Fundamental of COA

Topics

→ Computer
→ Types of computers
→ Representation of basic information in digital system.
→ Basic Computer Model and different units of computer.
→ computer Architecture vs. Computer Organization
→ Basic types of Architecture.
→ Number System
→ Number System Conversion
→ Complements.
→ Subtraction using complements
→ Representation of numbers in binary
→ Binary Codes.
→ Simple Binary Arithmetic
→ Overflow and signed addition

Computer:

- A computer is an electronic device that performs general tasks.
- A computer is an electronic device that takes input from input device, process that input and sends the results to output device.

- Two types of computers
  - → Analog
  - → Digital
- Analog computers work on analog signals. The analog signals are continuous in nature.
- Digital computers work on digital signals. The digital signals are discrete in nature.
- A digital computer is a digital system that performs various computational tasks.

> The word digital implies that the information in the digital computer are represented by the limited number of discrete values

Representation of Basic Information:

- computer is a digital divice which works on two levels of signals.
  - -High level signal (+5V or 12V) ⇒ ON
  - -Low level signal (0V) ⇒ OFF

Numerically

High represented by 1
Low represented by 0

- All the functionality of computer can be captured with 0 and 1 and its theoretical background corresponds to two valued boolean algebra.

- Digital computers use binary number system which has two digits (0 and 1).

- Basically binary number system is used to represent information and manipulation of information in the computer.

- The smallest unit of information that is represented in a computer is known as a bit

List:

Bit	0 or 1
Nibble	4 bits
Byte	8 bits
(KB) Kilobyte	1024 bytes
(MB) Megabyte	1024 KB
(GB) Gigabyte	1024 MB
(TB) Terabyte	1024 GB
(PB) Petabyte	1024 TB
(EB) Exabyte	1024 PB
(ZB) Zettabyte	1024 EB
(YB) Yottabyte	1024 ZB

Digital computers

- can be divided into four categories
  - → Embedded computers
  - → Personal computers
  - → Severs and Enterprise systems
  - → Supercomputers and Grid computers.

Above mentioned categories of computers vary in size cost, computational power and intended use.

Main functions of a computer:

- Data Processing
- Data Storage
- Data Movement
- Control → coordinates the use of the information

- A computer system sometimes can be & subdivided into two functional entities.
  - Hardware
  - Software

- Hardware:
  - → Physical entities of a device.
  - → consists of all electronic components and electro-mechanical devices.

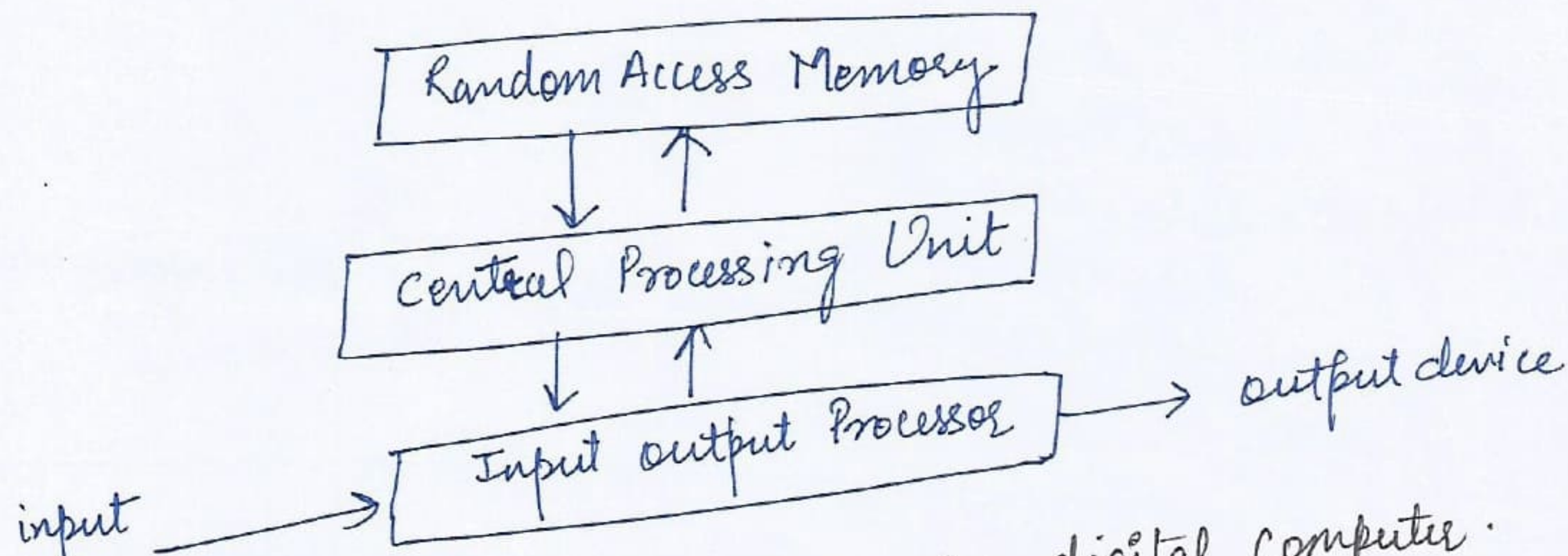
- Software: computer software consists of instructions and data that the computer manipulates to perform various data processing tasks.

Program:

→ A sequence of instructions to perform a specific task.

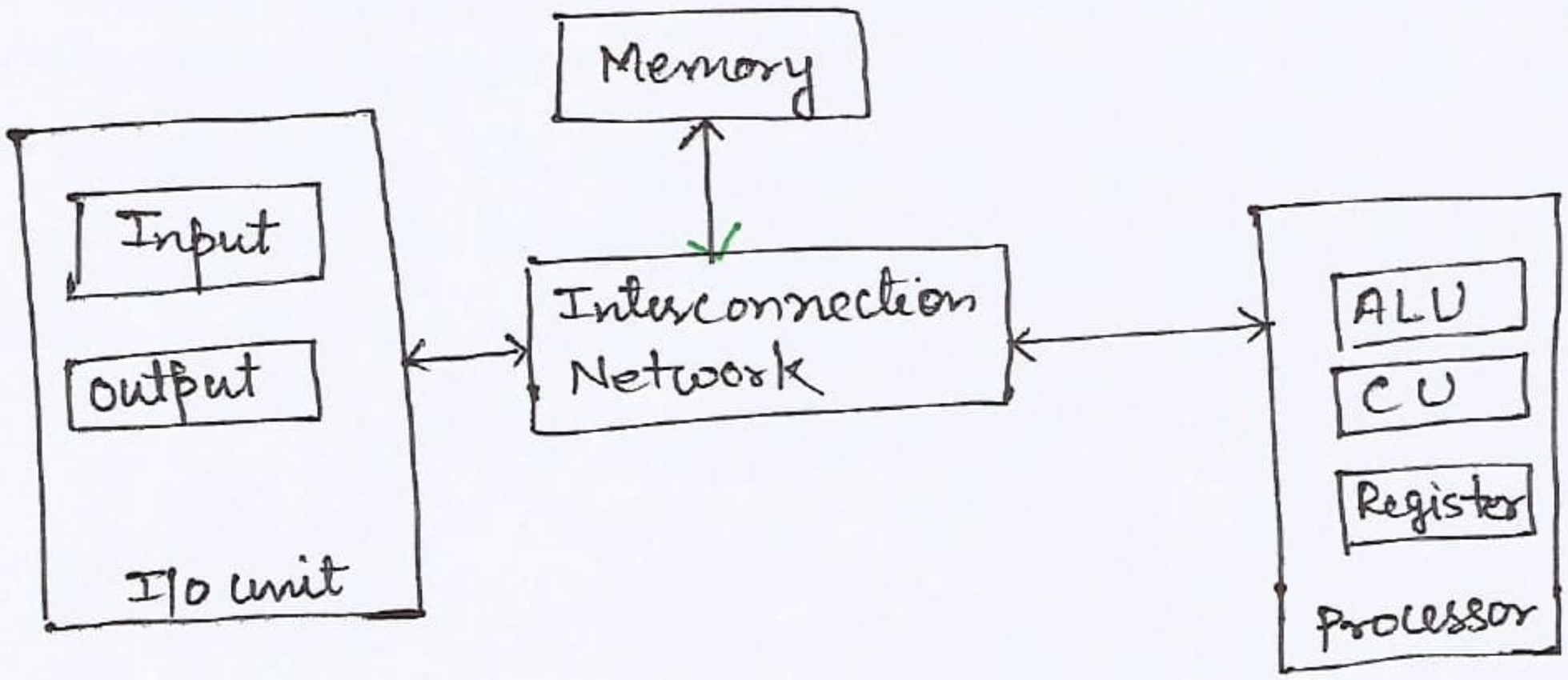
Computer Hardware: computer hardware is usually devided into three major parts

