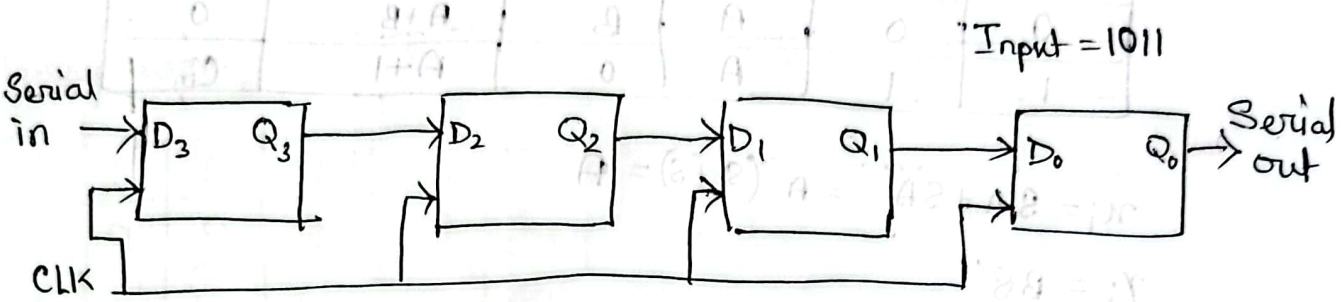
Sub :

Day

Time : Date : / /

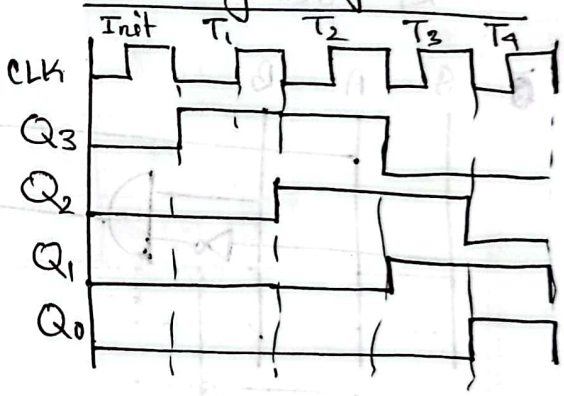
Register : n bit register → n टा flipflop → n bits binary info.

Shift register : register that shifts left right.



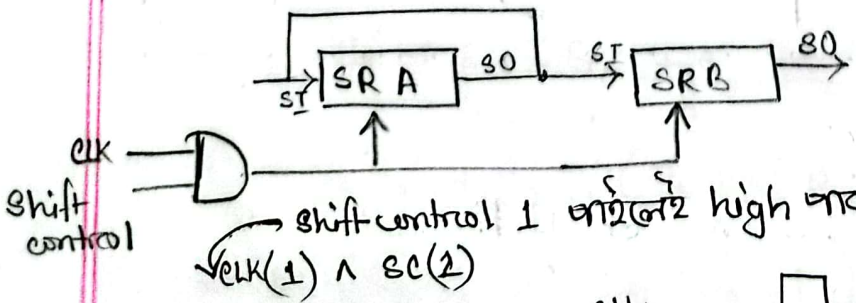
CLK	Q_3	Q_2	Q_1	Q_0
Init	0	0	0	0
T_1	1	0	0	0
T_2	1	1	0	0
T_3	0	1	1	0
T_4	1	0	1	1

Timing Diagram



Serial Transfer using Shift Register :

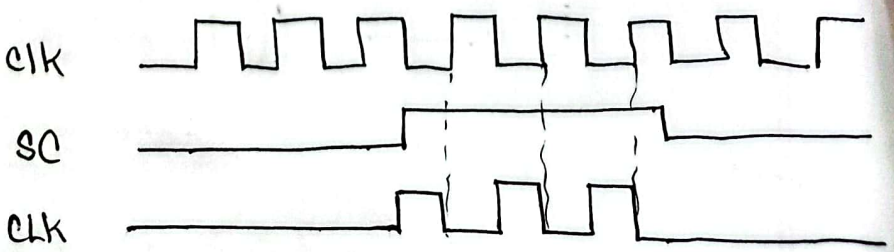
Info is transfered 1 bit at a time by SHIFTING bits out of the SOURCE REG into the DESTINATION REG.



SC input determines when and by how many times the reg are shifted.

shift control 1 असेल तर high मध्ये !

CLK(1) ∧ SC(1)



Timing pulse	SRA	SRB	SO of B
Init	1011	0010	0
T_1	1101	1001	1
T_2	1110	1100	0
T_3	0111	0110	0
T_4	1011	1011	1

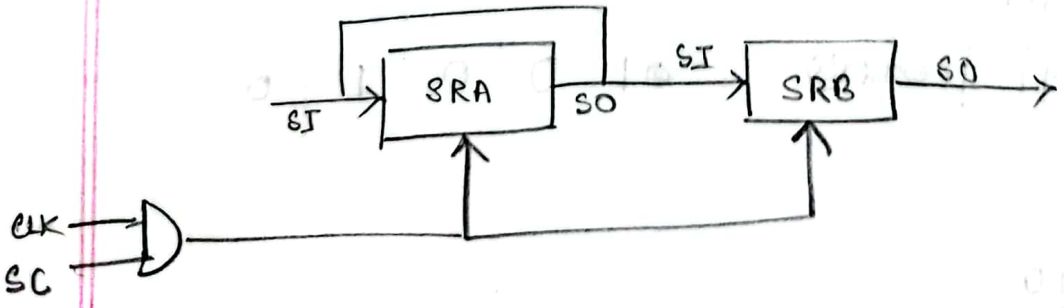
* SRA LSB SRB-তে MSB হিসেবে দিল, নিজে MSB-তেও নিল
* SRB-এর LSB SO.

4th shift এর পর SRB-তেও SRA-এর value চলে আসছে।
(4bits)

(*) nth shift এর জন্য যখন SC 0 হয়ে যায় তখন দুইটা reg-এ same value থাকে → n bit নিয়ে ~~এস~~ দেখায় দিব!

Quiz Example: SRA = 11011, B = 00101, 5th shift show →

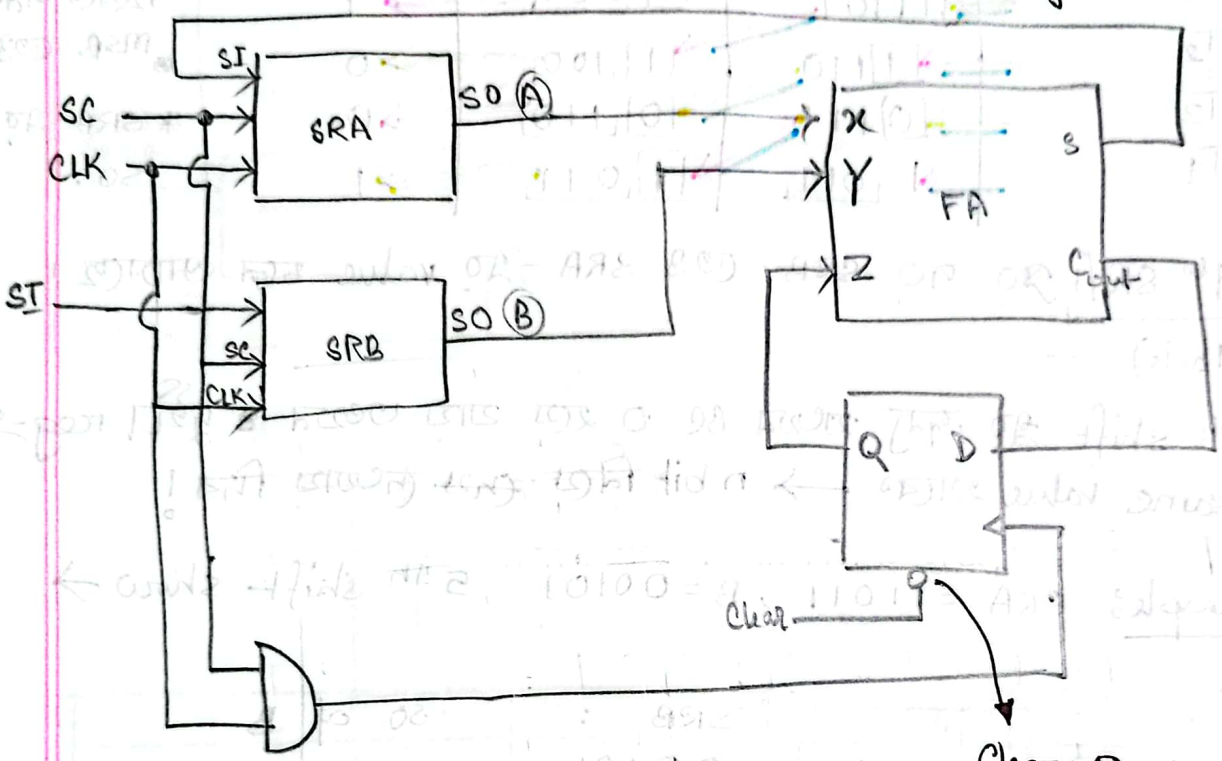
TP	SRA	SRB	SO of B.
Initial	11011	00101	1
T1	11101	10010	0
T2	11110	11001	1
T3	01111	01100	0
T4	10111	10110	0
T5	11011	11011	1



Sub: Serial adder with shift register.

Day / Time: / Date: / /

*Diff btwn SA and Parallel FA → wires IC SA-তে কম লাগে but FA-তে বেশি লাগে debugging-এ ঝামেলা



Clear=0 : মানে clear সব (Active low)

	SRA	SRB	x	Y	Z	S	C
Initial	0010	0001	0	1	0	1	0
T1	1001	000	1	0	0	1	0
T2	1100	00	0	0	0	0	0
T3	0110	0	0	0	0	0	0
T4	0011	খালি ☺	1	0	0	1	0

*SRA-তে MSB তে sum দিবে

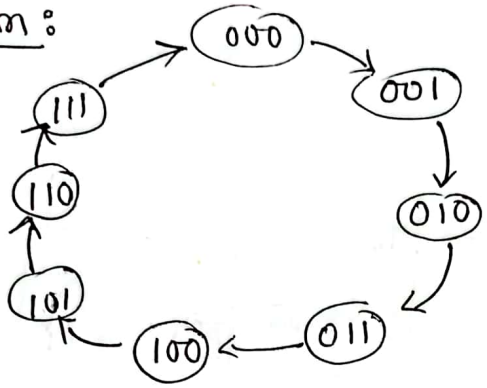
0010
(+)0001
ans: → 0011

Sub: 3 bit Binary Counter.

যেকোনো FF দিয়ে আসবে [T দিয়ে দেখানো]

Day / Time / Date

State Diagram:



1bit Excitation serial for ffs

Q	QN
0	0
0	1
1	0
1	1

D
0
1
0
1

DT
0
-
1
0

Short cuts
next state বসবে
xor-এর মতো

J	K
0	x
1	x
x	1
x	0

S	R
0	x
1	0
0	1
x	0

J = 01 xx
K = xx 10
S = 010 x
R = x 010

State Table:

A_3	A_2	A_1	N_3	N_2	N_1	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

T_3

A_3 \ A_2A_1	00	01	11	10
0			1	
1			1	

$T_3 = A_2A_1$

T_2

A_3 \ A_2A_1	00	01	11	10
0		1	1	
1		1	1	

$T_2 = A_1$

T_1

A_3 \ A_2A_1	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$T_1 = 1$

Sub :

Day

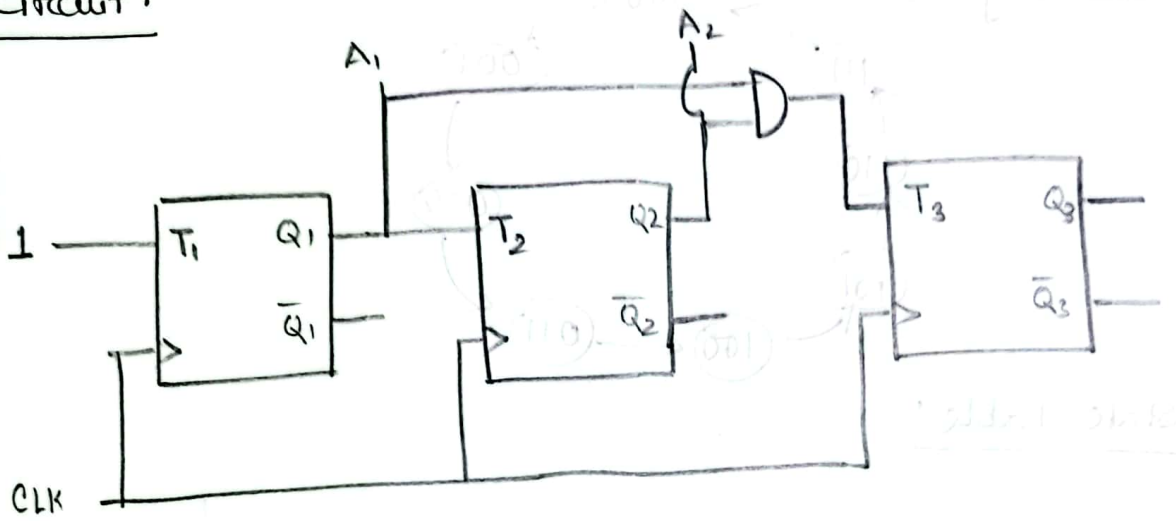
Time : Date :

Circuit :

$T_1 = 1$

$T_2 = A_1$

$T_3 = A_2 A_1$



D flipflop: কি বানাই → register (store করতে helpful).

JK flipflop:

P	N	Operation	J	K	JK
0	0	Hold	0	0(x)	00
1	1		0(x)	0	
0	0	Reset	0	(x)1	01
1	0		0(x)	1	
0	1	Set	1	(x)0	10
1	1		1(x)	0	
0	1	Toggle.	1	(x)1	11
1	0		1(x)	1	

FLIP FLOPS (~~4 bit~~ 1 bit).

Sub: Excitation table.

Day

Time: Date: / /

$Q_{t+1} = S + R'Q$

Q_t	Q_{t+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

$Q_{t+1} = JQ' + K'Q$

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

$Q_{t+1} = D$

Q_t	Q_{t+1}	T	
0	0	0	
0	1	1	Toggle.
1	0	1	
1	1	0	Toggle.

$Q_{t+1} = T \oplus Q$

Sub: Booth (MUL Algo)

ASP :

* repeatedly add one of the two predetermined values → A, S . to product P.
* then right shift P.

* m → multiplicand r → multiplier.
* x → bits in m , y → no. of bits in r.
* add + right shift বারবার হয় → y times.

Example → 13 × −6
↓ m ↓ r

ASP length → [x + y + 1] so if x = 5 bits, y = 5 bits then Length = 6.

For A : MSB তে m-এর value বসবে, rest (y+1) bits = 0.

For S : " " −m " " " " " = 0.

For P : MSB x bits = 0 ; merge r at right side ; rest = 0

*** rs-এর পর দেখা LSB তে 01 হলে P+A, 10 হলে P+S, 00 / 11 হলে do nothing, next step

STEPS Y times করব ***

let, 13 × −6 ; x = 5 , y = 5.

m = 01101 ; −m = 10010.
r = 11010

A = 01101 000 000 (5 bits, 6 bits)

S = 10010 000 000

P = 00000 11010 0 (5 bits, 5 bits, rest)

Step 1 : P = 00000 11010 0
(rs) → 00000 01101 0

Step 2 : P = 00000 01101 0
S = 10010 00000 0 (+)
10010 01101 0
(rs) → 11001 00110 1

Step 3 : P = 11001 00110 1
A = 01101 00000 0 (+)
~~1~~ 00110 00110 1
00011 00011 0

Carry থাকলে (overflowed) ignore করব

Step 4 : P = 00011 00011 0
S = 10010 00000 0 (+)
10101 00011 0
11010 10001 1

Step 5 : P = 11010 10001 1
~~A = 01101 00000 0 (+)~~
~~1 00111 10001 1~~
(rs) → 11101 01000 1 [Final prod]

Sub ::   Day / Time : / Date : / /

ASP method [* Find the product of 13×-6 where x=5bits, y=5bits.
or ll " " " 13×-6 where both m, r have 5bits

UV method [* Desig 5x5 booth's multiplier where m=13, r=-6 or 13x-6.

UV method : Best ; Accurate ans হয় ***

x = 13 → 01101 (13)

y = 5 → 11010 (-6)

∴ -y → 0.0110 (6)

(x-এর LSB হবে আগের → x-1)

00 → 0+U
01 → +y+U
10 → -y+U
11 → 0+U.

