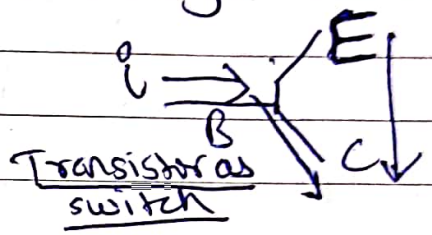
Date: ___/___/___

8-bit computer update - Ben

Book: Digital Computer Electronics.
by Albert Paul Malvino.

XOR: Q+ can invert input or pass the input using control sign

$A \oplus B = XOR = A.\overline{B} + \overline{A}.B$

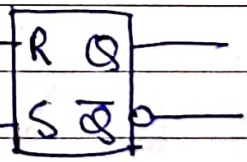


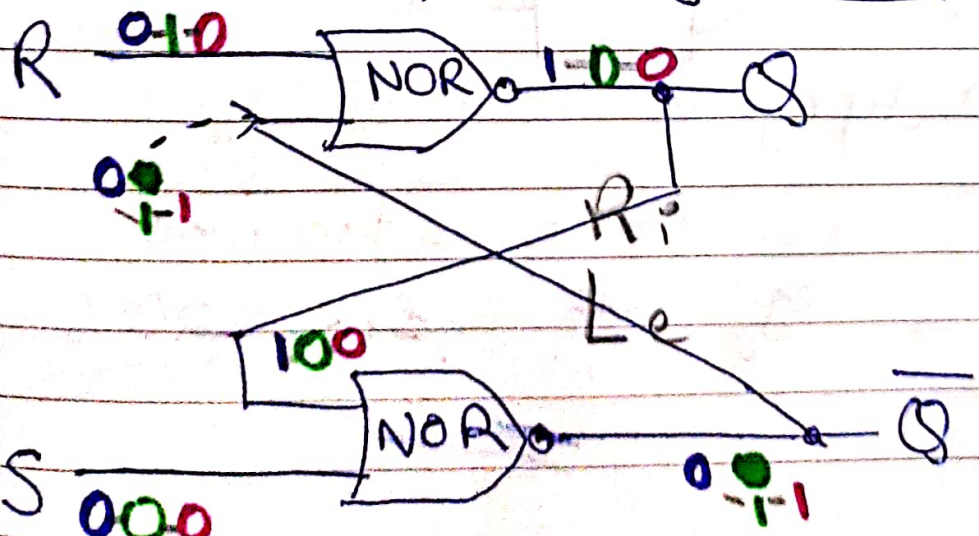
Latches & Flip Flops

- Building blocks of computer memory
- S-R Latch < NOR / NAND

Set reset latch

Press R→$\overline{Q}$ high; S→Q high



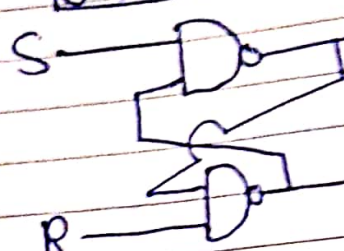


Haq, ek behtar zindagi ka.

Scanned with CamScanner

Date: 25/09/

NAND S-R latch (Gated NOR S-R latch)



S	R	Q	$\bar{Q}$
0	0	not used	
0	1	1	0
1	0	0	1
1	1	Memory (as before)	

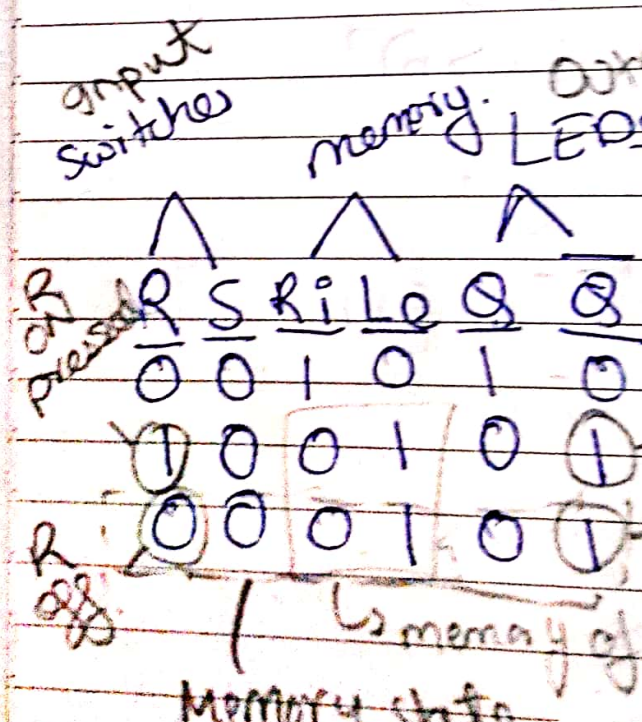
S	R	Q	$\bar{Q}$
0	0	Memory (as before)	
0	1	0	1
1	0	1	0
1	1	not used	

S=1, R=1, Q=0, $\bar{Q}$=0

S=0 R=0; Q=0 $\bar{Q}$=1 (2 cases random)

OR Q=1 $\bar{Q}$=0

- The basic storage element is called latch or latches to "0" or "1"
- To change (reset) to (glow LED) and (store in memory) press S.



NOR:

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

Haq, ek behtar zindagi

Scanned with CamScanner

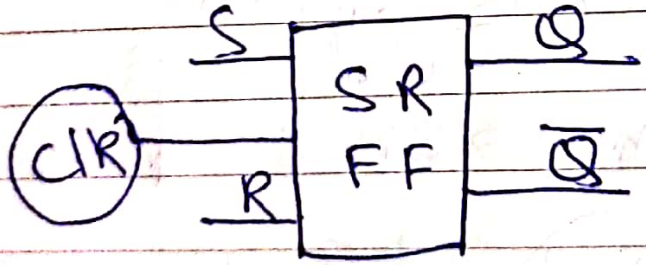
Date: __/__/__

- Clock? - a signal.

| : leading edge
| : falling edge
— : High
_ : Low.

$$\uparrow f = \frac{1}{T\downarrow}$$

we can set SR FF to only operate when clk



— or | or |

Duty cycle = Ratio of time: $\frac{\text{signal}}{\text{tot time}}$

eg. here = $\frac{t/2}{t} = 50\%$

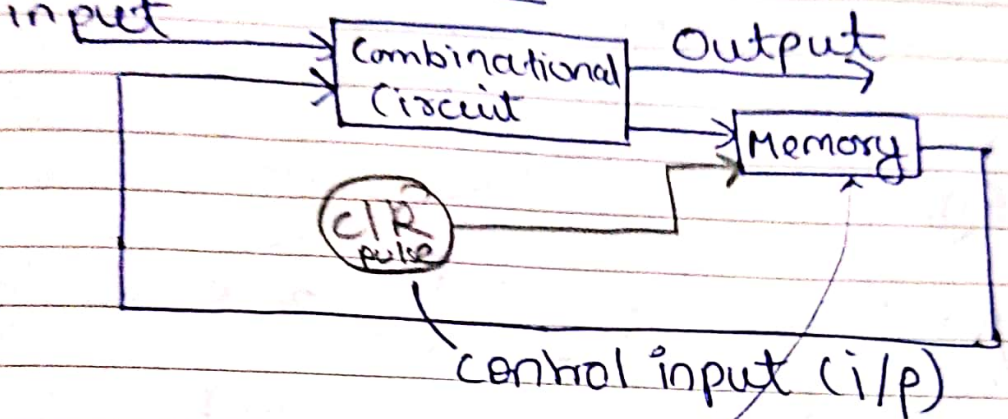
Haq, ek behtar zindagi k

Scanned with CamScanner

Date: / /

Triggering

A sequential circuit is:



Triggering methods — level (High)
edge — +ve edge, -ve edge

Memory is changed/switched based on the control i/p from the clk pulse (the stored state)

