KREATRYX

KNotes

DIGITAL CIRCUITS

www.kreatryx.com

Contents

Manual for K-Notes 2
Number Systems and Boolean Algebra 3
Combinational Logic Circuits 12
Sequential Logic Circuits 15
A/D and D/A Converters 25
Microprocessor 31

Manual for K-Notes

Why K-Notes?

Towards the end of preparation, a student has lost the time to revise all the chapters from his / her class notes / standard text books. This is the reason why K-Notes is specifically intended for Quick Revision and should not be considered as comprehensive study material.

What are K-Notes?

A 40 page or less notebook for each subject which contains all concepts covered in GATE Curriculum in a concise manner to aid a student in final stages of his/her preparation. It is highly useful for both the students as well as working professionals who are preparing for GATE as it comes handy while traveling long distances.

When do I start using K-Notes?

It is highly recommended to use K-Notes in the last 2 months before GATE Exam (November end onwards).

How do I use K-Notes?

Once you finish the entire K-Notes for a particular subject, you should practice the respective Subject Test / Mixed Question Bag containing questions from all the Chapters to make best use of it.

Number Systems and Boolean Algebra

- Decimal : Radix = 10; Symbols = (0, 1, 2, 3...........9)
- Binary : Radix = 2; Symbols = (0, 1)
- Hexadecimal : Radix = 16; Symbols = (0, 1, 2..............., 9, A, B,..............., F)
- Octal : Radix = 8; Symbols = (0, 1, 2, ..., 7)

For radix N, following digits are possible

(0, 1, 2, ..., N-1)

- To convert a number from radix 'x' to bare 10 or decimal.

Eg. $(136)_x \rightarrow (?)_{10} = 1.x^2 + 3.x + 6.x^0 = \left(x^2 + 3x + 6\right)$

Complimentary Number Representation

A – B = A + (- B) $= A + (\text{compliment of} + B)$

<u>For a base – r system</u>

$(r-1)'\text{s compliment} = r^n - r^{-m} - N$

$r'\text{s compliment} = r^n - N$

Where r = base

N = given number

n = no. of digits in integer part of N

m = no. of digits in decimal part of N

eg. For $(378.67)_{10}$

N = 378.67 ; m = 3 ; n = 2 ; r = 10

Boolean Algebra

- Compliment

  $0 \rightarrow 1$

  $1 \rightarrow 0$

  Represented as $A \rightarrow \bar{A}$

  And $\overline{(\bar{A})} = A$

- AND function

$0.0 = 0$	$A.A = A$
$0.1 = 0$	$A.1 = A$
$1.0 = 0$	$A.0 = 0$
$1.1 = 0$	$A.\bar{A} = 0$

- OR Function

$0 + 0 = 0$	$A + A = A$
$0 + 1 = 1$	$A + 1 = 1$
$1 + 0 = 1$	$A + 0 = A$
$1 + 1 = 1$	$A + \bar{A} = 1$

Laws of Boolean Algebra

1) Commutative Law

$$OR = A + A = B + A$$
$$AND = AB = BA$$
$$NAND = \bar{A}\bar{B} = \bar{B}\bar{A}$$

2) Associative Law

$$OR = (A + B) + C = A + (B + C)$$
$$AND = (A\,B)C = A(B\,C)$$

3) Consensus Law

$AB + \bar{A}C + BC = AB + \bar{A}C$

4) Distributive Law

$A.(B + C) = AB + AC$

Dual : Convert all $\begin{matrix} . \to + \\ + \to . \\ 1 \to 0 \\ 0 \to 1 \end{matrix}$

A + (BC) = (A+B)(A+C)

5) De – Morgan's Law

- NOR operation is same as bubbled AND

$\overline{A + B + C..................} = \bar{A}.\bar{B}.\bar{C}.............$

- NAND operation is same as bubbled OR

$\overline{A + B + C..................} = \bar{A} + \bar{B} + \bar{C}.............$

6) Transposition Law

$A.B + \bar{A}C = (A + C)(\bar{A} + B)$

Operator precedence

1) Parenthesis
2) NoT
3) AND
4) OR

$\downarrow$ Decreasing priority

Minterms, Maxterms & Properties

Minterm : It is a standard product term i.e. a product term which contains all variables of a given function either in normal form or compliment form.

Maxterm : it is standard sum term i.e. a sum term which contains all the variables of the function either in normal or compliment form.

F (A, B, C) = min terms	$F(A,B,C) = \text{max terms}$
$\bar{A}\bar{B}\bar{C} = m_0\ (0,0,0)$	$\bar{A}+\bar{B}+\bar{C} = M_7$
$\bar{A}\bar{B}C = m_1\ (0,0,1)$	$\bar{A}+\bar{B}+C = M_6$
$ABC = m_7\ (1,1,1)$	$A+B+C = M_0$

Properties

1) $n - \text{variable function} \rightarrow \{2^n \text{ minterms } \& \ 2^n \text{ maxterms}\}$

2) $M_j = \bar{m}_j \ \& \ m_j = \bar{M}_j$

3) $m_{iD} = M_{(2^n-1-i)}$; $M_{iD} = m_{(2^n-1-i)}$; D = indicates dual

4)

a) $\sum_{i=0}^{2^n-1} mi = 1$

b) $\prod_{j=0}^{2^n-1} m_j = 0$

Note : The output of XOR and XNOR gate contains half the total number of minterms.

Forms of Boolean function

1) Sum of product (SOP) form = DNF (Distinjunctive Normal Form)
2) Canonical SOP form = DCF (Disjunctive Canonical Form)
3) Product of sum (POS) form = CNF (Conjunctive Normal Form)
4) Canonical POS form = CCF (Conjunctive Canonical Form)

Eg. Convert $F(A,B,C) = \bar{A} + A\bar{C} + AB\bar{C}$ to Canonical SOP form :

$$F = \bar{A} + A\bar{C} + AB\bar{C}$$

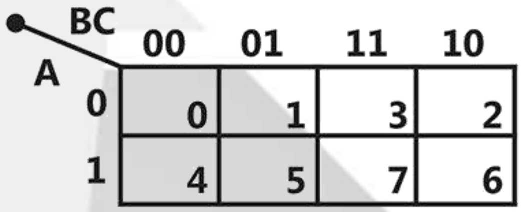
$$= \underset{m_0}{\bar{A}\bar{B}\bar{C}} + \underset{m_1}{\bar{A}\bar{B}C} + \underset{m_2}{\bar{A}B\bar{C}} + \underset{m_3}{\bar{A}BC} + \underset{m_4}{A\bar{B}\bar{C}} + \underset{m_6}{AB\bar{C}}$$

$F = \sum m(0,1,2,3,4,6) \rightarrow$ canonical SOP form

$= \pi M(5,7) \rightarrow$ canonical POS form

Karnaugh Map

<u>3 – variable K – map</u>

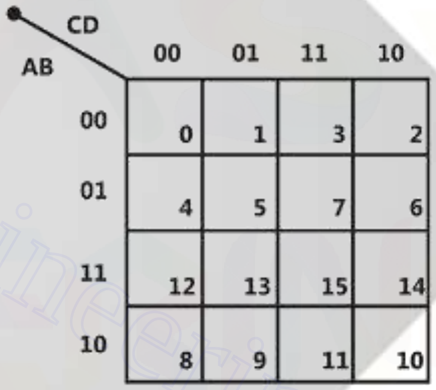


Octant $\rightarrow$ group of 8 minterms

Quad $\rightarrow$ group of 4 min terms

Pair $\rightarrow$ group of 2 min terms

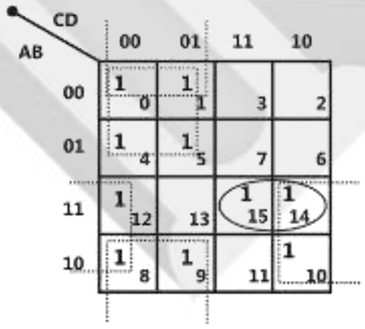
<u>4 – variable k – map</u>



All corners of k – map

(0, 2, 8, 10) $\rightarrow$ Quad

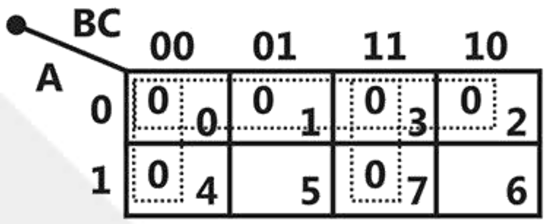
Eg. $F(A,B,C,D) = \sum m(0,1,4,5,6,8,9,10,12,14,15)$



$F = \bar{A}\bar{C} + AD + B\bar{D} + \bar{B}\bar{C} + ABC$

Eg. $F(A,B,C) = \pi M(0,1,2,3,4,7)$

$$F = A(B+C)(\overline{B}+\overline{C})$$



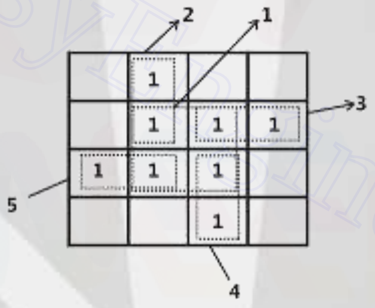
Implicant : it is the set of all adjacent min terms

Eg. Pair, quad, octants

Prime Implicant : It is an implicant which is not a subset of another implicant.

