Computer System Architecture

CSEG1014

School of Computer Science
University of Petroleum and Energy Studies
Bidholi, Dehradun, Uttarakhand - 248007

ABOUT ME: DR. MOHAMMAD AHSAN

- PhD – National Institute of Technology Hamirpur (Himachal Pradesh).
- M.Tech – National Institute of Technology Hamirpur (Himachal Pradesh).
- Qualified UGC NET June-2015 and UGC NET Nov-2017 for Assistant Professor.
- Qualified GATE 2012, GATE 2013 and GATE 2021.
- Teaching Experience: NIT Andhra Pradesh, NIT Hamirpur, and UPES.

COURSE OBJECTIVES

- To develop understanding of Computer Models and its usage.
- To develop understanding of ALU Design.
- To conceptualize the understanding of Control Unit design, Memory, IPC, Control Design.
- To develop understanding of Memory & Input/output organization Overview.

COURSE CONTENT

- **UNIT 1. Introduction**
  - Evolution of Computer Systems, Von Neumann Architecture, Moore's Law, Computer Types, Functional Units, Devices (Input, Output, Storage & Communication Devices), Memory System (RAM, ROM, Cache, VM, etc.), Introduction to Logic Gates, Truth Table, K-Map, Latch Flip Flops (J, K & D), Encoder & Decoder, MUX & DEMUX, Registers & Counters, Binary Number system, Overview of RISC/CISC, RISC vs. CISC.
- **UNIT 2. ALU Design**
  - Computer Organization and Design, Instruction Codes, Op-Code, Computer registers, Computer Instructions, CPU stack Organization, Instruction Formats, Instruction types, Timing and control, Instruction and Instruction sequencing, Instruction Cycle, Memory Reference Instructions, Addressing modes, Program Control, Types of Interrupts, Adder & Subtractor.
- **Unit 3. Control Unit Design**
  - Introduction, Instruction Interpretation & Execution, Control Transfer, Fetch Cycle, Micro programmed Control, Control Memory, Micro programmed vs. Hardwired Control Unit, Nano Programming, Superscalar processing.

COURSE CONTENT

- **Unit 4. Memory Organization**
  - Memory Locations & Addresses, Semiconductor Memory, Static and Dynamic Memory, Main Memory, Auxiliary Memory, Associative Memory, Cache Memory, Secondary Memories: Optical Magnetic Tape, Magnetic Disk and Controllers.
- **Unit 5. Input/Output Organization**
  - I/O and their brief description, Bus Interface, Bus arbitration, Data Transfer, Types of Interrupts, I/O Interrupts, Channels, Direct Memory Access, I/O processing.

COURSE CONTENT

SUGGESTED READINGS:

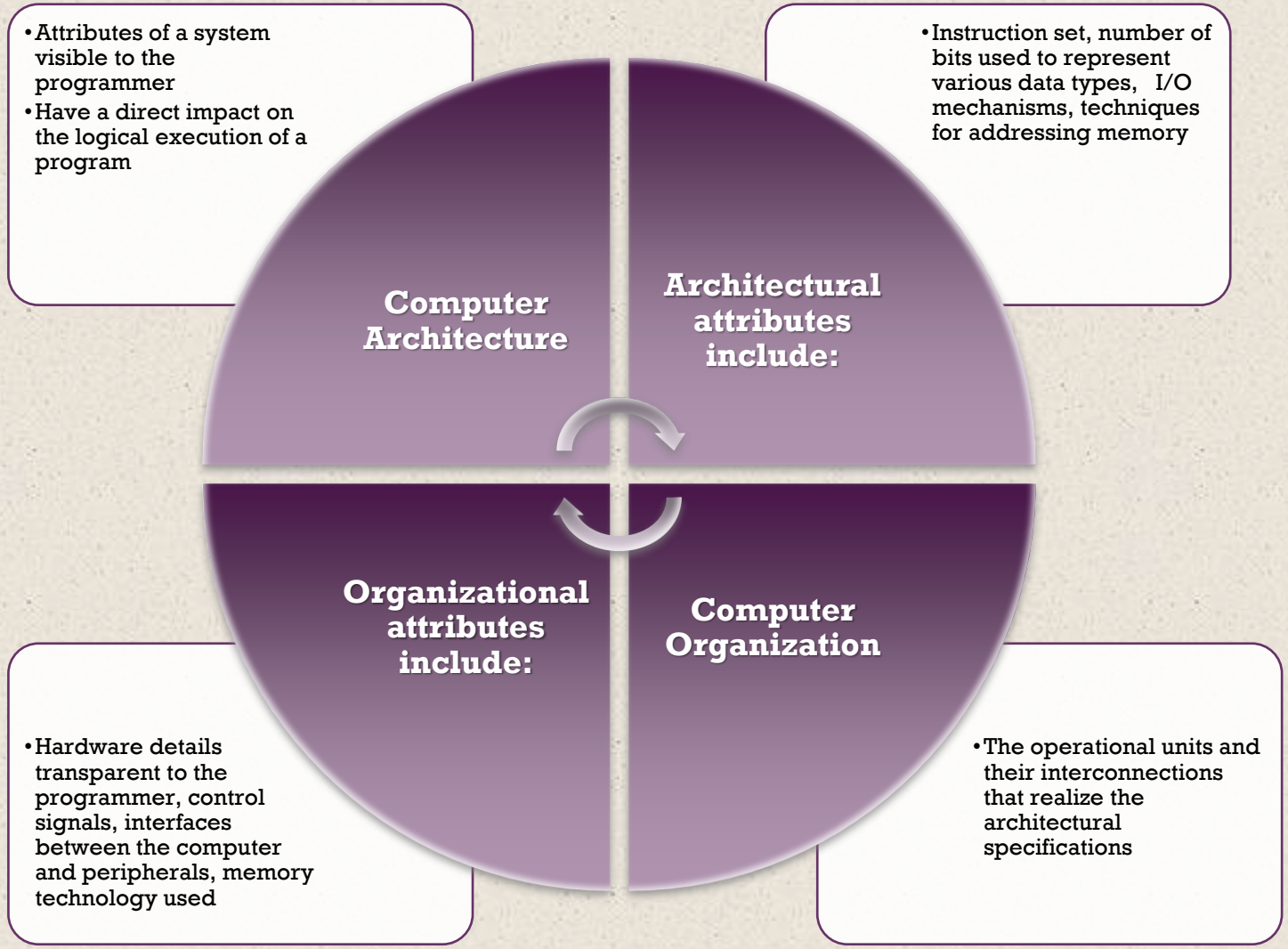
- **Text Books**
  1. “Computer System Architecture”, 3rd edition, M. Morris Mano, Pearson Publications.
- **Reference Books**
  1. “Computer Organization and Architecture”, Sixth Edition, William Stallings, Pearson Publications.
  2. “Fundamental of Digital electronics", second edition, A. Anand Kumar, PHI publications
  3. “Computer Organization and Architecture”, Third Edition, John P. Hayes, TATA McGraw-Hill.

Unit 1

Introduction

Computer Architecture

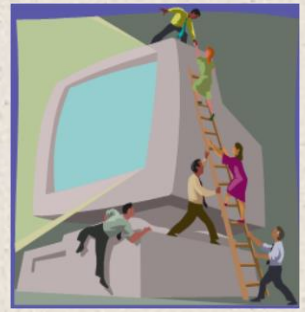
Computer Organization



Structure and Function

- Hierarchical system
  - Set of interrelated subsystems
- Hierarchical nature of complex systems is essential to both their design and their description
- Designer need only deal with a particular level of the system at a time
  - Concerned with structure and function at each level

- Structure
  - The way in which components relate to each other
- Function
  - The operation of individual components as part of the structure



Function

- There are four basic functions that a computer can perform:
  - Data processing
    - Data may take a wide variety of forms and the range of processing requirements is broad
  - Data storage
    - Short-term
    - Long-term
  - Data movement
    - Input-output (I/O) - when data are received from or delivered to a device (peripheral) that is directly connected to the computer
    - Data communications – when data are moved over longer distances, to or from a remote device
  - Control
    - A control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions

Structure

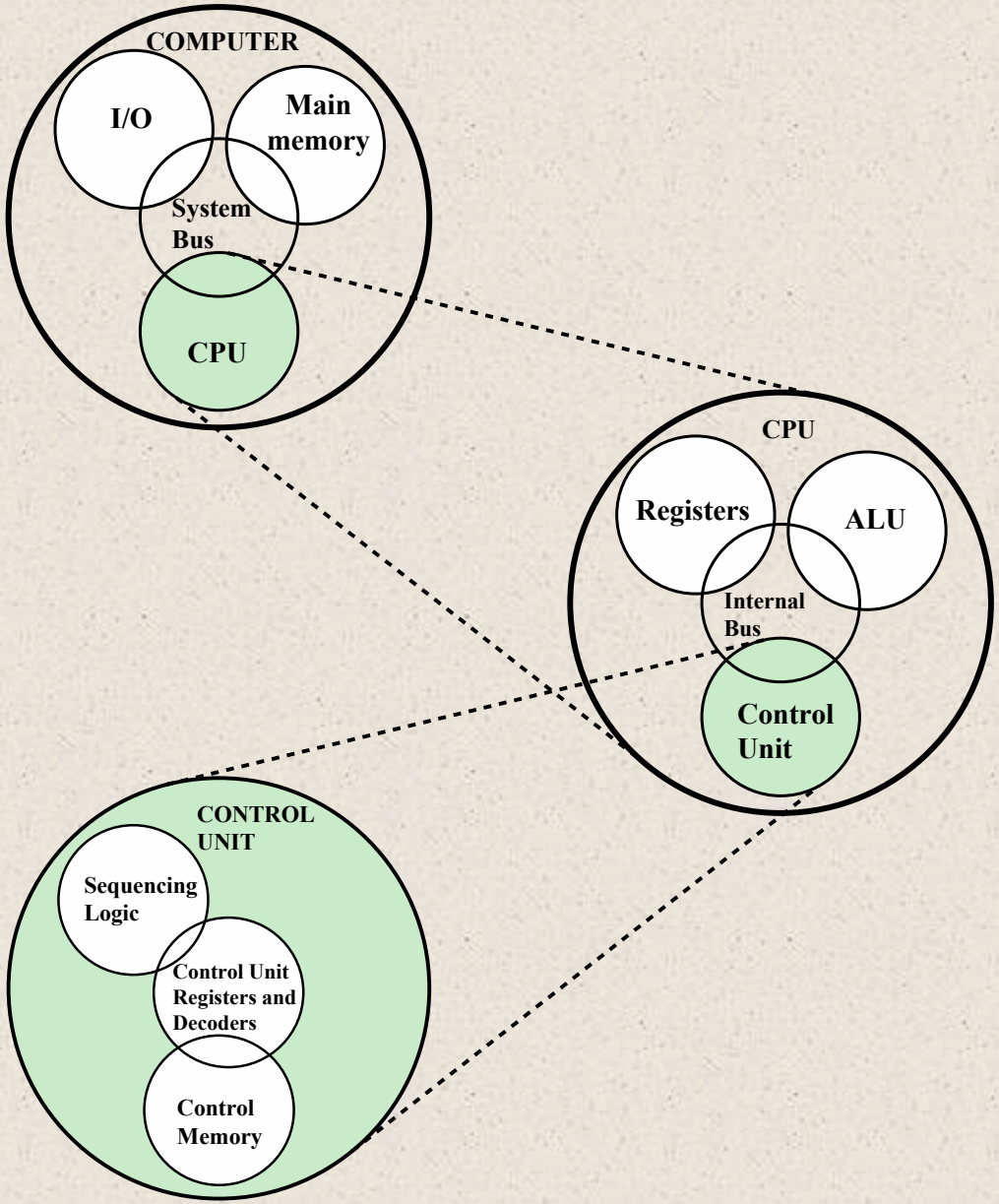
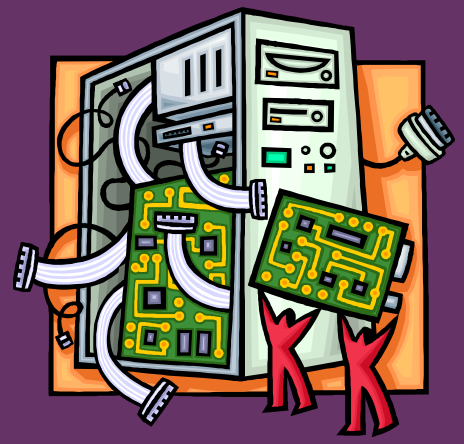


Figure 1.1 A Top-Down View of a Computer

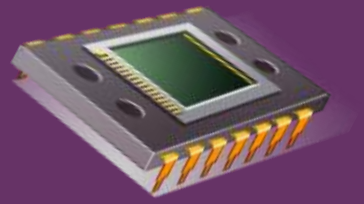
There are four main structural components of the computer:

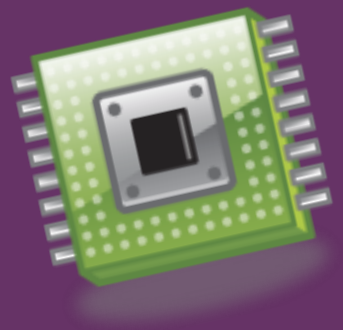


- CPU – controls the operation of the computer and performs its data processing functions
- Main Memory – stores data
- I/O – moves data between the computer and its external environment
- System Interconnection – some mechanism that provides for communication among CPU, main memory, and I/O

CPU

Major structural components:





- Control Unit
  - Controls the operation of the CPU and hence the computer
- Arithmetic and Logic Unit (ALU)
  - Performs the computer's data processing function
- Registers
  - Provide storage internal to the CPU
- CPU Interconnection
  - Some mechanism that provides for communication among the control unit, ALU, and registers

Cache Memory

- Multiple layers of memory between the processor and main memory
- Is smaller and faster than main memory
- Used to speed up memory access by placing in the cache data from main memory that is likely to be used in the near future.
- A greater performance improvement may be obtained by using multiple levels of cache, with level 1 (L1) closest to the core and additional levels (L2, L3, etc.) progressively farther from the core

Multicore Computer Structure

- Central processing unit (CPU)
  - Portion of the computer that fetches and executes instructions
  - Consists of an ALU, a control unit, and registers
  - Referred to as a processor in a system with a single processing unit
- Core
  - An individual processing unit on a processor chip
  - May be equivalent in functionality to a CPU on a single-CPU system
  - Specialized processing units are also referred to as cores
- Processor
  - A physical piece of silicon containing one or more cores
  - Is the computer component that interprets and executes instructions
  - Referred to as a *multicore processor* if it contains multiple cores

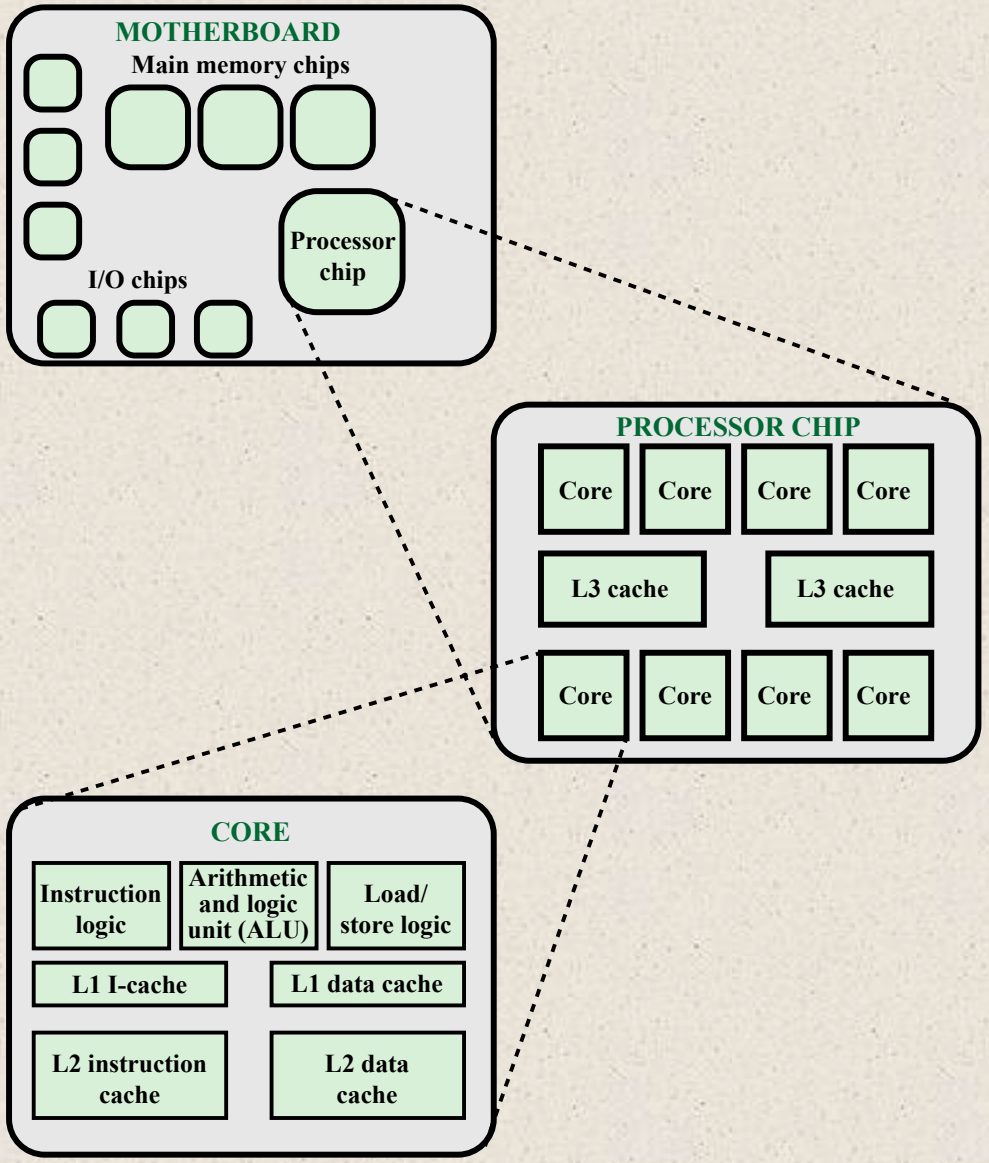
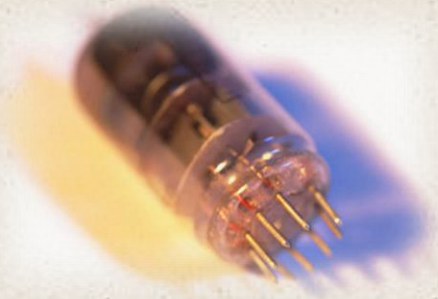


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

History of Computers

First Generation: Vacuum Tubes



- Vacuum tubes were used for digital logic elements and memory
- IAS computer
  - Fundamental design approach was the stored program concept
    - Attributed to the mathematician John von Neumann
    - First publication of the idea was in 1945 for the EDVAC
  - Design began at the Princeton Institute for Advanced Studies
  - Completed in 1952
  - Prototype of all subsequent general-purpose computers

