Date

Practice Questions

Ques1 Convert following to Decimal, Hexadecimal and Octal form

a) $(10110 1.1101)_2$

Binary to Decimal

$10\,11\,01.\,11\,01$

$2^5\,2^4\,2^3\,2^2\,2^1\,2^0\;\;2^{-1}\,2^{-2}\,2^{-3}\,2^{-4}$

$2^5+2^3+2^2+2^0+2^{-1}+2^{-2}+2^{-4}$

$32+8+4+1+0.5+0.25+0.0625$

$(45.8125)_{10}$

A - 10
B - 11
D - 13
E - 14
F - 15

Binary to Hexadecimal

$\underbrace{10}_{8\,4\,2\,1}\;\underbrace{11\,01}_{8\,4\,2\,1}.\;\underbrace{11\,01}_{8\,4\,2\,1}$

$(2D.D)_{16}$

Binary to Octal

$\underbrace{10}_{4\,2}\;\underbrace{11\,01}_{4\,2\,1}.\;\underbrace{110}_{4\,2\,1}\;\underbrace{1}_{4}$

$(55.64)_8$

b) $(11111011.100101)_2$

Binary to Decimal

$1\;1\;1\;1\;1\;0\;1\;1.\;1\;0\;0\;1\;0\;1$

$2^7\;2^6\;2^5\;2^4\;2^3\;2^2\;2^1\;2^0\;\;2^{-1}\;2^{-2}\;2^{-3}\;2^{-4}\;2^{-5}\;2^{-6}$

$2^7+2^6+2^5+2^4+2^3+2^1+2^0+2^{-1}+2^{-4}+2^{-6}$

$128+64+32+16+8+2+1+0.5+0.0625+0.015625$

251.578125

Binary to Hexadecimal

$\underbrace{1\;1\;1\;1}_{8\,4\,2\,1}\;\underbrace{1\;0\;1\;1}_{8\,4\,2\,1}.\;\underbrace{1\;0\;0\;1}_{8\,4\,2\,1}\;\underbrace{0\;1\;0\;0}_{8\,4\,2\,1}$

$(FB.94)_{16}$

Binary to Octal

$\underbrace{01}_{4\,2}\;\underbrace{1\;1\;1\;1}_{4\,2\,1}\;\underbrace{0\;1\;1}_{4\,2\,1}.\;\underbrace{1\;0\;0}_{4\,2\,1}\;\underbrace{1\;0\;0}_{4\,2\,1}\;\underbrace{0}_{4}$

$(373.450)_8$

What makes you happy?
#HappyCollegeDays

c) $(10110.1101)_2$

Binary to decimal

$2^4 + 2^2 + 2^1 + 2^{-1} + 2^{-2} + 2^{-3}$

$16 + 4 + 2 + 0.5 + 0.25 + 0.125$

$= 22.875$

Binary to Hexadecimal

0010 1101 . 1010

2 D . A

$(2D.A)_{16}$

Binary to Octal

010 110 . 110 100

2 6 . 6 4

$(55.64)_8$

What makes you happy?
#HappyCollegeDays

Ques4 Convert the following numbers to binary

a) $(12.0625)_{10}$

b) $(41.375)_{10}$

a) 1100.

0.0625×2	$= 0.125$	0
0.125×2	$= 0.25$	0
0.25×2	$= 0.5$	0
0.5×2	$= 1.0$	1

$(1100.0001)_2$

b) 41.

101001

0.375×2	$= 0.750$	0
0.750×2	$= 1.50$	1
0.50×2	$= 1.0$	1

2 | 41	1
2 | 20	0
2 | 10	0
2 | 5	1
2 | 2	0
2 | 1	
	1

(1100.011)

What makes you happy?
#HappyCollegeDays

Date

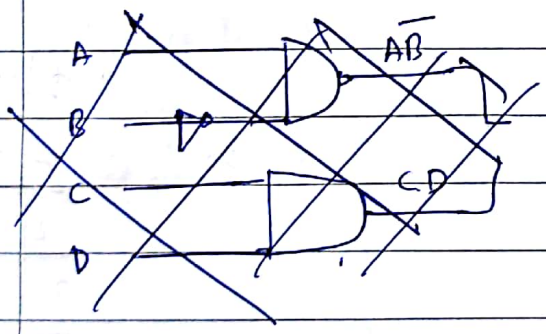
Question $F(A,B,C,D) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,5)$

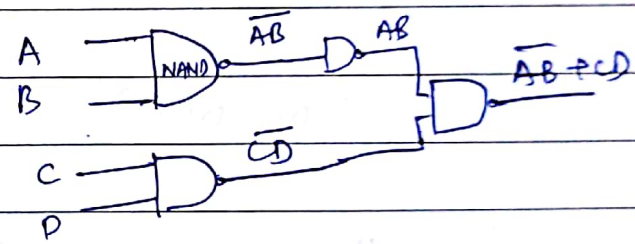
Using NAND

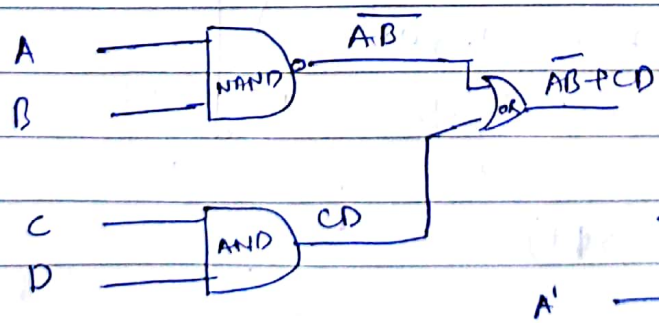
	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	X 0	1 1	1 3	X 2
$\bar{A}B$	4	5 X	1 7	6
AB	12	13	1 15	14
$A\bar{B}$	8	9	1 11	10

$Y = \bar{A}\bar{B} + CD$

Anal.





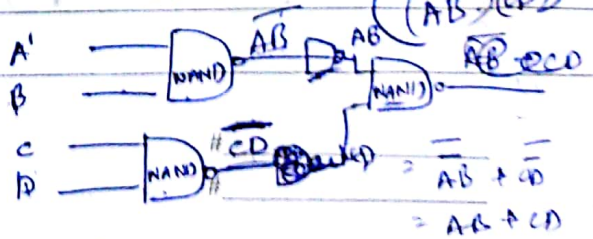


$\overline{(\bar{A}\bar{B})(CD)}$

$AB + \overline{CD}$

$\overline{(AB)(\bar{C}\bar{D})}$

$\bar{A}\bar{B} + CD$



$= \bar{A}\bar{B} + \bar{C}\bar{D}$

$= AB + CD$

$\overline{\bar{A}\bar{B}+CD}$

$\overline{(\bar{A}\bar{B}\cdot\bar{C}\bar{D})}$

$\bar{A}\bar{B}\cdot\bar{C}\bar{D}$ $\bar{A}\bar{B}+CD$

What makes you happy

#HappyCollegeDays

Deriving all gates from NAND Gate

NOT, AND, NOR, OR, XOR, XNOR

NAND.