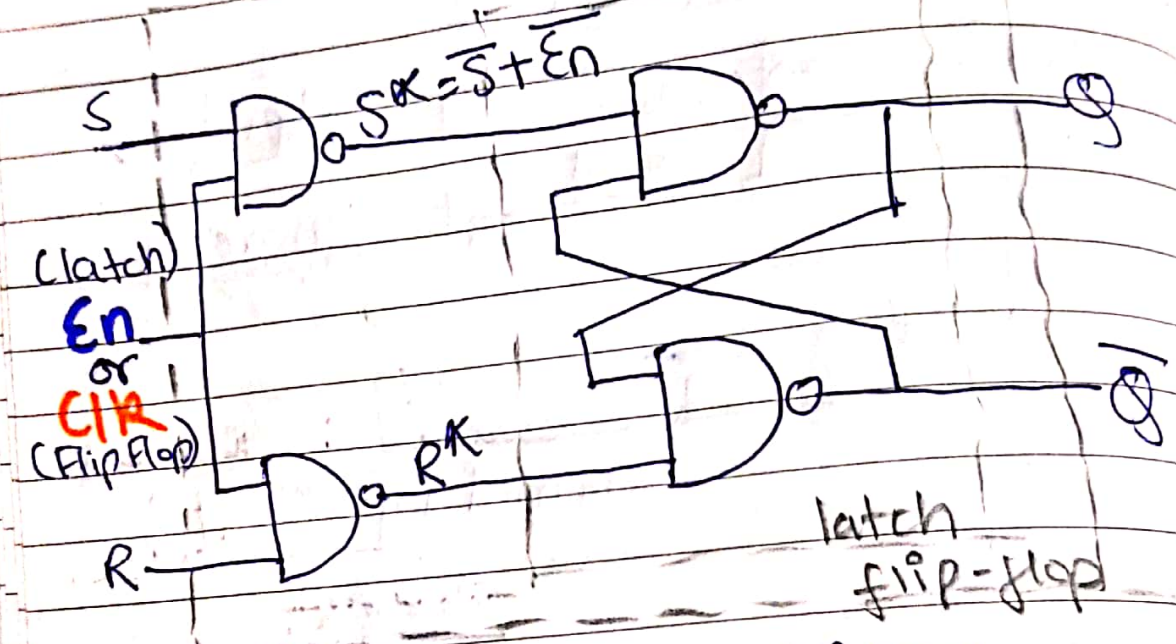
If we are i/p Memory cond. eg. 11 then o/p will be the same as what was just before it.

Haq, ek behtar zindagi ka.

Scanned with CamScanner

Date:

Difference bet/w Latch & Flip-Flop



latch En : level trigger

En : 1 $S^* = \overline{S}$

En : 0 $S^* = 1$ $R^* = 1$ (memory)

Flip Flop Clk ; edge trigger ↓ / ↑

S-R flip flop

$$S^* = (S.clk) = \overline{S} + \overline{clk}$$

$$R^* = \overline{(R.clk)} = \overline{R} + \overline{clk}$$

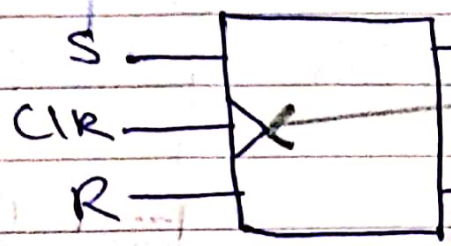
Haq, ek behtar zinda

Scanned with CamScanner

Date: __/__/__

Put here

S_d	R_d	Q	$\overline{Q}$
0	0	Not used	
0	1	1	0
1	0	0	1
1	1	Memory	



Indicate edge trigger. Otherwise, level.

to get

S_d	R_d	CLK	S	R	Q	$\overline{Q}$
1	1	0	(Irrelevant)		Memory	
1	1	1	0	0	Memory.	
1	0	1	0	1	0	1
0	1	1	1	0	1	0
0	0	1	1	1	Not used	

Scanned with CamScanner

Date: / /

Characteristic table; Excitation table for S-R flip-flop

Truth table

Clk	S	R	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Invalid (X)

Q_{n+1} ← new state; Q_n ← old state

Characteristic Table

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

Excitation Table

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

S always 0
R → 1 or 0 ∴ Y

Left colms	Right cols
Input	O/P

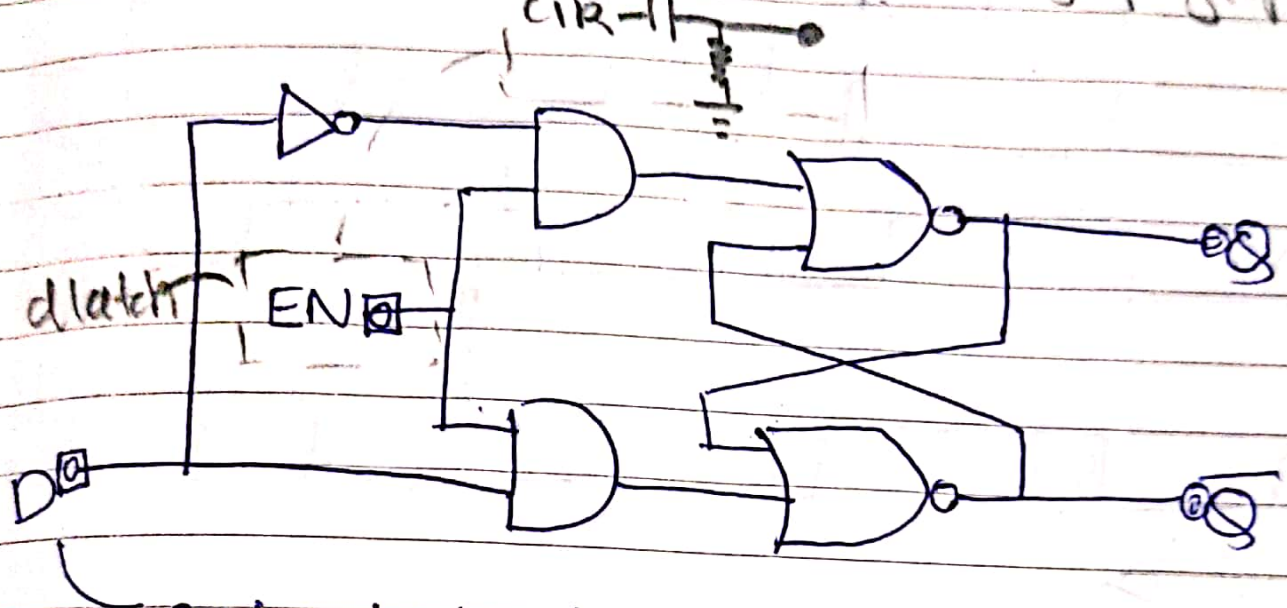
Using flip-flop conversions
counters.

Scanned with CamScanner

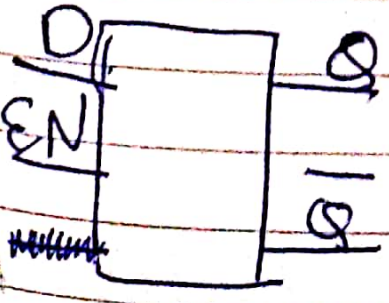
Date: ___/___/___

D-Latch

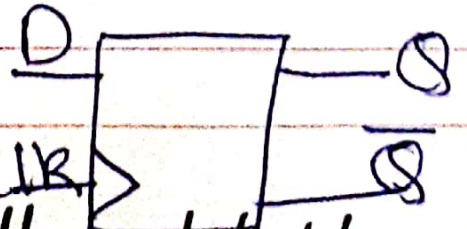
⊢ a-flip-flop.



- Only 1 bit of input when enable is pressed.
- If no enable the switch would happen but no latching.
- Great way to store 1 bit of info. Or like computer registers.
- Q would be removed ~~at~~ and input would be given with pressing D when enable is pressed.