- A Von Neumann Architecture

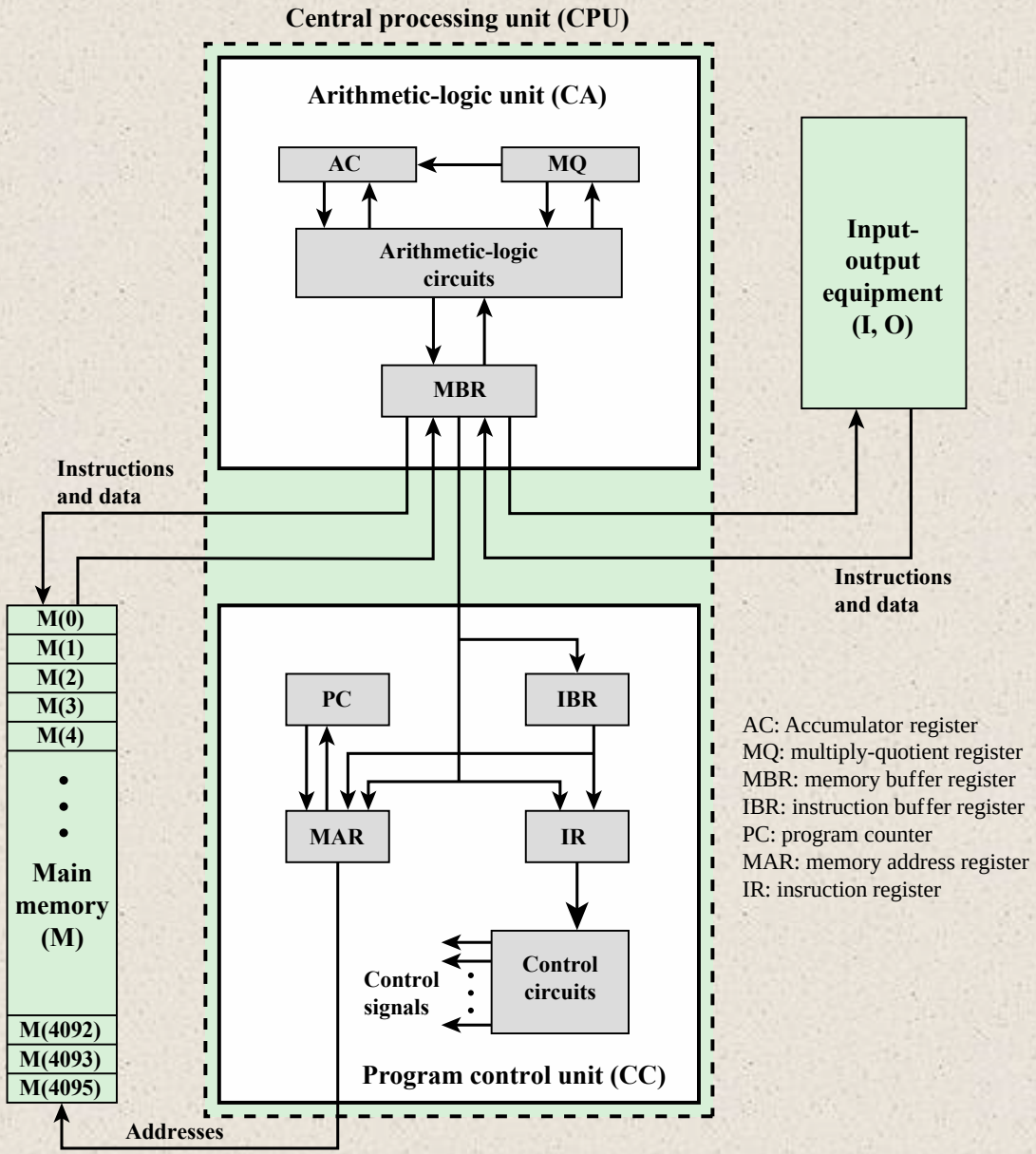
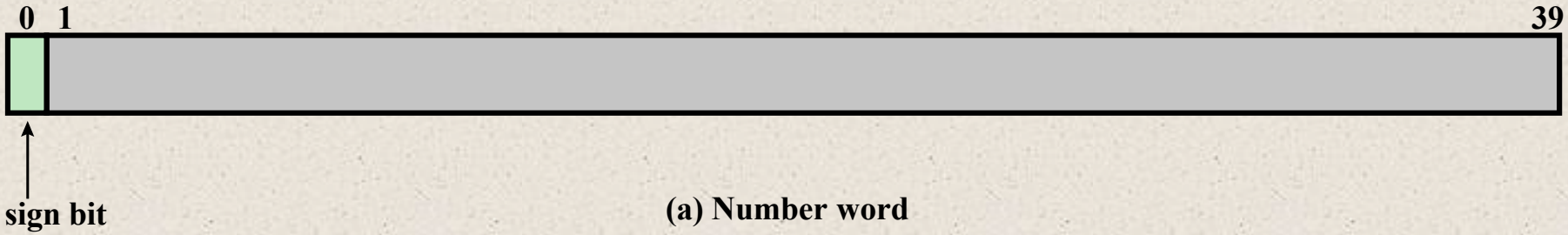
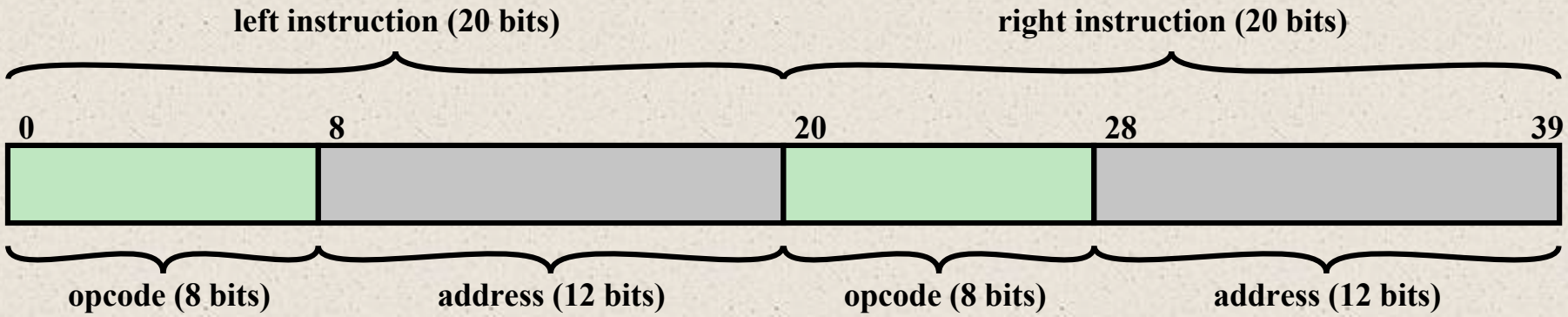


Figure 1.6 IAS Structure



(a) Number word



(b) Instruction word

Figure 1.7 IAS Memory Formats

Registers

Memory buffer register (MBR)	• Contains a word to be stored in memory or sent to the I/O unit • Or is used to receive a word from memory or from the I/O unit
Memory address register (MAR)	• Specifies the address in memory of the word to be written from or read into the MBR
Instruction register (IR)	• Contains the 8-bit opcode instruction being executed
Instruction buffer register (IBR)	• Employed to temporarily hold the right-hand instruction from a word in memory
Program counter (PC)	• Contains the address of the next instruction pair to be fetched from memory
Accumulator (AC) and multiplier quotient (MQ)	• Employed to temporarily hold operands and results of ALU operations

History of Computers

Second Generation: Transistors

- Smaller
- Cheaper
- Dissipates less heat than a vacuum tube
- Is a *solid state device* made from silicon
- Was invented at Bell Labs in 1947

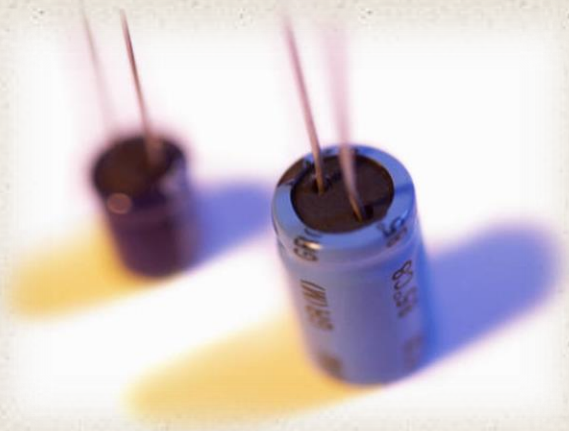
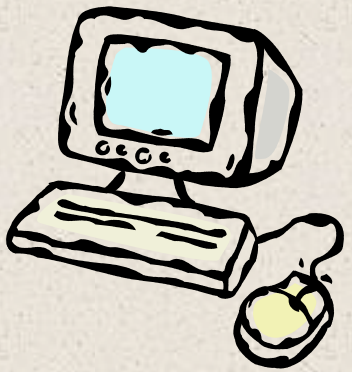


Table 1.2
Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1957–1964	Transistor	200,000
3	1965–1971	Small and medium scale integration	1,000,000
4	1972–1977	Large scale integration	10,000,000
5	1978–1991	Very large scale integration	100,000,000
6	1991-	Ultra large scale integration	>1,000,000,000

Second Generation Computers

- Introduced:
  - More complex arithmetic and logic units and control units
  - The use of high-level programming languages
  - Provision of *system software* which provided the ability to:
    - Load programs
    - Move data to peripherals
    - Libraries perform common computations



History of Computers

Third Generation: Integrated Circuits

- 1958 – the invention of the integrated circuit
- *Discrete component*
  - Single, self-contained transistor
  - Manufactured separately, packaged in their own containers, and soldered or wired together onto masonite-like circuit boards
  - Manufacturing process was expensive and cumbersome
- The two most important members of the third generation were the IBM System/360 and the DEC PDP-8

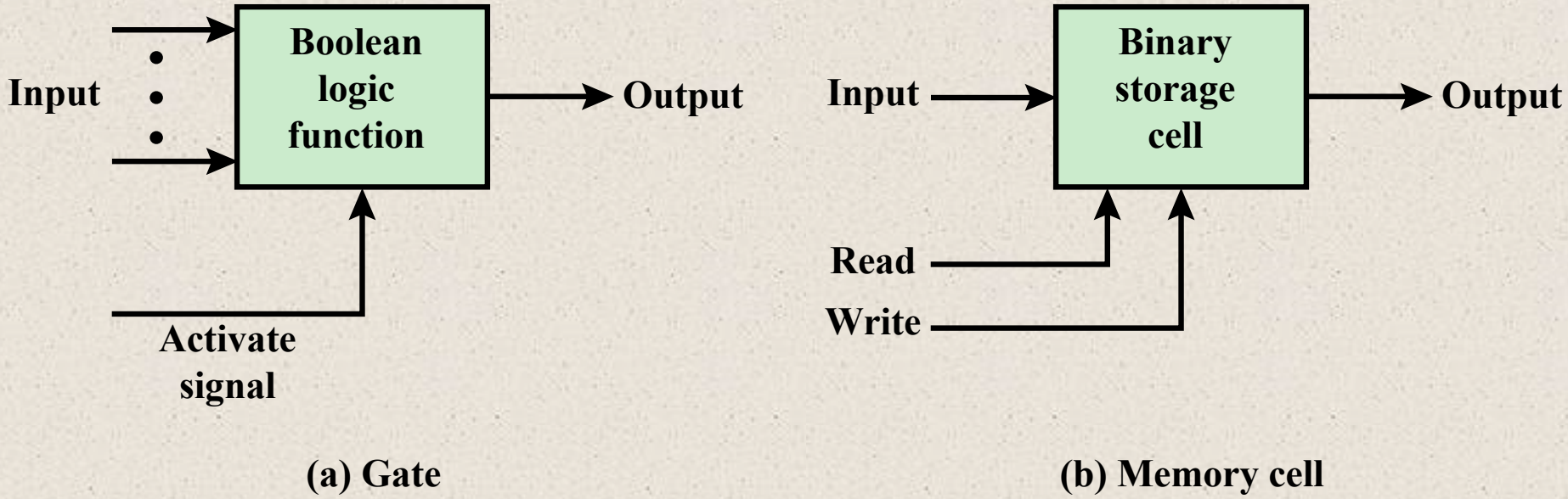


Figure 1.10 Fundamental Computer Elements

Integrated Circuits

- Data storage – provided by memory cells
- Data processing – provided by gates
- Data movement – the paths among components are used to move data from memory to memory and from memory through gates to memory
- Control – the paths among components can carry control signals

- A computer consists of gates, memory cells, and interconnections among these elements
- The gates and memory cells are constructed of simple digital electronic components
- Exploits the fact that such components as transistors, resistors, and conductors can be fabricated from a semiconductor such as silicon
- Many transistors can be produced at the same time on a single wafer of silicon
- Transistors can be connected with a processor metallization to form circuits

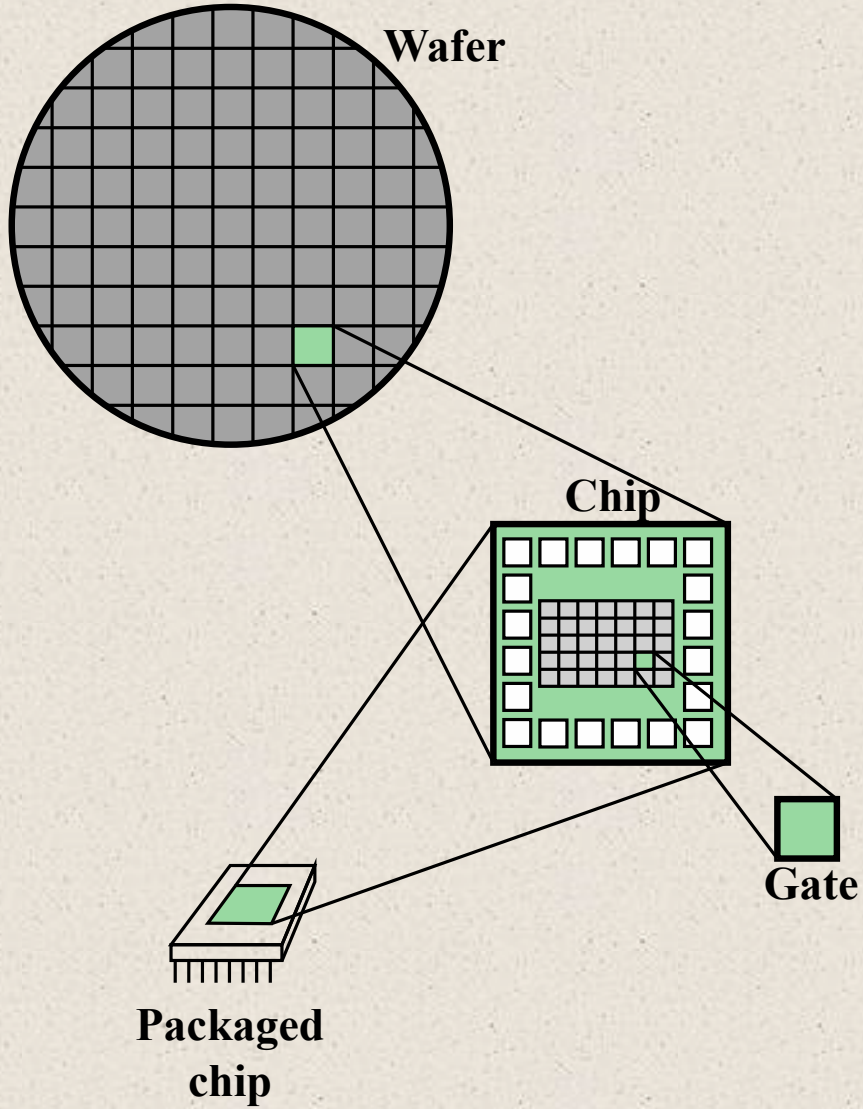


Figure 1.11 Relationship Among Wafer, Chip, and Gate

Moore's Law

1965; Gordon Moore – co-founder of Intel

Observed number of transistors that could be put on a single chip was doubling every year

The pace slowed to a doubling every 18 months in the 1970's but has sustained that rate ever since

Consequences of Moore's law:

- **The cost of computer logic and memory circuitry has fallen at a dramatic rate**
- **The electrical path length is shortened, increasing operating speed**
- **Computer becomes smaller and is more convenient to use in a variety of environments**
- **Reduction in power and cooling requirements**
- **Fewer interchip connections**

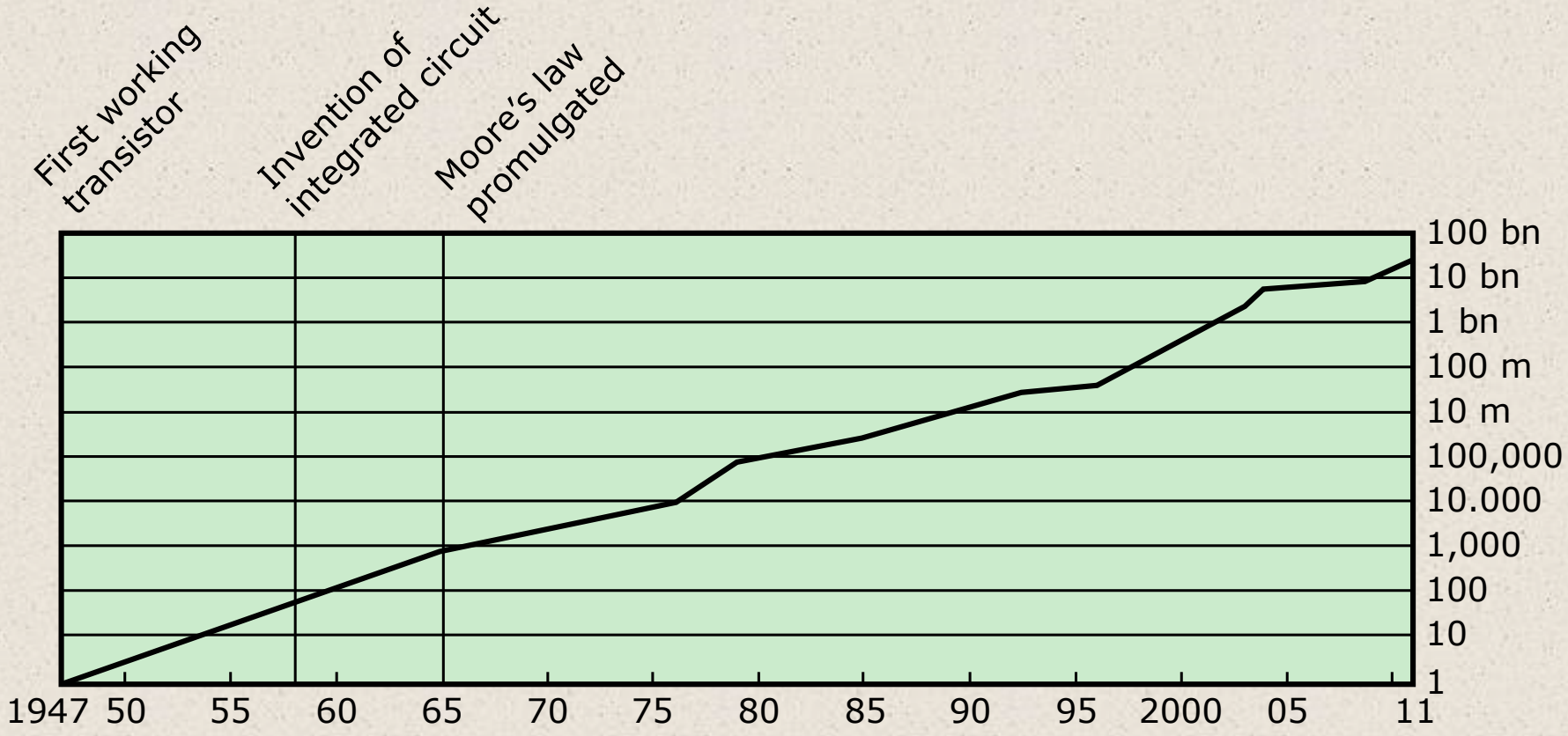
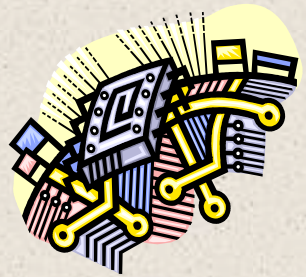


Figure 1.12 Growth in Transistor Count on Integrated Circuits (DRAM memory)

Later Generations

LSI
Large Scale Integration

VLSI
Very Large Scale Integration



Semiconductor Memory
Microprocessors

ULSI
Ultra Large Scale Integration

Evolution of Intel Microprocessors

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2,300	3,500	6,000	29,000	29,000
Feature size (μm)	10	8	6	3	6
Addressable memory	640 Bytes	16 KB	64 KB	1 MB	1 MB

(a) 1970s Processors

Evolution of Intel Microprocessors

	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz - 12.5 MHz	16 MHz - 33 MHz	16 MHz - 33 MHz	25 MHz - 50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8 - 1
Addressable memory	16 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB

(b) 1980s Processors

Evolution of Intel Microprocessors

	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz - 33 MHz	60 MHz - 166 MHz,	150 MHz - 200 MHz	200 MHz - 300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

(c) 1990s Processors

Evolution of Intel Microprocessors

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 4960X
Introduced	1999	2000	2006	2013
Clock speeds	450 - 660 MHz	1.3 - 1.8 GHz	1.06 - 1.2 GHz	4 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1.86 billion
Feature size (nm)	250	180	65	22
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1.5 MB L2/15 MB L3
Number of cores	1	1	2	6

(d) Recent Processors

Types of Computers

- A computer is a programmable electronic device that accepts raw data as input and processes it with a set of instructions (a program) to produce the desired result as output.
- We can categorize computer in two ways: on the basis of data handling capabilities and size.
- Based on the data type handling, computers can be categorized as Digital, Analog, and Hybrid.

