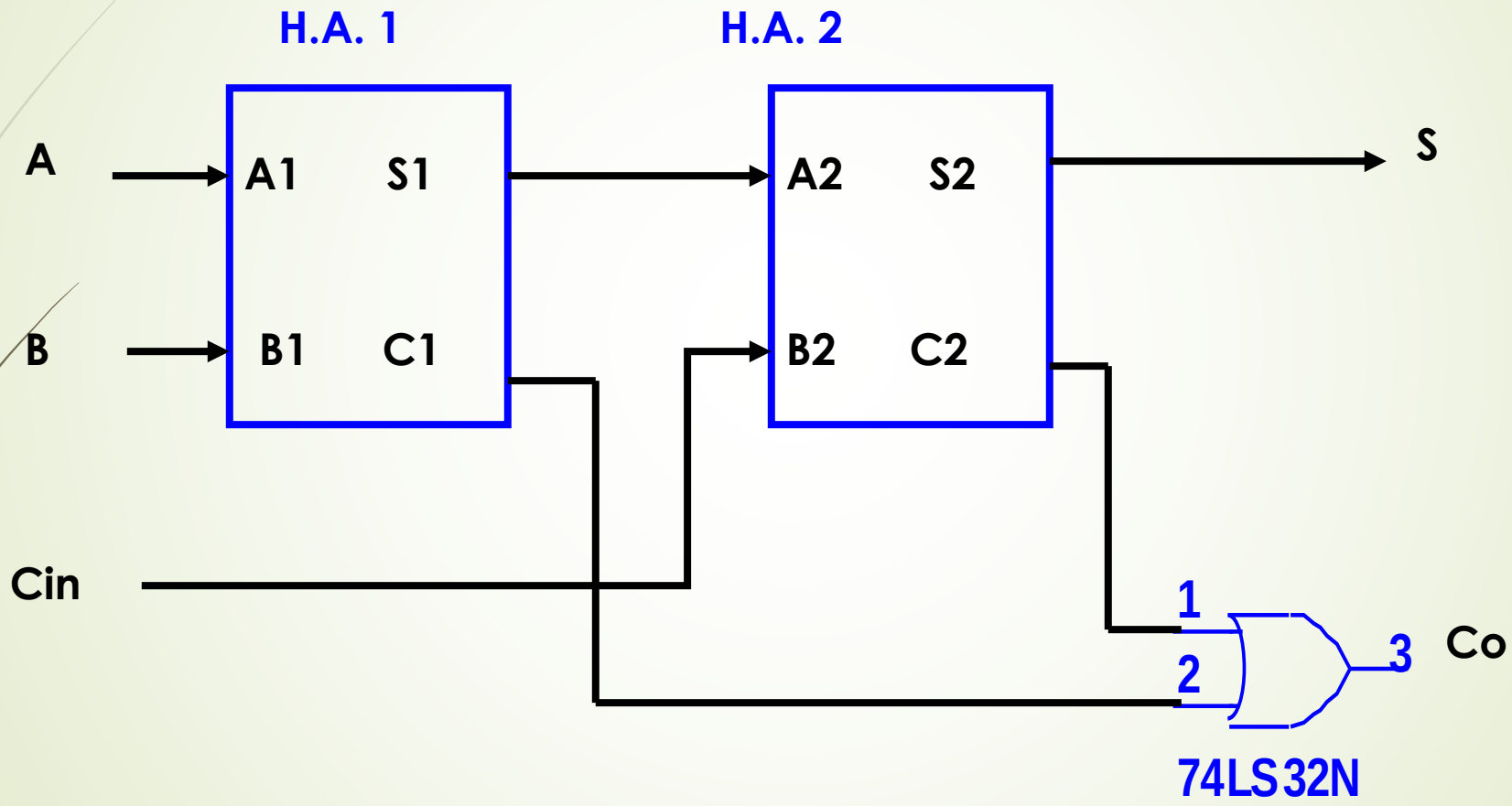
$$\text{Sum} = (A \oplus B) \oplus Cin$$

$$\text{Carry out} = A.B + B.Cin + A.Cin$$

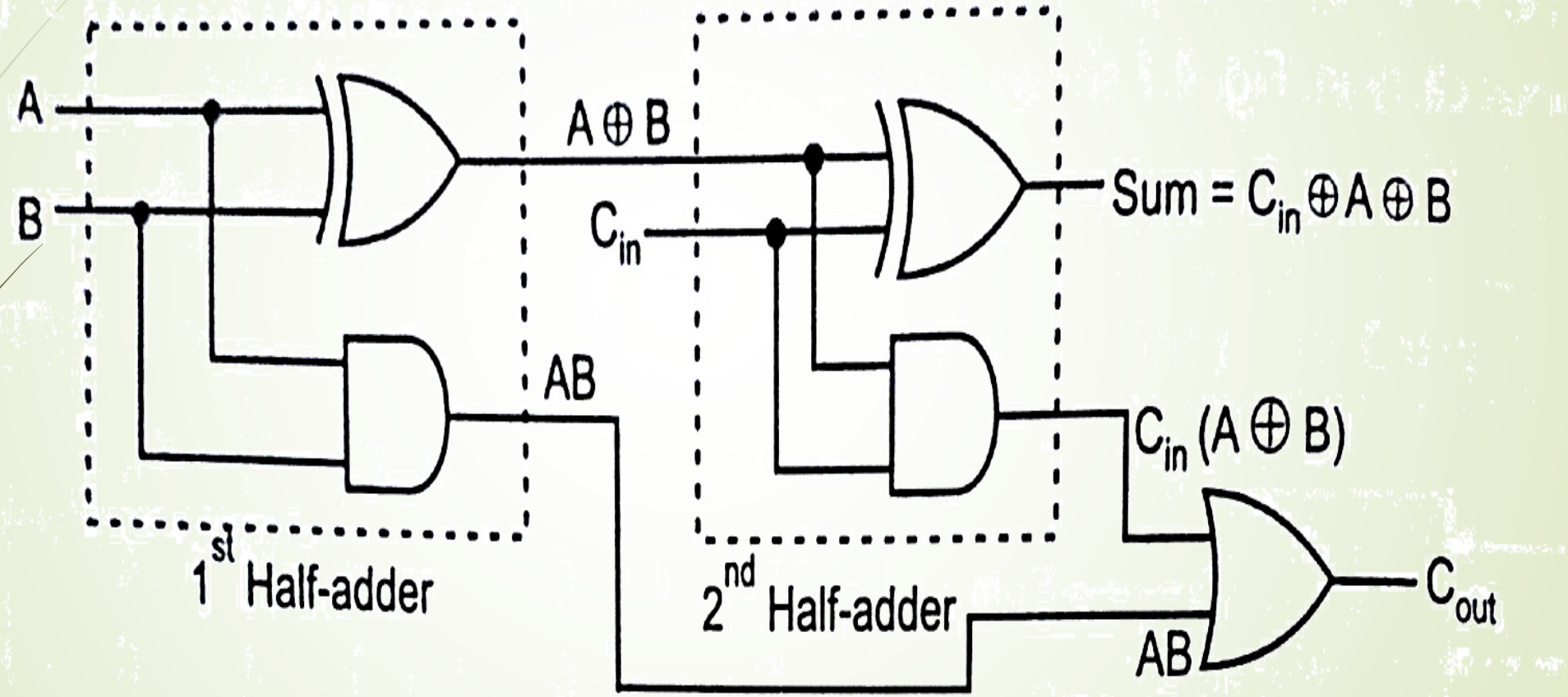
Logic Diagram For Full Adder



Full Adder using Half Adder:

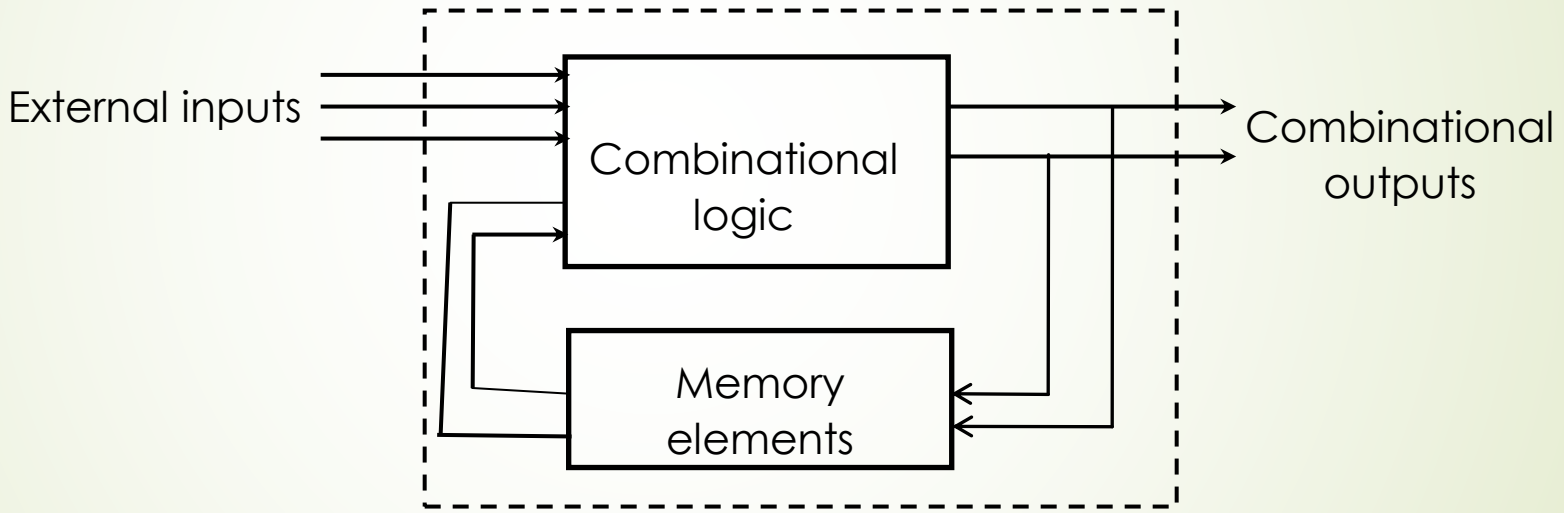


Full Adder Using Half Adder



Sequential Circuits

- **A sequential circuit consists of a feedback path, and employs some memory elements.**

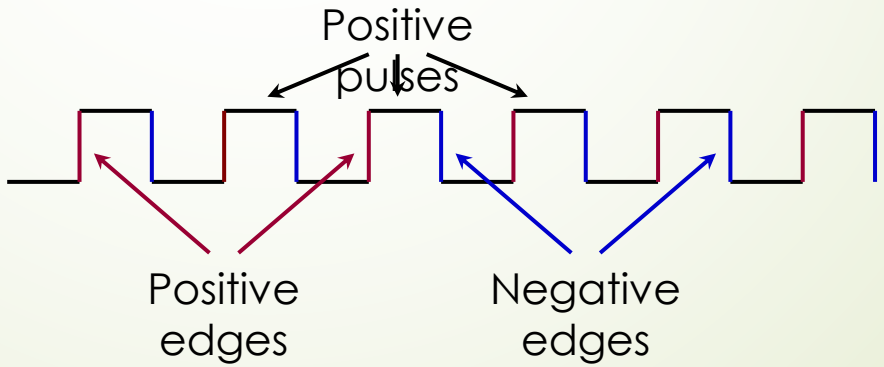


Sequential circuit = Combinational logic + Memory Elements

Memory Elements

- **It is a device which can remember value indefinitely, or change value on command from its inputs.**