0	0	1
0	1	1
1	0	1
1	1	0

AND using NAND

$X \cdot Y = (X \text{ NAND } Y) \text{ NAND } (X \text{ NAND } Y)$

$= \overline{(X \cdot Y)} \text{ NAND } \overline{(X \cdot Y)}$

Using De-morgan's law

$= (\bar{X} + \bar{Y}) \text{ NAND } (\bar{X} + \bar{Y})$

$= \overline{(\bar{X} + \bar{Y}) \cdot (\bar{X} + \bar{Y})}$

$= \overline{(\bar{X} + \bar{Y})} + \overline{(\bar{X} + \bar{Y})}$ [De-morgans first theorem]

$= \bar{\bar{X}} \cdot \bar{\bar{Y}} + \bar{\bar{X}} \cdot \bar{\bar{Y}}$

$= X \cdot Y + X \cdot Y$

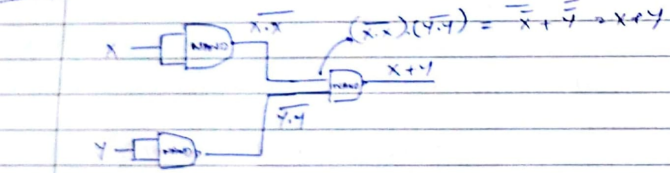
$= XY$

Date

(ii) OR

$X+Y = (X \text{ NAND } X) \text{ NAND } (Y \text{ NAND } Y)$

Proof



$(\overline{X \cdot X})(\overline{Y \cdot Y}) = \overline{\overline{X}} + \overline{\overline{Y}} = X + Y$

(iii) NOT

NOT X = X NAND X

$\overline{X \cdot X}$

$\overline{X} + \overline{X}$

$\overline{X}$

2-level circuit

Rules for NAND & NAND

i) Derived simplified SOP

ii) Draw circuit diag using AND, OR, NOR

iii) Replace all gates with NAND

What makes you happy?

#HappyCollegeDays

Date

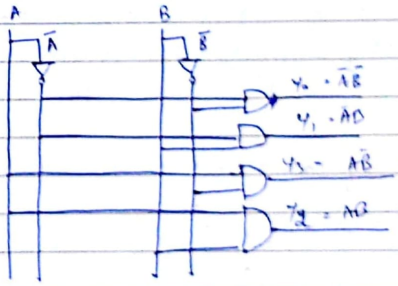
Decoder Circuits

$n \rightarrow 2^n$

Inputs Outputs

2 : 4 line Decoder

A	B	O/p	
0	0	Y_0	$\bar{A}\bar{B}$
0	1	Y_1	$\bar{A}B$
1	0	Y_2	$A\bar{B}$
1	1	Y_3	AB



What makes you happy?

Date

A	B	C	o/p	
0	0	0	Y_0	$\bar{A}\bar{B}\bar{C}$
0	0	1	Y_1	$\bar{A}\bar{B}C$
0	1	0	Y_2	$\bar{A}B\bar{C}$
0	1	1	Y_3	$\bar{A}BC$
1	0	0	Y_4	$A\bar{B}\bar{C}$
1	0	1	Y_5	$A\bar{B}C$
1	1	0	Y_6	$AB\bar{C}$
1	1	1	Y_7	ABC

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.

Implement following boolean functions using 3:8 decoder and external gates

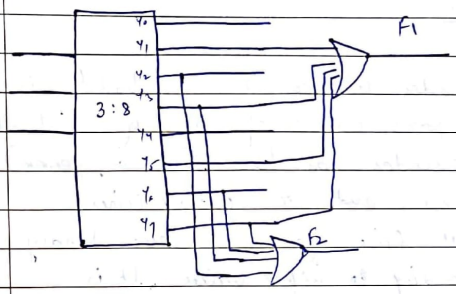
$f_1(A,B,C) = \Sigma m(1,3,5,7)$

$f_2(A,B,C) = \Sigma m(2,3,6,7)$.

What makes you happy?
#HappyCollegeDays

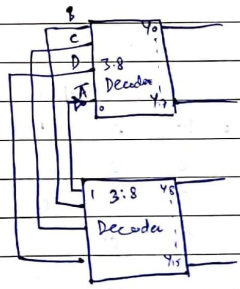
#_____
#_____

Date



Ques Design:

4:16 decoder using two 3:8 decoder



A is passed as an enable

if A is 0 (switched off the first chip and)

A is 1 (Niche wala 3:8 on hojaega) At a time only 8 outputs will be there. Coz 1 chip will be switched on at a time. If A → 0 (Y_0 -- Y_7 Niklega)

A → 1 (Y_8 — Y_{15} Niklega)

What makes you happy?
#HappyCollegeDays

Date

Encoder

An encoder is combinational digital circuit that performs inverse operation of a decoder and encoder can have 2^n (or fewer) input lines and n output lines. The output lines generate the binary code corresponding to input values. It is assumed that only one input has a value of 1 at any given time otherwise the circuit is unpredictable.

$2^n \rightarrow n$

$I/p \rightarrow o/p$ & 3 encoder

Inputs							Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

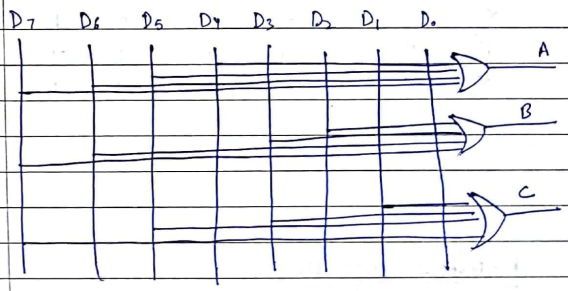
(jin D values k lie A, B, C e 1 aur hain, add krdo)

Date

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$



Priority encoder Encoder circuit which includes priority function.

If 2 or more inputs are equal to 1 at the same time, the input having the highest priority will take the precendence.

Date

4:2 encoder

				Outputs		
D_0	D_1	D_2	D_3	Y_1	Y_0	V (valid)
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

valid ye btata hai ki jo output hain vo valid hai ya ni. To first row jahan sab 0 hai to output bhi ni aega, jo bhi aiyega garbage hoga so valid is 0, Baakion mein 1

K-Map Simplification

For Y_1

	$\bar{D_2}\bar{D_3}$	$\bar{D_2}D_3$	D_2D_3	$D_2\bar{D_3}$
$\bar{D_0}\bar{D_1}$		1	1	1
$\bar{D_0}D_1$		1	1	1
D_0D_1		1	1	1
$D_0\bar{D_1}$		1	1	1

$Y_1 = D_2 + D_3$

Y_1 mein D_2 aur D_3 ke corresponding one hai

Tob like D_2 (1) aur D_3 (0) pe Y_1 1 hai

D_2 (X) aur D_3 (1) pe Y_1 1 hai

Date

Toh k map mein pehle jahan D_2(1) aur D_3(0) hai i.e $D_2\bar{D_3}$ un poore column pe 1 daalde. Similarly jahan D_3 1 ho wahan poore column mein 1 daaldo, baakion se koi pharak ni padta.

For Y_0.

	$\bar{D_2}\bar{D_3}$	$\bar{D_2}D_3$	D_2D_3	$D_2\bar{D_3}$
$\bar{D_0}\bar{D_1}$		1	1	
$\bar{D_0}D_1$	1	1	1	
D_0D_1	1	1	1	
$D_0\bar{D_1}$		1	1	

Yaha D_1 (1) D_2(0) D_3 (0) se matlab hai

Aur D_3 (1) se matlab hai

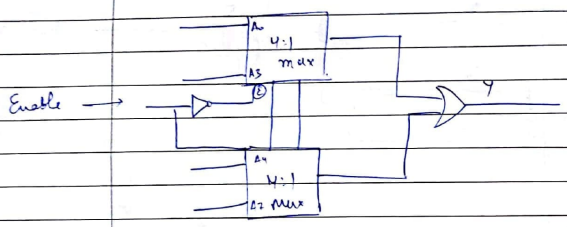
$Y_0 = D_3 + D_1\bar{D_2}$

For V.