Analogue Computer

- Analog Computers are particularly designed to process analog data.
- Analogue data is continuous data that changes continuously and cannot have discrete values. We can say that analogue computers are used where we don't need exact values always such as speed, temperature, pressure and current.
- Analogue computers directly accept the data from the measuring device without first converting it into numbers and codes. They measure the continuous changes in physical quantity and generally render output as a reading on a dial or scale.
- ***Speedometer*** and ***mercury thermometer*** are examples of analogue computers.

- **Digital**
  - Personal computers are an example of a digital computer.
  - These computers accept input in the form of 0s and 1s. The computer processes binary input and provides the output.
  - These computers perform all the logical & arithmetical operations.
  - Any input given in any language is first converted into binary language and then the computer processes the information.
  - Examples – laptops, PCs, mobile phones, desktops, etc.

- Hybrid
  - Hybrid computers are a mix of both analog and digital computers. These computers perform a high level of calculations.
  - Hybrid computers are quick and efficient. They take input in analog form, convert it into digital form, and then process it to produce an output.
  - scientists are also using hybrid computers for complex calculations.
  - For example, in hospitals to measure the heartbeat of the patients, and at research institutes to measure earthquakes and other natural calamities.
  - a processor is used in petrol pumps that converts the measurements of fuel flow into quantity and price.

- **On the basis of size**, the computer can be of *five* types:

- **Supercomputers**
  - a powerful computer that can process large amounts of data and do a great amount of computation very quickly.
  - Supercomputers are particularly used in ***scientific and engineering applications*** such as weather forecasting, scientific simulations and nuclear energy research. The first supercomputer was developed by ***Roger Cray in 1976***
  - NASA uses supercomputers for launching space satellites and monitoring and controlling them for space exploration.

- **Mainframe computer**
  - Mainframe computers are designed to ***support hundreds or thousands of users simultaneously***.
  - At their core, mainframes are high-performance computers with large amounts of memory and data processors that process billions of simple calculations and transactions in real time.
  - Examples – IBM z Series, System z9, etc.

- Mini-computers
  - Mini-computers are also known as "Midrange Computers." They are not designed for a single. They are multi-user computers designed to support multiple users simultaneously.
  - they are generally used by small businesses and firms. Individual departments of a company use these computers for specific purposes.
  - For example, the admission department of a University can use a Mini-computer for monitoring the admission process.

Microcomputer

- Microcomputers are nothing but personal computers.
- These are single-chip systems.
- These are useful for personal use and can perform all the basic functions of the computer.
- Microcomputers require very little space and are comparatively inexpensive. Such computers have the most minimalistic requirement in terms of I/O devices. And have all the circuitry mounted on a single PCB.
- For example tablets, I pads, smartwatches, laptops, desktops

Workstations

- Workstation computers are for single usage and professional purposes.
- These are like our basic laptops and desktops but with added superior features. For example, double-processor motherboard, added graphic card, ECC RAM, etc.
- The workstations are more powerful as compared to generic PCs.
- These can handle heavy-duty functions. Like animation, CAD, audio & video editing, professional gaming, etc.
  Examples: Apple PowerBook G4, SPARC CPU, MIPS CPU, etc.

Von Neumann Architecture

- Named after John von Neumann, he designed a computer architecture whereby data and instructions would be retrieved from memory, operated on by an ALU, and moved back to memory (or I/O)
- This architecture is the basis for most modern computers .
- It has three building blocks.
  - The Memory
  - I/O devices
  - The CPU

- Instructions in memory are executed sequentially unless a program instruction explicitly changes the order

- Contains a single path, between the main memory system and the control unit of the CPU

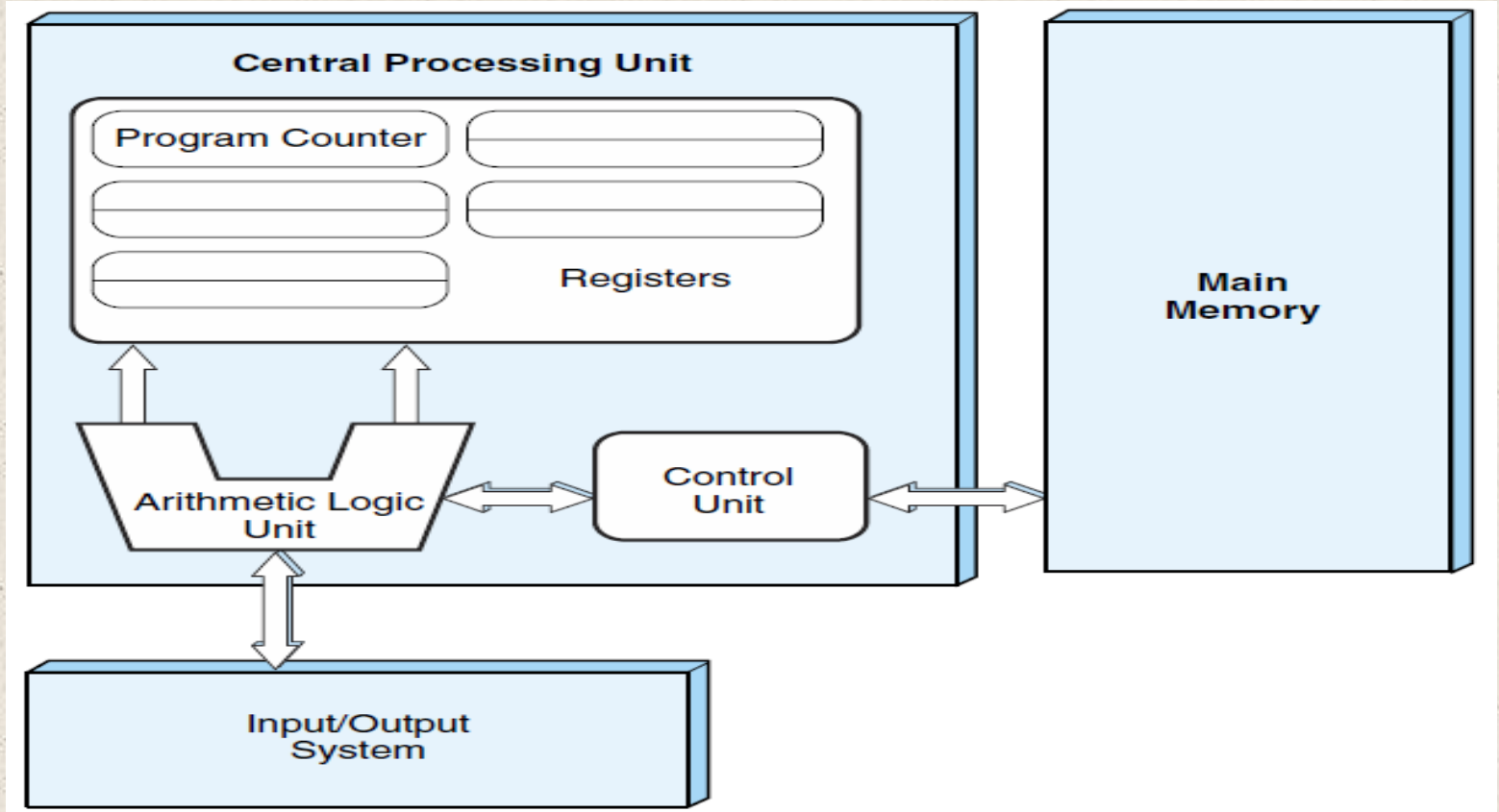


FIGURE 1.4 The von Neumann Architecture

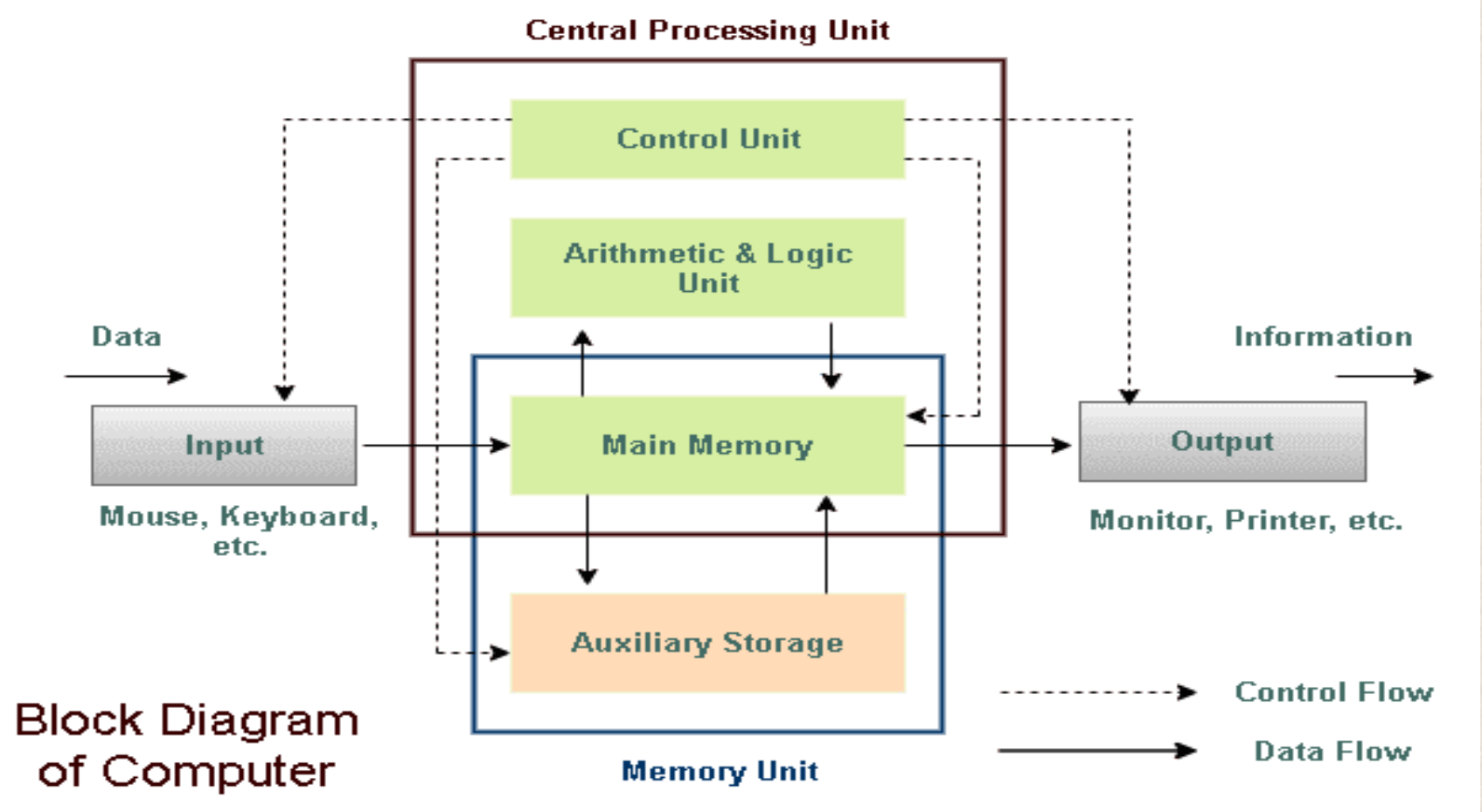
- The von Neumann architecture operates on the *fetch-execute cycle*
  - Fetch an instruction from memory as indicated by the Program Counter register
  - Decode the instruction in the control unit
  - Data operands needed for the instruction are fetched from memory
  - Execute the instruction in the ALU storing the result in a register
  - Move the result back to memory if needed

- There is a single pathway used to move both data and instructions between memory, I/O and CPU
  - the pathway is implemented as a bus
  - the single pathway creates a bottleneck known as the *von Neumann bottleneck*

- A variation of this architecture is the *Harvard architecture* which separates data and instructions into two pathways

- Another variation, used in most computers, is the system bus version in which there are different buses between CPU and memory and memory and I/O.

Functional Units of Computer



Block Diagram of Computer

Input Device

- Devices which are used to feed programs and data to the computer.
- These devices convert the input data into a digital form that is acceptable by the computer system.
- Examples: keyboard, mouse, scanner, touch screen, punch cards, light pen, joy stick, track ball, voice recognition systems.

Keyboard

Mouse

Scanner

Touch Screen

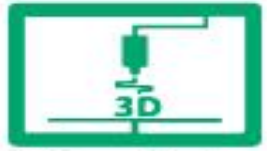
Output Devices

- The device that receives data from a computer system for display, physical production, etc. is called output device.
- It converts digital information into human understandable form.
- Example: monitor, projector, headphone, speaker, printer, etc.

Display monitor

Speaker

Printer



3D printer

Memory Unit

- Memory unit is that part of the computer system which is used to store the data and instructions to be processed.
- Types:
  i. Main memory (primary memory, internal memory)
  ii. Auxiliary storage (secondary memory)
- Characteristics of memory units:
  - **Access time**: time required to locate and retrieve a particular data from the storage unit.
  - **Storage capacity**: amount of data that can be stored by a memory unit.
  - **Cost**
  - Faster access time, higher storage capacity and low costs are desirable.

Main Memory

- It is characterized by the faster access time, less storage capacity and higher costs as compared to auxiliary storage units.
- Programs and data are loaded into the main memory before processing.
- The CPU interacts directly with the main memory to perform read or write operation.
- It is of two types: i) RAM and ii) ROM.

RAM (Random Access Memory)

- It is a volatile memory. Volatile memory stores information based on the power supply.
- If the power supply fails/ interrupted/stopped, all the data and information on this memory will be lost.
- It temporarily stores programs/data which has to be executed by the processor.
- RAM is further classified into two parts
  - SRAM
  - DRAM

Static RAM (SRAM)

- The word **static** indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to volatile nature.
- SRAM chips use transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed on a regular basis.

Dynamic RAM (DRAM)

- DRAM, unlike SRAM, must be continually **refreshed** in order to maintain the data.
- This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second.
- DRAM is used for most system memory as it is cheap and small.
- All DRAMs are made up of memory cells, which are composed of capacitors and transistors.

ROM (**Read Only Memory**)

- The memory from which we can only read but cannot write on it. This type of memory is non-volatile.
- The information is stored permanently in such memories during manufacture.
- A ROM stores such instructions that are required to start a computer. This operation is referred to as **bootstrap**.
- ROM chips are not only used in the computer but also in other electronic items like washing machine and microwave oven.
- Mainly there are three types of ROM:
  - PROM
  - EPROM
  - EEPROM

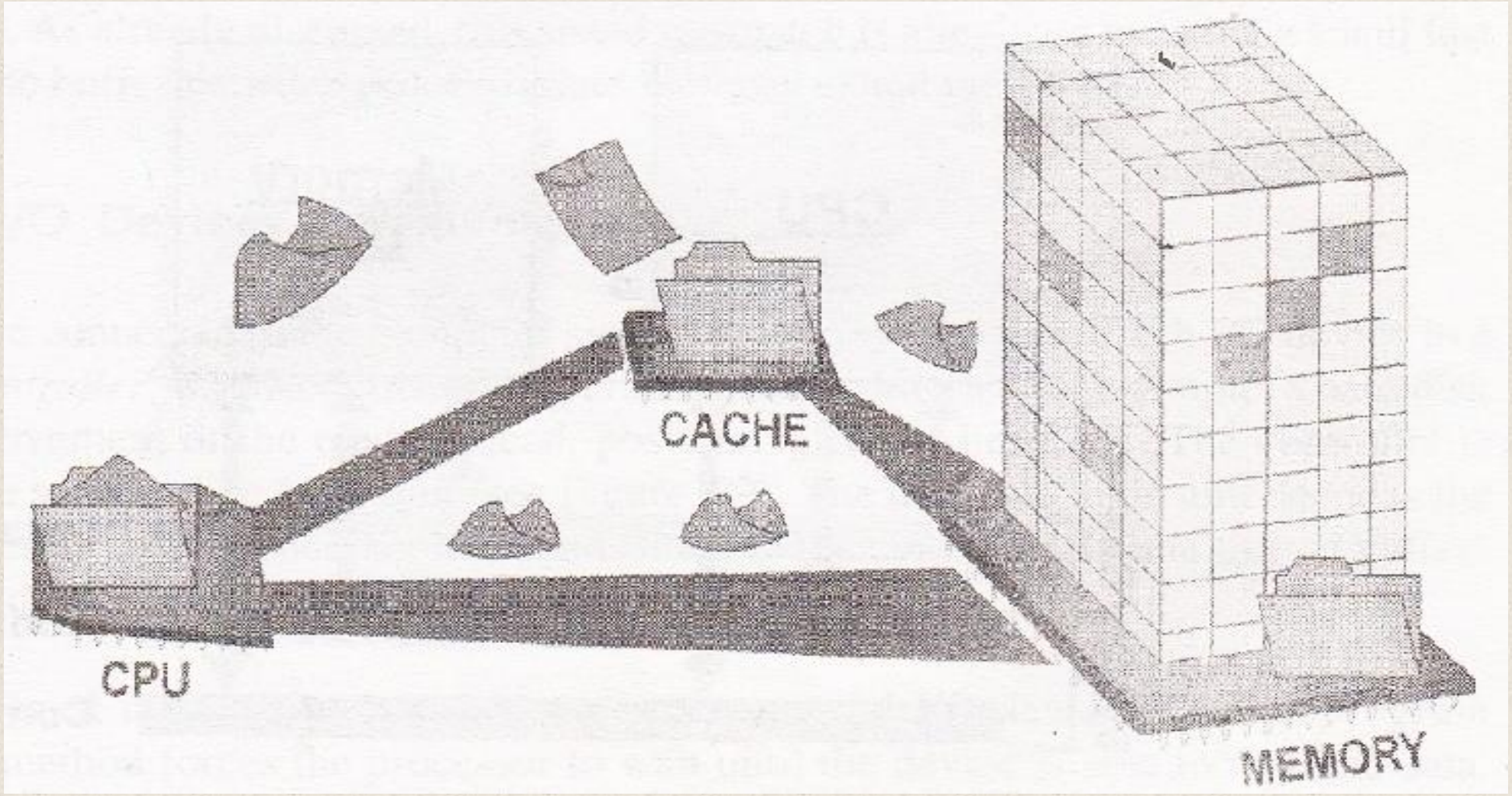
- PROM
  - **Programmable Read Only Memory**
  - This read-only memory is modifiable once by the user.
  - The user purchases a blank PROM and uses a PROM program to put the required contents into the PROM. Its content can't be erased once written.

- **EPROM**
  - Erasable Programmable Read Only Memory
  - EPROM is an extension to PROM where you can erase the content of ROM by exposing it to Ultraviolet rays for nearly 40 minutes.

- EEPROM
  - EEPROM is programmed and erased electrically.
  - It can be erased and reprogrammed about ten thousand times. Both erasing and programming take about 4 to 10 ms (millisecond).
  - In EEPROM, any location can be selectively erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip. Hence, the process of reprogramming is flexible but slow.

Cache Memory

- Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU.
- Cache Memory holds frequently requested data and instructions so that they are immediately available to the CPU when needed.
- Cache memory is costlier than main memory or disk memory but more economical than CPU registers.
- Cache Memory is used to speed up and synchronize with a high-speed CPU.