Essential PI (EPI) : It is a prime implicant which contains at least one min terms which is not covered by other prime implicant.

1) PI,Non PI
2) PI, EPI
3) PI, EPI
4) PI, EPI
5) PI, EPI

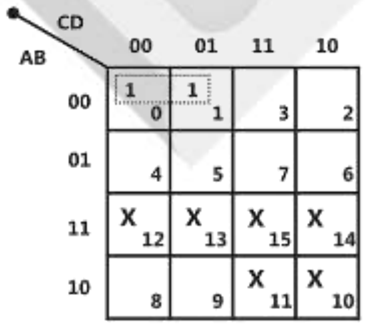


Don't care condition

In a digital system, for a non – occurring input, the output can be taken as either one or zero during simplification & it is called don't care condition.

Eg. $X(A,B,C,D) = \sum m(0,1) + d(10,11,12,13,14,15)$;

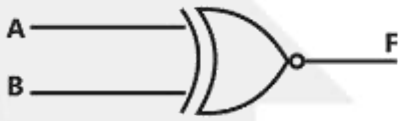
$X = \overline{A}\overline{B}\overline{C}$



Logic Gates

1) Equivalence Gate = Ex – NOR Gate

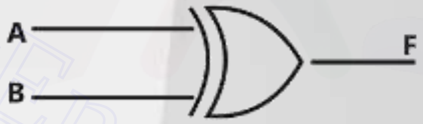
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1



$F = A \otimes B = AB + \overline{A}\overline{B}$

2) Staircase connection = Ex – OR Gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



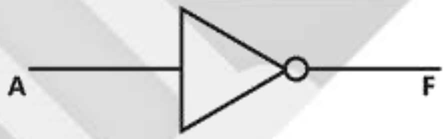
$F = A \oplus B = A\overline{B} + \overline{A}B$

- In Ex – OR, output = 1 if input has odd no. of 1's
- In Ex – NOR, output = 1, if input has even no. of 1's

3) Inverter

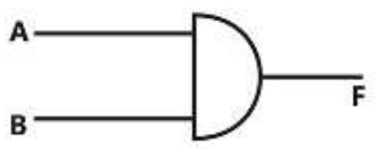
A	F
0	1
1	0

$F = \overline{A}$



4) AND GATE

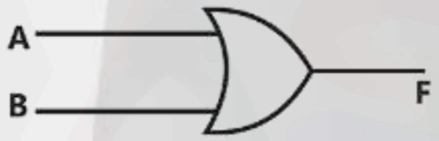
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



F = A . B

5) OR GATE

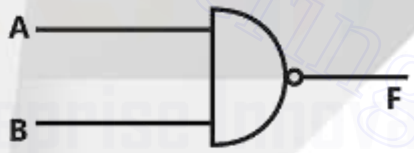
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



F = A + B

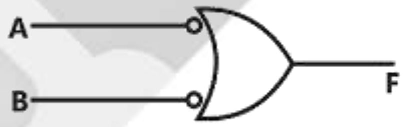
6) NAND GATE

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



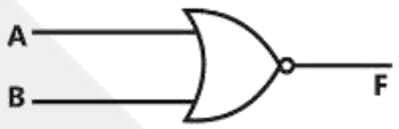
F = $\overline{A}\overline{B}$

This gate is equivalent to



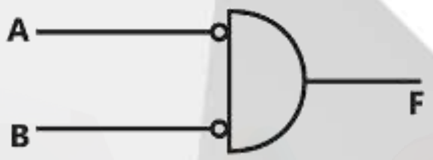
7) NOR GATE

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



$F = A + B$

This gate is equivalent to



CODES :-

1) Binary coded decimal code (BCD) :-

a) Each digit of decimal number is represented by binary equivalent.
b) It is 4 bit binary code.
c) eg. $(943)_{\text{decimal}} \rightarrow \overset{9}{1001}\ \ \overset{4}{0100}\ \ \overset{3}{0011}$

$(943)_{10} = (100101000011)_2$

2) Gray Code :-

a) Only one bit in the code group changes when going from one step to the next.
b) For 3-bit

$000 \rightarrow 001 \rightarrow 011 \rightarrow 010 \rightarrow 110 \rightarrow 111 \rightarrow 101 \rightarrow 100$

Combinational Logic Circuits

1) Half Adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

$S = A \oplus B$

$C = AB$

1 Half adder = 1 XOR Gate & 1 AND Gate

- To implement a half adder using NAND Gates, 5 NAND Gates are required.
- To implement a half adder using NOR Gates, 5 NOR Gates are required.

2) Half Subtractor

A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = A \oplus B$

$B = \bar{A}\,B$ = borrow

- To implement a half sub tractor 5 NAND or 5 NOR Gates are required.

3) Full Adder :-

A	B	C_i	S	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

C_i = Carry input

C_{i+1} = Carry Output

$S = A \oplus B \oplus C$

$C_{i+1} = AB + BC_i + AC_i$

- To implement full adder using NAND & NOR Gates 9 Gates are required.

4) Full Subtractor:-

A	B	b_i	D	b_{iH}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

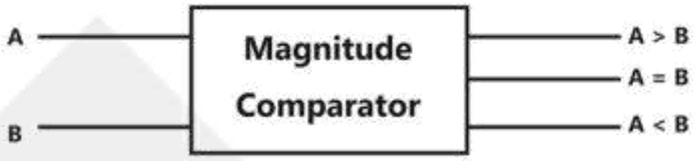
$D = A \oplus B \oplus b_i$

$b_{i+1} = \bar{A}B + Bb_i + \bar{A}b_i$

- To implement full sub tractor using NAND or NOR Gates 9 Gates are required.

Magnitude Comparator

For 2 bit Magnitude comparator



$A = A_1A_0 \quad ; \quad B = B_1B_0$

A_1	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

$$(A = B) = (A_1 \oplus B_1).(A_0 \oplus B_0)$$

$$(A > B) = A_1\bar{B}_1 + (A_1 \odot B_1)A_0\bar{B}_0$$

$$(A < B) = \bar{A}_1B_1 + (A_1 \odot B_1)\bar{A}_0B_0$$

Decoder

2 – 4 decoder

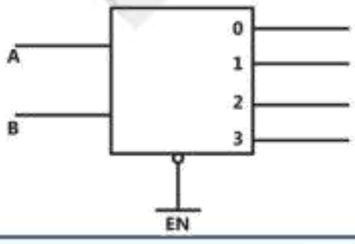
- Active high output

$0 = \bar{A}\,\bar{B}$

$1 = \bar{A}\,B$

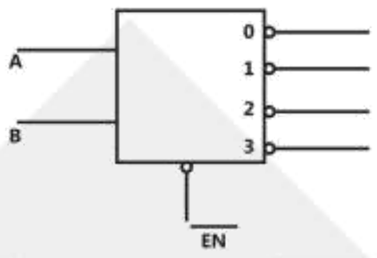
$2 = A\,\bar{B}$

$3 = A\,B$



- Active low output

  $0 = A + B$
  $1 = A + \overline{B}$
  $2 = \overline{A} + B$
  $3 = \overline{A} + \overline{B}$



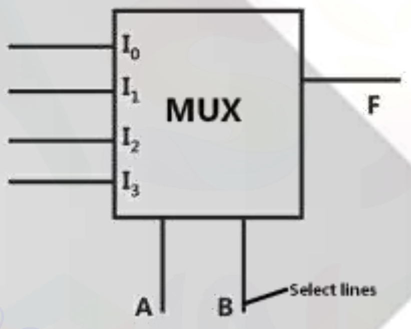
- Each output of a decoder with active high output represents a min term & hence it can be used to implement any SOP expression.
- Each output of a decoder with active low output represents a max term and hence can be used to implement any POS expression if AND Gate is used and SOP expression if NAND Gate is used.

Multiplexer

4 – 1 MUX

$F = \overline{A}\overline{B}I_0 + \overline{A}BI_1 + A\overline{B}I_2 + ABI_3$

$2^n - 1$ MUX requires n – select lines.

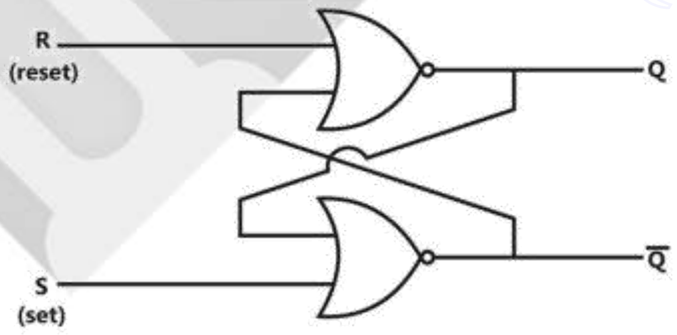


- A $2^n : 1$ MUX can be used to implement any SOP expression with (n+1) variable with n variables applied at select lines & $(n+1)^{th}$ variable & its complement & 1 & 0 serve as input to MUX.