D latch



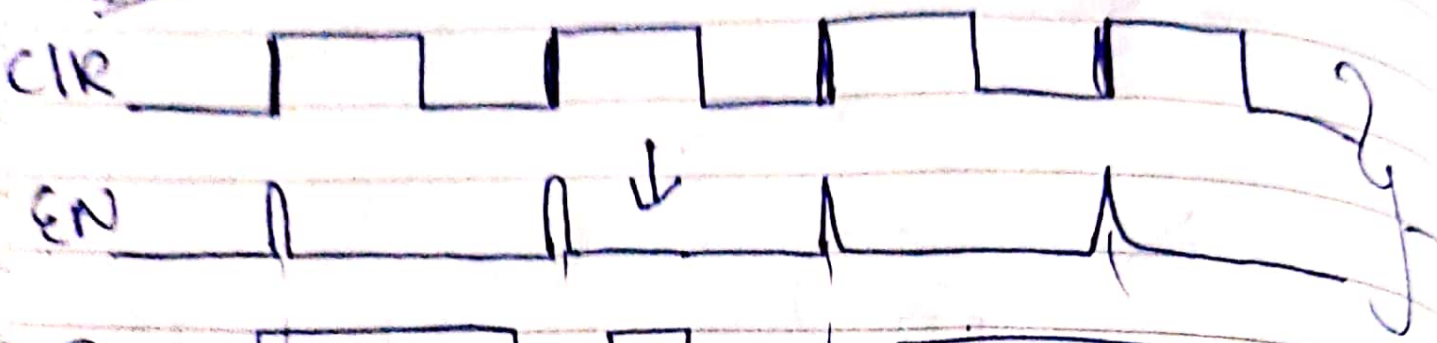
D-flip flop.

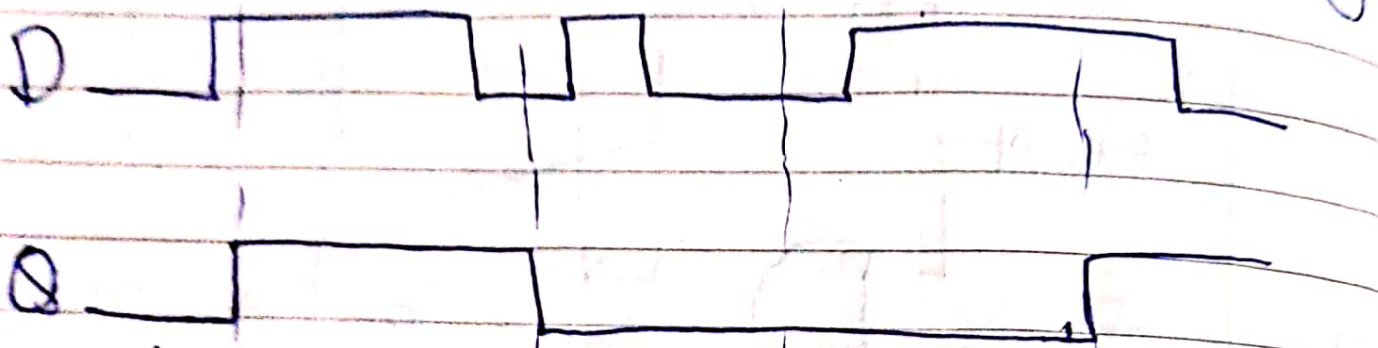
Haq, ek behtar zindagi ka.

Scanned with CamScanner

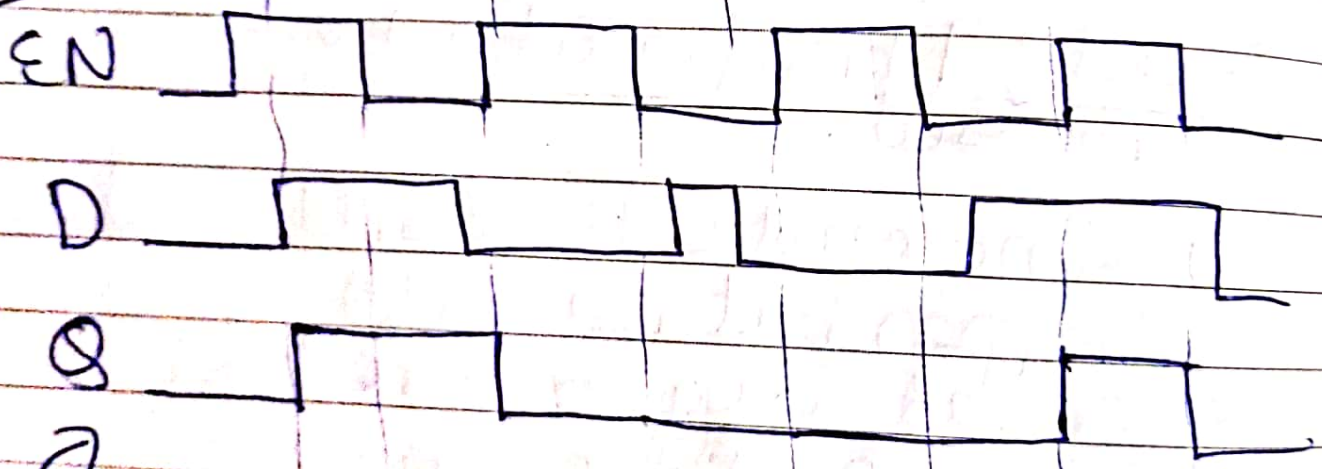
Date: __/__/__

Flip flop





Latch

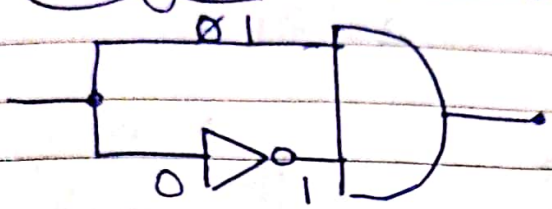


* Q will be whatever D is in EN

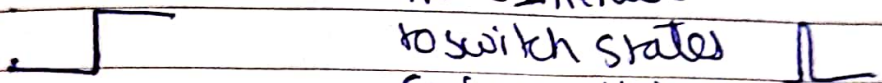
Scanned with CamScanner

Date: ___/___/___

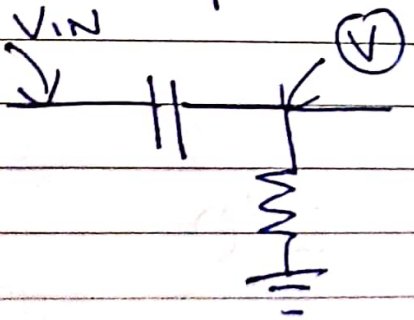
Way 1: (need 4-5 inverters) ← to pick up the delay

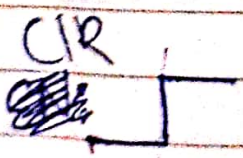


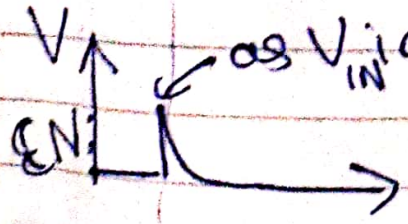
↳ time interval to switch states so for small time it will give



Way 2: Capacitor Resistor (better)







as V_{IN} increases. current flows (cap charges) ↑ Pot. at (V)

But soon cap is charged. $+V_{IN}-V_{IN}$ = no current

(V) drops.

