Polytechnic Version

Quick Learner : Computer System Architecture

Sharmilla binti Sulong | Adibah binti Ali
Politeknik Kuala Terengganu

QUICK LEARNER : COMPUTER SYSTEM ARCHITECTURE

First Published 2022
e ISBN 978-967-2240-35-8
Politeknik Kuala Terengganu

Published by :
Politeknik Kuala Terengganu
20200 Jalan Sultan Ismail,
Kuala Terengganu, Terengganu

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data
Sharmilla Sulong, 1977-
Quick Learner : Computer System Architecture / Sharmilla Binti Sulong,
Adibah Binti Ali. – Polytechnic Version.
Mode of access: Internet
eISBN 978-967-2240-35-8
1. Computer systems.
2. Computer architecture.
3. Computer organization.
4. Government publications--Malaysia.
5. Electronic books.
I. Adibah Ali, 1979-.
II. Title.
004

Abstract

This eBook designed for student to understand the basic concepts on which the stored program digital computer is formulated. The content of this eBook is written for polytechnic students.

This eBook introduces the basic knowledge of computer architecture and computer organization. It focuses on describing of function of each unit in Computer System in Chapter 1, applying appropriate method to solve arithmetic problem in numbering system and sequential logic circuit in Chapter 2, writing assembly program in Chapter 3 and foundation knowledge of Central Processing Unit in Chapter 4.

As there already many books written on computer architecture in the market, this eBook attempts to distinct itself by using mind mapping to help students visualize the concepts easily and practice drill after completing a topic. This allows students to better understand the topics they have learned while reinforcing existing skills. Therefore, this eBook helps students revise the topics taught in the classroom without having to read the notes that long and tedious.

It is hoped that this eBook will help both lecturer and students in making classroom learning as enjoyable as possible. This eBook can also be used for independent self-learning.

Table of Content

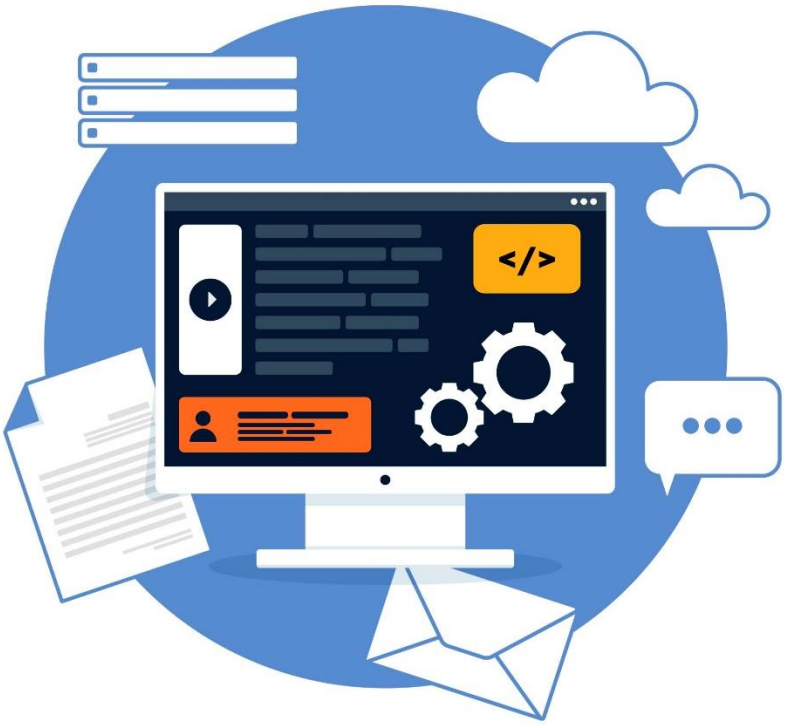
This chapter describes briefly the computer system.

Topic : The Computer System

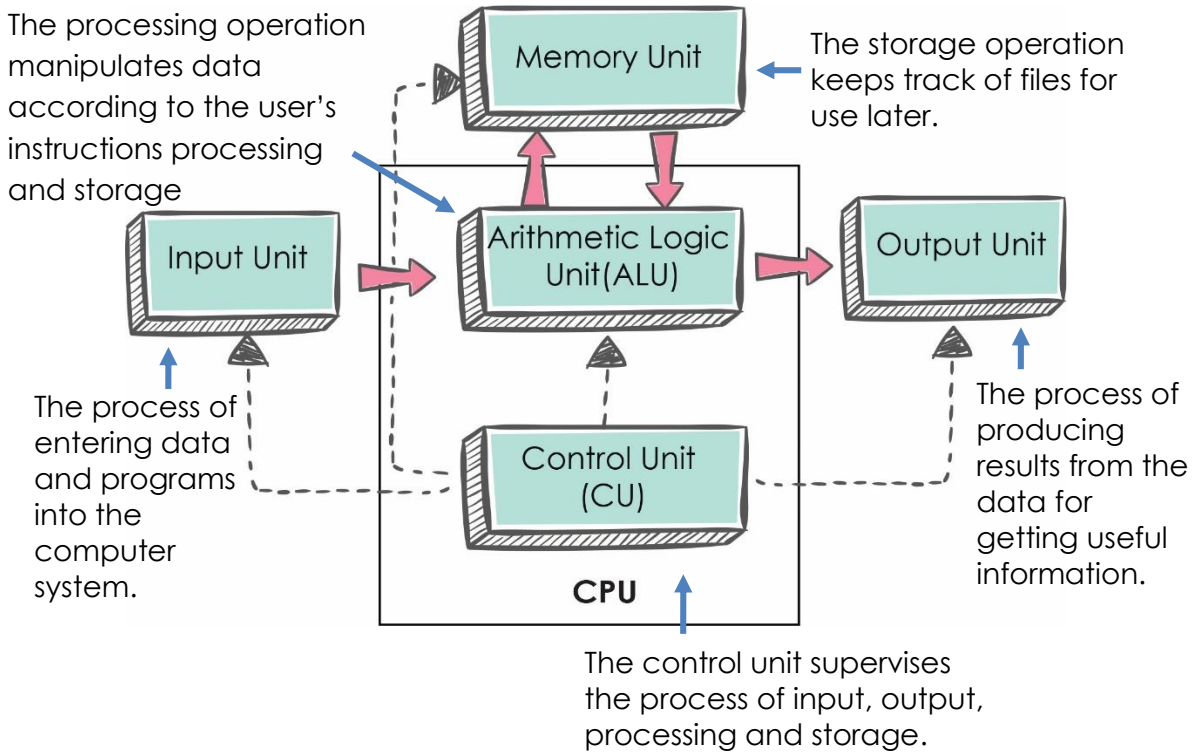
Introduction

A computer system is a basic, complete and functional hardware and software setup with everything needed to implement computing performance. Computer System is a collection of entities (hardware, software and liveware) that are designed to receive, process, manage and present information in a meaningful format.

A computer system, therefore, is a computer combined with peripheral equipment and software so that it can perform desired functions. The components of a computer are designed to interact with one another, and this interaction plays an important role in the overall system operation.



Block Diagram of Computer System

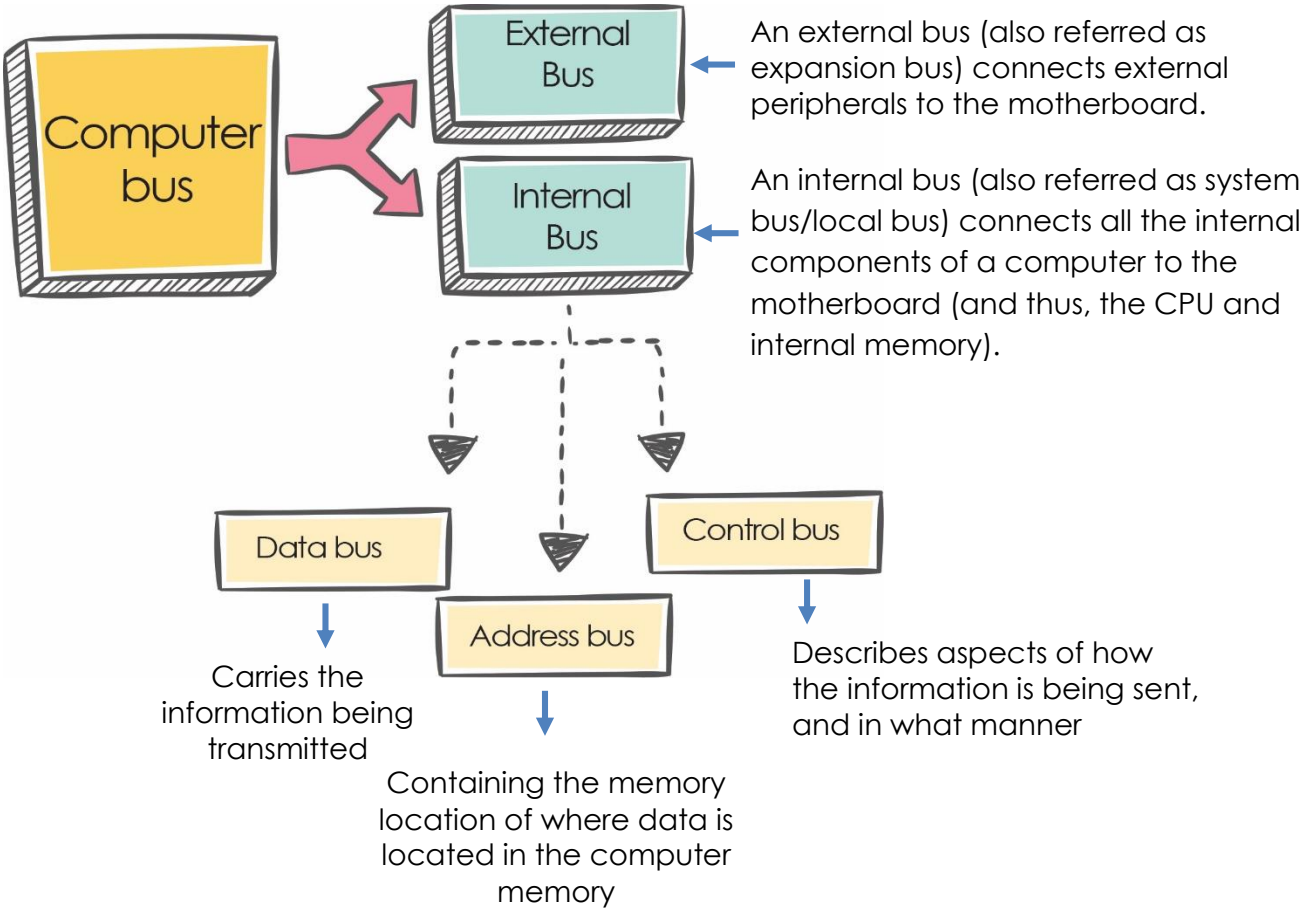


Computer Bus

In computer architecture, a bus is a communication system that transfers data between components inside a computer, or between computers. Computer bus is a subsystem that transfers data between components inside a computer, or between computers.

The bus contains multiple wires with addressing information describing the memory location of where the data is sent or retrieved. Each wire in the bus carries a bit(s) of information, which means the more wires a bus has, the more information it can address.

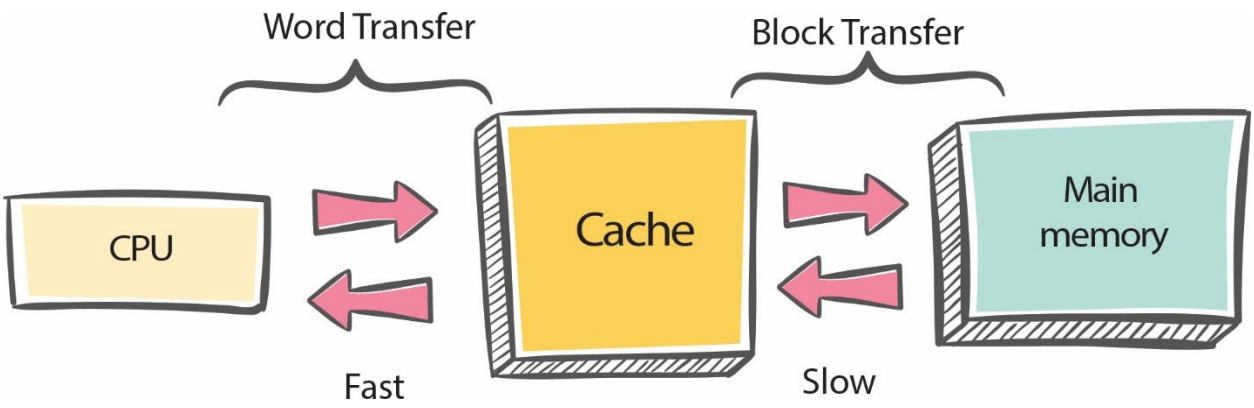
Types of Bus



The most common buses and how they are used with a computer.

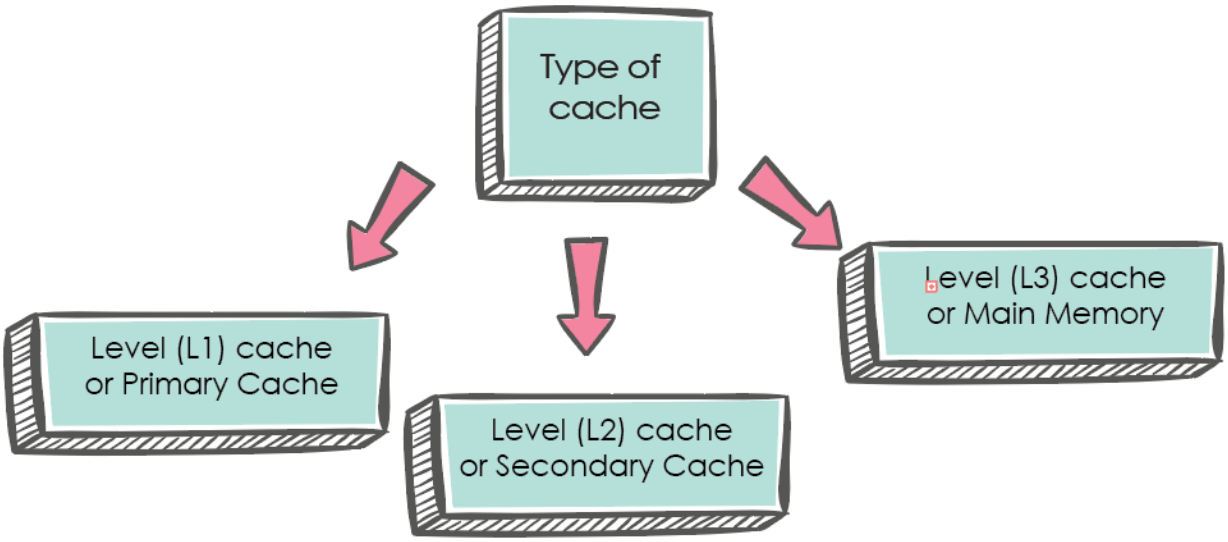
- **eSATA** and **SATA** - Computer hard drives and disc drives.
- **PCIe** - Computer expansion cards and video cards.
- **USB** - Computer peripherals.
- **Thunderbolt**- Peripherals connected through a USB-C cable.

Cache



- A special very-high-speed memory called a cache, is used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.
- The transformation of data from main memory to cache memory is called mapping. The mapping functions are used to map a particular block of main memory to a particular block of cache. This mapping function is used to transfer the block from main memory to cache memory.

Type of Cache Memory



- **Level 1 (L1) cache or Primary Cache**
  - the fastest memory that is present in a computer system
  - The Size of the L1 cache very small comparison to others that is between 2KB to 64KB, it depends on computer processor.
  - It is an embedded register in the computer microprocessor(CPU).
  - The Instructions that are required by the CPU that are firstly searched in L1 Cache.

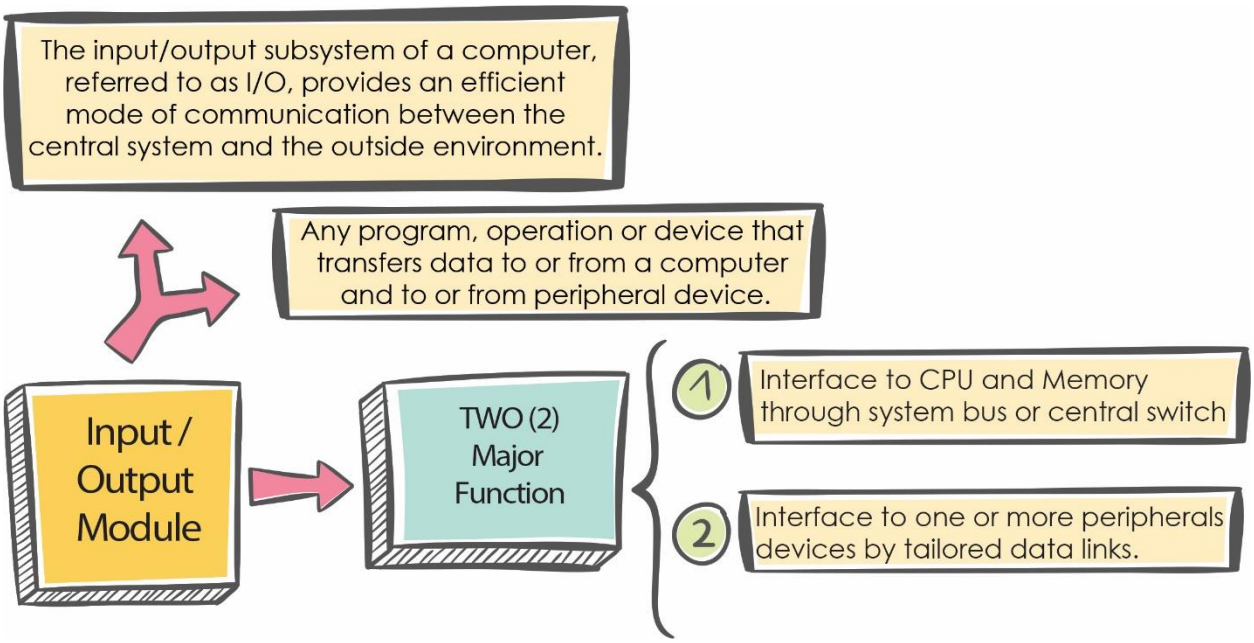
- **Level 2 (L2) cache or Secondary Cache**
  - Slower than L1 cache, but bigger in size
  - The size of the L2 cache is more capacious than L1 that is between 256KB to 512KB.
  - L2 cache is Located on computer microprocessor.
  - After searching the Instructions in L1 Cache, if not found then it searched into L2 cache by computer microprocessor.