Sequential Logic Circuits

1) SR Latch

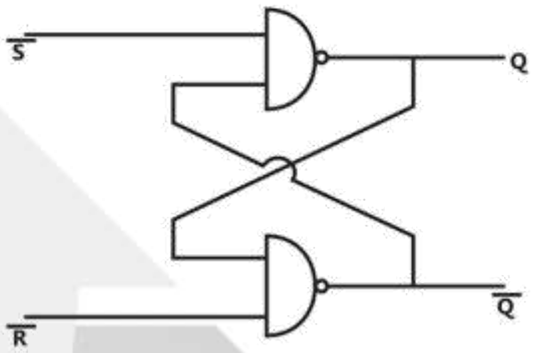
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	0



- S=1, R=1 and Q_{n+1}=0 is impractical state

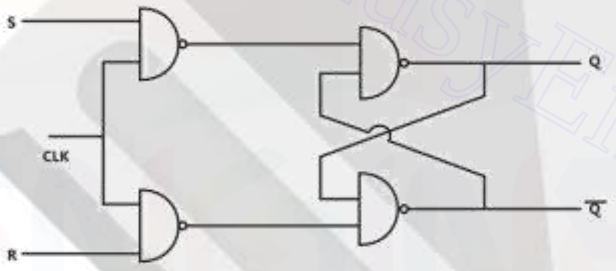
2) $\bar{S}\,\bar{R}$ Latch

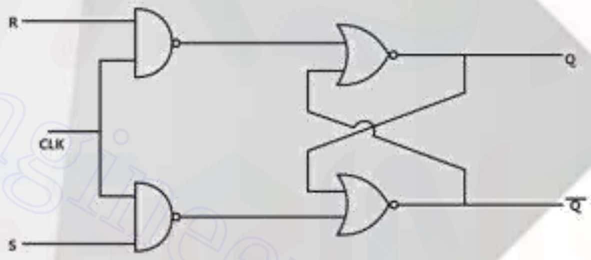
$\bar{S}$	$\bar{R}$	Q_{n+1}
0	0	1
0	1	1
1	0	0
1	1	Q_n



- S=1, R=1 and Q_{n+1} =1 is impractical state

3) Clocked SR Flip Flop





- When ClK = 0, the flip flop retains its previous state.
- When ClK = 1

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	

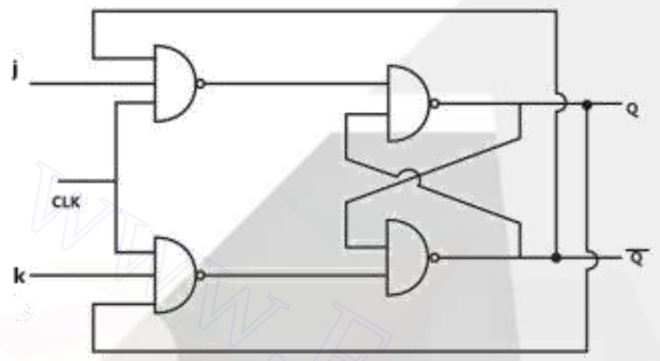
Ambiguous state

- Characteristics equations : $Q_{n+1} = S + \bar{R}.Q_n$

4) <u>J – K Flip Flop</u>

To convert SR flip flop to a JK flip flop.

$S = j\bar{Q}$; $R = KQ$

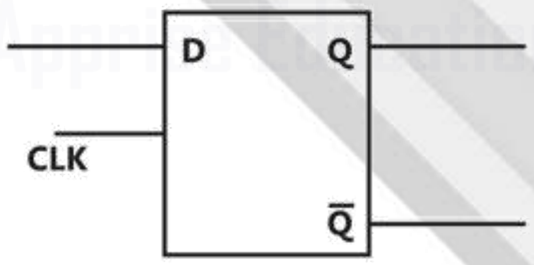


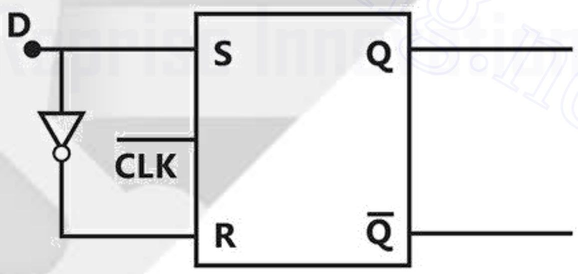
Characteristics equation

$Q_{n+1} = j\bar{Q}_n + \bar{K}Q_n$

J	K	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	$\bar{Q}_n$

5) <u>D – Flip Flop</u>

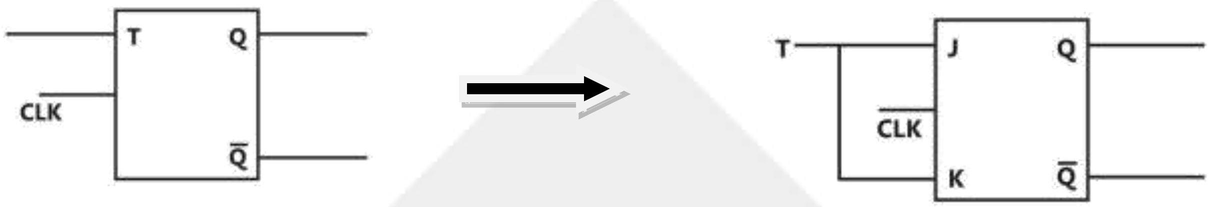




D	Q_{n+1}
0	0
1	1

Characteristics equation $Q_{n+1} = D$

6) T – Flip Flop

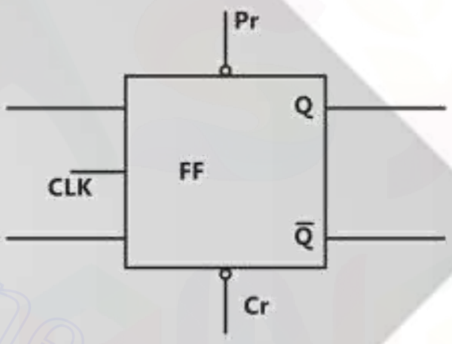


T	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

Characteristics equation

$$Q_{n+1} = T \oplus Q_n$$

Asynchronous or direct input



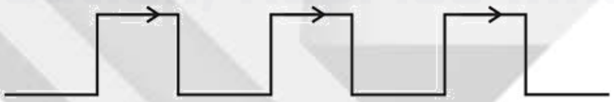
CLK	P_r	C_r	Q_{n+1}
0	0	1	1
0	1	0	0
1	1	1	

o\p depends on characteristic table of flip-flop

- Preset and clear input when enabled set or reset the flip flop irrespective of the state of clock.

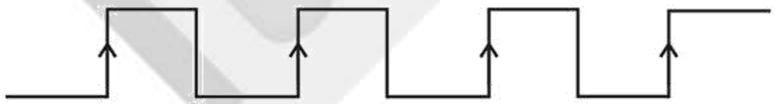
Types of Triggering

1) **Level Triggered FF**

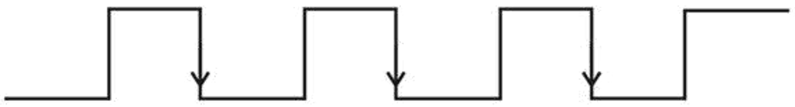


2) **Edge Triggered FF**

a) +ve edge triggered



b) – ve edge triggered



- Level triggered FF are called as latch and edge triggered FF are called as Flip Flops.

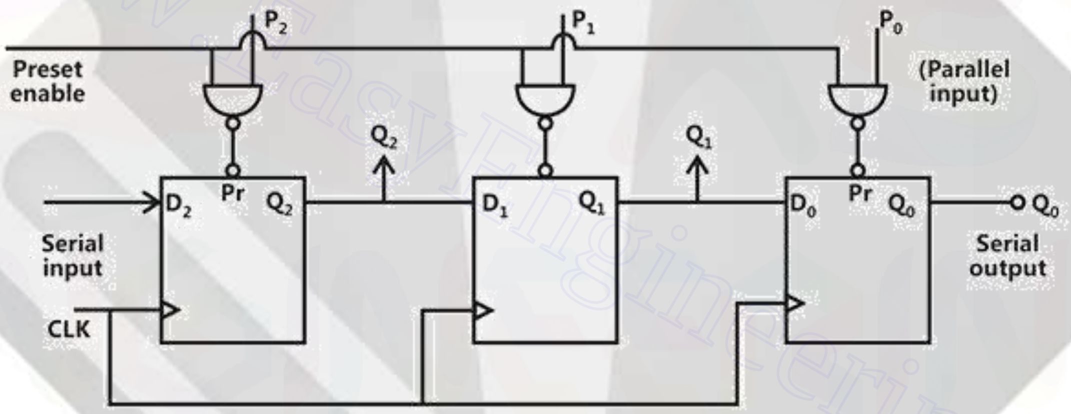
Race around Condition

- When a FF is in toggle mode, then due to propagation delay of gates involved in construction of FF, output toggles multiple times instead of once & this is called as Race around condition.
- This only occurs in level triggered FF.
- To avoid this problem, Master – slave configuration is used.

Note: Whenever a FF is in toggle mode, output frequency is half of input frequency.

