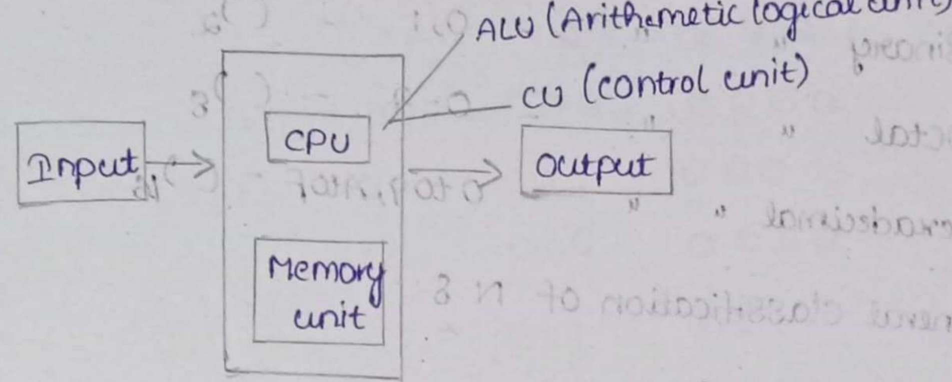
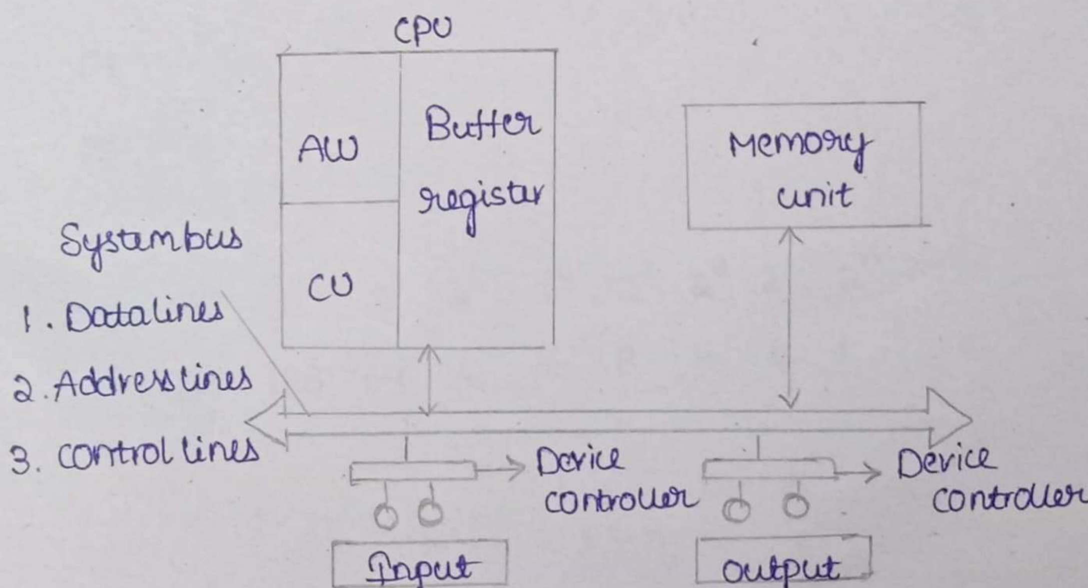
28/4/22 Functional unit of computer

Introduction::



* Memory register
  (or)
  Buffer register → Temporary storage
* CU → to control all the operations
* Memory unit → primary unit
  &
  Secondary unit → (It will fetch instruction by instructions
* cpu can access primary memory or directly can access input.

29/4/22 Basic functional units of a computer::
(Interconnection)



Number Systems:-

1. Decimal Number System - 0 to 9 - $(\ )_{10}$
2. Binary " " - 0, 1 - $(\ )_2$
3. Octal " " - 0-7 - $(\ )_8$
4. Hexadecimal " " - 0 to 9, A to F - $(\ )_{16}$

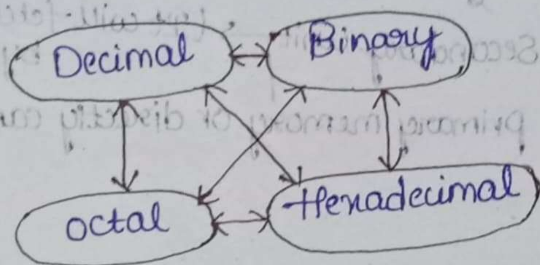
General classification of N.S:-

1. Positional NS
2. Non-positional NS

I, II, III, - - -

we can't perform any arithmetic op's

It is used to represent the roman numbers



Representation of Binary numbers:-

NO.	1-bit	2-bit	3-bit	4-bit
	1	2 1	4 2 1	8 4 2 1
0	0	00	000	0000
1	1	01	001	0001
2		10	010	0010
3			011	0011
4			100	0100
5			101	0101
6			110	0110
7			111	0111
8				1000
9				1001
10				1010
11				1011
12				1100
13				1101
14				1110
15				1111