- **Flip-flops are memory elements that change state on clock signals.**
- **Clock is usually a square wave.**



Memory Elements

Two types of triggering/ activation:

1. pulse-triggered
2. edge-triggered

1. Pulse-triggered:

1. latches
2. ON = 1, OFF = 0

2. Edge-triggered:

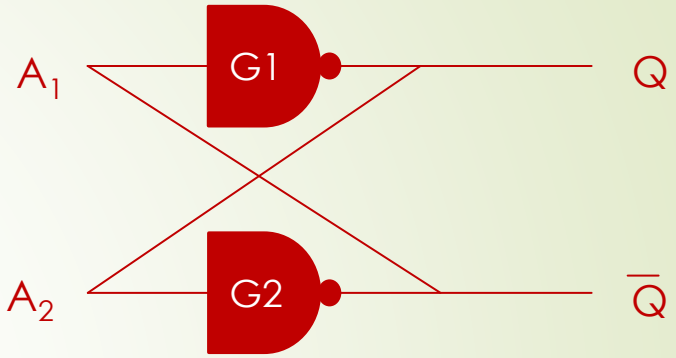
1. flip-flops
2. positive edge-triggered

(ON = from 0 to 1; OFF = other time)

3. negative edge-triggered

(ON = from 1 to 0; OFF = other time)

- **1 Bit Memory Cell :**

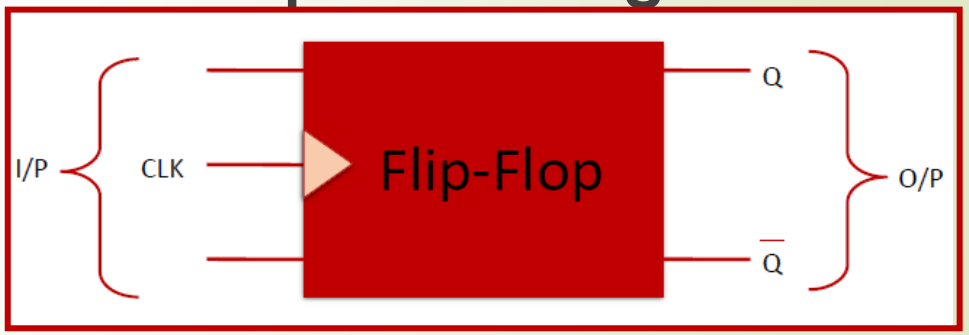


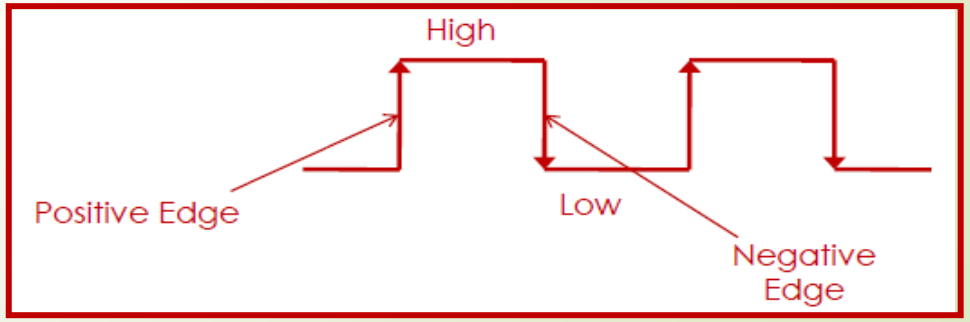
Cross Couple Inverter s as Memory Element

- The output Q and $\overline{Q}$ are always Complementary.
- The Circuit has two stable State.(i.e. Q=1 Set State and Q=0 Reset State).
- The Circuit continues to remain in the same state referred to as Memory.
- The information is latched or locked in this circuit ,so it is also referred as **Latch.**

Flip- Flops

- A flip-flop is a circuit that has two stable states and can be used to store state information.
- Flip-flop is a bistable logic device i.e. its outputs have two stable states.
- The state of the flip-flop is change only with the inputs and clock signal.
- The clock signal is only edge triggered clock. Either positive edge or negative edge triggered clock.
- Flip-flop is basic sequential circuit.
- Types of Flip-flops:
  1. S- R Flip-flops
  2. J- K Flip-flops
  3. D- Flip-flops
  4. T- Flip-flops

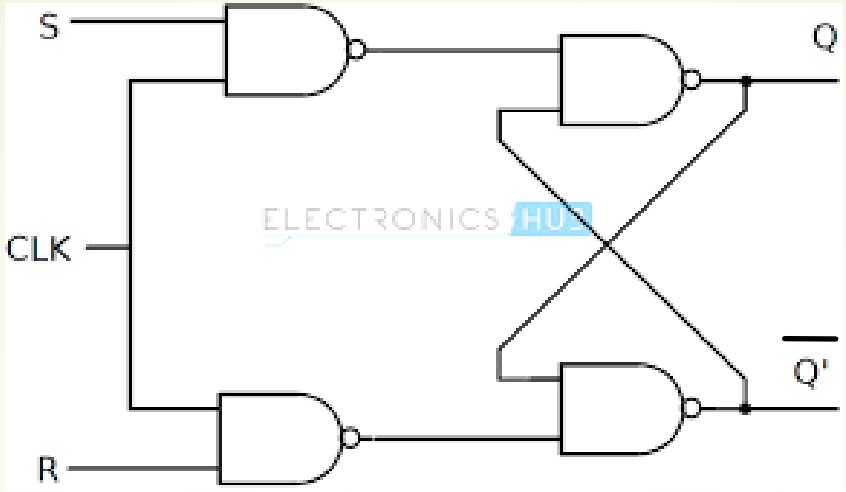




Flip-Flops

Types: 1. S – R Flip-Flop
2. J – K Flip-Flop
3. D Flip-Flop
4. T Flip-Flop

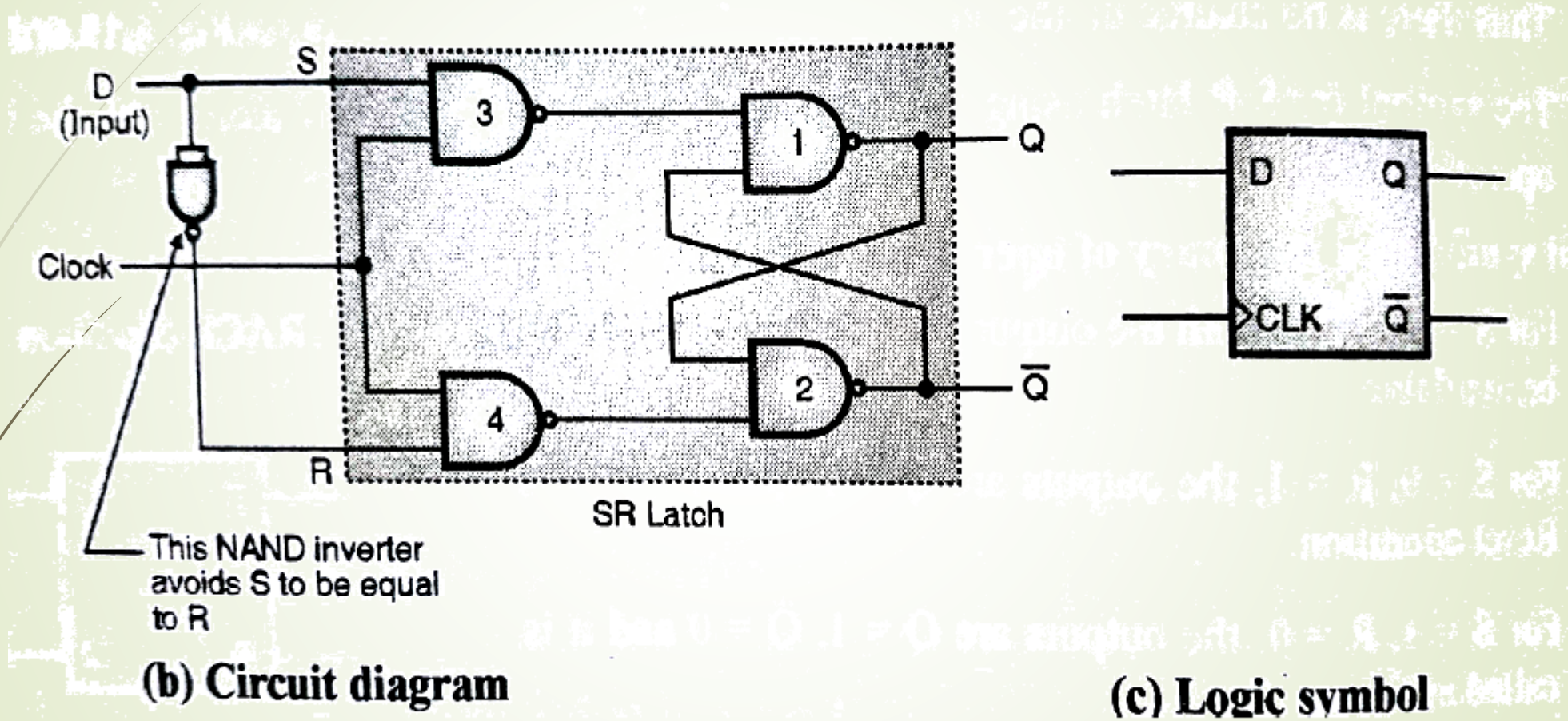
Basic SR Flip Flop (Using NAND Gate)



S	R	Qn+1	$\overline{Qn+1}$	
0	0	Qn	$\overline{Qn}$	NO CHANGE
0	1	0	1	Reset
1	0	1	0	Set
1	1	1	1	Race

Q. What do you mean by flip flop? Explain the operation of D-flip flop.

Positive Edge Triggered D- Flip Flop



Positive Edge Triggered D- Flip Flop

	Inputs		Outputs		State
	CLK	D	Q_{n+1}	$\bar{Q}_{n+1}$	
FF disabled	0	×	Q_n	$\bar{Q}_n$	No change
	1	×	Q_n	$\bar{Q}_n$	No change
	↓	×	Q_n	$\bar{Q}_n$	No change
FF responds only to the positive edges	↑	0	0	1	**Q follows D input**
	↑	1	1	0	

From the truth table it is clear that the Q output of the flip-flop follows the D input.