Applications of FF

1) **Shift Register**



3 – bit shift Register

$Q_2\ Q_1\ Q_0 \rightarrow$ parallel output

$P_2\ P_1\ P_0 \rightarrow$ parallel input

1) **Serial input parallel output (SIPO)**

CIK	serial i / p	Q_2	Q_1	Q_0
0	–	0 ↘	0 ↘	0
1	1	1 ↘	0 ↘	0
2	0	0 ↘	1 ↘	0
3	1	1	0	1

Parallel output

For n – bit, time taken = nxT   T=clock period

2) Serial input serial output (SISO)

ClK	serial o / p	Q_2	Q_1	Q_0
0	–	0 ↘	0 ↘	0
1	1	1 ↘	0 ↘	0
2	0	0 ↘	1 ↘	0
3	1	1 ↘	0 ↘	1
4	–	–	1 ↘	0
5	–	–	–	1

Serial output

For n – bits, time taken = (2n - 1)T, T = clock period

3) Parallel input parallel output (PIPO)

Parallel input can be fed to register using preset enable and for input to propagate to parallel output, it does not require any clock pulse.

4) Parallel input serial output (PISO)

Suppose $P_2\ P_1\ P_0 = (101)_2$

ClK	Q_2	Q_1	Q_0
–	1 ↘	0 ↘	1
1	–	1 ↘	0
2	–	–	1

Serial output

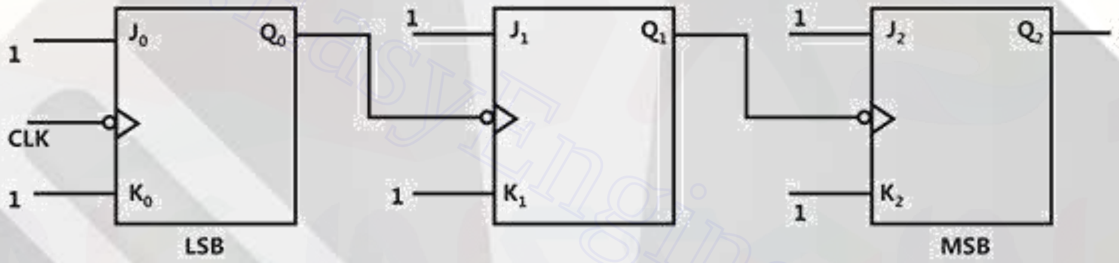
For n – bits, (n-1) clock cycles are required.

COUNTERS

Asynchronous Counters

- Different ff are applied with different clocks.
- No. of stages in a counter are called as modules of a counter.
  MOD – 5 Counter = 5 Stages
- $2^n \geq N$
  N = no. of bits or no. of flip flop required
  N = no. of stages in a counter
- If MOD – M and MOD – N counter are cascaded, resultant counter is MOD – (MN)

Ripple Counter



ClK	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

This is a MOD – 8 up counter

In a n – bit ripple counter, propagation delay of each ff is $t_{pd\ ff}$, then time period of ClK is

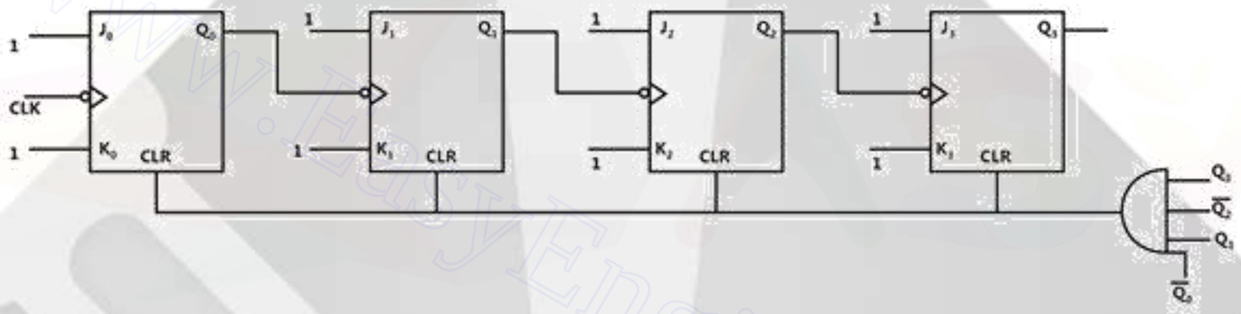
$$t_{ClK} \geq nt_{pdff} \Rightarrow f_{ClK} \leq \frac{1}{nt_{pdff}}, \quad \boxed{f_{max} = \frac{1}{n \times t_{pdff}}}$$

Note:

i) –ve edge trigger → Q as clock → up counter
ii) +ve edge trigger → $\bar{Q}$ as clock → up counter
iii) –ve edge trigger → $\bar{Q}$ as clock → down counter
iv) +ve edge trigger → $\bar{Q}$ as clock → down counter

BCD Counter (Decade Counter)

- 4 Flip flops are required.

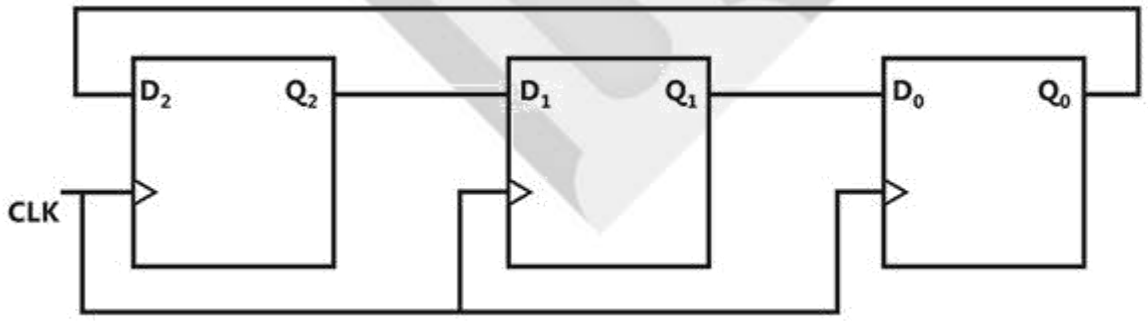


This counter counts from 0000 – 1001

And as soon as count is incremented to 1010, then CLR input of ff is asserted and all ff are reset to 0 and count again becomes 0000, so this counter counts from 0 – 9.

Ring Counter (Synchronous Counter)

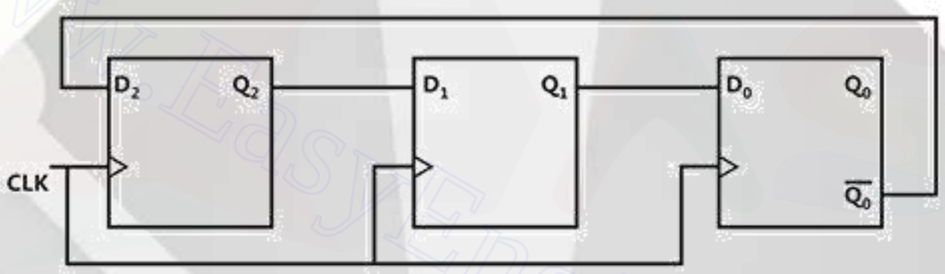
The last FF output is connected to first FF input.



ClK	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	1	0	0
3	0	1	0
4	0	0	1

A n – bit synchronous counter has n – status.

Johnson Counter (Twisted Ring Counter)



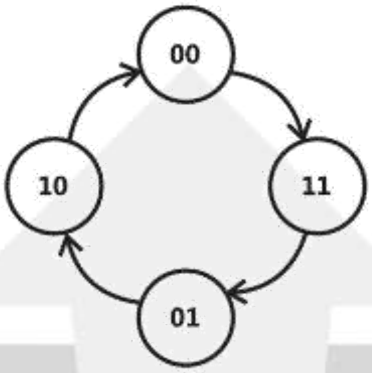
ClK	Q_2	Q_1	Q_0
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1
6	0	0	0

A n – bit Johnson counter has 2n states.
$T = (2n)\ T_{CLK}$

Synchronous counter design for given sequence

- Suppose counting sequence is $0 \to 3 \to 1 \to 2 \to 0$
- Using positive edge triggered D – FF

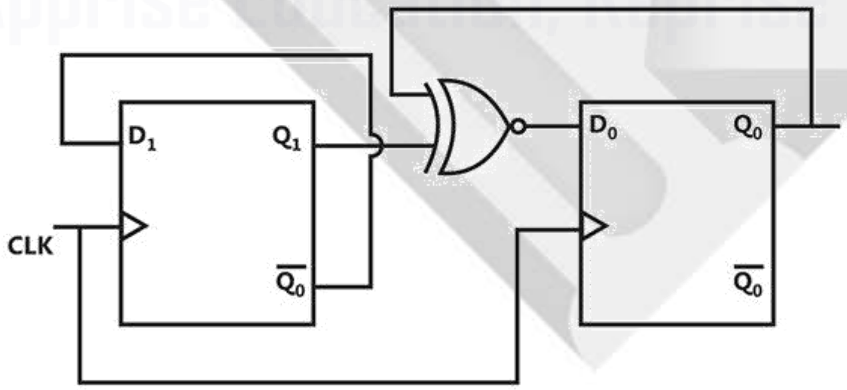
State diagram



Excitation Table

Present state		Next State		D_1	D_0
Q_1	Q_0	Q_1^+	Q_0^+		
0	0	1	1	1	1
1	1	0	1	0	1
0	1	1	0	1	0
1	0	0	0	0	0

$D_1 = \bar{Q}_1$; $D_2 = Q_1 \odot Q_0$



Conversion of different flip flops

From	SR Flip Flop	JK Flip Flop	D Flip Flop	T Flip Flop
SR FF	-	$S = J\bar{Q}$ $R = kQ$	$S = D$ $R = \bar{D}$	$S = T\bar{Q}$ $R = TQ$
JK FF	$J = S$ $K = R$	-	$J = D$ $K = \bar{D}$	$J = T$ $K = T$
D FF	$D = S + \bar{R}Q$	$D = J\bar{Q} + \bar{k}Q$	-	$D = T \oplus Q$
T FF	$T = S\bar{Q} + RQ$	$T = J\bar{Q} + kQ$	$T = D \oplus Q$	-

A/D and D/A Converters

Digital to Analog Converter (DAC)

- Resolution
  The change in analog voltage corresponding one LSB increment in digital input.