0	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Date

Implement 8:1 MUX using 4:1 MUX



Ques. $F(A,B,C) = \Sigma m(1,3,5,6)$ using 4:1 mux

Not a K-map

fix a variable (say A)

	00	01	10	11
	D_0	D_1	D_2	D_3
$\bar{A}$ (·)	0	(1)	2	(3)
A (·)	4	(5)	(6)	7
	0	1	A	$\bar{A}$

Circle the Σm

Look Columnwise

if no circled — 0

Both circled — 1

one circled — look at respective row fixed element/element i.e. A or $\bar{A}$ in this case

What makes you happy?
#HappyCollegeDays

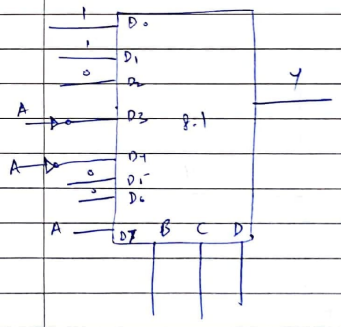
#_____
#_____

Date

Ques. $F(A,B,C,D) = \Sigma m(0,1,3,4,8,9,15)$ using 8:1 mux

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)

Plot this as inputs → 1 1 0 $\bar{A}$ $\bar{A}$ 0 0 A



What makes you happy?
#HappyCollegeDays

#_____
#_____

There variables are selection lines in mux

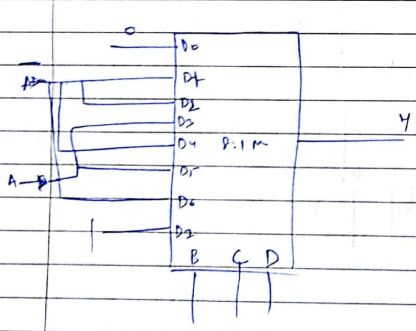
Date

Ques $F(A,B,C,D) = \sum m(0,3,5,8,9,10,12,14)$

$= \sum m(1,2,4,6,7,11,13,15)$

using 8:1 mux

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	0	$\bar{A}$	$\bar{A}$	$\bar{A}$	$\bar{A}$	$\bar{A}$	$\bar{A}$	1



What makes you happy?
#HappyCollegeDays

Ques Implement full adder using two 4:1 mux

A	B	C	Sum	Carry
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

Sum. $\sum m(1,2,4,7)$

Cout $\sum m(3,5,6,7)$

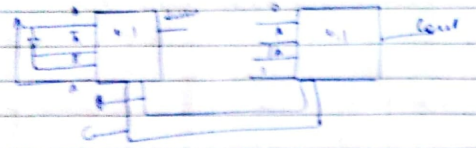
Sum.

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	A	$\bar{A}$	$\bar{A}$	A

Cout

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7

What makes you happy?
#HappyCollegeDays

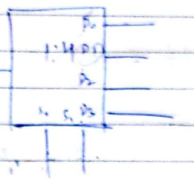


Demultiplexer

A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.

The selection of specific output lines is controlled by the values of n selection lines.

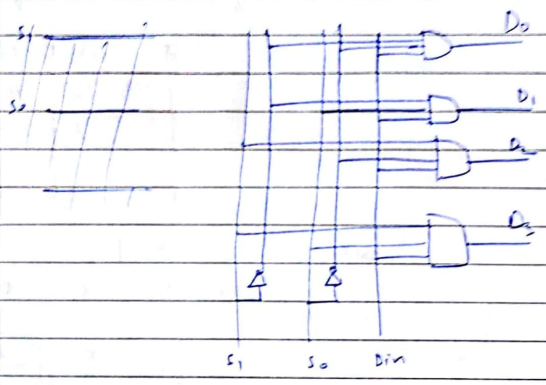
$n \rightarrow$ selection lines



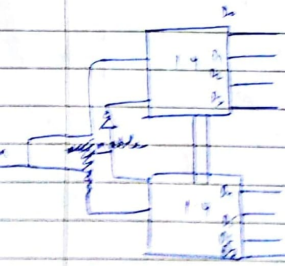
S_1, S_0 select the D value

Din selects the 0 or 1 to be outputted on the corresponding D value

Inputs			Outputs			
S_1	S_0	Din	D_0	D_1	D_2	D_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1



Ques Implement 3:8 demux using two 1:4 demux



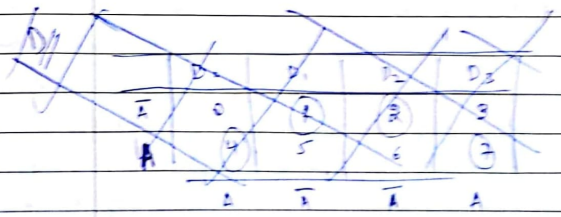
we add Enable because there should be 3 data selection lines in total

Ques Implement full subtractor using demultiplexer

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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

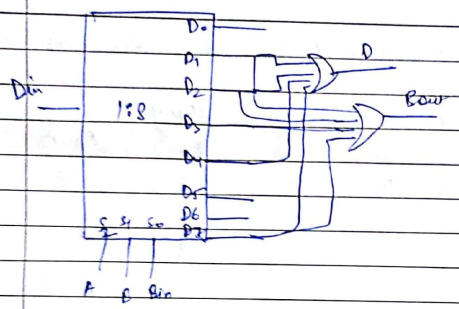
$D = \Sigma_m(1, 2, 4, 7)$

$B_{out} = \Sigma_m(1, 2, 3, 7)$



Ques.

Date



Ques Difference between combinational and sequential circuits.

Combinational	Sequential
1. Output variables are at all times dependent on the combⁿ of input variables	1. They depend not only on the present input variables but also on past history of these input variables
2. Memory unit is not required	2. It is required
3. They are faster	3. They are slower because of involvement of memory
4. Easy to design e.g. Parallel Adder	4. Comparitively harder to design eg. Serial Adder

#_____
#_____

Date

Sequential Circuits

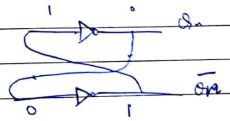
1. Latches (foundation of SC)

Latches are bistable elements (0 or 1) which are used as basic building blocks of most sequential circuits

Main difference between latches and flip flops is in the method used for changing their state
Flip flops normally samples its inputs and changes its outputs at times determined by clocking signal

Latch checks all of its inputs continuously and changes its outputs accordingly at any time independent of the clocking ~~state~~ signal

Latch

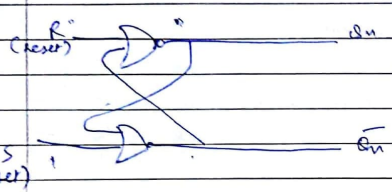


#_____
#_____

Q_n refers to current state
Q_{n+1} refers to the next state

Date

S R latch



NOR

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

Input S	Input R	Output Q_n	Output Q_{n+1}	
0	0	0	0	No change in next state
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	X	Indeterminate
1	1	1	X	

What makes you happy?
#HappyCollegeDays

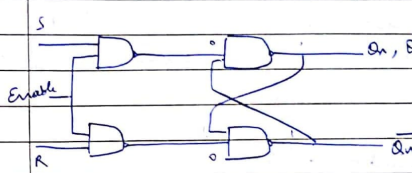
#
#

Date

Gated SR latch

NAND.

S	R	
0	0	1
0	1	1
1	0	1
1	1	0



In the SR latch we have seen that output changes occurs quickly after the input changes occurs i.e. latch is sensitive to S and R inputs at all times. However it can be used to create a latch i.e. sensitive to these inputs only when the enable is active.