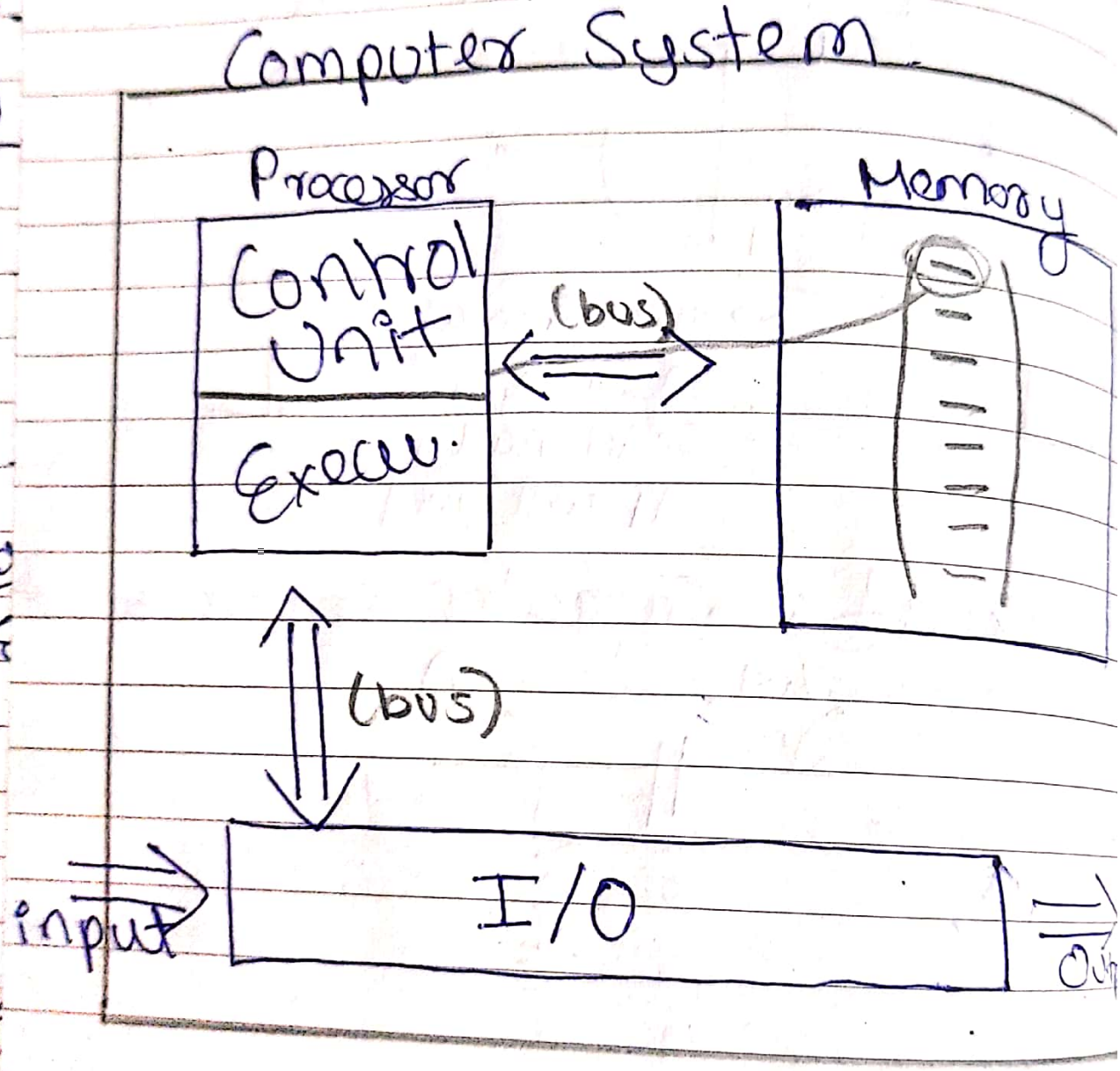
Haq, ek behtar zindagi ka.

Scanned with CamScanner

Date: 26

COA (Computer Org. & Architecture)

Intro



Program
- Instructions
  - Fetch
  - Decoded
  - Execute

Haq, ek behtar zindg

Scanned with CamScanner

6/05/

Date: __/__/__

- ⟺ : buses (communication)
- Main job of a computer is to execute programs.
- Memory : Harddisk, floppy, cd/dvd,
  RAM (diff types), ROM (diff ty)
  Cache (L1, L2, L3)
  (Store info)
- Why study COA?
  Understand exactly at WHAT TIME, WHICH INFO is stored in WHICH MEMORY
- Processor → CU (Control Unit)
  Execu. (Execution Unit)

Execu
- ALU (Arithmatic Logical Unit)
- IMP: How are numbers stored?
  5 : 101 X
  would have been true if it was an unsigned no. Signed sys is diff. All ALGORITHMS are FOR SIGNED SYSTEMS.

Scanned with CamScanner

Date: __/__/__

- Adder circuits: DNA of Arithmetic.
  * Basis of Subtraction:
    Add^n with 2's compliment
  * Basis of Multiplication
    Booths Alg. Uses add^n & sub^n
  * Basis of division
    Restoring div (more acc.)
    Non-Restoring div (faster)
  * " of floating pt. arith.
    signed exponent mantissa
  * x^y & (x)x(x) & $\sqrt{}$ & tan & $\sqrt{}$
    All adder circuit

Control Unit (Fetch + decode)

- Job is to fetch & decode instructions.
- What is a program?
  Ans) Set of instructions stored in memory.
- How is it fetched?
  Ans) Via the buses.
- What is decoding?
  Instructions are in opcode already in 0's and 1's. Compiler's create .bin / .obj file forms. Haq, ek behtar zindagi k

Scanned with CamScanner

Date: __/__/__

The instructions were already in machine Language inside the memory. Decoding DOES NOT mean converting to machine language.
DECODING means UNDERSTANDING that binary pattern what does that binary format signify.

opcode
eg. 0010 : add
1010 : subtract
0011 : multiply
1011 : divide.

every instruction will have a particular opcode.
DECODING is analyzing the opcode

- What is pipelining?
- format of an instruction?
- Addressing modes of Instr?
- How do give operands?
- How does CU decode?

Ans) · Hard-wired CU's ← delay elem, state table, seq. counter
· micro-programed CU's
Willkie's design typical single address field
dual addr. field hori/vert.

Scanned with CamScanner

Memory

- job is to store
- agenda: How info is stored into computer.

- cache (l1, l2, l3)
  - Unified cache
  - split cache
  - look through cache
  - look aside cache
  - direct mapping
  - set-associative mapping

- Virtual memory management
  - segmentation
  - paging.

Book → chop → pages → para → info
want to access.

Ex: • Tell: (logical address)
so-so chop so-so topic
so-so line.

Date: __/__/__

we want to know where.
Hard disk → RAM → cache → ALU

we want to know when. (ALU cache)

conversion from:
logical address
↓
virtual addr.
↓
physical addr.
↓
cache mem. addr.

address Translation.

Hard disk (book) (file)
↓ (broken down to)
RAM (pages)
↓ (")
Cache (section)
↓ (sent to)
Processor (orator who works on it)

Ability to know: (addr. Translation)
- whether it has come?
- If yes, where it is in that section?

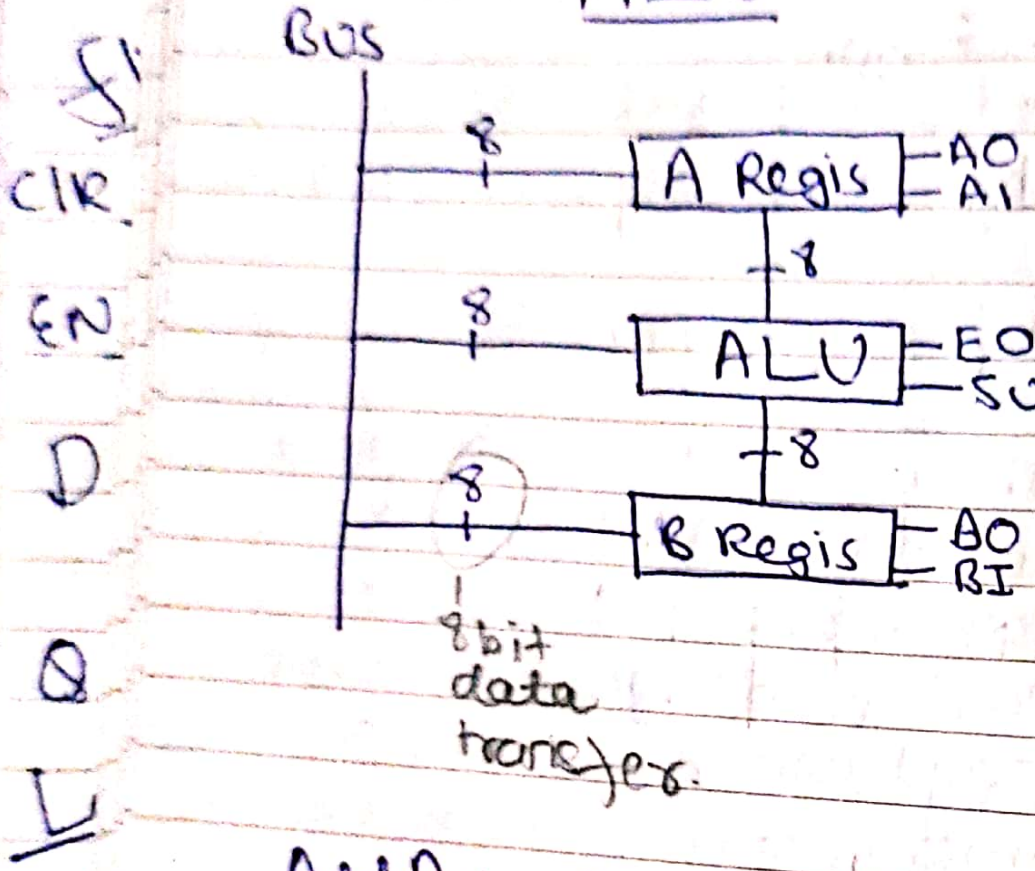
- RAM: SRAM, DRAM
SDRAM, RDRAM
- ROM: ROM, PROM, EPROM, EEPROM, FLASHROM.