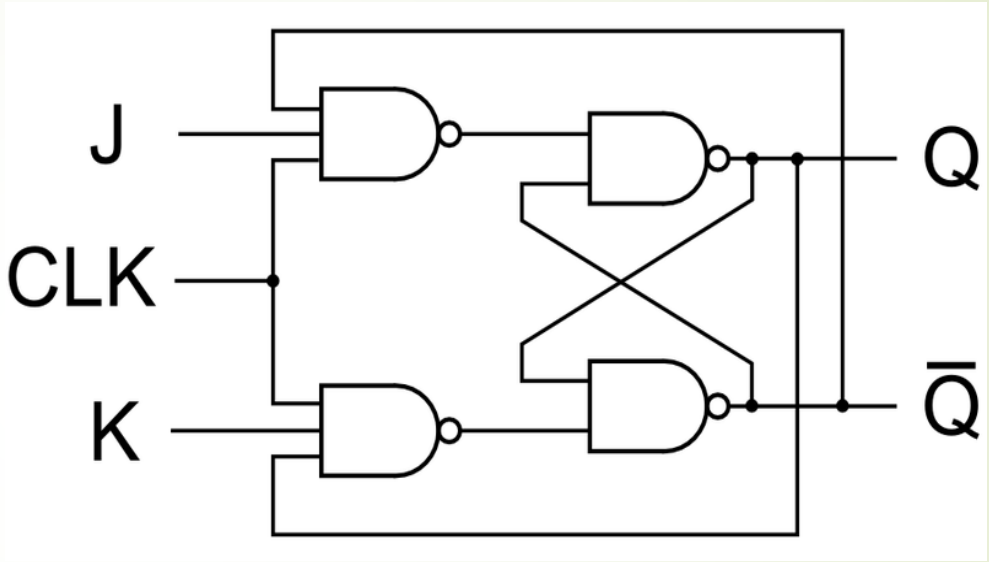
Negative Edge Triggered D- Flip Flop

	Inputs		Outputs		Comment
	CLK	D	Q_{n+1}	$\bar{Q}_{n+1}$	
FF Disabled	0	×	Q_n	$\bar{Q}_n$	No change
	1	×	Q_n	$\bar{Q}_n$	No change
	↑	×	Q_n	$\bar{Q}_n$	No change
FF responds only to the negative edges	↓	0	0	1	**Q output follows D input**
	↓	1	1	0	

JK FLIP-FLOP

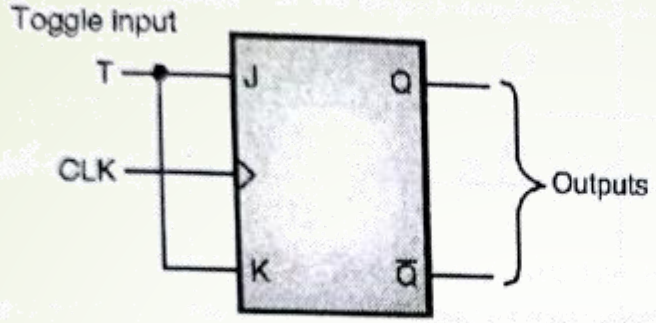
Four Modes Of Operation

The 4 modes of operation are: **hold, set, reset, toggle**

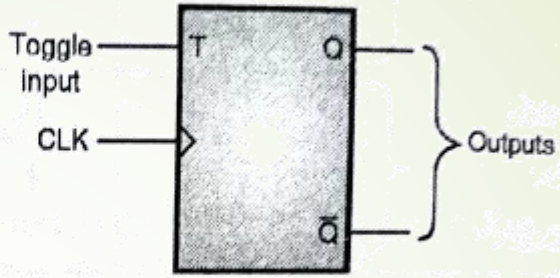


J	K	Q	Q'	Mode
0	0	Q	Q'	Hold
1	0	1	0	Sets
0	1	0	1	Resets
1	1	Q'	Q	Toggle

Positive Edge Triggered T- Flip-flops



(a) JK FF is converted into T flip-flop



(b) Logic symbol of positive edge triggered T flip-flop

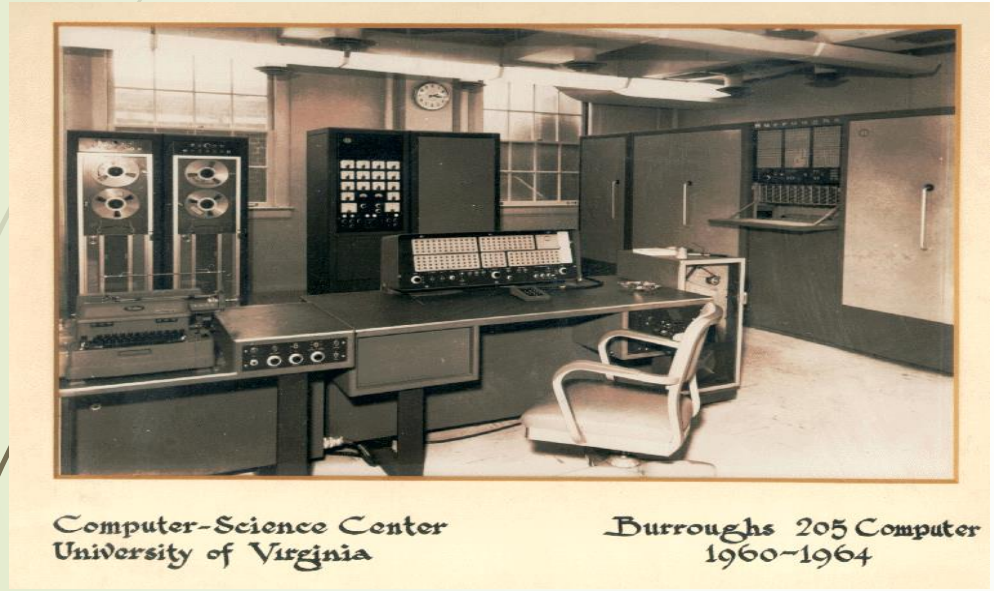
Inputs		Outputs		State
CLK	T	Q_{n+1}	$\bar{Q}_{n+1}$	
↑	0	Q_n	$\bar{Q}_n$	No change
↓	×	Q_n	$\bar{Q}_n$	No change
1	×	Q_n	$\bar{Q}_n$	No change
0	×	Q_n	$\bar{Q}_n$	No change
↑	1	$\bar{Q}_n$	Q_n	Toggle

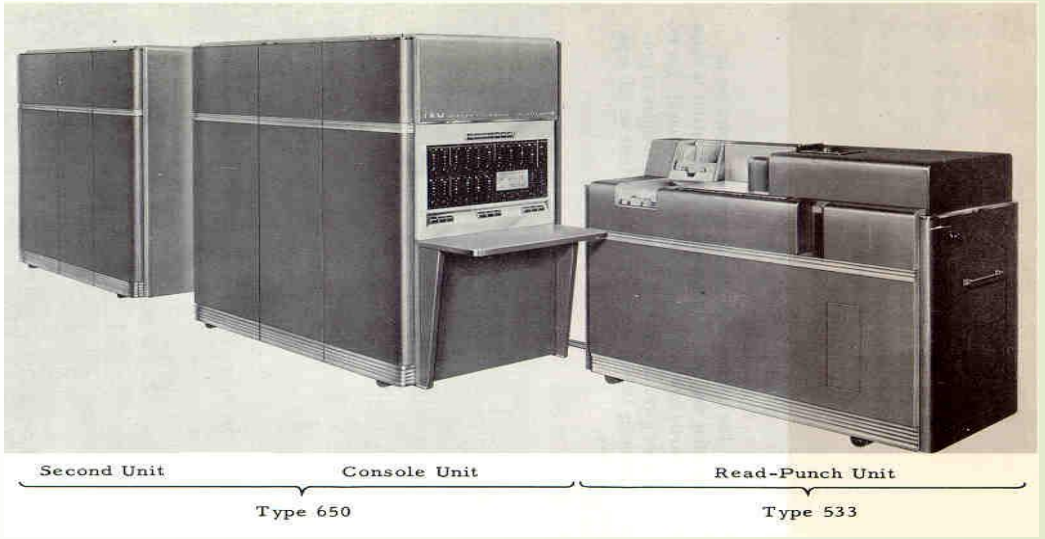
Evolution of Computers

- **First generation (1939-1954) - vacuum tube**
- **Second generation (1954-1959) - transistor**
- **Third generation (1959-1971) - IC**
- **Fourth generation (1971-present) - microprocessor**

Evolution of Computers

- First generation (1939-1954) - vacuum tube

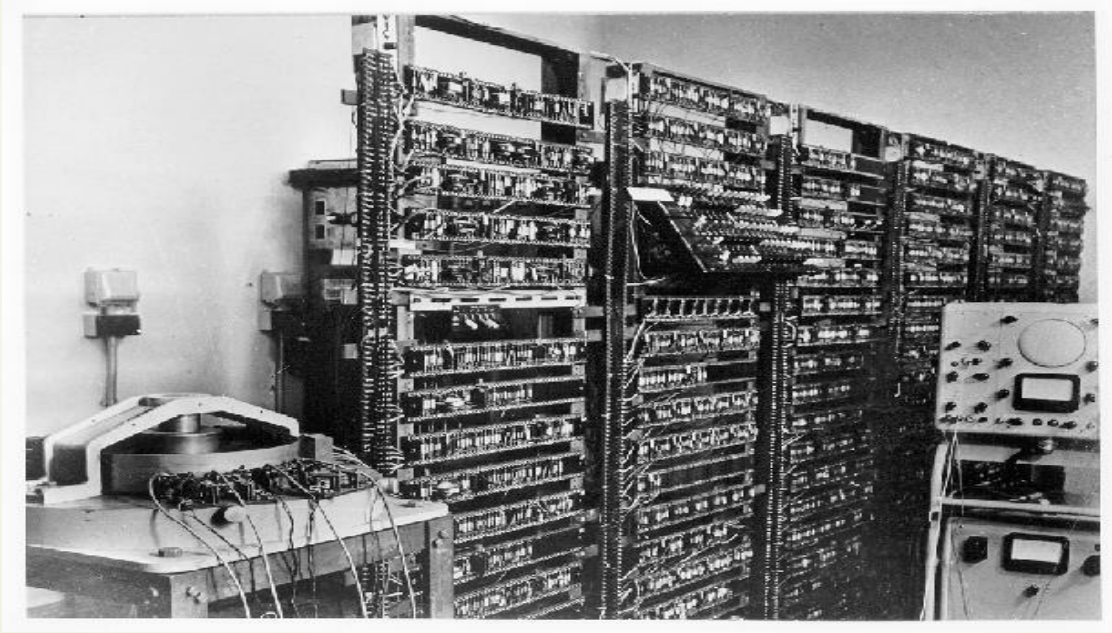




IBM 650, 1954

Evolution of Computers

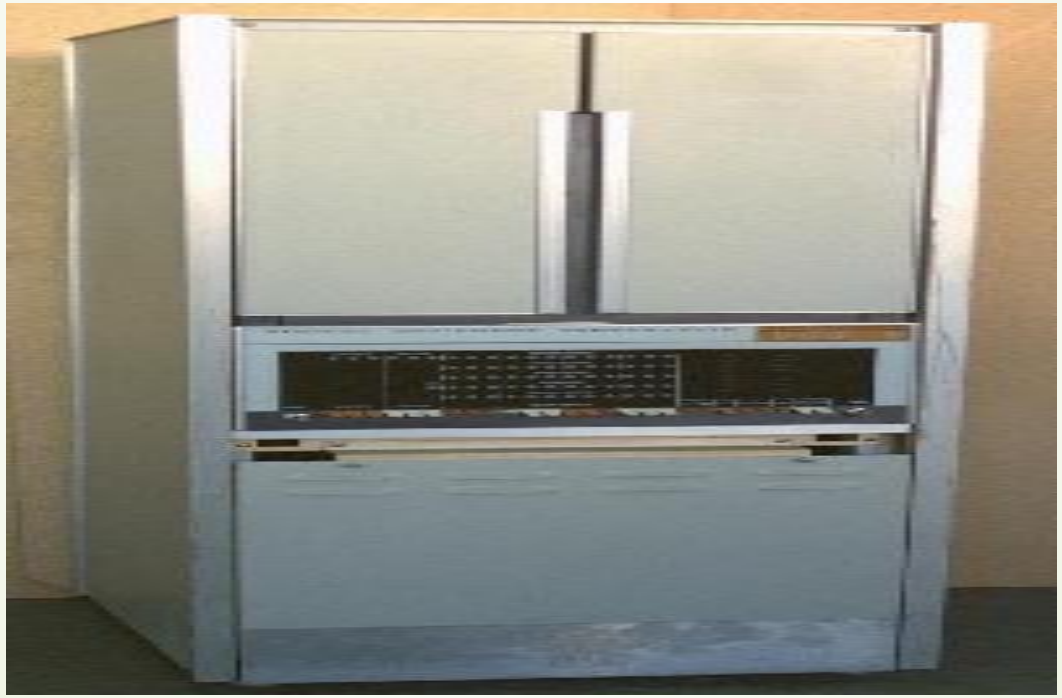
- **Second generation (1954-1959) - transistor**