- CPU (Central Processing Unit)
- The main memory
- Input Output Processor



Block diagram of a digital computer.
(Von-Neumann Architecture)

Basic Computer Model and different functional units of computer



Central Processing Unit: (Processor):-

- controls the operation of the computer and performs its data processing functions.
- Processor has components
  - control unit
  - ALU (Arithmetic Logic unit)
  - Registers
  - CPU interconnections.

- control unit controls the operation of CPU.
- ALU performs the data processing functions such as execution of arithmetic and ~~logic~~ & logic operations
- Registers provides internal storage to CPU as temporary storage of data.
- CPU interconnection: ~~comm~~ Some communication mechanism among CU, ALU and registers.

Input Unit

. With the help of input unit data from outside can be supplied to the computer. ~~program~~ Program and data is read into main storage from the input device or secondary storage under the control of CPU.

Example of input devices: Keyboard, Mouse, Hard disk, CD-ROM etc.

Output Unit:

- With the help of output unit computer results can be provided to the user, or it can be stored in storage device permanently for future use.
- Output data from main storage goes to output device under the control of CPU.

Example of output devices: Monitor, Printer, Hard disk etc.

Memory Unit:

. The memory unit is used to store the data and program
. CPU can work with information stored in memory unit (primary memory or main memory).
. These are basically semiconductor memories.

Two types of main memory

- Volatile memory (Random Access Memory)
- Non-volatile memory (Read only Memory)

System interconnection:

An interconnection network provides the means for the functional units to exchange information and coordinate their actions.

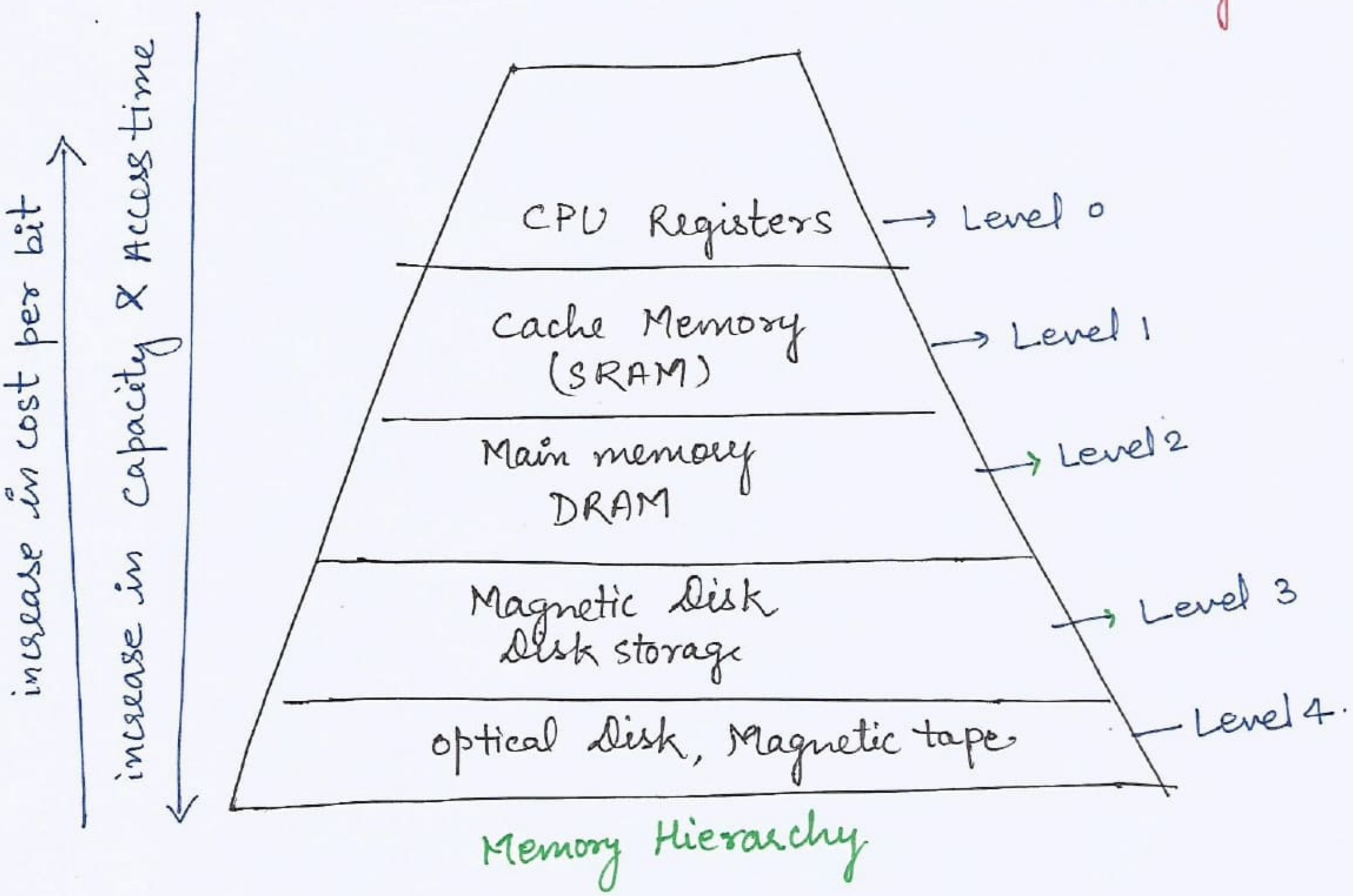
Example: System Bus

secondary Memories:

- secondary Memories are another kind of storage apart from primary memory.
- Secondary memories are non volatile memory.
- It is used for permanent storage of data and program.

Example of secondary memories:

- Hard disk, Floppy disk, magnetic tapes (magnetic devices)
- CD-ROM (optical devices)
- Thumb drive (or Pen drive) — is semiconductor memory.



Memory Hierarchy

Computer Architecture vs Computer Organization :

- computer Architecture refers to those attributes of the system visible to a programmer.
- computer organization refers to the operational units and their interconnections that realize the architectural specifications.

As an example, it is an architectural design issue whether a computer will have a multiply instruction. It is an organizational issue whether that instruction will be implemented by a special multiply unit or by method of repeated additions by using add unit of the system.

Difference between computer Architecture and computer Organization

CA	CO
• Architecture describes what the computer does	• organization describes how the computer does it.
• CA deals with the functional behavior of the computer	• CO deals with structural relationship.
• CA deals with high level design issues	• CO deals with low level design issues.
• For designing a computer, its architecture is fixed first.	• For designing a computer organization is decided after its architecture.

Computer Architecture:

- Computer architecture refers to those parameters of a computer system that are visible to a programmer or those parameters that have a direct impact on the logical execution of a program.

Example: Examples of architectural attributes include the instruction set, the number of bits used to represent different data types, I/O mechanisms, and techniques for addressing memory.

Computer Organization:

Computer Organizational refers to the operational units and their interconnections that realize the architectural specifications.