	U	V	x	x-1	
Initial	00000	00000	01101	0	
	(+) 00110				
	00110	00000	00110 (right shifted)	1	step-1
rs →	00011	rs → 00000			
	(+) 11010				
	11101	00000	00110	1	
rs →	11110	rs → 10000	rs → 00011	0	step-2
	(+) 00110				
carry বাদ →	00100	10000	00011	0	
rs →	00010	rs → 01000	rs → 00001	1	step-3
	00010	01000	00001	1	
rs →	00001	00100	00000	1	step-4
	(+) 11010				
	11011	00100	00000	1	
rs →	11101	10010	00000	0	step5

Final result = 11101 10010

Sub :

Day

Time : Date : / /

Hardware design (uv)

① Initialization :

$U \leftarrow 0$

$V \leftarrow 0$

$x \leftarrow$ input (m)

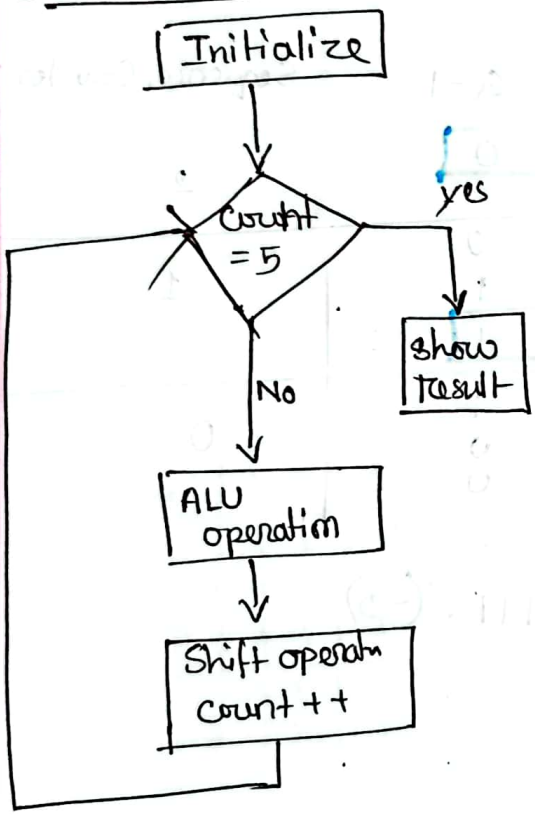
$Y \leftarrow$ " (n)

$x-1 \leftarrow 0$

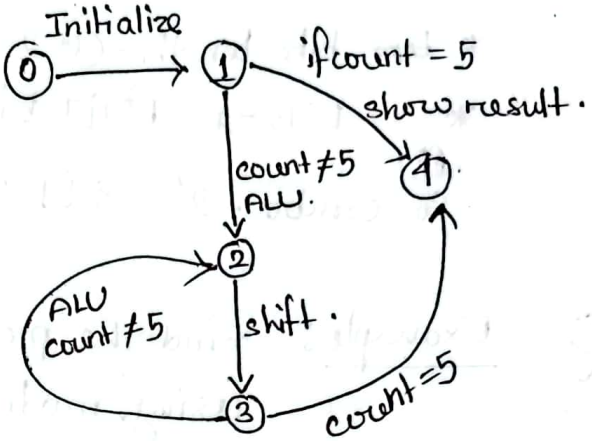
count $\leftarrow 0$

count ++ (y times).

② Flowchart :



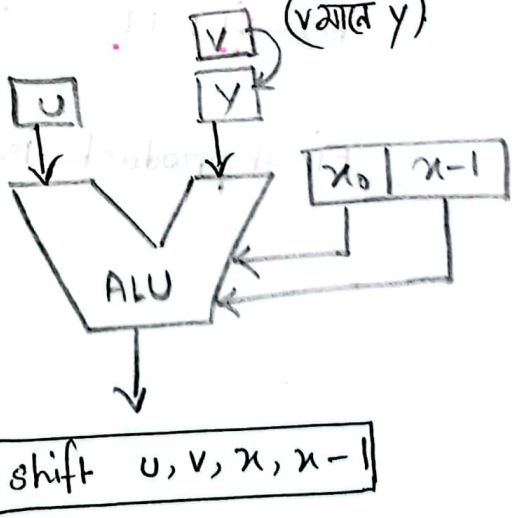
③ State Diagram :



④ Architecture :

ALU operation

(x) S_1	$(x-1)$ S_0	operation
0	0	U+0
0	1	U+Y
1	0	U−Y
1	1	U+0



Sub :

Day

Time : Date : / /

ASP, UV – তে Problem বেশি iteration হইলে time-space complexity বেড়ে যায় !

Solution : modified booth's algo.

* ~~bit~~ bits length-কে 2 দিয়ে ভাগ করে নেয়
* LSB-র সাথে আরো 1 bit নিয়ে কাজ করে

So, combo. $2^3 = 8$ টি (0-7)

i+1	i	i-1	operation	
0	0	0	0×m	0
0	0	1	+1×m	m
0	1	0	+1×m	m
0	1	1	+2×m	2m
1	0	0	-2×m	-2m
1	0	1	-1×m	-m
1	1	0	-1×m	-m
1	1	1	0×m	0

নোট

Example : Find the prod of $\overbrace{1101}^{m} \times \overbrace{0011}^{r}$ using modified booth's algo.

Sol:

$-m = 0011$ (2's comp)

n = 4 no. of bits

operation	A	Q (r বসাও)	Q-1	Sequence Counter = n/2 = 4/2 = 2
-1×m	0000	00 11	0	
	0011			2
	0011	0011	0	
rs1 →	0001	1001	1	
rs2 →	0000	1100	1	1
+m	1101			
	1101	1100	1	
rs1 →	1110	1110	0	0
rs2 →	1111	0111	0	

∴ Final product is 1111 0111. (-9)

Sub :

বড়
10bit.

Example: 111 0011 011 × 0000 101001
(m) (r)

So, m = 1110011011 (−101)

−m = (0001100100 + 1) = 0001100101 (101)

r = 0000101001 (41)

2m = 1100110110 [Left shift করলে twice value পেয়ে যাই] shortcut

−2m = 0011001010.

SC = 10/2 = 5

	A	Q	Q−1	SC
	00000 00000	00001 01001	0	5
+1×m	11100 11011			
	1110011011	00001 01 001	0	4
rs1→	1111001101	10000 10 100	1	
rs2→	1111100110	1100001010	0	
−2×m	0011001010			
	×0010110000	1100001010	0	3
rs1→	0001011000	0110000101	0	
rs2→	0000101100	0011000010	1	
−1×m→	0001100101			
	0010010001	0011000010	1	2
rs→	0001001000	1001100001	0	
rs2→	0000100100	0100110000	1	
+1×m	1110011011			
	1110111111	0100110000	1	1
rs1→	1111011111	1010011000	0	
rs2→	1111101111	1101001100	0	
	1111101111	1101001100	0	0
rs1→	1111110111	1110100110	0	
rs2→	1111111011	1111010011	0	

Final result: 1111111011 1111010011

Sub : MEMORY DEVICE

Day
Time : Date : / /

Memory Device :
* যেখানে Bin info store
বাখিস [Device – যন্ত্র] MUST
* যেটা থেকে যখন দরকার
তখন info collect করে নেই.

Memory Unit :
* collection of cells capable of storing large quantity of bin info.

Types : RAM – Random Access Mem (Read Write both).
ROM – Read Only Mem

* non volatile – off করে দিলেও stored থাকে [example- storing OS]
* Fixed set of bin info stored.
* info গুলা user দ্বারা specified হয়ে আসে
* প্রকার info will be embedded in the unit to form interconnection pattern
* includes both DECODER and OR GATES in an IC package.
* Func. implementation – programming দিয়ে করি
* eliminates extra wires.
* প্রকারভাব IC -তে set করে ফেললেও power off করলেও connection fixed থাকে।

*** ROM might have fused or broken internal links.

ROM DESIGN : 32×8 ROM, or 256-bit ROM

$2^n \times m$ → output line
↓
input line

information needed : 32×8 ROM – এ ($2^5 \times 8$)

$n = 5$ [n input]
$m = 8$ [m output]
words / distinct address $= 32$ $[2^n]$
bits in each word $= 8$ [m]
range $= 00000 - 11111$
0 31
→ total 32 টা

Address : each bit combination of the input variables.
Word : " " " coming out of output lines.

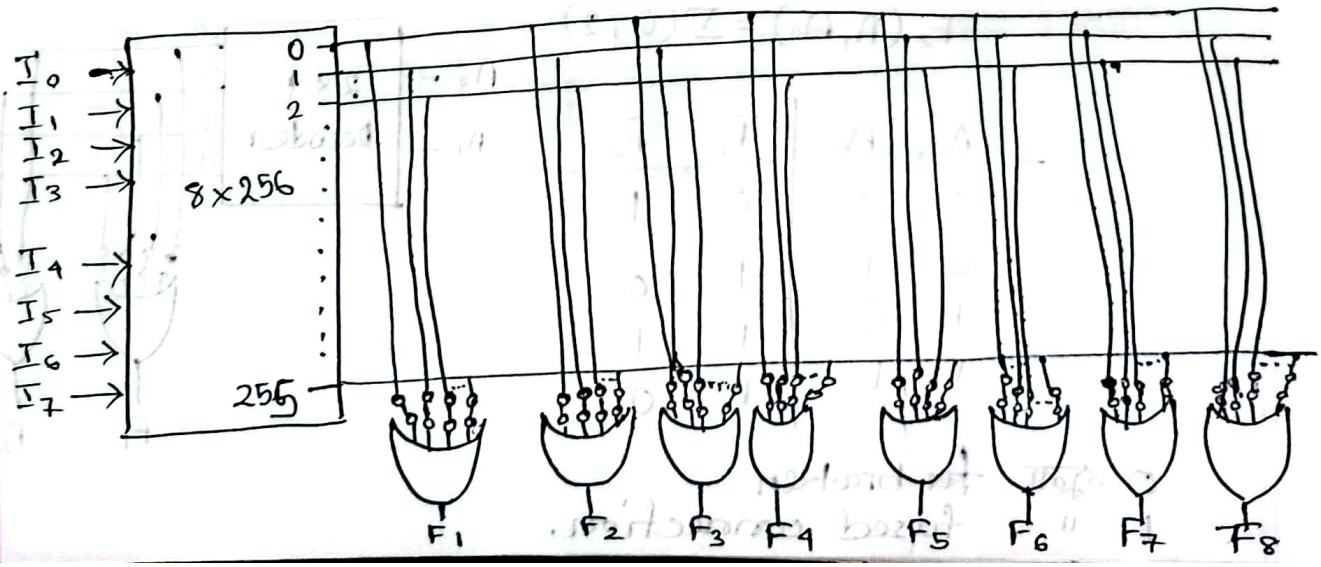
Maths :

Steps : ① bit দেওয়া থাকলে word size দিয়ে ভাগ করব
② ভাগ করে যা পাব সেই x word size ROM design করব
③ Info বের করব → n = ? ; m = ? ; words/distinct address কয়টি?
bits in each words = ? ; range = ?
④ Design.

Example: 2048-bit ROM having word size 8 bits each.

So, ① 2048/8 = 256.
② So 256 x 8 ROM design or $2^8 \times 8$ ROM.
③ n = 8 m = 8 words = 256, each words bits = 8
range = $(0000\ 0000 - 1111\ 1111)_2$ or $(0 - 255)_{10}$

④ Design :



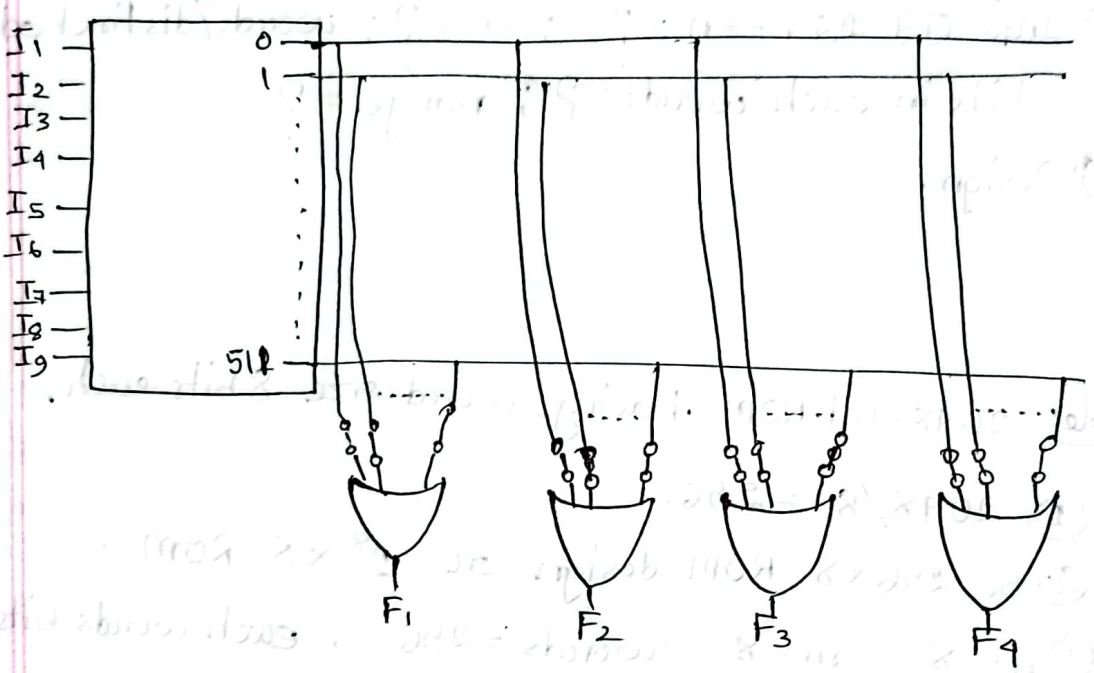
Example: 2048-bit ROM having word 4 bits each.

2048/4 = 512

So, 512 × 4 ROM or $2^9 \times 4$ ROM.

n = 9 m = 4 words = 512 bits per word = 4.

range = 0000 – 1111.



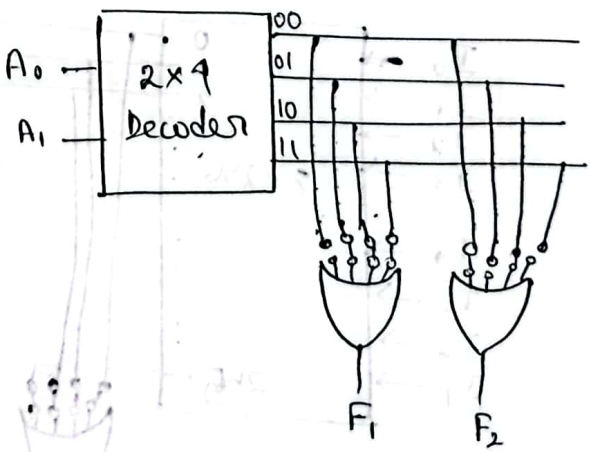
Another type of Math: Draw AND-OR Gates ROM from given info.

Given, $F_1(A_1, A_0) = \Sigma(1, 2, 3)$

$F_2(A_1, A_0) = \Sigma(0, 2)$

A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

Diagram:



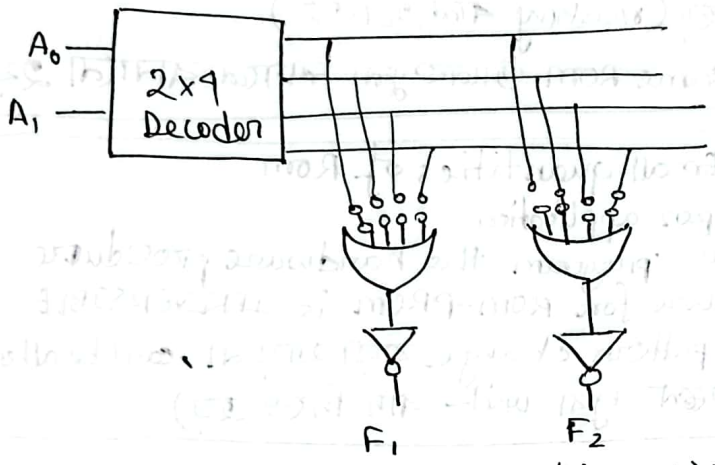
0 হলো ~~fa~~ broken

1 " fused connection.