CACHE
CPU
MEMORY

Auxiliary Memory

- Main memory has limited storage capacity and is either volatile (RAM) or read-only (ROM). Thus, a computer system needs auxiliary storage to permanently store the data or instructions for future use.
- It is non-volatile and has larger storage capacity than the main memory.
- Slower and cheaper than the main memory.
- It can not be accessed directly by the CPU.
- Contents of auxiliary storage need to be first brought into the main memory for the CPU to access.
- Example: Hard Disk Drive (HDD), CD/DVD, Memory Card, etc.

Central Processing Unit (CPU)

- It is the electronic circuitry of a computer that carries out the actual processing and usually referred as the brain of the computer.
- The CPU is given instructions and data through programs.
- The CPU performs arithmetic and logic operations as per the given instructions.

- The CPU comprises of three parts:
  - i) the control unit,
  - ii) the arithmetic & logic unit, and
  - iii) the main memory.
- **Control Unit (CU)**:
  - It controls the operations of the entire computer system.
  - The control unit gets the instructions from the programs stored in the main memory, interpret these instructions and subsequently directs the other units to execute the instructions.
  - The control unit is a component of a computer's central processing unit that coordinates the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices how to respond to a program's instructions.

Arithmetic & Logic Unit (ALU):

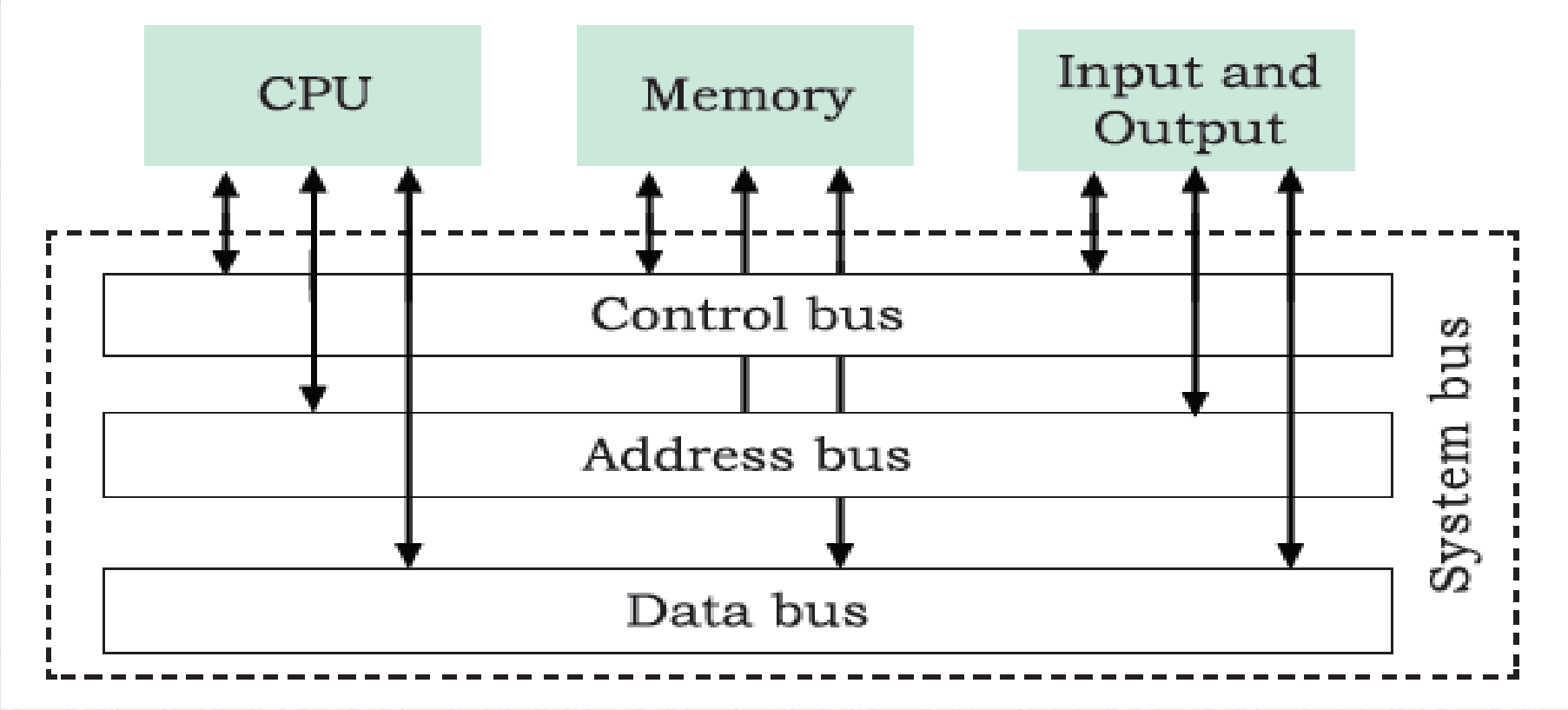
- Most of all the arithmetic and logical operations of a computer are executed in the ALU (Arithmetic and Logical Unit) of the processor.
- It performs arithmetic operations like addition, subtraction, multiplication, division and also the logical operations like AND, OR, NOT operations.

Data Transfer

- Data Transfer between Memory and CPU:
  - Data are transferred between different components of a computer system using physical wires called bus.
- Bus is of three types:
  - Data bus – to transfer data between different components.
  - Address bus – to transfer addresses between CPU and main memory.
  - Control bus – to communicate control signals between different components of a computer.

..contd

- All these three buses collectively make the system bus.



…contd

- The CPU places on the *address bus*, the address of main memory location from which it wants to read data or to write data.
- While executing the instructions, the CPU specifies the read/write control signal through the *control bus*.
- The CPU may require to read data from main memory or write data to main memory, a data bus is bidirectional.
- Control bus and address bus – unidirectional.
- In case of read operation, the CPU specifies the address, and the data is placed on the data bus by a dedicated hardware, called memory controller.

LOGIC GATES

- A logic gate is a device that acts as a building block for digital circuits . They perform basic logical functions that are fundamental to digital circuits. In a circuit, logic gates will make decisions based on a combination of digital signals coming from its inputs .

TYPES OF LOGIC GATES

There are Three types of basic Logic Gates

- **OR Gate**
- **AND Gate**
- **NOT Gate**

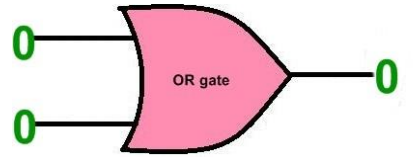
OR GATE

The OR gate is a digital logic that implements logical disjunction. An OR gate produces a high output when any one of the input is high .It produces a low output when all the inputs are low. (X=A+B)

2 Input OR gate Truth Table

INPUTS		OUTPUTS
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Symbol :



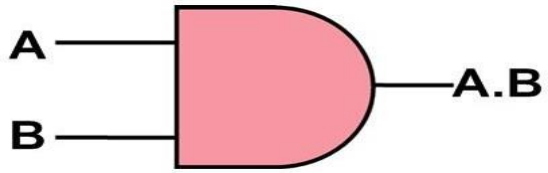
AND GATE

- It will produce a high output when all the inputs are high otherwise the output is low . (X=A.B)

Truth Table of 2 input AND gate

Inputs		Outputs
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Symbol :



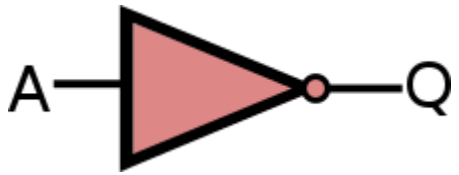
NOT GATE

It produces high output when the input is low and vice versa. The NOT gate is also called as an inverter. (Q=A’)

Truth Table

Input	Output
A	Y
0	1
1	0

Symbol :



Universal Gates

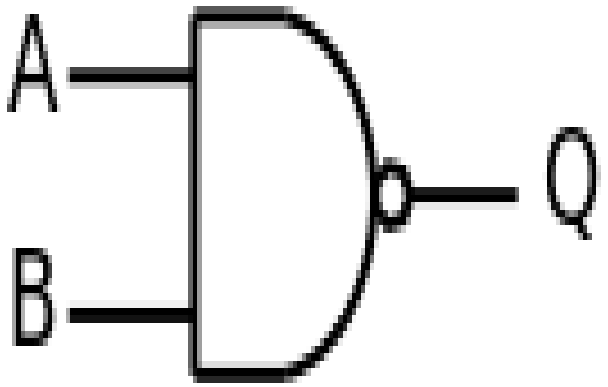
- A universal gate is a gate which can implement any Boolean function without need to use any other gate type.

Types of Universal Gate

- **NAND Gate**
- **NOR Gate**

NAND GATE

NAND gate is AND gate followed by NOT gate. (Q=(A.B)’)



NAND Gate

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

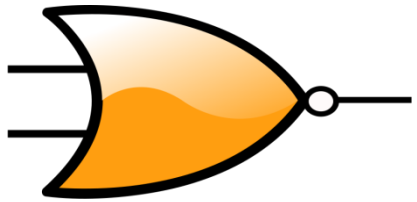
NOR GATE

NOR gate is OR gate followed by NOT gate. (X=(A+B)')

2 input NOR gate truth table

INPUTS		OUTPUTS
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Symbol :



Exclusive-OR Gate

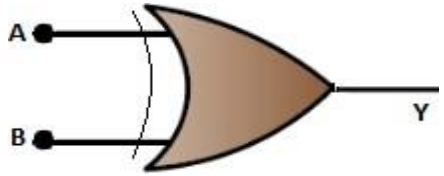
When both inputs are same, it gives low output.

Output Equation $Y = (A \oplus B) = A'.B + A.B'$

Truth Table

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Symbol -



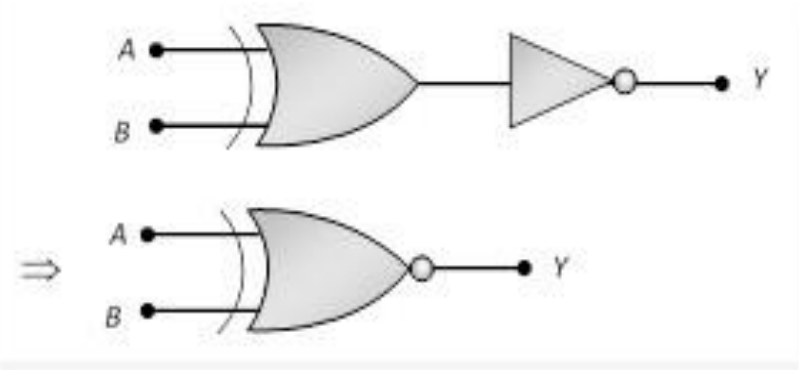
Exclusive-NOR Gate

The Ex NOR gate gives high output when all the inputs are at same logic level.

Truth Table

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Symbol -



Boolean Algebra

- The Boolean algebra is mainly used for simplifying and analyzing the complex Boolean expression. It is also known as **Binary algebra** because we only use binary numbers in this. **George Boole** developed the binary algebra in **1854**.
- Boolean Algebra is used to analyze and simplify the digital (logic) circuits.
- It deals with the binary variables and logic operations.
- The variables are designed by letters such as A, B, x, y.
- The basic logical operations are AND, OR and complement.

Properties of Boolean Algebra

- **Annulment Law**

When the variable is AND with 0, it will give the result 0, and when the variable is OR with 1, it will give the result 1, i.e.,

$x.0 = 0$

$x+1 = 1$

- **Identity Law**

When the variable is AND with 1 and OR with 0, the variable remains the same, i.e.,

$x.1 = x$

$x+0 = x$

- **Idempotent Law**

When the variable is AND and OR with itself, the variable remains same or unchanged, i.e.,

$x.x = x$

$x+x = x$

- **Complement Law**

When the variable is AND and OR with its complement, it will give the result 0 and 1 respectively.

$$x.x' = 0$$

$$x+x' = 1$$

- **Double Negation Law**

This law states that, when the variable comes with two negations, the symbol gets removed and the original variable is obtained.

$$((x)')' = x$$

- **Commutative Law**

This law states that no matter in which order we use the variables. It means that the order of variables doesn't matter in this law.

$$x.y = y.x$$

$$x+y = y+x$$

❑ **Associative Law**

This law states that the operation can be performed in any order when the variables priority is of same as '*' and '/'.

$$(x.y).z = x.(y.z)$$

$$(x+y)+z = x+(y+z)$$

❑ **Distributive Law**

This law allows us to open up of brackets. Simply, we can open the brackets in the Boolean expressions.

$$x+(y.z) = (x+y).(x+z)$$

$$x.(y+z) = (x.y)+(x.z)$$

De Morgan Law

- The complement of the product (AND) of two Boolean variables (or expressions) is equal to the sum(OR) of the complement of each Boolean variable (or expression).

$$(x.y)' = (x)' + (y)'$$

- The Complement of the sum (OR) of two Boolean variables (or expressions) is equal to the product(AND) of the complement of each Boolean variable (or expression).

$$(x + y)' = (x)'.(y)'$$

Boolean Algebra Simplification Table

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

Boolean Functions and Expressions

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

$F(A, B, C, D)$	=	$A + \overline{BC} + ADC$	Equation No. 1
Boolean Function		Boolean Expression	

Here the left side of the equation represents the output Y. So we can state equation no. 1

$$Y = A + \overline{BC} + ADC$$

Example: F(A,B,C,D)=A+BC'+D

The output will be high when A=1 or BC'=1 or D=1 or all are set to 1. The truth table of the above example is given below. The 2^n is the number of rows in the truth table. The n defines the number of input variables. So the possible input combinations are $2^3=8$.

Inputs				Output
A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

The Boolean Expression: A.(A + B)

Multiplying out the brackets gives us:

	A.(A+B)	Start
multiply:	A.A + A.B	Distributive Law
but:	A.A = A	Idempotent Law
then:	A + A.B	Reduction
thus:	A.(1 + B)	Annulment Law
equals to:	A	Absorption Law

Boolean Expression: (A + B)(A + C)

Again, multiplying out the brackets gives us:

	(A + B)(A + C)	Start
multiply:	A.A + A.C + A.B + B.C	Distributive law
but:	A.A = A	Idempotent Law
then:	A + A.C + A.B + B.C	Reduction
however:	A + A.C = A	Absorption Law
thus:	A + A.B + B.C	Distributive Law
again:	A + A.B = A	Absorption Law
thus:	A + B.C	Result

Boolean Expression: AB(BC + AC)

	$AB(\overline{B}C+AC)$	Start
multiply	$A.B.\overline{B}.C + A.B.A.C$	Distributive Law
again:	$A.A = A$	Idempotent Law
then:	$A.B.\overline{B}.C + A.B.C$	Reduction
but:	$B.\overline{B} = 0$	Complement Law
so:	$A.0.C + A.B.C$	Reduction
becomes:	$0 + A.B.C$	Reduction
as:	$0 + A.B.C = A.B.C$	Identity Law
thus:	ABC	Result

Then the Boolean expression of $AB(\overline{B}C+AC)$ is reduced to "ABC".

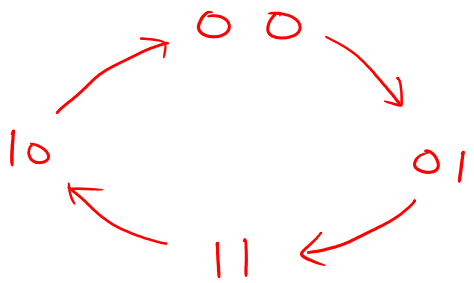
$$(A + B + C)(A + B + C)(A + B + C)$$

STANDARD FORMS OF BOOLEAN EXPRESSIONS

- All Boolean expressions, regardless of their form, can be converted into either of two standard forms: the sum-of-products form or the product-of- sums form.
- Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

K-MAP

- It is used when output is 0, 1, or x (don't care)
- When number of variables are 2, 3, 4, 5 (upto 5 variables)
- In K-MAP gray code representation is used
- K-MAP is graphical representation

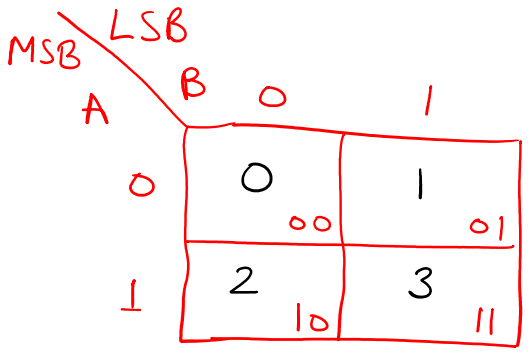


Gray code representation

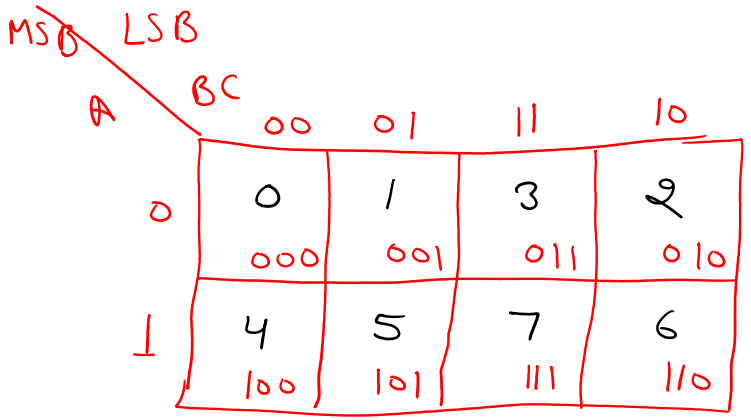
⇒ each successive term is changed by only one bit.