Examples: Examples of organizational attributes include those hardware details transparent to the programmer such as control signals, interfaces between the computer and peripherals, the memory technology used.

Two Basic Types of Architectures:

- Von Neumann
- Harvard

Von-Neumann Architecture:

- describes a general ~~and~~ framework or structure, ~~that~~ for a computer hardware, programming and data.
- vast majority of computers use this principle.
- Stored-Program principle.
- The principle says that:

> data and as well as instructions used to manipulate that data, should be stored together in the same memory area of the computer and instructions are carried out sequentially (one instruction at a time.

- The instructions and data use the same signal pathways and memory.

Example - Personal Desktops.

Harvard Architecture:

- Harvard architecture uses the physically separate storage and signal pathways for their instructions and data.

Example:

- Microcontroller (single-chip microcomputers) based computers

Topic 2

Number system

Number system provides a systematic way to represent and manipulate numbers:

Examples:

Decimal Number system
Octal number system
Binary Number System
Hexa decimal Number system

Decimal Number system:

Base or radix is 10

> Base or radix defines how many digits/symbols will be used to form number in that number system

Ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

→ weighted number system

→ Every digit position has a weight which is a power of 10.

Examples:

$234 = 2\times10^2 + 3\times10^1 + 4\times10^0$

$250.67 = 2\times10^2 + 5\times10^0 + 0\times10^{-1} + 6\times10^{-1} + 7\times10^{-2}$

Binary Number system:

- Base or radix is 2.
- Two digits: 0 and 1
- weighted number system

- Every digit has a weight that is a power of 2.
- Binary digits are called bits.

Examples:

$110 = 1\times2^2 + 1\times2^1 + 0\times2^0$

$101.01 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-1}$

Octal Number System:

- Base or radix is 8
- 8 Digits: 0, 1, 2, 3, 4, 5, 6, 7
- Weighted code.
- Every digit has a weight that is power of 8.

Examples.

$(254)_8 = 2\times8^2 + 5\times8^1 + 4\times8^0$

$(254.67)_8 = 2\times8^2 + 5\times8^1 + 4\times8^0 + 6\times8^{-1} + 7\times8^{-2}$

Hexadecimal Number system:

- Hex – six
- decimal – Ten.

Base or radix is 16.

Symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Note that symbol A is equivalent to 10
B is equivalent to 11
C is equivalent to 12
D is equivalent to 13
E is equivalent to 14
F is equivalent to 15

Number System Conversion:

Any base to decimal conversion: Base = r

Any number $d_{n-1}d_{n-2}\cdots d_1 d_0 \cdot d_{-1}d_{-2}\cdots d_{-m}$ can be converted into decimal as:-

decimal value = $d_{n-1}\times r^{n-1} + d_{n-2}\times r^{n-2} + \cdots + d_1\times r^1 + d_0\times r^0 + d_{-1}\times r^{-1} + d_{-2}\times r^{-2} + \cdots + d_{-m}\times r^{-m}$.

Binary to Decimal conversion

Example

① $(101011)_2 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$
$= (43)_{10}$

② $(.0101)_2 = 0\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$
$= .3125$

③ $(101.11)_2 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2}$
$= (5.75)_{10}$

Octal to decimal conversion:

$(23.17)_8 = 2\times8^1 + 3\times8^0 + 1\times8^{-1} + 7\times8^{-2}$
$= 16 + 3 + .125 + .109375$
$= (19.234375)_{10}$

Hexadecimal to decimal conversion:-

$(1A.23)_{16} = 1\times16^1 + A\times16^0 + 2\times16^{-1} + 3\times16^{-2}$
$= 16 + 10 + .125 + .01171875$
$= (26.13671875)$

Decimal to any base r:

$d_{n-1}\, d_{n-2} \text{ ---} d_1 d_0 \cdot d_{-1}\, d_{-2} \text{ ---} d_{-m}$

divide integer part by r.

multiply fractional part by r.

Decimal to Binary:

Examples:

$(239)_{10} = (\text{--})_2$

		Remainder
2	239	1
2	119	1
2	59	1
2	29	1
2	14	0
2	7	1
2	3	1
2	1	1
	0	

$(239)_{10} = (11101111)_2$

$(.634)_{10} = (\text{---})_2$.

$.634 \times 2 = 1.268$
$.268 \times 2 = 0.536$
$.536 \times 2 = 1.072$
$.072 \times 2 = 0.144$
$.144 \times 2 = 0.288$
$.288 \times 2 = 0.576$
$.576 \times 2 = 1.152$

$(.634)_{10} = (10100001\text{---})_2$

Decimal to octal conversion:

$(205.5)_{10} = (\text{---})_8$.

8	205	5
8	25	1
8	3	3
	0	

$.5 \times 8 = 4.0$

$(205.5)_{10} = (315.4)_8$

~~Hexadecimal to Decimal~~

Decimal to Hexadecimal:

Example $(205.5)_{10} = (---)_{16}$

16	205	D
16	C	C
	0	

$.5 \times 16 = 8.0$

$(205.5)_{10} = (CD.8)_{16}$.

Binary to Octal conversion:

For integer part

→ Scan the binary from right to left

→ Translate each group of three bits into the corresponding octal digits.

→ Add leading zeros if necessary.

For the fractional part:

→ Scan the binary number from left to right.

→ Translate each group of three bits into the corresponding octal digits.

→ Add trailing zeros if necessary.

Example. $(1100101.1101)_2$ —— $(\ ?\ )_8$

001 100 101 . 110 100

1 4 5 6 4

$(1100101.1101)_2$ $(145.64)_8$

Binary to hexadecimal conversion:

For integer part:

→ Scan given binary number from right to left
→ Translate each group of 4 bits into the corresponding hexadecimal digit.
→ Add leading zeros if necessary.

For fractional part:

→ Scan the binary number from left to right
→ Translate each group of 4 bits into the corresponding hexadecimal digit
→ Add trailing zeros if necessary.

Example. $(1100101.1101)_2 \Longrightarrow (?)_{16}$

Binary no.	Hexa decimal digit
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

$$0110\,0101 \cdot 1101$$
$$\leftarrow \quad \leftarrow \quad \rightarrow$$
$$6 \quad 5 \quad D$$

$$(1100101.1101)_2 = (65.D)_{16}$$

Binary to Decimal NS

$(101001.1011)_2$

$= 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 1\times32 + 0 + 1\times8 + 0 + 0 + 1 + .50 + 0 + .125 + .0625$

$= 32 + 8 + .50 + .125 + .0625$

$= 41 + .6875$

$= (41.6875)_{10}$

Octal to Decimal NS.

$(51.54)_8 = (\quad)_{10}$

$\Rightarrow 5\times8^1 + 1\times8^0 + 5\times8^{-1} + 4\times8^{-2}$

$= 40 + 1 + 5\times.125 + 4\times.015625$

$= 41 + .625 + .0625$

$= (41.6875)_{10}$

Hexadecimal to Decimal :-

$(29.B)_{16} = (\quad)_{10}$

$= 2\times16^1 + 9\times16^0 + B\times16^{-1}$

$= 2\times16 + 9\times1 + 11\times.0625$

$= 32 + 9 + .6875$

$= (41.6875)_{10}$

question convert 41.6875 into binary, Octal and hexadecimal Number system.

Ans (1) $(41.6875)_{10}$ into binary

2	41	1
2	20	0
2	10	0
2	5	1
2	2	0
2	1	1
	0	

$.6875 \times 2 = 1.3750$
$.375 \times 2 = 0.750$
$.750 \times 2 = 1.500$
$.500 \times 2 = 1.000$

$(41.6875)_{10} \Rightarrow (101001.1011)_2$

(2) $(41.6875)_{10}$ into Octal NS.

8	41	1
8	5	5
	0	

$.6875 * 8 = 5.5000$
$.5000 \times 8 = 4.0000$

$(41.6875)_{10} \Rightarrow (51.54)_8$

(3) $(41.6875)_{10}$ into Hexadecimal NS

16	41	9
16	2	2

$.6875 \times 16 = 11.0000$

$(41.6875)_{10} \Rightarrow (29.B)_{16}$

Question. Convert

(a) $(7562)_{10}$ to Octal.

