MADE EASY

ELECTRICAL ENGINEERING

Computer Fundamental

By.Sagar Sir

COMPUTER FUNDAMENTALS

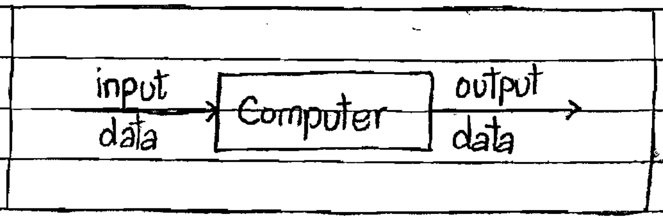
■ SYLLABUS

1. Data Representation
2. Computer Architecture
3. Computer Organization
4. Operating System
5. Networking
6. Programming Elements

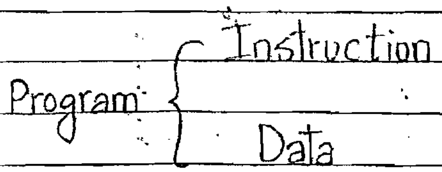
■ KEYWORDS

1. Computer

It is a computational machine used to process data under the control of a program application initiated by the user.



2. Program



3. Instruction

It is a binary code designed inside the processor to perform some task.

Binary code - Bind with - Operation

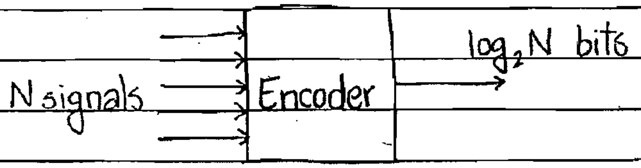
Eg. CPU-X supports 8 different operations, then
opcode size = $\log_2 8$ bits
{binary code} = 3-bit

say, 000 → multiplication,
001 → subtraction
101 → addition & so-on

These opcodes are decided by the designed

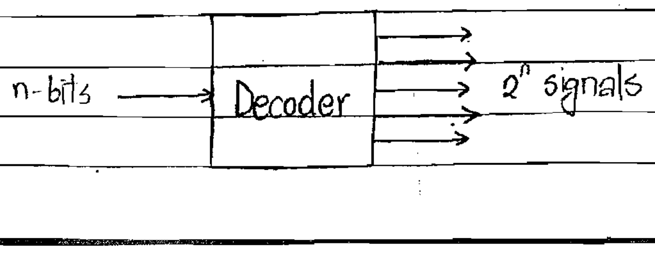
4. Encoding

In this process, N signals are represented using $\log_2 N$ bits



5. Decoding

In this process, an n-bit decoder produces 2^n o/p signals.



6. Data

Data is a binary code which is associated with a value based on the data format.

Binary code - Bind with - value

Eg. $(101)_2 \xrightarrow{\text{unsigned}} 5$

$\xrightarrow{\text{sign mag.}} -1$

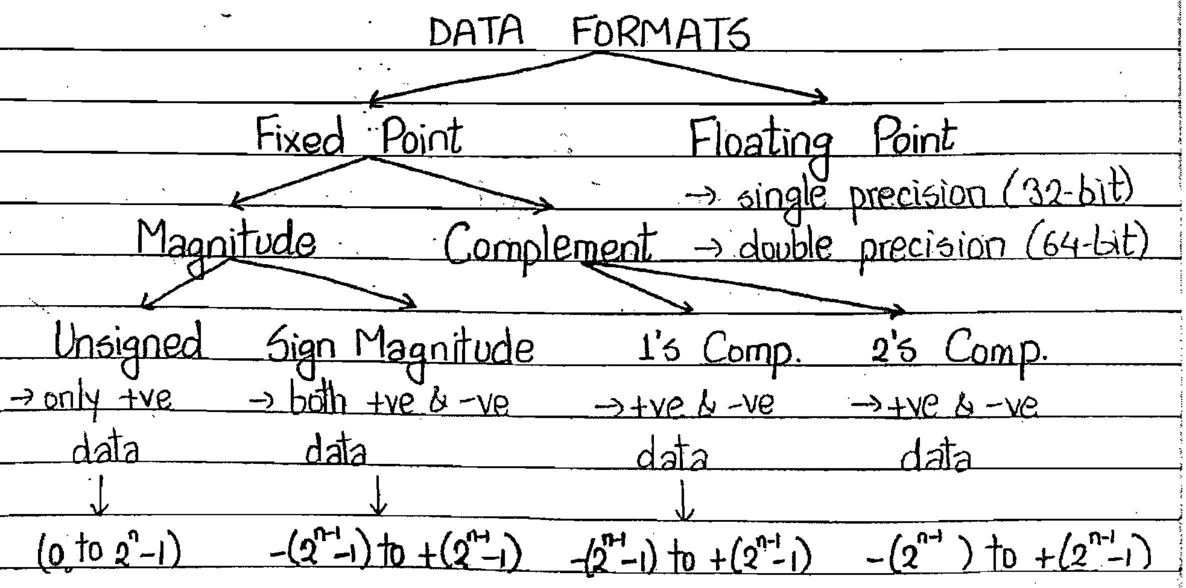
$\xrightarrow{\text{2 comp.}} -3$

$\xrightarrow{\text{1's comp.}} -2$

$\xrightarrow{\text{Floating pt.}}$ fraction

■ DATA REPRESENTATION

In the computer system, data is always represented in binary. Different formats are used in the Computer systems to defined the data, described as follows:



DATA FORMATS

Fixed Point

Floating Point
→ single precision (32-bit)
→ double precision (64-bit)

Magnitude

Complement

Unsigned
→ only +ve data
↓
$(0 \text{ to } 2^n-1)$

Sign Magnitude
→ both +ve & -ve data
↓
$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$

1's Comp.
→ +ve & -ve data
↓
$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$

2's Comp.
→ +ve & -ve data
$-(2^{n-1}) \text{ to } +(2^{n-1}-1)$

□ Fixed Point Data

4-Bit Code	Unsigned	Sign Magnitude	1's Comp.	2's Comp.
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

- Consider a hypothetical system with a word length of 12 bit. What is the range of a 2's complement data possible in this system?

→ $n = 12$ → range : -2^{11} to $2^{11}-1$

: -2048 to $+2047$

- What is the range of a signed data?

→ Signed ≠ Sign Magnitude.

Signed data { Sign Magnitude
1's complement
2's complement

Signed data:
- Sign Magnitude
- 1's complement
- 2's complement

when a particular format isn't specified, the default format i.e. 2's complement is taken.

∴ range : -128 to +127

- What is the range of a data possible to process on a 8-bit CPU?

→ Since the data type itself isn't specified, so we consider default type i.e. unsigned data.

∴ range : 0 to 255

* Sign Extension

This concept is used in the signed data, to preserve the sign of a data when the data is stored in a large storage space.
Sign Extension means replication of MSB of data

Eg. -7 : 1001 → store in a 8-bit space

$+9 \leftarrow \underbrace{0000}_{\text{0's padding (zero's padding)}}1001$ X

$\underbrace{1111}_{\text{MSB replication (sign extension)}}1001 \xrightarrow{\text{2's comp}} -7$ ✓

Note: 0's padding is used in ~~Si~~ Unsigned data &
Sign extension is used in Signed data

□ Bit Overflow

Eg. 4 bit data

```
   1111    +15
 + 1111    +15
 (1) 1110  +30
 ↳ bit overflow {carry}
```

→ Carry Flag holds the range exceeding condition of an unsigned data, represented by Cy

n-bit + n-bit = (n+1)-bit
↳ carry flag is this additional bit

if Cy=1, then carry is present i.e. range has been exceeded

if Cy=0, no carry is present.

Eg. 4-bit data

```
    1000    8
    1001   +9
  (1)0001  17
  ↳Cy=1 ⇒ out of range
```

□ Multiplication

→ In the multiplication process, two steps are performed.
i) generation of partial products
ii) summation of partial products

→ Partial products are generated based on multiplier bits i.e. when multiplier bit is 1, multiplicand itself comes out as partial product else partial product is 0.

→ After the generation of p.p., provide the summation to produce the final product.

```
           multiplicand   multiplier
E.g.           1111   *   1111
              10 1
               1111  )
            011       |
              1111    |  partial
           011        |  products
             1111     |
         010          |
            1111     )
        ----------
        11100001 → final product = 225
        \______/
          8-bit
```

Note: (n bit) * (n bit) = 2n bits
→ register "pair" is used to store final product

· Consider the following multiplication

$$(10w1z)_2 * (15)_{10} = (y01011001)_2$$

what are the values of w, y & z ?

→ $(10w1z)_2 * (1111)_2 = (Y0101 1001)_2$

```
     2 2 1
     10w1z
   2 10w1z          1+w+1+z = 0
   1 10w1z          1+w+1+1 = 0
     10w1z                ⇒ w=1   → cy = 2
  1010110 0z
```

$\underline{y=1}$ ↙ $\hookrightarrow \underline{z=1}$, else $(4w+z+18)*15 = (256y+89)$

$\Rightarrow (w,y,z) \equiv (1,1,1)$

□ Division

→ In the division procedure, bits are scanned from MSB to LSB in bit-wise sequence.

→ After scanning, dividend value is compared with divisor value.

→ If Dividend ≥ Divisor, then put '1' in quotient & subtract the divisor from the dividend & scan the next bit.

→ If Dividend < Divisor, then put '0' in quotient & scan the next bit

→ Continue the process till all the bits are serviced.

Eg. 00000111 ÷ 0010