Manchester University Experimental Transistor Computer

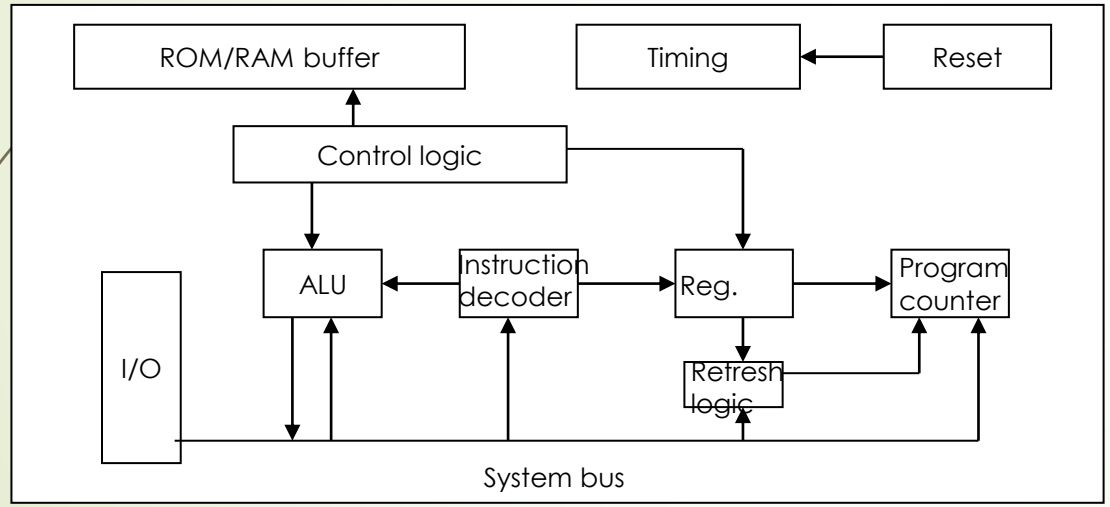
Evolution of Computers

- **Third generation (1959-1971) - IC**

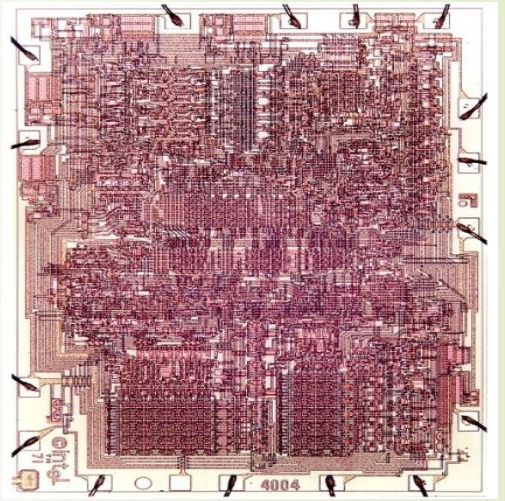


Evolution of Computers

- Fourth generation (1971-present) - microprocessor
  - In 1971, Intel developed 4-bit 4004 chip for calculator applications.



Block diagram of Intel 4004



http://www.intel.com

4004 chip layout

Digital Computers can be classified into three categories:

1.Microcomputer
2.Minicomputer
3.Maxi computer

1. **Microcomputer:** A **microcomputer** is a small, relatively inexpensive computer with a microprocessor as its central processing unit (CPU).It includes a microprocessor, memory and minimal input/output (I/O) circuitry mounted on a single printed circuit board (PCB).

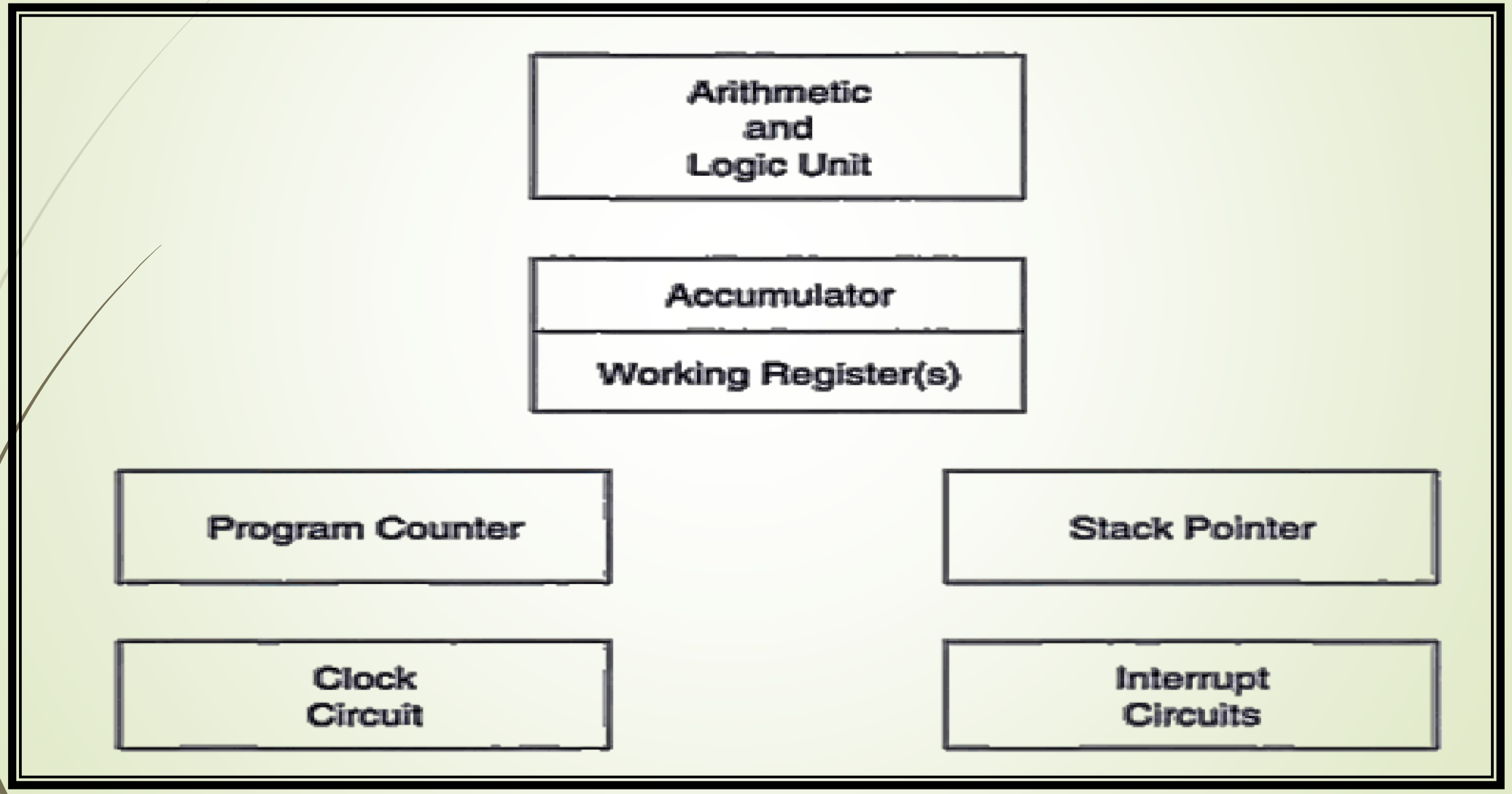
2. **Minicomputer:** A **minicomputer** is a type of **computer** that possesses most of the features and capabilities of a large **computer** but is smaller in physical size.

3. **Maxi Computer: Mainframes or Super computers. C**apable of handling data processing needs of, say, head office of a bank, or a big multinational company. Mainframe computer systems have larger storage and the speed of processing is also very high

Microprocessor

- Computer's Central Processing Unit (CPU) built on a **single Integrated Circuit (IC)** is called a **microprocessor**.
- It is a programmable, multipurpose, clock -driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as output.
- Fetch, Decode and Execute

Block Diagram of Microprocessor (MPU)



Fundamental Block Diagram of Microprocessor

Consists:

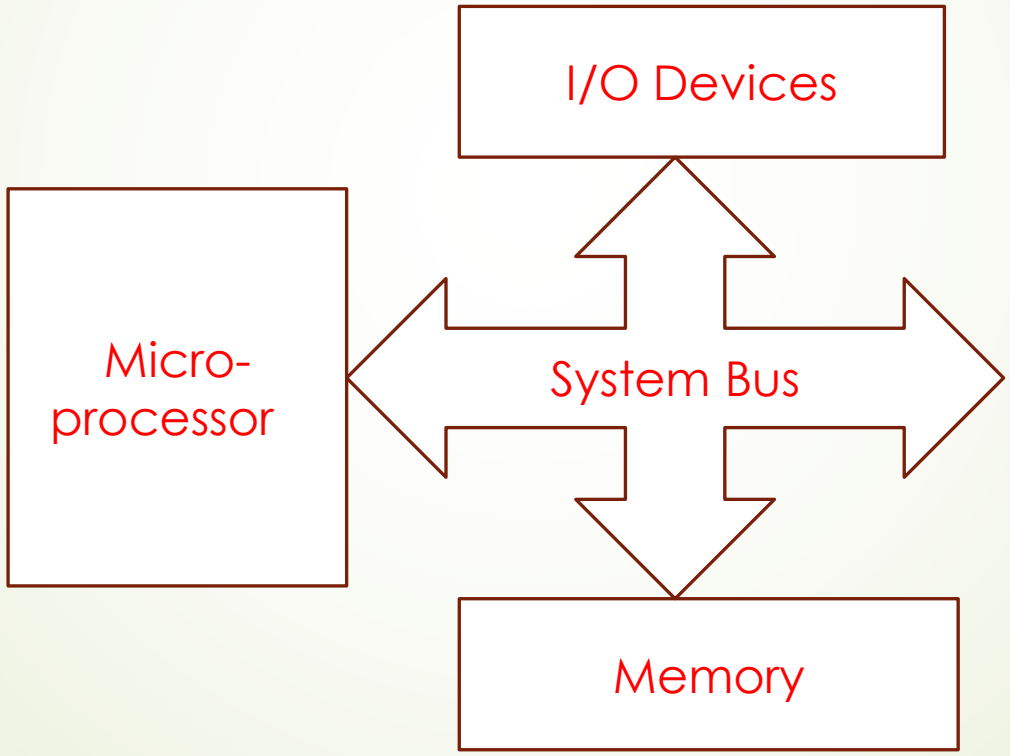
- **ALU(Arithmetic/Logic Unit):**
  - Unit which performs various computing functions on data.
  - Arithmetic operations like addition, subtraction and
  - Logic operations like AND, OR, Ex-OR.
- **Register Array:**
  - Consists various registers identified by letters B,C,D,E,H,L.
  - These registers are primarily used to store data temporarily during execution of program and accessible to user through instructions.