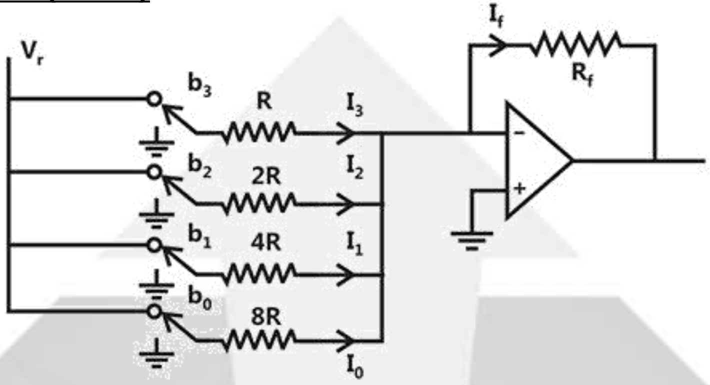
- $$\text{Resolution} = \frac{V_r}{2^n - 1}$$

  V_r = reference voltage corresponding to logic 1
  N = no. of bits

- $$V_{analog} = \text{Resolution} \times \text{Decimal equivalent of binary i/p}$$
  $$\%\text{resolution} = \frac{1}{2^n - 1} \times 100\%$$

- Resolution of R – 2R ladder type DAC is
  $$\text{Resolution} = \frac{V_r}{2^n}$$

1) Weighted Resister DAC (4 - bit)



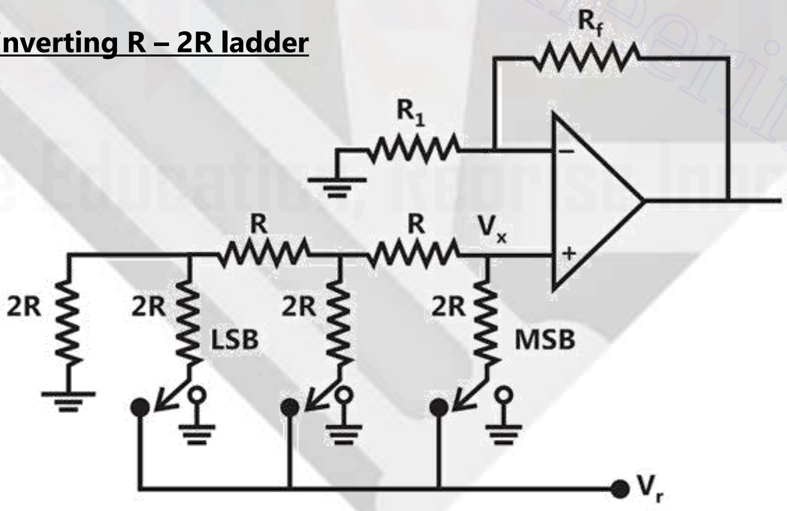
$$I_3 = \frac{V_r}{R} \times b_3 \;;\; I_2 = \frac{V_r}{2R} \times b_2 \;;\; I_1 = \frac{V_r}{4R} \times b_1 ;\; I_0 = \frac{V_r}{8R} \times b_0$$

$$V_0 = -\left(I_3 + I_2 + I_1 + I_0\right)R_f = \frac{-V_r}{8R} R_f \left(b_0 + 2b_1 + 4b_2 + 8b_3\right)$$

$$\text{LSB Resistance} = \left(2^n - 1\right) \text{ MSB Resistance}$$

2) R – 2R ladder

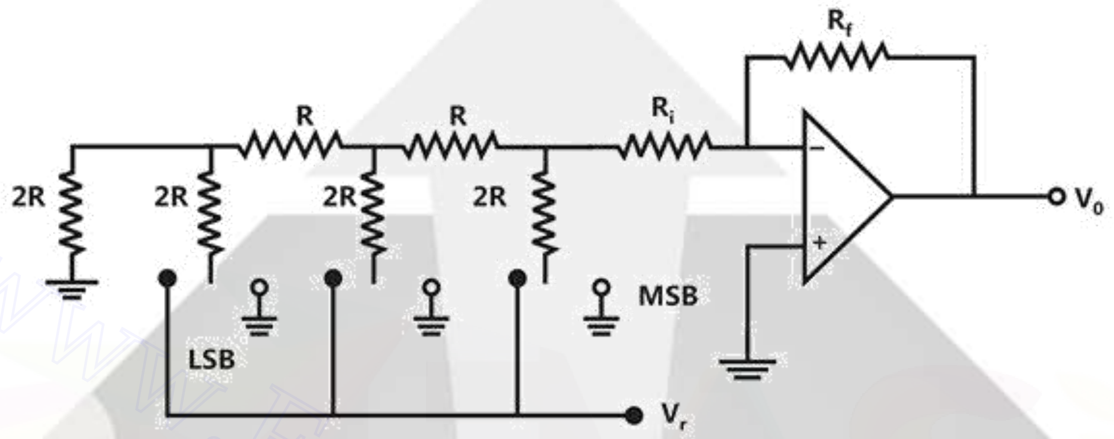
a) 3 – bit Non – inverting R – 2R ladder



$$V_0 = \left(1 + \frac{R_f}{R_1}\right)V_x = \frac{V_r}{2^n} \sum_{i=0}^{n-1} 2^i\, b_i \left(1 + \frac{R_f}{R_1}\right)$$

$$= \text{Resolution} \times \text{Decimal} \times \text{gain}$$

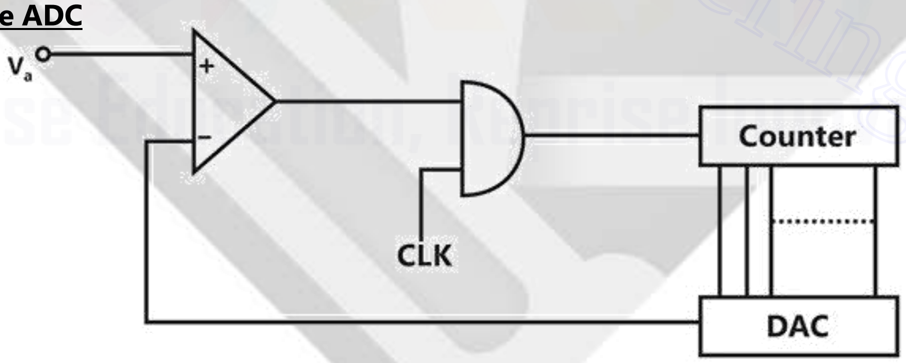
b) 3 – bit R – 2R Inverting ladder



$$V_0 = \frac{V_r}{2^n} \times \sum_{i=0}^{n-1} 2^i\, b_i \left(\frac{-R_f}{R_i + R} \right)$$

Analog to Digital Converter (ADC)

a) Counter Type ADC



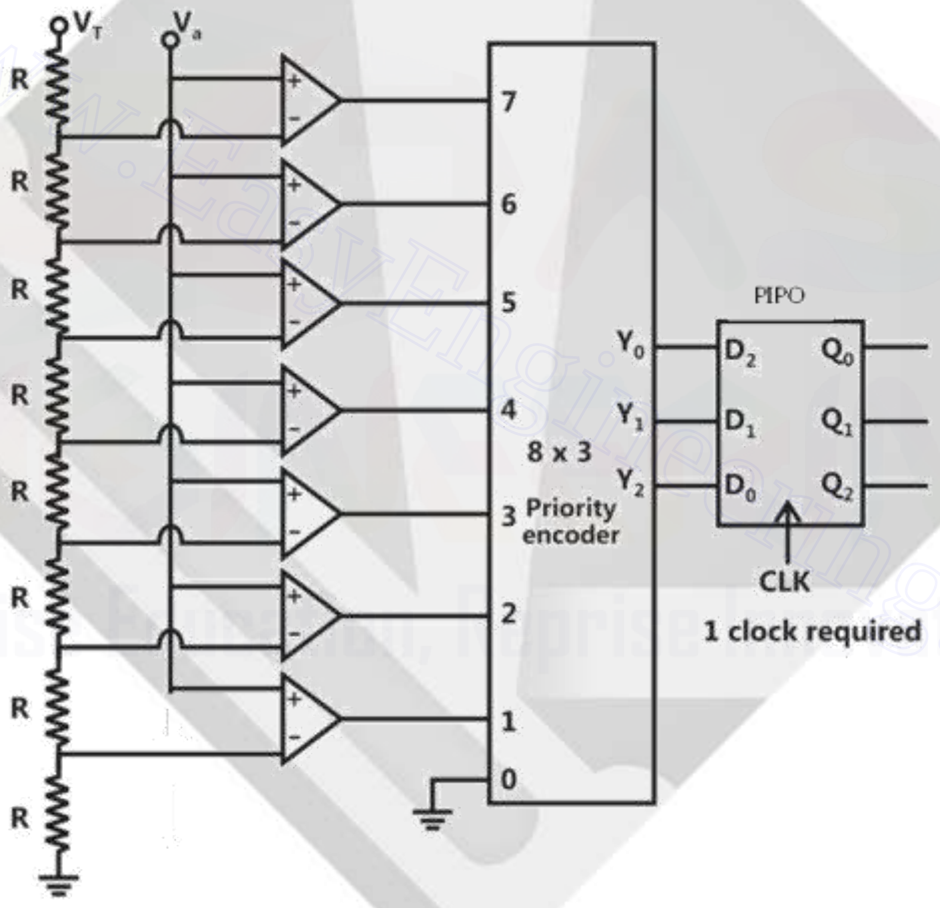
Maximum number of clock pulse required for n – bit conversion is $\left(2^n - 1\right)$