But যখন ROM with AND-OR Invert Gates বলবে

0 → fused হবে
1 → broken "

∴ output line - এ not gate add হবে



Example: Draw a ROM that accepts 3bit input and generates output that are square of outputs.

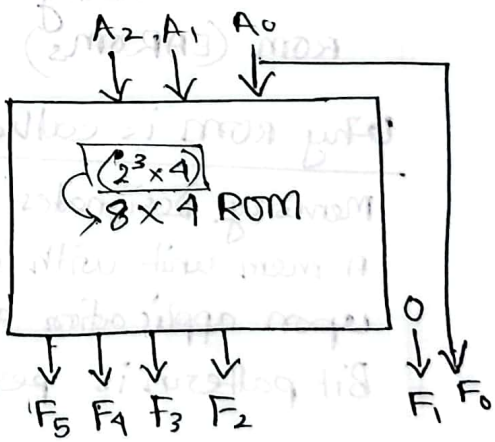
⇒ Truth table:

(6bit cause last input 7 are $7^2 = 49 \rightarrow 49$ binary-তে 6bit লাগে)

dec	A_2	A_1	A_0	F_5	F_4	F_3	F_2	F_1	F_0	dec
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	1	0	0	4
3	0	1	1	0	0	1	0	0	1	9
4	1	0	0	0	1	0	0	0	0	16
5	1	0	1	0	1	1	0	0	1	25
6	1	1	0	1	0	0	1	0	0	36
7	1	1	1	1	1	0	0	0	1	49

0 → = A_0

ROM:



Sub :

Day

Time : Date : / /

Types of ROM :

① mask programming

② PROM (Programmable)

Manufacturer makes it during last fabrication according to the customer's truth table.
Output – designed হয় (masking বলে এটাকেই)
WHEN MADE : – same ROM অনেকগুলা লাগলে বানানো হয়

*Economical for small quantities of Rom.
* broken links – as per application
*PROM programmers program the hardware procedure.
* hardware procedure for ROM-PROM is IRREVERSIBLE
(একবার করলে pattern change করা যায় না, can't be altered
change করতে হলে পুরা unit বাদ দিতে হয়)

Erasable PROM :

(5-10) minutes / given period of time – short ultraviolet radiation দিলে – ভেতরের info মুছে দেয়।
ROM Reset Renew হয়ে গেলে reprogram করা যায়।
or Electrically - এটা করা যায় Electrically Alterable ROM (EAROMs)

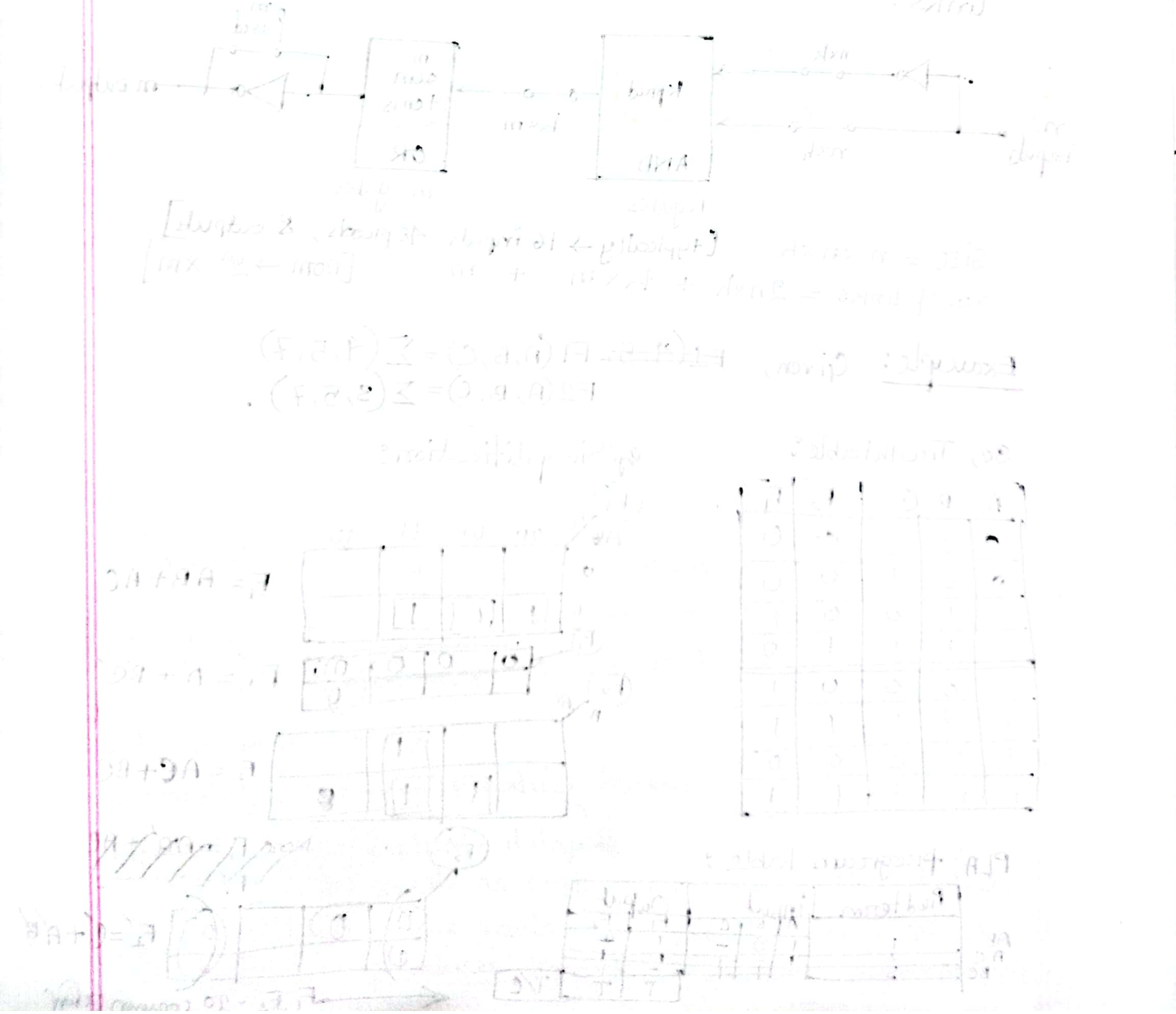
Why ROM is called Rom ?

Memory Designates a storage unit.
A mem unit with a fixed word pattern that can be read out upon application of a given address.
Bit pattern is permanent can't be chayed during normal optuh

Sub: Usage of ROM

Day
Time: Date: / /

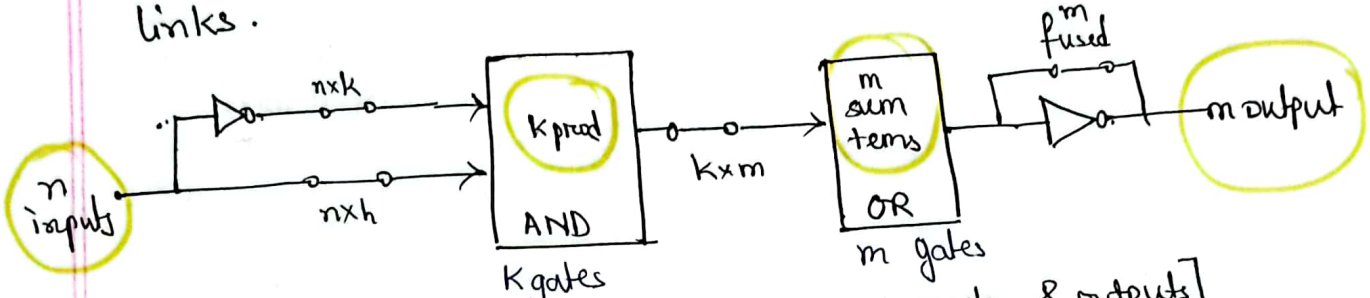
* Implement complex combo circuit from TT.
* Converting one bin code to another (ASCII to EBCDIC etc)
* For arithmetic functions — exple: multipliers
* Displaying characters in a cathode ray tube
* For applications with a large no. of input-outputs
* In Digital system's control units (microprogrammed CU).



① mask programmable [PLH program table लगेगा]
② Field " [FPLA] → like PROM

Sub: PLA (Programmable Logic Array).

* logic design with programmable AND-OR gates.
* eradicates too much 'Don't care usage' problem.
* ROM - से बेहतर but [doesn't generate all MINTERMS. no full decoding.]
* Replacement of decoder → by AND gates
* Fn implemented in sum of Prod (SOP) [minterms]. by fused/broken links.



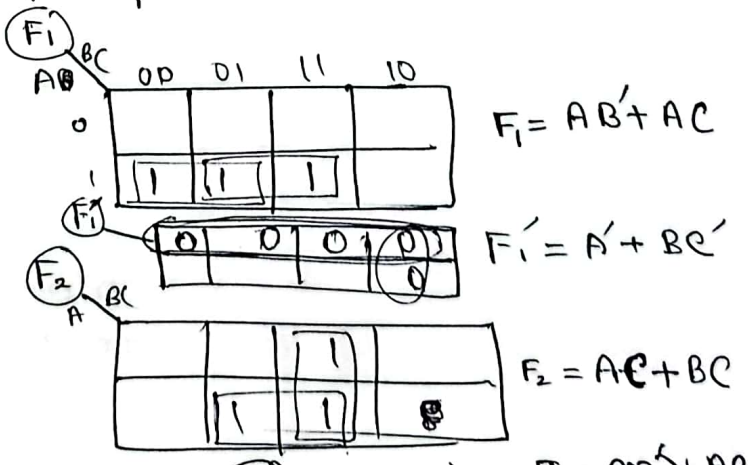
Size = m×n×k [typically → 16 inputs, 48 prods, & outputs]
no. of links = 2nxk + k×m + m. [ROM → 2^n × m]

Example: Given, ~~F1(4,5,~~ $F1(A,B,C) = \Sigma(4,5,7)$
$F2(A,B,C) = \Sigma(3,5,7)$.

So, Truthtable:

A	B	C	F_2	F_1
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Eq Simplification:

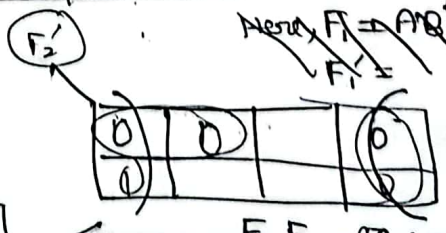


$F_1 = AB' + AC$

$F_1' = A' + BC'$

$F_2 = AC + BC$

Hence, $F_1 = AB' + AC$
$F_1' =$



$F_2' = C' + A'B'$

F_1, F_2 - को common लेकर

PLA program table:

Prod term	input A	input B	input C	output F_1	output F_2
AB' (1)	1	0	–	1	–
AC (2)	–	1	1	–	1
BC (3)	1	1	–	1	1
				T	T

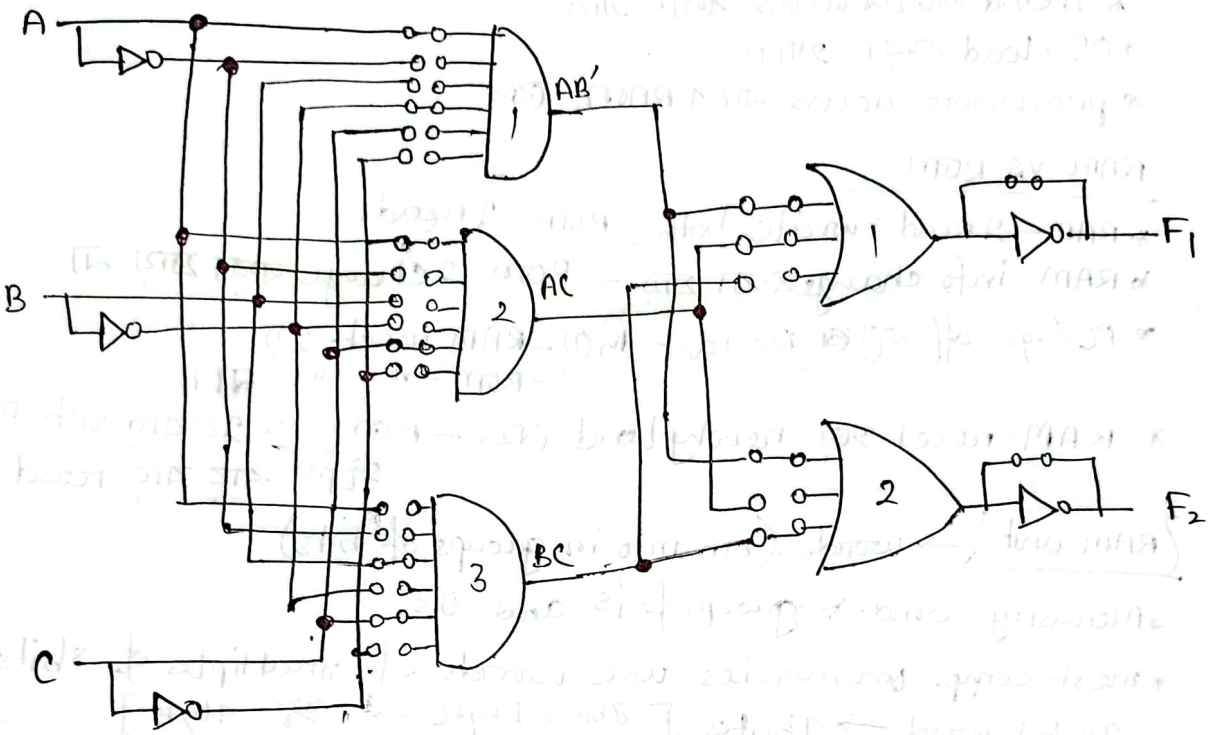
T/C

Sub :

Day

Time : Date : / /

Diagram:



Designing Digital system with PLA.

* reduced no. of distinct prod terms.
* no. of literals don't matter — we have all input variable.
* both Truth + complement (F and F') are simplified.
* Fewer prod terms.
* provides common product terms.

Application :
- [*] controls datapath.
- [*] used as counter, decoders, Bus interface.
- [*] defines various states in an intrem. set and produces next state [by conditional BRANCHING]

Sub (Memory and PL) RAM

Day
Time : Date : / /

~~Memory Device :~~

* যেখান তথ্য access করা যায়
* OS · load করা থাকে
* processor access করে RAM-তে

RAM vs ROM

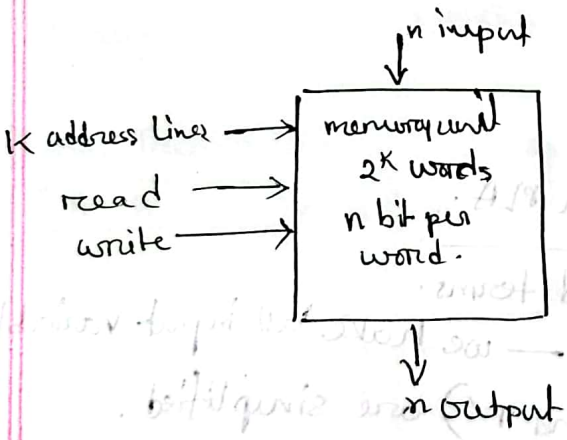
* RAM-এ read-write both, ROM-এ read.
* RAM-info change করা যায় — ROM-এ change করা যায় না
* PC/sys · off করলে Restart দিলে-RAM reset হয়
  -ROM-" " না।
* RAM-reset হয়, newly Load নেয় — ROM-এ একবার set করে দুইবার বার বার read করি

RAM unit → words (bin info in groups of bits).

*Memory word → group of 1's and 0's.

*most comp. memories use words of multiples of 8bits.

32 bit word → 4 bytes. [8 bit = 1 byte → 32/8 = 4 bytes]



if K=2.

$2^K = 2^2 = 4$ words.

$2^{(\text{address line})}$ = words.

addresses 0 up to $2^K - 1$ can be found

suppose, → 10 bit address line
1024 x 16 memory (how many bytes is this memory module).

= 16384 bits

= (16384/8) = 2048 bytes. = 2KB.

Sub :

Day

Time : Date : / /

* Math-এ address line বলে দিবে → (k) m বলা থাকবে
→ each address-এ word.