Fundamental Block Diagram of Microprocessor

- **Control Unit:**
  - Provides necessary timing & control signals to perform all the operations in Mp.
  - Controls flow of data among Mp, memory & peripherals.

System Bus:

- Is Communication path between Microprocessor & Peripherals
- This is nothing but group of wires to carry bits.



Communication among Microprocessor , Peripherals & Memory is performed

- ADDRESS BUS
- DATA BUS
- CONTROL BUS.

ADDRESS BUS:

- Unidirectional.

  -Bits flow in one direction from Microprocessor to Peripherals or memory.

- MP uses Address bus to perform first function i.e. to Identify peripheral or memory location(with its address).
- If address bus is 16-bit.
- It can address 65,536 memory locations i.e. 64KB memory.
- 8085 has 16 address lines
- 8088 has 20 address lines
- Pentium processor has 32 address lines

DATA BUS:

- Bidirectional.

  -Bits flow in both direction between MP & Peripherals or memory.
- MP uses data bus to perform second function i.e. to transfer binary data between MP & peripherals or memory locations.
- If data bus is 8-bit.
- It enables MP to manipulate 8-bit data ranging from 00 to FF i.e. 256 numbers.
- The largest number that can appear on data bus is 11111111 i.e. 255 in decimal.
- 8085 has 8 data lines
- 8086,Zilog Z8000, Motorola 68000 has 16 data lines
- Intel 80386/486 have 32 data lines

CONTROL BUS

- IS comprised of varies single lines that carry synchronization signals.

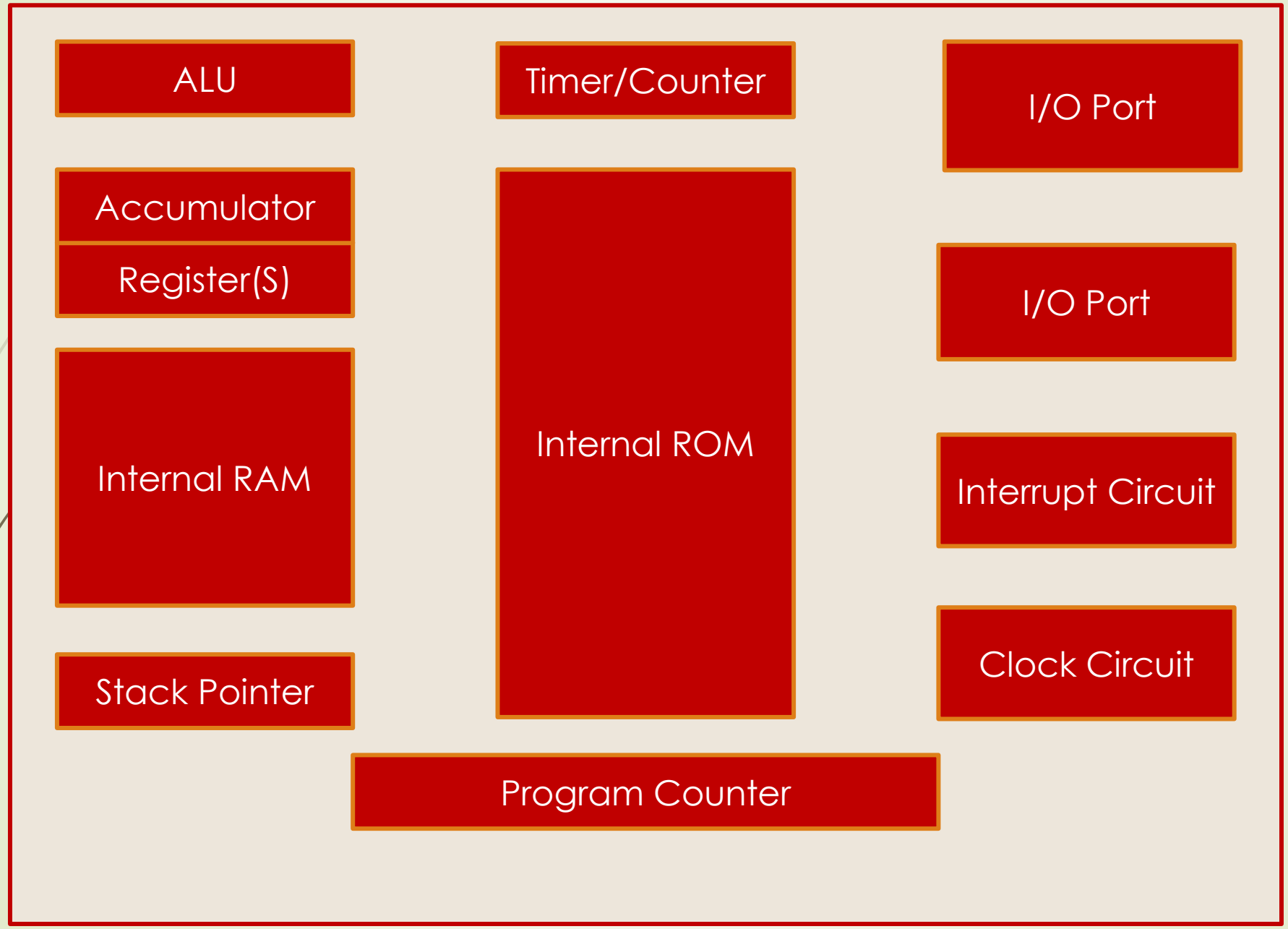
  -Bits flow in one direction from MP to Peripherals or memory.

- MP uses these lines to perform third function i.e. to provide timing signals.
- MP generates specific control signals for every operation.
- E.g. memory Read, memory Write, I/O Read, I/O Write.

- **Memory:**
  - Stores binary information as instructions & data & provides it to Mp whenever necessary.
  - Two sections;
    - ROM(Read only memory): stores program that do not needs alterations(e.g. Monitor program of microcomputer).
    - R/W M(Read/Write memory) or RAM(Random Access memory): stores user program & data. Stored information can be easily read & altered.
- I/O(Input / Output):
  - It communicates with outside world.
  - Two types of devices: Input and Output.
  - These are also known as **Peripherals**.

- **Input Devices:**
  - Keyboard, Switches & an ADC(Analog to Digital Convertor) etc.
  - Transfers data from out side world to Microprocessor .
- **Output Devices:**
  - LEDs, CRT, Video Screen, Printer, X-Y plotter, DAC(Digital to Analog Converter) etc.
  - Transfers data from Mp to out side world.

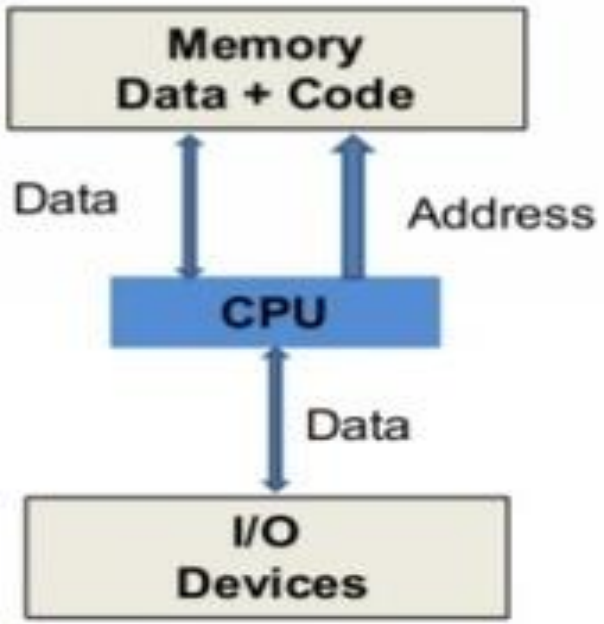
Block diagram of Microcontroller



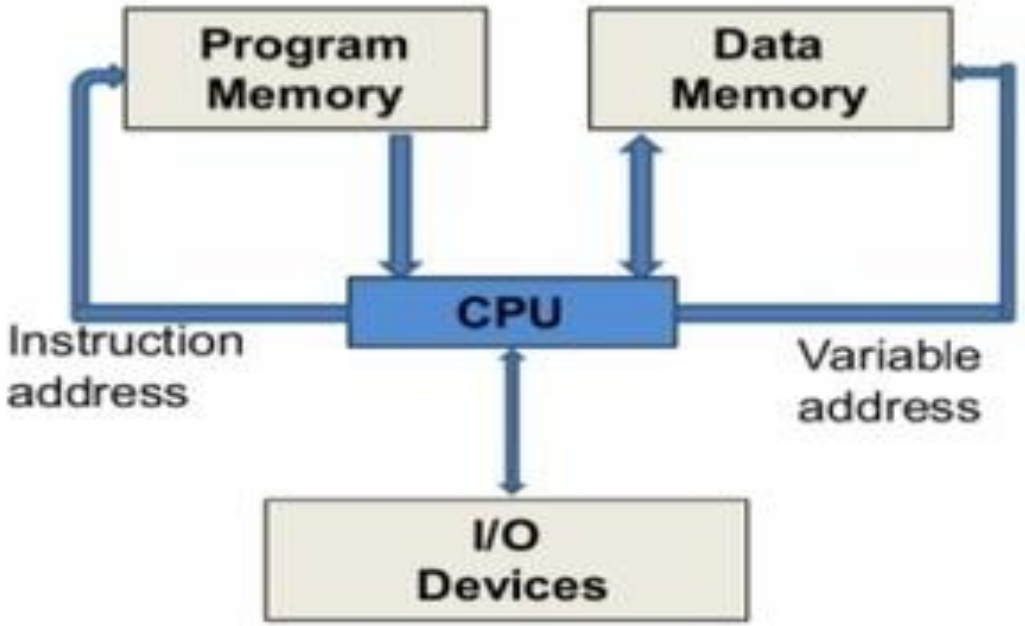
Microcontroller	Microprocessor
Inbuilt RAM or ROM	Do not have Inbuilt ROM or RAM
Inbuilt Timer	Do not have Inbuilt Timer
Input & Output Ports are available	Input & Output Ports are not available , Requires extra device like 8250 or 8255
Inbuilt Serial Port	Do not have Inbuilt Serial Port, Requires extra device like 8250 or 8255
Separate memory to store program & data	Program & data are stored in same memory
Many multifunction pins on the IC	Less multifunction pins on IC
Boolean operation is possible directly	Boolean operation is not possible directly
Few Instructions to read/ read data to / from external memory	Many Instructions to read/ read data to / from external memory
Microcontroller based System required Less Hardware	Microprocessor based System required more Hardware

Microcontroller	Microprocessor
Micro controllers are based on Harvard architecture where program memory and Data memory are separate.	Microprocessors are based on Von Neumann model / architecture where program and data are stored in same memory module.
Less access time for built in memory & Input - output devices	More access time for built in memory & Input – output devices
Offers Less flexibility in design	Offers more flexibility in design
System Cost is low	Expensive
Used mainly in washing machine, MP3 players, remote control, Traffic light control, toys, automobile engine control	Mainly used in personal computers
It is a Heart of Embedded system.	It is a Heart of Computer system.
Less processing power & High power consumption	High processing power & less power consumption
Typically 8/16 bit	Typically 32/64 – bit
Example : 8051, PIC Microcontroller	Examples :Intel’s 8085, 8086, 80386, 80486 etc.