K-MAP

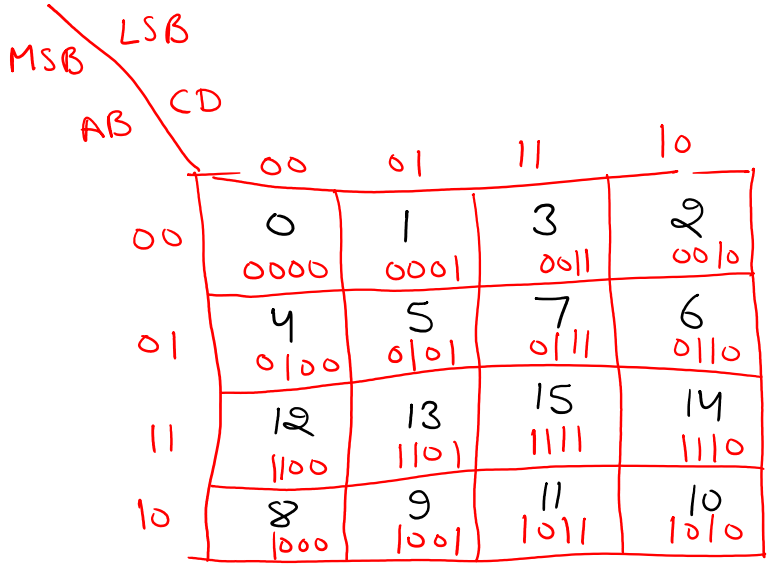
- Two variable:



Three variable:

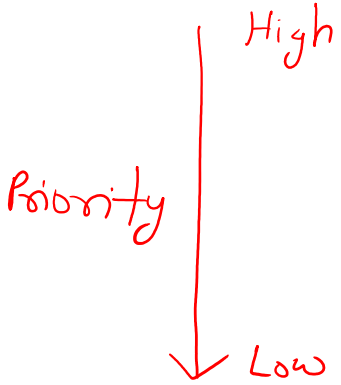


Four variable:



K-MAP

- Procedure:
  i. Octets
  ii. Quads
  iii. Pairs
  iv. Single term
  v. Remove redundant



K-MAP

With two variables:

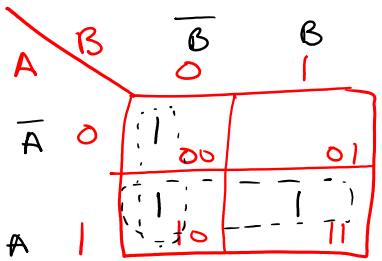
A	B	Decimal Equivalent
0	0	0
0	1	1
1	0	2
1	1	3

- Simplify:

Question: $f(A, B) = \Sigma m(0, 2, 3)$

Solution: ↓ largest (0,2,3) ⇒ 3 ↓ minimum number of variables required to represent it = 2

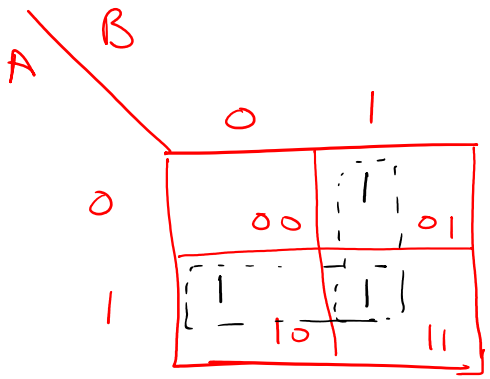
use two-variable k-map:



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

K-MAP

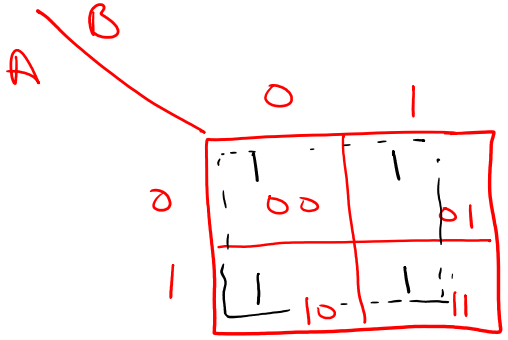
- Simplify:
- f(A, B) = Σm(1, 2, 3)



$f(A,B) = A + B$

K-MAP

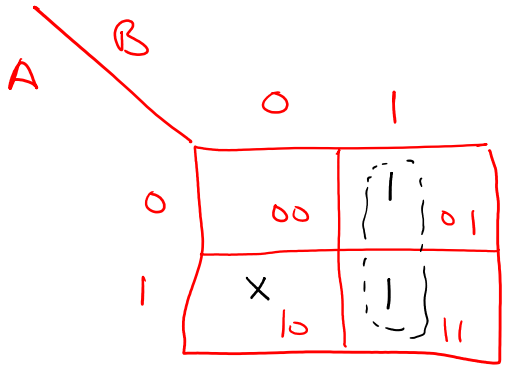
- Simplify:
- f(A, B) = Σm(0, 1, 2, 3)



$f(A, B) = 1$

K-MAP

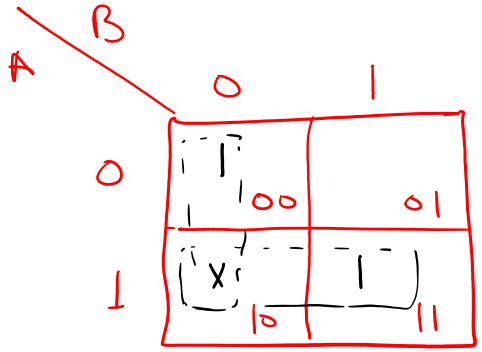
- Simplify:
- f(A, B) = $\sum$m(1, 3) + $\sum$d(2)



$f(A,B) = B$

K-MAP

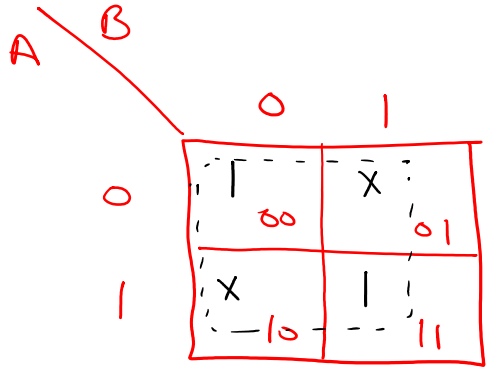
- Simplify:
- f(A, B) = Σm(0, 3) + Σd(2)



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

K-MAP

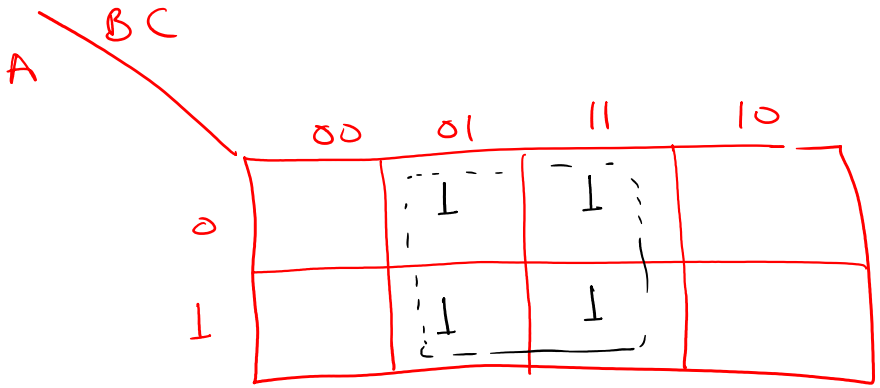
- Simplify:
- f(A, B) = Σm(0, 3) + Σd(1, 2)



$f(A,B) = 1$

K-MAP

- Simplify:
- f(A, B, C) = Σm(1, 3, 5, 7)

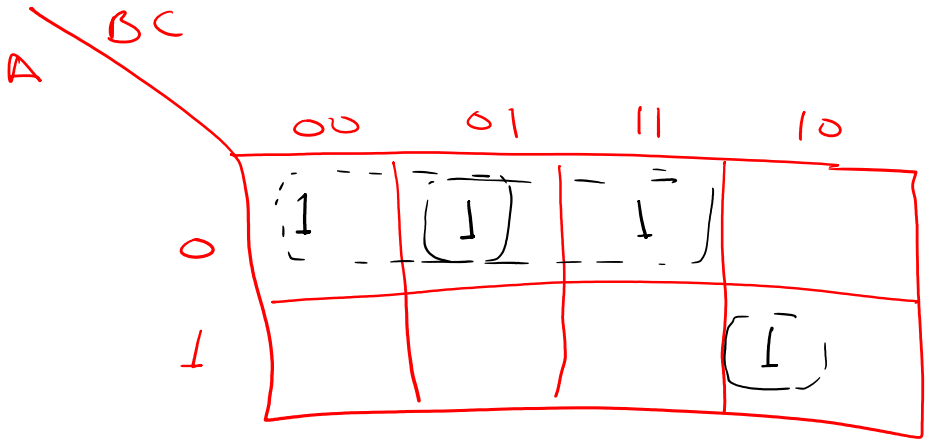


$f(A, B, C) = C$

⇒

K-MAP

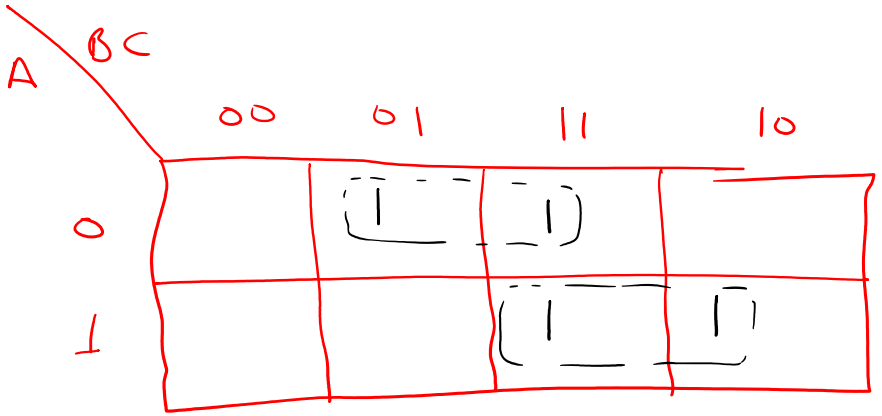
- Simplify:
- f(A, B, C) = Σ m(0, 1, 3, 6)



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

K-MAP

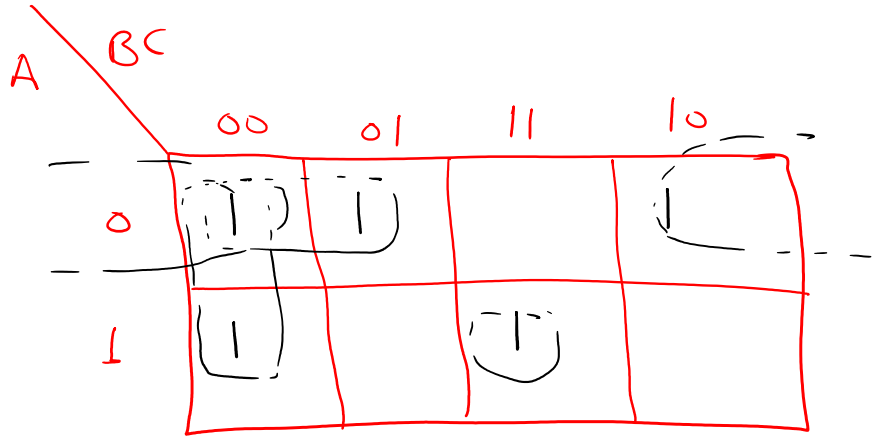
- Simplify:
- f(A, B, C) = Σm(1, 3, 6, 7)



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

K-MAP

- Simplify:
- f(A, B, C) = Σ m(0, 1, 2, 4, 7)

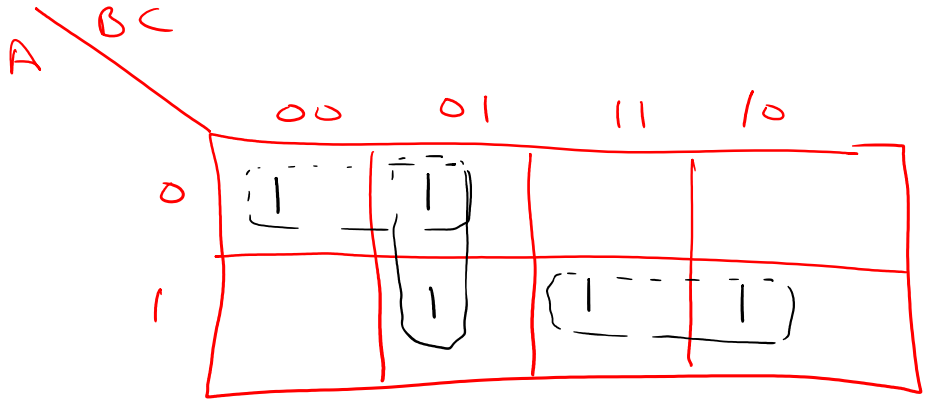


$$f(A,B,C) = \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C} + ABC$$

K-MAP

- Simplify:
- $f(A, B, C) = \Sigma m(0, 1, 5, 6, 7)$

NOTE: K-MAP provides minimize expression but not necessarily unique.



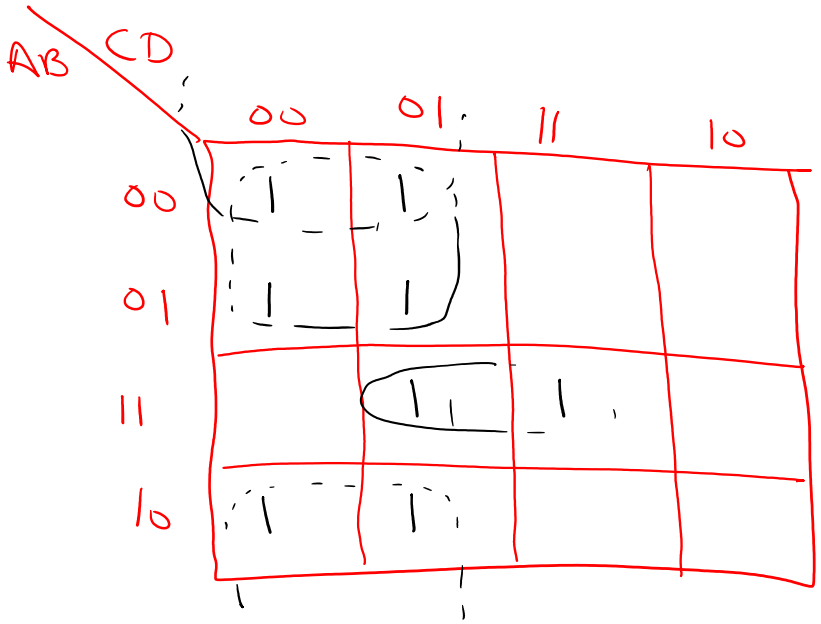
$f(A,B,C) = \overline{A}\,\overline{B} + \overline{B}C + AB$

OR $= \overline{A}\,\overline{B} + AC + AB$

} two solutions.

K-MAP

- Simplify:
- f(A, B, C, D) = Σm(0, 1, 4, 5, 8, 9, 13, 15)



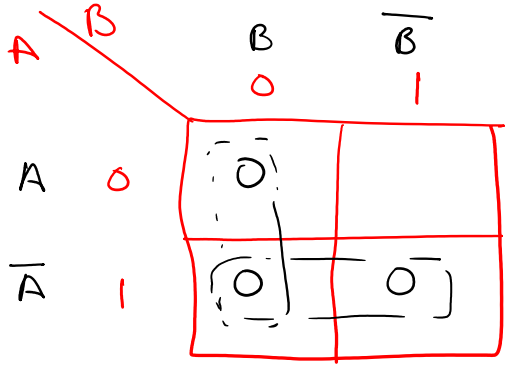
$$f(A,B,C,D) = \bar{B}\bar{C} + \bar{A}\bar{C} + ABD$$

K-MAP

- Simplify:
- f(A, B, C, D) = Σm(0, 2, 8, 10, 14) + Σd(5, 15)

K-MAP

- Simplify:
- f(A, B) = ΠM(0, 2, 3)

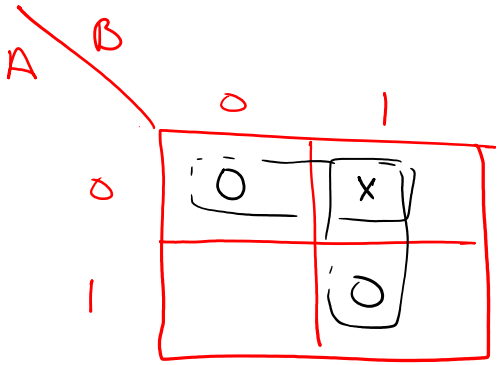


product of sum

$f(A,B) = \overline{A} \cdot B$ ← POS form

K-MAP

- Simplify:
- f(A, B) = ∏M(0, 3) + ∏d(1)

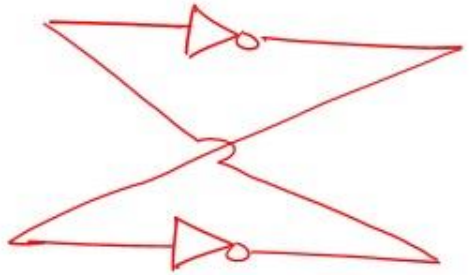


$$f(A,B) = A\overline{B}$$

Flip Flops

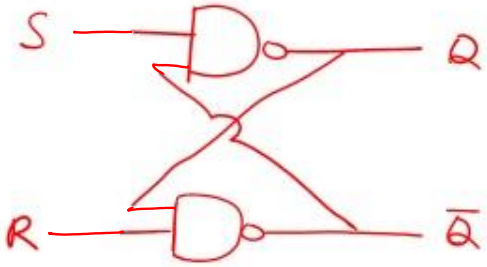
- Basic memory element.
- It can store 1 bit.
- FF has two outputs, complemented to each other.
- It has two stable state hence it is known as Bistable multivibrator.
- Content:
  - SR latch — NAND / NOR
  - SR FF
  - JK FF
  - D FF
  - T FF

Latch



(latch using NOT gate)

SR latch using NAND:

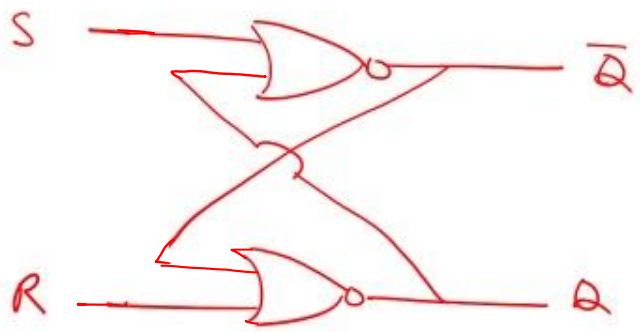


Truth table:

S	R	Q
0	0	Invalid ($\because Q = \bar{Q} = 1$)
0	1	1
1	0	0
1	1	Previous state (no change)

Latch

SR latch using NOR gate:



Truth table:

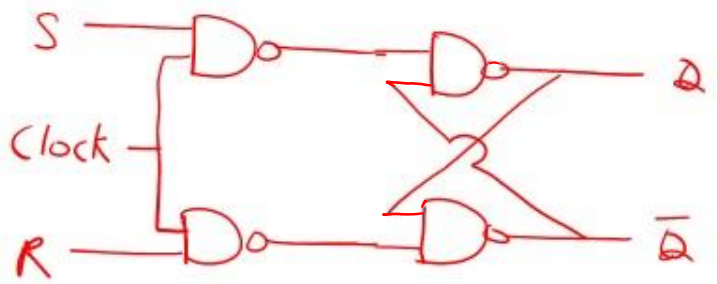
S	R	Q
0	0	Previous state
0	1	0
1	0	1
1	1	Invalid ($Q = \bar{Q} = 0$)

Flip Flop

SR Flip flop: S : Set

R : Reset

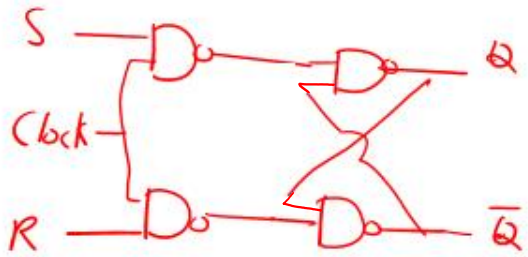
Circuit:



Truth table:

Clock	S	R	Q_{n+1}	
0	*	*	Previous state (Q_n)	} hold state
1	0	0	Q_n	
1	0	1	0	→ Reset
1	1	0	1	→ Set
1	1	1	Invalid	→ unused

Flip Flop



Disadvantage:

* Invalid state present when S=1 and R=1

To avoid this:

JK FF is used.