So → $2^k \times m$

= ☐ bit

= (☐ / 8) . bytes.

range = $0 - (2^k - 1)$

Static RAM

① internal latch (flipflop)

② Clock pulse লাগে (voltage দেওয়া লাগে → high power consume করে)

③ flipflop বেশি লাগে (low density)

④ expensive

⑤ faster

Dynamic RAM

② Capacitor

② Voltage দেওয়া লাগে না (low power consume করে)

③ Capacitors অনেকগুলো লাগে (high density)

④ cheap

⑤ বারবার charge করা লাগে — time loss হয় তাই slower.

Volatile

① Info is lost when power's off

② RAM

③.

Non-volatile

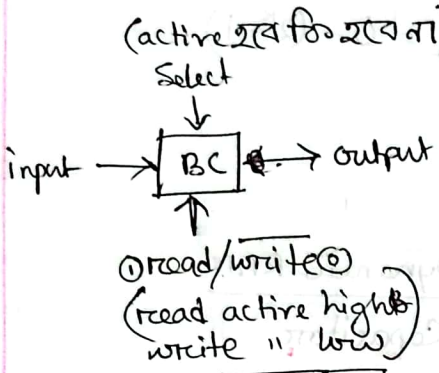
① Info retains even if power's off

② ROM (set to their values once and after that are read)

③ CD: a piece of polycarbonate where a spiral track is impressed — which is a series of indentations (pits) seperate by flat areas (land)

④ Magnetic disks: stored data by direction of magnetization

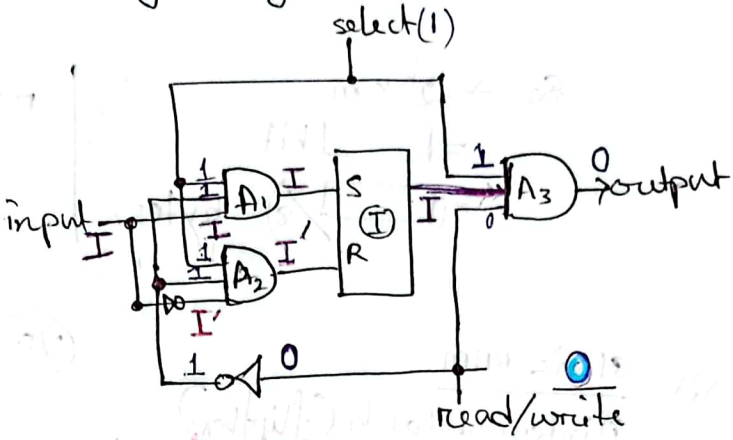
RAM Block diagram:



SR latch:

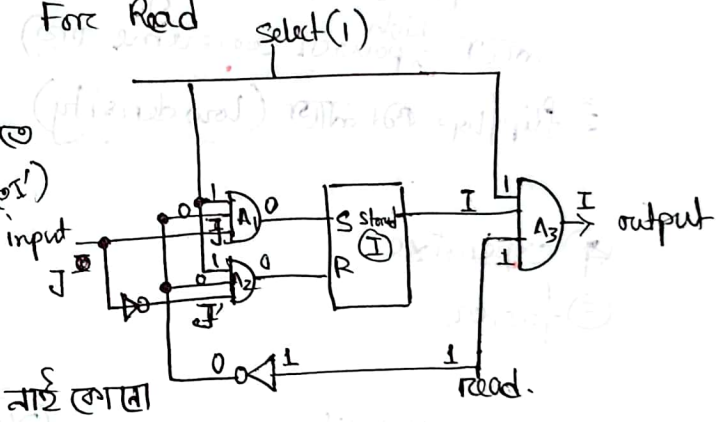
	I	0
0	1	0
1	0	1
0	0	No change

Logic diagram: For Write



write: write থেকে 0 যাবে A_3
আবার not gate-1 হয়ে $A_1 A_2$ তে
select 1 পাঠাবে $A_1 A_2$ তে
input I পাঠাবে $A_1 A_2$-তে (A_2-তে I')
$A_1 A_2$ থেকে I, I'. SR-এ যাবে
I store হবে A_3-তে যাবে।
A_3 তে 1.0.I → so 0 বের হবে।
latch-এ I store হলো output নাই কোনো।

For Read



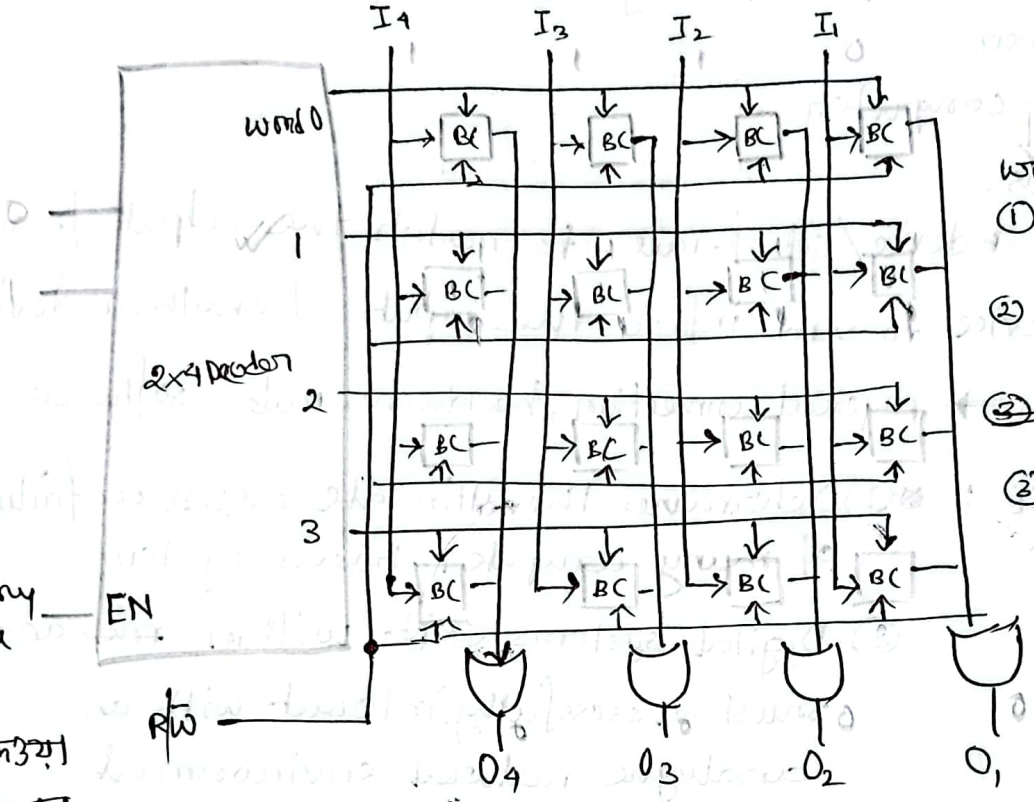
Read: read থেকে 1 যাবে A_3 তে আবার not gate-এ 0 হয়ে $A_1 A_2$ তে।
select- আবার 3 '1 পাঠাবে $A_1 A_2$ তে।
input J পাঠাবে A_1, A_2-তে (A_2-তে J').
$A_1 A_2$ 0 পাঠাবে SR-কে cause (1,0,J) and (1,0,J') input respectively.
SR-এ 0,0 গেলে → 'no change' but latch-এ I stored ছিল।
So, A_3 তে (1, I, 1) যাবে — output হয়ে I বের হবে।

row ≡
column ||||

4×4 RAM ($2^2 \times 4$ RAM)

8×4
16×3

$2^2 \times 4$ যেহেতু addressline 2টি



memory enable
↓
must দেওয়া লাগবে নাহলে কাজ করবে না

address = 01
data = 1110

write :
① address decoder → দিয়ে active করব BC.
② input store করবে যেই address → এ as soon as $\overline{w}$ signal দিবে
③ ~~output~~
③ output 0.

Read
① address decode করব
② BC-তে যা থাকবে তা output → এ show

*** Commercial RAM → RAM consists of thousands of words with each word (1-64) bits.

Sub : Interfacing.

Day
Time : Date : / /

Place where independent systems communicates with each other

Human – computer.
↓
OS.

Interface → device/set of rules to match the output of one device to send info to the input of another device

can be → physical connection, hardware, rules, softwares, process

IMPORTANCE : ① Determines the ultimate success or failure of many computer-based system.

② Digital systems exist within and must successfully interact with an analogue natural environment

IO → to standardize interfaces to devices

Unreliable devices → media failures + transmission errors

Unpredictable/slow " →

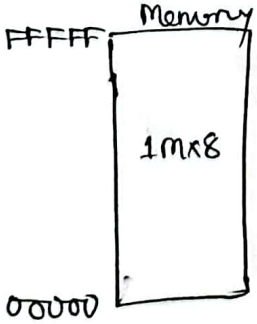
CPU + IO interfacing (2ways)

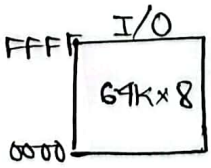
① Isolated I/O / port mapped I/O (PMIO) / I/O mapped I/O.

→ locations are isolated from memory

→ desidered I/O port is selected by I/O address

→ transfered (data) btw I/O and μprocessor is accessed by IN 01 , OUT 02 (port এ্যাড্রেস accumulator register এর)





5 ta bit address দেওয়া থাকলে memory address বুঝাবে

4 bit দেওয়া থাকলে বুঝাবে I/O-র address

* আলাদা address bus থাকে
" data " "

Sub :

Day

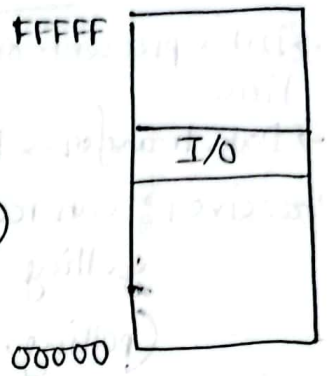
Time : Date : / /

② Memory mapped I/O (MMIO)

→ একটাই address bus use হয়

→ memory-র instruction I/O-তেও ব্যবহার করতে পারব to access CPU

→ register-ও same (I/O-র জন্য use করা যাবে)



MMIO	VS	PMIO
① devices are mapped to same memory address space		① Devices are mapped in different spaces dedicated address space
② Hardware expensive		② Cheap
③ Bus are same. (address bus)		③ Different address bus are used.
④ Same registers, instructions are used to access the CPU.		④ Different. like IN and OUT is used for 8086 MProcessor

Parallel Data Transfer [একবারে একসাথে transfer করি – serial-র reachat a time]

(4 types)

① Simple I/O:

→ receiver (CPU): Data is always present, can be r/w at anytime

→ CPU sender: receiver is always ready

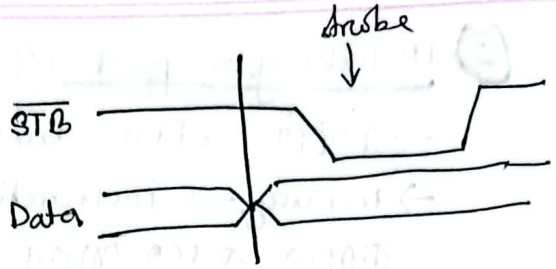
Limit:

* all devices are not always ready.

Not time dependent.

Exple → ① LED output
② reading from switch

② Simple Strobe I/O :



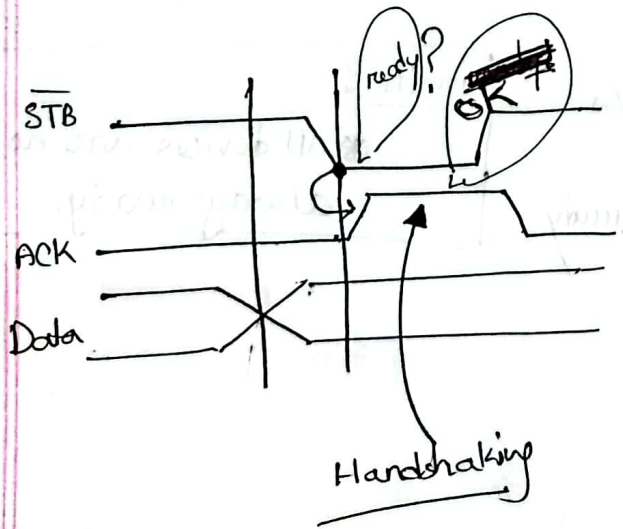
→ Data present only at a certain time.
→ Data transfer → time dependent
receiver : can read only after getting a strobe signal (polling, interrupt)
exple → keyboard.
Sender (msg) strobe দিয়ে বলে receiver-কে data পাঠানোর আগে
✓ polling → continously check করে receiver যে sender strobe পাঠিয়েছে কি না
✓ interrupt → একটা কাজের মধ্যে অন্য data receive করা start করে (priority).
exple → keyboard

limits :
receiver-কে strobe দেয় but sender-এর বুঝার উপায় নাই receiver ready কি না

Works well for low rate data transfer tho.
Sending-receiving speed differs.

③ Single-handshake I/O / Strobed I/O :

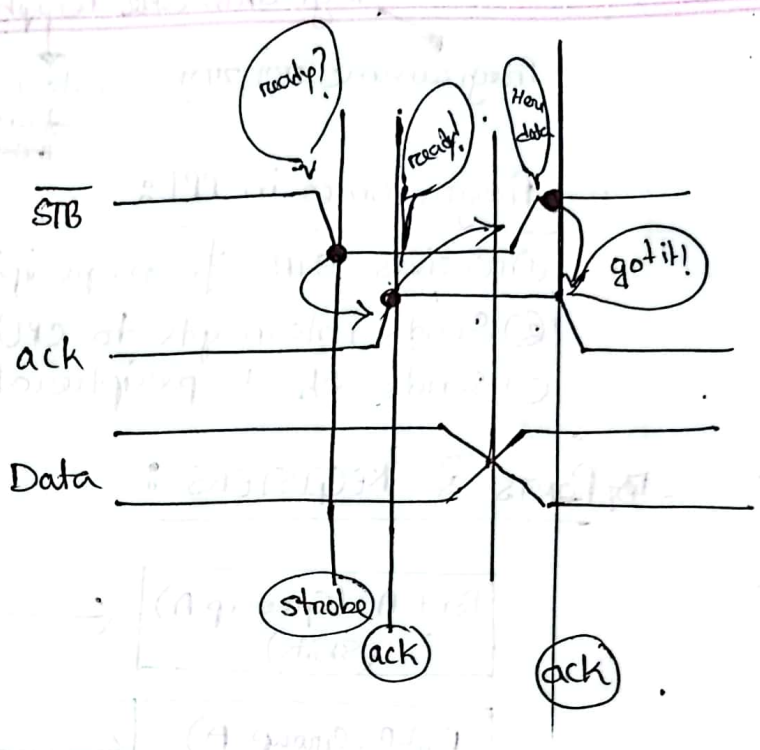
→ Strobe-পেয়ে receiver ack. পাঠিয়ে দেয়



limits :
Sender still doesn't know if receiver is ready
অনেকগুলো devices ready receive করার জন্য multiple connection-এর ক্ষেত্রে sender still doesn't know who is ready - data bus get blocked

④ Double-handshake I/O:

→ more coordination
→ Data is put when receiver ready
~~→ Strobe is sent by sen~~
→ sender : strobe দেয়
receiver: জানায় যে সে ready
sender : data পাঠায়
receiver: ack করে যে পাইছে



Implementation:

* Direct : connect CPU and I/O devices
* CPU determines when to send/receive
  - → polling (continous check
  - → interrupt (priority বেশি যেটার হলে আগে করে)
* polling vs interrupt :

Depends on application requirements.
→ polling - CPU service provide করে ; interrupt - I interrupt handler
→ polling regular intervals - হয় ; interrupt can occur anytime
→ Ready bits indicates polling service ; request line indicates interrupt service
→ Polling wastes CPU cycle ; interrupt does not

But interrupt is better as the CPU can focus on other tasks — it can reduce latency

Should CPU do the handshakes ?