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Conversion methods::

Binary to decimal::

1. Convert 11011 into decimal

$(\overleftarrow{11011}) \Rightarrow 1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0$

$= 16+8+0+2+1 = 27$

2. $(11010.101)_2 = (?)_{10}$

$(11010.101)_2 = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

$= 16+8+0+2+0.5+0.25+0.125$

$= (26.875)_{10}$

Octal to decimal::

1. $(1054)_8 = (?)_{10}$

$(1054)_8 = 1\times8^3+0\times8^2+5\times8^1+4\times8^0$

$= 512+0+40+4$

$= (556)_{10}$

2. $(22.74)_8 = (?)_{10}$

$(22.74)_8 = 2\times8^1+2\times8^0+7\times8^{-1}+4\times8^{-2}$

$= 16+2+0.875+0.0625$

$= (18.9315)_{10}$

Hexadecimal to decimal conversion::

1. $(A23)_{16} \Rightarrow 10\times16^2+2\times16^1+3\times16^0$

$= 2560+32+3$

$= (2595)_{10}$

2. $(A23.357)_{16} \Rightarrow 10\times16^2+2\times16^1+3\times16^0+3\times16^{-1}+5\times16^{-2}+1\times16^{-3}$

$= 2595+0.1875+0.0195+0.0002441$

$= (2595.20727)_{10}$

Decimal to Binary conversion:

1. $(21)_{10} = (?)_2$

$$\begin{array}{r|ll} 2 & 21 & -1 \\ 2 & 10 & -0 \\ 2 & 5 & -1 \\ 2 & 2 & -0 \\ & 1 & \end{array}$$

Shortcut

16	8	4	2	1
1	0	1	0	1

$16+4+1 \Rightarrow 21$

$(21)_{10} = (10101)_2$

2. $(38.15)_{10}$

$$\begin{array}{r|ll} 2 & 38 & \\ 2 & 19 & -0 \\ 2 & 9 & -1 \\ 2 & 4 & -1 \\ 2 & 2 & -0 \\ & 1 & -0 \end{array}$$

$0.15 \times 2 = 0.3 \Rightarrow 0$

$0.3 \times 2 = 0.6 \Rightarrow 0$

$0.6 \times 2 = 1.2 \Rightarrow 1$

$0.2 \times 2 = 0.4 \Rightarrow 0$

$0.4 \times 2 = 0.8 \Rightarrow 0$ iterate

$0.8 \times 2 = 1.6 \Rightarrow 1$

$(38.15)_{10} = (100110.001)_2$

Octal to binary conversion:-

1. $(71)_8 = (?)_2$

$((111)\ (001))_2$

$\Rightarrow (110001)_2$

2. $(765.031)_8 = (?)_2$

$(111\ 110\ 101.\ 000\ 011\ 001)_2$

Binary to octal conversion:-

1. $(10010.1011)_2$

Empty units 0

$\underset{2}{010}\ \underset{2}{010}.\ \underset{5}{101}\ \underset{4}{100} \Rightarrow (22.54)_8$

Binary to Hexadecimal conversion:-

1. 1011001001.100001

$\underset{2}{0010}\ \underset{C}{1100}\ \underset{9}{1001}.\ \underset{8}{1000}\ \underset{4}{0100} \Rightarrow (2C9.84)_{16}$

Octal to Hexadecimal conversion:-

1. $(256)_8 = (\ )_{16}$

Step-1:- convert each binary digit into its 3-bits binary code

$(\overset{2}{010}\ \overset{5}{101}\ \overset{6}{110})$

Step-2:- convert " " " " " 4-bits " "

0|1010|1110

$\underset{(0}{0000}\ \underset{A}{1010}\ \underset{E)_{16}}{1110}$

Hexadecimal to octal conversion:

1. $(2AB)_{16} \rightarrow (?)_8$

2 A B

0010 1010 1011 ⇒ Each value into 3-bits

$\underbrace{001}_{1}\underbrace{010}_{2}\underbrace{101}_{5}\underbrace{011}_{3} \Rightarrow (1253)_8$

Hexadecimal to binary conversion:

1. $(47A)_{16}$

4 7 A

0100 0111 1010

$\Rightarrow (010001111010)_2$

Hexadecimal to decimal conversion:

$(A23)_{16} \Rightarrow A \times 16^0 + 2 \times 16^1 + 3 \times 16^0$

$= 2560 + 32 + 3 = (2595)_{10}$

Decimal to octal conversion:

