ECAP268   COMP. SYSTEM ARCHITECTURE

CH:-1:- BINARY SYSTEM

Number Systems:- It is the system of naming the numbers

- It is the mathematical notation for representing no. of a given set by using digits or other symbols in a consistent manner.
- A no. can be represented differently in diff. number system.
  e.g:- the two numbers $(2A)_{16}$ & $(52)_8$ both refer to the same quantity $(42)_{10}$ but the representations are different.

⇒ Value of any digit:- can be determined by:-
- The digit
- Its position in the no.
- The base of the number system

Types of Number Systems:- There are 4 most common N/s :-

i) Decimal N/s:- A no. is represented as a string of digits.

- ~~Decimo~~ These are positional no. that have a base or radix of 10 or Dec or simply D.
- In a dec. no. there are ten such digits that may be used, ranging in value from 0 to 9.
- Number's value = a weighted sum of the digits.
  Number's value = digit $\times 10^{k}$ + digit $\times 10^{k}$
  where k = (position number-1)

e.g:- $1234_{10} = 1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0$
$= 1000 + 200 + 30 + 4$
$= 1234_{10}$

$986_D = 9\times10^2 + 8\times10^1 + 6\times10^0$
$= 900 + 80 + 6$
$= 986_D$

ii) Binary N/s:-

- The term comp. numbering formats refers to the schemes implemented in digital comp. & calculators to ~~be~~ represent numbers.
- Base is 2 or 'b' or 'B' or 'Bin'
- Two symbols : 0 & 1
- Each bit in the no. is weighted by the power of 2.
- These no. are made of binary digits.
- Binary Digits (BITs) can be represented electronically:

0 (no signal)

1 (signal)

- All the info. in the digital comp. is represented as bit patterns.
- 01010101 – bit pattern
- A unit of 4 bits – a nibble (or nybble)
  - ,, ,, ,, 8 ,, – a byte or an octet
  - ,, ,, ,, 16 bits – a word

Representation of Bit Pattern:-

e.g:- 0101 0101

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

There are 8 bits in the above table.
Bit 0 is called Least Significant Bit (LSB)
Bit 7 is called Most Significant Bit (MSB)

Representation of Unique States :-

- A single bit can ~~be~~ represent 2 unique states : 0 1.
- ∴, if you take two bits, you can use them to represent 4 unique states : 00, 01, 10 & 11
- If 3 bits, 8 unique states :- 000, 001, 010, 100, 011, 101, 110, 111
- With every bit we add, we double the no. of states we can represent. ∴, the expression for the no. of states with $\boxed{n}$ bits is $\boxed{2^n}$

iii) Octal N/S :- Base is 8 or 'O' or 'Oct'
- 8 symbols (0 to 7).
- Each no. is weighted by the power of 8.

iv) Hexadecimal N/S :- Base is 16 or 'H' or 'Hex'
- 16 symbols are used: 10 digits & 6 letters (0 to 9 & A to F)
  {1, 2, 3, 4, 5, 6, 7, 8, 9, A = 10, B = 11, C = 12, D = 13, E = 14, F = 15}
- E.g:- AB12, 876F, FFFF etc.
- Each symbol is weighted by the power of 16

Number Conservsions :-

⇒ Decimal to Binary :-
- Divide the no. by 2
- Get the int. quotient for the next iteration
- ,, ,, remainder for the binary digit.
- Repeat the steps until the quotient becomes 0.

Example :- $(41)_{10}$ ~~41/2~~

	Q	R
2	41	
2	20	1
2	10	0
2	5	0
2	2	1
2	1	0
2	0	1

↑ ⇒ $(101001)_2$ Ans

$(35)_{10} \Rightarrow$

	Q	R
2	35	
2	17	1
2	8	1
2	4	0
2	2	0
2	1	0
	0	1

$\uparrow (100011)_2$

$(30)_{10} \Rightarrow$

	Q	R
2	30	
2	15	0
2	7	1
2	3	1
2	1	1
	0	1

$\uparrow (11110)_2$

⇒ Decimal to Octal:-

- Divide the no. by 8
- Get the int. quotient for the next iteration.
- " " remainder for the binary digit.
- Repeat the steps until the quotient becomes 0

Example:- $(153)_{10} \Rightarrow$

	Q	R
8	153	
8	19	1
8	2	3
	0	2

$\uparrow \Rightarrow (231)_8$

$(670)_{10} \Rightarrow$

	Q	R
8	670	
8	83	6
8	10	3
8	1	2
	0	1

$\uparrow (1236)_8$

⇒ Decimal to Hexadecimal :- Divide the no. by 16 & same process.

Example: $(4735)_{10} \Rightarrow$

	Q	R
16	4735	
16	295	15(F)
16	18	7
16	1	2
	0	1

$\uparrow (127F)_{16}$

$(2020)_{10} \Rightarrow$

	Q	R
16	2020	
16	126	4
16	7	14 (E)
	0	7

$\uparrow (7E4)_{16}$

⇒ Binary to Decimal:-

- Multiply each bit with the power of 2 from (LSB to MSB) (0 to n)
- Add all the product values.