→ NO, we will connect an interfacing device to CPU and I/O. → Intel 8255

*** μprocessor

Sub: Intel 8255 (PPI)

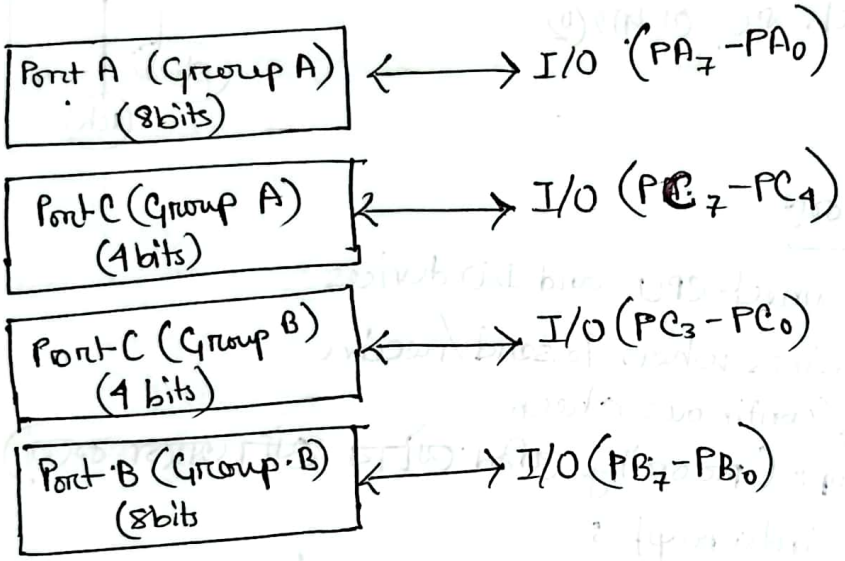
Programmable Peripheral Interface

Programme করা যায় | external ~~device interface~~ | connect করতে

Programmes in PPI:

① receives $\overline{STB}$ from peripheral
② Sends interrupts to CPU
③ sends ack to peripheral at proper time.

PORTS & REGISTERS:



***
I/O pin = 24.
ports = 3 (A, B, C).
port A, B → I/O
port C → I/O + handshake signals.
~~port C~~ (Group A) port A+
↳ upper part of port
~~and port~~
(Group B) port B+
↳ lower port C

PORT SELECTION:

আগের page-এ diagram

$\overline{CS}$	A1	A0	Selected port
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	Control Register
1	x	x	8255 not selected

Sub : Read/Write Control logic

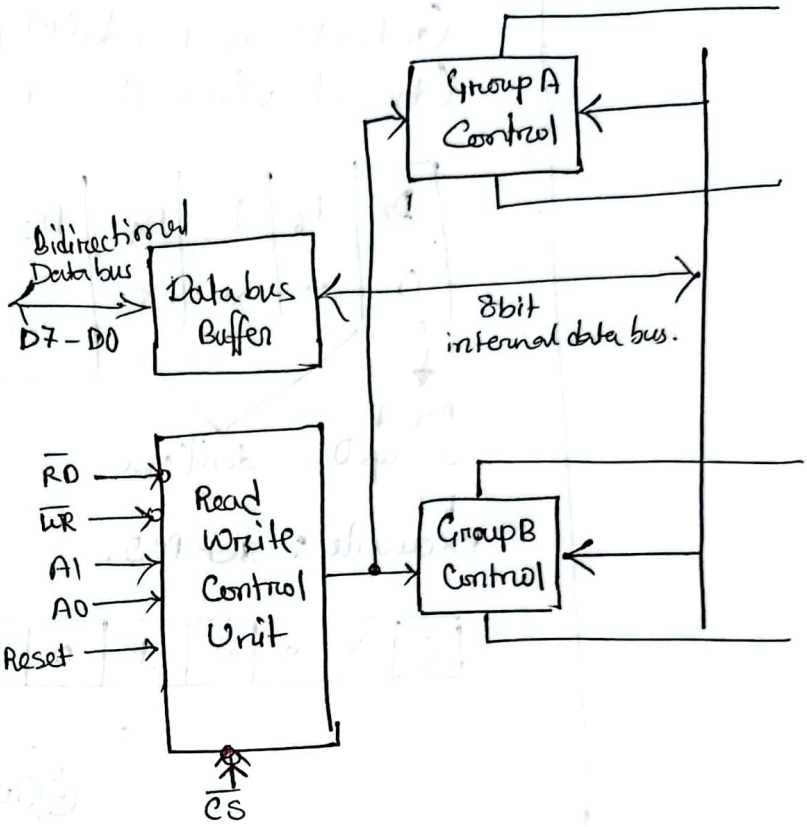
Day
Time : Date : / /

Pins :

(CS) → Chip Selection, active low;
enables communication btwn 8255 and CPU

(RD) → Read, active low;
enables 8255 to send data to CPU;
-allows CPU to read from 8255

(WR) → Write, active low;
enables CPU to write data or control words into 8255.

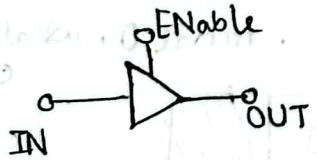


(A0 & A1) → port select 0
port select 1.
in conjunction with RD and WR - selection of ports and registers are done.
Normally connected with the least significant bits of address bus.

(Reset) → active high;
initializes control reg to 9B1t .
all ports are set to input mode again

TRI STATE BUFFER :

* buffer - এ data আসলেই store করে but এটির enable pin এ যদি high (true) থাকলে output পায় (stored টা) [behaves like normal buffer]
* low থাকলে high Impedance pass করে, disconnecting output



Enable	Input	Output
False	False	hiZ
False	True	hiZ
True	False	False
True	True	True

Sub : Operational Modes [control words]

Day
Time : Date : / /

(a) → Bit set/reset Mode (BSR mode).
(sets (1) or resets (0) portC).
(doesn't affect the operation of I/O mode)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	x	x	x				S/R

- D_7: BSR=1 always 0
- D_6, D_5, D_4: don't care
- D_3, D_2, D_1: portC bit selection
- D_0: set = 1, reset = 0.

Example : set PC5.

0	x	x	x	1	0	1	1

set = 1.

$(5)_{10} = (101)_2$

(b) → Input Output Mode (I/O mode).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1			PA	PC_A		PB	PC_B

Input = 1
output = 0

- D_7: I/O (C) always 1
- D_6, D_5: group A mode
  - 00 → mode 0 (simple I/O)
  - 01 → mode 1 (single handshake)
  - 1x → mode 2 (double ")
- D_4: PA
- D_3: PC_A — upper
- D_2: group B mode
  - 0 — mode 0
  - 1 — mode 1
- D_1: PB
- D_0: PC_B — lower

example: B and upper portC → input
lowerC and A port → output. mode 0, what's the controlword

Ans:

1	0	0	0	1	0	1	0

→ 8AH control word pass করবে

Sub :

Day

Time : Date : / /

practice: A input B output C lower input
mode 2 mode 1 C upper output

1	1	X	1	0	1	0	1

D5H or F5H.

(*) Simple I/O. করলে 16 possible combo as port এটি (A, B, CA, CB)
↳ $2^4 = 16$ combinations.

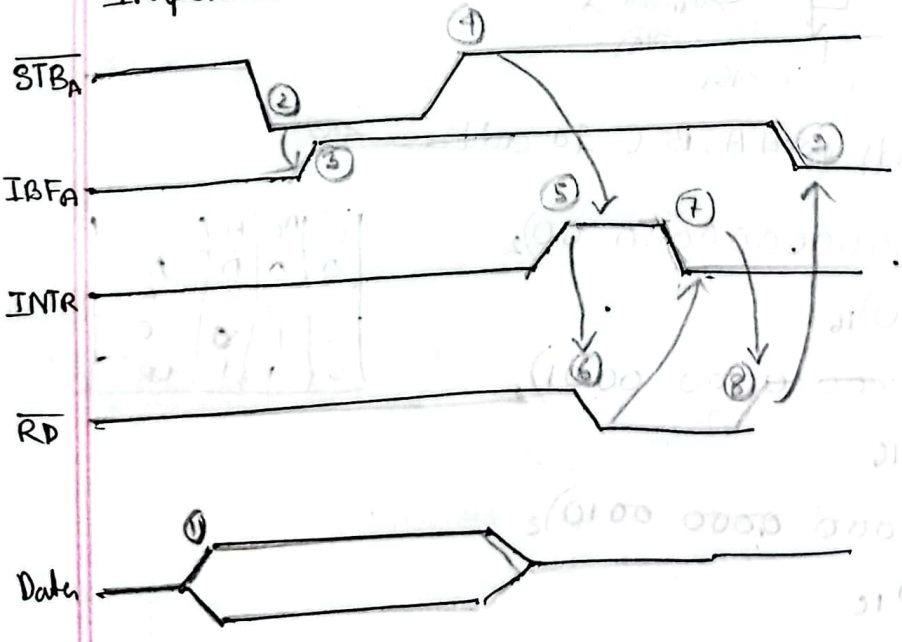
~~I/O mode~~

(*) Single handshake : port B → $PC_0 - PC_2$.
port A → input হলে $PC_3 - PC_5$.
output " PC_3, PC_6, PC_7.

handshake signals -এর জন্য use করব

I/O-এর জন্য port C use করব

Important: input mode 1 on port A:



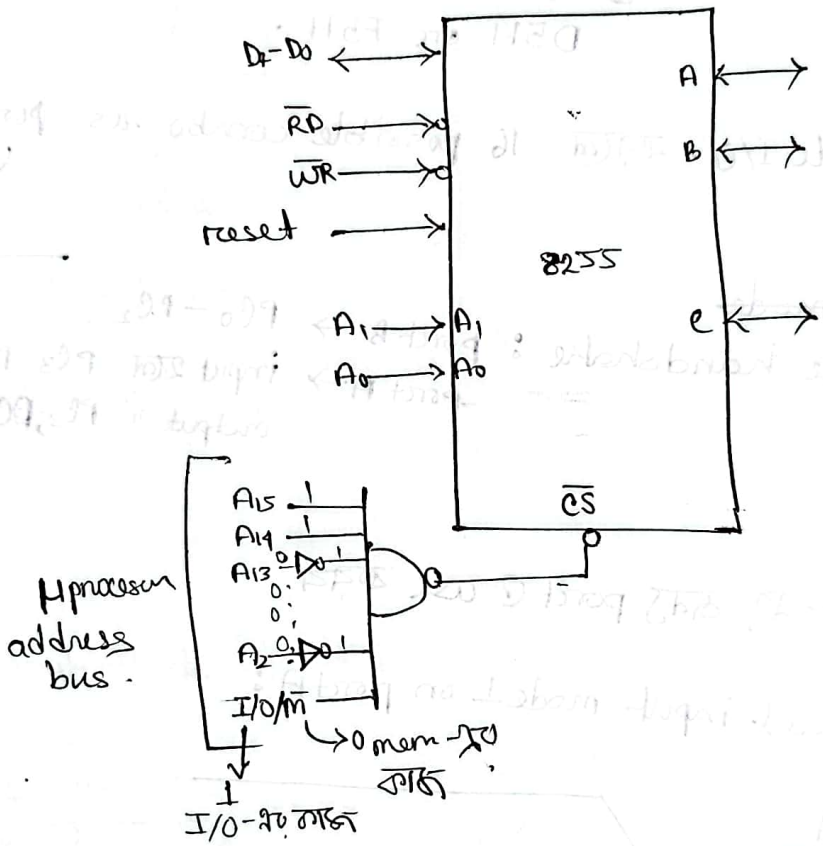
Sub :

Day

Time : Date :

(*) Double handshake : port A → bidirectional 8bit I/O bus
port B :→ can be mode1 or mode0

CPU–8255 (μprocessor-এর সাথে 8255-এর connection) :



Math: 82C55A-এর জন্য port A, B, C-এর address কত?

for port A = $(1100\,0000\,0000\,0000\,00)_2$
$= (C'000)_{16}$

for port B = $(1100\ 0000\ 0000\ 0001)_2$
$= (C\,001)_{16}$

for port C = $(1100\ 0000\ 0000\ 0010)_2$
$= (C002)_{16}$

$\overline{CS}$	A_1	A_0	
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	CR

Sub : Control Logic Design.

Day
Time : Date : / /

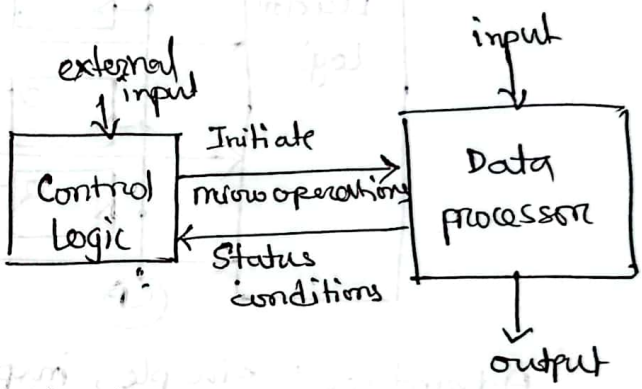
processor data store করে, process করে → data মূলত binary info (or control info)

*DATA : Discrete elements of info that are manipulated by microoperations

* Control info provides command signals for specifying sequence of microoreperations.

* Logic Design of a digital system is a process for deriving the digital circuits that perform data processing and provide control signals.

Control Unit functions applications :



① Binary variables that controls selectors and enable pins of registers are generated

② outputs of control unit selects + enable data processor and determines control unit's next state itself

③ Provides time sequence

④ Generates sequence signals of microoperations

⑤ Internal state dictates the control functions.

⑥ Next state depends on status conditions.

⑦ Directly related to data processor.

Representation :

→ Timing Diagram : clarifies timing sequence and other relationships among the various control signals in the system

→ Flowchart : specifies the sequence of procedural steps and decision paths for an algorithm.

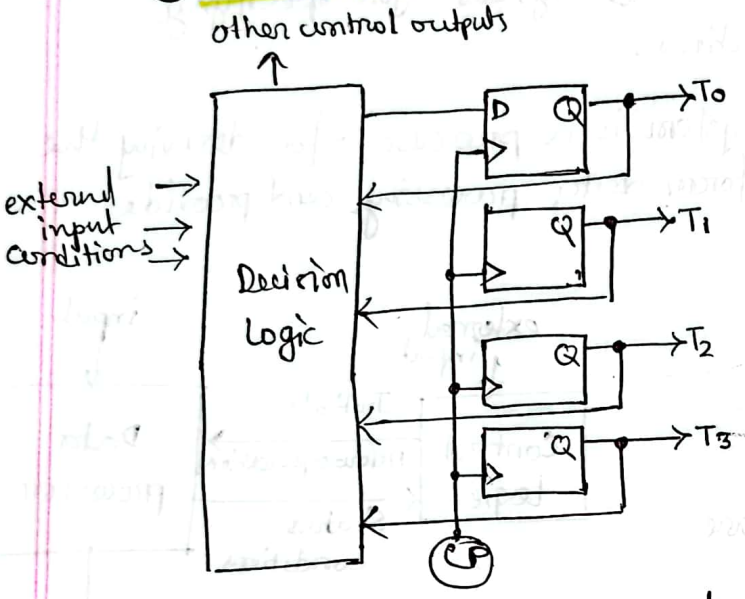
Sub: Control Organizations

Day
Time: Date: / /

Major goal :- logical + straightforward logic develop করা

4 methods :

① One flip flop per state method :



* only one ff at a ~~pos~~ time
* a single bit to propagate from one ff to other under control of decision logic
* each ff represents a state and is activated only when control bit is transferred to 1 [0 হলে active হয় না]

Advantage : simple, implements sequential circuit decreasing combinational circuits required.
saves design efforts.

Disadvantage : The ∄ more ff needed the more costly

*** External inputs না থাকলে, Control circuit single bit shift register হয়ে যায়।
Control sequence বার বার repeat হলে ring counter-এর মতো হবে।
এজন্য ~~FF~~ 1 FF ~~cont~~ per state কে ring counter controller-3 বলে।

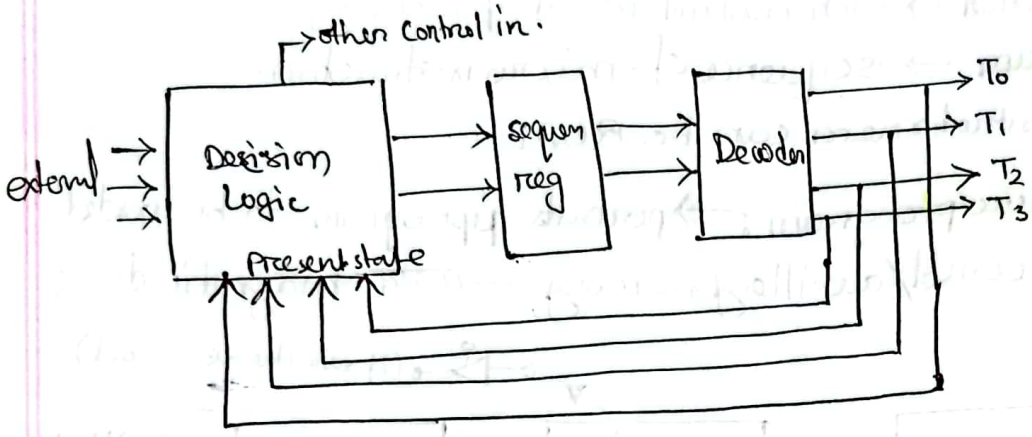
*** next state depends on external inputs and T states.