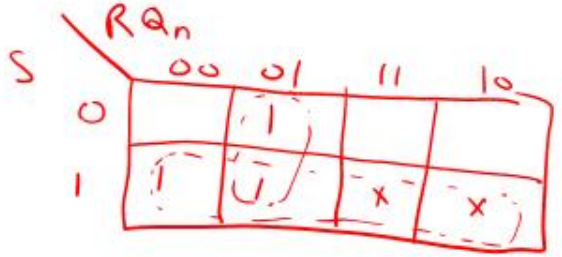
Characteristic table:

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

Excitation table:

Q_n	Q_{n+1}	S	R
0	0	0	*
0	1	1	0
1	0	0	1
1	1	*	0

Characteristic equation:

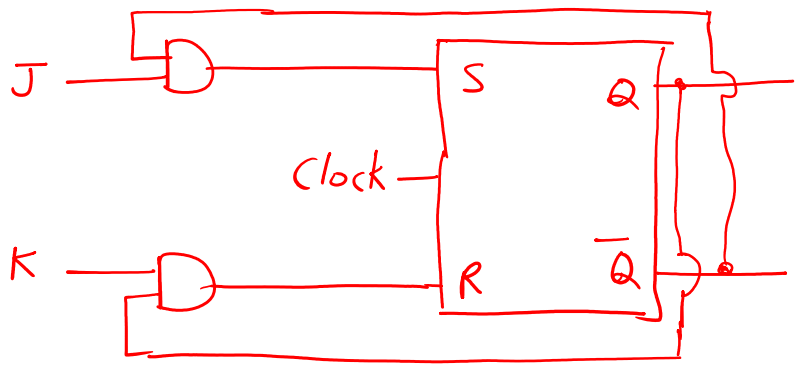


$$Q_{n+1} = S + \bar{R}Q_n$$

and $S \cdot R = 0$

Flip Flop

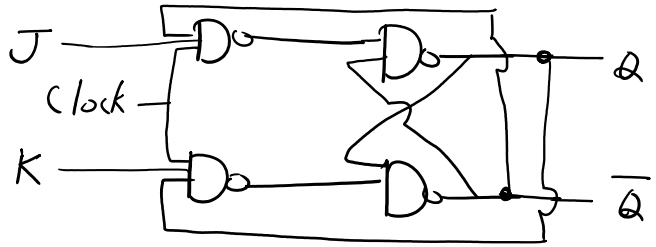
JK Flip Flop :



$S = J\bar{Q}$

$R = KQ$

JK FF using NAND gate :

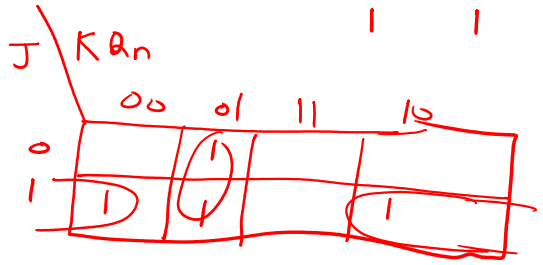


Truth table →

Clock	J	K	Q_{n+1}
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\bar{Q}_n$

Characteristic table →

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

Flip Flop

JK flip flop:

Excitation table:

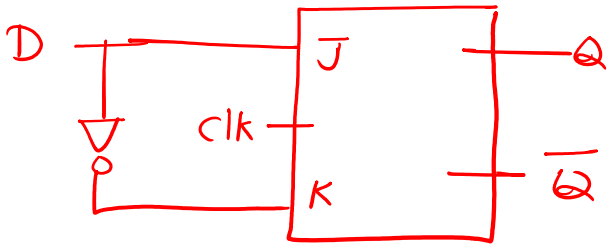
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

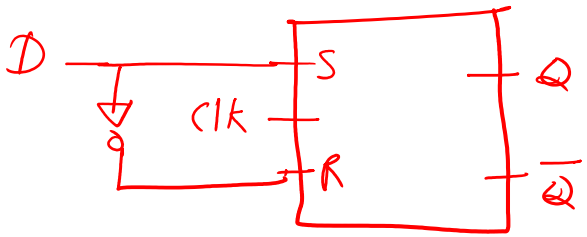
Flip Flop

D Flip Flop:

$J = D$

$K = \overline{D}$

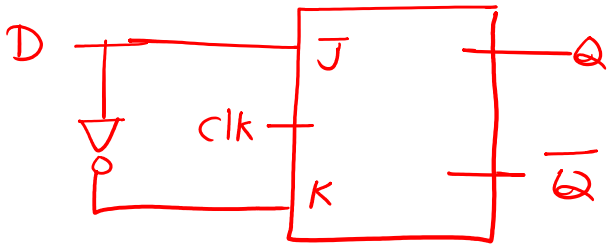


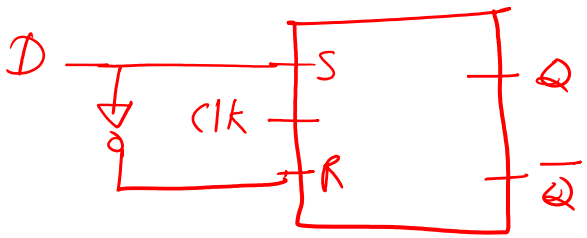


Truth table:

Flip Flop

D Flip Flop:





Truth table:

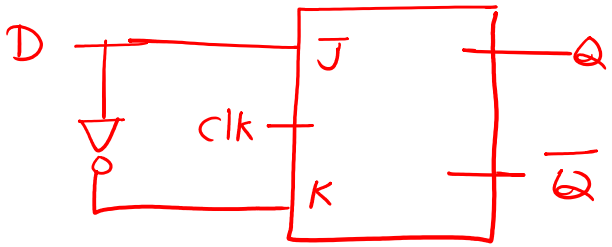
Clk	D	Q_{n+1}
0	*	Q_n
1	0	0
1	1	1

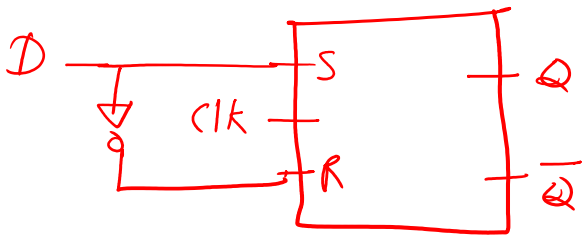
Characteristic table:

?

Flip Flop

D Flip Flop:





Truth table:

clk	D	Q_{n+1}
0	*	Q_n
1	0	0
1	1	1

Characteristic table:

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

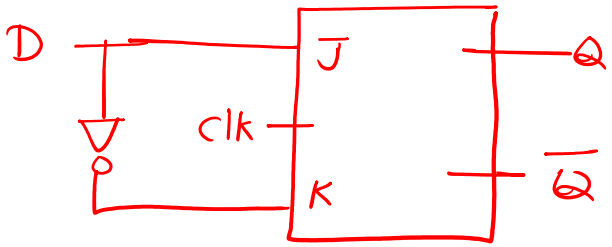
Excitation table:

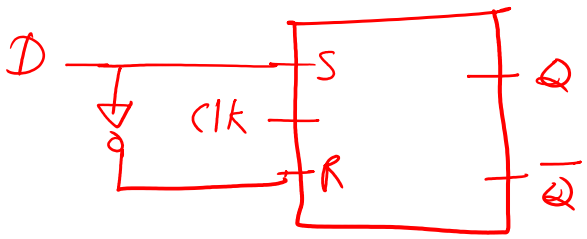
$Q_{n+1} = D$

∴ It is also called as transperent latch

Flip Flop

D Flip Flop:





Truth table:

clk	D	Q_{n+1}
0	*	Q_n
1	0	0
1	1	1

Characteristic table:

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

$Q_{n+1} = D$

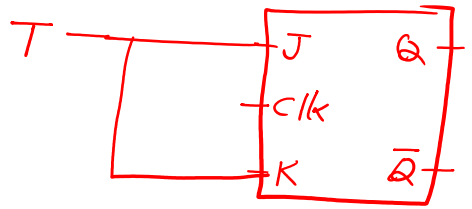
Excitation table:

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

∴ It is also called as transperent latch

Flip Flop

T - Flip Flop :



$J = K = T$

Truth table :

Clk	T	Q_{n+1}
0	x	Q_n
1	0	Q_n
1	1	$\bar{Q}_n$

Characteristic table :

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table :

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

$$Q_{n+1} = \bar{T} Q_n + T \bar{Q}_n = T \oplus Q_n$$

Flip Flop

Important:

J	K	Q_{n+1}	
0	0	Q_n	SR
0	1	0	SR, D-FF
1	0	1	SR, D-FF
1	1	$\overline{Q_n}$	T-FF

Excitation table:

Q_n	Q_{n+1}	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	0	1
1	1	x	0	x	0	1	0

Encoder

- Encoder is the combinational circuit which has many inputs and many outputs.
- Encoder is used to convert other code to binary.

↳ Octal to binary

or Hexadecimal to binary

Encoder

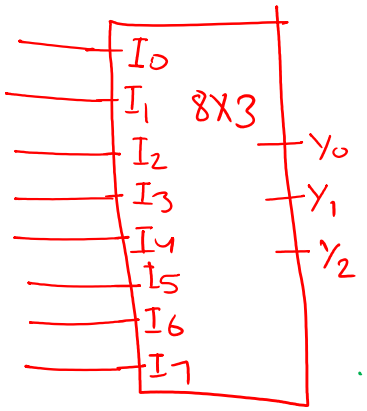
- Example: Octal to binary encoder:

Truth table:

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Encoder

Example: Octal to binary encoder:



Truth table:

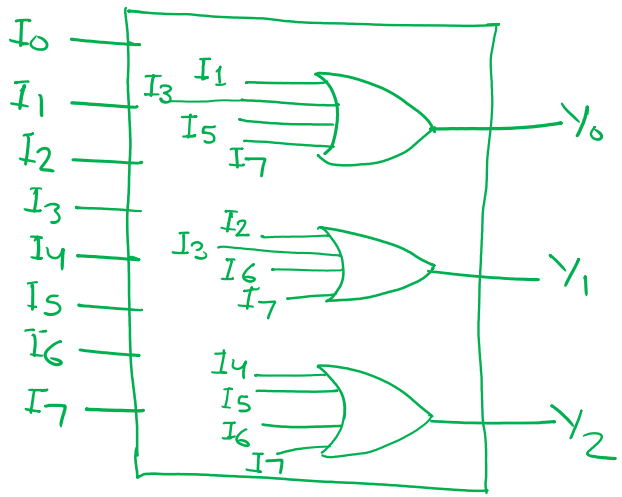
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Logical expression

$Y_0 = I_1 + I_3 + I_5 + I_7$

$Y_1 = I_2 + I_3 + I_6 + I_7$

$Y_2 = I_4 + I_5 + I_6 + I_7$



Decoder

- Decoder is a combinational circuit which has many inputs and many outputs.
- It is used to convert binary data to other code.

Binary to Octal (3 x 8)

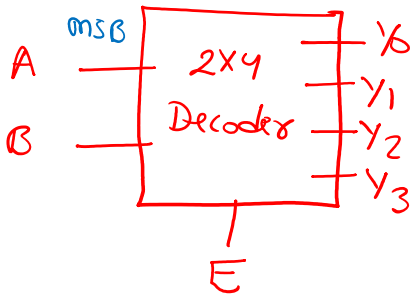
Binary to Decimal (4 x 16)

Binary to Hexadecimal (4 x 16)

* 2 to 4 decoder is minimum possible decoder.

Decoder

2x4 decoder:



Truth table:

E	A	B	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

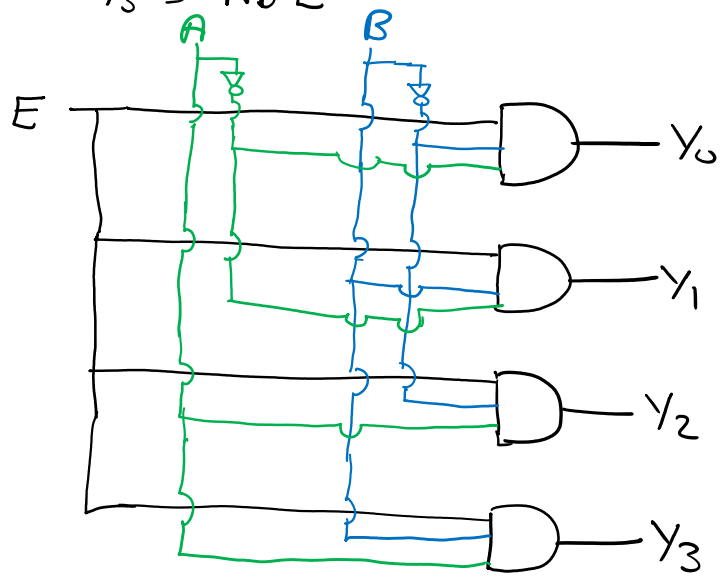
Logical expressions:

$Y_0 = \bar{A}\bar{B}E$

$Y_1 = \bar{A}BE$

$Y_2 = A\bar{B}E$

$Y_3 = ABE$

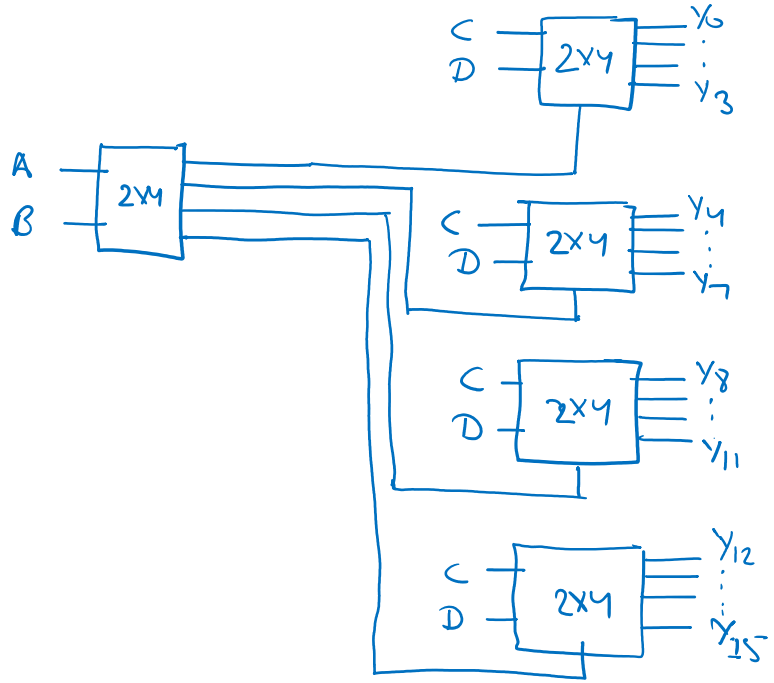


Decoder

* Implementation of higher-order decoder using lower order decoder.

Q: Implement 4X16 decoder using 2X4 decoder.

Soln:



Try this with this circuit.

A	B	C	D
1	0	1	0

result

$Y_{10} = 1$

Decoder

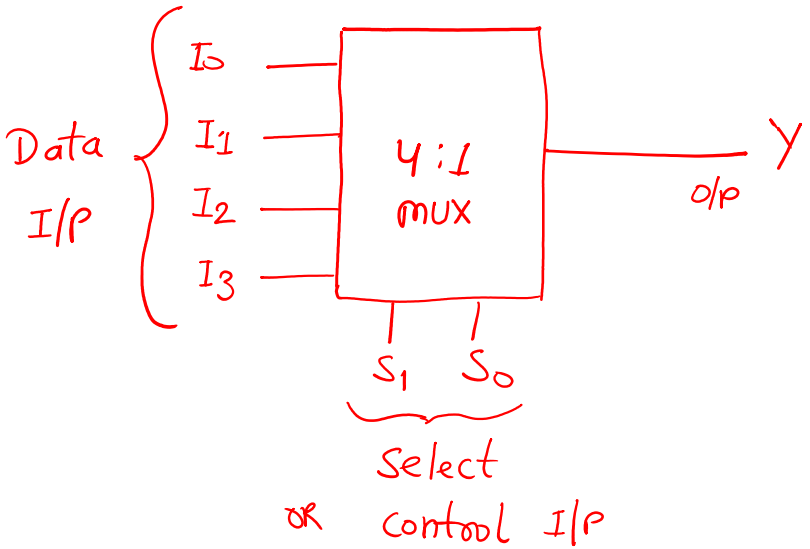
1. How many 2x4 decoders will be used in designing a 6x64 decoder?

2. ∴ ∴ 4x16 ∴ ∴ ∴ ∴ ∴ ∴ ∴ 8x256 ∴ ?

Multiplexer (MUX)

- Many inputs and one output.
- It is a combinational circuit.
- Depending on control or select input, one of the input is transferred to the output line.
- Also known as data selector or, many to one circuit or, universal logic circuit or, parallel to serial circuit.

I/P : input
O/P : output



$$m = 2^n$$

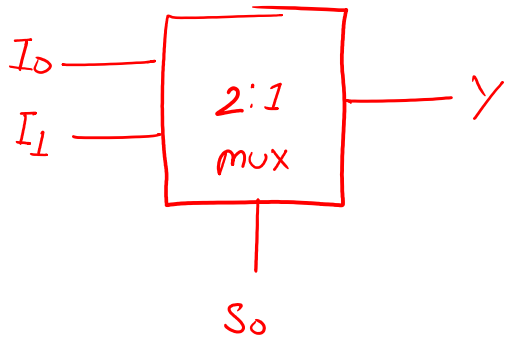
or $$n = \log_2 m$$

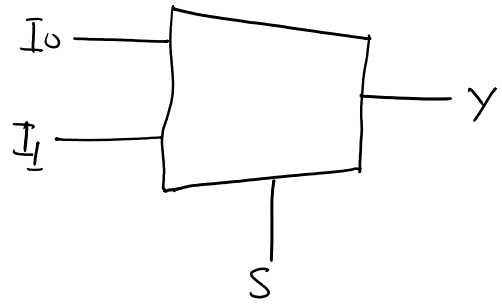
where m = number of data I/P

n = number of select I/P (or control I/P)

MUX

2 : 1 MUX :





Symbol of mux

Truth table :

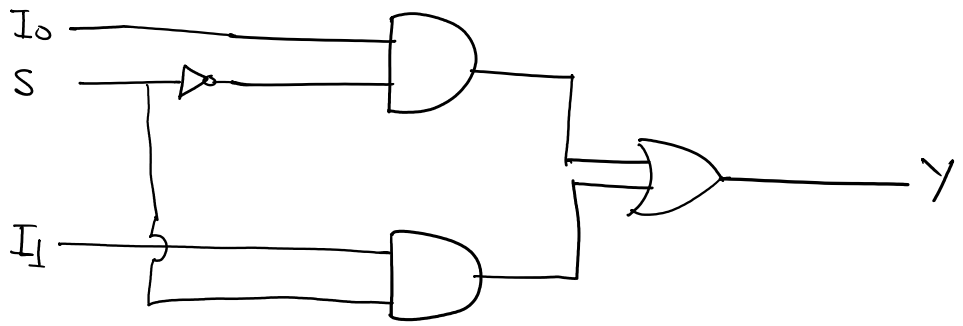
S	Y
0	I_0
1	I_1

Logical Expression :

$$Y = \bar{S} I_0 + S I_1$$

MUX

Implementation :

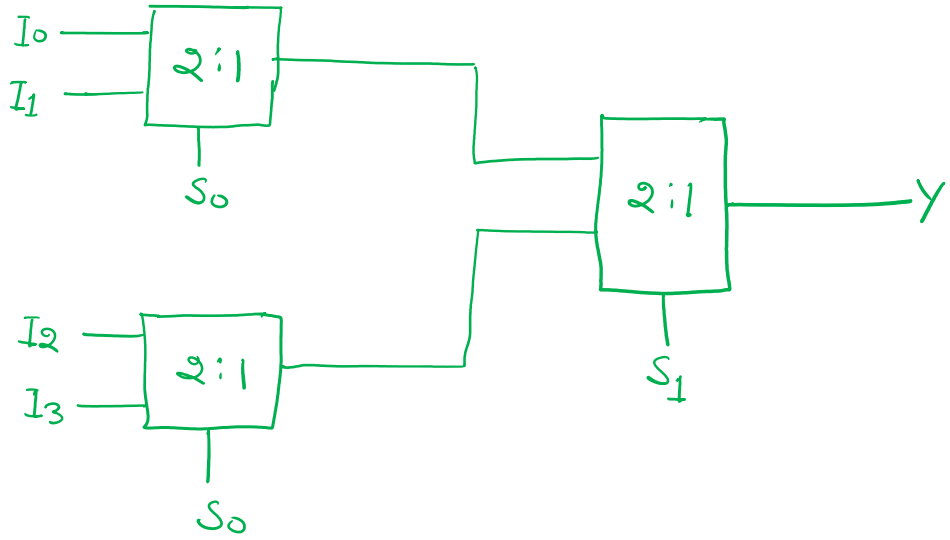


In MUX, generally AND gate followed by OR gate.