Max^m Conversion time = $\left(2^n - 1\right)T_{CLK}$

b) Parallel Comparator Type

- For n – bit $\begin{cases} (2^n - 1) \text{ comparators required} \\ 2^n \text{ resistors required} \\ 2^n \times n \text{ priority encoder} \end{cases}$

- This is called as Flash ADC.

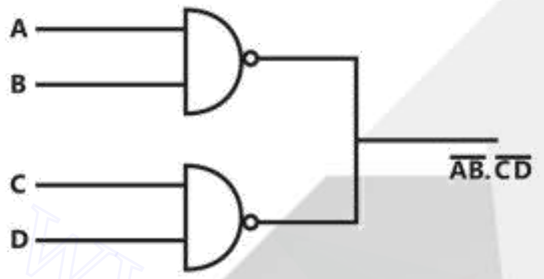


- Fastest ADC of all

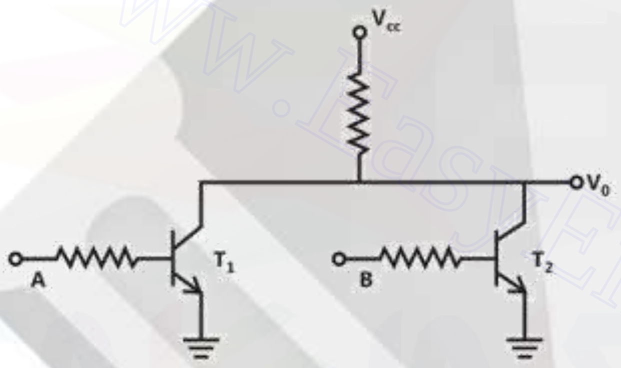
- For SAR & Dual slope ADC, refer EMMI K – Notes.

Logic Families

1) RTL (Resistor Transistor Logic)



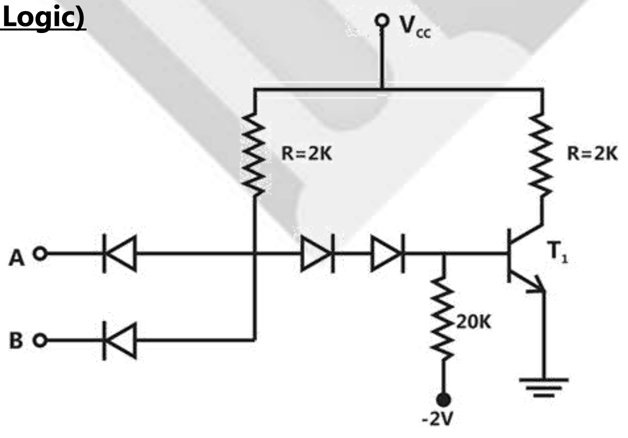
(Wired AND Logic)



(Basic NOR Gate)

A	B	T_1	T_2	V_0
0	0	cut – off	cut – off	1
0	1	cut – off	saturation	0
1	0	saturation	cut – off	0
1	1	saturation	saturation	0

2) DTL (Diode Transistor Logic)

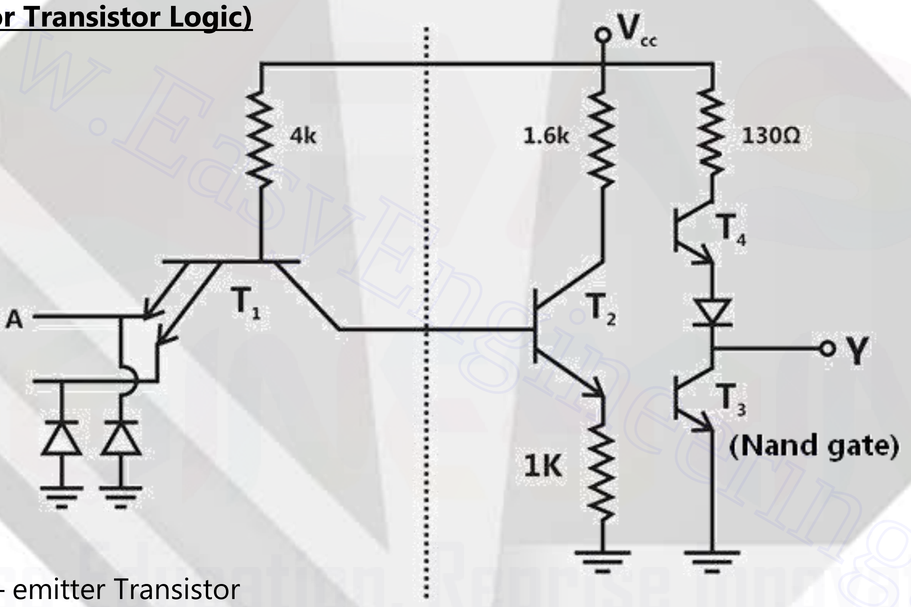


A	B	T_1	Y
0	0	OFF	1
0	1	OFF	1
1	0	OFF	1
1	1	ON	0

$\longrightarrow$ NAND Gate

- When all input are high then D_A & D_B are reverse biased and D_1, D_2. Become forward biased and T_1 becomes ON and output becomes low.

3) TTL (Transistor Transistor Logic)



T_1 : multi – emitter Transistor

A	B	T_1	T_2	T_3	T_4	Y
0	0	A	C	C	S	1
0	1	A	C	C	S	1
1	0	A	C	C	S	1
1	1	A	S	S	C	0

A : Active

C : Cut – off

S : Saturation

Microprocessor

The 8085 Microprocessor

- It is an 8 bit up (microprocessor)
- It is an 40 – PIN IC
- Its data – bus has 8 bits
- Its address bus has 16 bits
- It is capable of addressing 64 K of memory

Address Bus:

- It is 16 bits of length
- It is unidirectional bus.
- It is decided in to 2 parts namely

Lower order address bus $(A_0 - A_7) \rightarrow$ is also called "Line number "

Higher order address bus $(A_8 - A_{15}) \rightarrow$ is also called "page number "

Interrupts and externally initiated operations: -

- The 8085 up has 5 interrupts signals that can be used to interrupt a program execution
- It also accepts external interrupts to provide acknowledgement (ack) to the external device.
- Here TRAP, RST – 7.5, RST – 6.5, RST – 5.5, INTR are called Hardware interrupts.

1. INTR
   - It is abbreviated as interrupt request
   - It is used as general purpose interrupt
   - It has least or 5th priority
   - It is a non-vectored interrupt
   - Address is provided by user or external device
   - It is a level triggered signal.
2. INTA
   - It is abbreviated as interrupt acknowledge
   - It is an output signal.
3. TRAP
   - It has highest priority
   - It is the only non-maskable interrupt.