Von Neumann vs. Harvard Architecture



Von Neumann Machine



Harvard Machine

Applications of Microprocessor & Microcontroller

- Embedded systems
- Data acquisition system
- Military Applications
- Calculators
- Personal Computers
- Laptops
- Processes Controller
- Communication systems

Number Systems

Scope

- Various Number Systems
- Inter-conversions between various number systems
- 1's and 2's representations
- Examples

Number Systems

- It is system to express a number or quantity with the help of various symbols.

Various Number Systems

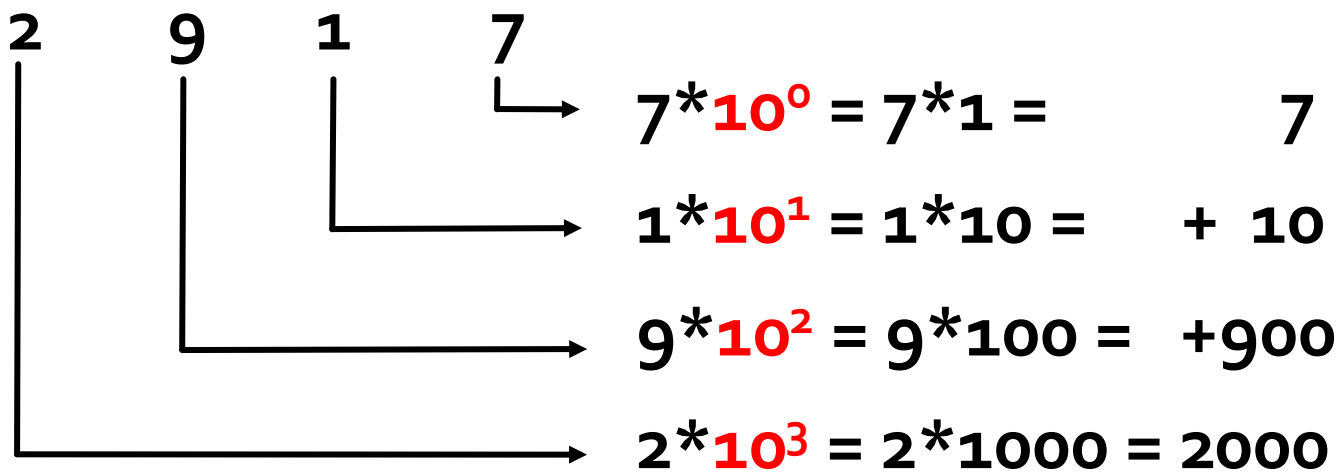
- Decimal
- Binary
- Octal
- Hexa-Decimal

Radix

- Its a number.
- It represents the total number of symbols used in a specific number system to represent the quantity.
- It gives the weightage of each digit in a given number.

Radix

- Radix = 10, for decimal number system
- 1) Total number of symbols = 10
  viz. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- 2) (2917) belongs to decimal number system



Number Systems

Sr. No.	Number Systems	Radix	Symbols Used
1	Binary	2	0, 1
2	Octal	8	0, 1, 2, 3, 4, 5, 6, 7
3	Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
4	Hexa-Decimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Hexa-Decimal Number System

- Radix = 16
- Symbols Used:
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Digits: 10 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  Letters: 6 A, B, C, D, E, F

Letters	A	B	C	D	E	F
Values	10	11	12	13	14	15

CONVERSIONS

Decimal

Binary/Octal/Hexa-D
(Integer & Fractions)

Binary/Octal/Hexa-D

Decimal
(Integer & Fractions)

Binary

Octal/Hexa-D

Octal/Hexa-D

Binary

Octal  Hexa-D

CONVERSIONS

Decimal Number System → Binary/Octal/Hexadecimal Number System

Decimal → **Binary/Octal/Hexa-D (Integer & Fractions)**

- **Integers: Successive Division By Radix (2/8/16)**
- **Fractions: Successive Multiplication By Radix (2/8/16)**

CONVERSIONS

Decimal Number System → Binary/Octal/Hexadecimal Number System

- **(Integer Part Only)**
- **Procedure**
  - **Step 1: Write only the Integer part of given decimal number**
  - **Step 2: Divide the given decimal successively by desired Base/Radix (2/8/16).**
  - **Step 3: Record Remainders and Quotients after every division.**
  - **Step 4: Record all remainders in reverse order**

CONVERSIONS

Decimal Number System ⟶ Binary Number System

- **Double Dabble Method:**

- **Example 1:** $(25)_{10} = (?)_2$

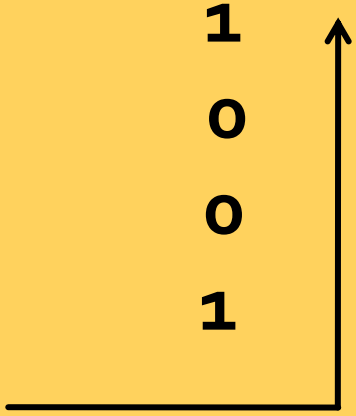
Step 1: Write the given decimal number's Integer part only

Step 2: Divide the given decimal successively by 2 till you get quotient as 1.

Step 3: Record Remainders after every division.

Step 4: All remainders taken in reverse order gives equivalent Binary Number.

2	25	1
2	12	0
2	6	0
2	3	1
2	1	



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

CONVERSIONS

Decimal Number System ⟶ Octal Number System

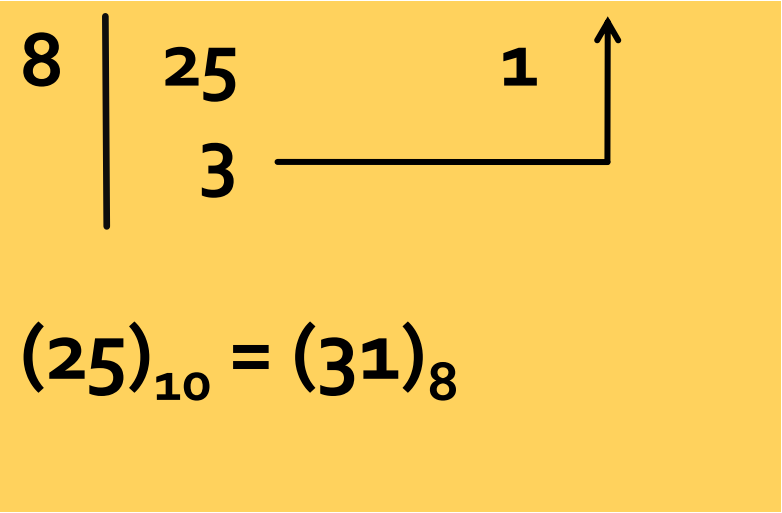
- Example 1: $(25)_{10} = (?)_8$

Step 1: Write the given decimal number's Integer part only

Step 2: Divide the given decimal successively by 8.

Step 3: Record Remainders after every division.

Step 4: All remainders taken in reverse order gives equivalent Octal Number.



$(25)_{10} = (31)_8$

CONVERSIONS

Decimal Number System ⟶ HexaDecimal Number System

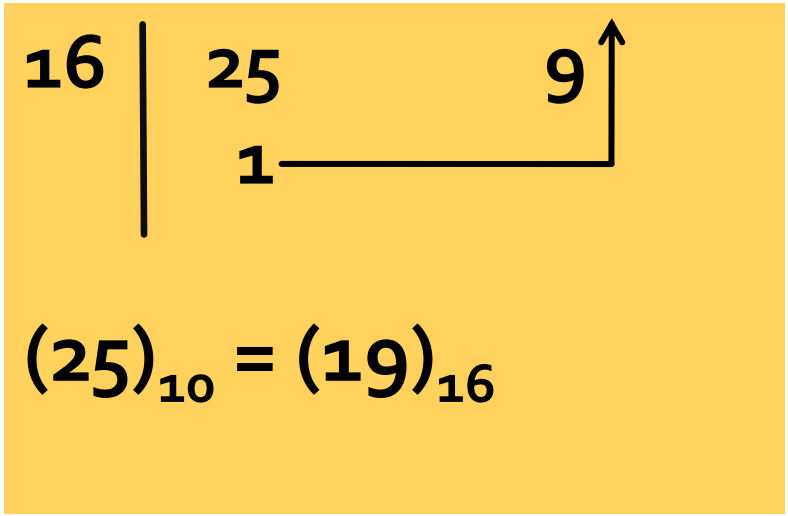
- **Example 1:** $(25)_{10} = (?)_{16}$

Step 1: Write the given decimal number's Integer part only

Step 2: Divide the given decimal successively by 16.

Step 3: Record Remainders after every division.

Step 4: All remainders taken in reverse order gives equivalent Octal Number.



$(25)_{10} = (19)_{16}$

CONVERSIONS

Decimal Number System ⟶ B/O/H Number System

- $(94)_{10} = (?)_{2}$
- $(94)_{10} = (?)_{8}$
- $(94)_{10} = (?)_{16}$

CONVERSIONS

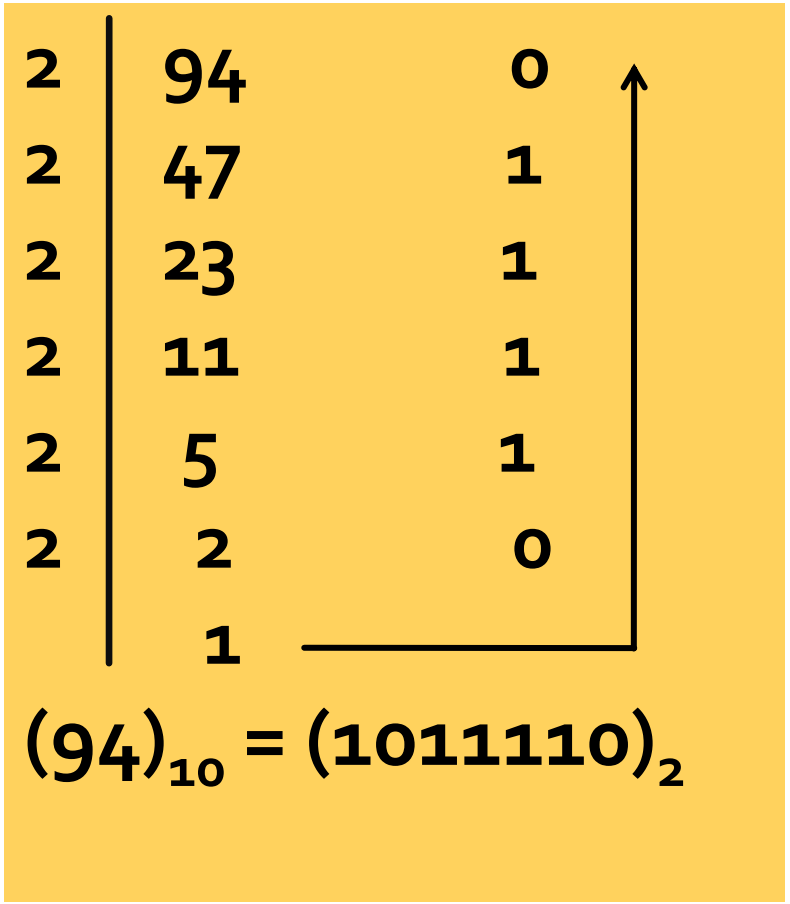
Decimal Number System ⟶ Binary Number System