Haq, ek behtar zindagi ka.

Scanned with CamScanner

ALU

Date: 27/05



Add^n: A + B

Sub^n: A + (2's compliment of B)

8 bit adder: 4bit + 4bit adder.

carry over

eg 4bit adder: DM74LS283

XOR : used for negating binary —

eg. DM74LS86.

Octal bus Transceivers: SN54/74LS245

(Tri-state buffers)

Has at

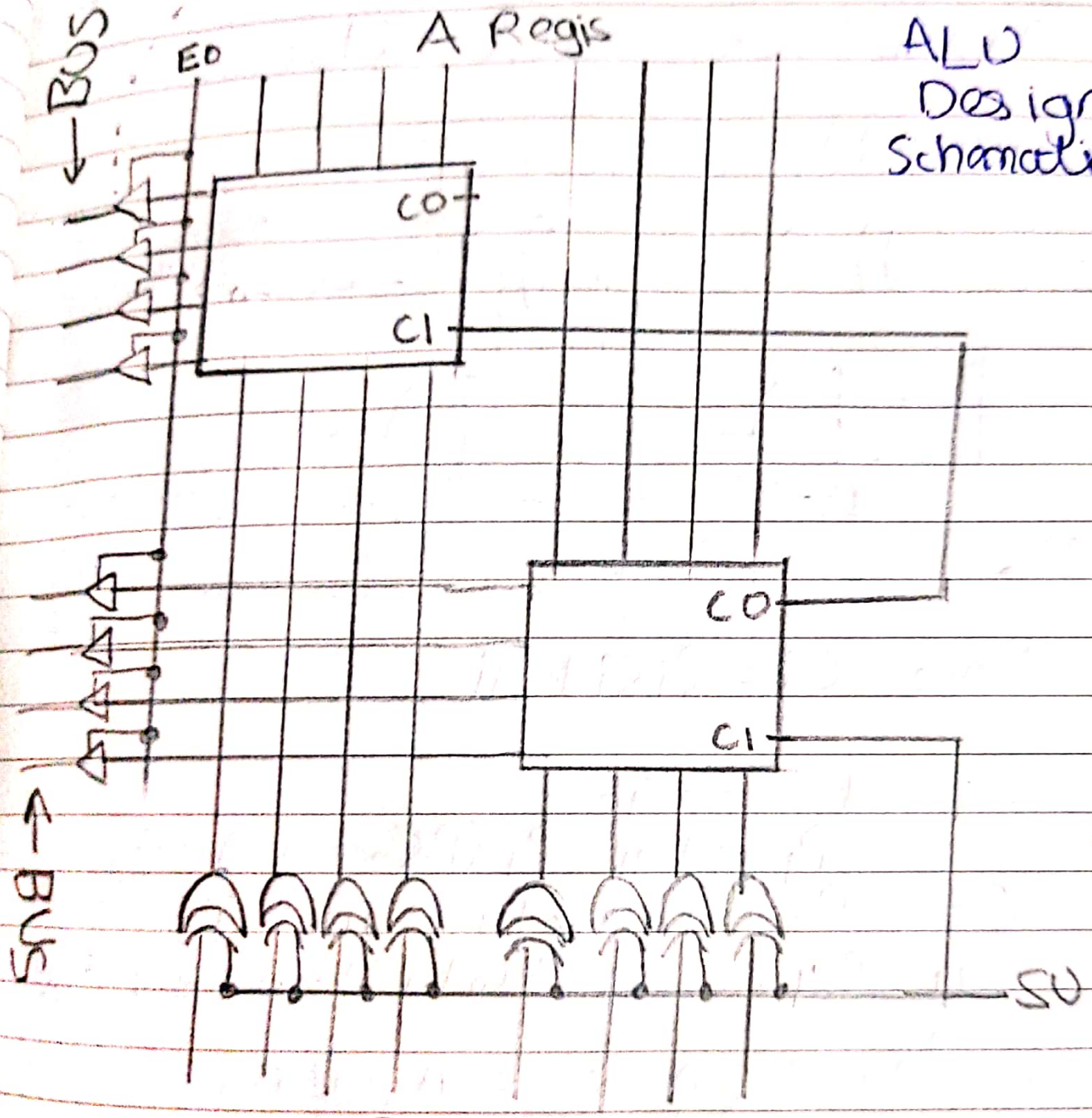
Scanned with CamScanner

05

Date: ___/___/___

→ tri-state buffers: data exits which open when EO = 1

ALU Design Schematic



XOR

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

for negating binary;
when; A = 1; Q = B
passing A = 0; Q = B
& carry over 1 (SU) when

creates 1's compliment

Haq, ek behtar zindagi ka.

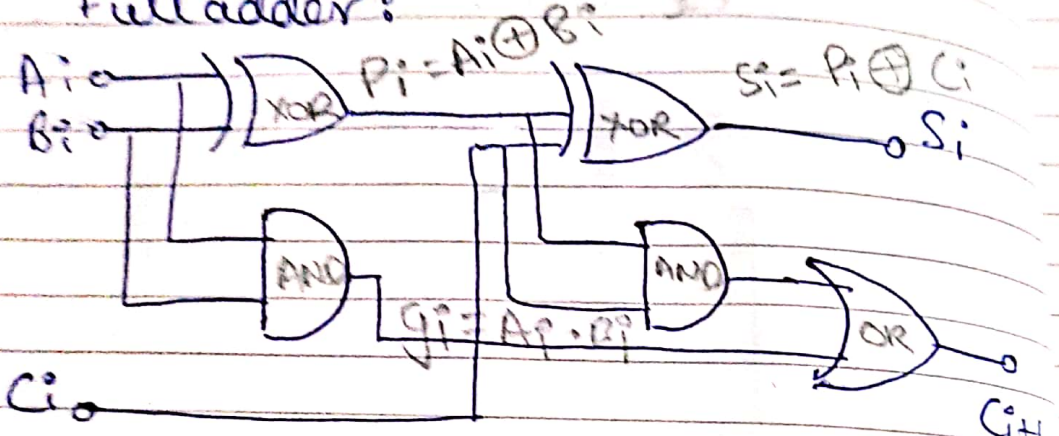
negating; creates 2's

Scanned with CamScanner

Date:

Carry look ahead adder (CLA)

Full adder:



Now; $C_{i+1} = P_i C_i + G_i$

$C_0 = C_0$ — (1)

$C_1 = P_0 C_0 + G_0$ — (2)

$C_2 = P_1 C_1 + G_1 = P_1 (P_0 C_0 + G_0) + G_1$

$C_3 = P_2 (P_1 (P_0 C_0 + G_0) + G_1) + G_2$

$C_4 = P_3 (P_2 (P_1 (P_0 C_0 + G_0) + G_1) + G_2) + G_3$

carry propagator: $P_i = f(A, B) = A_i \oplus B_i$

carry generator: $G_i = f(A, B) = A_i \cdot B_i$

So all C's $= f(A, B, C_0)$

only depending on C_0

Thereby, overcoming delay.

