Unit:-02 Register Transfer and Micro-operations

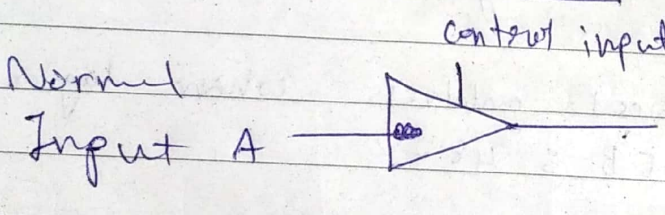
- R1 : processor register
- MAR : Memory Address Register
- PC : Program counter
- IR : Instruction Register
- SR : Status Register.

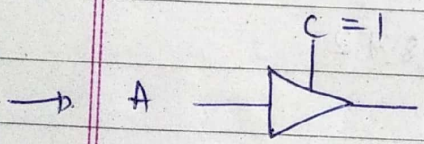
- Conditional transfer $P : R_2 \leftarrow R_1$.
  it happens only if $P = 1$.
- A common Bus system is a scheme for transferring information between registers in a multiple-register configuration.

* Three state bus buffers :

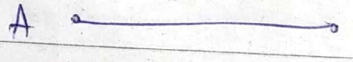
logic-0, logic-1, high impedance (Hi-Z).

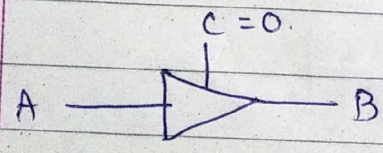
three states



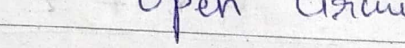


Buffer.





Open Circuit



** Data being read/wrote is called memory word. (called M).

→ Ex:- M[0016] : the memory contents at address 0x0016.

Read : DR ← M[0016] (or) DR ← M[AR]

Write : M[0016] ← DR

DR: Data register; AR: Address register

* Arithmetic micro-operations :-

Addition : $R_3 \leftarrow R_1 + R_2$

Subtraction : $R_3 \leftarrow R_1 - R_2$

$R_3 \leftarrow R1 + \overline{R2} + 1$ ($\overline{R2} + 1$: 2's complement)

One's complement : $R2 \leftarrow \overline{R2}$

2's " : $R2 \leftarrow \overline{R2} + 1$

Increment : $R2 \leftarrow R2 + 1$

Decrement : $R2 \leftarrow R2 - 1$

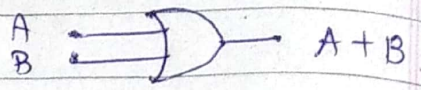
* Arithmetic circuit :-

→ An arithmetic circuit performs seven distinct arithmetic operations and the basic component of it is the parallel adder.

→ Output of a binary adder $D = A + Y + C_{in}$

* Logic Micro-operations :-

(1) OR : $\vee$, $+$



→ Ex :-

$$\begin{array}{r} 1001110_2 \\ 1101011_2 \\ \hline 1101111_2 \end{array}$$

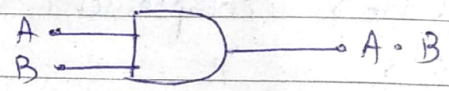
→ P + Q : R1 ← $R_2 + R_3$

OR ADD

→ R4 ← R5 $\vee$ R6

OR

(2) AND : $\wedge$, $\cdot$



→ Ex :-

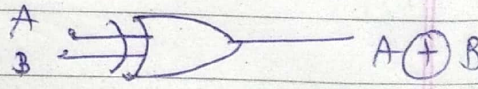
$$\begin{array}{r} 1001110_2 \\ 1010110 \\ \hline 0000110_2 \end{array}$$

(3) NOT : $\overline{\ }$ Complement. A — $\bar{A}$

→ Ex :- A 1001111010_2

$\therefore \bar{A} = 0110000101_2$

(4) XOR : $\oplus$



→ Ex:- $100110_2 \oplus 1010110_2$

XOR
↓
same → 2 or 4

$$\begin{array}{r} 1\,0\,0\,1\,1\,0_2 \\ \oplus\ 1\,0\,1\,0\,1\,1\,0_2 \\ \hline \text{Ans: } 1\,1\,1\,0\,0\,0\,0_2 \end{array}$$

* Other logic micro-operations :-

(1) selective set operation :
→ basically OR operation.

$$\begin{array}{r} 0\,1\,0\,0_2 \\ 1\,0\,0\,0_2 \\ \hline \text{Ans: } 1\,1\,0\,0_2 \end{array}$$

(2) selective complement (toggling) operation :
→ basically XOR operation.