Sub :

Day

Time : Date : / . . /

② Sequence Register and Decoder Method.



* sequencing-এ এই register use হয়
* register is decoded to provide one output for each state
* register + decoder → MSI

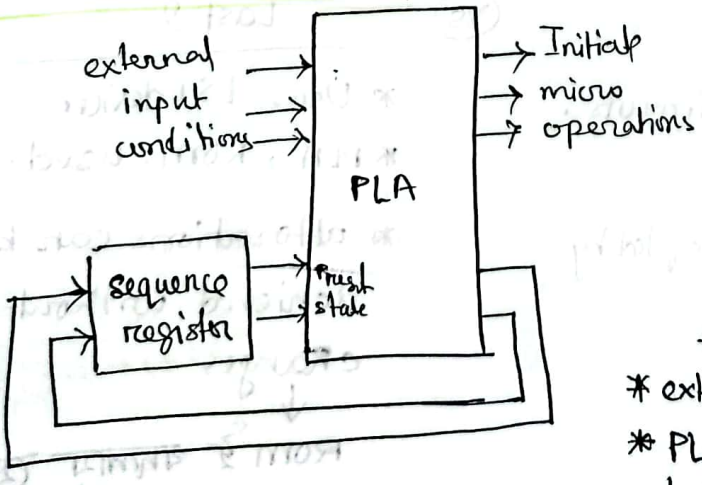
For n FF in sequence register → 2^n circuit states
2^n outputs from decoder.

example 4 bit register → 16 state → 4×16 decoder.

next state হলো present state আর external inputs-এর উপর dependent

*** It is called a counter-decoder as the decoded T states go to the decision logic again as present-state and helps to generate next state combining with external inputs and other control inputs, then repeat the process to generate new T states using Decoder.

③ PLA Control method:



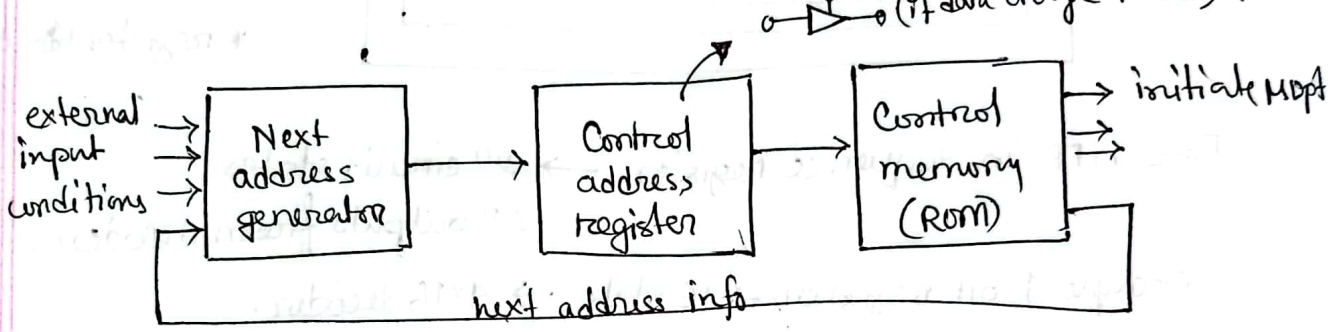
* PLA-তে যত combination লাগে তা so decoder তত লাগে না।
* যেই কয়টা state লাগবে ততগুলোই output দিবে
* IC + wires reduction
* external sequence reg present state দেয়
* PLA takes decisions which microopt to take — next state বাদি output হলো

④ Microprogram Control

→ Unit whose control variables are stored in a memory.

* Microinstruction → each control word of memory.
* Microprogram → sequence of microinstruction.
→ ~~Can be~~ control mem can be ROM

Dynamic microprogram: → permits μprogram to be loaded from pc consol/auxillary memory — যেমন magnetic disk



* address generate করে register-এ store করবে — register ROM-এ pass করে — ROM address থেকে data extract করে show করে
* Address ও data change করতে চাইলে buffer register-এ transfer করে নিতে হবে।
* ROM next address-এর info generator-কে সাহায্য দেয়

First 2

* Uses SSI and MSI circuits.
* Hard wired control.
* New changes is adopted by rewiring.

(VS)

Last 2

* Uses LSI devices
* PLA, ROM used.
* alterations can be achieved without wiring changes
↓
ROM-ই বদলায় ফেলা যায়

Sub: Hard Wired Control Maths.

Day
Time: Date: / /

Example-1

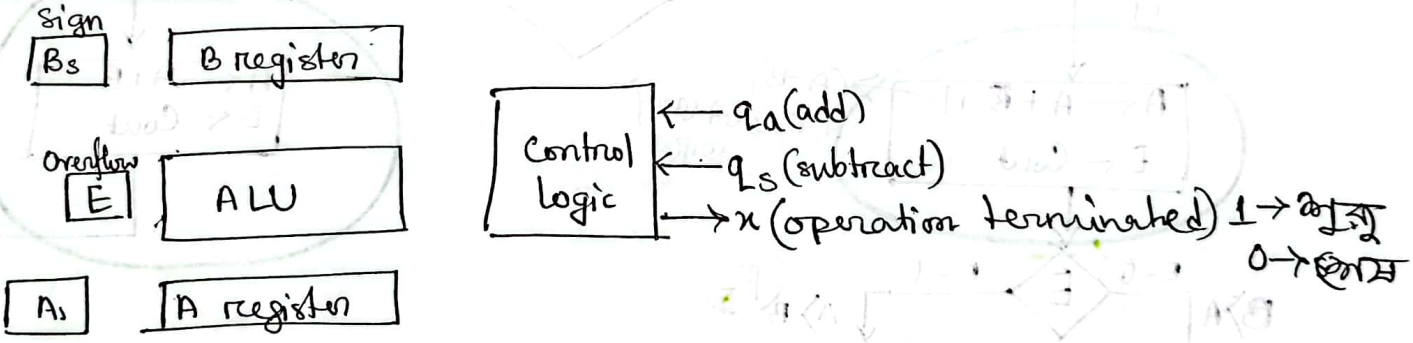
① Statement of the problem:

→ add two numbers.
→ may exceeds storage capacity by 1 bit (overflow) ff-A store করব.
→ subtraction করা যাবে.
→ ✱ numbers = fixed-point binary sign-magnitude.

exple- ⓘ 0111
sign ↓ magnitude

Steps:
① ✱ Problem stated.
② ✱ initial equipement assuption
③ Algorithm formulation
④ Data processor Design.
⑤ Control Logic Design

② Equipements: A,B register, A_s, B_s (sign bit), E(overflow), ALU, Control logic.



③ Algorithm:

Add

① (+A) + (+B)
② (+A) + (-B)
③ (-A) + (+B)
④ (-A) + (-B)

⑤ (+A) − (+B)
⑥ (+A) − (+B)
⑦ (-A) − (+B)
⑧ (-A) − (+B)

0	0
0	1
1	0
1	1

✓ converted
⑤ (+A) + (+B)
⑥ (+A) + (-B)
⑦ (-A) + (+B)
⑧ (-A) + (-B) → add-এর সাথে মিলে গেছে

So, (±A) ± (±B)

✓ ~~কার~~ ২টি methode 4টির convert করবার দরকার

(±A) + (±B)

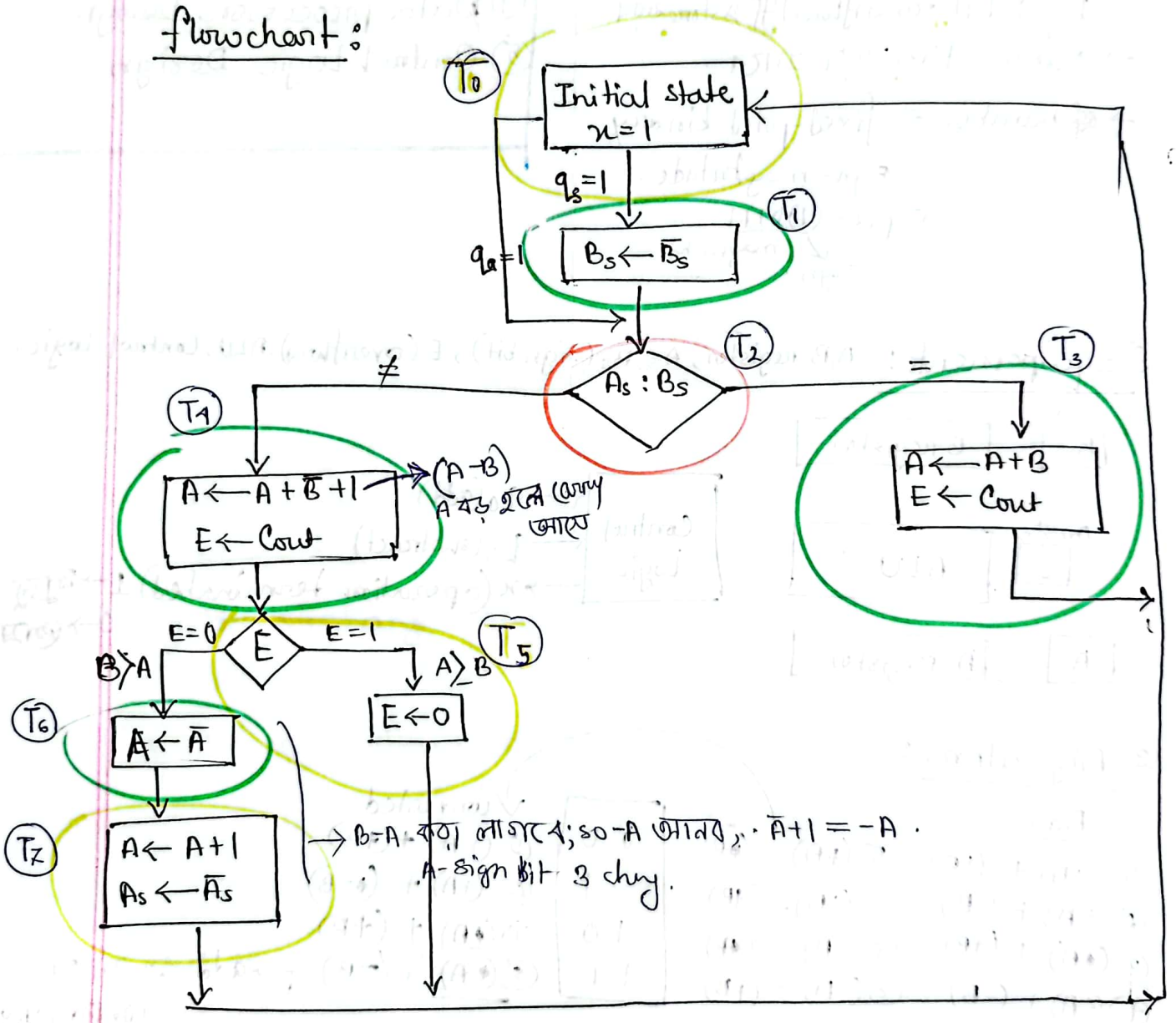
$(+A)+(+B) = +(A+B)$ যেমন → $(+2)+(+5) = +7$

$(+A)+(-B) = +(A-B)$ when $(A>B)$; $-(B-A)$ when $(B>A)$

$(-A)+(+B) = -(B-A)$ " " ; $+(B-A)$ " $(B>A)$

$(-A)+(-B) = -(A+B)$ যেমন → $(-2)+(-5) = -7$

flowchart :

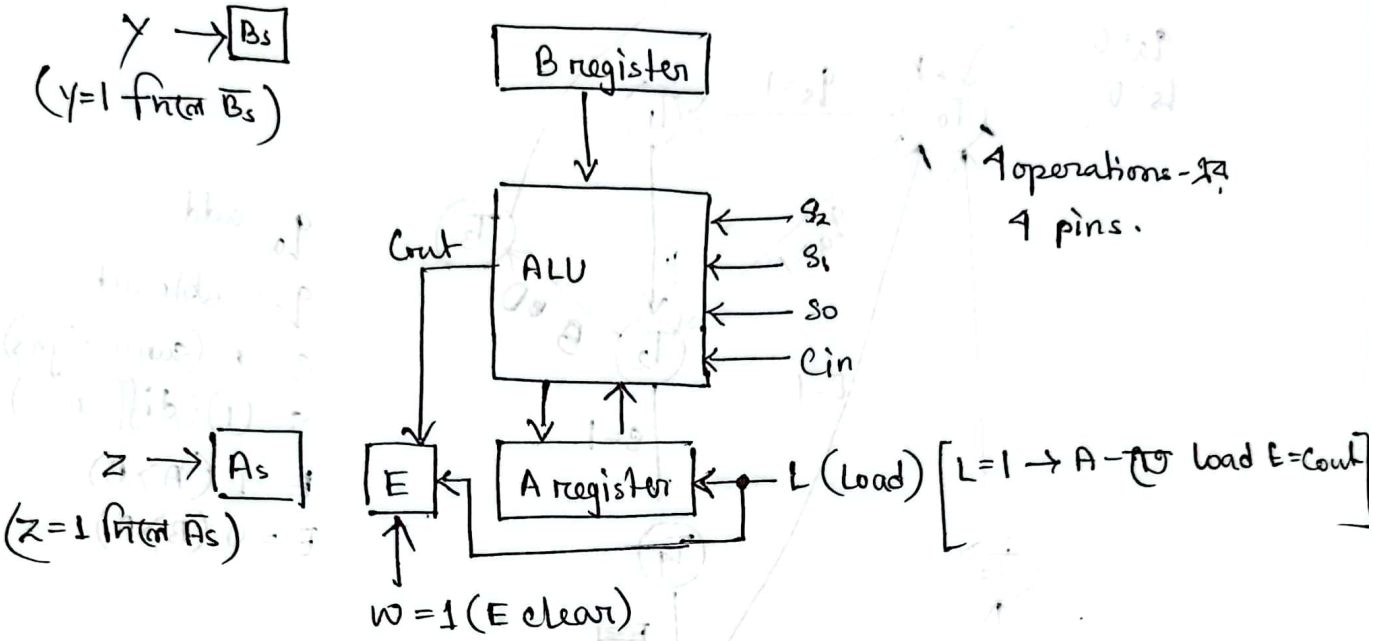


Sub :

Day

Time : Date : / /

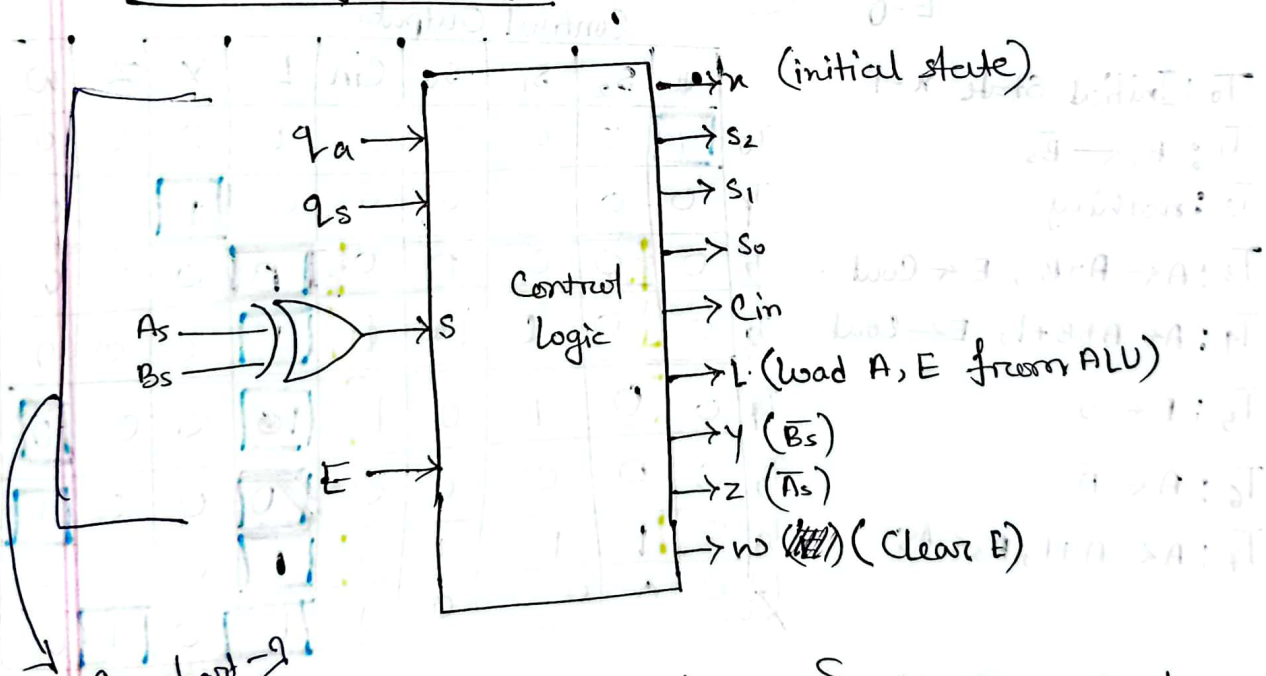
④ Data processor



1 ⇒ A operations-এর 4 pins.