- **Double Dabble Method:**

- **Example 4:**

- $(94)_{10} = (?)_2$

  $(94)_{10} = (1011110)_2$



CONVERSIONS

Decimal Number System ⟶ Octal Number System

- **Example 4:**
- $(94)_{10} = (?)_8$

  $(94)_{10} = (136)_8$

8	94	6
8	11	3
	1	

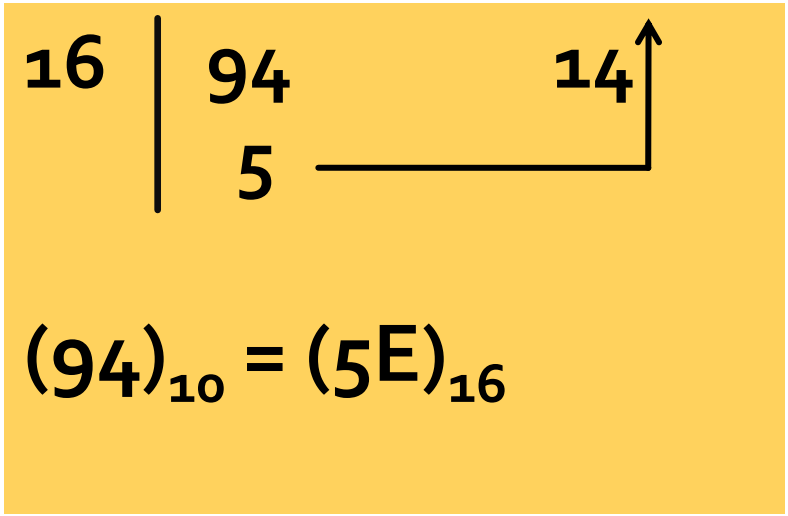
$(94)_{10} = (136)_8$

CONVERSIONS

Decimal Number System ⟶ Hexa Decimal Number System

- **Example 4:**
- $(94)_{10} = (?)_{16}$

  $(94)_{10} = (56)_{16}$



$(94)_{10} = (5E)_{16}$

CONVERSIONS

- **Exercise:**

* $(200)_{10} = (?)_2$
* $(10)_{10} = (?)_2$
* $(71)_{10} = (?)_2$
* $(29)_{10} = (?)_2$
* $(99)_{10} = (?)_2$
* $(40)_{10} = (?)_2$
* $(44)_{10} = (?)_2$
* $(401)_{10} = (?)_2$
* $(15)_{10} = (?)_2$
* $(129)_{10} = (?)_2$
* $(89)_{10} = (?)_2$
* $(13)_{10} = (?)_2$
* $(101)_{10} = (?)_2$

CONVERSIONS

Decimal Number System ⟶ Binary Number System

- **Double Dabble Method:**
- **Exercise:**

* $(200)_{10} = (11001000)_2$
* $(71)_{10} = (1000111)_2$
* $(99)_{10} = (1100011)_2$
* $(44)_{10} = (101100)_2$
* $(15)_{10} = (1111)_2$
* $(89)_{10} = (1011001)_2$
* $(101)_{10} = (1100101)_2$
* $(10)_{10} = (1010)_2$
* $(29)_{10} = (11101)_2$
* $(40)_{10} = (101000)_2$
* $(401)_{10} = (110010001)_2$
* $(129)_{10} = (10000001)_2$
* $(13)_{10} = (1101?)_2$

CONVERSIONS

Decimal Number System ⟶ Binary/Octal/Hexadecimal Number System

- **Fractions Part: Successive Multiplication**
- **Procedure**
  - **Step 1: Write only the Fractional part of given decimal number**
  - **Step 2: Multiply the given decimal (fraction) successively by desired Base/Radix (2/8/16).**
  - **Step 3: Record Integer part after every Multiplication and carry fraction part for further multiplication.**
  - **Step 4: All recorded Integers taken in Forward order gives Binary equivalent of given decimal Number**

CONVERSIONS

Decimal Number System ⟶ Binary Number System

- **Example 1:** $(0.45)_{10} = (?)_2$

- Step 1: Write only the Fractional part of given decimal number
- Step 2: Multiply the given decimal successively by 2.
- Step 3: Record Integer part after every Multiplication.
- Step 4: All recorded Integers taken in Forward order gives Binary equivalent of given decimal Number

$0.45 * 2 = 0.9$ 0

$0.9 * 2 = 1.8$ 1

$0.8 * 2 = 1.6$ 1

$0.6 * 2 = 1.2$ 1

$0.2 * 2 = 0.4$ 0 ↓

$(0.45)_{10} = (0.01110)_2$

CONVERSIONS

Decimal Number System → Octal Number System

- **Example 1:** $(0.45)_{10} = (?)_8$

- Step 1: Write only the Fractional part of given decimal number
- Step 2: Multiply the given decimal successively by 8.
- Step 3: Record Integer part after every Multiplication.
- Step 4: All recorded Integers taken in Forward order gives Binary equivalent of given decimal Number

$0.45*8 = 3.6$ 3 ↓
$0.6*8 = 4.8$ 4
$0.8*8 = 6.4$ 6
$0.4*8 = 3.2$ 3
$0.2*8 = 1.6$ 1

$(0.45)_{10} = (0.34631)_8$

CONVERSIONS

Decimal Number System ⟶ Hexa Decimal Number System

- **Example 1:** $(0.45)_{10} = (?)_{16}$

- Step 1: Write only the Fractional part of given decimal number
- Step 2: Multiply the given decimal successively by 8.
- Step 3: Record Integer part after every Multiplication.
- Step 4: All recorded Integers taken in Forward order gives Binary equivalent of given decimal Number

$0.45*16 = 7.2$ — 7
$0.2*16 = 3.2$ — 3
$0.2*16 = 3.2$ — 3
$0.2*16 = 3.2$ — 3
$0.2*16 = 3.2$ — 3

$(0.45)_{10} = (0.7333)_{16}$

CONVERSIONS

Decimal Number System ⟶ B/O/H Number System

- $(0.683)_{10} = (?)_{2}$
- $(0.683)_{10} = (?)_{8}$
- $(0.683)_{10} = (?)_{16}$

CONVERSIONS

Decimal Number System ⟶ Binary Number System

- **Example 2:**
- $(0.683)_{10} = (?)_2$
- $(0.683)_{10} = (0.10101)_2$

$0.683 * 2 = 1.366$ 1
$0.366 * 2 = 0.732$ 0
$0.732 * 2 = 1.464$ 1
$0.464 * 2 = 0.928$ 0
$0.928 * 2 = 1.856$ 1 ↓

$(0.683)_{10} = (0.10101)_2$

CONVERSIONS

Decimal Number System ⟶ Octal Number System

- **Example 2:**
- $(0.683)_{10} = (?)_8$
- $(0.683)_{10} = (0.53554)_8$

$0.683 * 8 = 5.464$ — 5

$0.\ 464 * 8 = 3.712$ — 3

$0.712 * 8 = 5.696$ — 5

$0.\ 696 * 8 = 5.568$ — 5

$0.\ 568 * 8 = 4.544$ — 4

$(0.683)_{10} = (0.53554)_8$

CONVERSIONS

Decimal Number System ⟶ Hexa Decimal Number System

- **Example 2:**
- $(0.683)_{10} = (?)_{16}$
- $(0.683)_{10} = (0.AED91)_{16}$

0.683*16 = 10.928 | 10 | A
0. 928*16 = 14.848 | 14 | E
0. 848*16 = 13.568 | 13 | D
0. 568*16 = 9.088 | | 9
0. 088*16 = 1.408 | | 1

$(0.683)_{10} = (0.AED91)_{16}$

CONVERSIONS

Decimal Number System ⟶ Binary Number System

- **Example 6:** $(35.567)_{10} = (?)_2$

Integer

2	35	1
2	17	1
2	8	0
2	4	0
2	2	0
	1	

$(35)_{10} = (100011)_2$

Fraction

0.567*2 = 1.134 — 1
0.134*2 = 0.268 — 0
0.268*2 = 0.536 — 0
0.536*2 = 1.072 — 1
0.072*2 = 0.144 — 0

$(0.\ 567)_{10} = (0.10010)_2$

$(35.567)_{10} = (100011.\ 10010)_2$

CONVERSIONS

Decimal Number System → Octal Number System

- **Example 6:** $(35.567)_{10} = (?)_8$

Integer

8	35	1
8	4	

$(35)_{10} = (41)_8$

Fraction

$0.567 * 8 = 4.536$ — 4

$0.536 * 8 = 4.288$ — 4

$0.288 * 8 = 2.304$ — 2

$0.304 * 8 = 2.432$ — 2

$0.432 * 8 = 3.456$ — 3

$(0.567)_{10} = (0.44223)_8$

$$(35.567)_{10} = (41.44223)_8$$

CONVERSIONS

Decimal Number System ⟶ Hexa D Number System

- **Example 6:** $(35.567)_{10} = (?)_{16}$

Integer

16	35	3
16	2	

$(35)_{10} = (23)_{16}$

Fraction

$0.567 * 16 = 9.072$ — 9

$0.536 * 16 = 1.152$ — 1

$0.288 * 16 = 2.432$ — 2

$0.304 * 16 = 6.912$ — 6

$0.432 * 16 = 14.592$ — E

$(0.567)_{10} = (0.9126E)_{16}$

$$(35.567)_{10} = (23.9126E)_{16}$$

CONVERSIONS

Binary Number System ⟶ Decimal Number System

- **Step 1:** Write the given binary number.
- **Step 2:** Write the binary weightage below each number

Binary Point

Binary Weightage	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Decimal Value	128	64	32	16	8	4	2	1	0.5	0.25	0.125

- **Step 3:** Cancel the weightage, which is placed below zero because any number multiplied by zero is zero.
- **Step4:** Add the remaining numbers.