- **Level 3 (L3) cache or Main Memory**
  - The L3 cache is larger in size but also slower in speed than L1 and L2,
  - It's size is between 1MB to 8MB.
  - In Multicore processors, each core may have separate L1 and L2, but all core share a common L3 cache.
  - L3 cache double speed than the RAM.

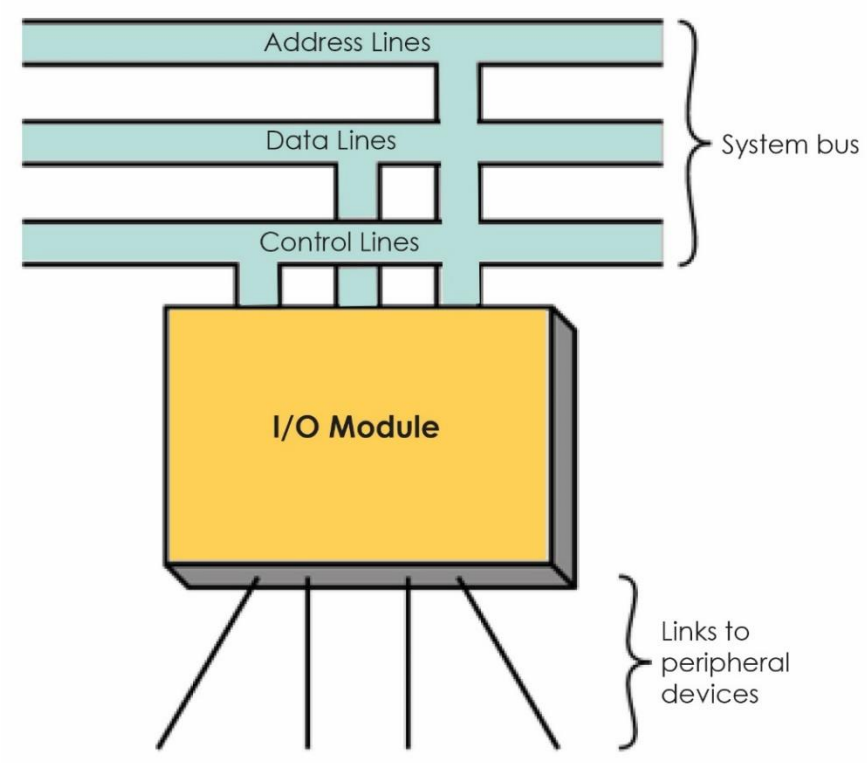
Input/Output Module

I/O module stands for Input/Output module, which is a device that acts as the connective bridge between a computer system at one end and an I/O or peripheral device of some kind at the other, such as a printer, webcam or scanner.

Input/Output Modules (I/O Modules) act as mediators between the processor and the input/output devices.



Generic Model of I/O Module

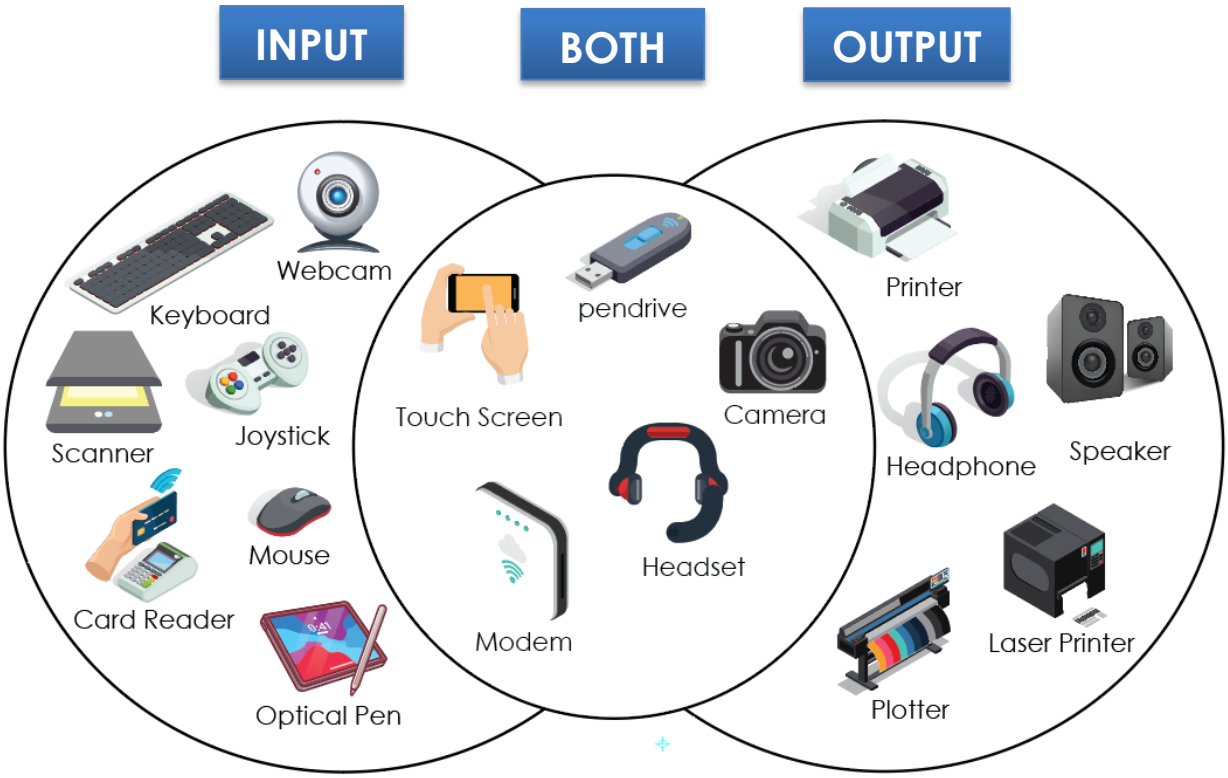


- Interface to CPU and Memory via system bus or central switch
- Interface to one or more peripherals devices by tailored data links.

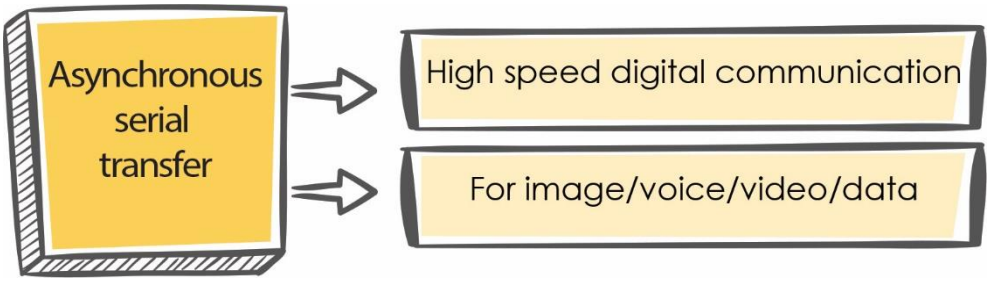
Input/Output Device

An input/output (I/O) device also called as **IO Device** is a hardware device that has the ability to accept **inputted**, **outputted** or **other processed** data.

IO devices allow the computer system to interact with the outside world by moving data into and out of the system. An **input device** is used to bring data into the system. An **output device** is used to send data out of the system.

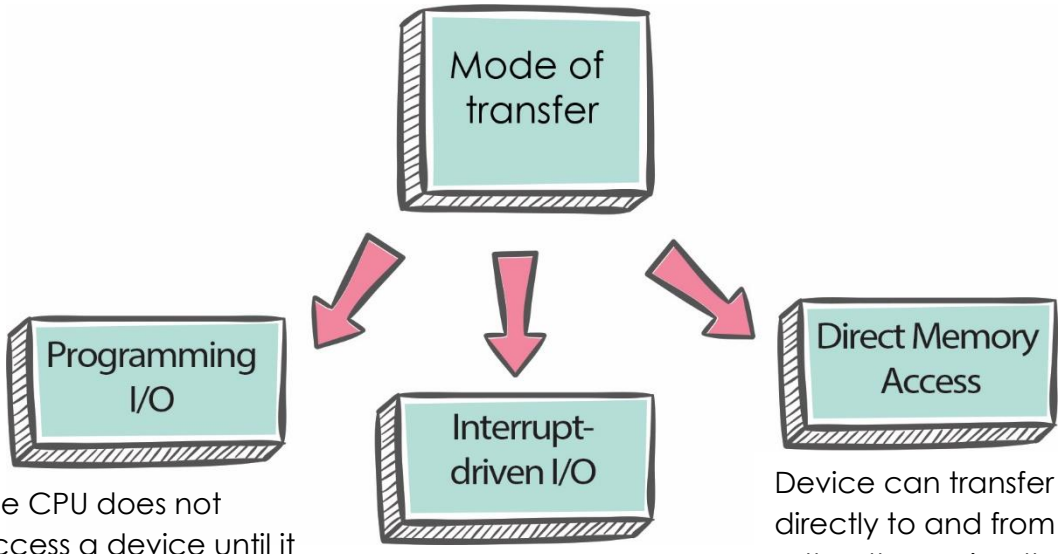


Asynchronous Serial Transfer



Parallel VS Serial

Parallel	Serial
Transfer all at a time	Each bit sent in sequence at a time
Fast	Slow
Require many wires	Require 1 pair of conductor



The CPU does not access a device until it needs servicing, and so it does not get caught up in busy-waits.

The processor issues an I/O command, continues to execute other instruction, and is interrupted by the I/O module when the latter has completed its work.

Device can transfer data directly to and from memory rather than using the CPU as an intermediary, and can thus relieve congestion on the system bus.

Activity 1

1. Draw a block diagram to illustrate the basic organization of computer system and describe the functions for each unit.

2. A bus is a communication pathway connecting two or more device. Describe the concept of interconnection within a computer system as follows :
   a. Draw interconnection structures

b. Describe the functions for each bus

3. Draw the I/O Module Diagram and explain how it works.

4. Describe the I/O bus and interface modules

Activity 2

Instruction : Answer all questions.

I/O Module	Programming I/O	Address Bus	Data Bus
Interrupt Driven I/O	Direct Memory Access	Control Bus	Tags

i. _____ carries the information being transmitted.

ii. _____ identifies where the information is being sent.

iii. _____ describes aspects of how the information is being sent, and in what manner.

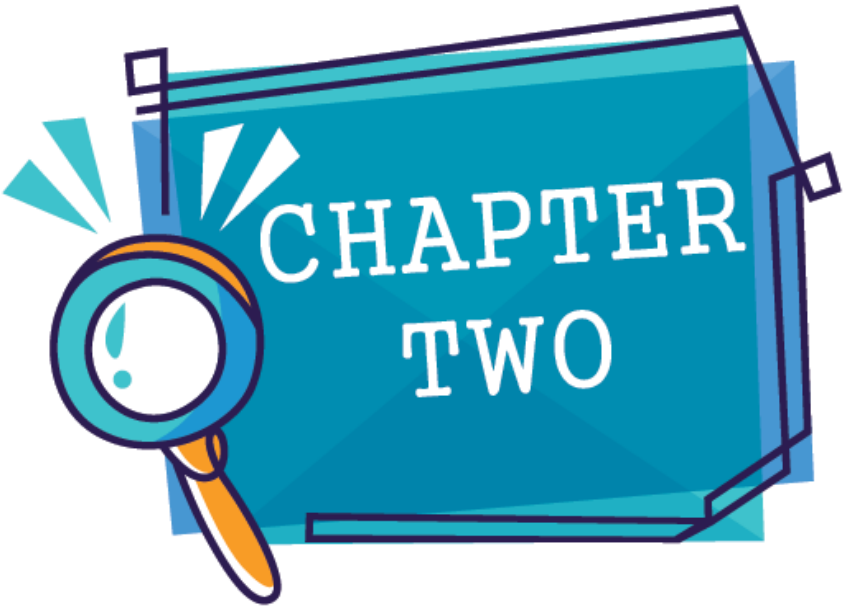
iv. With _____, data are exchanged between the processor and the I/O module.

v. With _____, the processor issues an I/O command, continues to execute other instruction, and is interrupted by the I/O module when the latter has completed its work.

vi. A _____ device can transfer data directly to and from memory rather than using the CPU as an intermediary.

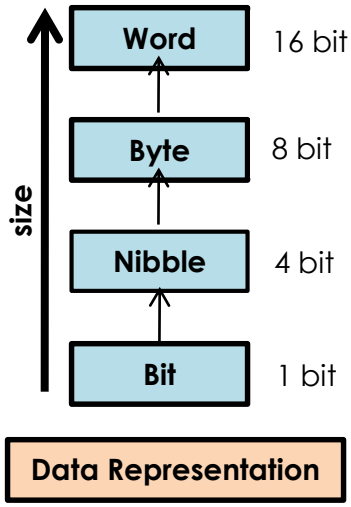
vii. Any program, operation or device that transfers data to or from a computer and to or from peripheral device is called _____.

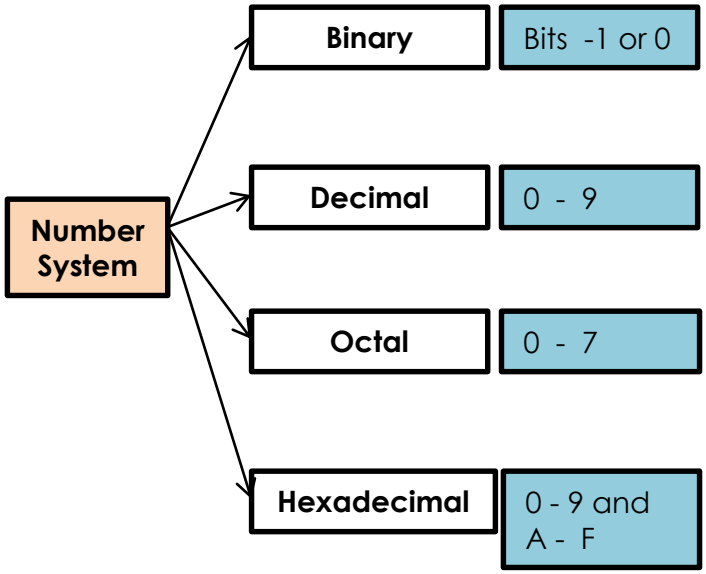
viii. _____ are used identify where cached data originated.



This chapter focuses on the method to solve arithmetic problem in numbering system and sequence logic circuit.

Topic : Data Representation





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

One to One Comparison

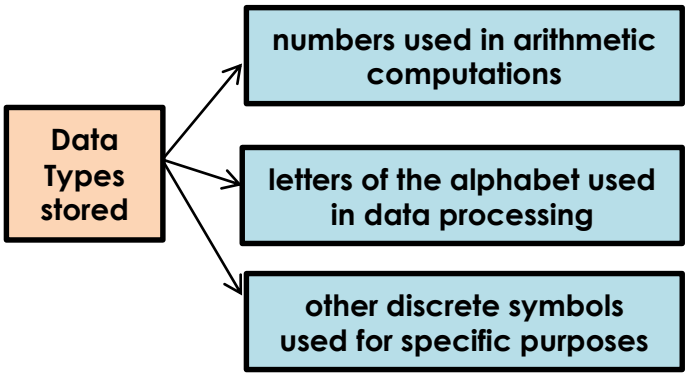


Table : Conversion of Number System

From \ To	DECIMAL	BINARY	OCTAL	HEXADECIMAL
DECIMAL		Devide by 2	Devide by 8	Devide by 16
BINARY	Multiply each bit by 2^n		-Group bit in 3's, starting on right. -Convert to octal digit.	-Group bit in 4's, starting on right. -Convert to hexa digit.
OCTAL	Multiply each bit by 8^n	-Convert each octal digit to a 3-bit equivalent binary representation.		-Use a binary as an intermediary
HEXADECIMAL	Multiply each bit by 16^n	-Convert each hexa digit to a 4-bit equivalent binary representation.	-Use a binary as an intermediary	

CONVERSION OF NUMBER BASES

Decimal to Binary, Octal, Hexadecimal

Convert 174_{10} to binary:

Division by 2	Quotient	Remainder
174/2	87	0
87/2	43	1
43/2	21	1
21/2	10	1
10/2	5	0
5/2	2	1
2/2	1	0
1/2	0	1

So 174_{10} = $\mathbf{10101110_2}$

Convert 1792_{10} to octal :

Division By 8	Quotient	Remainder
1792/8	224	0
224/8	28	0
28/8	3	4
3/8	0	3
0	done.	

So 1792_{10} = $\mathbf{3400_8}$

Convert 1792_{10} to hexadecimal :

Division By 16	Quotient	Remainder
1792/16	112	0
112/16	7	0
7/16	0	7
0	done.	

So 1792_{10} = $\mathbf{700_{16}}$

Convert $(0.6875)_{10}$ to binary :

0.6875 x 2 = 1.3750
0.3750 x 2 = 0.7500
0.7500 x 2 = 1.5000
0.5000 x 2 = 1.0000
0.0000

So 0.6875_{10} = $\mathbf{0.1011_2}$

CONVERSION OF NUMBER BASES

Binary to Octal, Decimal and Hexadecimal

Convert 111001_2 to decimal:

binary number:	1	1	1	0	0	1
power of 2:	2^5	2^4	2^3	2^2	2^1	2^0

$111001_2 =$
$1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
$= 57_{10}$

Convert 111001_2 to octal:

111 001
7 1

Octal:	0	1	2	3	4	5	6	7
Binary:	000	001	010	011	100	101	110	111

So $111001_2 = 71_8$

Convert 111001_2 to hexadecimal

Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal:	0	1	2	3	4	5	6	7

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

0011 1001
3 9

So $111001_2 = 39_{16}$

Convert $(101.01)_2$ to decimal:

1 0 1. 0 1
↓ ↓ ↓ ↓ ↓
2^2 2^1 2^0 2^{-1} 2^{-2}
$4 + 0 + 1 + 0 + 1/2^2 = 5.25$

$(101.01)_2 \rightarrow (5.25)_{10}$

CONVERSION OF NUMBER BASES

Octal to Binary, Decimal and Hexadecimal

Convert 37_8 to decimal :

$3\times8^1+7\times8^0 = 24+7 = 31$

So $37_8 = 31$

Convert 37_8 to binary :

Octal: 0 1 2 3 4 5 6 7
Binary: 000 001 010 011 100 101 110 111

3 **7**
011 111

So $37_8 = 011111_2$

Convert 37_8 to hexadecimal :

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

1. **3** **7**
   011 111
2. 011111 -> 0001 1111
   1 F
3. $37_8 = 1F_{16}$

Convert 37.45_8 to binary :

3 **7** . **4** **5**
011 111 100 101

So $37.45_8 = 011111 . 100101_2$

CONVERSION OF NUMBER BASES

Hexadecimal to Binary, Decimal and Octal

Convert $7DE_{16}$ to decimal:

$= (7 * 16^2) + (13 * 16^1) + (14 * 16^0)$

$= (7 * 256) + (13 * 16) + (14 * 1)$

$= 1792 + 208 + 14$

$= 2014$

So $\mathbf{7DE_{16}} = 2014$

Convert $7DE_{16}$ to binary :

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

7	**D**	**E**
0111	1101	1110

So $\mathbf{7DE_{16}} = 011111011110_2$

Convert $7DE_{16}$ to Octal :

1.