(b) $(1938)_{10}$ to Hexadecimal

(c) $(175)_{10}$ to binary

8	7562	2
8	945	1
8	118	6
8	14	6
8	1	1
	0	

$(7562)_{10} = (16612)_8$

(b) $(1938)_{10}$ to Hexadecimal NS.

16	1938	2
16	121	9
16	7	7
	0	

$(792)_{16} = (1938)_{10}$

(c) $(175)_{10}$ to binary

2	175	1
2	87	1
2	43	1
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1
2	0	

$(175)_{10}$

$= (1010\ 1111)_2$

Topic 3

Complement:

→ used for subtraction operation
→ used for logical manipulation

Types of complements:

→ Radix complement (or r's complement)
→ Diminished radix complement (or (r-1)'s complement)

Base/radix = 10
→ 10's complement
→ 9's complement

Base/radix = 8
→ 8's complement
→ 7's complement

Base/radix = 2
→ 2's complement
→ 1's complement

Base/radix = 16
→ 16's complement
→ 15's complement.

Note: r's complement of N = (r-1)'s complement of N + 1

Note: complement of the complement gives the original number back

complement of complement of $N = N$

$$\overline{(\overline{N})} = N$$

How to find 9's complement of a decimal number N:

→ Subtract each digit of N from 9.

Example 9's complement of 2536 = 9999 - 2536
= 7463

How to compute 10's complement of a decimal number:

→ 10's complement of N = 9's complement of N + 1

Example. 10's complement of 2536 = 9's complement of 2536+1

= 9999 - 2536 + 1

= 7463 + 1

= 7464

How to compute 7's complement of an octal number:

→ Subtract each digit of octal number from 7.

Example. 7's complement of $(2645)_8$ = 7777 - 2645

$= (5132)_8$

How to compute 8's complement of an octal numbes N:

→ 8's complement of N = 7's complement + 1

Example. 8's complement of 2645.56 = 7777.77 - 2645.56

+1

= 5132.21 + 1

= 5132.22

How to compute 15's complement of an hexadecimal number N

→ Subtract each digit from F.

Example. 15's complement of B218 = FFFF - B218

$= (4DE7)_{16}$

15's complement of 256.A2 = FFF.FF - 256.A2

$= (DA9.5D)_{16}$

16's complement of an hexadecimal number:

16's complement of N = 15's complement of N + 1

Example. 16's complement of B218 = FFFF - B218 + 1

= 4DE7 + 1

= $(4DE8)_{16}$

16's complement of 256·A2 = FFF - FF - 256·A2 + 1

= DA9·5D + 1

= DA9·5E

1's complement of a binary number:

To obtain 1's complement of a binary number change 1's to 0's and 0's to 1's.

Example 1's complement of $(1011000)_2 = (0100111)_2$

2's complement of a binary number:

2's complement of N = 1's complement + 1

Example. 2's complement of (1011000) = (1's complement of 1011000) + 1

= 0100111 + 1

= 0101000

Subtraction using complements:

Subtraction of a number from another can be accomplished by adding the complement of the subtrahend to the minuend

$$\begin{array}{l} M \rightarrow \text{Minuend} \\ -N \rightarrow \text{Subtrahend} \end{array} \Rightarrow \begin{array}{l} M \text{ Minuend} \\ + \text{ complement of } N \end{array}$$

There are two types of complements

- → r's complement
- → $(r-1)$'s complement

Subtraction with r's complement:

The subtraction of two n-digit unsigned numbers $(M-N)$ in base r can be done as follows:

① Add the minuend to the r's complement of subtrahend N

$$M + (r^n - N) \Rightarrow M - N + r^n$$

Now two cases can occur ⓐ $M \geq N$

ⓑ $M < N$

② if $M \geq N$, the sum will produce an end carry, r^n, which is discarded. what is left is the result $(M-N)$

⇒ if carry occurs discard it

③ if $M < N$, then sum does not produce end carry and it is equal to $r^n - (N-M)$ which is r's complement of $(N-M)$.

To obtain the result (final answer), take the r's complement the sum and place a negative sign in front.

⇒ if carry does not occurs then take r's complement and put negative sign

Subtraction with (r-1)'s complement:

The subtraction of two n-digit unsigned numbers (M - N) in base r can be done as follows.

① Add the minuend M to the (r-1)'s complement of subtrahend N.

$$M + ((r-1)\text{'s complement of } N) \Rightarrow M + r^n - 1 - N$$
$$= M - N + r^n - 1$$

Now two cases can occur (a) $M \geq N$

(b) $M < N$

② if $M \geq N$, the sum will produce an end carry. Add this carry back to the result, called end-around carry.

Remove the carry and add it back to result.

③ if $M < N$, then sum does not produce an end carry and it is equal to $r^n - 1 - (N - M)$ which is (r-1)'s complement of (N-M). To obtain the result (final answer), take (r-1)'s complement of the sum and place a negative sign in the front.

⇒ if carry does not occur then take (r-1)'s complement of the sum and put negative sign.

Ques: Find subtraction of (72532 - 3250) using 9's complement

Ans: Both numbers must have the same number of digits

M = 72532

N = 03250

Subtraction using 1's complement:

$$\begin{array}{r} M \\ -N \end{array} \Rightarrow \begin{array}{r} M \\ +\ 1\text{'s complement}(+N) \\ \hline \text{Result.} \end{array}$$

case ① if end carry occurs, then add this end carry to the LSB of Result to get final result.

case ② if no end carry & occurred, then Result is negative and is in 1's complement form.

Example ①

$$\begin{array}{r} 26 \\ -5 \\ \hline ? \end{array}$$

$M = 26 \Rightarrow 11010$

$N = +5 \Rightarrow 00101$

$-N = -5 \Rightarrow 11010$

$\Rightarrow \begin{array}{r} M \\ +1\text{'s complement}(+N) \end{array} \Rightarrow \begin{array}{r} 11010 \\ 11010 \\ \hline (1)\,10100 \\ \quad\quad +1 \\ \hline 10101 \Rightarrow \text{Final result.} \end{array}$

Final Result = + (10101)

$= +(21)_{10}$

$\Rightarrow (21)_{10}$.

Example ②

$$\begin{array}{r} +5 \\ -26 \\ \hline ? \end{array}$$

$M = 5 \Rightarrow (000101)_2$

$+N = 26 \Rightarrow 011010$

$-N = -26 \Rightarrow 100101$

$$\begin{array}{r} 5 \\ +(-26) \end{array} \Rightarrow \begin{array}{r} 000101 \\ 100101 \\ \hline 101010 \Rightarrow \text{No end carry} \Rightarrow \text{Result is negative} \end{array}$$

Final Result $\Rightarrow -010101 \Rightarrow (-21)_{10}$

Subtraction using 2's complement:

$M - N = ?$.

can be written as

$M + (-N) = ?$ $\Rightarrow$ M + 2's complement $(+N) = ?$ Result.

case ① if end carry occurs $\Rightarrow$ then simply discard the end carry.

case ② if no end carry occurs $\Rightarrow$ then Result is negative and is in 2's complement form.

Example ①.

$$\begin{array}{r} 15 \\ -12 \\ \hline +3 \end{array}$$

$M = 15 \Rightarrow (1111)_2$.

$N = +12 \Rightarrow (01100)_2$.

$-N$ = 2's complement $(+12)$

$\Rightarrow$ 10100

Addition:

$$\begin{array}{r} \overset{1}{0}\overset{1}{1}\overset{1}{1}\overset{1}{1}1 \\ +10100 \\ \hline 00011 \end{array}$$

← make number of digits equal in both the numbers.

↑ end carry occurred, discard it.

Final result $\Rightarrow (00011)_2 \Rightarrow +3$.