L=1 → A-এ load E=Cout

⑤ Control Logic Design



যাতে flowchart-এ decision নিতে পারে তাদের input CL-এ।

Data processor-এর inputs-এর 'CL'-এর output

As Bs XOR cause ⇒
0 0 → 0 → dular
0 1 → 1 → sign
1 0 → 1 → sign
1 1 → 0 → dular

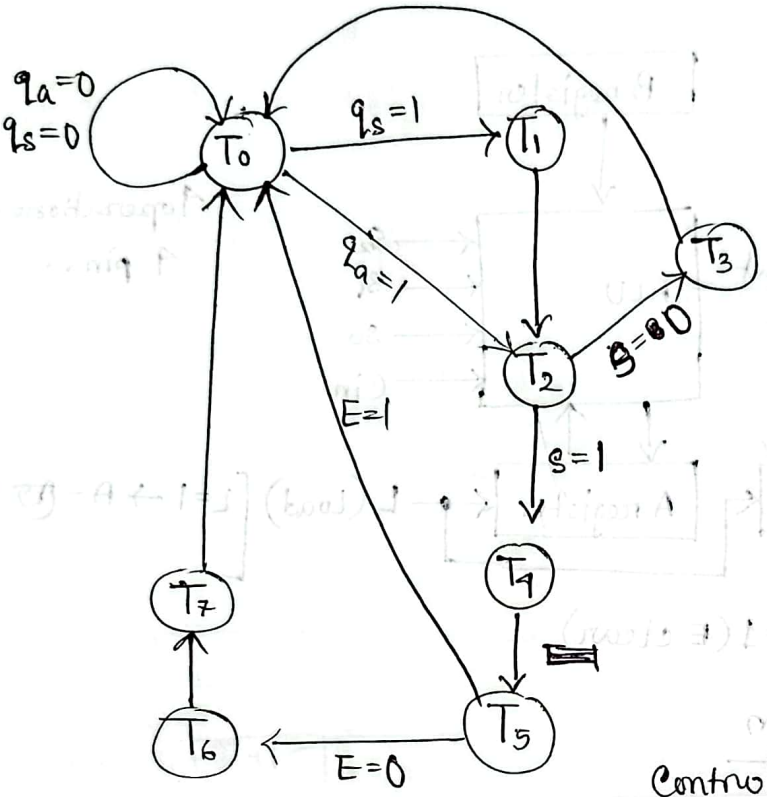
sign same হলে sign রাখার দিব
sign আলাদা হইলে কাজ করা লাগবে conditions-এর

Sub :

Day

Time : Date : / /

Control State Diagram :-



q_a = add

q_s = subtract

S = 0 (same signs)

S = (1) (diff ")

E = 1 (A > B)

E = 0 (B > A)

T_0 : Initial State n=1

T_1 : $B_s \leftarrow \overline{B_s}$

T_2 : nothing

T_3 : $A \leftarrow A+B$, $E \leftarrow Cout$

T_4 : $A \leftarrow A+\overline{B}+1$, $E \leftarrow Cout$

T_5 : $E \leftarrow 0$

T_6 : $A \leftarrow \overline{A}$

T_7 : $A \leftarrow A+1$, $A_s \leftarrow \overline{A}S$

Control Outputs

	q_a	S_2	S_1	S_0	Cin	L	Y	Z	W
T_0	1	0	0	0	0	0	0	0	0
T_1	0	0	0	0	0	0	1	0	0
T_2	0	0	0	0	0	0	0	0	0
T_3	0	0	0	1	0	1	0	0	0
T_4	0	0	1	0	1	1	0	0	0
T_5	0	0	0	0	0	0	0	0	1
T_6	0	1	1	1	0	1	0	0	0
T_7	0	0	0	0	1	1	0	1	0

A+B → 0010

(A-B) = A + $\overline{B}$ + 1 → 0101

A = $\overline{A}$ → 1110

(arrow দুকানোর জন্য)

Flip Flop input functions :

$DT_0 = \bar{q}_a \bar{q}_s T_0 + T_3 + T_5 + T_7$

$DT_1 = q_s T_0$

$DT_2 = T_1 + q_a T_0$

$DT_3 = \bar{S} T_2$

$DT_4 = S T_2$

$DT_5 = T_4$

$DT_6 = \bar{E} T_5$

$DT_7 = T_6$

Boolean functions for output (vertically নিব)

$x = T_0$

$S_2 = T_6$

$S_1 = T_4 + T_6$

$S_0 = T_6 + T_3$

$Cin = T_4 + T_7$

$L = T_3 + T_4 + T_6 + T_7$

$y = T_1$

$z = T_7$

$w = T_5$

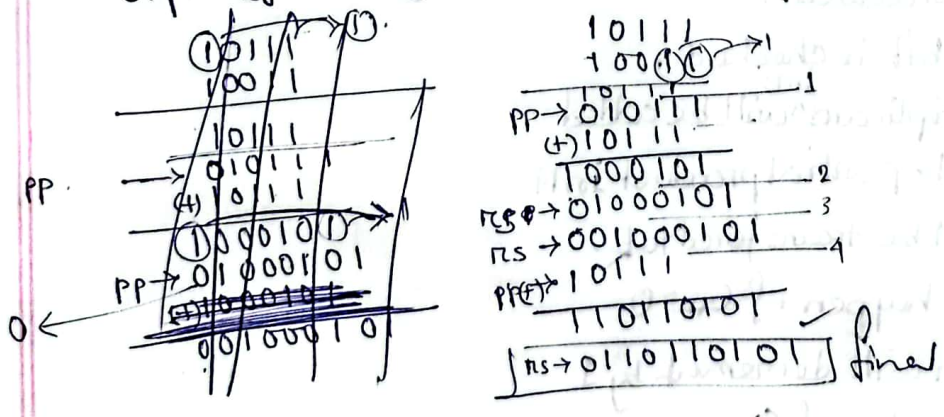
Example-2

① Statement of the problem:

Multiplication বাবব দুটো binary number-কে multiplier ~~multiplicand~~ - কে LSB (last bit) = 1 (add multiplicand with partial product then right shift)

" " LSB = 0 (right shift the partial product)

Expli (বুঝার জন্য - exam-এ লিখবে না)



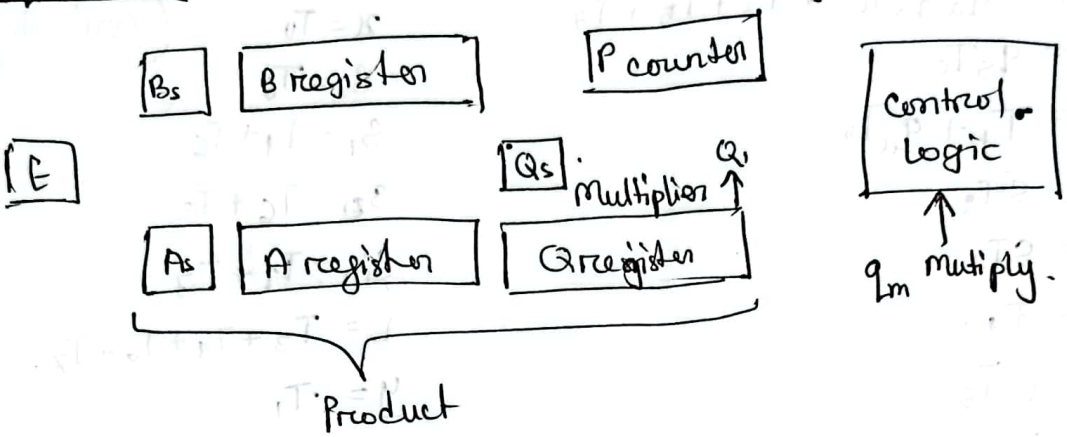
Sub :

Day

Time :

Date : / /

② Equipement :



③ Algorithm :

* multiplicand is in B and multiplier in Q, their signs in Bs and Qs.

When $q_m = 1$ process starts.

The two signs are compared by XOR gate.

If alike → As gets plus sign (0)

If differs → As " neg " (1)

Registers A and E are cleared.

Sequence counter P is set to K = no. of bits in the multiplier.

Next we will enter a loop that forms partial products..

Multiplier Q_1 bit is checked, if 1 the multiplicand (in B) will be added to the present partial product in A.

Carry bit will be transfered to E.

Nothing will happen if $Q_1 = 0$.

The P counter will decrement by 1 regardless the value of Q,

Register A, Q, E are shifted once to the right.

B ← multiplicand Bs ← sign

Q ← multiplier Qs ← sign.

A ← A + B (multiplication PP)

E ← Cout

P ← K, K = no. of bits of multiplier

Shift operation :

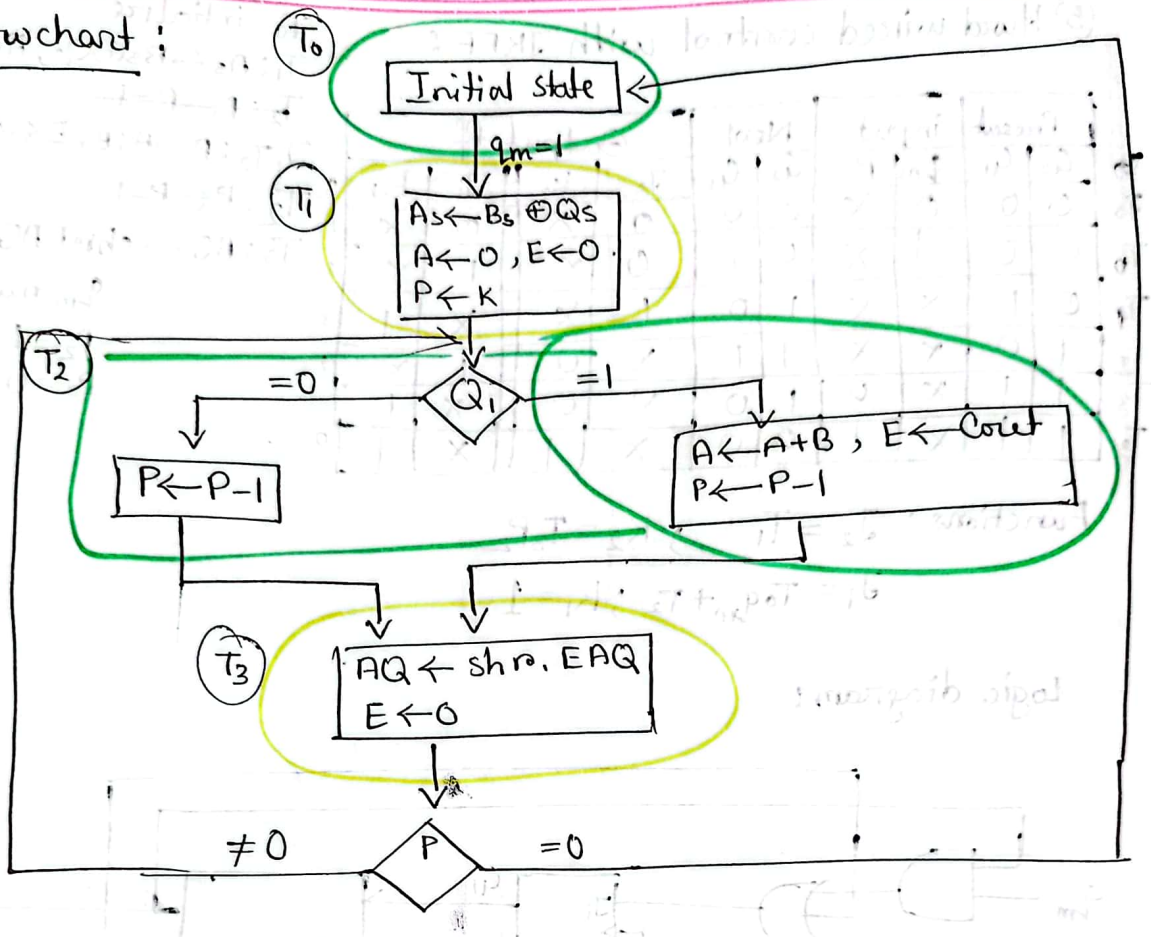
AQ ← shr EAQ, E ← 0.

Sub :

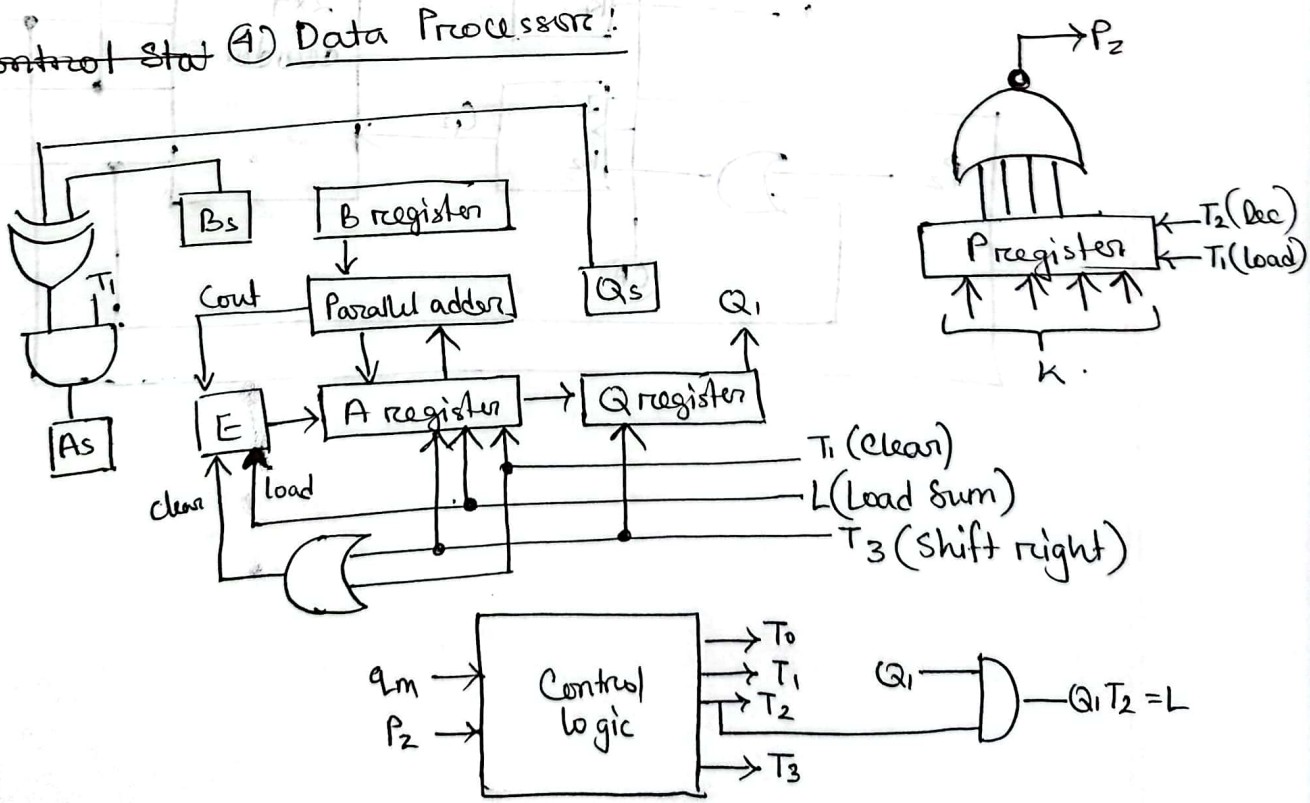
Day

Time : Date : / /

flowchart :



~~Control stat~~ ④ Data Processor :



Day

Sub :