En	S	R	Q_n	Q_{n+1}	
1	0	0	0	0	NC
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	Indeterminate
1	1	1	1	X	
0	X	X	0	0	
0	X	X	1	1	

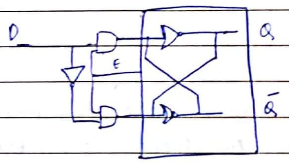
What makes you happy?
#HappyCollegeDays

#
#

Date

D-latch

Indeterminant state (1 1 wali) has to be avoided. Hence we put a not gate so that the 1 1 state never occurs



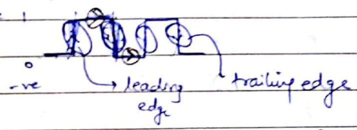
E	D	Q	$\bar{Q}$
0	X	0	0
1	0	0	1
1	1	1	0

D- latch mein jo input do wahi output aata hai

Date

Flip- Flops

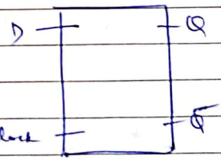
Clock



↳ edge triggered (Flip Flops)
↳ level triggered (latch)

If sequential circuits provides an output through leading edge or trailing edge they are known as edge triggered

→ ⓐ level triggered



Date

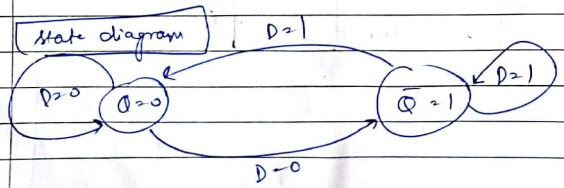
	Clock	D	Q	Q_{now}	$\overline{Q_{next}}$
level trigger	0	x	0	0	1
	1	x	0	0	1
edge trigger	↑	0	x	0	1
	↑	1	x	1	0

Kaiparak ni padta Q se jo input wali output isliye ~~level~~ edge trigger ~~ka~~ mein Q wala x hai.

Flip-flop actual truth table

Clock	D	Q	$\overline{Q}$
↑	0	0	1
↑	1	1	0

State diagram

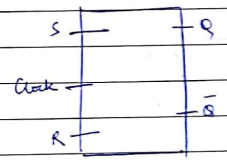


Date

Excitation table

Q	Q_{next}	D
0	0	0
0	1	1
1	0	0
1	1	1

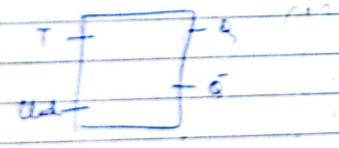
S-R Flip Flop



	R	S	Q_{next}
↑			

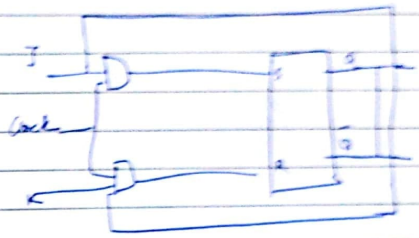
J	K	Q	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

Toggle

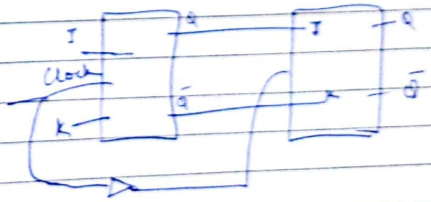


T	Q	$\bar{Q}$
0	1	0
1	0	1

J-K Flip Flop

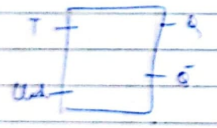


Clock	J	K	Q
↓	0	0	Hold (off)
↓	0	1	0
↓	1	0	1
	1	1	Toggle (on)



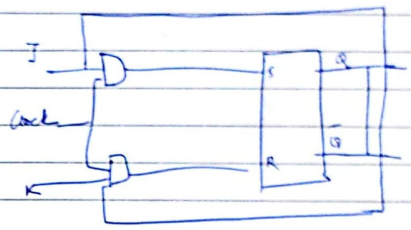
S	R	Q	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

Toggle

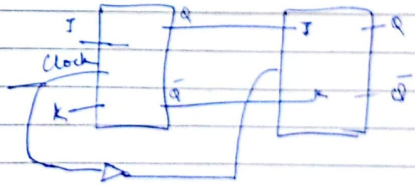


T	Q_n	$\bar{Q}_n$
0	1	0
1	0	1

J-K Flip Flop



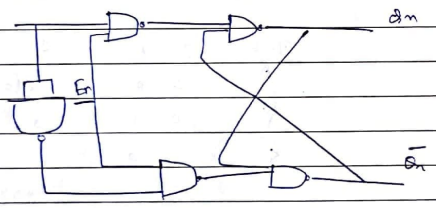
Clock	J	K	Q
↓	0	0	Hold (off)
↓	0	1	0
↓	1	0	1
	1	1	Toggle (on)



Date

Gated D latch

S	R	Q_{n+1}	
0	0	NC	
0	1	0	(Reset)
0	0	1	(Set)
1	1	*	(Indeterm)



NAND

				D	Q_n	Q_{n+1}
0	0	1		0	0	0
0	1	1	0→1	0	1	0
1	0	1		1	0	1
1	1	0		1	1	1

(NC

Date

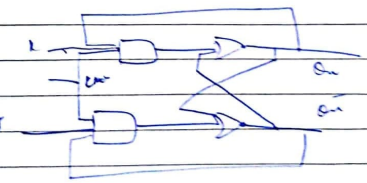
Generalised table

In	D	Q_{n+1}
1	0	0
1	1	1
0	X	Q_n

Previous looking at this state table of SR latch we realize that both inputs are same, the output either doesn't change or it is invalid or indeminant. In many practical applications these input conditions aren't required. Hence in D-latch they are avoided by making them complement of each other.

JK latch

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

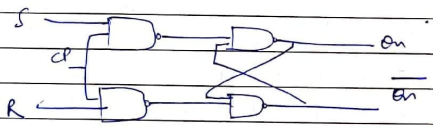


Date

	$J\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$
$\bar{Q}_n$	0	0	1	1
Q_n	1	0	0	1

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

SR Flip Flop



S	R	Q_{n+1}
0	0	Nc (No change)
0	1	0 (reset)
1	0	1 (set)
1	1	✗ (Indeterm

What makes you happy?
HappyCollegeDays

#
#

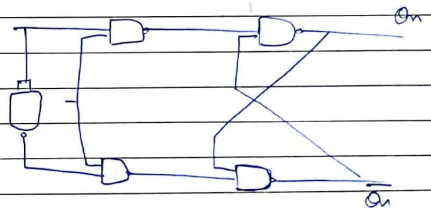
RACE AROUND CONDITION

Date

JK latch avoids the indeterminant input cond^n of SR latch by connecting $\bar{Q}_n$ and Q outputs to the Jk excitation input.

Jk latch circuit isn't practical. In this circuit the output is fed back to the input. Change in the output results a change in the input. Due to this in the +ve half of the clock pulse. If the Jk pulse, both are high (1,1) that output toggles continuously. This cond^n is called as Race around condition

D – Flip Flop



D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

→

D	Q_{n+1}
0	0
1	1

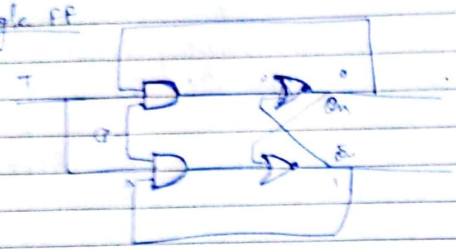
What makes you happy?
HappyCollegeDays

#
#

AND 0 → 0
NOR 1 → 0

Date

Toggle FF



T	Qn	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

T	Qn+1
0	Qn
1	$\bar{Q}n$

Date

Excitation Tables

The tables which list the required inputs for a given change of state are called as excitation tables.

1) SR

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

2) JK FF

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

J	K	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	$\bar{Q}n$

Date

3) D FF

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

D	Q_{n+1}
0	0
1	1

4) T FF

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Toggle action

0 → conditions toggle inactive

What makes you happy?
#HappyCollegeDays

Date

Conversion of flip flops

1) SRFF to Dff

T — Excitation Table I — State table

Inputs →

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

State table

D	Q_{n+1}
0	0
1	1

Excitation

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

What makes you happy?
#HappyCollegeDays

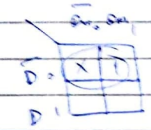
Date

Logical simplification

For S:

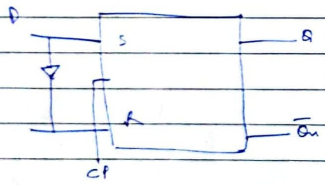
$S = D$

For R



$R = \bar{D}$

Logic Diagram



What makes you happy?