Example ②.

$$\begin{array}{r} 12 \\ -15 \\ \hline -3 \end{array}$$

$M = 12 \Rightarrow 1100$

$N = +15 \Rightarrow 01111$

$-N$ = 2's complement of $(+15)$

$\Rightarrow 10001$

$$\begin{array}{r} M \\ +\underline{(-N)} \end{array} \Rightarrow \begin{array}{r} 01100 \\ +\,10001 \\ \hline 11101 \end{array}$$

↑ no end carry occurred. Hence Result is negative and in 2's complement.

Result: $11101 \Rightarrow -3$

$-16\ \ +8\ \ +4\ \ +1$

Final Result $= (11101)_2 \Rightarrow (-3)_{10}$

Ques- Subtract 15 from 12 with 8 bit representation using 2's complement.

$M = 12 \Rightarrow (00001100)_2$

$N = 15 \Rightarrow (00001111)_2$.

$-N = -15 \Rightarrow (11110001)_2$.

$$\begin{array}{r} M \\ +(-N) \end{array} \Rightarrow \begin{array}{r} 00001100 \\ +\,11110001 \\ \hline 11111101 \end{array}$$

↑ no end carry occurred. Hence Result is negative.

Result $= (-3) \Rightarrow (11111101)_2$.

$+64\ +32$

$1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1$

$-128\qquad +16\ \ +8\ \ +4\ \ +1$

Topic 4

Representing Numbers in binary:

- How many distinct numbers can be represented in n bits?
  - → Each bit can be in one of two states (0 or 1)
  - → Total number of possible combinations

$$2 \times 2 \times 2 - - - - \text{to } n \text{ terms} = 2^n$$

→ Number can be unsigned or signed.
  - → A unsigned number has only a magnitude, but no sign.
  - → A signed number has both a magnitude and also a sign (positive or negative).

Unsigned binary numbers:

- An n-bit binary number can have 2^n distinct combinations
- minimum = 0 and maximum $2^n - 1$
- For example for n=3, the 8 distinct combinations are.

0 - 000 min = 0
1 - 001
2 - 010
3 - 011
4 - 100
5 - 101
6 - 110
7 - 111 max = $2^3 - 1 = 7$

- An n bit binary number $b_{n-1}\, b_{n-2} - - - b_1\, b_0$
- Equivalent decimal value = $b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} - - - b_1 \times 2^1 + b_0 \times 2^0$

Signed Integer Representation:

There are three approaches to represent signed integer(positive or negative).

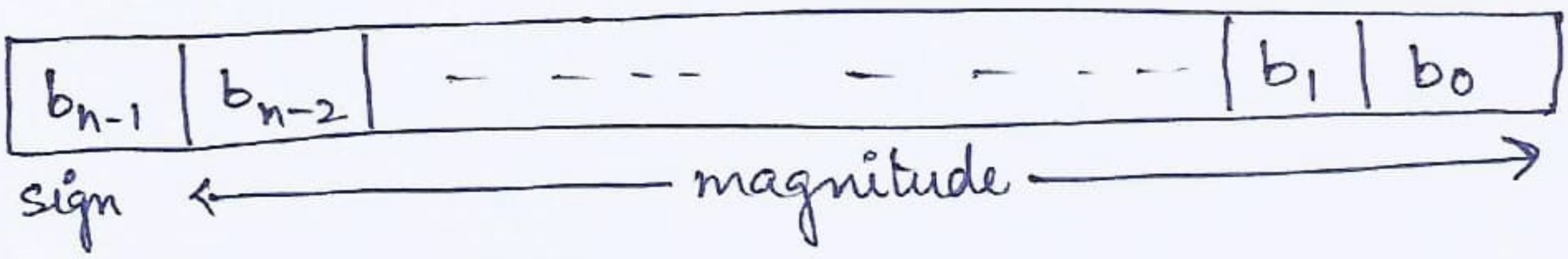
(i) Signed magnitude representation
(ii) One's complement representation
(iii) Two's complement representation

(i) Signed magnitude representation:

- For n-bit number representation
  → The Most significant bit (MSB) indicate sign.
    0 → positive
    1 → negative
  → The remaining (n-1) bit represent the magnitude

→ Range of numbers to be represented with n bits in signed magnitude method.

$$-(2^{(n-1)}-1) \text{ to } +(2^{n-1}-1)$$



Problem or disadvantage of signed magnitude representation

- Two different representations of zero.

+0: 0 0000000
−0: 1 0000000

(ii) Ones complement representation method:

→ positive numbers are represented exactly as in signed magnitude method.

→ negative numbers are represented in 1's complement form.

→ The signed 1's complement representation of $-N$ is obtained by complementing all bits of $+N$, including sign bit.

Example To find the representation of ~~+5~~ -5, first find $+5$ and then take 1's complement.

$+5 = 0101$

$-5 = 1010$

→ Range of numbers that can be represented in 1's complement

$$\text{minimum} = -(2^{n-1}-1)$$

$$\text{maximum} = +(2^{n-1}-1)$$

→ It also has two different representations of zero

$+0 \quad 00000000$

$-0 \quad 11111111$

→ Advantage of 1's complement representation
→ subtraction can be done using addition.

Twos complement Representation:

- positive numbers are represented exactly as in signed magnitude method.
- negative numbers are represented ~~as~~ in 2's complement form.

→ How to compute 2's complement of a number:

complement every bit of the number (1 to 0 and 0 to 1), and then add one to resulting number.

→ Range of numbers that can be represented in 2's complement form:

$$\text{minimum} = -2^{n-1}$$

$$\text{maximum} = +(2^{n-1}-1)$$

→ Advantages of 2's complement representation:

- → unique representation of zero
- → subtraction can be done using addition

→ Almost all computers today use 2's complement representation for storing negative numbers.

The signed 2's complement representation is obtained by taking 2's complement of positive number including its signed bit.

signed number.	Signed magnitude method	1's complement method	2's complement method.
+14	0 0001110	00001110	00001110
-14	1 0001110	11110001	11110010

Topic 5

Binary Codes:

→ Binary codes provide different ways to convert a decimal number into 0 and 1 form.

→ There are various codes given following:

- → Binary
- → BCD
- → Excess-3
- → Gray code
- → 2421 code and so on

Here the major concern is how many (minimum) number of bits are used to represent a decimal number in different code representation.

Binary number system:

→ uses two digits 0 and 1
→ These two digits are also called bits
→ Every digit position has a weight that is a power of 2.
→ Base or radix is 2.

Example:

$(6)_{10} = (110)_2 = 1\times2^2 + 1\times2^1 + 0\times2^0$

$(5.25)_{10} = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$

$= (101.01)_2$

BCD: Binary Coded decimal Number Representation

→ It is sometimes desirable to manipulate numbers in decimal instead of converting them to binary.

→ Decimal to binary and binary to decimal conversion process is complex.

→ One popular code to represent decimal digits is BCD.

→ In BCD, each decimal digit is represented by its 4-bit binary equivalent.

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

Example 1

Decimal

BCD.

2 3 8
↓ ↓ ↓
0010 0011 1000

Example 2

Decimal 12.39

BCD. 00010010.00111001

Note combination of 4 bits is called a nibble

Addition of BCD codes:

When we add two BCD numbers, we may have to go to for a correction step where 6 (0110) is added to one of the nibble.

→ 0110 is added to the nibble when either a nibble is one of the six invalid combinations (1010, 1011, 1100, 1101, 1110, 1111) or there is a carry in the previous nibble.

→ Examples:

```
 23  →   0010  0011
+46  →  +0100  0110
----     ----------
 69       0110  1001
```

```
 23 →   0010  0011
+48   +0100  1000
----   ----------
 71     0110  1011
            +0110
            ------
        0111  0001
```

```
          (1)
 28 →   0010  1000
+39 →  +0011  1001
----    ----------
 67     0110  0001
             +0110
             ------
        0110  0111   (carry generated)
```

Gray codes:

→ Gray code is a type of non-weighted binary code where successive codeword differs only in one bit.