Haa, ek behtar zinda

Scanned with CamScanner

$C_4 = (A_3 \oplus B_3)((A_2 \oplus B_2)((A_1 \oplus B_1)$

Date: __/__/__

* CLA: 'predict the carry'

$C_0 = C_0$

$P_0 C_0 + G_0$

$C_1 = (A_0 \oplus B_0) \cdot C_0 + A_0 \cdot B_0$

$S_0 = P_0 \oplus C_0$

$= A_0 \oplus B_0 \oplus C_0$

$C_2 = (A_1 \oplus B_1) C_1 + A_1 B_1$

$= (A_1 \oplus B_1) \cdot ((A_0 \oplus B_0) C_0 + A_0 B_0) + A_1 B_1$

$S_2 = P_1 \oplus C_1$

$= (A_1 \oplus B_1) \oplus ((A_0 \oplus B_0) \cdot C_0 + A_0 B_0)$

$C_1 = P_0 C_0 + G_0$

$C_2 = P_1 C_1 + G_1$

$S_2 = P_2 \oplus C_2$

$= (A_2 \oplus B_2) \oplus [(A_1 \oplus B_1) \cdot C_1 + A_1 B_1]$

$C_3 = P_2 C_2 + G_2$ $\quad P_1 C_1 + G_1$

$(A_1 \oplus B_1)((A_1 \oplus B_1) \cdot C_0 + A_0 B_0) + A_1 B_1$

$S_3 = P_3 \oplus C_3$

$= [A_3 \oplus B_3] \oplus (A_2 \oplus B_2 \cdot C_2 + A_2 B_2)$

Haq, ek behtar zindagi ka.

$(A_1 \oplus B_1)[(A_1 \oplus B_1) C_0 + A_0 B_0]$

Scanned with CamScanner

Date:

S
CLR
EN
D
Q
I

1 2 4 8 16 32 64 128
1 1 0 1 0 0 0 0

1+2+8 = (11) → A

1111 = (15) → BUS

(15) → A

(10100(5) → BUS

(5) → B

EN ON →

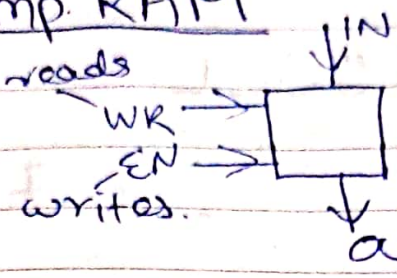
00101 (26) → BUS
4 16

Scanned with CamScanner

Date: 29/05/

8-bit Comp. RAM

1-b-Register :



RAM: Random access memory.

Static	Dynamic.
Combined registers.	Cap + Transistor.
Val. stored perm. till the power is not cut. Bulkier. More expensive. Faster.	↑ Stored as charge on it keeps discharging. But by refresher circuit we keep rewriting each bit.

We are going to create 16 bytes RAM (static) module.

16 bytes = 16 x 8 = 128 bit.

(16 → words, 8 → bit-lines.)

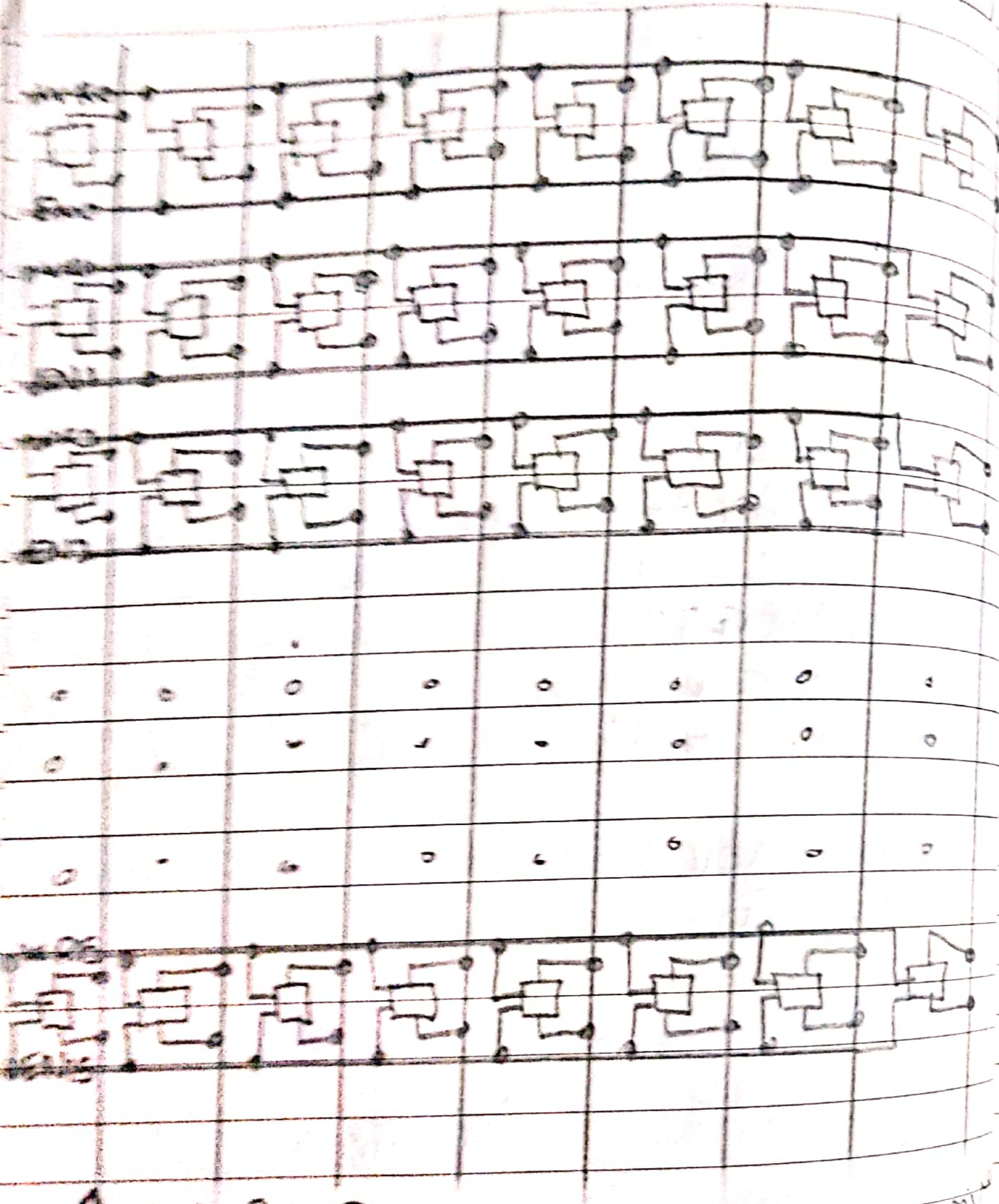
We are going to specify which word to write on by using 4-bit addresses and making their decoders

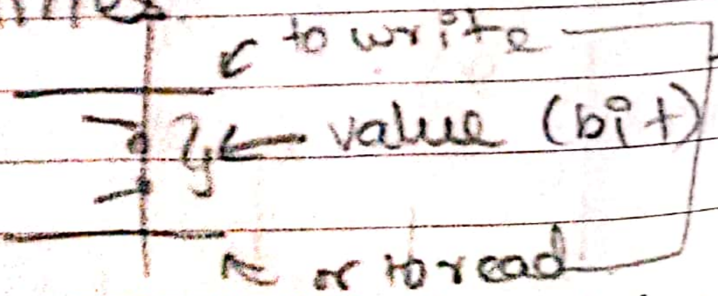
Haq, ek behtar zindagi ka.