#

Date

2) SR FF & T FF

Excitation table — State Table

SR

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

TF

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

Inputs

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

K-map simplification

for S.

	$\bar{Q}_0$	Q_1
$\bar{T}_0$		X
T_1	1	

$S = T\bar{Q}_n$

for R

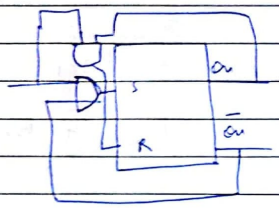
	$\bar{Q}_n$	Q_n
$\bar{T}$	X	
T		1

$R = TQ_n$

#

Date

Logic Diagram



SR FF & JK FF

Excitation | State

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

J	K	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

for S.

for R.

J	K	Qn	Qn+1	S	R
0	0	0	0	0	X
0	1	0	0	0	X
1	0	1	1	X	0
1	1	1	0	0	1

#_____
#_____

Date

K-map.

(Plot corresponding 1 and X)

J \ KQn	0 0	0 1	1 1	1 0
	$\overline{K}\,\overline{Qn}$	$\overline{K}\,Qn$	$K\,Qn$	$K\,\overline{Qn}$
$\overline{J}_0$		X		
J_1	1	X		1

$S = J\,\overline{Qn}$

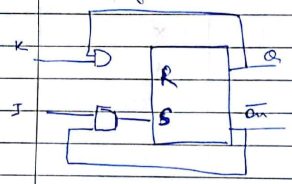
	0 0	0 1	1 1	1 0
	$\overline{K}\,\overline{Qn}$	$\overline{K}\,Qn$	$K\,Qn$	$K\,\overline{Qn}$
$\overline{J}_0$	X		1	X
J_1			1	

$R = K\,Qn$

#_____
#_____

Date

Logic Diagram



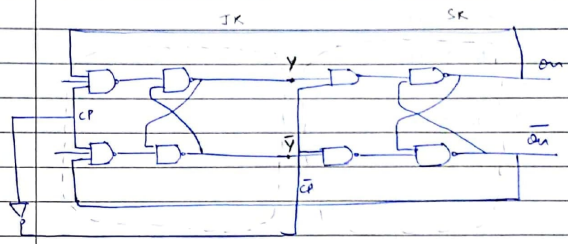
Master Slave flip flop
(JK) (SR)

The master slave combination can be constructed for any kind of flip flop. It consists of clock, J-K flipflop as master as clock SR flip flop as a slave. Clock signal is connected directly to the master but it is connected through inverter to the slave ff. Then info present at the J and K inputs is transmitted to the output of master flip flop at the +ve clock pulse and it is held there until the -ve clock pulse occurs after which it is allowed to pass through the output of slave flip flop.

What makes you happy?
HappyCollegeDays

#_____
#_____

Date



CP	Q_n	I	K	Y	Q_{n+1}
+ve	0	0	0	0	NC
-ve	0	0	0	NC	0
+ve	0	0	1	0	NC
-ve	0	0	1	NC	0
+ve	0	1	0	1	NC
-ve	0	1	0	NC	1
+ve	0	1	1	1	NC
-ve	0	1	1	NC	1
+ve	1	0	0	1	NC
-ve	1	0	0	NC	1
+ve	1	0	1	0	NC
-ve	1	0	1	NC	0
+ve	1	1	0	1	NC
-ve	1	1	0	NC	1
+ve	1	1	1	0	NC
-ve	1	1	1	NC	0

What makes you happy?
HappyCollegeDays

#_____
#_____

Unit - 2

Digital Systems

1) Set of registers (are a combo of flip flops are +ve edge triggered
2) Set of microoperations (done on register content)
3) Control signals

Register Transfer Language (RTL)

1) To notate a register we can use alphabets/numerals
2) transfer operation ←
3) seperate microoperation ,
4) Subfield of a register ( )
5) Seperation from a control func" :

control signals of R_2 being applied to R_1

Date

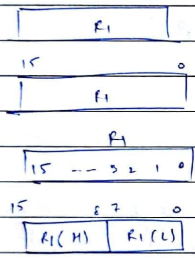
e.g. $P: R_1 \leftarrow R_2$, $R_4 \leftarrow R_3$

↳ operating microoperations

↳ i.e. if (P = 1) transfer content of R_2 to R_1

Different ways on

How to declare any register



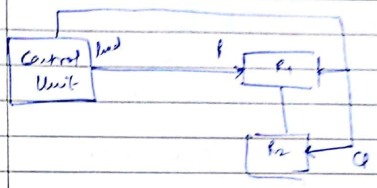
What makes you happy?

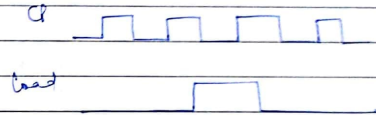
HappyCollegeDays

#

#

Date

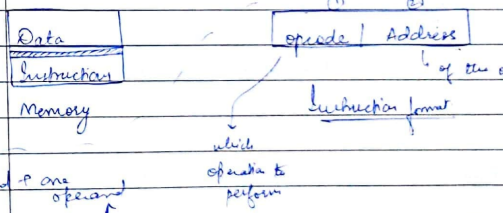




Stored Program Organization

According to this,

Data and instructions are stored together in the RAM



The 2nd operand is considered to be part of a register which is known as Accumulator (implied register)

processor

register

What makes you happy?

HappyCollegeDays

Date

one operand from instruction format
operation instruction format
another operand from accumulator

(PC/AR)
address bus
data bus (IR, DR, TR, AC)
RAM [4096x16]

12 blocks of bits 0 — 12
4096 = 2^{12} 4096 lines
4095
(memory is divided into 12 locations)

Program Counter [11 PC 0]
PC holds the address of instruction (for max bits 12 bits store hoga)
(Instruction fetch hone ke liye)

Address Reg (add of data operand) [11 AR 0]

Instruction Register (Holds the instruction [15 0]
size of all locations is 16

What makes you happy?
#HappyCollegeDays

Date

Data Register (Holds data (operand)) [15 DR]

Temporary Register (holds temporary results) [15 TR 0]

Accumulator (is a processor register) [15 AC 0]

Input Register (holds input data) [7 INPR 0]
(holds input (input is in ASCII (8bit)))

Output Register (hold output) [7 OUTR 0]

4096 words
each word holds 16 bit

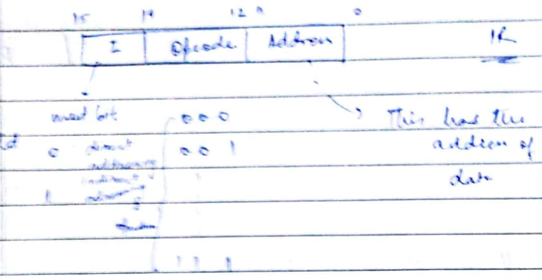
Instruction format goes through the IR, data fetched from Program Counter.

What makes you happy?
#HappyCollegeDays

instruction ka address fetched from PC, not

Adding a mod bit

X
ye address se nahi lata

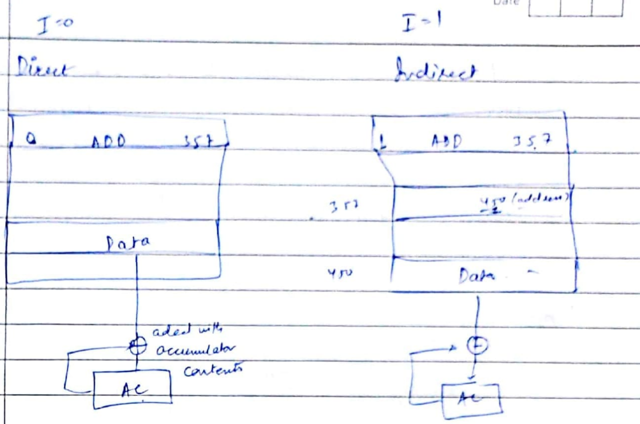


(can hold

Direct addressing (Mod bit 0) : If we go to the address then we fetch data only when the bit is 0 / go the

Indirect addressing (Mod bit 1) : If mod bit is 1, go to address location but you won't fetch data you'll get an address which in turn holds a data (kind of like a pointer addr access)

Date



Bus.