MUX

* Implementation of higher order mux with lower order mux :

(A) Implement 4:1 mux using 2:1 mux



4:1 mux :

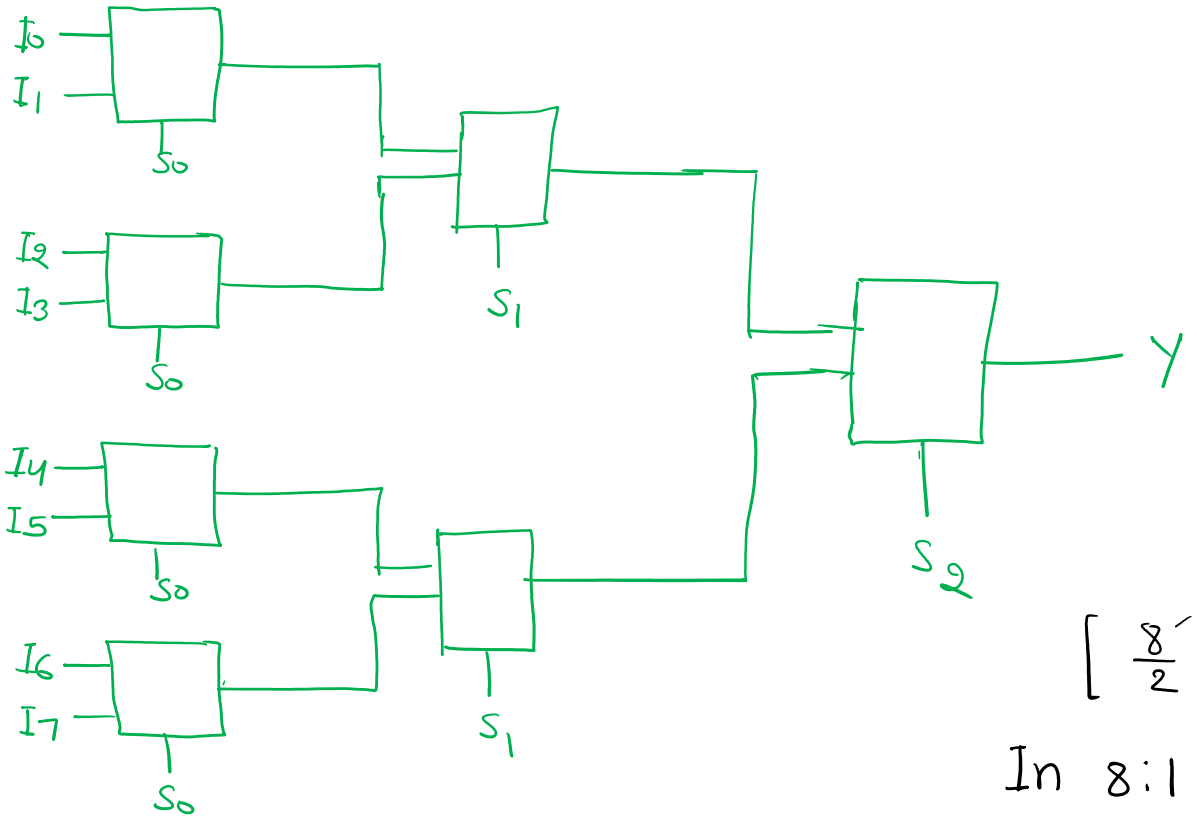
↳

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

In 4:1, number of 2:1 mux = 3

MUX

(B) Implement 8:1 mux using 2:1 mux:



$$\left[\frac{8}{2} + \frac{4}{2} + \frac{2}{2} = 4+2+1 = 7 \right]$$

In 8:1, number of 2:1 MUX = 7

MUX

(C) 16:1 ⟶ 2:1 mux

(D) 64:1 ⟶ 2:1 mux

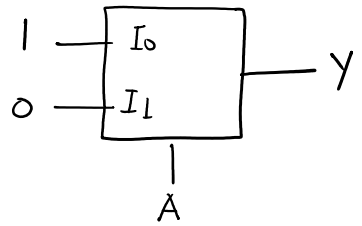
(E) 256:1 ⟶ 2:1 mux

* For 2^n:1 mux, $2^n - 1$ 2:1 mux are required.

MUX

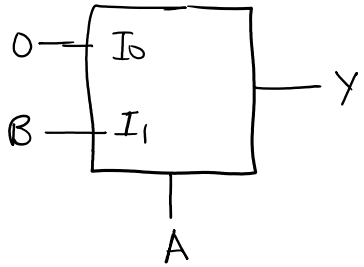
MUX as universal:

2:1 MUX:



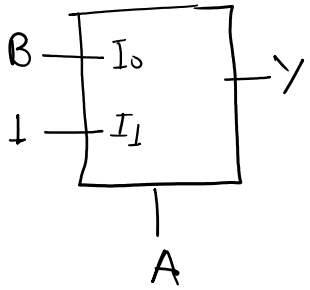
(NOT)

1 MUX (2:1) is required



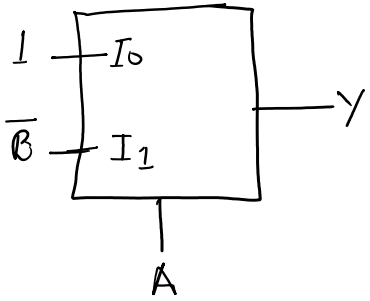
(AND)

1 MUX (2:1) is required



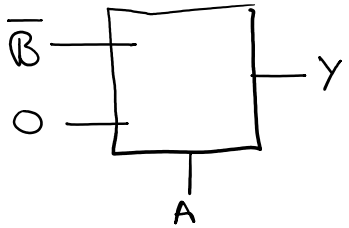
(OR)

1 MUX (2:1) is required



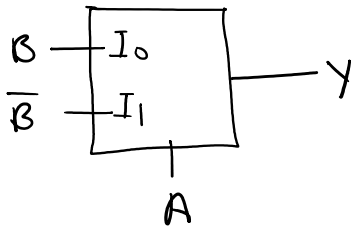
(NAND)

2 MUX (2:1) are required.



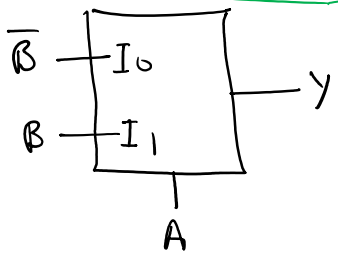
(NOR)

2 MUX (2:1)



(EX-OR)

2 MUX (2:1)



(EX-NOR)

2 MUX (2:1)

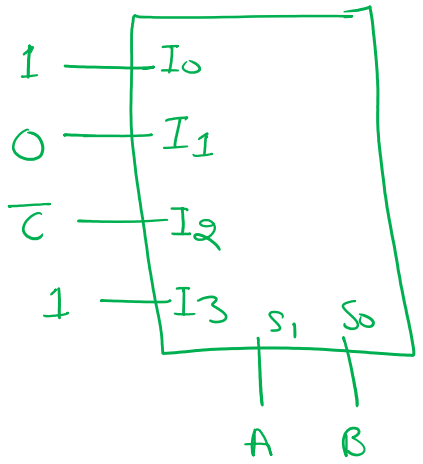
MUX

* Implementation of given logical expression:

Q: $f(A, B, C) = \Sigma m(0, 1, 4, 6, 7)$

Solution:

	00 I_0	01 I_1	10 I_2	11 I_3
$\overline{C}$	(0)	2	(4)	(6)
C	(1)	3	5	(7)



(AB as select line)

MUX

Q: Implement logical expression:

$$f(A, B, C) = \Sigma m(1, 2, 3, 5, 6, 7)$$

i) AB as select line

ii) AC : : :

iii) BC : : :

Solution:

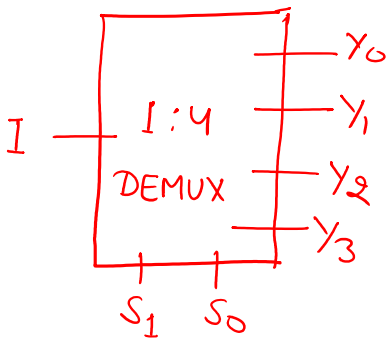
i) C, 1, C, 1

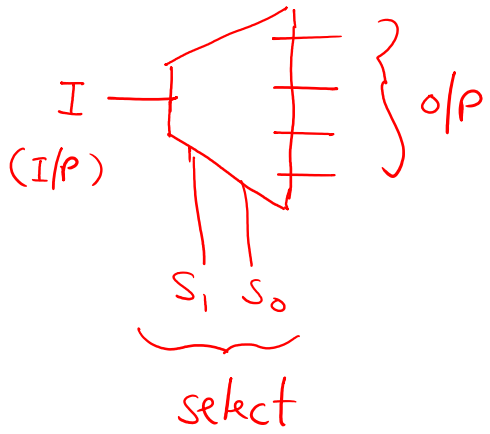
ii) B, 1, B, 1

iii) 0, 1, 1, 1

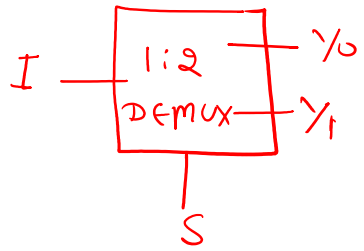
Demultiplexer (DEMUX)

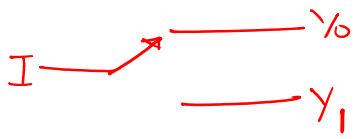
- Single input (I/P), multiple output (O/P).
- DEMUX is combinational circuit.
- Depending on the select input, I/P is transferred to one of the O/P.
- Also known as 1 to many circuit, or data distributor.





DEMUX





Truth table:

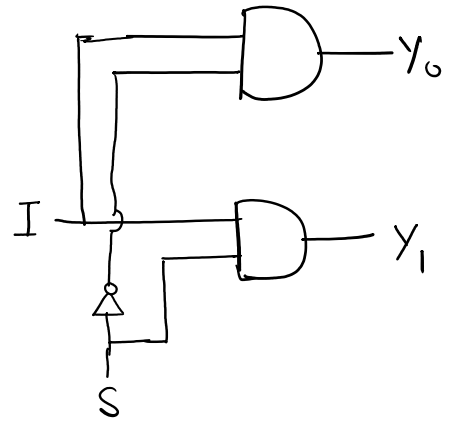
S	Y_1	Y_0
0	0	I
1	I	0

Expression:

$$Y_0 = \bar{S} I$$

$$Y_1 = S I$$

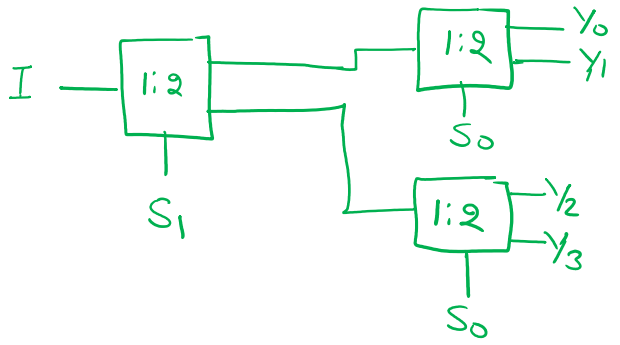
Implementation



DEMUX

* Implementation of higher order DEMUX from lower order DEMUX.

(i) 1:4 DEMUX —— 3 using —— 1:2 DEMUX



S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

(ii) 1:8 DEMUX —— ? —— 1:2 DEMUX (Ans: 7)

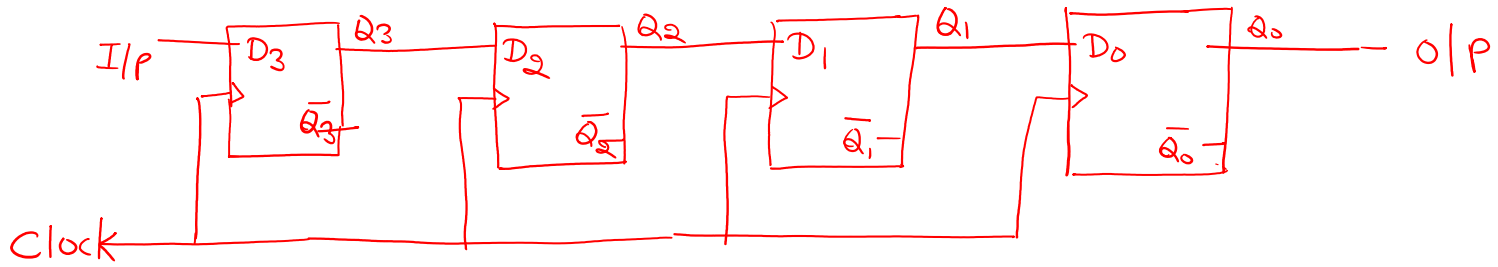
(iii) 1:64 DEMUX —— ? —— 1:8 DEMUX (Ans: 9)

Register

- Registers are used to store group of bits.
- To store n bits, n flipflops are cascaded in register.
- Registers are of four types (Depending on I/P and O/P) : -
  i. SISO (Serial In Serial Out)
  ii. SIPO (Serial In Parallel Out)
  iii. PISO
  iv. PIPO
- Depending on application, registers are of two types:
  i. Shift register
  ii. Storage register

Register

- SISO



Input	Q_3	Q_2	Q_1	Q_0	Clock
1 0 1 1	0	0	0	0	0
	1	0	0	0	1
	1	1	0	0	2
	0	1	1	0	3
	1	0	1	1	4

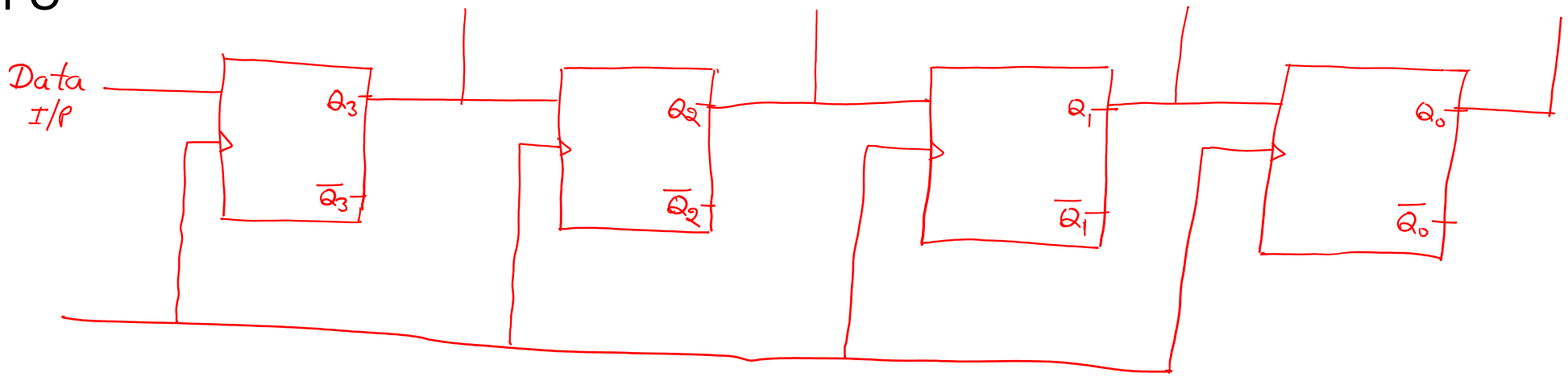
* n-bit data storage requires n clock pulse
* SISO register used to provide n clock pulse delay to I/P data.

$$\text{delay} = nT_{clk}$$

* To provide n-bit data serially out it requires (n-1) clocks.

Register

- SIPO

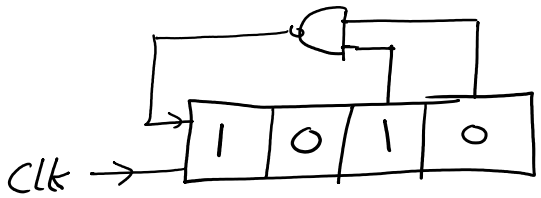


* In SIPO register
  - to provide n-bit data serially-in, it requires n-clock pulses
  - to provide parallel out it requires 0 (zero) clock pulse.

Register

- SIPO

Question: The circuit shown in the figure. is 4-bit SIPO register which is initially loaded with 1010. If three clock pulses are applied then data in the system is :



(a) 1010

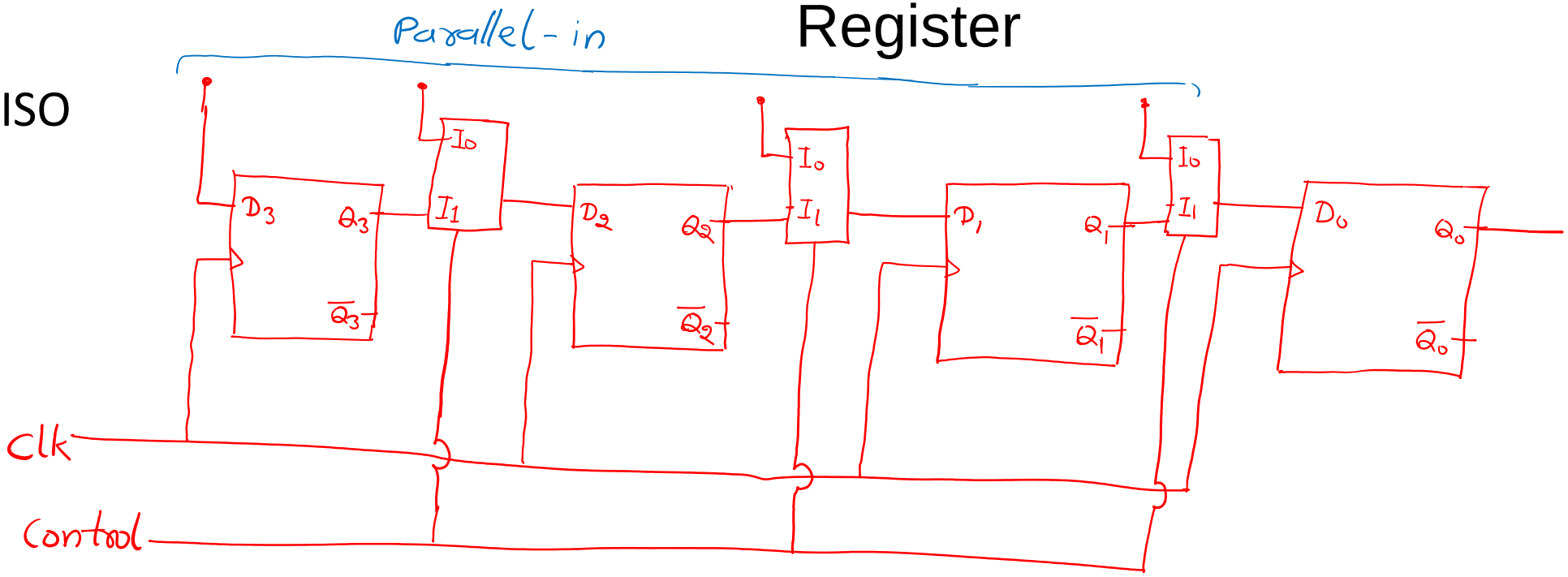
(b) 1101

(c) 1111

(d) 0000

Register

- PISO

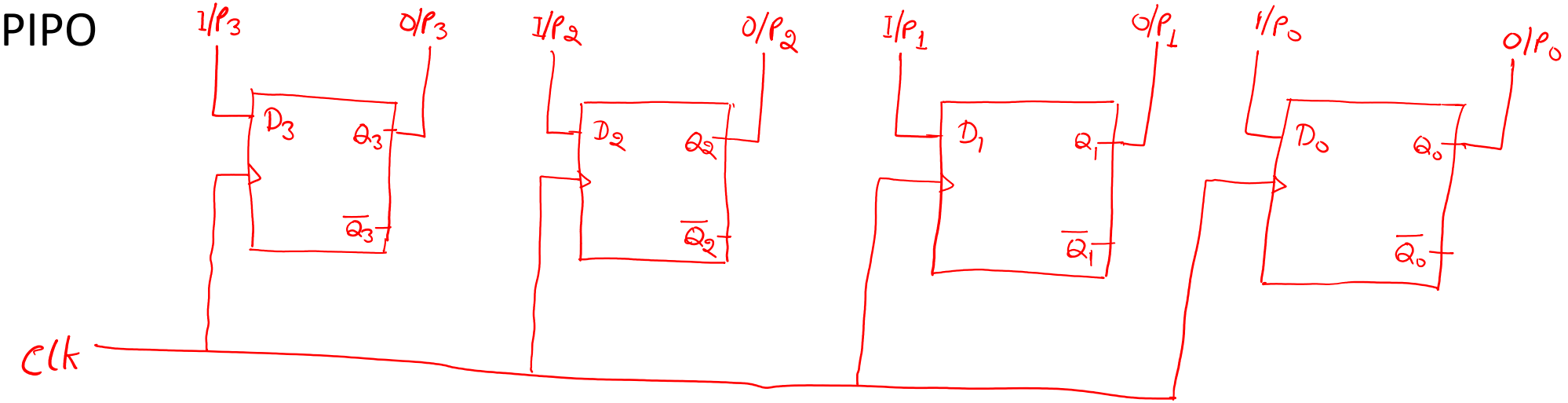


* set
  control = 0 for Parallel-in
  control = 1 for serial out

* To provide n-bit parallel-in, it requires 1 clock pulse.
  ∴ ∴ ∴ serial-out, ∴ ∴ (n-1) clock pulse.