→ Any code with this property is called cyclic code.

> A code C is cyclic if
> (i) C is linear code.
> (ii) Any cyclic shift of a codeword is also a codeword, i.e whenever
> $a_0, a_1, a_2 - - - a_{n-1} \in C$ then also
> $a_{n-1}\, a_0\, a_1 - - - a_{n-2} \in C$

→ Gray code is cyclic code.

→ Gray code is also called self reflecting code.
  → suppose we have the gray code representation for m bits.
  → To obtain the gray code representation for (m+1) bits, we write down two m-bit representations one below the other, with the second one being the mirror image of the first one.
  → We then add 0 at the beginning of every code in the first group, and a 1 at the beginning of every code in the second group.

How to write a gray code and Example of self reflection

Two bit gray code.

0 0
0 1
1 1
1 0

Three bit gray code.

0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0

Four bit gray code.

0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0

cyclic property

0 0 0 1 ● (valid code word)

1 0 0 0 (also a valid code word)

Applications of Gray codes:

→ used in Karnaugh Map

→ to measure the angle of rotation of a wheel.

Binary to gray code conversion:

Let $g_{n-1}\ g_{n-2} \dots g_1\ g_0$ and $b_{n-1}\ b_{n-2} \dots b_1\ b_0$ denote n-bit Gray code and its binary equivalent respectively.

$$g_i = b_i \oplus b_{i+1} \text{ for } 0 \leq i \leq n-2$$

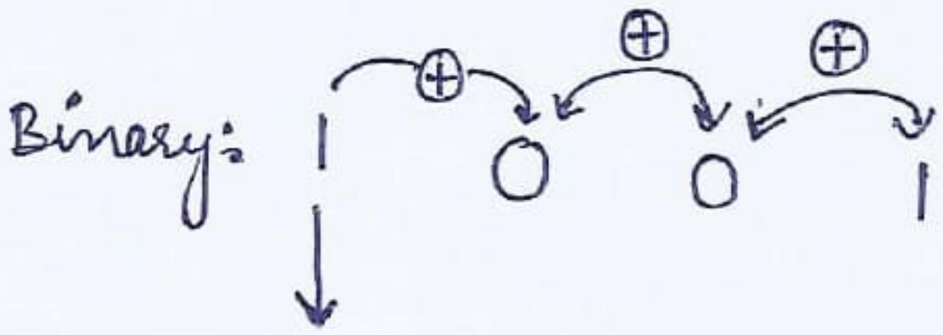
$$g_{n-1} = b_{n-1}$$

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Example.

Binary number: 1 0 0 1

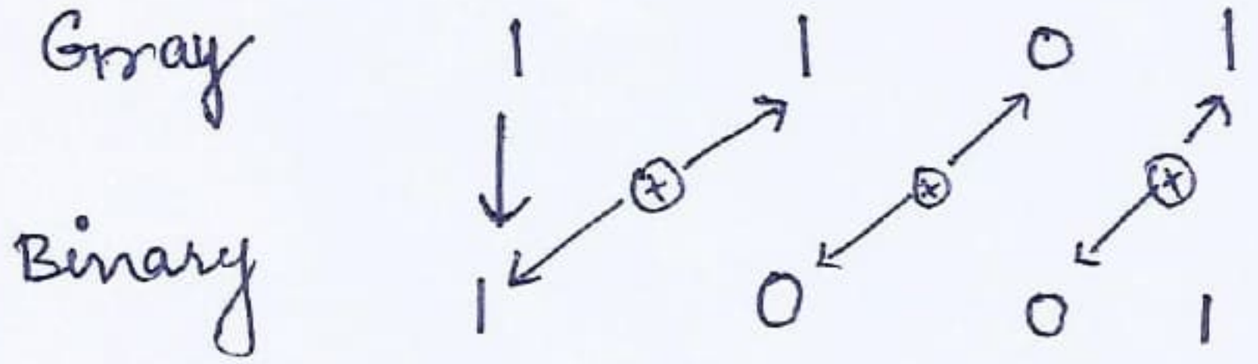
Gray code : ?



1 1 0 1 ⇒ Gray code.

Gray to binary conversion:

→ we start with the left most bit and proceed to the least significant bit and do:



Excess-3 codes:

→ It is non-weighted type of code.

→ To convert any decimal number into its excess-3 form, add 3 to each decimal digit and then write its 4 bit binary equivalent

→ It is a self complementary code.

Example: convert 2 3 8 into Excess-3 code.

```
  2   3   8
+ 3   3   3
-------------
  5   6   11
  ↓   ↓   ↓
 0101 0110 1011 → Excess-3 code of 238
```

Decimal digits	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Self complementary code:

⇒ Self complementary means that the 1's complement of an excess-3 code decimal of a number is the excess-3 code for the 9's complement of that decimal number.

Self complementary codes:

→ the 9's complement of a decimal number can be easily obtained by replacing 1's by 0's and 0's by 1's.

→ Example:- Excess-3
-2421

Simple Binary Arithmetic

Binary addition:

Input bits A	Input bits B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$(01010)_2 + (10111)_2 \Rightarrow ?$

Carry: 1 1 1 0

```
  01010
+ 10111
-------
100001
-------
```

Binary Subtraction

Input bits A	Input bits B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$(10010)_2 - (01100)_2 = ?$

Borrow: 0 1 0 0

```
  10010
- 01100
-------
  00110
-------
```

Binary multiplication:

Input bits A	Input bits B	Product
0	0	0
0	1	0
1	0	0
1	1	1

```
  10011001
      X 1110
-----------
  00000000
 10011001X
100110010X
1001100100X
-----------
10000101 0
-----------
```

Binary division

```
110) 101010 (111
   -110
   ----
    1001
   -110
   ----
    0011 0
     -110
     ----
      000
```

101 < 110.

1010 > 101

1001 > 110

110 = 110.

Overflow:

→ When two numbers are added, then result may be larger than can be held in the word size register being used. This condition is called overflow.

→ whenever the result exceeds its range, overflow is said to be occurred.

→ Result is out of range.

Range for n-bet			
unsigned	Signed magnitude	1's complement	2's complement
0 to $(+2^n - 1)$	$+(2^{n-1}-1)$ to $-(2^{n-1}-1)$	$+(2^{n-1}-1)$ to $-(2^{n-1}-1)$	-2^{n-1} to $+(2^{n-1}-1)$

Note: ① when numbers of different signs ared added, then overflow never occurs.

② when numbers of same signs are added, then overflow may occur.

different signs of operands ⇒ No overflow
same signs of operands ⇒ overflow may occur

Overflow Detection:
→ Unsigned addition
→ Signed addition

Unsigned Addition: The presence of carryout will denote the overflow.

Example (1)

$$\begin{array}{r} 11 \\ +\ 2 \\ \hline 13 \end{array} \Rightarrow \begin{array}{r} 1011 \\ +\ 0010 \\ \hline 1101 \end{array}$$

Number of bits = 4

Range = $2^4 - 1 \Rightarrow 15$

↑ no carryout ⇒ no overflow occurred.

Example (2)

$$\begin{array}{r} 11 \\ 8 \\ \hline 19 \end{array} \Rightarrow \begin{array}{r} 1011 \\ +\ 1000 \\ \hline 10011 \end{array}$$

number of bits = 4

Range upto 15

19 is larger than 15

↑ carryout ⇒ overflow occurred.

Signed Addition :

→ signed numbers can be represented using

1. Signed magnitude
2. 1's complement
3. 2's complement

Signed magnitude Addition

→ Sign bits does not participate in the operation

→ The presence of carryout will denote the overflow.

→ when both the operands have same sign, copy the sign bit in the result signbit and add the magnitude.

→ when both the operands have different signs, copy the sign bit of larger magnitude in the sign bit of result and subtract the smaller magnitude from the larger magnitude.