7	**D**	**E**
0111	1101	1110

2.

011	111	011	110
3	7	3	6

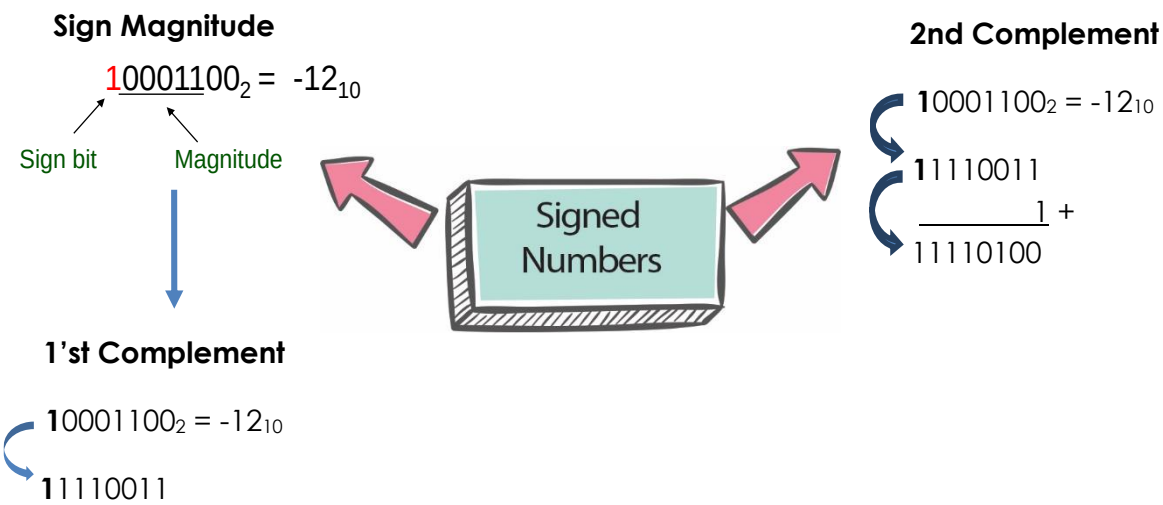
3. $\mathbf{7DE_{16}} = 3736_8$

Convert $7DE.\ 1A_{16}$ to binary :

7	**D**	**E**	**.**	**1**	**A**
0111	1101	1110		0001	1010

So $\mathbf{7DE.\ 1A_{16}} = 011111011110\ .\ 00011010_2$

How To Represent Signed Numbers



Sign Addition In 2's Complement

Example 1 :

+ 6	00000110
+13	00001101
+19	00010011

Example 2 :

- 6	11111010
+13	00001101
+7	00000111

Example 3 :

+6	00000110
-13	11110011
-7	11111001

Example 4 :

-6	11111010
-13	11110011
-19	11101101

Overflow Example :

+70	0 1000110	-70	1 0111010
+80	0 1010000	-80	1 0110000
+150	1 0010110	-150	0 1101010

** An overflow may occur if the two numbers added are both either positive or negative.

Arithmetic Operation In Different Number Bases

Binary Addition and Subtraction

Example : $(11110)_2$ to $(10111)_2$

Solution:

```
  1  1  1  1  1  1 <------------ carries
     1  1  1  1  0  1
+       1  0  1  1  1
---------------------
  1  0  1  0  1  0  0
```

Example : subtract (10111) from (1001101)

Solution:

```
          1        10      <------------ borrows
     0 ~~1~~0 10  0 ~~0~~ 10

    ~~1~~ ~~0~~ ~~0~~ ~~1~~ ~~1~~ ~~0~~ 1
-           1  0  1  1  1
------------------------
     0  1  1  0  1  1  0
```

Octal Addition and Subtraction

Example : $(136)_8 + (636)_8$

Solution:

```
     1          <---- carry
   1 3 6
   6 3 6
   -----
   7 7 4
```

Example : $(765)_8 - (446)_8$

Solution:

```
     5 8+5
   7~~6~~ ~~5~~
   4 4 6
   -----
   3 1 7
```

Hexadecimal Addition and Subtraction

Example : Add $(59F)_{16}$ and $(E46)_{16}$

Solution:

```
        Carry
        1 1
        5 9 F
      + E 4 6
      ---------
      1 3 E 5
```

F + 6 = $(21)_{10}$ = (16 x **1**) + 5 = 15

5 + E = $(19)_{10}$ = (16 x **1**) + 3 = 13

Example : $(3F57A)_{16}$ - $(C85E)_{16}$

Solution:

```
          E   21  6   26
      3   ~~F~~ ~~5~~ ~~7~~ ~~A~~ 16
-         C   8   5   E 16
-----------------------------
      3   2   D   1   B 16
-----------------------------
```

Binary Coded Decimal (BCD) or 8421 Code

The binary coded decimal code, abbreviated as BCD, is a method that uses binary digits "0" and "1". ON state represents "1" and OFF state represents "0". Each digit is called a bit. This coding system has been used since the first computer. This coding system deals with decimal and binary numbers.
Each decimal number requires 4 bits to code them.

BCD is a decimal number with each **digit encoded** to its **binary equivalent**. **Each digit** of a decimal is represented by its **four-bit binary equivalent** (1 to 9). A BCD number is **not** the same as a straight binary number. The primary advantage of BCD is the **relative ease of converting** to and from decimal.

a) Convert the number **874_{10} to BCD 8421:**

8	7	4	(decimal)
1000	0111	0100	(BCD)

874_{10} = 1000 0111 0100 $_{BCD8421}$

b) Convert **0110100000111001_{BCD}** to decimal

0110	1000	0011	1001
6	8	3	9

0110100000111001_{BCD} = 6839_{10}

Activity 1

1. Perform arithmetic operations (additional and subtraction) in different number base.

   i) Perform the following additional in the binary number system.

a) $1101101_2 + 1010_2$	b) $1001_2 + 111_2$
c) $1100_2 + 101_2$	d) $11111_2 + 1111_2$
e) $356_8 + 176_8$	f) $AB89_{16} + ABCD_{16}$

ii) Perform the following subtractions in the binary number system.

a) $1101_2 - 110_2$	b) $1100_2 - 101_2$
c) $1001_2 - 11_2$	d) $10001_2 - 1011_2$
e) $726_8 - 473_8$	f) $ABCF_{16} - 6ED_{16}$

2. Convert binary, octal and hexadecimal numbers to different bases and vice-versa.

   i) Convert each of the following binary numbers into its equivalent in the octal and hexadecimal.

a) 1110_2	b) 101011010_2
c) 1010001011_2	d) 11100100110_2
e) 11011010_2	f) 11111110_2

ii) Convert each of the following octal numbers into its equivalent in the binary number.

a) 37_8	b) 724_8
c) 61_8	d) 45_8
e) 71.45_8	f) 23.146_8

iii) Convert each of the following hexadecimal numbers into its equivalent in the binary number.

a) $1C_{16}$	b) $F2_{16}$
c) 45_{16}	d) $8EA_{16}$
e) $ABC.12_{16}$	f) $47.5B_{16}$

Activity 2

1. Show the number below to sign magnitude, 1's complement and 2's complement.

Number	Sign Magnitude	1's Complement	2's Complement
i. + 17			
ii. – 45			
iii. – 34			

2. Solve the problem below by using 2nd complement:

   a. 45 - 26

   b. -17 + 19

   c. $7A_{16} - 15_{16}$

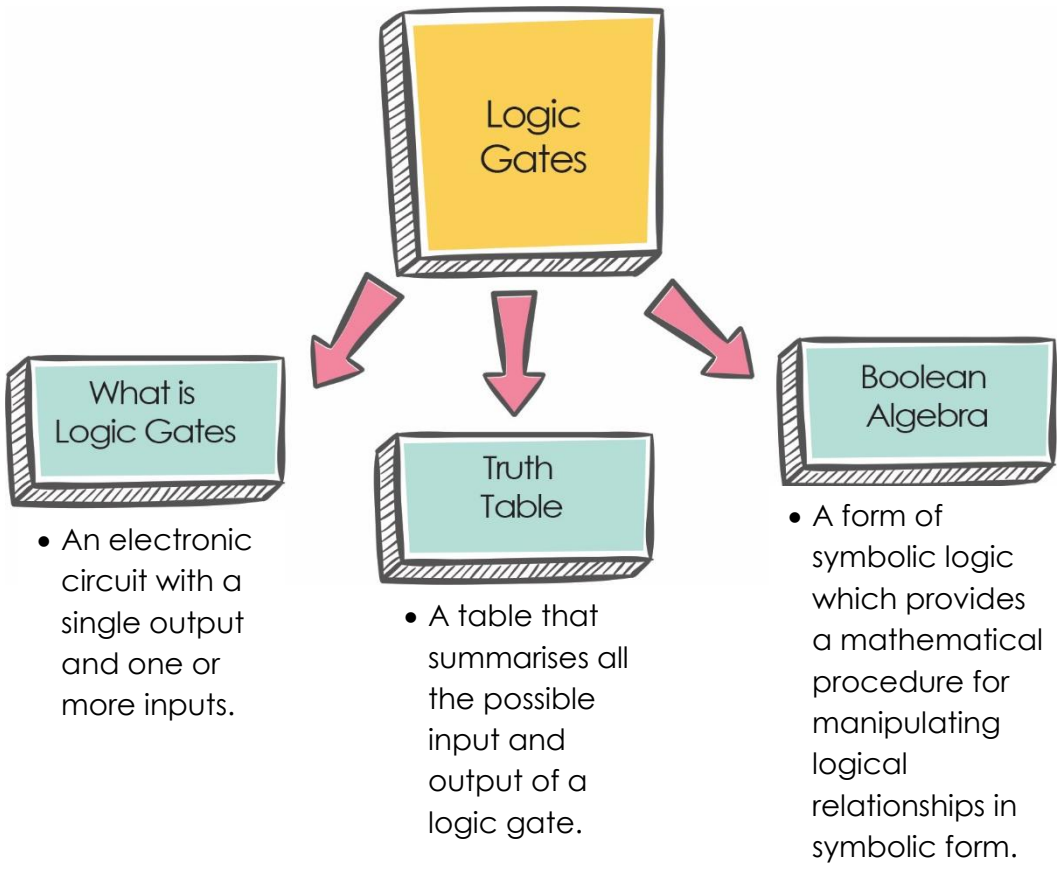
3. Write the following decimal numbers into BCD 8421 code.

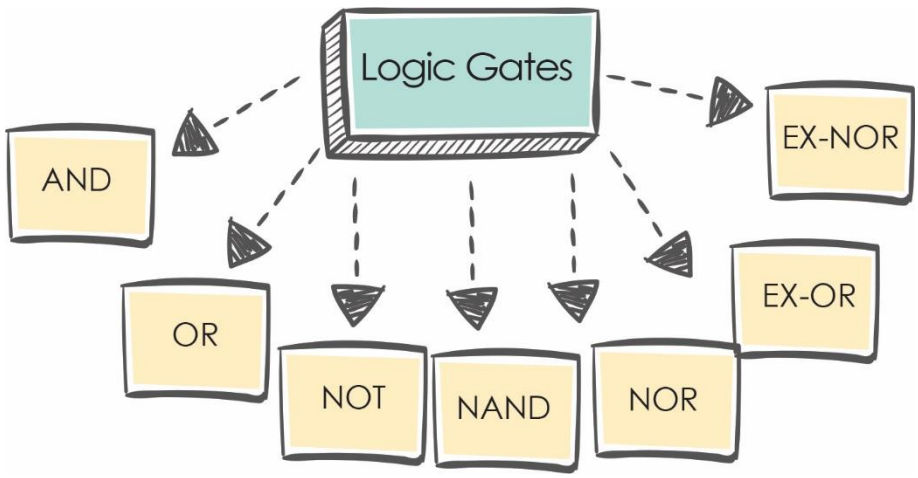
   i. 2573

   ii. 9287

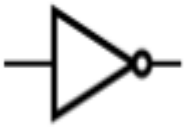
Topic : Logic Gates

Boolean algebra is a mathematical system operating on binary digits or bits specifically 0's and 1's. They perform several mathematical operations. Digital circuits that have one or more inputs, but only one output that can perform logical operations are called logic gates.

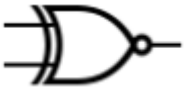




Basic Logic Gates

Type	Symbol	Boolean Algebra	Truth Table
AND		Y = A . B	Input: **A**, **B**; Output: **X** 0 0 0 0 1 0 1 0 0 1 1 1
OR		Y = A + B	Input: **A**, **B**; Output: **X** 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$Y = \overline{A}$	Input: **A**; Output: **X** 0 1 1 0

Combinational Logic Gates

Type	Symbol	Boolean Algebra	Truth Table
NAND		$Y = \overline{A + B}$	Input: **A**, **B**; Output: **X** 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$Y = \overline{A \cdot B}$	Input: **A**, **B**; Output: **X** 0 0 1 0 1 0 1 0 0 1 1 0
EX-OR		$Y = A + B$	Input: **A**, **B**; Output: **X** 0 0 0 0 1 1 1 0 1 1 1 0
EX-NOR		$Y = \overline{A + B}$	Input: **A**, **B**; Output: **X** 0 0 1 0 1 0 1 0 0 1 1 1

Activity 1

Exercises 1 - 6 are short answer or design questions.

1. Differentiate between a gate and a circuit.
2. Notational methods are used for describing the behavior of gates and circuits. Identify the three of notational methods and describe about the notations.
3. How many input signals can a gate receive and output signals can a gate produce?
4. Give the three notation or representations of a NOT gate.

Activity 2

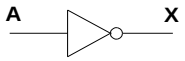
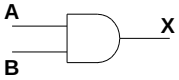
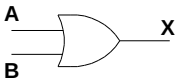
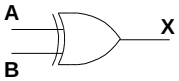
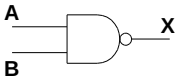
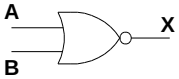
Exercises 1- 10, mark the answers True or False :

1.	Logic diagrams and truth tables are equally powerful in expressing the processing of gates and circuits.	
2.	Boolean expressions are more powerful than logic diagrams in expressing the processing of gates and circuits.	
3.	A NOT gate accepts two inputs.	
4.	The output value of an AND gate when both inputs are 1 is 1.	
5.	The AND and OR gates produce opposite results for the same input	
6.	The output value of an OR gate when both inputs are 1 is 1.	
7.	The output of an OR gate when one input is 0 and one input is 1 is 0.	
8.	The output value of an XOR gate is 0 unless both inputs are 1.	
9.	The Active High gate produces the opposite results of the XOR gate.	
10.	A gate can be designed to accept more than two inputs.	

Activity 3

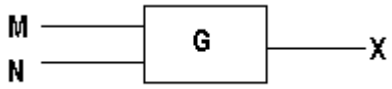
For Exercises 1 - 12, match the gate with the diagram or description of the operation.

A. AND
B. OR
C. NOT
D. NAND
E. NOR
F. XOR

1.	Inverts its input.	
2.	Produces a 1 only if all its inputs are 1 and a 0 otherwise.	
3.	Produces a 0 only if all its inputs are 0 and a 1 otherwise.	
4.	Produces a 0 only of its inputs are the same and a 1 otherwise.	
5.	Produces a 0 of all its inputs are all 1 and a 1 otherwise.	
6.	Produces a 1 if all its inputs are 0 and a 0 otherwise.	
7.		
8.		
9.		
10.		
11.		
12.		

Activity 4

1. The diagram shows a logic gate G whose truth table is a shown in the table below.



M	N	X
0	0	0
1	0	0
0	1	0
1	1	1

Answer : _____

2. A NOR gate with input signals M = 01010101 and N = 011010100 . What is the output signal of the logic gate.

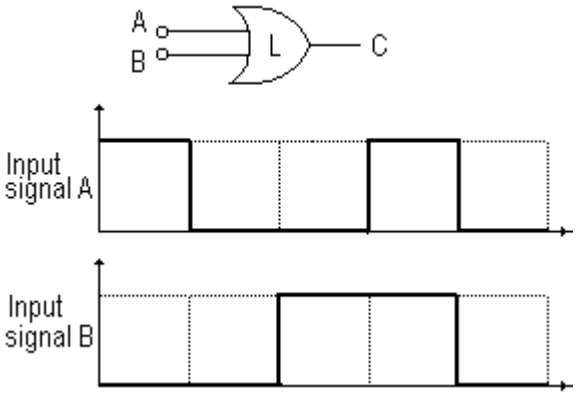
Answer : _____

3. The figure shows a logic gate with inputs P and Q.

If the input P = 0011010 and the input Q = 1100011 , what is the output X?

Answer : _____

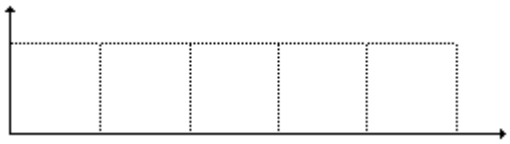
4. The diagram shows a logic gate ,L with input signals A and B.



(i) Name the logic gate L.

…….