- It is a vectored interrupt.
- Also called RST 4.5
- This is both edge and level triggered signal.
- Its vectored address = $(0024)_H$

  Trick : since it is a RST – 4.5

  So, $4.5 \times 8 = 36 \xrightarrow[\text{decimal}]{\text{Hexa}} (24)_H \Rightarrow (0024)_H$

4. RST – 7.5
   - Is has 2 nd highest priority
   - It is maskable interrupt
   - It is a vectored interrupt.
   - It is edge triggered only
   - It vectored address = $(003C)_H$
5. RST – 6.5
   - It has 3rd highest priority.
   - It is a maskable interrupt.
   - It is a vectored interrupt.
   - It is level triggered.
   - It vectored address = $(0034)_H$.
6. RST – 5.5
   - If has 4th highest priority.
   - It is a maskable interrupt.
   - It is a vectored interrupt.
   - It is level triggered
   - Its vectored address = $(002C)_H$

8085 Microprocessor Flags

The flags are affected by the arithmetic and logic operations in the ALU :-

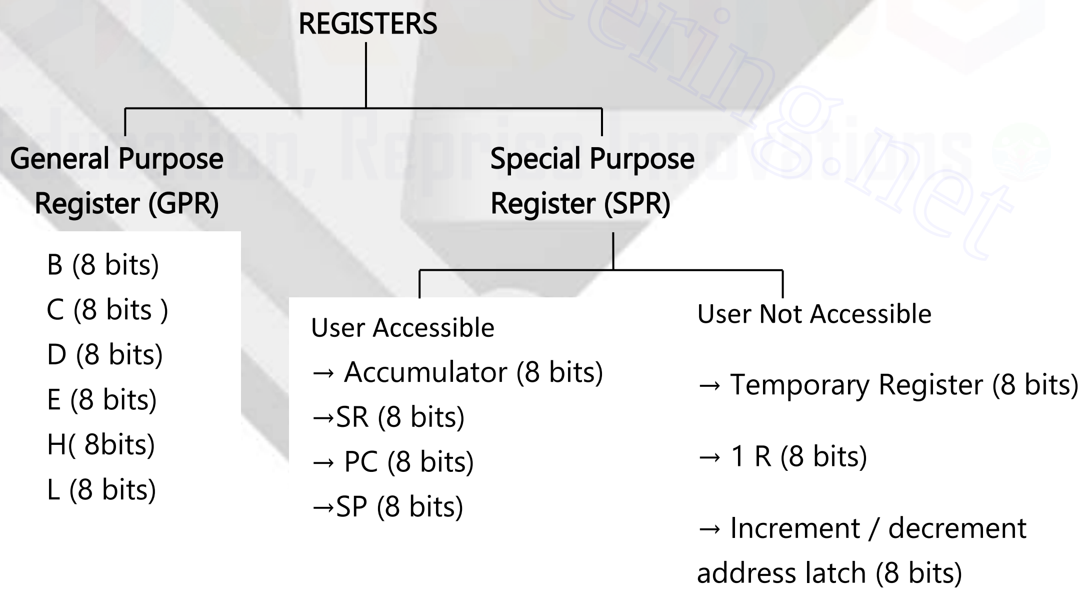
In most of these operations the result is stored in accumulator therefore the flags generally reflect data conditions in the accumulator with some exceptions. The descriptions and conditions of the flag as follows:

- **Sign flag (S)** :- After execution of an arithmetic or logic operation, if bit D_7 of the result (usually in the accumulator) is 1, the sign flag is set .

- **Zero Flag (Z)** : - The zero flag is set if the ALU operations result in 0, and the flag is reset if the result is not 0. This flag is modified by the result in the accumulator as well as in other registers.
- **Auxiliary carry flag (AC)** : - In an arithmetic operation, when a carry is generated by digit D_3 and passed on to digit D_4 the AC flag is set.
- **Parity Flag (P)** : After an arithmetic or logical, operation, if the result has an even number of 1s, the flag is set. If it has an odd number of 1s, the flag is reset.
- **Carry flag (CY)** : If an arithmetic operation results in a carry, the carry flag is set; otherwise it is reset.
-

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		A_C		P		CY

- Among the five flags, the AC flag is used internally for BCD arithmetic the instruction set does not include any conditional jump instruction based on the AC flag of the remaining four flags, the Z and CY flags are those most commonly used.



- Possible register pairs are :
  ⇒ B – C (16 bits)
  ⇒ D – E (16 bits)
  ⇒ H – L (16 bits)

- **Accumulator (A)** : -
  ⇒ It is a 8 – bits SPR and user accessible
  ⇒ It acts as one source of operand to the ALU and destinations to the result.
  ⇒ During I/O data transfer, data is transferred between accumulator (A) and I/O device.

- **Status register (SR) :-**
  ⇒ It is also called " Flag registers"
  ⇒ It is used to store ALU results
  ⇒ "FLAGS" are used for testing of data conditions
  ⇒ PSW (program status word) = Accumulator + flag register. Also PSW is a 16 bit register.

- **Program counter (PC) :-**
  ⇒ It is a a 16 – bit SPR which is accessible
  ⇒ It is required to keep track of the address of the next instruction to be fetched from the memory of execution.
  ⇒ In other words we can say, PC provides the address of next instruction to memory which has to be executed
  ⇒ when a byte is fetched then PC automatically incremented by 1 to point to next memory location.
  ⇒ when the microprocessor is reset, the PC sets to 0

- **Stock pointer :**
  It is a 16 bit SPR used as memory pointer SP provides the address of stack top or top address of stack.
  ⇒ A memory location in R/W memory is called "STACK". It is a part of RAM, which is used during subroutines PUSH and POP operations.

Instruction Set:

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
LXI r_P , 16 bit Data (load register pair immediately)	[rp] ← 16 bit data [rh] ← 8 MSB's of data [rl] ← 8 LSB's of data	Lx 1H, 2800 H i.e. |L| ← [00] (H) ← [28]	MC=1+2=3	4T + (3T x 2) = 10T	NO FLAGS are Affected
LDA address (Load accumulator direct)	| A | ←|[address]|	LDA 2400 H	1 + 3 = 4	4T + (3T x 3) = 13T	No flags are Affected
STA address (STORE accumulator direct)	|[address]| ← |A|	STA 2000H	MC=1+2+1 = 4	4 T + (2x3T) + 3T = 13T	No flags are affected
LHLD address (load H – L pair direct)	| L | ← [|address|] | H | ← [|address + 1|]	LHLD2500 H	MC = 1 + 4 = 5	4T + (4 x3T) = 16T – states	No flags are affected
SHLD address (Store H – L pair direct)	[|address|] ← | L | |[address + 1]| ← | H |	SHLD 2500 H	MC=1+2+2 = 5	4T + (2 x 3T) + (2 x 3T) =16 T states	No flags are affected
LDAX r_P : (Load accumulator indirect)	| A | ←|[rp]|	LDAX B	MC = 1 + 1 = 2	4T + 3 T = 7T – States	No flags are affected
STAXr_P : (store accumator indirect)	|[rp]| ←|A|	STAXD	MC = 1 + 1 =2	4T + 3T = 7T	NO Flags are Affected
XCHG : (Exchange the content of H – L pair D – E pair)	|H – L| ↔ |D – E|	XCHG data	MC = 1	4T states	No flags are Affected

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
MOV r_1, r_2 (move the content of one register into another register)	$\lvert r_1 \rvert \leftarrow \lvert r_2 \rvert$	MOV A, B	1 MC	4 T - STATE	NO FLAGS Affected
MOV r, M (Move the content of Memory to register)	$\lvert r \rvert \leftarrow \lvert M \rvert$ or $\lvert r \rvert \leftarrow \lvert [H-L] \rvert$	MOV B, M	MC = 1 + 1 = 2	4T + 3T = 7T	No flags Affected
MOV M, r (Move the content of register to memory)	$\lvert M \rvert \leftarrow \lvert r \rvert$ or $\lvert [H-L] \rvert \leftarrow \lvert r \rvert$	MOV M, C	MC = 1 + 1 = 2	4 T + 3 T = 7T	No flags affected
MVI r, data (Move immediate data to register)	$\lvert r \rvert \leftarrow$ data	MVI A, 05	MC = 1 + 1 = 2	4T + 3 T = 7 T	No flags affected

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
MVI M, data (move immediate data to memory)	|[H – L]| ← |data| or |M| ← |data|	LXI H, 2400 H MVI M, 08 HLT ⇒ Halt	MC = 1+ 1 + 1 = 3	4 T + 3T+3T = 10T –states	No flags are affected
ADD r (Add register to accumulator)	| A | ← | A | + | r |	ADD B	MC = 1	4T states	All flags are affected
ADC r : (Add register with carry to accumulator)	| A | ← | A | + | r | + | cs |	ADC B	MC = 1	4T – states	All flags are affected
ADD M : (add memory to accumulator)	| A | ← | M | or | A | ← | [H – L] |	ADD M	MC = 1 + 1 =2	4T + 3 T = 7T – States	All flags are affected
ADC M : (add memory to accumulator)	| A | ← | A | + | M | + | CS | or | A | ← | A | + | [H – L] | + | CS |	ADC M	MC = 1 + 1 =2	4T + 3 T = 7T – States	All flags are affected
ADI data : (Add immediate data to accumulator)	| A | ← | A | + data	ADI 08	MC = 1 + 1 =2	4T + 3T = 7T	All Flags are Affected
ACI data : (add with carry immediate data to accumulator)	|A| ← |A| + data + |cs|	ACI 08	MC = 1 + 1 =2	4T +3T = 7T	All flags are Affected
SUB r : (subtract register from accumulator)	|A| ← |A| – |r|	SUB 08	MC = 1	4 T	All flags are affected

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
SBB r : (subtract register from accumulator with Borrow)	$\|A\| \leftarrow \|A\| + \|r\| - \|cs\|$		MC = 1	4T	All flags are affected
SUB M : (subtract memory from accumulator)	$\|A\| \leftarrow \|A\| + \|M\|$ or $\|A\| \leftarrow \|A\| - [\|H-L\|]$		MC = 1 + 1 =2	4T + 3T =7T states	All flags are affected
SBB M : (subtract memory from accumulator alone with borrow)	$\|A\| \leftarrow \|A\| + \|M\|$ or $\|A\| \leftarrow \|A\| - [\|H-L\|]$		MC = 1 + 1 =2	4T + 3 T = 7T – States	All flags are affected
SUI data (subtract immediate data from accumulator)	$\|A\| \leftarrow \|A\| - \text{data}$		MC = 1 + 1 =2	4T + 3 T = 7T – States	All flags are affected
SBI data (subtract immediate from accumulator with borrow)	$\|A\| \leftarrow \|A\| - \text{data} - \|CS\|$		MC = 1 + 1 =2	4T + 3T = 7T	All Flags are Affected
INR r (increment register content by 1)	$\|r\| \leftarrow \|r\| + \|01\|$		MC = 1	4T	All flags are Affected except CY
INR M (increment memory content by 1)	$\|M\| \leftarrow \|M\| + [01]$ Or $[H-L] \leftarrow \|[H-L]\| + [01]$		MC= 1+1+1=3	4T+3T+3T =10T	All flags are affected except CY