Example:- $(11001010)_2$

1 1 0 0 1 0 1 0

$0 \times 2^0 = 0$
$1 \times 2^1 = 2$
$0 \times 2^2 = 0$
$1 \times 2^3 = 8$
$0 \times 2^4 = 0$
$0 \times 2^5 = 0$
$1 \times 2^6 = 64$
$1 \times 2^7 = 128$

$\Rightarrow 2 + 8 + 64 + 128 = (202)_{10}$

Another method:- Take power of 2 in right hand manner:-

128	64	32	16	8	4	2	1	
1	1	0	0	1	0	1	0	– The no.

⇒ At last Add all where is 1

→ Binary to Octal:- Same process but at last divide the no. by 8.

Example:- $(111110)_2 \Rightarrow 2+4+8+16+32 = (62)_{10}$ & then repeat the same process of as of decimal to Octal.

32 16 8 4 2 1

$$\Rightarrow \begin{array}{r|c|c} 8 & 62 & \\ \hline 8 & 7 & 6 \\ \hline & 0 & 7 \end{array} \uparrow \Rightarrow (76)_8$$

$(1011001)_2 = 1+8+16+64 = (90)_{10}$

$$= \begin{array}{r|c|c} 8 & 90 & \\ \hline 8 & 11 & 2 \\ \hline 8 & 1 & 3 \\ \hline & 0 & 1 \end{array} \uparrow = (132)_8$$

→ Binary to Hexadecimal:- Same as Binary to Octal but it will be 16 in place of 8.

Example:- $(111110)_2 = (62)_{10}$

$$= \begin{array}{r|c|c} 16 & 62 & \\ \hline 16 & 3 & 14 = (E) \\ \hline & 0 & 3 \end{array} \uparrow \; (3E)_{16}$$

→ Octal to Decimal:-

- Multiply each bit with power of 8 (from LSB to MSB) (0 to n)
- Add all the products.

Example:- $(345)_8 = 3\times8^2 + 4\times8^1 + 5\times8^0 = 192+32+5 = (229)_{10}$

→ Octal to Binary:- Same process as above then divide the decimal no. by 2 until the quotient remains 0.

Example:- $(345)_8 = (229)_{10}$

$$= \begin{array}{r|c|c} 2 & 229 & \\ \hline 2 & 114 & 1 \\ \hline 2 & 57 & 0 \\ \hline 2 & 28 & 1 \\ \hline 2 & 14 & 0 \\ \hline 2 & 7 & 0 \\ \hline 2 & 3 & 1 \\ \hline 2 & 1 & 1 \\ \hline & 0 & 1 \end{array} \uparrow \; (11100101)_2$$

→ Octal to Hexadecimal:- Same as above but 16 instead of 8.

Example:- $(345)_8 = (229)_{10}$

$$= \begin{array}{r|c|c} 16 & 229 & \\ \hline 16 & 14 & 5 \\ \hline & 0 & 14 \end{array} \Rightarrow (E5)_{16}$$

→ Hexadecimal to ~~Binary~~ Decimal:-

- Multiply each bit with power of 16 (from LSB to MSB) (0 to n)
- Add all the products.

Example:- $(7DE)_{16} = 7\times16^2 + (D=13)\times16^1 + (E=14)\times16^0$

$= 1792 + 208 + 14 = (2014)_{10}$

⇒ Hexadecimal to Binary :- Same as above then divide the dec. no. by 2 until the quoteint is 0.

Example :- $(7DE)_{16} = (2014)_{10}$ =

2	2014	
2	1007	0
2	503	1
2	251	1
2	125	1
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
2	0	1

↑ $(11111011110)_2$

⇒ Hexadecimal to ~~Dec~~ Octal :-

Same as above then divide the decimal no. by 8 until the quoteint is 0.

Example :- $(7DE)_{16} = (2014)_{10}$ =

8	2014	
8	251	6
8	31	3
8	3	7
	0	3

↑ $(3736)_8$

Complement :-

$U = \{a, e, i, o, u\}$
$A = \{a, e, i\}$] - Two sets

⇒ Complement of A i.e., $A' = U - A = \{o, u\}$

When we are taking the complement, just involve the elements which are not present in the set.

In boolean algebra, the values can be 0 or 1. So, $U = \{0, 1\}$

If $A = \{0\}$, then the complement will be $A' = \{1\}$

$A = \{1\}$, ,, ,, ,, ,, ,, $A' = \{0\}$

There are 2 types of complements for each r-based system.

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

When we substitute the value of base here, these are referred to as:-

- 2's complement & 1's complement (for Binary no.)
- 10's ,, & 9's ,, (for decimal no.)