1. $(574)_{10} \Rightarrow (?)_8$

$$\begin{array}{r|l} 8 & 574 \\ 8 & 71 - 6 \\ 8 & 8 - 7 \\ & 1 - 0 \end{array} \Rightarrow (1076)_8$$

2. $(18.6875)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 18 \\ & 2 - 2 \end{array}$$

$0.6875 \times 8 = 5.5 = 5$

$= 4$

$\Rightarrow (22.54)_8$

2/5/22

Data representation:

1. Data types — Alphabets (lower case, upper case), Numeric (Decimal, Binary, Octal, Hexadecimal), Special Symbols ⇓ (&, $, -) } Types of data that we can give input to the computer

2. compliments — NO.s: Signed −ve (1), unsigned +ve (0)

⇓

⇒ To deal with −ve numbers

⇒ In conversion of Subtraction to addition

⇒ Classified into 2 types

1. (r−1)'s compliment $[(r^n-1)-N]$ ∵ r = radix (or) base

2. r's compliment $[r^n-N]$

Decimal → 9's & 10's

Binary → 1's & 2's

Ex:- $-(132)_{10}$

$r=10;\ N=132;\ n=3$ (no. of digits)

$$\begin{array}{r} 999 \\ (-)\ 132 \\ \hline 867 \end{array}$$ is the 9's compliment

$$\begin{array}{r} 1 \\ \hline 868 \end{array}$$ ⇒ 10's compliment

$\Rightarrow (10^3-132)$

$= 1000-132$

$= 868$

$-(1001)_2$

$r=2;\ N=1001;\ n=4$

1001 ⇒ 0110 ⇒ 1's compliment

$$\begin{array}{r} 1 \\ \hline 0111 \end{array}$$ ⇒ 2's compliment

$(476)_8$

$r=8;\ N=476;\ n=3$

$[(8^3-1)-(476)_8]$

$=(511)_{10}-(476)_8$

$=(777)_8-(476)_8$

$=(301)_8$

8 | 511 - 7
8 | 63 - 7
7

* To deal with ~~decimal~~ no.s we have two representations

1. Fixed point representation
2. floating " "

Ex:·· 736.0 → integer (extremely right) } fixed point
.736 → ~~float~~ fraction (extremely left) }

7.36 } floating point
73.6
736.4

Fixed point representation:·

* Either signed (or) unsigned

⇒ To deal with signed no.s, we have 3 methodologies

1. Signed magnitude representation
2. " 1's compliment
3. " 2's "

Ex:· $-(14)_{10}$

1 → (10001110)
↳ magnitude will be +ve

2 → (11110001)
↳ 1's compliment of magnitude

3 → (11110010)
↳ 2's compliment of magnitude

Arithemetic addition:-

Adding the no.s in signed 2's complement system require only addition and complementation"

* Add the two no.s, including their sign bits.
* Discard any carryout of the sign (leftmost) bit position

Note:- -ve no.s must be initially be in 2's complement and if the sum obtained after the addition is -ve it is in 2's compliment form

Ex:-

+6	00000110
+13	00001101
+19	00010011

-6	00000110 (6)
	11111001 (1's complement)
	1
	11111010
+13	00001101
+7	1 00000111

Arithemetic Subtraction:

2's complement Subtraction includes following steps

* Take the 2's complement of number to be Subtracted including the sign bit

* Add it the next number

* Discard the carry out of the sign bit position.

Ex: $-6-(-13) = +7$

-6

2's complement of -13

11111101
00001101

00000111

5/5/22

Floating point representation:

⇒ The number should be represented in the form of

$m \times r^{e}$ (e → exponent; m ⇒ mantissa; r → radix) (or) $s \times b^{e}$ (e → exponent; s ⇒ Significant; b → base)

Ex: 4.2×10^{8} (4.2 → m, 10 → r, 8 → e)

IEEE notation:

It can be of 16-bits, 32-bits, 64-bits, 128-bits

16-bits → Half-precision

32-bits → Single

64-bits → double

128-bits → quadtraple

32-bits representation:

Sign bit	Exponent (8 bits)	mantissa
1-bit		23 bits

← 32-bits →

Ex: ~~2450.00~~ $(250.03125)_{10}$

$(250)_{10} = 11111010$

$.03125 = 0.00001$

⇒ $(11111010.00001)_2$

1	1	1	1	1	0	1	0	.	0	0	0	0	1
8	7	6	5	4	3	2	1						

$(1111101000001) \times 2^{8}$

$m \times r^{e}$

0	00001000	1111101000001 0 ---- 0 (23)

for exponent → add zeroes to left

for mantissa → " " " right

2. $(102.9375)_{10}$

$(1100110.1111)_2$

1	1	0	0	1	1	0	.	1	1	1	1
7	6	5	4	3	2	1					

$(11001101111) \times 2^7$

$m \times r^e$

0	00000111	1100110111100----0

(12 ---- 23)

$$\begin{array}{r|l} 2 & 102 - 0 \\ 2 & 51 - 1 \\ 2 & 25 - 1 \\ 2 & 12 - 0 \\ 2 & 6 - 0 \\ 2 & 3 - 1 \\ & 1 \end{array}$$