(ii) Draw the output signal C in the graph below.

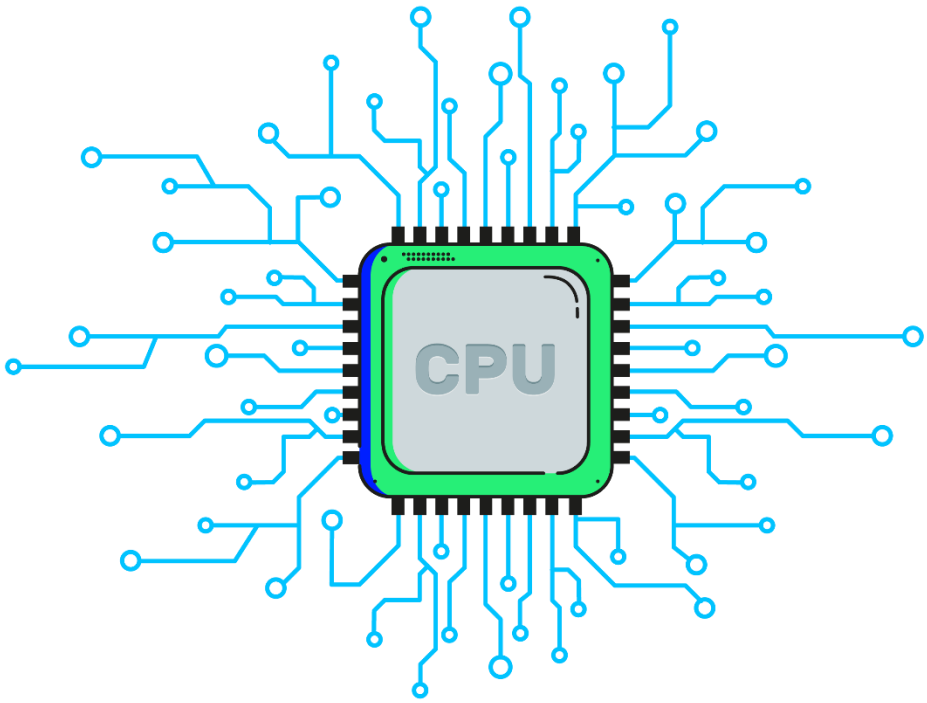


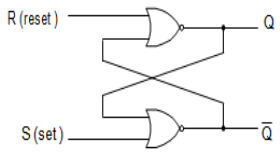
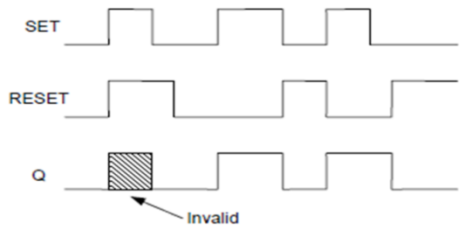
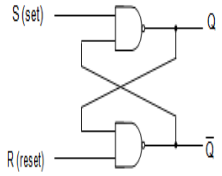
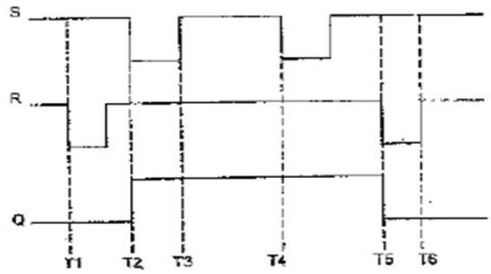
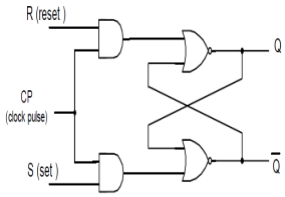
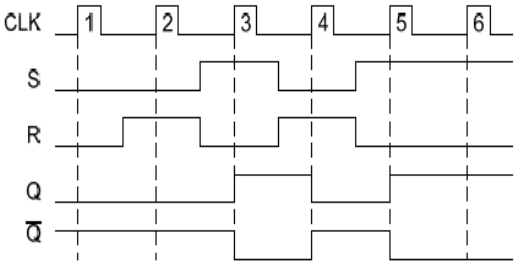
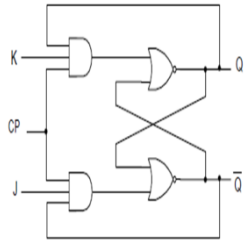
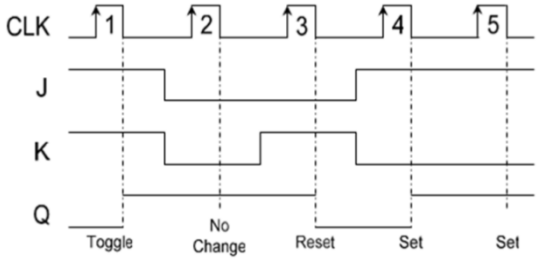
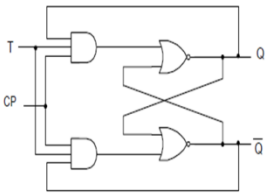
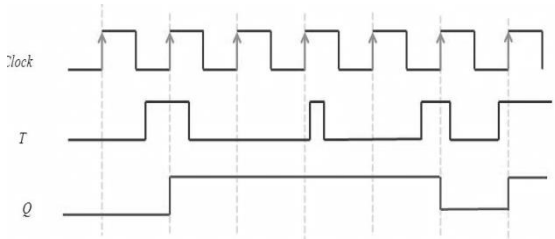
Topic : Flip Flop

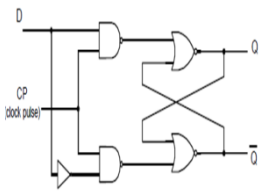
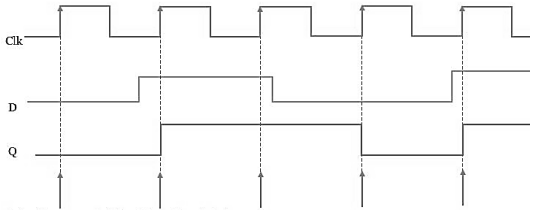
Sequential Logic Circuit

Sequential logic circuit is a memory property circuit and have output that depend on the previous output(s) and current inputs. In order to provide the previous input or output a memory element is required to be used. Thus a sequential circuit needs memory element. Also required clock input.

In general, a sequential circuit is synchronised by the clock signal (pulse) – synchronised circuit. The basic block diagram for a sequential circuit is memory device called *flip-flop* that consist of 2 stable operational states (outputs) Q and $\bar{Q}$ **.** Flip-flop is a circuit that has two stable states and can be used to store state information**.**



<table>
<tr><th>TYPE</th><th>LOGIC GATES</th><th>TRUTH TABLE</th><th>TIMING DIAGRAM</th></tr>
<tr>
<td>NOR GATES SR FLIP FLOP (active HIGH)</td>
<td> 
</td>
<td>
<table>
<tr><th>S</th><th>R</th><th>Operation</th></tr>
<tr><td>0</td><td>0</td><td>No Change</td></tr>
<tr><td>0</td><td>1</td><td>Reset</td></tr>
<tr><td>1</td><td>0</td><td>Set</td></tr>
<tr><td>1</td><td>1</td><td>Invalid</td></tr>
</table>
</td>
<td> 
</td>
</tr>
<tr>
<td>NAND GATES SR FLIP FLOP (active LOW)</td>
<td> 
</td>
<td>
<table>
<tr><th>S</th><th>R</th><th>Operation</th></tr>
<tr><td>0</td><td>0</td><td>Invalid</td></tr>
<tr><td>0</td><td>1</td><td>Reset</td></tr>
<tr><td>1</td><td>0</td><td>Set</td></tr>
<tr><td>1</td><td>1</td><td>No Change</td></tr>
</table>
</td>
<td> 
</td>
</tr>
<tr>
<td>Clocked SR</td>
<td> 
</td>
<td>
<table>
<tr><th colspan="2">Input</th><th>CLK</th><th>Output</th></tr>
<tr><th>S</th><th>R</th><th></th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Not Changing</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>invalid</td></tr>
</table>
</td>
<td> 
</td>
</tr>
<tr>
<td>JK FLIP FLOP</td>
<td> 
</td>
<td>
<table>
<tr><th colspan="2">Input</th><th>Clock</th><th>Output</th></tr>
<tr><th>J</th><th>K</th><th></th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>No Changing</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Toggle</td></tr>
</table>
</td>
<td> 
</td>
</tr>
<tr>
<td>T FLIP FLOP</td>
<td> 
</td>
<td>
<table>
<tr><th>Clock</th><th>T</th><th>Q_{x+1}</th></tr>
<tr><td>1</td><td>0</td><td>Q</td></tr>
<tr><td>1</td><td>1</td><td>$\overline{Q}$</td></tr>
</table>
</td>
<td> 
</td>
</tr>
</table>

<table>
<tr>
<td>D FLIP FLOP</td>
<td> 
</td>
<td>
<table>
<tr><th colspan="2">Input</th><th>Clock</th><th>Output</th></tr>
<tr><th>S</th><th>R</th><th></th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Not Changing</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>invalid</td></tr>
</table>
</td>
<td> 
</td>
</tr>
</table>