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
INX rP : (increment the content of register pair by 1)	| rp | ← | rp | + | 0001 |	INX H INX SP INX C	MC = 1	4T	All flags except CY are affected
DCR M : (Decrement the content of memory by 1)	| M | ← | M | + | 01 | or | H – L | ← | H – L | – | 01 |		MC = 1 + 1 +1 = 3	4T + 3T+3T =10T	All flags are affected except CY flag
DCX rP: (Decrement the content of Memory by)1	| rp | ← | rp | – |0001 |	DCX B DCX SP DCX H	MC = 1	6T	No flags are affected
DAA : (Decimal adjust accumulator after addition)	DAA		MC = 1	4T	All flags are affected
DAD rP (Double addition register pair)	| H – L | ← | H – L | + |rp|		MC = 1 + 2 =3	4T +(2 x 3T) = 10T	Only carry (CY) is affected
ANA r: (And register with accumulator)	| A | ← | A | ∧ | r |		MC = 1	4T – States	All flags are Affected AC=1, CY= 0
ANA M : (And memory with accumulator)	|A| ← |A| ∧ |M| |A| ← |A| ∧ |[H – L]|		MC = 1+1 =2	4T+3T=7T - State	All flags are affected AC =1, CY = 0

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	FLAGS AFFECTED
ANI data : (And immediate data with accumulator)	| A | ← | A | ∧ | data |		MC = 1+1=2	4T +3T=7T	All flags affected AC = 1, CY =0
ORA r (OR register with accumulator)	| A | ← | A |V| M |		MC = 1	4T	All flags are affected CY = 0, AC = 0
ORA M : (OR memory with accumulator)	| A | ← | A |V|M |		MC = 2	7T	CY = 0, AC=0
ORI data (or data immediate with accumulator)	| A | ← | A |V| data |		MC = 2	7T	CY = 0, AC = 0
XRA r : EXOR register with accumulator	| A | ← | A |$\forall$|r |		MC =1	4T	All flags are affected and AC = 0, CY = 0
XRA M : EXOR memory with accumulator	| A | ← | A |$\forall$|M |		MC =1	4T	All flags are affected and AC = 0, CY = 0
XRI data: EXOR immediate data with accumulator	| A | ← | A |$\forall$data		MC =2	7T	All flags are affected AC =0, CY = 0
CMA : (Complement with Accumulate)	| A | → | $\vec{A}$ |		MC = 1	4T	No flags are affected
CMP r: (Compare register with accumulator)	| A | ← | A | − | r |		MC = 1	4T	All flags are Affected

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
CMP M : (compare memory with accumulator)	$\|A\| \leftarrow \|A\| - \|M\|$		MC = 1+1 =2	4T+3T=7T	All flags are affected
CPI data : (compare immediate data with accumulator)	$\| A \| \leftarrow \| A \| - \| \text{data} \|$		MC =2	4T +3T=7T	All flags affected
CMC : (complement the carry status)	$\| CS \| \leftarrow \| \overline{CS} \|$		MC = 1	4T	No flags are affected except carry flag
STC (Set carry status)	$\| CS \| \leftarrow 1$		MC = 1	4T	NO flags are affected except CY flag
RST n: (Restart)	$[\| SP - 1\|] \leftarrow \| PC \|_H$ $\|[SP - 1]\| \leftarrow [PC]_L$ $[\|SP\|] \leftarrow SP - 2$ $\|PC\| \leftarrow 8$ times		MC = 1+2=3	6T+(3T x 2)= 12T – states	No flags are affected
Push rP: (Push the content of register pair to stack)	$\| SP - 1\| \leftarrow \| rh \|$ $\|[SP - 2]\| \leftarrow \| r \|$ $\|SP\| \leftarrow \|SP\| - 2$		MC =1+2=3	6T + (3T x 2) = 12T – state	No flags are affected
POP rP : (POP the content of register pair which has been saved from stack)	$\| r \| \leftarrow \| sp \|$ $\| rh \| \leftarrow \| sp + 1\|$ $\| sp \| \leftarrow \| sp \| + 2$		MC =1 +2=3	10T – state	No flags are affected
SPHL : (move the content of HL pair to SP)	$\| H - L \| \leftrightarrow \| SP \|$		MC = 1	6T	No flags are affected

INSTRUCTION	SYMBOLIC FORM	EXAMPLE	MACHINE CYCLE	T-STATE	F LAGS AFFECTED
XTHL : (Exchange stock top with H – L pair)	| L | ↔ | SP | | H | ↔ |SP| +1		MC = 5	4T + (3T x 2) + (3T x 2) = 16 T	No flags are Affected
IN Port address : (Input to accumulator from I/O part)	|A| ← |port |		MC = 1+1+1 =3	4T+3T+3T= 10T	No flags are affected
Out port address : (output to accumulator to I/o part)	$(\text{port}) \leftarrow \|A\|$		MC =3	4T+3T+3T= 10T	No flags are affected
HLT : (Halt)	HLT		MC = 1	5 T – state	No flags are affected
PCHL :(jump to address specified by H – L pair)	$\|PC\| \leftarrow \|H-L\|$ $\|PCH\| \leftarrow \|H\|$ $\|PCL\| \leftarrow \|L\|$		MC = 1	6T – state	NO flags are affected
Unconditional JMP instruction			MC = 1+2=3	4T+(2 x3T)= 10T –states	

RLC : (Rotate accumulator left)

Symbolic form : $[A_{n+1}] \leftarrow [A_n]$

$[A_0] \leftarrow [A_7]$

$[CS] \leftarrow [A_7]$

The content of the accumulator is rotated left by one bet

RRC : (Rotate accumulator right)

Symbolic form : $[A_7] \leftarrow [A_0]$

$[CS] \leftarrow [A_0]$

$[A_n] \leftarrow [A_{n+1}]$

The content of the accumulator is rotated right by one bit

RAL : (Rotate accumulator left through carry)

Symbolic form : $[A_{n+1}] \leftarrow [A_n]$

$[CS] \leftarrow [A_7]$

$[A_0] \leftarrow [CS]$

The content of the accumulator is rotated left one bit through carry.

RAR : - (Rotate accumulator right through carry)

Symbolic form : $[A_n] \leftarrow [A_{n+1}]$

$[CS] \leftarrow [A_0]$

$[A_7] \leftarrow [CS]$

The content of the accumulator is rotated right one bit through carry.

Conditional JMP instruction : -

OPCODE	Operand	Description
JC	16 – bit	jump on carry (if result generate CY = 1)
JNC	16 – bit	jump on carry (cy = 0)
JZ	16 – bit	jump on zero (if result generate or Z = 1)
JNZ	16 – bit	jump one no zero (Z = 0)

JP	16 – bit	jump on plus (if D_7 = 0, s =0)
JM	16 – bit	jump on minus (if D_7 =1 and S = 1)
JPE	16 – bit	jump on even parity (p = 1)
JPO	16 – bit	jump on odd parity (P = 0)

Unconditional CALL instruction :

When it is executed, microprocessor will store address of next instruction is STACK

MC = 1 + 2 + 2 = 5

6T + (3T x 2) + (3T x 2) = 18T – states

No flags are affected

Conditional CALL :

CC	call subroutine if carry flag is set (CY =1)
CNC	call subroutine if carry flag is reset (CY = 0)
CZ	call subroutine if zero flag is set (z = 1)
CNZ	call subroutine if zero flag is reset (z = 0)
CM	call subroutine if sign flag is set (s =1, negative number)
CP	call subroutine if sign flag is reset (s =0, positive number)
CPE	call subroutine if parity flag is set (P =1, even parity)
CPC	call subroutine if parity flag is reset (P =0, odd parity)

Unconditional RET instruction :

It will change program – sequence from subroutine to main program.

MC = 1 + 2 = 3

4T + (3T x 2) = 10T – states

No flags are affected.

Conditional RET instruction :

RC	Return if carry flag is set (CY =1)
RNC	Return if carry flag is reset (CY = 0)
RZ	Return if zero flag is set (z = 1)
RNZ	Return if zero flag is reset (z = 0)
RM	Return if sign flag is set (s =1, negative number)
RP	Return if sign flag is reset (s =0, positive number)
RPE	Return if parity flag is set (P =1, even parity)
RPC	Return if parity flag is reset (P =0, odd parity)

RST n : (restart)

Symbolic form : $[(Sp - 1) \leftarrow [PC]_H]$

$(Sp - 2) \leftarrow (PC)_L$

$(Sp) \leftarrow (Sp - 2)$

$[PC] \leftarrow 8 \text{ times } n$

MC = 1 + 2 =3

6 T + (3T x 2) = 12 T – states

No flags are affected