Example ① Signed Magnitude Method.

$$\begin{array}{rl} & (+6) \Rightarrow \quad 0\,|\,110 \\ + & \underline{(+2)} \qquad\quad + 0\,|\,010 \\ & +8 \qquad\quad\; 0\,|\,1000 \end{array}$$

carryout ⇒ overflow occurred.

Ans = +8

Range = $\pm 2^{4-1} - 1 \Rightarrow \pm 7$

Example ②

$$\begin{array}{rl} & (-6) \Rightarrow \quad 1\,|\,110 \\ + & \underline{(+2)} \qquad\quad + 0\,|\,010 \\ & -4 \qquad\quad\; 1\,|\,100 \end{array}$$

6 is larger than 2. Then subtract magnitude of 2 from six. and copy the sign bit of (-6) to the result.

⇒ no overflow as result is in range.

Example ③

$$\begin{array}{rl} & \quad\;\; 1 \\ & (+6) \Rightarrow \quad 0\,|\,110 \\ + & \underline{+(-2)} \qquad\quad 1\,|\,010 \\ & +4 \qquad\quad\; 0\,|\,100 \end{array}$$

⇒ no overflow.

No carryout

Example ④

$$\begin{array}{rl} -6 & \Rightarrow \quad 1\,|\,110 \\ -2 & \Rightarrow \quad 1\,|\,010 \\ & \quad\; 1\,|\,1000 \end{array}$$

carryout ⇒ overflow occurred.

⇒ Overflow Detection in 1's complement and 2's complement Addition

① when operands are same signed numbers but the result is having different (opposite) sign, then overflow occurs.

② if carry bit into the sign bit position and carry out of sign bit position is not same, then overflow occurs.

⇒ X-OR gate can be used ⇒ if both the carries will be different then X-OR gate will give high output.

1's complement addition :-

```
          1  1
(+6) ⇒    0  110
(+2) ⇒    0  010
-------------------
+8        1  000
```

① $c_{in} = 1$, $c_{out} = 0$ ⇒ different ⇒ overflow occurred.

② both number have sign bits 0 and 0 while result has sign bit 1.

```
          1
(+6) ⇒    0  1 1 0
+(-2) ⇒  +1  1 0 1
-------------------
+4       10 0 1 1
         ↑      +1
         ---------
         0  1 0 0
```

+2 = 0 0 1 0

-2 ⇒ 1 1 0 1

Final Ans ⇒ 0 1 0 0 ⇒ No overflow occurred

Note: if carryout occurs in the 1's complement addition/subtraction then this carryout will be added to LSB (Least Significant bit) to get final result.

```
-6 ⇒   1 0 0 1
+2 ⇒  +0 0 1 0
--------------
-4     1 0 1 1
```

+6 ⇒ 0 1 1 0

-6 ⇒ 1 0 0 1

Final Ans: -(1's complement of 1011) ⇒ -0100 ⇒ -4

both signs are different ⇒ no overflow

Result is having signbit as 1. So the number Result is negative and is in 1's complement form

$$
\begin{array}{r}
(-6) = \quad 1\,0\,0\,1 \\
+(-2) = +\ 1\,1\,0 \\
\hline
1\,0\,1\,1\,0
\end{array}
$$

↑ sign bit of result is different from sign bits of operands.

$cin = 0$
$cout = 1$ → different ⇒ overflow occurred.

+6 = 0 1 1 0
−6 = 1 0 0 1
+2 = 0 0 1 0
−2 = 1 1 0 1

Addition using 2's complement:

$$
\begin{array}{r}
+6 \rightarrow \quad 0\,1\,1\,0 \\
+2 \rightarrow \quad 0\,0\,1\,0 \\
\hline
+8 \quad\quad 1\,0\,0\,0
\end{array}
$$

⇒ overflow occurred.

+6 ⇒ 0 1 1 0
−6 ⇒ 1 0 1 0

$$
\begin{array}{r}
-6 \Rightarrow \quad 1\,0\,1\,0 \\
+2 \quad\quad 0\,0\,1\,0 \\
\hline
-4 \quad\quad 1\,0\,0\,0
\end{array}
$$

⇒ no overflow

↑ sign bit is one. Result is negative and is in 2's complement.

Final result = −(2's complement of 1000)
= −(0100)
= −4

+2 = 0 0 1 0
−2 =

$$
\begin{array}{r}
-6 = \quad 1\,0\,1\,0 \\
-2 = \quad 1\,1\,1\,0 \\
\hline
-8 \Rightarrow 1\,1\,0\,0\,0
\end{array}
$$

↑ carryout ⇒ discard it

no overflow ⇒ sign bits are same. Range ⇒ -2^{4-1}
$= -8$

Final result ⇒ 1000 ⇒ −8

(in 2's complement method 1000 is for −8)

$$
\begin{array}{rr}
(-6) \Rightarrow & 1\,0\,1\,0 \\
+(-3) \Rightarrow + & 1\,1\,0\,1 \\
\hline
 & 1\,0\,1\,1\,1
\end{array}
$$