Scanned with CamScanner

16 byte static RAM module schematic

Date:

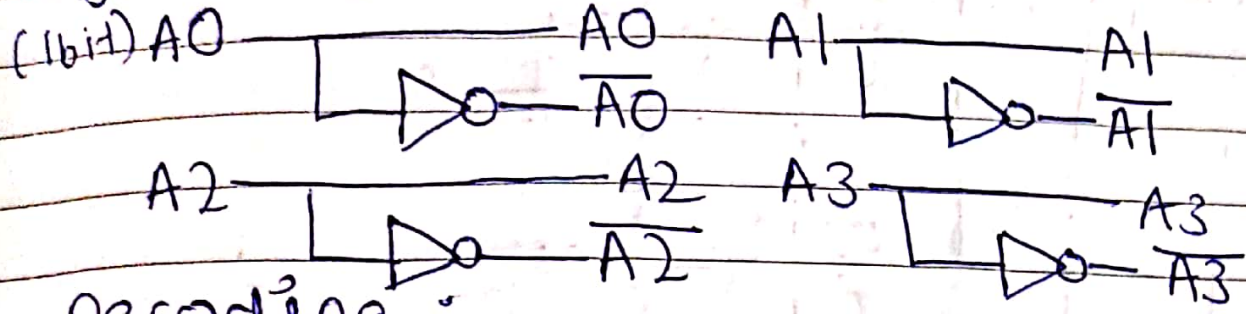




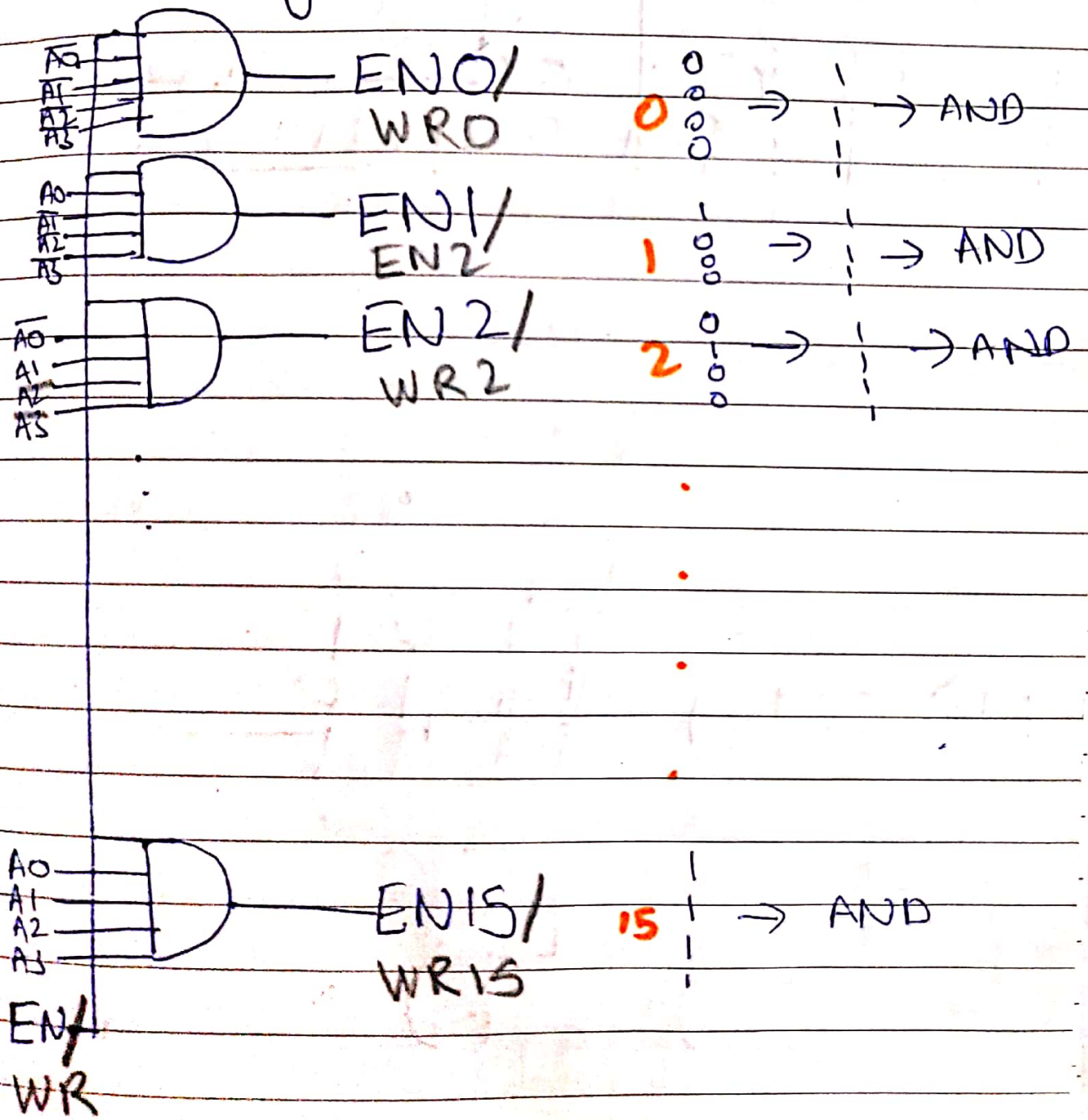
Address & Decoding

Date: ___/___/___

generating variables:



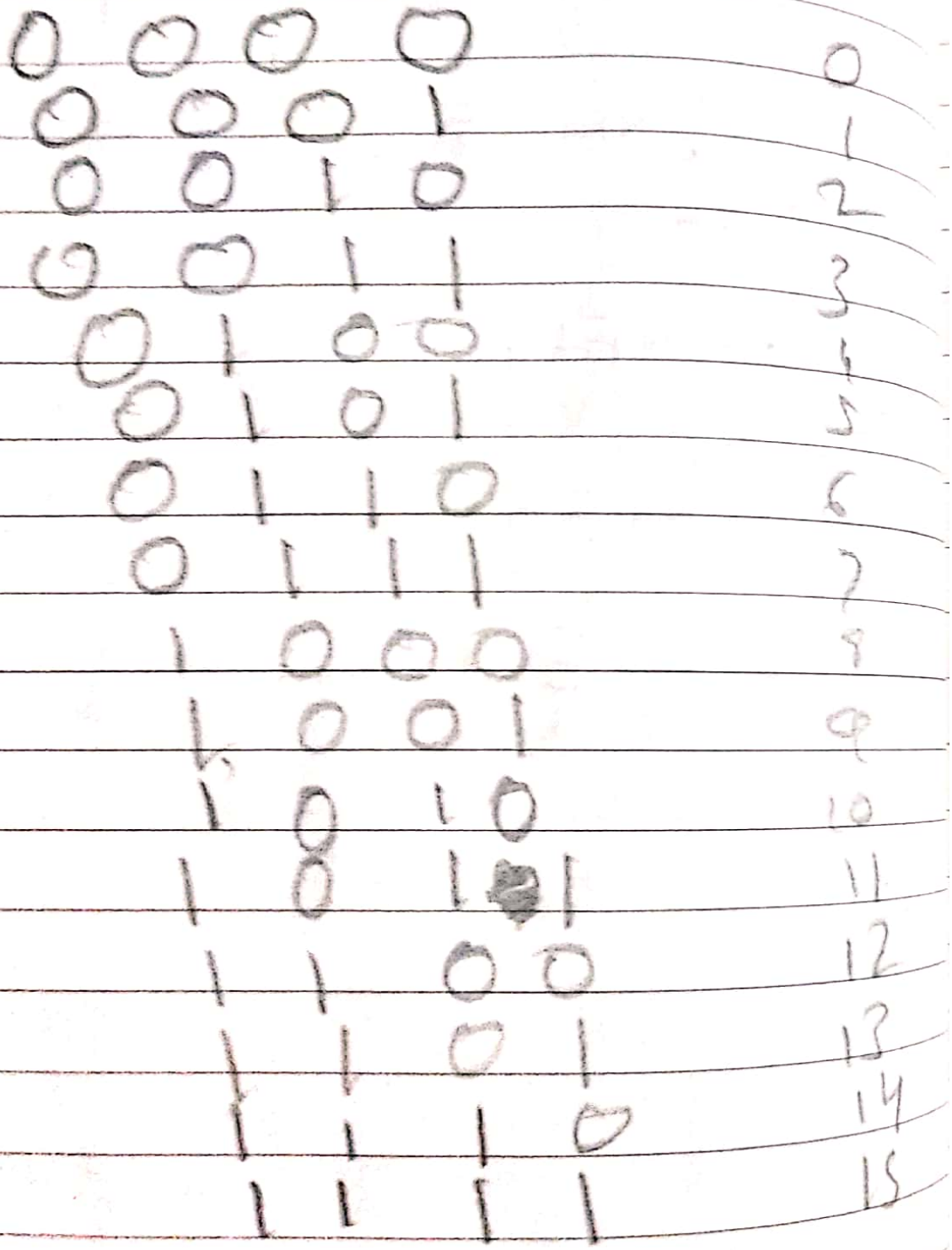
Decoding:



Scanned with CamScanner

Date:

Eg. 545/745189. (10:24)



Scanned with CamScanner

Date: 31/05/

Asynchronous Inputs

in D ff's the D input is triggered when the clock is active.