Data bus, Address Bus, Control Bus

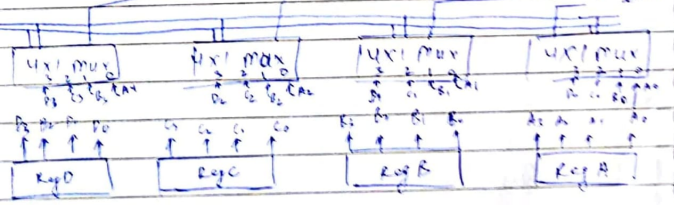
Why we Bus?

(Avoiding complexity of the architecture of so many register lines)

In a typical digital computer, we have many registers and paths must be provided to transfer info from one register to another. The no. of wires will be excessive if seperate lines are used between each register and all other registers in the system i.e why we have a common bus

A bus structure consists of a set of common lines, one for each bit of a register through which binary information is transferred one at a time.

Control signals determine which register is to be selected



S_0	S_1	o/p
0	0	Reg A
0	1	Reg B
1	0	Reg C
1	1	Reg D

The select lines of every mux is being provided

A bus system with multiplexers is registers of n bits each to produce a bus common lines. The no. of mux needed is n

Ques A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

a) How many selection inputs are each mux
b) What is size of mux needed
c) How many mux in the bus.

a) Selection inputs = 4
b) 16 x 1
c) 32

Date

Ques A computer uses a memory unit with 256K words of 32 bits each. A binary instruction has 4 parts. The indirect bit, an opcode, register code part to specify 64 registers and address part.

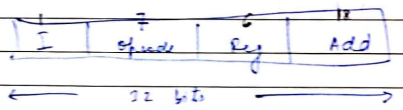
i) How many bits are there in opcode, register code and address part

ii) Indicate no of bits in each part

iii) How many bits are there in data and address inputs of the memory

Sol^n.

i)



64 registers are represented by (2^6) 6 bits

$256K = 256 \times 1024$

$= 2^8 \times 2^{10} = 2^{18}$

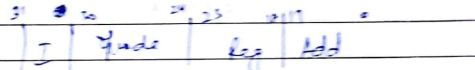
Opcode	Reg	Add
1	6	18
$32-(1+6+18)$		
$=7$ bits		

iii) 18 bits : Address bus

32 bits : Data bus

Date

ii)



Instruction Format

1) Memory Reference Instruction

2) Register Reference Instruction

3) Input Output Instruction

Computer may have instructions of several different lengths containing varying number of addresses. The no of address fields in the instruction format of a computer depends on the internal organization of its registers. It has most computers fall into one of the 3 types of CPU organization

1) 0 Address Instruction / Stack Organization

Date

$X = (A+B) * (C+D)$

Push A	Top of stack ← A
Push B	Top of stack ← B
Add	TOS ← (A+B)
Push C	TOS ← C
Push D	TOS ← D
Add	TOS ← (C+D)
Mul	TOS ← (A+B) * (C+D)
POP X	Memory[X] ← TOS

It is called 0 (Zero) because when operation is given like (add or multiply) we don't specify any address with it. Hence zero address instruction.

2) Single Address Instruction / Accumulator Organization

$X = (A+B) * (C+D)$

Instructions are LOAD (pick from memory) and STORE (place in memory)

You can specify one address in this

Accumulator | memory location of A

Date

Load A	AC ← M[A]
Add B	AC ← AC + M[B]
Store T	M[T] ← AC
Load C	AC ← M[C]
Add D	AC ← AC + M[D]
Mul T	AC ← AC * M[T]
Store X	M[X] ← AC

3) Two address instruction / General Register Organization

MOV (Register, Memory location)

$X = (A+B) * (C+D)$

MOV R1, A

MOV R1, A	R1 ← M[A]
Add R1, B	R1 ← R1 + M[B]
MOV R2, C	R2 ← M[C]
Add R2, D	R2 ← R2 + M[D]
Mul R1, R2	R1 ← R1 * R2
MOV X, R1	M[X] ← R1

What makes you happy?
#HappyCollegeDays

Date

Three Address Instruction / General Register Organization

$X = (A+B) * (C+D)$

Add R_1, A, B $R_1 \leftarrow M[A] + M[B]$

Add R_2, C, D $R_2 \leftarrow M[C] + M[D]$

Mul X, R_1, R_2 $M[X] \leftarrow R_1 * R_2$

Q- 256 K × 32

Input

Need PC, AR, IR, DR, TR, IR, OR, AC

PC 18

AR 18

IR, DR, TR, AC 32 Instruction Register 32

IR 8

OR 8

What makes you happy?
#HappyCollegeDays

Date

Addressing Mode

Specifies a rule for interpreting or modifying the address of instruction

Variety of addressing modes are supported

1. To give programming flexibility to the user.
2. To use the address field of the instruction efficiently

a) Implicit mode
b) Immediate mode
c) Direct Addressing Mode
d) Indirect
e) Register Direct
f) Register Indirect
Relative Addressing PC + Add field
Indexed Addressing
Base Register
Auto Increment / Auto Decrement

What makes you happy?
#HappyCollegeDays

Date

1) Three Address Instruction / General Register Organisation

$X = (A+B) * (C+D)$

Add R_1, A, B	$R_1 \leftarrow M[A] + M[B]$
Add R_2, C, D	$R_2 \leftarrow M[C] + M[D]$
Mul X, R_1, R_2	$M[X] \leftarrow R_1 * R_2$

Q- 256 k X 32

Fuid PC, AR, IR, DR, TR, IR, OR, AC

PC 18

AR 18

DR, TR, AC 32 Instruction Register 32

IR 8

OR 8

What makes you happy?
#HappyCollegeDays

Date

Addressing Modes

Specifies a rule for interpreting or modifying the address of instruction

Variety of addressing modes are supported

1. To give programming flexibility to the user.
2. To use the address field of the instruction efficiently by the computer.

a) Implicit mode
b) Immediate mode
c) Direct Addressing mode
d) Indirect
e) Register Direct
f) Register Indirect
Relative Addressing PC + Add field
Indexed Addressing
Base Register
Auto Increment / Auto Decrement

What makes you happy?
#HappyCollegeDays

Date

Memory (RAM)

	200	load to AC
	201	Add = 500
	202	Next instruction
PC 200	399	450
AR 400	400	700
XR 100		
AC	500	800
	600	400
	702	325
	800	300

Effective Address is the address from where we fetch the operand finally

Implicit Mode — CLA, CMA (Complement Accumulator)

Immediate Mode — (placing operand in place of Address)

Direct Add Mode —

Date

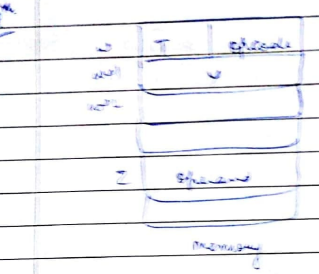
	EA	Operand
1)	X	X
2)	201	500
3)	500	800
4)	800	300
5)	X	400
6)	400	700
	202 + 500 = 702	325
	100 + 500 = 600	400

Base Reg + 500 = E.A.