XOR
↓
same → 0.

$$\begin{array}{r} 0\,0\,0\,1_2 \\ 1\,0\,0\,0_2 \\ \hline \text{Ans: } 1\,0\,0\,1_2 \end{array}$$

(3) Insert operation :
→ Step 1 : mask the desired bits
→ Step 2 : OR them with the desired value.

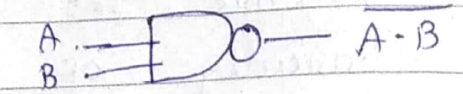
→ Ex:- R1 = 0110 1010 and we want to replace the left most 4 bits with 1001.

∴ step 1 : 0 1 1 0 1 0 1 0
∧ 0 0 0 0 1 1 1 1
0 0 0 0 1 0 1 0

↓

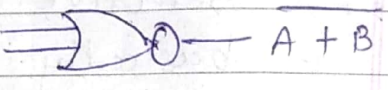
step 2 : 0 0 0 0 1 0 1 0
∨ 1 0 0 1 0 0 0 0
R1 → 1 0 0 1 1 0 1 0

(4) NAND : ∧, −



→ Ex : 1 0 0 1 1 0
∧ 1 0 1 0 1 1
1 0 0 0 1 0
NOT 0 1 1 1 0 1 ← Ans :

(5) NOR : ∨, −



→ Ex : $1\ 0\ 0\ 1\ 1\ 0_2$
$1\ 0\ 1\ 0\ 1\ 1\ 0_2$
X $1\ 1\ 1\ 0\ 1\ 1\ 0_2$
NOT $0\ 0\ 0\ 1\ 0\ 0\ 1_2$ ← Ans :

(6) Set (Present) operation :

→ Force all bits into 1 by ORing them with a value in which all it's bits are being assigned to logic - 1.

→ Ex : $1\ 0\ 0\ 1\ 1\ 0_2 \vee 1\ 1\ 1\ 1\ 1\ 1_2 = 1\ 1\ 1\ 1\ 1\ 1_2$

(7) Clear (Reset) operation :

→ Force all bits into 0's by ANDing them with a value in which all bits are being assigned to logic-0.

→ Ex:- $100110_2 \wedge 000000_2 = 000000_2$

~~* Shift micro-operations :-~~

(8) Selective clear operation :

→ Ex:- A : 1 0 1 0 $\quad A \leftarrow A \cdot \overline{B}$

B : 1 1 0 0

0 0 1 0

put 0 in A's bit where there are corresponding 1's in B.

(9) Mask operation : AND

→ Ex:- 1 0 1 0 $\quad A \leftarrow A \wedge B$

1 1 0 0

1 0 0 0

(10) Clear operation : $A \leftarrow A \oplus B$

→ Ex:- A : 1 1 0 0 XOR

B : 1 0 1 0

0 1 1 0

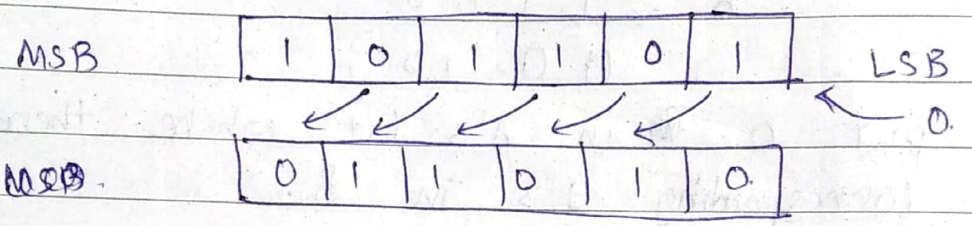
* Shift Operations :-

→ Used for serial transfer of data.