$0.9375 \times 2 = 1.87 = 1$

$0.87 \times 2 = 1.75 = 1$

$0.75 \times 2 = 1.25 = 1$

$0.5 \times 2 = 1 = 1$

3. $(250.75)_{10}$

$(11111010.11)_2$

1	1	1	1	1	0	1	0	.	1	1
8	7	6	5	4	3	2	1			

$(1111101011) \times 2^8$

$m \times r^e$

0	00001000	11111010110----0

(1 ---- 23)

$$\begin{array}{r|l} 2 & 250 - 0 \\ 2 & 125 - 1 \\ 2 & 62 - 0 \\ 2 & 31 - 1 \\ 2 & 15 - 1 \\ 2 & 7 - 1 \\ 2 & 3 - 1 \\ & 1 \end{array}$$

$0.75 \times 2 = 1.50 \Rightarrow 1$

$0.50 \times 2 = 1 \Rightarrow 1$

$0 \times 2 = 0 \Rightarrow 0$

4. $(173.75)_{10}$

$(10101101.11)_2$

1	0	1	0	1	1	0	1	.	1	1

$(1010110111) \times 2^8$

$m \times r^e$

0	00001000 ~~1010110111~~	1010110111----0

(1 ---- 23)

$$\begin{array}{r|l} 2 & 173 - 1 \\ 2 & 86 - 0 \\ 2 & 43 - 1 \\ 2 & 21 - 1 \\ 2 & 10 - 0 \\ 2 & 5 - 1 \\ 2 & 2 - 0 \\ & 1 \end{array}$$

$0.75 \times 2 = 1.50 \Rightarrow 1$

$0.50 \times 2 = 1 \Rightarrow 1$

Character representation:

1. BCD → Binary codeddecimal
2. EBCDIC → Extended binary coded decimal interchange code
3. ASCII → English (American)
4. ISCII → Indian standard code for information interchange
5. unicode → local languages

⇒ Character set = 26 uppercase letters, 26 lower case letters, 0 to 9 digits and special characters are used in computer

⇒ All these character set are denoted through numbers only

1. BCD:

⇒ This encoding system is not in the practise ~~to~~ right now.
This 2^6 bit encoding system

⇒ can handle $2^6 = 64$ characters only

2. ASCII (united states)

⇒ can handle english characters only ($2^7 = 128$ characters)

⇒ Most of the computers use this system

⇒ Each character has individual numbers

⇒ New version ($2^8 = 256$ characters i.e., 0 to 255 unique numbers)

⇒ ASCII code is equivalent to the uppercase letter 'A' is 65

⇒ Binary representation of ASCII (7 bit) value 1000001

" " " " (8 bit) " 01000001

3. ~~ISCII~~ EBCDIC

⇒ Similar to ASCII code with 8 bit representation

⇒ Formulated by IBM

ISCII ⇒ Handles the character of Indian local languages

⇒ can handle 256 characters (2^8)

EBCDIC ⇒ The input code in ASCII can be converted to EBCDIC system and vice-versa

ASCII ⇒ Now, it is integrated with unicode

unicode::

⇒ It was generated to handle all the coding system of universal languages

⇒ 16 bit code and can handle 65536 characters

Computer Organization vs computer Architecture:

Computer organization	computer Architecture
* Computer organization refers to the operational units and their interconnections that realize the architectural specifications	* Computer Architecture refers to those parameters of a compu System that are visible to a programmer or those param -rs that have a direct impa on the logical execution of a program.
* Examples of computer organiza -tional attributes include those hardware details transparent to the programmer	* Examples of Architecture inc instruction set, the no. of used to represent differe datatypes, I/o mechanism etc.

⇒ can handle 256 characters (2^8)

EBCDIC ⇒ The input code in ASCII can be converted to EBCDIC system and vice-versa

ASCII ⇒ Now, it is integrated with unicode

unicode::

⇒ It was generated to handle all the coding system of universal languages

⇒ 16 bit code and can handle 65536 characters

Computer Organization vs Computer Architecture:

Computer organization	Computer Architecture
* Computer organization refers to the operational units and their interconnections that realize the architectural specifications	* 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.
* Examples of computer organizational attributes include those hardware details transparent to the programmer	* Examples of Architecture includes instruction set, the no. of bits used to represent different datatypes, I/O mechanisms etc.