Activity 1

1. Draw the logic symbols and develop truth tables of each given flip flop below.
2.
   a) SR (NOR GATE)

   b) Clocked SR

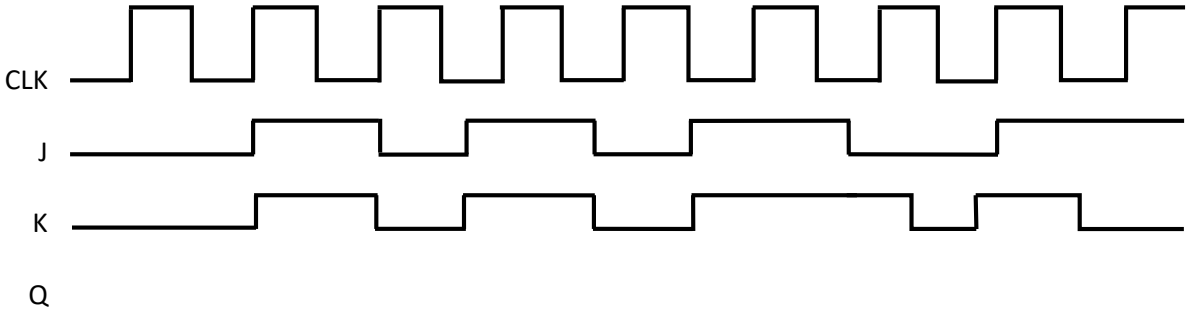
   c) JK

d) T

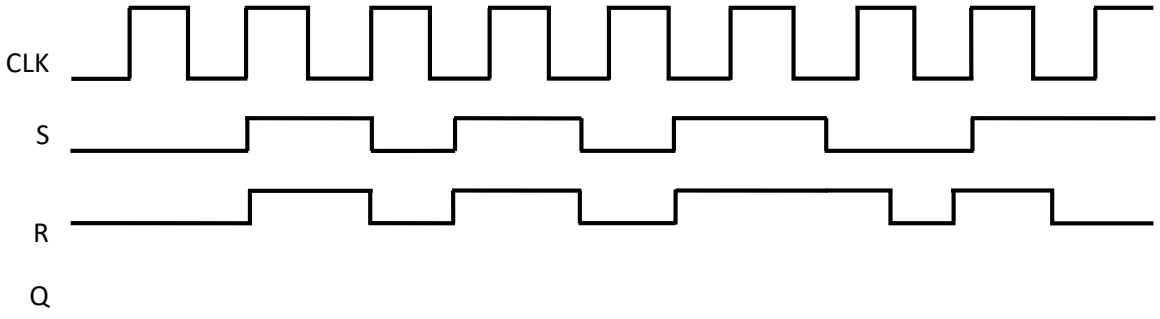
e) D

1. Draw the timing diagram of JK, Clocked SR, T and D flip-flop

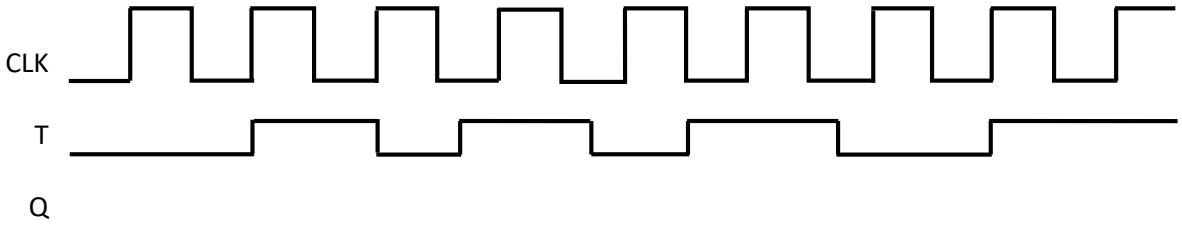
a) JK



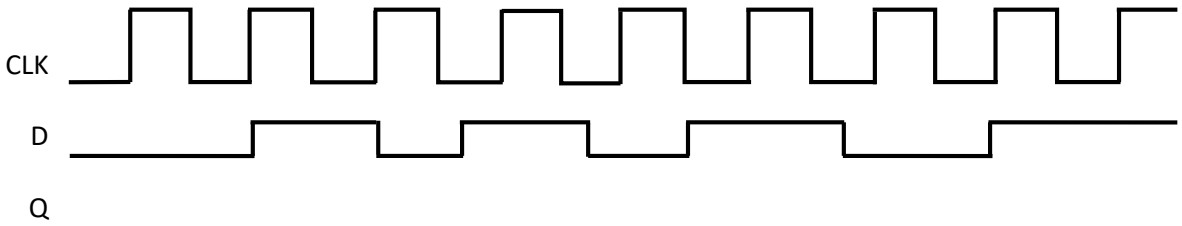
b) SR

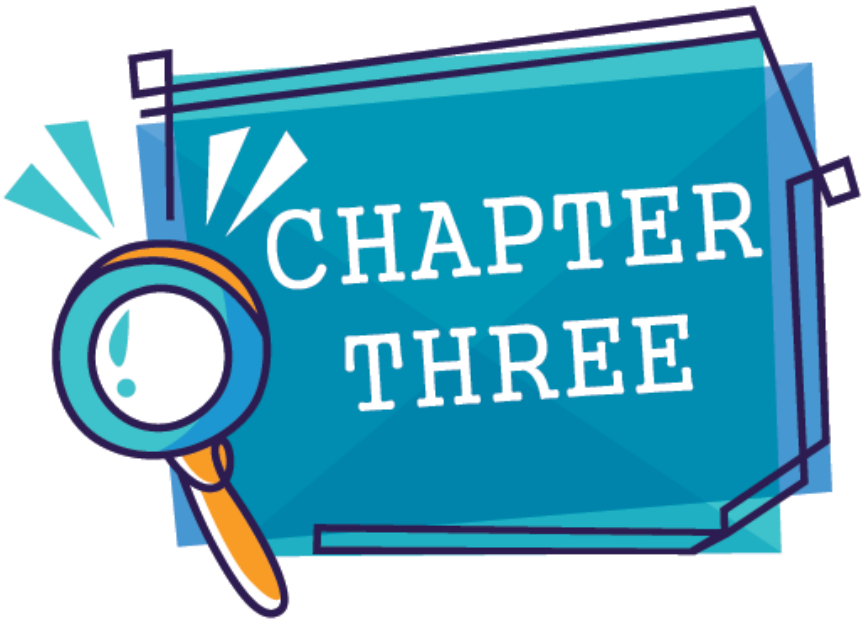


c) T



d) D

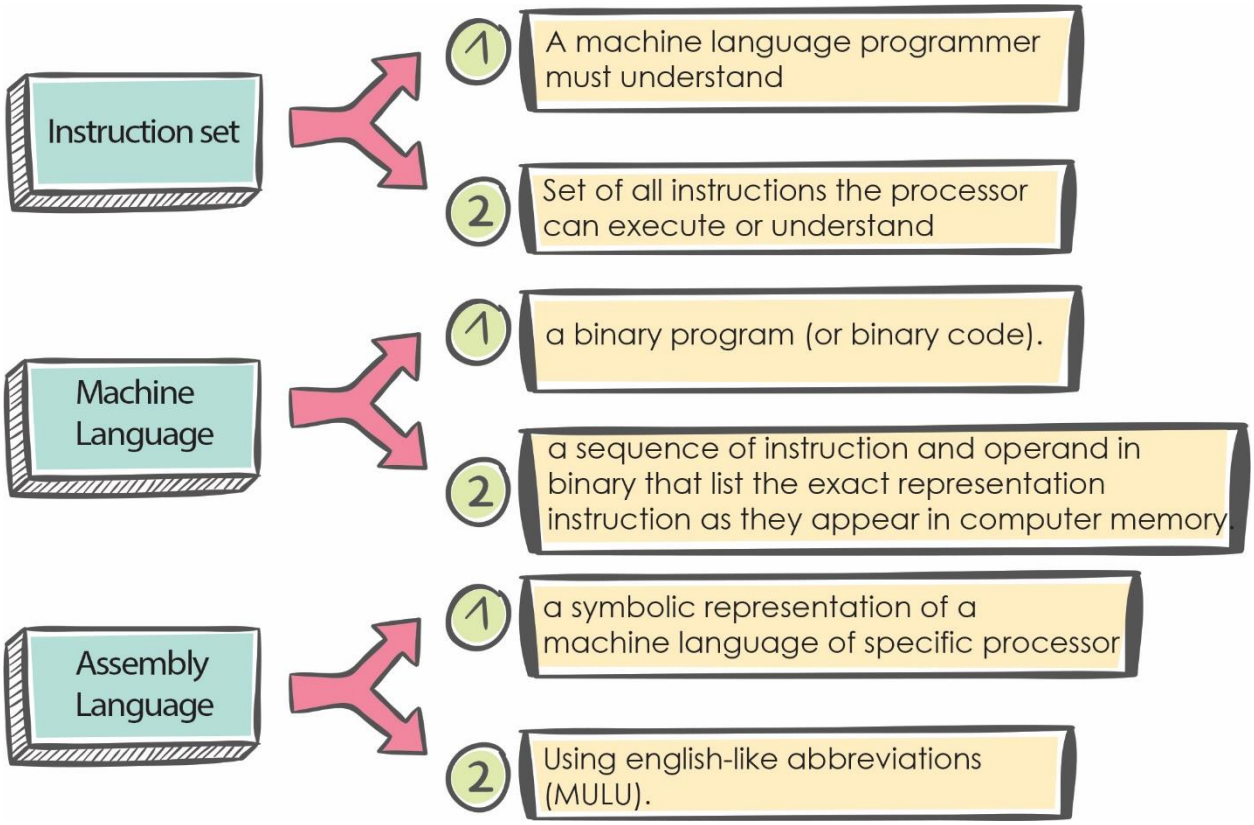


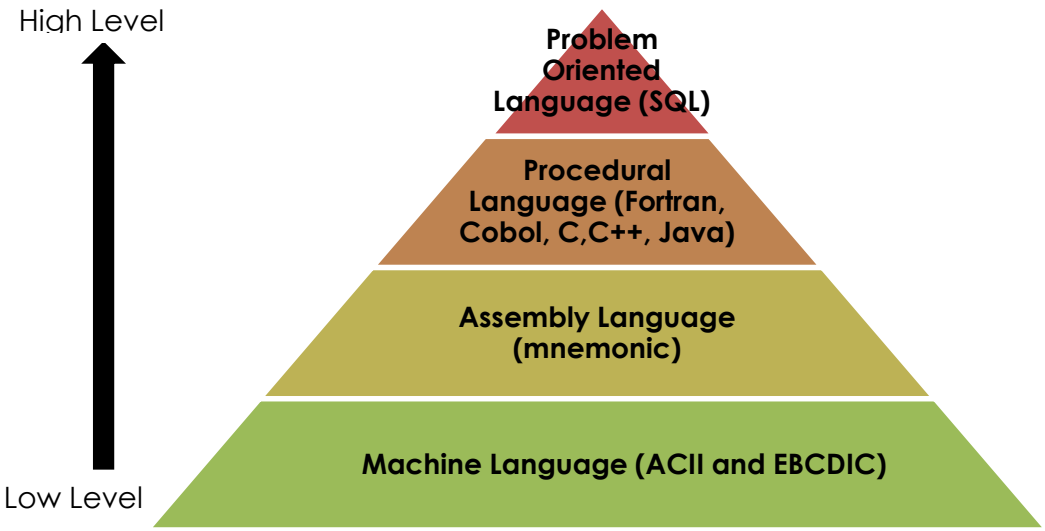


This chapter describes briefly about assembly language

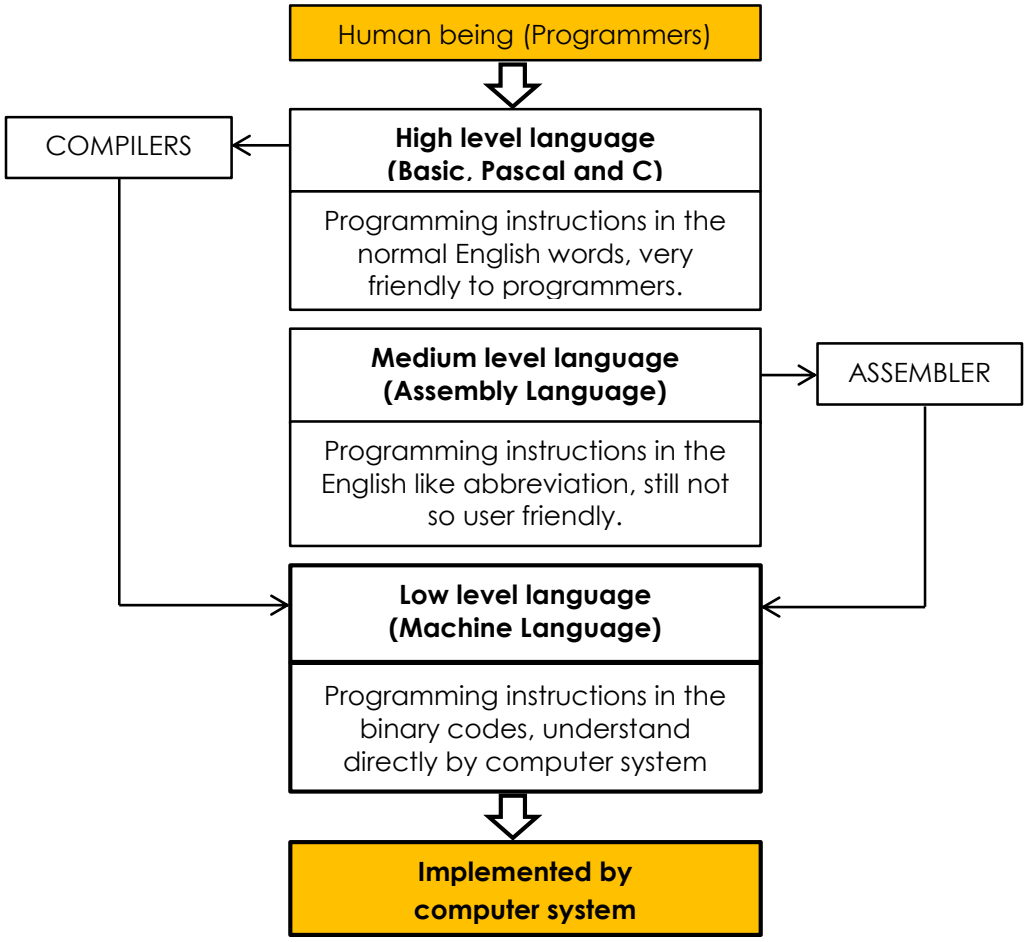
Topic : Assembly Language

Instruction Set, Machine And Assembly Language

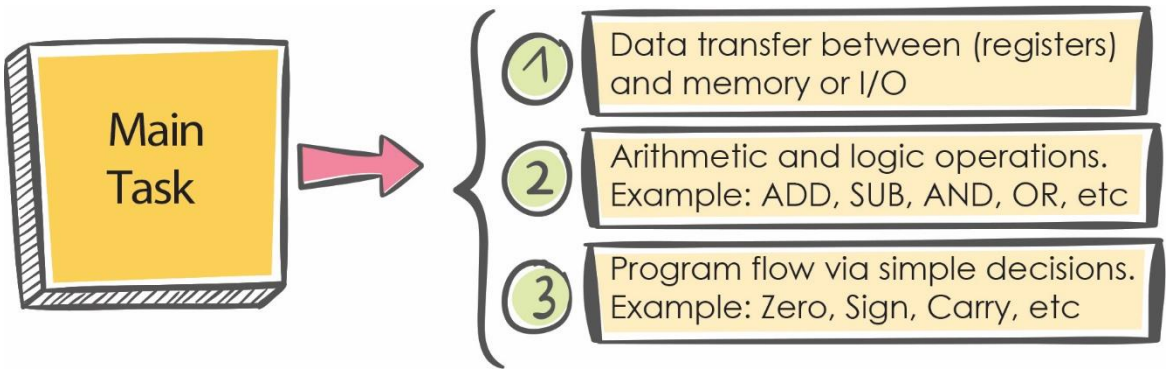


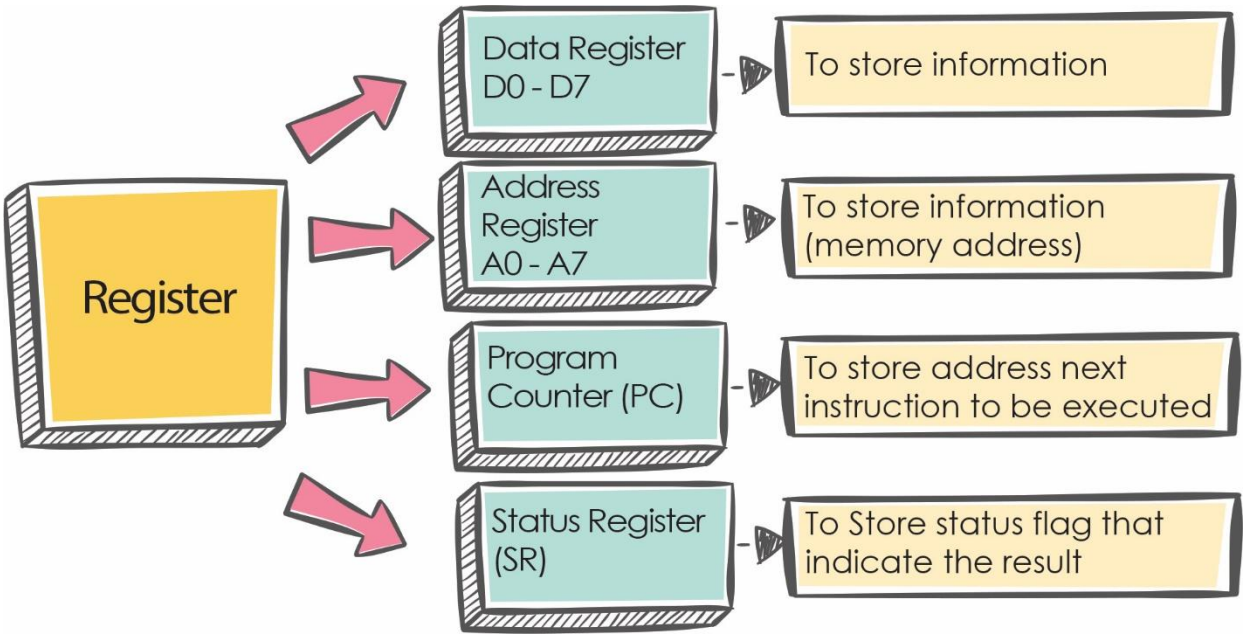


Hierarchy of Programming Language



Basic Information of Microprocessor Motorola 68000

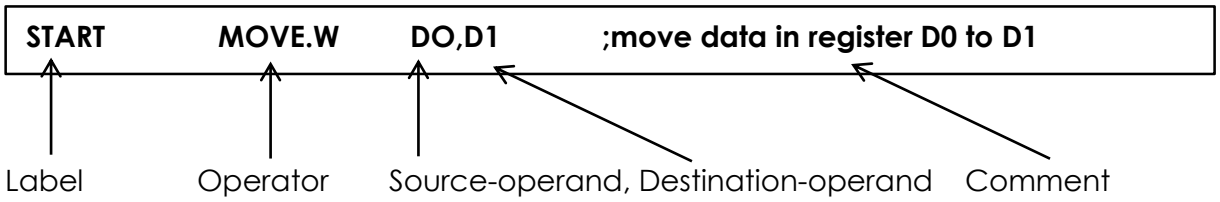




Instruction and Data Format

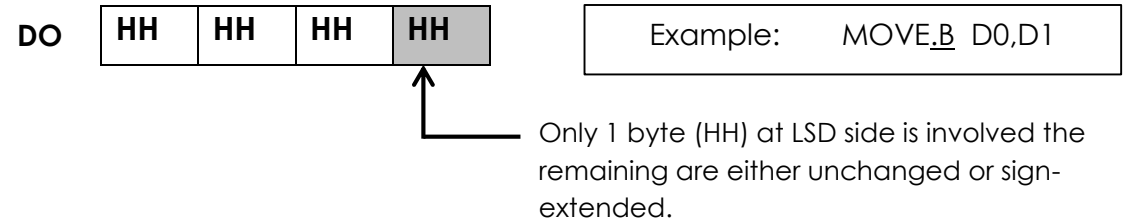
Instruction Format

Example of instruction statement :

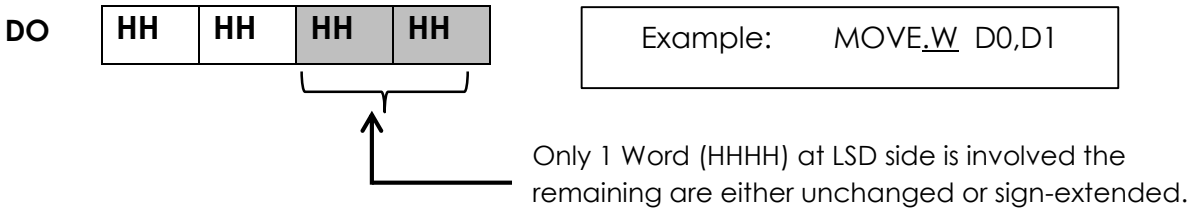


DATA FORMAT

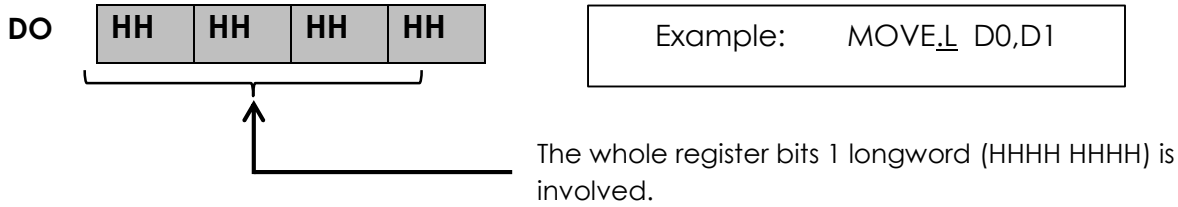
Byte .B



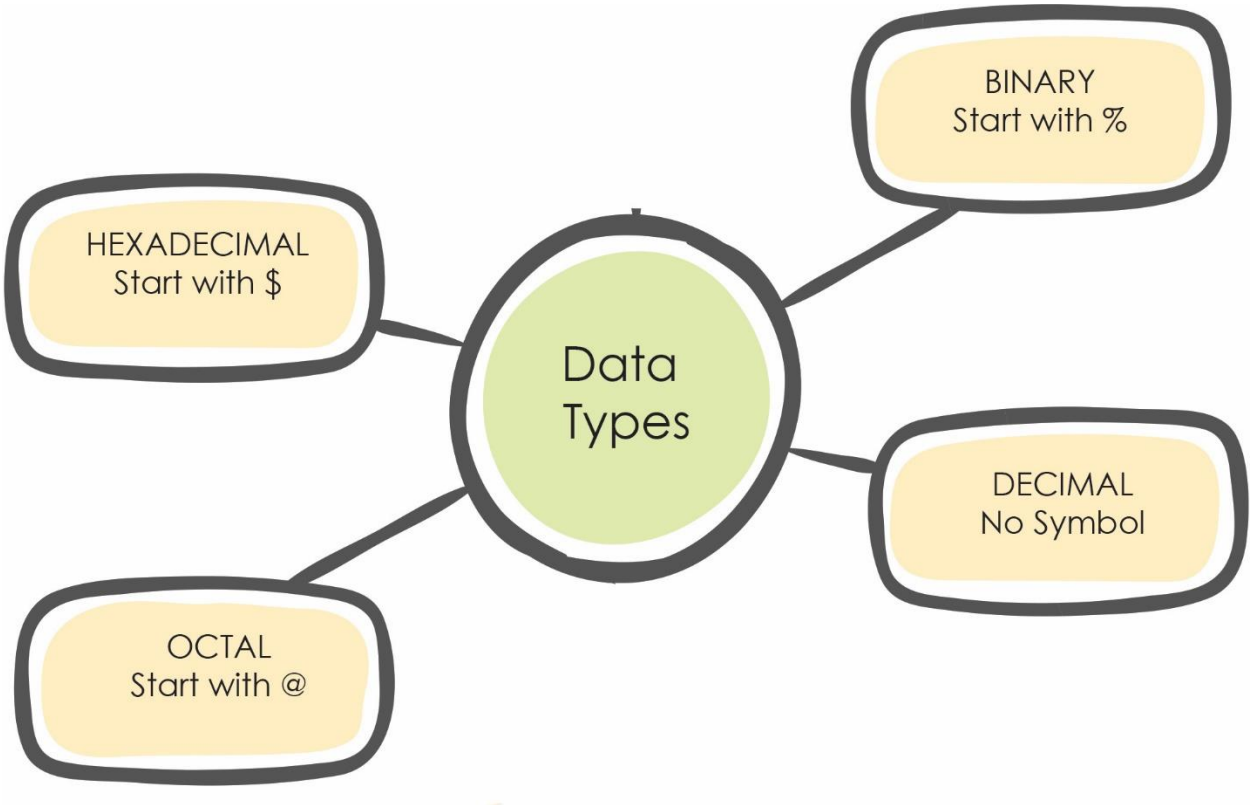
Word .W



Long.L



Data Types



Types of Addressing Modes

Implicit / Implied	RTS
Immediate	MOVE.B #$40,D0 MOVE.W #40,D5 MOVE.L #$30, D7
Absolute	MOVE.B $7000,D3 MOVE.L D4,$1234
Data Register Direct	MOVE.L D0,D7
Address Register Direct	MOVE.L A3,A1 MOVE.L A4,D5
Address Register Indirect	MOVE.L D2,(A0) MOVE.W (A3),D7
Address Register Indirect with Predecrement	MOVE.W –(A6),D0
Address Register Indirect with Postincrement	MOVE.W (A6)+,D0

Example of Instruction in Various Types of Addressing Modes

DATA TRANSFER

Example 1 :

MOVE.W #$72,D1
Before : D1 = $00200500
After : D1 = $00200072

Example 2 :

MOVE.B D0,D1
Before : D1 = $00200500 , D0 = $00002222
After : D1 = $00200522 , D0 = $00002222

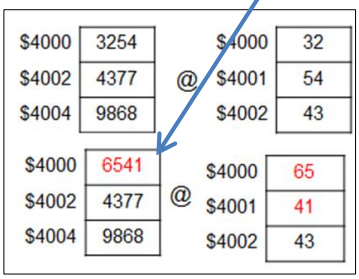
Example 3:

MOVE.B $3000,D1
Before : D1 = $00200500
After : D1 = $00200532

$3000	32
$3001	43
$3002	98

Example 4:

MOVE.W D6,$4000
Before: D6 = $AB206541
After : D6 = $AB206541



ARITHMETIC AND LOGIC OPERATION

Example 1 :

ADD.B D0, D1
Before : D0 = $00002222 , D1 = $00004444
After : D0 = $00002222 , D1 = $00204466

Example 2 :

SUB.W #$80,D3
Before : D3 = $AB206541
After : D3 = $AB2064C1

Example 3 :

MULU #2,D2
Before : D2 = $AB20FFFF
After : D2 = $0001FFFE

Example 4 :

AND.B #$3E,D1
Before : D1 = $12345674
After : D1 = $12345634

```
74   0111 0100
3E   0011 1110&&
     0011 0100
       3    4
```