Register

- PIPO



- for parallel-in it requires 1 clock pulse.
- ∴ parallel-out, ∴ ∴ 0 ∴ ∴ .

Counters

- Counters are basically used to count number of clock pulse applied.
- It can also be used for frequency divider, time measurement, frequency measurement, range measurement, and pulse width.
- Also used for waveform generator.
- With n-ff, maximum possible stage in the counter is 2^n.

$$N \leq 2^n$$

where N : number of stage

n : " " FF.

- Number of stage use in counter mean modulus of counter.

i.e. if MOD 5 counter then 5 stage.

Counters

- Depending on clock pulse applied, counters are of two types:

Asynchronous

1. Different FF are applied with different clock.
2. It is slower.
3. Fixed count sequence i.e up or down.
4. Example: Ripple counter

Synchronous

1. All FF are applied same clock.
2. It is faster.
3. Any count sequence is possible
4. Example: Ring counter

Counters

- NOTE:

i) -ve edge trigger ⟶ Q as clock ⟶ up counter

ii) +ve ∴ ∴ ⟶ $\overline{Q}$ ∴ ∴ ⟶ up counter

iii) −ve ∴ ∴ ⟶ $\overline{Q}$ ∴ ∴ ⟶ down counter

iv) +ve ∴ ∴ ⟶ Q ∴ ∴ ⟶ down counter

Counters

- Ripple Counter

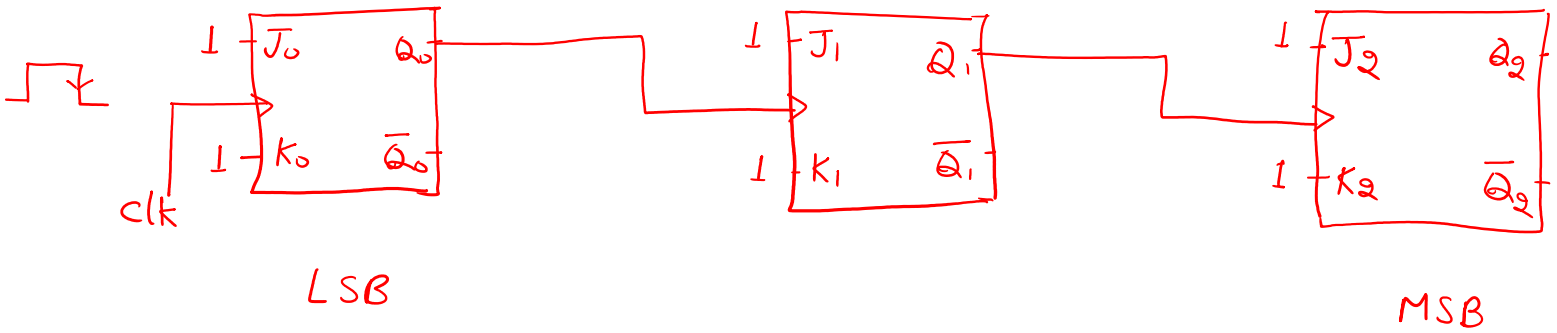
  * Only one FF is applied with external clock and other FFs clock is from previous FF output (Q or $\bar{Q}$).

  * The FF applied with external clock acts as LSB.

Counters

- Ripple Counter:

3-bit ripple counter (up counter) :-



* Q_n changes when Q_{n-1} changes from $1 \rightarrow 0$

Truth table:

clk	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

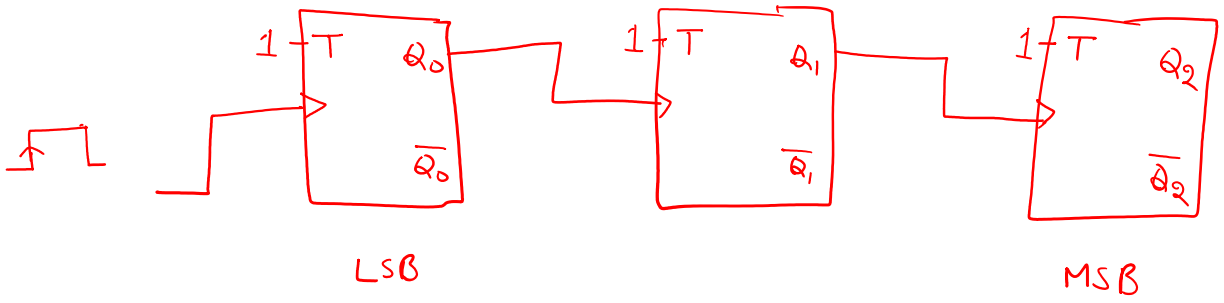
⇒ Up counter

⇒ It is also called MOD 8 ripple counter.

Counters

- Ripple Counter

* 3-bit ripple counter (Down-counter):



* Q_0 toggles for every clock pulse.
* Q_1 toggles when Q_0 changes from $0 \rightarrow 1$.
* Q_2 : : Q_1 : : : .

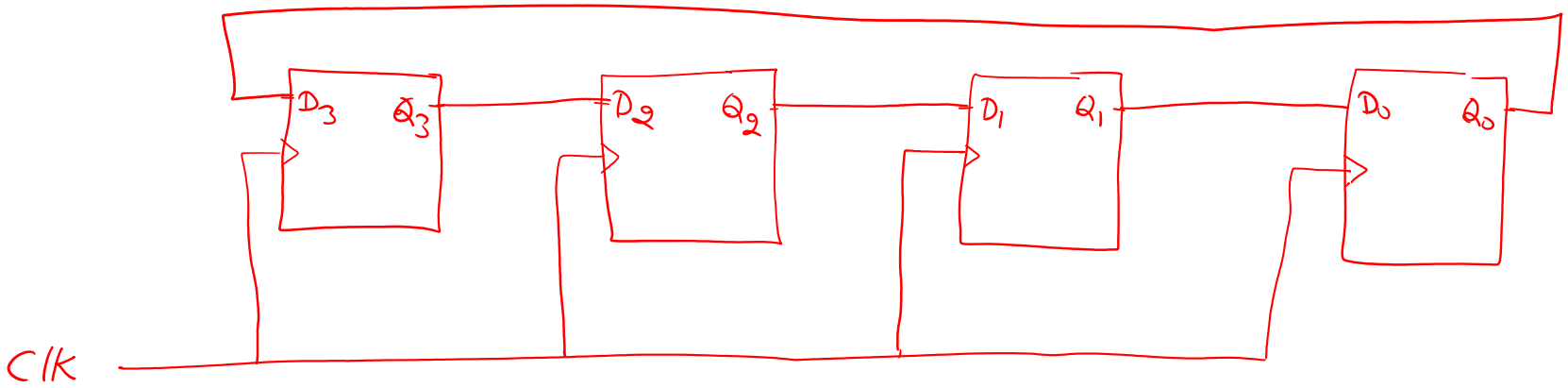
Truth table:

Clock	Q_2	Q_1	Q_0
0	0	0	0
1	1	1	1
2	1	1	0
3	1	0	1
4	1	0	0
5	0	1	1
6	0	1	0
7	0	0	1
8	0	0	0

Counters

- Ring Counter

Ring Counter:



* In 4-bit ring counter, 4 states are there. (ie: n FFs., n states)

Truth table:

CLK	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	
1	1	0	0	0	} 4-states
2	0	1	0	0	
3	0	0	1	0	
4	0	0	0	1	
5	1	0	0	0	

Binary Number System

- A computer is an information-processing machine that works by converting all kinds of information into binary numbers (1s and 0s). A computer treats any type of information (numbers, letters, words, etc.) as if it consisted simply of binary ones and zeros.
- The binary system is a way of representing data using 0s and 1s. This system is used by computers to represent all the data it works with.
- Binary Number System is a number system that is used to represent various numbers using only two symbols “0” and “1”.

- In the Binary Number System, we have two states “0” and “1” and these two states are represented by two states of a transistor. If the current passes through the transistor then the computer reads “1” and if the current is absent from the transistor then it read “0”. Thus, alternating the current the computer reads the binary number system.
- Each digit in the binary number system is called a “bit”.
- In the Binary Number System, we have a base of 2. The base of the Binary Number System is also called the radix of the number system.

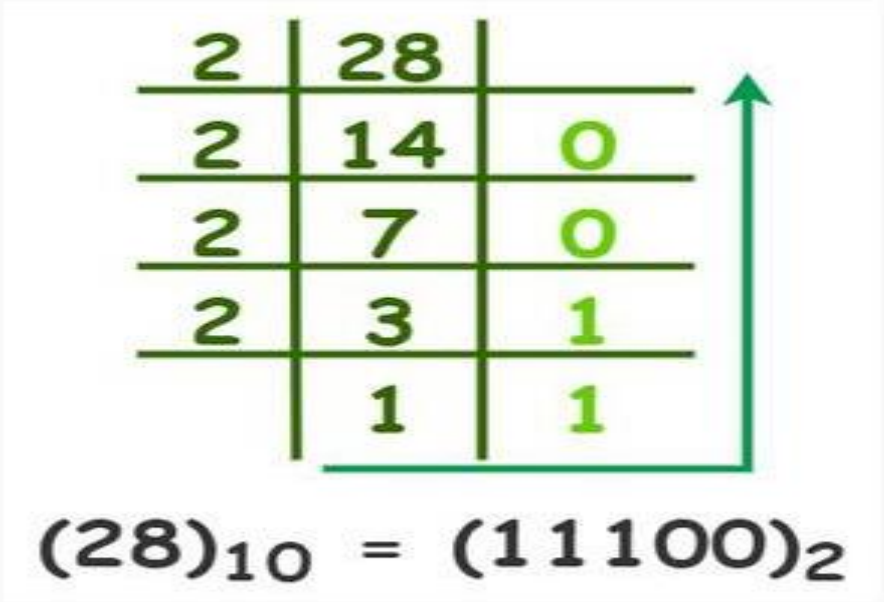
Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Decimal to Binary Conversion

- A decimal number is converted into a binary number by dividing the given decimal number by 2 continuously until we get the quotient as 1, and we write the numbers from downwards to upwards.

Example: Convert $(28)_{10}$ into a binary number.

Solution:



Example

10.75

Integral Part

$10 = (1010)_2$

Fractional Part

0.75 * 2 =>1.50 // take 1 and move .50 to next step

0.50 * 2 =>1.00 // take 1 and stop the process because no remainder

$0.75 = (11)_2$

Combining both integral and fractional,

$10.75 = (1010.11)_2$

Binary to Decimal Conversion

- A binary number is converted into a decimal number by multiplying each digit of the binary number by the power of either 1 or 0 to the corresponding power of 2.

Example: Convert $(10011)_2$ to a decimal number.

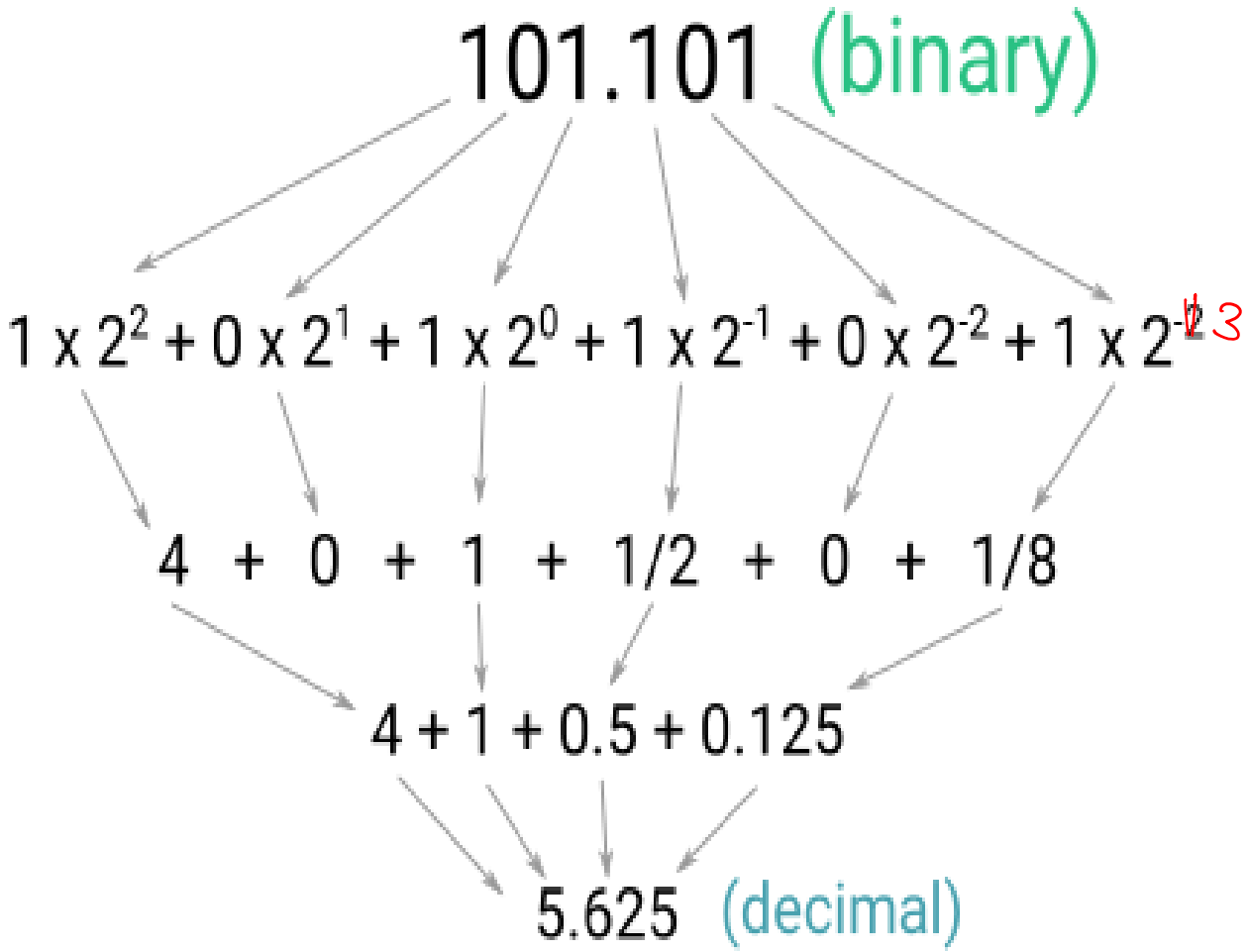
Solution:

The given binary number is $(10011)_2$.

$$(10011)_2 = (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$= 16 + 0 + 0 + 2 + 1 = (19)_{10}$$

Hence, the binary number $(10011)_2$ is expressed as $(19)_{10}$.


101.101 (binary)
1 x 2^2 + 0 x 2^1 + 1 x 2^0 + 1 x 2^-1 + 0 x 2^-2 + 1 x 2^-3
4 + 0 + 1 + 1/2 + 0 + 1/8
4 + 1 + 0.5 + 0.125
5.625 (decimal)

Binary Addition

- The four rules of binary addition are:

  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 =10 (write 0 and carry 1 to the next bit)

Example 1: 10001 + 11101

Solution:

```
          1
    1 0 0 0 1
(+) 1 1 1 0 1
-----------
  1 0 1 1 1 0
```

Example 2: 10111 + 110001

Solution:

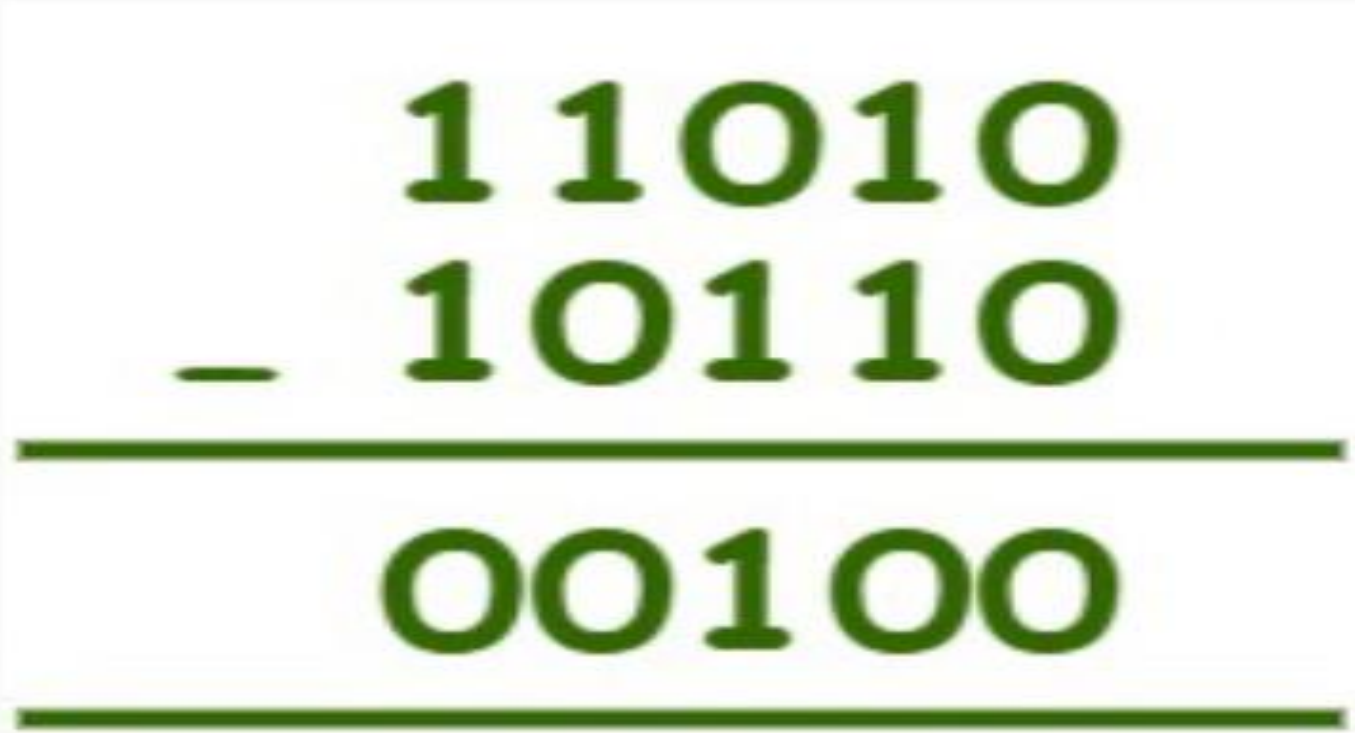
```
        1 1 1
        1 0 1 1 1
(+) 1 1 0 0 0 1
---------------
  1 0 0 1 0 0 0
```

Binary Subtraction

- **Rules and tricks:** Binary subtraction is much easier than the decimal subtraction when you remember the following rules:
  - 0 – 0 = 0
  - 0 – 1 = 1 (with a borrow of 1)
  - 1 – 0 = 1
  - 1 – 1 = 0

Example 5: Subtract $(11010)_2$ and $(10110)_2$

Solution:



Hence, $(11010)_2 - (10110)_2 = (00100)_2$