Time : Date : / /

⑤ Hard wired control with JKFF :

T_0 : initialize

T_1 : $A_S \leftarrow B_S \oplus Q_S$, $A \leftarrow 0$, $E \leftarrow 0$, $P \leftarrow k$

~~T_2 : $P \leftarrow P-1$~~

$Q_1 T_2$: $A \leftarrow A+B$, $E \leftarrow Cout$

T_2 : $P \leftarrow P-1$

T_3 : $AQ \leftarrow shr\, EAQ$, $E \leftarrow 0$

q_m multiply

$P_2 = 1$ if $P = 0$

$P_2 = 0$ if $P \neq 0$

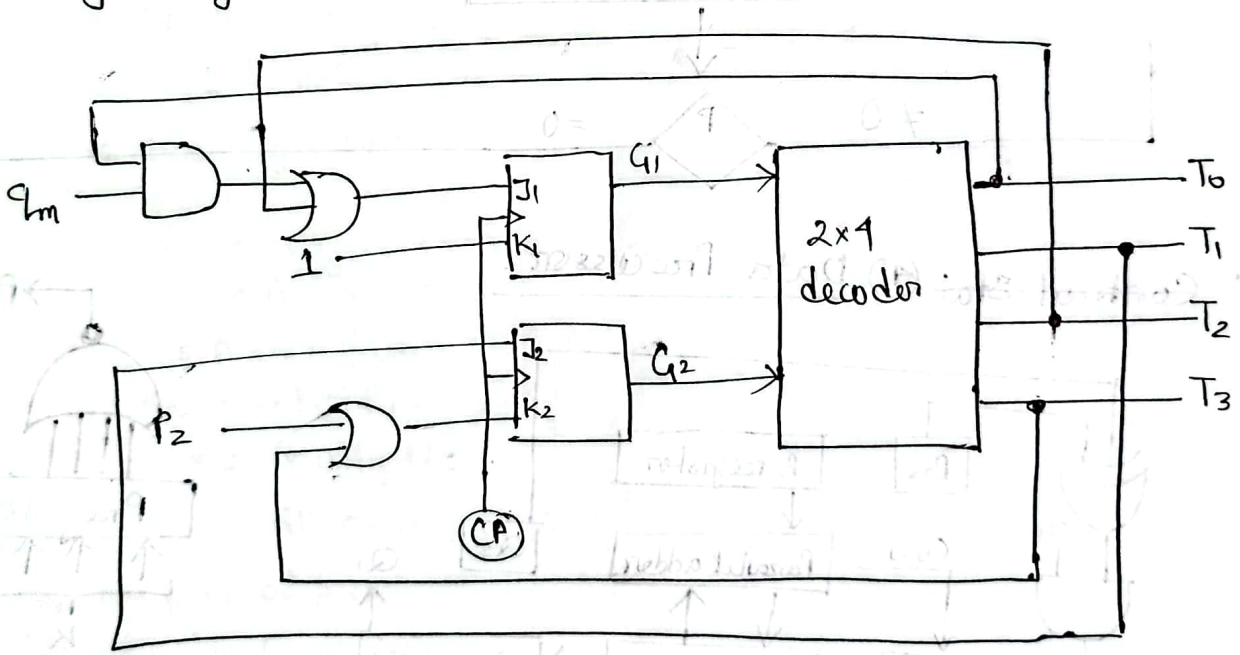
	Present		input		Next		Output ff			
	G_2	G_1	q_m	P_2	G_2	G_1	J_2	k_2	J_1	K_1
T_0	0	0	0	X	0	0	0	X	0	X
T_0	0	0	1	X	0	1	0	X	1	X
T_1	0	1	X	X	1	0	1	X	X	1
T_3	1	0	X	X	1	1	X	0	1	X
T_3	1	1	X	0	1	0	X	0	X	1
T_3	1	1	X	1	0	0	X	1	X	1

0 1 2 3 2 0

Functions : $J_2 = T_1$; $K_2 = T_3 P_2$

$J_1 = T_0 q_m + T_2$; $K_1 = 1$

Logic diagram :



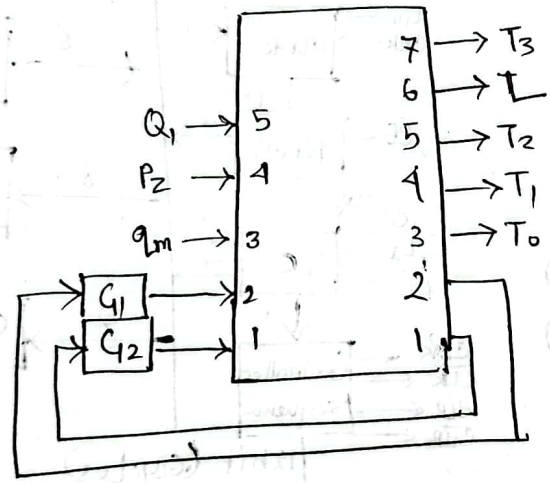
Sub :

Day

Time : Date : / /

State table for control circuit :

	present Q_2	present Q_1	input q_m	input P_2	input Q_1	next Q_2	next Q_1	Output T_0	Output T_1	Output T_2	Output T_3	
T_0	0	0	0	X	X	0	0	1	0	0	0	0
T_0	0	0	1	X	X	0	1	1	0	0	0	1
T_1	0	1	X	X	X	1	0	0	1	0	0	2
T_2	1	0	X	X	0	1	1	0	0	1	0	3
T_2	1	0	X	X	1	1	1	0	0	1	1	3
T_3	1	1	X	0	X	1	0	0	0	0	0	2
T_3	1	1	X	1	X	0	0	0	0	0	0	0
	1	2	3	4	5	1	2	3	4	5	6	7



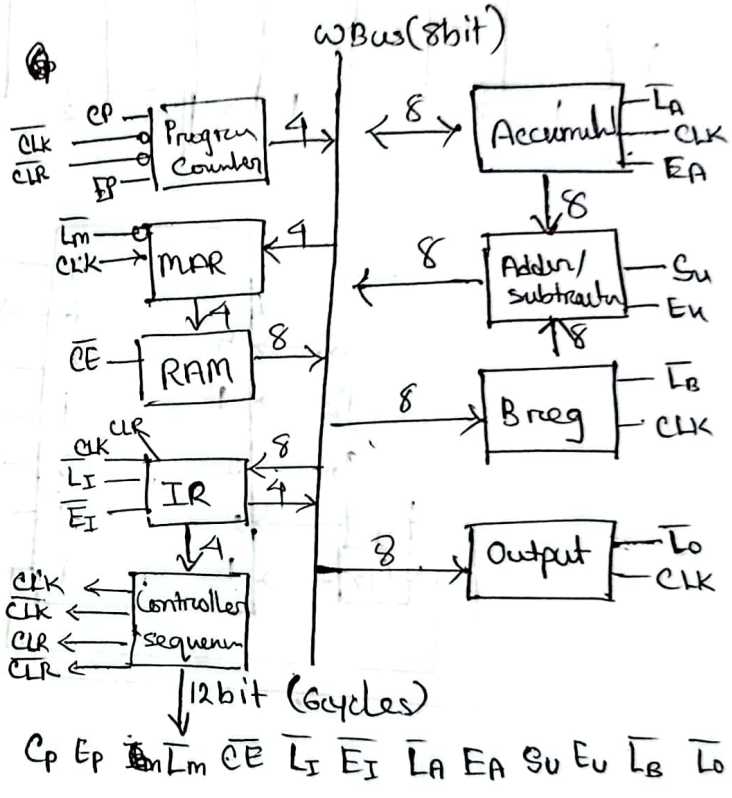
Sub : SAP-1

Day
Time : Date : / /

Simple As Possible Computer
* First stage towards modern computer
* very basic explained by Albert Paul Malvino.
* Covers advanced concepts.
* modern computer - ২০ ধাপ্পা নেয়
* organized bus
* W-bus - ২০ সাথে register - ২০ connected by tri state buffer

Architecture :

① 8 LEDs output
② 16 bytes of RAM
③ 5 instructions
→ 3 with 1 operand
→ 2 with implicit ঘ্যা
④ Registers
1. accumulator
2. Out register
3. B "
4. mem Address reg (MAR)
5. Instruction Register (IR)



Program Counter:

* generates address (4 bit); Cp is high.
* points address of next instruction to be fetched/executed
* sends address to WBus for MAR; Ep is high.

MAR :

* Stores address of data and instructions. from WBus (PC থেকে) ; Lm
* Sends to RAM to read.

Sub :

Day

Time : Date : / /

RAM:

* receives address from MAR
* reads the data and sends to IR by WBus.
  (CE)
* It's 16 x 8 RAM → 16 memory locations of 8bit each data.

IR:

* receives data or instruction, $\overline{L_I}$ low.
* breaks and sends upper nibble to controller / $\overline{E_I}$ is low
  lower nibble to Wbus

~~* send~~

Accumulator:

* Stores first data from W-bus. (8bit).
* sends data to Adder/subtractor for operation.
* Stores the output when $\overline{L_A}$ is low.
* Has two outputs:— one goes to adder/subtractor;
  another to W-bus by tri state buffer (E_A is high) shows.

B register:

* stores/loads the second data ; L_B goes low.
* Sends it to adder-subtractor.

Adder/Subtractor:

* Takes data from Acc and B reg.
* Adds when $S_u = 0 \rightarrow S = A + B$
  subtracts when $S_u = 1 \rightarrow S = A + \overline{B} + 1$
* is asynchronus
* when E_u is high, shows data in Wbus (sends).

Sub :

Day

Time :

Date : / /

Output register:

* after everything is done,
* accumulator's stored answer is received through W-bus when E_A is high and Lo is low.
* Show it through LEDs / 7 segments (bin display)

Controller:

* $\overline{CLR}$ is send to PC
* CLR is send to IR

] resets PC to 0000 wipes the IR's last instructions.

* 12 bit control bits go everywhere, do there things.

SAP1 instruction set:

Opcode	Mnemonics	Operations	Desp
0000	LDA	ACC ← RAM[MAR]	load RAM data to acc.
0001	ADD	ACC ← ACC+B	stores, add RAM data to acc
0010	SUB	ACC ← ACC−B	subtract RAM data from acc.
(EF) 1110	OUT	OUT ← acc	shows output from acc
(FF) 1111	HLT	CLK ← 0	Stops

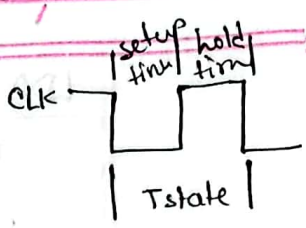
machine — assembly

8080 and 8085 μprocessor:

* 8080 → first widely used. (72 instructions)
* 8085 advanced.
* SAP1 is upward compatible with 8080/8085 instruction.
*

Sub: 6 Cycles (Fetch + Execute)

Day | Time : | Date : / /

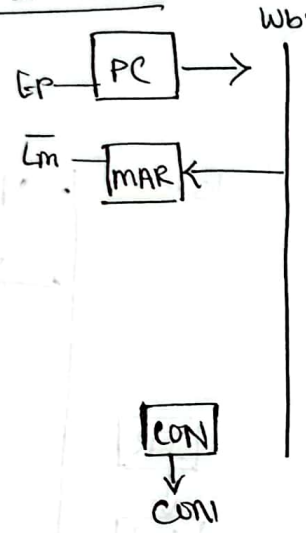


* after each cycle counter resets to 000001.
* initial state T1 starts with negative clock edge.

Fetch cycle

Same for all instructions.

T_1 State : (Address State)



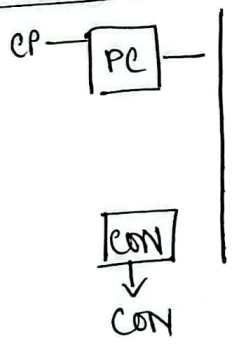
$CON = C_P \cdot E_P \overline{L_m} \overline{CE} \cdot \overline{L_I} \overline{E_I} \cdot \overline{L_A} E_A \cdot Su E_U \cdot \overline{L_B} \cdot \overline{L_O}$

$= 0\ 1\ 0\ 1\ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

$= 5 \quad E \quad 3$

$= 5E3H$

T_2 State : (Increment State)



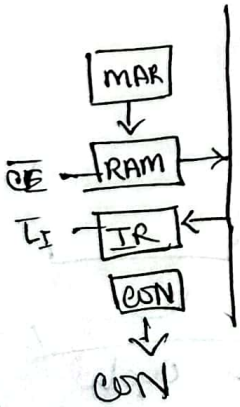
$CON = C_P \cdot E_P \overline{L_m} \overline{CE} \cdot \overline{L_I} \overline{E_I} \overline{L_A} E_A \cdot Su E_U \overline{L_B} \overline{L_O}$

$= 1\ 0\ 1\ 1\ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

$= B \quad E \quad 3$

$= BE31H$

T_3 State : (Memory State



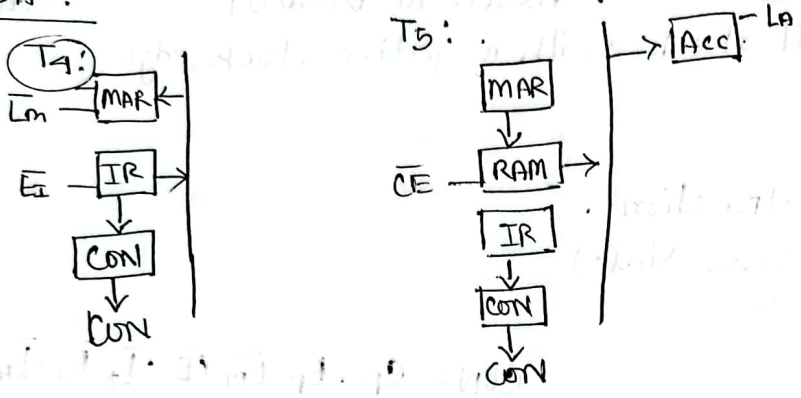
$CON = 0\ 0\ 1\ 0\ \ 0\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

$= 263H$

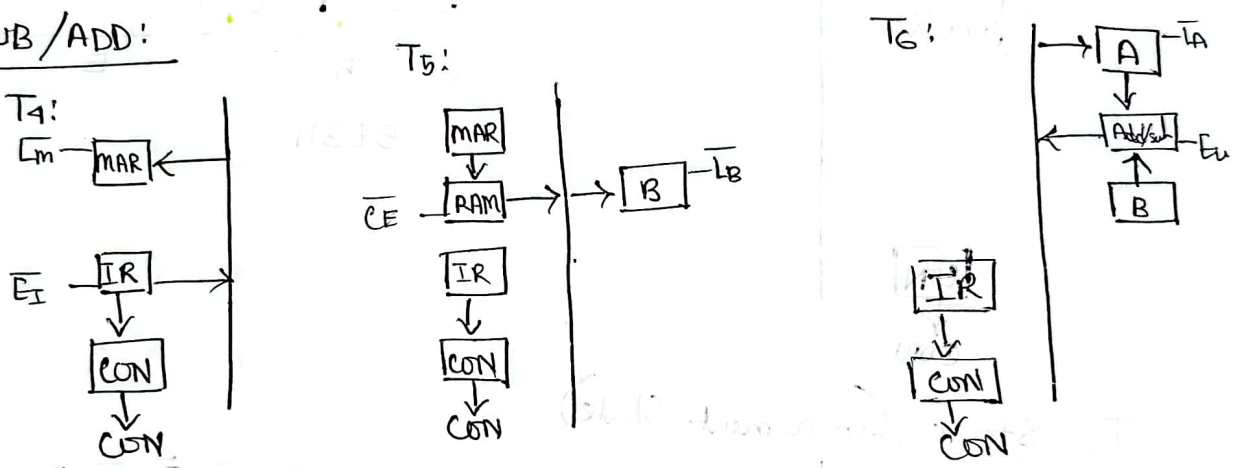
Sub : Execution Cycle

Day
Time : Date : / /

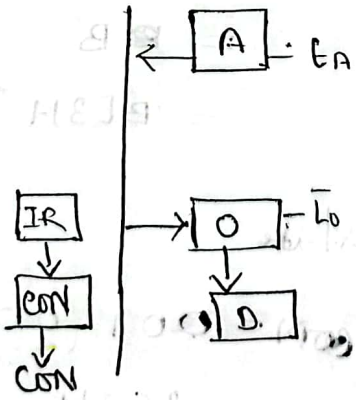
LDA :



SUB / ADD :



OUT : T4:



Total six states are called Machine cycle.