Ques Two word instruction stored at address stored at w. Address field of instruction stored at w+1 designated by symbol Y. Operand used during execution of the instruction is stored at address symbolized by Z. Index register contains value X. State how Z is calculated

a) Direct
b) Indirect
c) Relative
d) Indexed

Date



Direct $E = A$

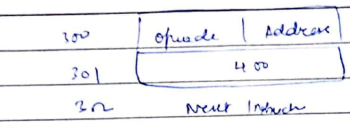
Indirect $E = [A]$

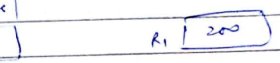
Relative $E = (PC) + A$

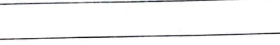
Indexed $E = A + (R)$

Ques: Instruction stored at 300. Address field 301 Address field has the value 400. Processor register R_1 contains 200. Evaluate effective address - Direct
Immediate
Relative
Register indirect
indexed (R_1 as index)

Date







Direct	400
Immediate	301
Relative	702 (PC + 400)
Register Indirect	200
indexed	600

Interrupts

multiple I/O's. When two these I/O's to transfer data / they have to tell CPU about doing the operation.

I/O communicates to CPU (Interrupt)
- External
- Internal (Traps)
- Software

External Interrupts are initiated outside of CPU & memory. Eg. power failure.

Date

I/O device data transfer

Timeout

Internal / Traps caused by currently running programs
e.g Protection violation in terms of OS
e.g Register / stack Overflow
Divide by 0
Opcode violation

Software Interrupt Internal and External interrupts are generated by hardware but software interrupts are initiated by the execution of an instruction.

e.g When you switch from user to kernel mode (Supervisor call)

Priority Interrupt (Multiple requests at the same time to be handled by the CPU)
1) Software approach (Polling)
2) Hardware approach

Self.

Register Mode
Address specified in the instruction is the register address.
- Designated operand need to be in register
- Shorter address than memory address
- Saving address field in the instruction
- Faster to acquire an operand from the memory addressing
- EA = IR(R)

Interrupts to be handled by the CPU
is handled by ISR (Interrupt Service Routine)
If hard disk sent Interrupt then we have to go to the location of that interrupt request (IRR) (Vector address)

Software polling (2-3 requests sometime) we have a CBA (Common Branch Address) It is a routine for all the devices hence the name common. They are present at a common branch address

Software method is slow but flexible (Changing priority of devices needs the routines to be changed, hence flexible)
Less costly. (no involvement of hardware)

What makes you happy?
#HappyCollegeDays

IO → Priority Output
PI → Priority Input

Date

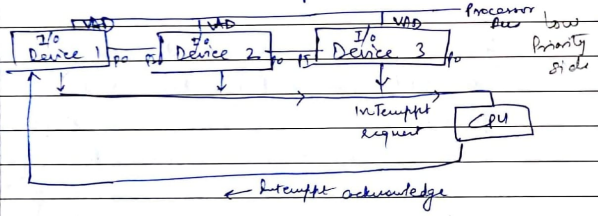
Hardware Method

b faster
b not flexible (changing the whole architecture)
b costly

2 approaches
1 Serial Hardware Approach
2 Parallel Hardware Approach

Syllabus wise only Serial Hardware Approach (Daisy chaining)

There are hardware components involved here



CPU acknowledges the processes and sends the interrupt acknowledgement signal

What makes you happy?
#HappyCollegeDays

Priorities are set according to data transfer speeds

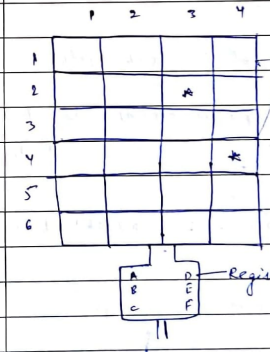
Date

Each of the device generates interrupts and passes (VAD) (the address of that IC) VAD is connected with the processor bus (address bus)

~~Eg. Device 2, Device 3 send interrupt~~

Serial fashion interrupts generating technique (Daisy Chaining)

(Topics online)
[Reduced Instruction Set Computer (RISC)
Complex Instruction Set Computer (CISC)]



RISC (software oriented approach)
LOAD A, 2x3
LOAD B, 4x4
MUL A,B
STORE 2*3;

CISC Complex instructions at the hardware

What makes you happy?
#HappyCollegeDays

PO → Priority Output
PI → Priority Input

Date

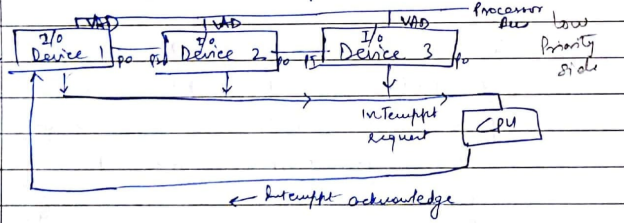
Hardware Method

- faster
- not flexible (changing the whole architecture)
- ~~was~~ costly

2 approaches
1. Serial Hardware Approach
2. Parallel Hardware Approach

Syllabus wein only Serial Hardware Approach (Daisy chaining)

There are hardware components involved here



CPU acknowledges the processes and sends the interrupt acknowledgement signal

What makes you happy?
#HappyCollegeDays

Priorities are set according to data transfer speeds

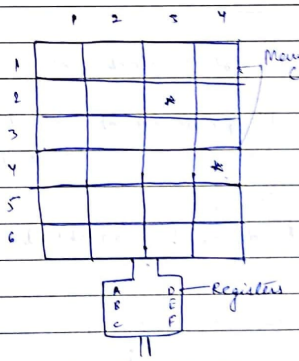
Date

Each of the device generates interrupts and passes (VAD) (the address of that ISR) VAD is connected with the processor bus (address bus)

Eg ~~Device 1, Device 2, Device 3 send interrupt~~

Serial fashion interrupt generating technique (Daisy Chaining)

Topics online)
- Reduced Instruction Set Computer (RISC)
- Complex Instruction Set Computer (CISC)



RISC (software oriented approach)
LOAD A, 2×3
LOAD B, 4×4
MUL A,B
STORE 2*3;

CISC Complex instructions on the hardware

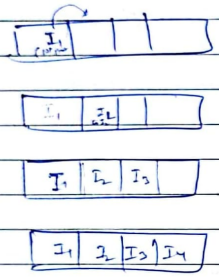
What makes you happy?
#HappyCollegeDays

Date

RISC mein PIPELINING implemented hai.
CISC mein nahi kyaakte

What is Pipelining?

Ans.



At the end of 4th clock cycle

1st process over at 4th clock cycle
2nd . . 5th clock cycle
3rd pncss

Because 1st process ke baad dusre bhi, nikalte hain

RISC pipeline because each instruction is of 1 clock cycle

What makes you happy?
HappyCollegeDays

Date

Clock cycles different hain Toh pipelining nahi hogi
Ye CISC mein hota hai
CISC mein multi clock cycle ki instruction hai

CISC microprogrammed control unit use karta hai (software)

RISC hardwired control unit (faster)

CISC addressing modes zyadda ha
RISC addressing modes kam hai

CISC no of instructions more
RISC no of instructions less

CISC no of registers less (coz more space taken by transistors)

RISC no of registers (more) (transistors less)

RISC (program code kam hota hai)

CISC Assembly code less in size thus occupying less RAM
RISC Assembly code more in size, more RAM

What makes you happy?
HappyCollegeDays

Instruction Cycle

1. Fetch the instruction
2. Decoding the instruction
3. Calculate the effective address if there is indirect addressing
4. Execute the instruction

(repeat) (has multiple instructions)
for each instruction these 4 steps will repeat