Example 5 :

NOT.B D1
Before : D1 = $12345655
After : D1 = $123456AA

```
55 01010101
   1010 1010 !
     A    A
```

Assembly Program

FORMAT OF WRITING ASSEMBLY PROGRAM

ORG $1000 - PC Loaded With $1000,
Start executing from here

PROGRAM
INSTRUCTION

END $1000 - Ending of the program

Example 1 :

A program that add 25 and 34.

```
ORG $1000
        MOVE.B #25, D0
        MOVE.B #34, D1
        ADD.B D0,D1
END $1000
```

Example 2 :

A program that solve the expression
! (4000_8 + 10111010_2 / $ACEF_{16}$)

```
ORG $1000
        MOVE.L #@4000, D0
        MOVE.L #%10111010, D1
        MOVE.L #$ACEF,D2
        DIVU.L  D1,D2
        ADD.L D0,D2
        NOT.L D2
END $1000
```

Activity 1

1. Complete the table below:

Bits of operation	Data size	Postfix	Sample instruction	Underline the affected Hex Digit
32	Longword			XXXXXXXX
16			MOVE.W	XXXX XXXX
8		.B		XXXXX XXXX

2. State the value of D1 and D2 after execution for each line

1.	MOVE.B #7, D1	D1=?	
2.	ADD.B #6, D1	D1=?	
3.	MOVE.W #352, D2	D2=?	
4.	ADD.W D1,D2	D1=?	D2=?

Activity 2

1. Given the value D1=0000CAFE and D2 = FFFF1222
   i. Calculate the value OR.B D2,D1

   ii. Calculate the value of NOT.W D2

2. Given the value D1 = ABBBBB12 AND D2=ACCC1251
   i. Calculate the value AND.B D2,D1

   ii. Calculate the value MOVE.W D1,D2

3. Given the value, D1 = 0000 FFFE and D2 = ABCD 1234.
   a) Calculate the value of OR.B D2, D1

   b) Calculate the value of ADD.W D2,D1

Activity 3

1. Identify the type of addressing mode in the following instruction.
   a. MOVE.B #8,D3

   b. MOVE.W D3,(A1)

   c. MOVE.W $1900,D2

   d. MOVE.L D1, D0

2. Write comment in the below table.

SYNTAX	COMMENT
ORG $6000	
MOVE.L #$FFFF 1234, D0	
MOVE.B (A1), D1	
ADD.W D2, D1	
MULU.W #$5D, DO	
NOT.W D2	
RTS	

Activity 4

D1 = 11223344 D2=AA69B250

1. State The Value Of Register D1 And D2 When The Instruction Below Is Executed:

   i. MOVE.W #$1235,D1

   ii. MOVE.B D2,D1

   iii. MOVE.B #%10101111,D2

   iv. MOVE.W $1000,D1
      a. 1000 88
      b. 1001 55

   v. MOVE.B D2,$1000

   vi. MOVE.W #77,D1

   vii. MOVE.L #$ABCD1111,D2

viii. MOVE.W D1,D2

ix. MOVE.W #@34,D1

x. MOVE.B #$11,D1

2. Calculate the value of the register below after execution :

D1 = $12122222

D2 = $12341515

i. ADD.B D2,D1

ii. SUB.B D1,D2

iii. ADD.W #@25,D2

iv. MULU.B #2,D1

v. MOVE.B D2,D1

vi. ADD.B D1,D2

3. Write an instruction based on statement below :

STATEMENT	INSTRUCTION
Transfer data from register D2 to register D3 in long size	
Transfer 10101011_2 to register D4 in long size	
Transfer data from address 5000 to register D2 in byte size	
Transfer $ABCD_{16}$ to register D1 in word size.	
Sub a data in data register D1 and D2 in word size	
Add a data in address register A1 and D3 in word size	
Divide a data in register D3 to register D5 in long size	
Multiply 45_8 to data in register D1 in word size	
Transfer a data from register D1 to register D3 and add a data in register D3 to register D4 in byte size.	

Activity 5

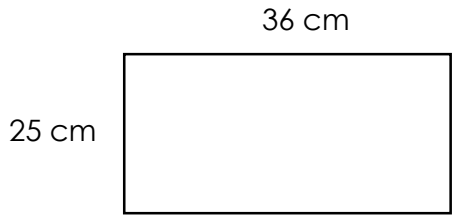
1. Write a programme using Assemble language to solve the operation below:

   a) $(BACA_{16} – 1234_{16})$ + NOT $(ADA_{16}$ AND $87_{16})$

   b) $(100_{10}$ AND $20_{10})$ + $(10_{10}$ / $A_{16})$ + NOT $FFFF_{16}$

c) $! ((76_8 / 1010_2) \mid\mid (ABCD_{16} \&\& 1234_{16}))$

2. Write a program that calculate the area of rectangle



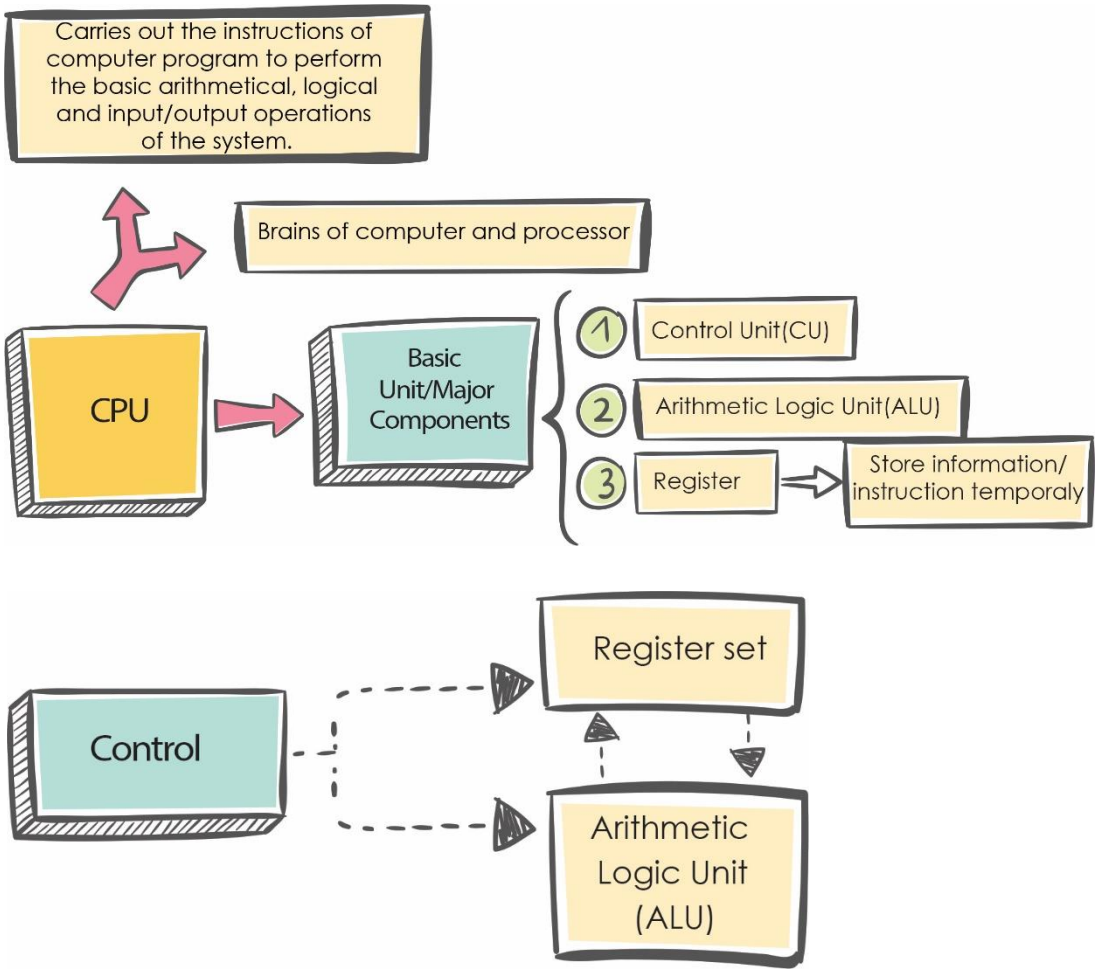
3. Write a program that calculate the average of two numbers. The numbers is 56 and 14.

4. Match the following addressing mode with its instruction sets examples

Data Register Direct	MOVE.B D0, (A0)
Address Register Direct	ADD.W D0, D1
Absolute	MOVE.W A2,A3
Register Indirect	MOVE.B $3000,A4
Immediate	AND.B #$F0, A3

This chapter explains briefly the central processing unit.

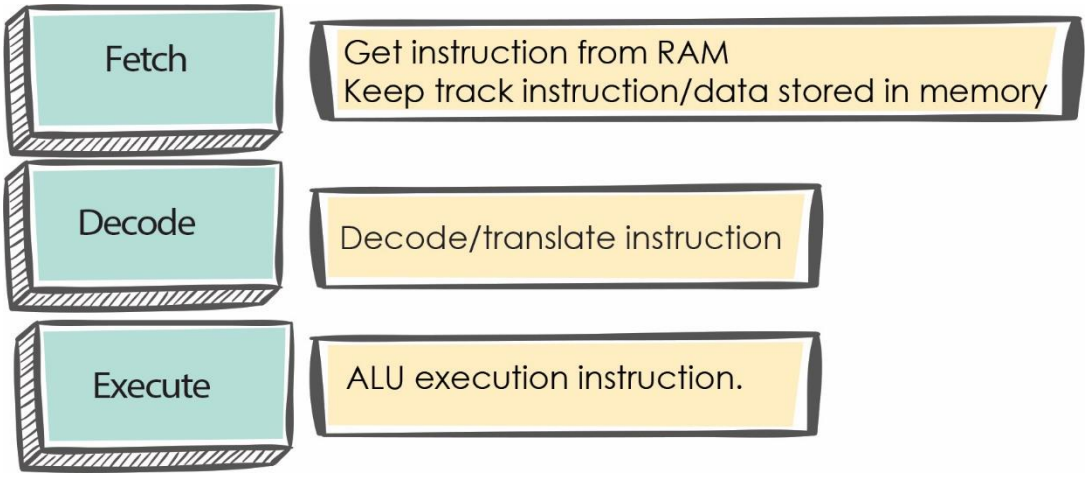
Topic : The Central Processing Unit

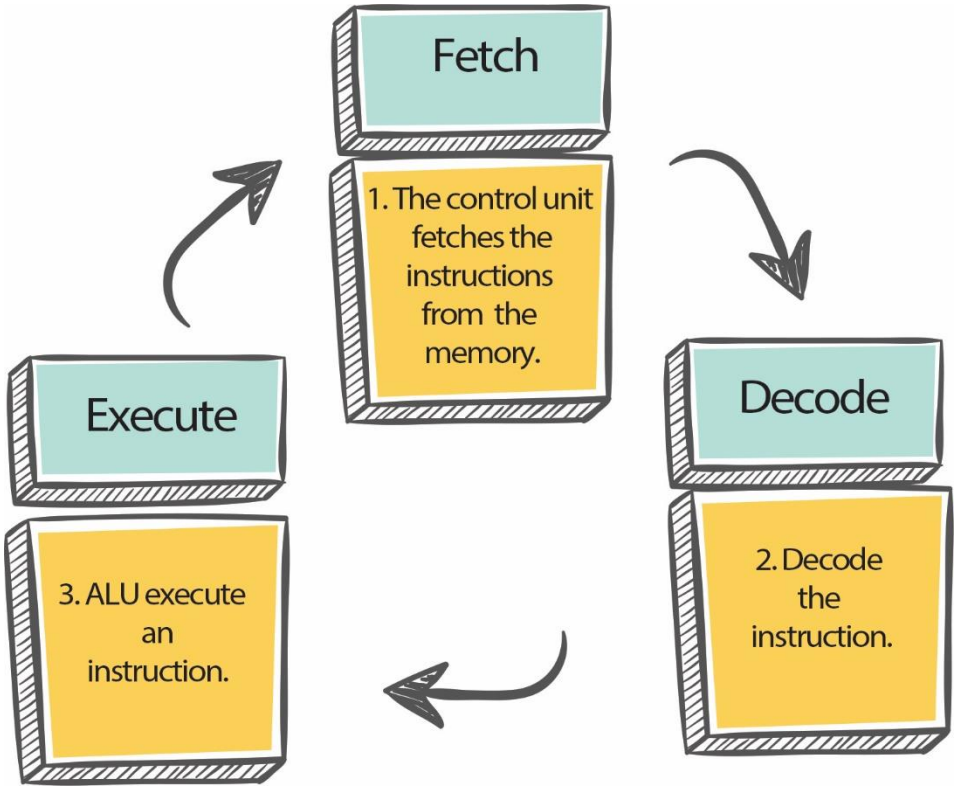


Instruction Cycle

The instruction cycle is the basic operational process of a computer system. It is the process by which a computer retrieves a program instruction from its memory, determines what actions the instruction describes, and then carries out those actions.

THREE (3) major phases of Instruction Cycle

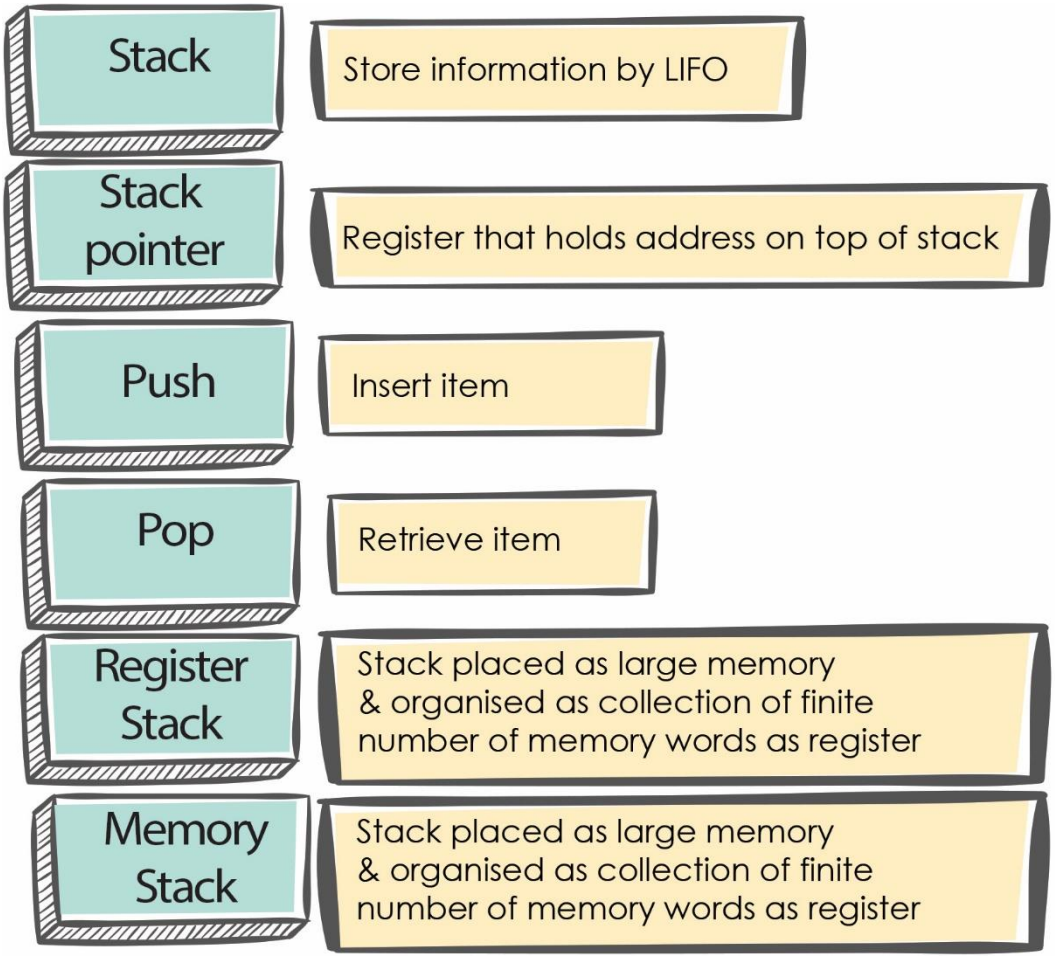


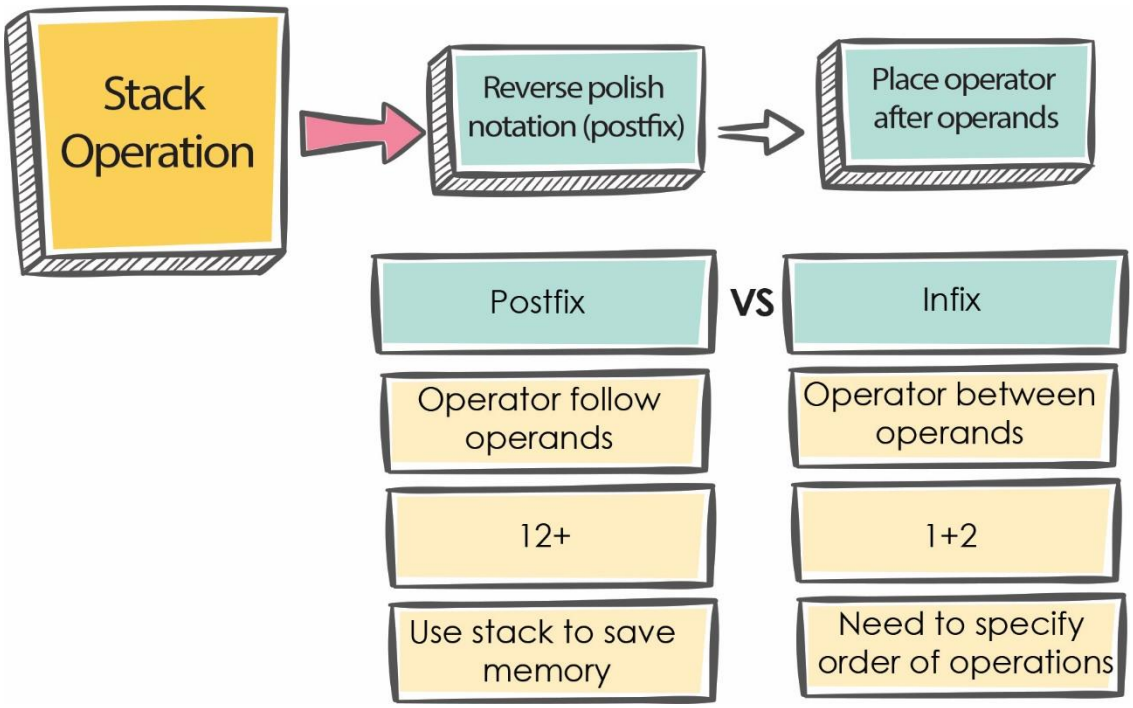


Stack

- A useful feature that is included in the CPU of most computers.
- A storage device that store information in such a manner that the item stored (in) last, is the first item retrieved (out)/LIFO.