1) Logical

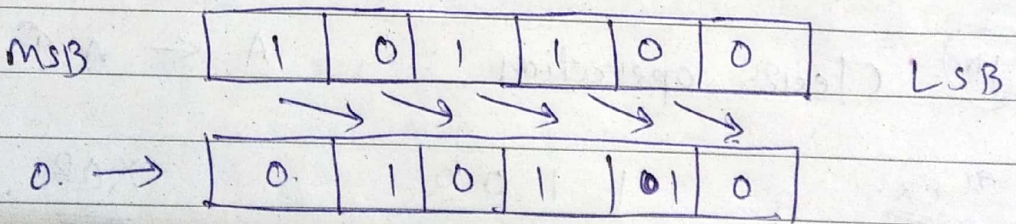
** Logical shift left :-

→ one position moves each bit to the left one by one.
→ The empty LSB is filled with zero and the MSB (most significant bit) is rejected.



** Logical shift right :-

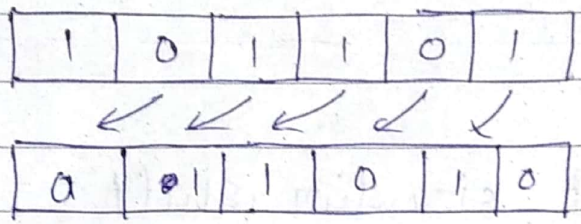
→ one position moves each bit to the right one by one.
→ The empty MSB is filled with zero and the LSB is rejected.



2) Arithmetic

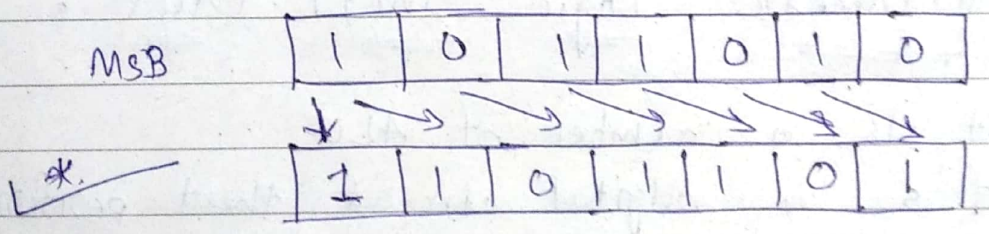
** Left arithmetic shift :-

→ one position moves each bit to the left one by one.
→ The empty LSB is filled with zero and MSB is rejected



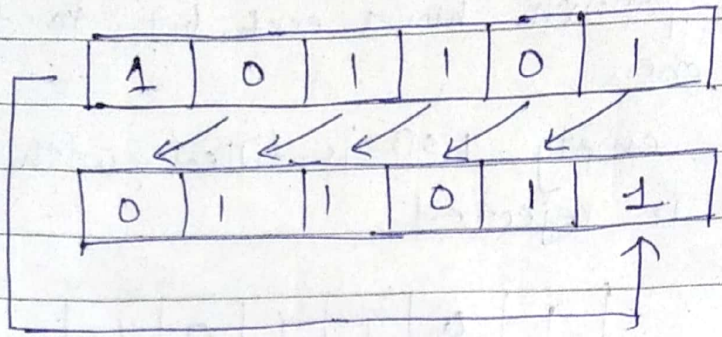
** Right arithmetic shift :-

→ one position moves each bit to the right one by one and LSB is rejected and MSB is filled with the value of the previous MSB.

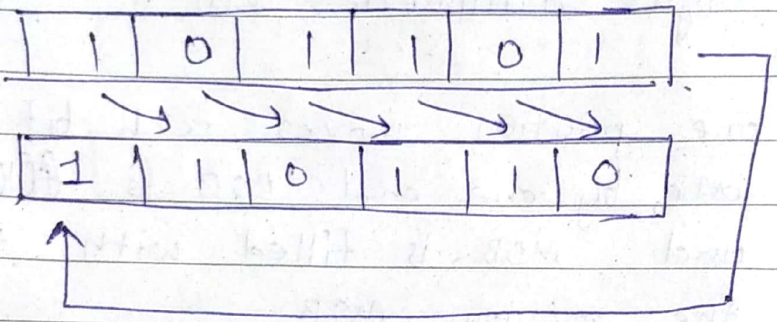


3.) Circular

** Left circular shift :-



** Right circular shift :-



* Arithmetic Logic Shift Unit :-

→ It is a member of ALU.
→ It's a digital circuit that performs logical, arithmetic and shift operations.

→ Rather than having individual registers calculating the micro operations directly, the computer deploys a number of storage registers which is connected to a common

operational unit known as an arithmetic logic unit (ALU)