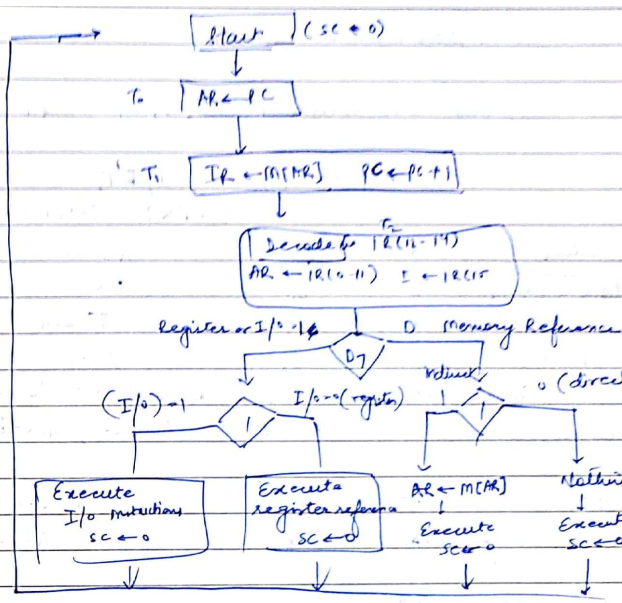
(a) Fetch and Decode:

			S_0	S_1	S_2	
Timing signal	T_0	$AR \leftarrow PC$	0	1	0	$(S_0 S_1 S_2 \; 1\,1\,1)$
	T_1	$IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$				

(decode) T_2 $D_7 \ldots D_0 \leftarrow IR(12\text{-}14)$, $AR \leftarrow IR(0\text{-}11)$, $I \leftarrow IR(15)$

↓ Opcode to decoder

↑ Mode bit



What makes you happy?
#HappyCollegeDays

What makes you happy?
#HappyCollegeDays

SC → sequence counter

Date

1) Register Reference Instruction

They are recognised when $D_7 = 1$ and $I = 0$

Control function : $D_7 I' T_3$

$B = IR(i)$ $\quad i = 0, 1, 2, \ldots, 11$

(clear accumulator)	CLA	rB_{11}	$AC \leftarrow 0$
clear E (flip flop)	CLE	rB_{10}	$E \leftarrow 0$
complement AC	CMA	rB_9	$\overline{AC} \leftarrow \overline{AC}$
complement E	CME	rB_8	$E \leftarrow \overline{E}$
Circulate right	CIR	rB_7	$AC(15) \leftarrow E$, $AC \leftarrow cir(AC)$, $E \leftarrow AC[0]$
Circulate left	CIL	rB_6	$AC(0) \leftarrow E$, $AC \leftarrow cil(AC)$, $E \leftarrow AC(15)$
Implicit	INC	rB_5	$AC \leftarrow AC + 1$
skip if accumulator is +ve	SPA	rB_4	$AC(15) = 0$, $PC \leftarrow PC + 1$
skip if accumulator is -ve	SNA	rB_3	$AC(15) = 1$, $PC \leftarrow PC + 1$
skip if accum is zero	SZA	rB_2	if $(AC = 0)$ $PC \leftarrow PC + 1$
skip if E flip flop is 0	SZE	rB_1	if $(E = 0)$ $PC \leftarrow PC + 1$
Halt	HLT	rB_0	$SC \leftarrow 0$

What makes you happy?
HappyCollegeDays

#

#

⇒ PPT lecture 14 slide 33

Date

2) Memory Reference Instructions

15	14	13	12	11	0
I	Opcode				Addr

D_0 AND.

$D_0 T_4$: $DR \leftarrow M[AR]$

$D_0 T_5$: $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$

D_1 ADD

$D_1 T_4$: $DR \leftarrow M[AR]$

$D_1 T_5$: $AC \leftarrow DR + AC$, $E \leftarrow$ cout , $SC \leftarrow 0$

D_2 LDA (Load to accumulator)

$D_2 T_4$: $DR \leftarrow M[AR]$

$D_2 T_5$: $AC \leftarrow DR$, $SC \leftarrow 0$

D_3 STA (~~Store to accum~~ Store to memory)

$D_3 T_4$: $M[AR] \leftarrow AC$, $SC \leftarrow 0$

IMP.

D_4 BUN (Branch Unconditionally) (Jump anywhere in the program)

$D_4 T_4$

$PC \leftarrow AR$

IMP

BSA (Branch and save return address)

$D_5 T_4$ $\quad M[AR] \leftarrow PC$, $AR \leftarrow AR + 1$

$D_5 T_5$ $\quad PC \leftarrow AR$, $SC \leftarrow 0$

What makes you happy?
HappyCollegeDays

#

#

Segment: Combinational circuit Registers

Date

Instruction Pipeline

Phases of an Instruction Cycle

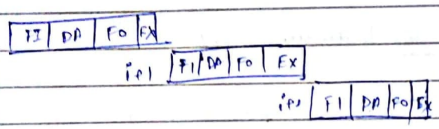
1) Fetch an instruction from memory
2) Decode the instruction (using Hardwired control unit)
3) Calculate effective address of the operand
4) Fetch the operands from memory
5) Execute the operation
6) Store the result in proper place

4 Stage Pipeline

FI	Fetch
DA	Decode
FO	Operand fetch
EX	Execute

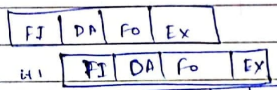
Date

Without pipelining



(12 clock cycles)

Pipelined



(6 clock cycles)

When a (branch) → jump / goto address or (interrupt) comes, → we have to go to Interrupt service Routine (ISR). It will jump to ISR.
In both these cases pipelining isn't possible

FI (Fetch)
DA (ALU)
EX (Exec)

What makes you happy?
#HappyCollegeDays

Date

Hazards of Pipelining:
3 types of problems

1. Structural Hazards - (Resource conflicts)
Hardware resources required by the instructions in concurrent overlapped execution can't be met

- FI and FO can't operate at the same time because same resource is being accessed.

2. Data Hazards (Data Dependency Conflicts)

Occurs when exec of an instruction depends on results of a previous instruction
Can be dealt with
- Hardware technique (Interlock, Forwarding)
- Software technique (Instruction scheduling compiler) for delayed load)

NOP (No operation) → wastes the CPU cycle starting
is an instruction

What makes you happy?
#HappyCollegeDays

Date

Control Hazards (when branch or interrupt like)

Branch target address is not known unless branch instruction is completed

Solutions

Prefetch Target Instruction
Branch Target Buffer
Loop Buffer
Branch Prediction
Delayed Branch

Associate memory (faster) (has a parallel search)
or
Translation look aside Buffer
Content addressable memory

What makes you happy?
#HappyCollegeDays

Date

RISC Pipeline

RISC is a machine that has a very fast clock cycle that executes at the rate of one instruction per cycle

Solves data hazard using DELAYED LOAD (mem. NOP)

Solves control hazards using DELAYED BRANCH
Compiler analyzes the instructions before and after the branch and rearranges the program sequence by inserting useful instructions in the delay step

What makes you happy?
#HappyCollegeDays

microoperations: operation performed on register

Date

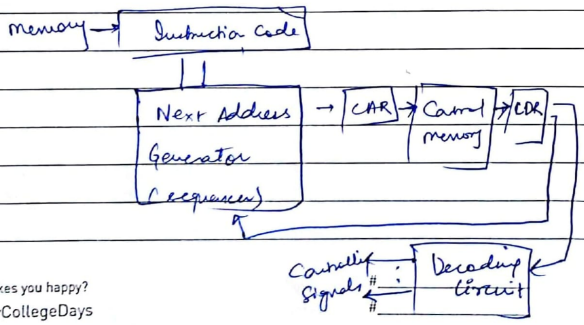
Microprogram Control Unit

Hardwired:
Control signals are generated using hardware

Microprogrammed:
- Consists of a program called microprogramme.
- Consists of microinstructions.
- It is a part of memory (ROM)
- program is responsible for generating control signals

The register in here is CAR: Control Address Register
CDR: Control Data Register

Size of CAR = 9 0 — Size 10
CDR = 19 0 — Size 20



What makes you happy?
HappyCollegeDays

Date

CDR is sometimes called as Pipeline register because The microinstruction in it which is generated microoperations generated and the next address to be placed in CAR

What makes you happy?
HappyCollegeDays

#
#