Terminologies in stack :





CISC VS RISC

Complex Instruction Set Computer(CISC)	Reduced instruction Set Computer(RISC)
Large number of instructions	Relatively few instructions
Large variety of addressing modes	Relatively few addressing modes
Variable-length instruction formats	Fixed-length, easily decoded instruction format
Instructions that manipulate operands in memory	Instructions that manipulate operands in register

Reverse Polish Notation

The Polish mathematician *Lukasiewicz* show that arithmetic expression can be represented in prefix notation. This representation often referred to as *Polish notation*; place the operator before the operand. The postfix notation, referred to as *reverse Polish notation* (RPN), places the operator after the operand.

The reverse Polish notation is in a form suitable for stack manipulation. The expression,

*A * B + C * D*

Is written in reverse Polish Notation as,

*AB * CD * +*

Proceeding from left to right, we first add A and B, then add D and

E. At this point we are left with:

(A + B)(D + E)C * F + *

Where *(A + B)* and *(D + E)* are each a single number obtained from the sum.The two operand for next * are *C* and *(D + E)*.These two numbers are multiplied and the product added to *F*.The final * cause the multiplications of the two terms.

Reverse Polish notation, combined with a stack arrangement of register, is the most efficient way known for evaluation the arithmetic expression. This procedure employed by some electronic calculators and also in some computer.

Reason why, the combination of stack and reverse polish notation is the most efficient way

1.Stack

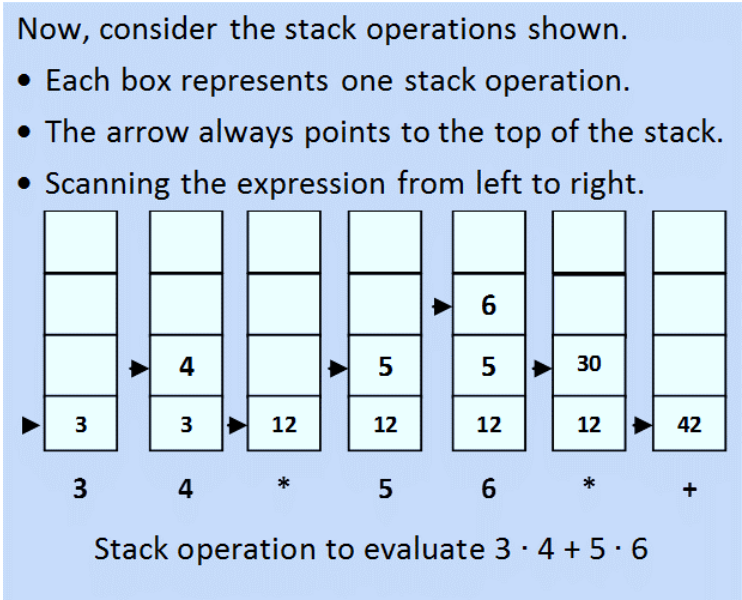
Particularly, useful for handling long, complex problem involving chain calculation.

2.Reverse Polish Notation.

Any arithmetic expression can be expressed in parentheses-free Polish notation. Conversion of arithmetic expression into Polish notation is the most efficient method for translating arithmetic into machine language instruction.

The procedure consist,

1. Converting arithmetic expression into its equivalent reverse Polish notation.
2. Reverse Polish Notation.
3. The operand is pushed into the stack in the order in which they appear.
   a) The two top most operands in the stack are used for the operation.
   b) The stack is popped the result of the operation replace he lower operand.



Reduced Instruction Set Computer (RISC)

In the early 1980s, a number of *computer designer* recommended that computers uses fewer instructions with simple constructs, so they can be executed much faster within the CPU without having to use memory often. This type of computer is classified as **reduced instruction set computer** or **RISC**

The major **characteristic of RISC** :

- Relatively few instruction
- Relatively few addressing modes
- Memory access limited to load and store instructions
- All operation done within the register of the CPU
- Fixed-length, easily decoded instruction format
- Single-cycle instruction execution
- Hardwired rather that micro-programmed control.

Activity 1

1. Draw and describe block diagram for the major component of CPU

2. Name and give a function of each component of A, B and C in Diagram 1 below :

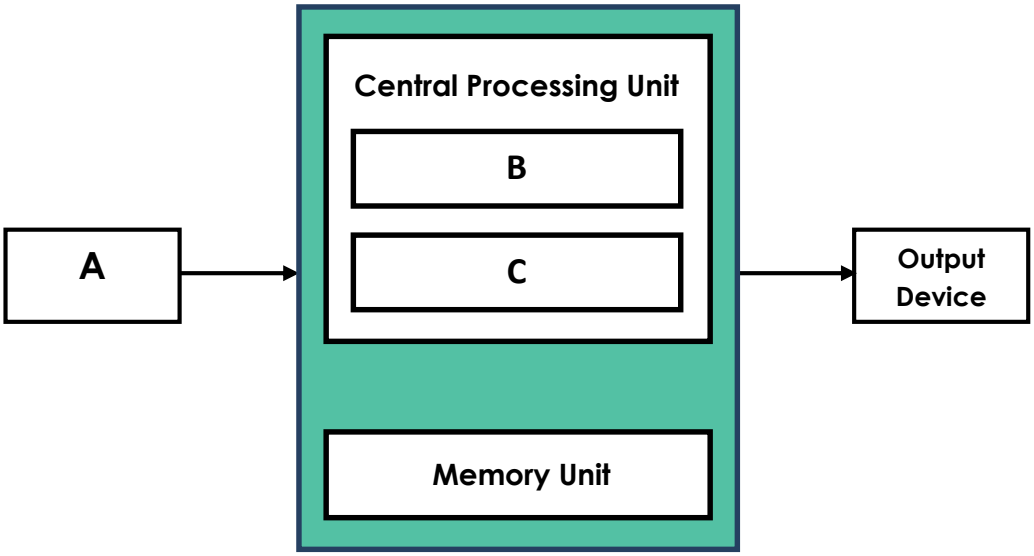


Diagram 1 : Central Processing Unit (CPU)

Item	Name of component	Function
A		
B		
C		

3. Match the following term with the appropriate description below.

Stack Pointer	Stack placed as large memory
Register Stack	Delete Item
Push	Insert Item
Memory Stack	Stand can be standalone unit
Execute	Holds address on top of stack
Pop	Determine the operation to be performed
Decode	Executes the instruction

4. Solve the equation and draw the stack using Reverse Polish Notation

   a) 2+3+4

   b) (2+3)*4

   c) 2*(5+2*3)

   d) (3+4)*(20-(3*4+2))

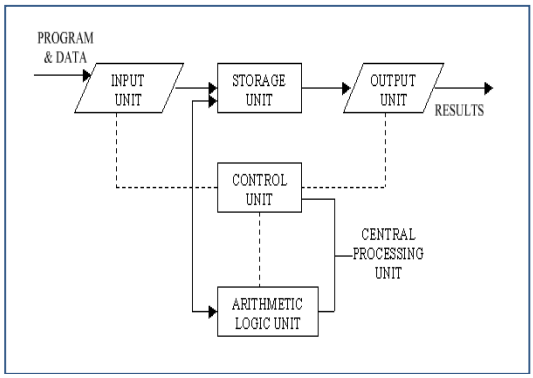
   e) 5*(3+4)-(2*(2+2*(1+2)))


ANSWER
SCHEME

Topic : The Computer System

Activity 1

4.

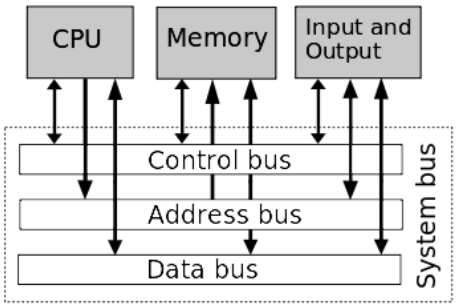


- Input: provide instruction or data to the system. In order for a computer to receive the requests and instructions of the user, some methods of inputting data and information to the computer are required.

- Output: Needs to display the result to the user and to communicate with the user and display information that is being worked on, output device is required

- Storage: Used to store instruction or data. Operation on data requires access for more than one time, so data and instruction have to be stored temporarily
- Control: Control the processing of instructions and the movement of data from one part of the CPU to another.
- ALU: Where arithmetic and Boolean logical calculations are performed

2.



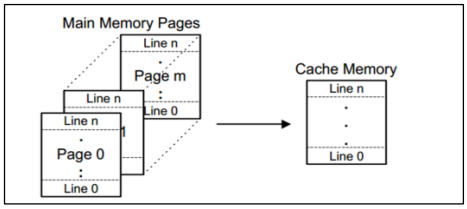
Control Lines:
- Used to control the access to and the use of the data and address lines.
- Typical control lines includes memory write, memory read, I/O write, I/O read, interrupt request and etc.

Address Lines:
- Used to designate the source or destination of the data on the bus
- The width of the address bus determines the maximum possible memory capacity of the system.

Data Lines:
- Provides a path for moving data between system modules
- Consists of 8,16 or 32 separates lines, the number of lines being transferred to as the width of the bus
- Each line carry only 1 bit of the time, the number of lines determines how many bits can transferred at a time

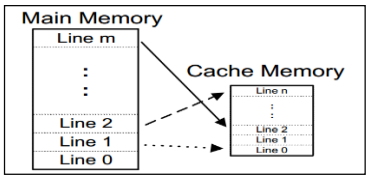
2.

a. Direct



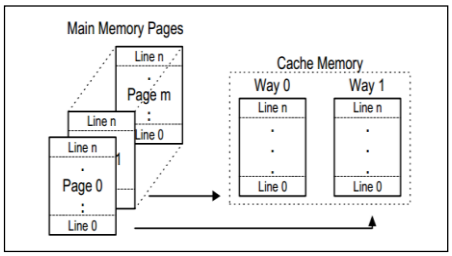
- Simplest technique
- Maps each block of the main memory into one possible cache line

b. Associative



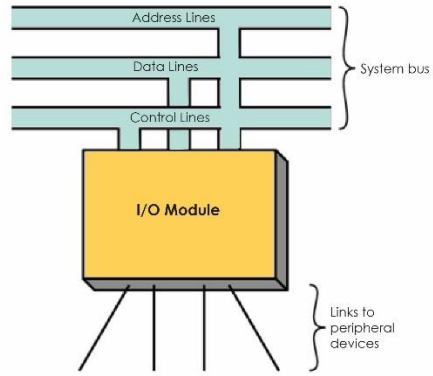
- Any block of main memory can potentially reside in any cache block position. This is much more flexible mapping method.
- Flexible – higher costs (must search all 128 tag patterns to determine if a given block is in cache.
- Existing blocks only need to be ejected if cache is full.

c. Set Associative



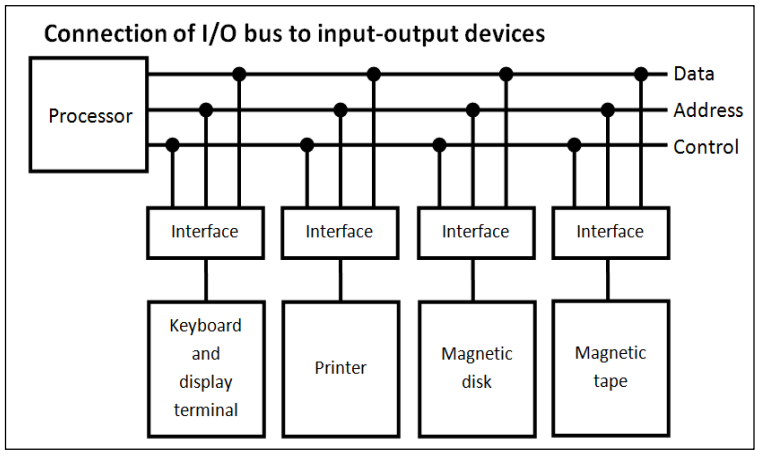
- Blocks of cache are grouped into sets, and the mapping allows a block of main memory to reside in any block of a specific set. From the flexibility point of view, it is in between to the other two methods.

3.



- I/O logic: control circuit through which the CPU and external devices communicate.
- Data registers: intermediate data between the external device and the computer system (e.g. memory, CPU).
- External device interface logic: control circuit through which transfers data and control signals to/from the I/O devices from/to the computer system.

4.



- It defines the typical link between the processor and several peripherals. The I/O Bus consists of data lines, address lines and control lines.
- The I/O bus from the processor is attached to all peripherals interface. To communicate with a particular device, the processor places a device address on address lines.
- Each Interface decodes the address and control received from the I/O bus, interprets them for peripherals and provides signals for the peripheral controller.
- It is also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller. For example, the printer controller controls the paper motion, the print timing.

Activity 2

i. Data Bus
ii. Address Bus
iii. Control Bus
iv. Programming I/O
v. Interrupt Driven I/O
vi. Direct Memory access
vii. Direct Mapping.
viii. Associative mapping
ix. I/O Module .
x. Tags

Topic : Data Representation

Activity 1

1. i) a) 11101112
b) 100002
c) 100012
d) 101110_2
e) 554_8
f) 15756_{16}

ii) a) 1112
b) 1112
c) 1102
d) 110_2
e) 233_8
f) $A4E2_{16}$

2. i)

a. Octal – 16	Hexadecimal - E
b. Octal – 532	Hexadecimal – 15A
c. Octal – 1213	Hexadecimal – 28B
d. Octal – 3446	Hexadecimal – 726
e. Octal – 332	Hexadecimal – DA
f. Octal – 376	Hexadecimal - FE

ii) a) 111112
b) 111010100_2
c) 100012
d) 100101_2
e) 111001.100101_2
f) 10011.001100110_2

iii) a) 11100_2
b) 11110010_2
c) 1000101_2
d) 100011101010_2
e) $101010111100 . 00010010_2$
f) $01000111. 01011011_2$

Activity 2

1.

Number	Sign Magnitude	1's Complement	2's Complement
i. + 17	00010001	00010001	00010001
ii. – 45	10101101	11010010	11010011
iii. – 34	10100010	11011101	11011110

2.

a. 45 - 26

```
 45              - 26
00101101         10011010
                 11100101
                        1 +
                 11100110

00101101
11100110 +
00010011
```

b. -17 + 19

```
 -17             19
10010001         00010011
11101110
        1 +      11101111
11101111         00010011 +
                 00000010
```

c. $7A_{16} + 15_{16}$

```
  7      A            1     5
0111 1010          0001   0101
                    10010101
                    11101010
                           1 +
                    11101011
01111010
11101011 +
01100101
```

4.

i. 2573

```
  2      5      7      3
0010   0101   0111   0011
2573  = 10010101110011BCD
```

ii. 9287

```
  9      2      8      7
1001   0010   1000   0111
9287  = 1101001010000111 BCD
```

Topic : Logic Gates

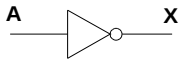
Activity 1

a. A gate accepts one or more input signals and produces an output signal. Each type of gate performs one logical function. A circuit is a combination of gates designed to accomplish a more complex logical function.

b. Boolean expressions use the operations of Boolean algebra to describe the behavior of gates and circuits. Logic diagrams use a graphical representation to describe the behavior of gates and circuits. Truth tables define the behavior of gates and circuits by showing all possible input and output combinations of the gates and circuits.

c. A gate can accept one or more input signals, but can produce only a single output value.

d. A is the input signal and X is the output signal.
Boolean expression: X = A'
Logic Diagram:



Truth Table:

A	X
0	1
1	0

NOT takes a binary input value and inverts it.