$+3 = 0\,0\,1\,1$

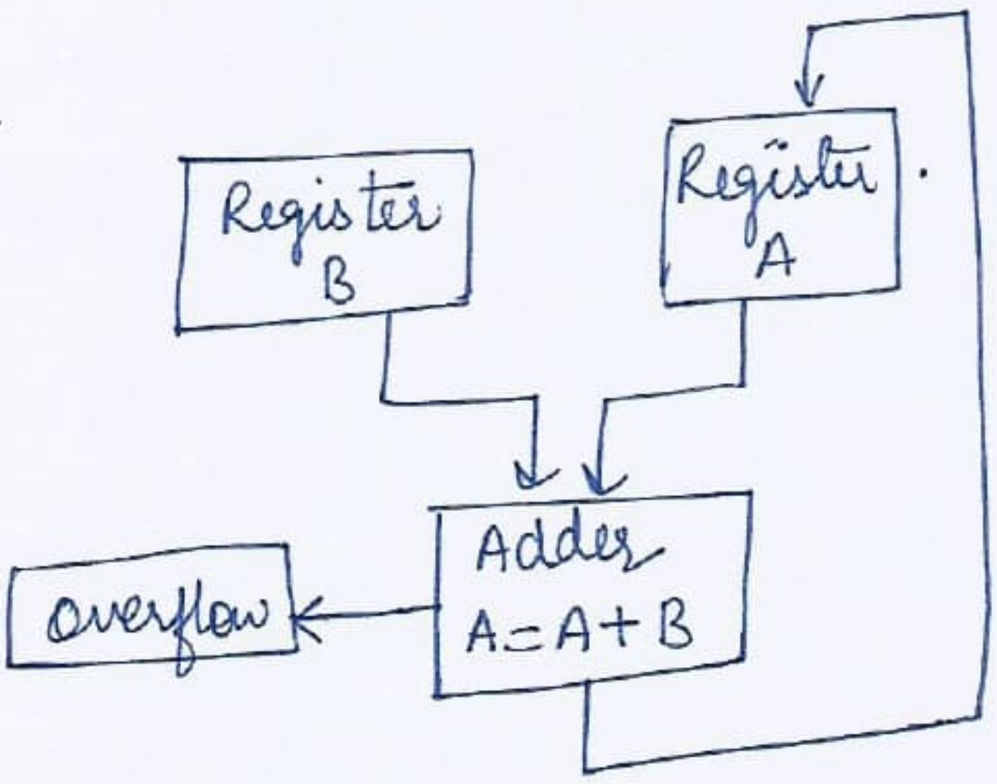
$-3 = 1\,1\,0\,1$

cout $\Rightarrow 1$

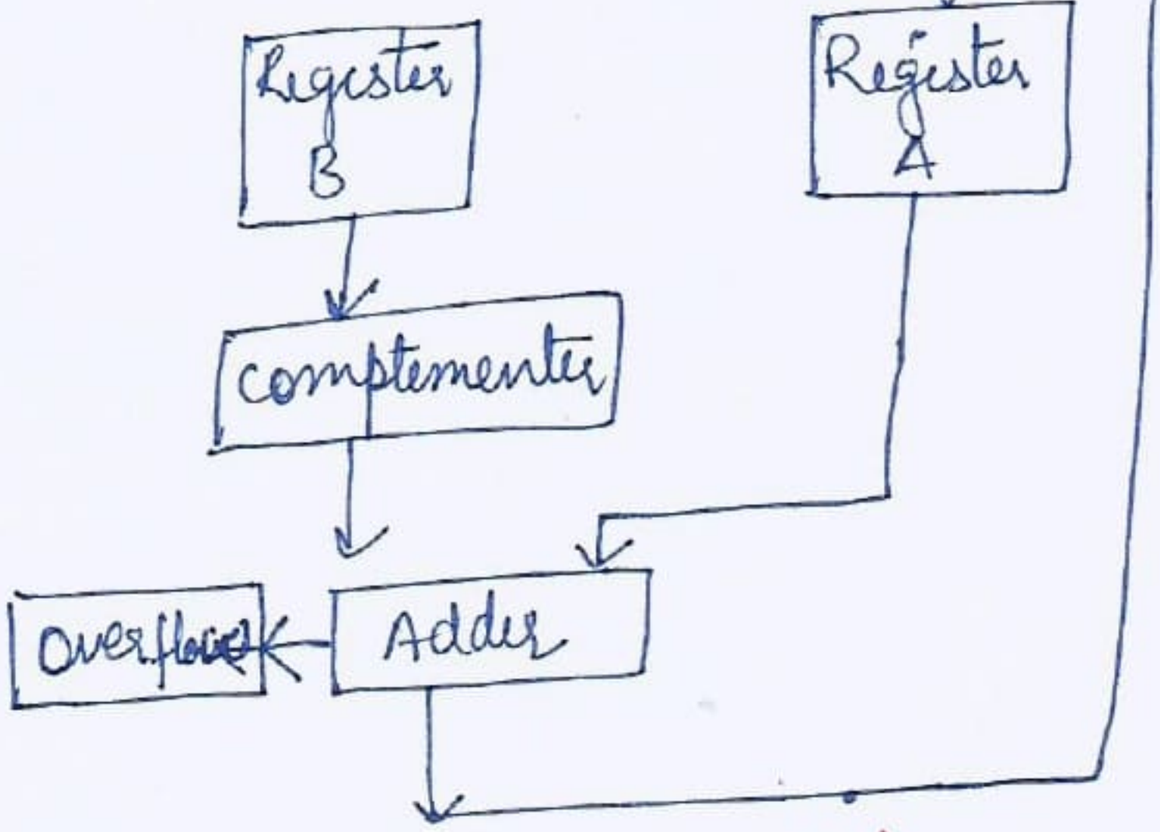
Cin $= 0 \Longrightarrow$ overflow occurred.

sign bits of operands and result are different $\Rightarrow$ overflow.

Flow chart for addition and subtraction



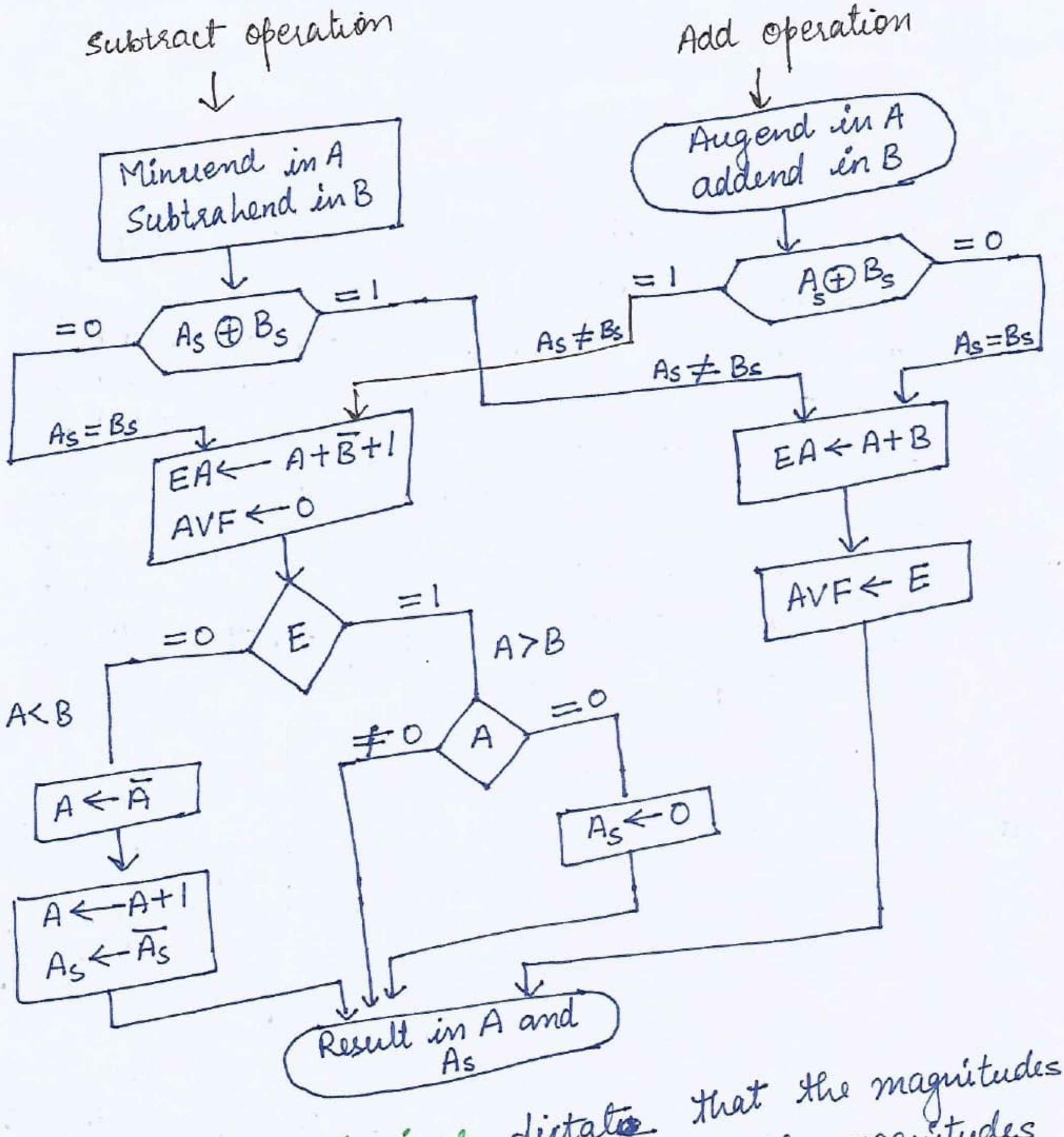
Flow chart for Addition



Flow chart for Subtraction

Flowchart for addition and subtraction:

Signed Magnitude addition and Subtraction



For addition, identical signs dictate that the magnitudes be added.

be subtracted. different signs require that the magnitudes

For Subtraction, different signs dictate that magnitudes be added, identical signs require that magnitude be subtracted.

AVF ⇒ Add Overflow Flip Flop that holds the overflow bit when A and B are added.

Operation	ADD	Subtract A>B	Subtract A<B	Subtract A=B
(+A) + (+B)	+(A+B)			
(+A) + (-B)		+(A-B)	-(B-A)	+(A-B)
(-A) + (+B)		-(A-B)	+(B-A)	+(A-B)
(-A) + (-B)	-(A+B)			
(+A) - (+B)		+(A-B)	-(B-A)	+(A-B)
(+A) - (-B)	+(A+B)			
(-A) - (+B)	-(A+B)			
(-A) - (-B)		-(A-B)	+(B-A)	+(A-B)

Hardware Implementation