⇒ Diminished Radix (r-1)'s :-

Given a no. N in base r having n digits, the (r-1)'s complement of $\boxed{N = (r^n - 1) - N}$

For decimal = $N = (10^n - 1) - N$ = 9's complement

10^n represents a no. that consists of a single 1 followed by n 0s.
$10^n - 1$ is a no. represented by n 9s.
For e.g:- if $n=4$, we have $10^4 = 10{,}000$ & $10{,}000 - 1 = 9{,}999$

For binary no. of 1's complement = $\boxed{N = (2^n - 1) - N}$
for e.g:- if $n=2$ then $2^4 = (10000)_2$ & $2^4 - 1 = (1111)_2$

Example:- Q.) 9's complement of 546700
$999999 - 546700 = 453299$

Q.) 1's complement of 1011000
$1111111 - 1011000 = 0100111$
Another method :- Just reverse the given no.
1011000 (given no.)
0100111 1's complement

→ Radix (r's) :- N in base r is defined as
$r^n - N$ for $N \neq 0$
0 for $N = 0$
The r's complement can also be obtained by adding 1 to the (r-1)'s complement
∴ $\boxed{r^n - N = [(r^n - 1) - N] + 1}$

Example:- Q.) 10's complement of 546700
First 9's complement = $999999 - 546700 = 453299$
⇒ $453299 + 1 = 453300$

Q.) 2's complement of 1011000
1's complement = 0100111
2's complement = 0100111
+ 1
0101000

→ Substraction of Complements :-
Q.) Using 10's complement, substract 72532 − 3250
~~M 72532~~ first of all both no. should be of same digit
Here, it is not.
Our M is 72532 & N is 3250
So, in N we will add 0 in front of 3 that is 03250

Now, we will get the 10's complement of 03250 that is
99999 - 03250 = 96749 = 9's complement
96749 + 1 = 96750 = 10's complement of N

Sum of M & 10's complement of N = 72532
+ 96750
Carry (1) 69282

Discard end carry here :- 69282 Ans

Q.) Using 10's complement, substract 3250 - 72532
(M) (N)

10's complement of N :- 99999 - 72532 = 27467 + 1 = 27468

Sum of M & 10's complement of N = 03250
+ 27468
30718

There is no end carry
Answer = -(10's complement of 30718)
= 99999 - 30718 = 69281 + 1 = 69282
= -69282

Q.) Using 2's complement substract 1010100 - 1000011
(M) (N)

2's complement of N = 0111100 + 1 = 0111101

Sum of M & 2's complement of N = 1010100
0111101
10010001

Discard the carry :- 0010001

Q.) Using 2's complement substract 1000011 - 1010100
(M) (N)

2's complement of N = 0101011 + 1 = 0101100

Sum of M & 2's complement of N = 1000011
0101100
1101111

There is no end carry.
So, Answer is -(2's complement of 1101111)
-(0010000 + 1) = -0010001

Representation :-

- +ve integers, including 0 can be represented as unsigned no.
However, to represent -ve integers we need a notation for -ve values.
- Bcoz of the H/w limitations comp. must represent everything with 1's & 0's, including the sign of the no.

As a consequence, it is customary to represent the sign with bit placed in the leftmost position of the no. The convention is to make the sign bit 0 for (+ve) & 1 for (-ve).

In addition to the sign, a no. may have a decimal (or binary) point. The position of binary point is needed to represent the fractions, int, or mixed int. fraction no.

There are 2 ways to ~~re~~ specify the position of a binary point in a register.

- By giving a fixed position
- By employing a floating point representation.

⇒ Fixed Point Representation:-

This method assumes that the binary point is always fixed in one position. The 2 positions most widely used are:-

i) The binary point in the extreme left of the register to make the stored no. a fraction.

ii) The binary point in the extreme right of the register to make the stored no. a int

In either case, the binary point is not present, but its presence is assumed from a fact that no. stored in the register is treated as a fraction or an int.

When the int binary no. is +ve, the sign is represented by 0 & the magnitude by a +ve binary no.

When -ve, the sign is represented by 1 but the rest of the no. may be represented by one of 3 possible ways

i) Signed magnitude representation.
ii) Signed 1's complement ,,
iii) ,, 2's ,, ,,

Example:- Consider the signed no. 14 stored in 14-bit register

00001110 = +14

for -14 :- i) 10001110 - Signed magnitude
ii) 11110001 - 1's complement
iii) 11110010 - 2's complement

⇒ Floating Point Representation:-

It uses a 2nd register to store a no. that designates the position of the decimal point in the 1st register

The floating point representation of a no. has 2 parts:-

i) A signed, fixed point no. called the $\boxed{\text{Mantissa}}$ 1st part

ii) The position of the decimal (or binary) point called $\boxed{\text{exponent}}$ 2nd part

Mantissa may be a fraction or an integer.

for e.g:- +6132.789

fraction +0.6132~~4~~789

exponent +04

The value of exponent indicates the actual position of the decimal point. This can also be represented as $+0.6132789 * 10^{4}$.