Activity 2

1. True	2. False	3. False	4. True	5. False
6. True	7. False	8. False	9. False	10. True

Activity 3

1. C	2. A	3. B	4. F	5. D	6. E
7. C	8. A	9. B	10. F	11. D	12. E

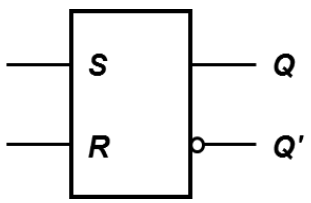
Activity 4

1. AND
2. 100101010_2
3. 1111101_2

Topic : Flip Flop

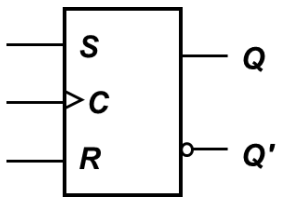
Activity 1

1. a) SR (NOR GATE)



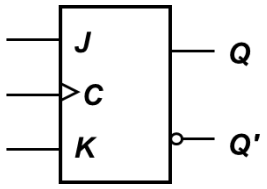
S R	Q
0 0	Q_0
0 1	0
1 0	1
1 1	Q=Q'=0

b) Clocked SR



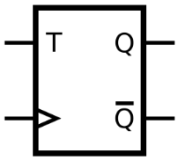
CLK	S R	Q
1	0 0	Q=Q'=1
1	0 1	1
1	1 0	0
1	1 1	Q_0

c) JK



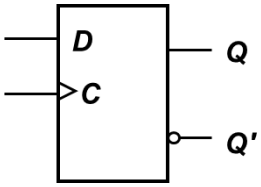
CLK	JK	Q(t+1)
1	0 0	Q(t)
1	0 1	0
1	1 0	1
1	1 1	Q(t)'

d) T



CLK	T	Q(t+1)
1	0	Q(t)
1	1	Q(t)'

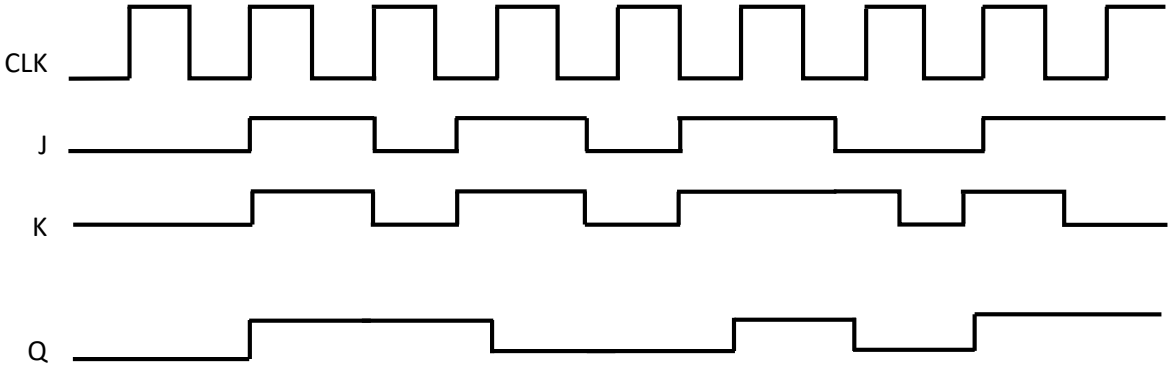
e) D



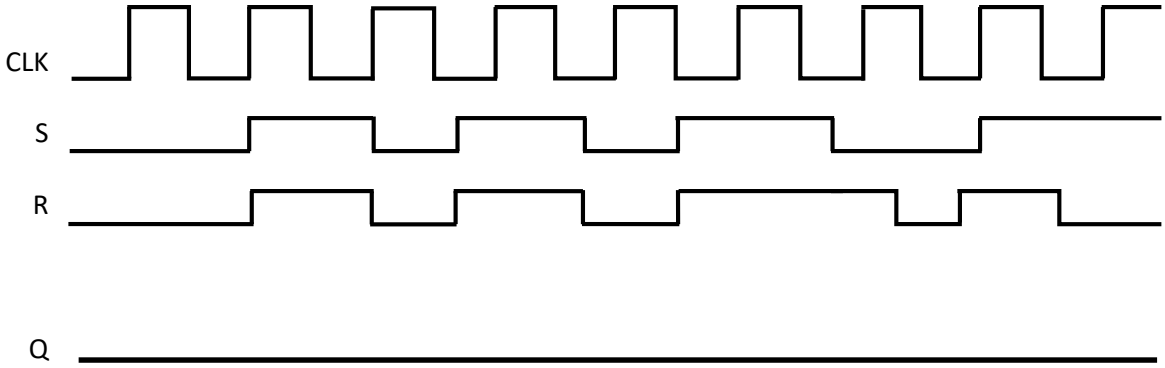
CLK	D	Q(t+1)
0	X	Q(t)
1	0	0
1	1	1

2. Draw the timing diagram of JK, Clocked SR, T and D flip-flop

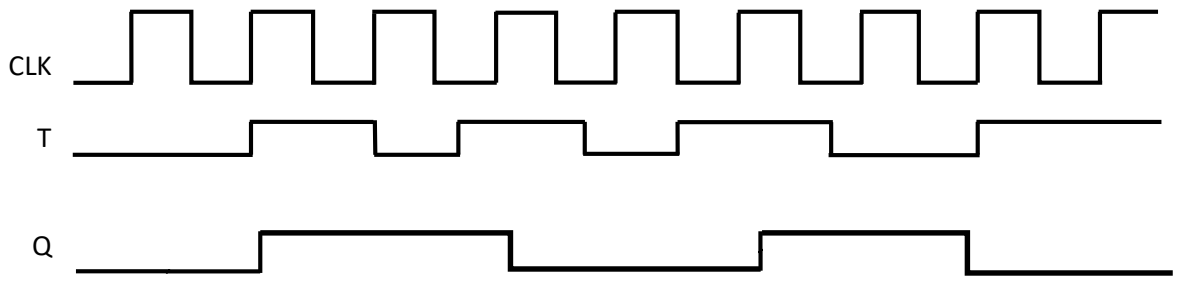
a) JK



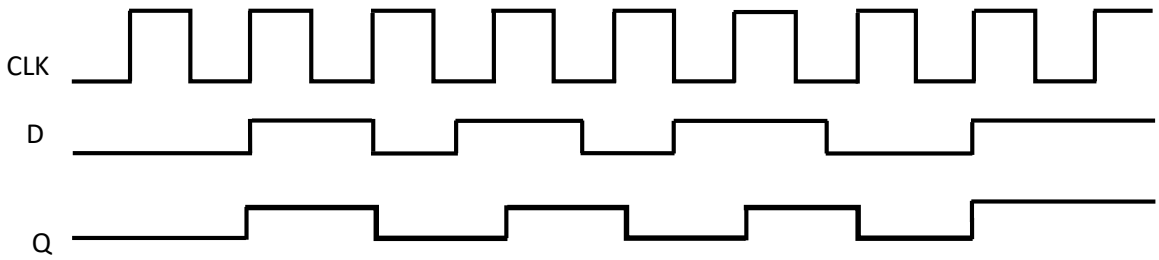
b)



c) T



d) D



Topic : Assembly Language

Activity 1

1.

Bits of operation	Data size	Postfix	Sample instruction	Underline the affected Hex Digit
32	Longword	.L	MOVE.L	XXXXXXXX
16	Word	.W	MOVE.W	XXXX XXXX
8	byte	.B	MOVE.B	XXXXX XXXX

2.

1.	MOVE.B #7, D1	D1=07	
2.	ADD.B #6, D1	D1=0D	
3.	MOVE.W #352, D2	D2=0160	
4.	ADD.W D1,D2	D1=0D	D2=016D

Activity 2

1.

i.

```
 F       E   2      2
1111  1110    0010 0010

11111110
00100010 ||
11111110
   F      E   -> D1=0000CAFE
```

ii.

```
D2 = FFFF1222
      1       2       2        2
   0001 0010 0010  0010
!  1110 1101 1101  1101
     E      D        D        D
D2 = FFFFEDDD
```

2.

i.

```
  5     1          1     2
0101 0001     0001 0010
01010001
00010010 &&
00010000
    1      0
D1 = ABBBBB10
```

ii.

```
D2 = ACCC BB12
```

3.

i.

```
  3     4          F      E
0011 0100     1111 1110
00110100
11111110 ||
11111110
   F    E
D1 = 0000FFFE
```

ii.

```
 1234
 FFFE +
11232

D1=00011232
```

Activity 3

1.

a. Immediate addressing

b. Data indirect addressing

c. Absolute addressing

d. Data Direct Addressing

2.

SYNTAX	COMMENT
ORG $6000	Starting address is 6000
MOVE.L #$FFFF 1234, D0	Move FFFF1234 hexa to register D0 in long size
MOVE.B (A1), D1	Move value in address register A1 indirectly to register D1 in byte size
ADD.W D2, D1	Add value in D2 to D1 in word size
MULU.W #$5D, DO	Multiply 5D hexa to value in D0 in word size
NOT.W D2	Not the value in register D2 in word size
RTS	End of statement

Activity 4

1.
   i. D1 = 11221235
   ii. D1 = 11223350
   iii. D2=AA69B2AF
   iv. D1 = 11228855
   v. 1000 50
   vi. D1 = 1122004D
   vii. D2= ABCD1111
   viii. D2=AA693344
   ix. D1 = 1122001C
   x. D1 = 11223311

2.

STATEMENT	INSTRUCTION
Transfer data from register D2 to register D3 in long size	MOVE.L D2,D3
Transfer 101010112 to register D4 in long size	MOVE.L #%10101011,D4
Transfer data from address 5000 to register D2 in byte size	MOVE.B $5000,D2
Transfer ABCD16 to register D1 in word size.	MOVE.W #$ABCD,D1
Sub a data in data register D1 and D2 in word size	SUB.W D1,D2
Add a data in address register A1 and D3 in word size	ADD.W A1,D3
Divide a data in register D3 to register D5 in long size	DIVU.L D3,D5
Multiply 45_8 to data in register D1 in word size	MULU.W #@45,D1
Transfer a data from register D1 to register D3 and add a data in register D3 to register D4 in byte size.	MOVE.B D1,D3 ADD.B D3,D4

3. D1 = $12122222 D2 = $12341515
   i. D1=12122237
   ii. D2 =1234150D
   iii. D2 =1234152A
   iv. D1 = 12122244
   v. D1=12122215
   vi. D2=12341537

Activity 4

1.
   a)

```
ORG $1000
    MOVE.W #$BACA,D1
    MOVE.W #$1234,D2
    SUB.W D1,D2
    MOVE.W #$ADA,D3
    MOVE.W #$87,D4
    AND.W D3,D4
    NOT.W D4
    ADD.W D2,D4
END $1000
```

b)

```
ORG $1000
    MOVE.W #100,D1
    MOVE.W #20,D2
    AND.W D1,D2
    MOVE.W #10,D3
    MOVE.W #$A,D4
    DIVU.W D3,D4
    MOVE.W #$FFFF,D5
    NOT.W D5
    ADD.W D2,D4
    ADD.W D4,D5
END $1000
```

c)

```
ORG $1000
    MOVE.W #@76,D1
    MOVE.W #%1010,D2
    DIVU.W D1,D2
    MOVE.W #$ABCD,D3
    MOVE.W #$1234,D4
    AND.W D3,D4
    OR.W D2,D4
    NOT.W D4
END $1000
```

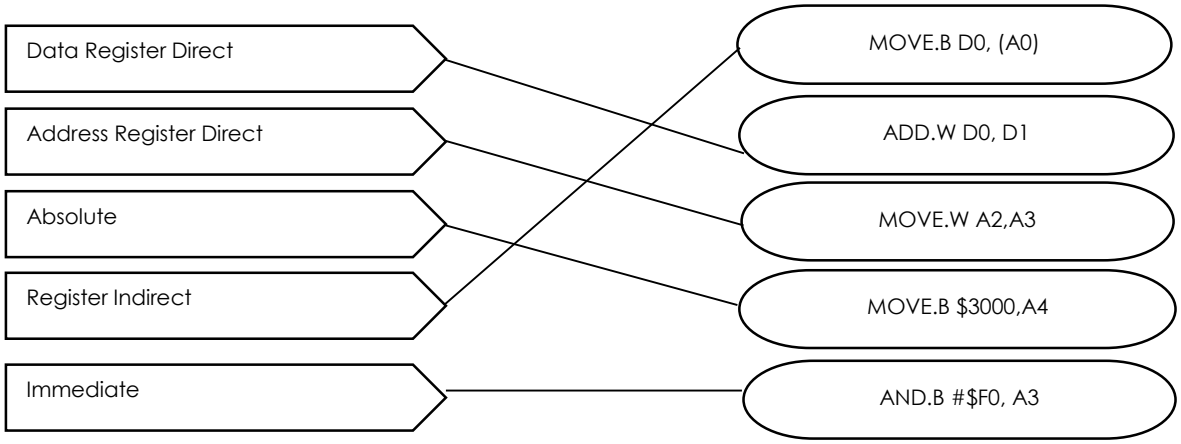
2.

```
ORG $1000
    MOVE.W #36,D1
    MOVE.W #25,D2
    MULU.W D1,D2
END $1000
```

3.

```
ORG $1000
    MOVE.W #56,D1
    MOVE.W #14,D2
    MULU.W D1,D2
    DIVU.W #2,D2
END $1000
```

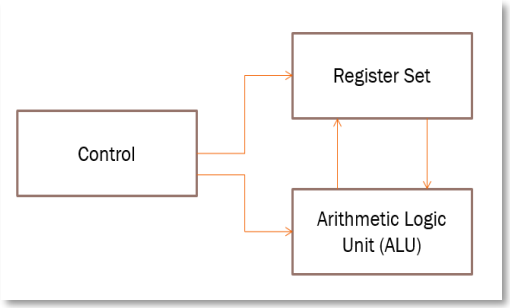
4.



Topic : The Central Processing Unit

Activity 1

1.

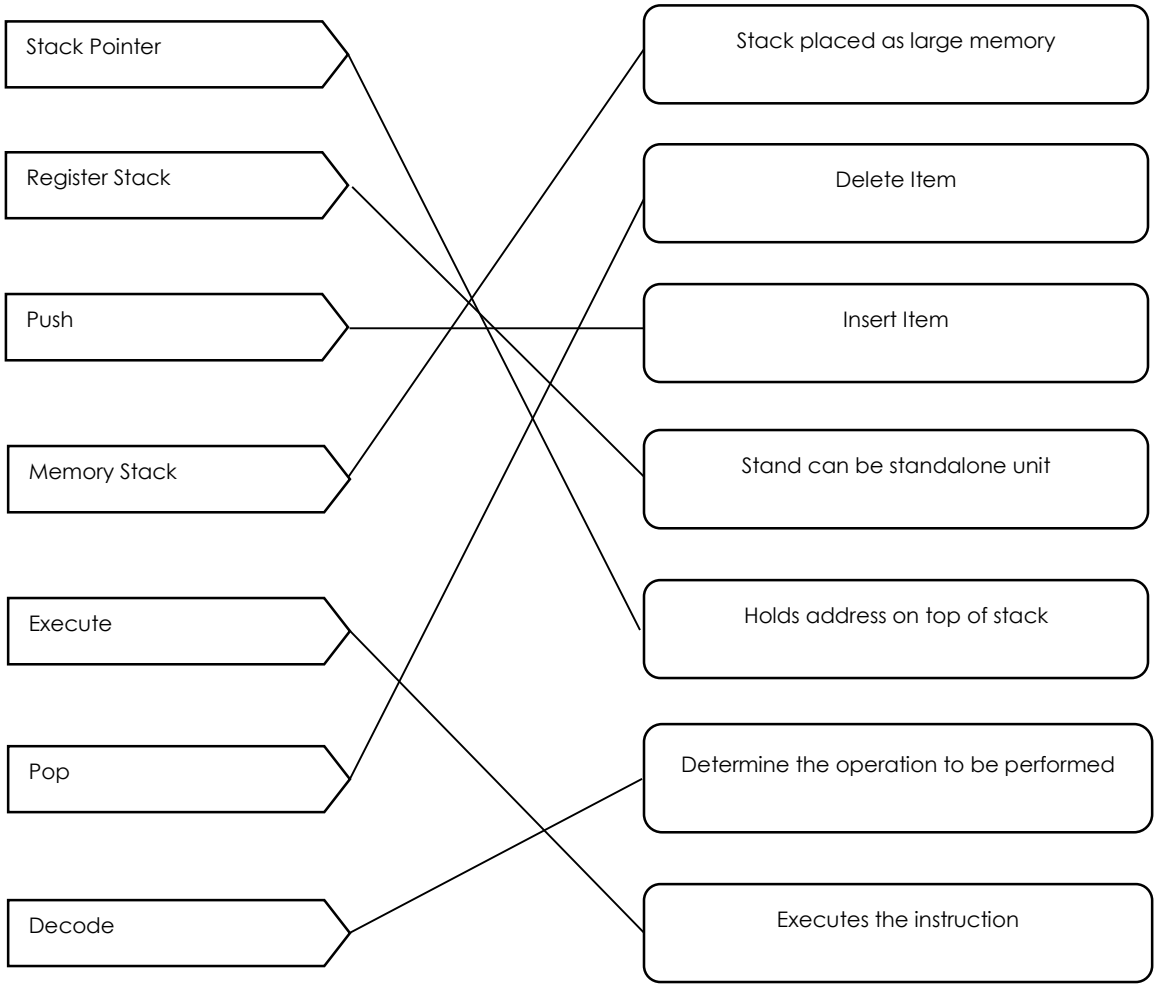


- Control Unit: The control unit executes the instructions, sends control signals to and receive control signals from devices.
- ALU: handles arithmetic calculations and performs logical calculations and makes judgement like "if A > B is true".
- Register: Store data and programs

2.

Item	Name of component	Function
A	INPUT DEVICE	operation recognizes input from keyboard or mouse.
B	REGISTER SET	Store intermediate data used during the execution of the instructions
C	ALU	Perform the calculation, sorting and comparison operation

3.


Stack Pointer
Register Stack
Push
Memory Stack
Execute
Pop
Decode
Stack placed as large memory
Delete Item
Insert Item
Stand can be standalone unit
Holds address on top of stack
Determine the operation to be performed
Executes the instruction

4. Solve the equation and draw the stack using Reverse Polish Notation

a) 2+3+4
= 234++
= 9

		4		
	3	3	7	
2	2	2	2	9

b) (2+3)*4
= 23+4*
= 20

	3		4	
2	2	5	5	20

c) 2*(5+2*3)
= 2523*+*
= 22

			3			
		2	2	6		
	5	5	5	5	11	
2	2	2	2	2	2	22

c) (3+4)*(20-(3*4+2))
= 34+2034*2+-*
= 42

						4		2			
					3	3	12	12	14		
	4		2	20	20	20	20	20	20	6	
3	3	7	7	7	7	7	7	7	7	7	42

e) 5*(3+4)-(2*(2+2*(1+2)))
= 534+*22212+*+*-
=19

								2					
								1	3				
						4	2	2	2	6			
		4				3	2	2	2	2	8		
	3	3	7		2	20	2	2	2	2	2	16	
5	5	5	5	35	35	7	35	35	35	35	35	35	19

Bibliography :

David. A. P. &. John. L. H. (2018). Computer Organization and Design The Hardware / Software Interface (RISC-V Edition). Book Aid International.(ISBN: 0128122757)

John. L. H. & David. A. P. (2017). Computer Architecture A Quantitative Approach Sixth Edition. Katey Birtcher. (ISBN: 0128119055) © N U R S H A M I N I E B I

Ledin. J. (2020). Modern Computer Architecture and Organization: Learn x86, ARM,and RISC-V architectures and the design of smartphones, PCs, and cloud servers 1st Edition, Kindle Edition. Packt Publishing. India. (ISBN:1838984399)

Stallings. W. (2018). Computer Organisation and Architecture Design for Performance (11th Edition). United State: Pearson Education. (ISBN:9780134997193)

Stefano. M. (2020). Architecture Computer. Kindle Edition. Amazon. (ASIN:B08NYX5VF3)

Terbitan

POLITEKNIK MALAYSIA
KUALA TERENGGANU

e ISBN 978-967-2240-35-8
9 7 8 9 6 7 2 2 4 0 3 5 8