CONVERSIONS

Binary Number System ⟶ Decimal Number System

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0 .	2^{-1}	2^{-2}	2^{-3}
Binary Weightage											
Decimal Value	128	64	32	16	8	4	2	1	0.5	0.25	0.125

2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
1	1	0	1	.	1	0	1
$1*2^3$ =8	$1*2^2$ =4	$0*2^1$ =0	$1*2^0$ =1	.	$1*2^{-1}$ =0.5	$0*2^{-2}$ =0	$1*2^{-3}$ =0.125

(1*8) + (1*4) + (0) + (1*1) . (1*0.5)+ 0 +(1*0.125)

8 + 4+ 0 + 1 . 0.5 + 0 + 0.125

$(1101.101)_2 = (13.625)_{10}$

CONVERSIONS

Octal Number System ⟶ Decimal Number System

	8^4	8^3	8^2	8^1	8^0 .	8^{-1}	8^{-2}	8^{-3}
Octal Weightage								
Decimal Value	4096	512	64	8	1	0.125	0.0156	0.0019

8^3	8^2	8^1	8^0	.	8^{-1}	8^{-2}	8^{-3}
5	6	3	2	.	4	7	1
$5*8^3$	$6*8^2$	$3*8^1$	$2*8^0$	.	$4*8^{-1}$	$7*8^{-2}$	$1*8^{-3}$

(5*512)+(6*64)+(3*8)+(2*1) . (4*0.125)+(7*0.0156)+ (1*0.0019)

= 2560 + 384 + 24 + 2 . 0.5 + 0.1092 + 0.0019

$(5632.471)_8 = (2970.6112)_{10}$

CONVERSIONS

Hexa-D Number System ⟶ Decimal Number System

	16^3	16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}
Hexa Decimal Weightage							
Decimal Value	4096	256	16	1	0.0625	0.0039	0.000244

	16^2	16^1	16^0	.	16^{-1}	16^{-2}	16^{-3}
	3	F	D	.	8	4	0
	$3*16^2$	$15*16^1$	$13*16^0$	.	$8*16^{-1}$	$4*16^{-2}$	

$(3*256)+(15*16)+(13*1)$. $(8*0.0625)+(4*0.0039)+0$

$= (768) + (240) + (13)$. $(0.5) + (0.0156) + 0$

$(3FD.84)_{16} = (1021.515625)_{10}$

Hexa-Decimal Number System

- Radix = 16
- Symbols Used:
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Digits: 10 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  Letters: 6 A, B, C, D, E, F

A	B	C	D	E	F
10	11	12	13	14	15

CONVERSIONS

Decimal

Binary/Octal/Hexa-D
(Integer & Fractions)

Binary/Octal/Hexa-D

Decimal
(Integer & Fractions)

Binary

Octal/Hexa-D

Octal/Hexa-D

Binary

Octal  Hexa-D

Binary Representation

Decimal value	Weighted binary value			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

CONVERSIONS

Binary Number System ⟶ Octal Number System

- Step 1: Make group of 3 bits
- Step 2: Find their octal values.
- Example 1 $(010101010110111)_2 = ()_8$
- Step 1: 0 101 010 110 111
- 0 101 010 110 111
- 0 5 2 6 7
- $(010101010110111)_2 = (5267)_8$

CONVERSIONS

Binary Number System ⟶ Octal Number System

- **Step 1:** Make group of 3 bits
- **Step 2:** Find their octal values.
- Example 2 $(010101101.011100)_2 = ()_8$
- **Step 1:** 010 101 101 . 011 100
- 010 101 101 . 011 100
- 
- 2 5 5 . 3 4
- $(010101101.011100)_2 = (255.34)_8$

CONVERSIONS

Octal System ⟶ Binary Number System

- **Procedure**

- **Step 1:** Convert each Octal digit into a 3 bit binary code

Example 1 $(136)_8 = ()_2$

- 1 = ⟹ 001
- 3 ⟹ 011
- 6 ⟹ 110
- $(136)_8 = (001\ 011\ 110)_2$

CONVERSIONS

Binary Number System ⟶ Hexa Decimal System

- **Step 1:** Make group of 4 bits
- **Step 2:** Find their Hex values.
- Example 1 $(0\ 1010\ 1011\ 0111)_2 = ()_{16}$
- **Step 1:** 0 1010 1011 0111

Incomplete	Grp 3	Grp 2	Grp 1
0000	1010	1011	0111
Complete	Grp 3	Grp 2	Grp 1

CONVERSIONS

Binary Number System ⟶ Hexa Decimal System

- Step 2: Find their Hex values
- Hex Values:
- 0000 1010 1011 0111
- 0 10 11 7
- 10, 11: Invalid Symbols in Hexa Decimal System
- 10: A 11: B
- 0000 1010 1011 0111
- 0 A B 7
- $(0101010110111)_2 = (0AB7)_{16}$

CONVERSIONS

Hexa Decimal System ⟶ Binary Number System

- **Procedure**
- **Step 1:** Convert each hex digit into a 4 bit binary code

Example 1 $(C39)_{16} = ()_2$

- C = 12 ⟹ 1100
- 3 ⟹ 0011
- 9 ⟹ 1001
- $(C39)_{16} = (1100\ 0011\ 1001)_2$

CONVERSIONS

Hexa Decimal System ⟶ Binary Number System

Step 1: Convert each hex digit into a 4 bit binary code

Example 2 $(AB01)_{16} = ()_2$

- A = 10 ⟹ 1010
- B = 11 ⟹ 1011
- 0 ⟹ 0000
- 1 ⟹ 0001
- $(C39)_{16} = (1010\ 1011\ 0000\ 0001)_2$

CONVERSIONS

Octal Number System ⟶ Hexa Decimal System

- Step 1: Convert Octal to its Binary Equivalent
- Step 2: Convert Binary to its Hexa decimal Equivalent.
- Example 1 $(615.25)_8 = ()_{16}$
- Step 1: Convert each Octal digit into a 3 bit binary code
- Step 2: Make group of 4 bits starting from LSB and write their HEX equivalents
- Step1:

- 6 ⟶ 110
- 1 ⟶ 001
- 5 ⟶ 101
- 2 ⟶ 010
- 5 ⟶ 101

$(615.25)_8 = (110\ 001\ 101\ .\ 010\ 101)_2$

CONVERSIONS

Octal Number System ⟶ Hexa Decimal System

- $(615.25)_8 = (110\ 001\ 101\ .\ 010\ 101)_2$
- Step 2: Convert Binary to its Hexa decimal Equivalent
- $(110\ 001\ 101\ .\ 010\ 101)_2 = 110001101.010101$
- 1 1000 1101 . 0101 01
- 0001 1000 1101 . 0101 0100
- 1 8 13 . 5 4
- 1 8 D . 5 4
- $(615.25)_8 = (18D.54)_{16}$

CONVERSIONS

Hexa Decimal Number ⟶ Octal Number System

- Step 1: Convert Hexa Decimal Number to its Binary Equivalent
- Step 2: Convert Binary to its Octal Equivalent.
- Example 1 $(786.13)_{16} = ()_8$
- Step 1: Convert each HexD digit into a 4 bit binary code
- Step 2: Make group of 3 bits starting from LSB for integer part and MSB for fractional part by adding '0's at the end if required. Write octal equivalent for each group.
- Step1:

- 7 ⟶ 0111
- 8 ⟶ 1000
- 6 ⟶ 0110
- 1 ⟶ 0001
- 3 ⟶ 0011

$$(786.13)_{16} = (0111\ 1000\ 0110\ .\ 0001\ 0011)_2$$

CONVERSIONS

Hexa Decimal Number ⟶ Octal Number System

- $(786.13)_{16} = (0111\ 1000\ 0110\ .\ 0001\ 0011)_2$
- Step 2: Convert Binary to its Octal Equivalent
- $(0111\ 1000\ 0110\ .\ 0001\ 0011)_2 =$
- 0111 1000 0110 . 0001 0011
- 011 110 000 110 . 000 100 110
- 3 6 0 6 . 0 4 6
- $(786.13)_{16} = (3606.046)_8$

BCD: Binary Coded Decimal

- BCD: easy conversion process for higher decimal numbers.
- In BCD, each digit in a given decimal number is coded separately

Binary

EX: $(15)_{10}$

Double dabble

$(15)_{10} = (1111)_2$

BCD

EX: $(15)_{10}$

$(1)_{10}: (0001)_2$

$(5)_{10}: (0101)_2$

$(15)_{10} = (0001\ 0101)_{BCD}$

BCD: Binary Coded Decimal

Binary		BCD	
Decimal	Binary	Decimal	BCD
1	0001	1	0001
2	0010	2	0010
3	0011	3	0011
4	0100	4	0100
5	0101	5	0101
6	0110	6	0110
7	0111	7	0111
8	1000	8	1000
9	1001	9	1001

BCD: Binary Coded Decimal

Binary		BCD	
Decimal	Binary	Decimal	BCD
11	1011	11	0001 0001
12	1100	12	0001 0010
13	1101	13	0001 0011
14	1110	14	0001 0100
15	1111	15	0001 0101
and so on.....		and so on...	

BCD: Binary Coded Decimal

- **Advantages of BCD**

Easy conversion to and from decimal number system.

Exercise

Convert following decimal numbers into their BCD representation

$23\ (0010\ 0011)_{BCD}$, $67\ (0110\ 0111)_{BCD}$,

$911\ (1001\ 0001\ 0001)_{BCD}$,

$124\ (0001\ 0010\ 0100)_{BCD}$,

$671\ (0110\ 0111\ 0001)_{BCD}$

1'S & 2'S Representations

- In Decimal number system, negative numbers are represented using ' - ' symbol/notation.
- Binary system has only two valid symbols ie '0' and '1'.
- No other symbols are there to represent negative number/signed numbers.

1'S & 2'S Representations

- 1's and 2's representations are used to represent signed numbers for Binary numbers.
- These transformations are performed on Binary numbers.

1'S & 2'S Representations

- 1's Complement
- Method:
- Replace '0' by '1' and '1' by '0'
- It is represented by overhead bar
- Example 1:
- if A = $(10101110)_2$, Find 1's complement of A.
- $\bar{A} = (01010001)_2$

1'S & 2'S Representations

- Example 2:
- if A = $(1110010)_2$, Find 1's complement of A.
- $\bar{A} = (0001101)_2$

1'S & 2'S Representations

- 2's Complement
- Method:
- 2's complement = 1's complement + 1
- If A is the binary number then $\bar{A}$ is the 1's complement and Á is 2's complement of A
- $Á = \bar{A} + 1$

1'S & 2'S Representations

- Example 1:
- if A = $(1110010)_2$, Find 2's complement of A.
- 1's complement of A, $\bar{A} = (0001101)_2$
- 2's complement of A, Á = Ā+1

- Á = Ā+1
- 0001101
- +1
- 0001110
- Á = 0001110

1'S & 2'S Representations

- Example 2:
- if A = $(11001100)_2$, Find 2's complement of A.
- 1's complement of A, $\bar{A} = (00110011)_2$
- 2's complement of A, Á = $\bar{A}$+1

- Á = $\bar{A}$+1
- 00110011
- +00000001
- 00110100
- Á = 00110100

Information

- Binary Number System
- Single Digit: Bit
- A Group of 4 Bits: Nibble
- A Group of 8 Bits: Byte
- A Group of 16 Bits: Word
- A Group of 32 Bits: Double Word
- LSB: Least Significant Bit
- MSB: Most Significant Bit

Alternative Short cuts

2's complement

$A = (1101101010)_2$

Start from RHS

Look for first 1.

Till first 1, write all digits as they are

After first one , invert all left over bits

$A = (1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0)_2$

$Á = (0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0)_2$

Alternative Short cuts

2's complement

$A = (1100001001)_2$

Start from RHS

Look for first 1.

Till first 1, write all digits as they are

After first one , invert all left over bits

$A = (1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1)_2$

$Á = (0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1)_2$

Alternative Short cuts

2's complement

$A = (100000)_2$

Start from RHS

Look for first 1.

Till first 1, write all digits as they are

After first one , invert all left over bits

$A = (1\ 0\ 0\ 0\ 0\ 0)_2$

$\acute{A} = (1\ 0\ 0\ 0\ 0\ 0)_2$

Alternative Short cuts

2's complement

$A = (001110101)_2$

Start from RHS

Look for first 1.

Till first 1, write all digits as they are

After first one , invert all left over bits

$A = (0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1)_2$

$Á = (1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1)_2$

Alternative Short cuts

Here, Basically we are taking required powers of 2 such that their addition becomes the given number

Ex1: $15 = 8+4+2+1 = (1111)_2$

$25 = 16+8+1 = 16+8+0+0+1 = (11001)_2$

$30 = 16+8+4+2 = 16+8+4+2+0 = (11110)_2$