2 inputs which have no relation to the clk are called asynch. inputs.

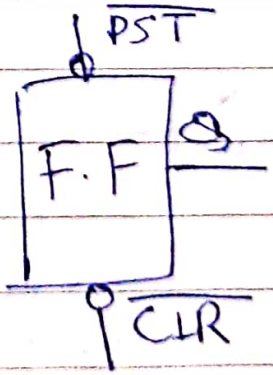
PRESET → $Q_{n+1} \rightarrow 1$ → output

Ac High = 1, Ac. low = 0

CLEAR → $Q_{n+1} \rightarrow 0$ → Output

Ac High = 1, Ac. Low = 0

PST	CLR	Q_{n+1}
0	0	Prohibited
0	1	1
1	0	0
1	1	Q_n



Both Ac. Low

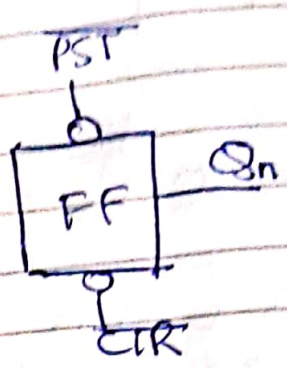
Flipped for Ac. High.

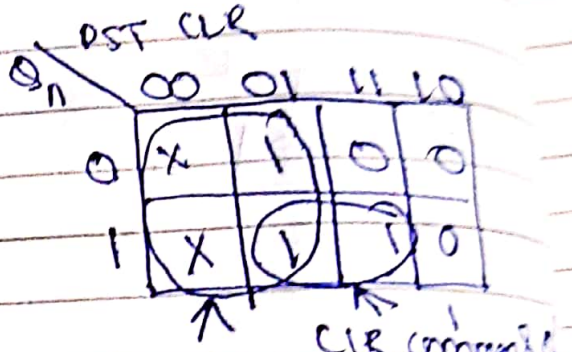
Ugo ek behtar zindagi ka.

Scanned with CamScanner

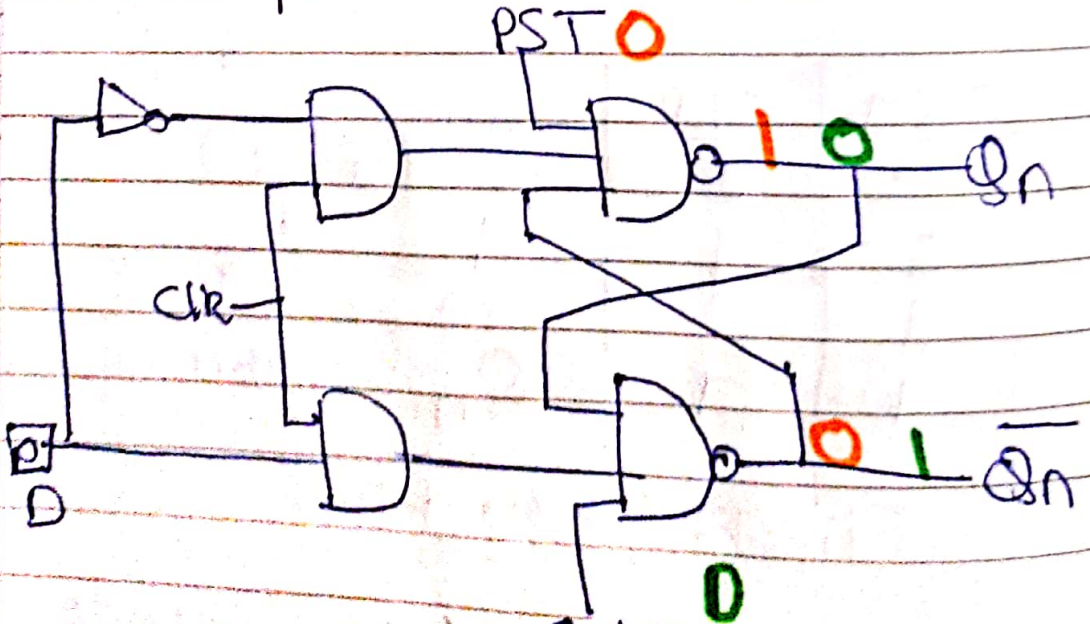
Date:

Q_n	PST	CLR	Q_{n+1}
0	0	0	X
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	X
1	0	1	1
1	1	0	0
1	1	1	1





characteristic eq: $Q_{n+1} = \overline{PST} + Q_n \cdot \overline{CLR}$

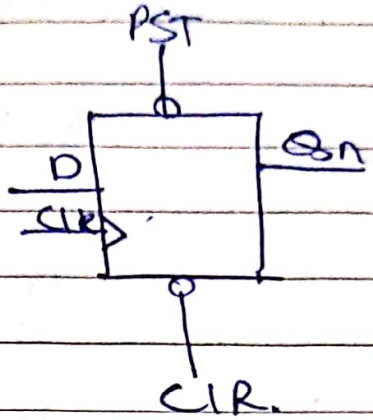


Haq, ek behtar zindag

Scanned with CamScanner

Date: 01/06/

PST	CLR	Qn
0	0	X
0	1	1
1	0	0
1	1	Qn

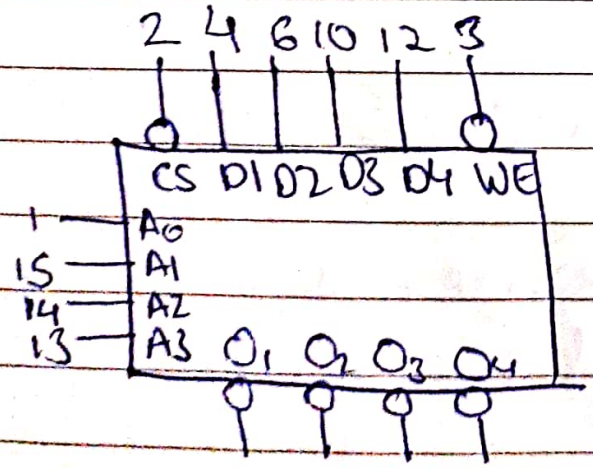


RAM (logism) (Ben eater)

Vids: Part 1 —

54S/74S189 } chip
54LS/74LS189 } name.

16 - 4 bit words
so we use 2 chips.



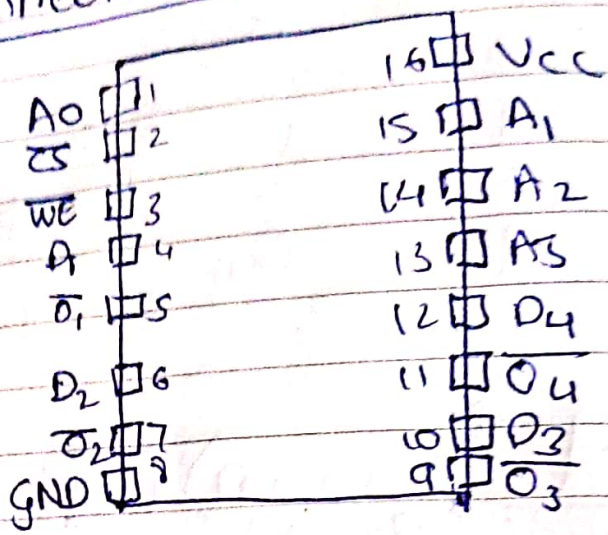
Logic Symbol

we are going to have to reinvert these: we will use DM7404 chips.

Scanned with CamScanner

Date:

Connection diagram.



CS: Chip Select: Ac. Low

$A_0A_1A_2A_3$: 4bit addr.
$P_1D_2P_3D_4$: input (4bit)
$\overline{O_1}\,\overline{O_2}\,\overline{O_3}\,\overline{O_4}$: inverted output

$\overline{WE}$: Write Enable.

When we power up we get some garbage values inside

For memory address register:
74LS173.

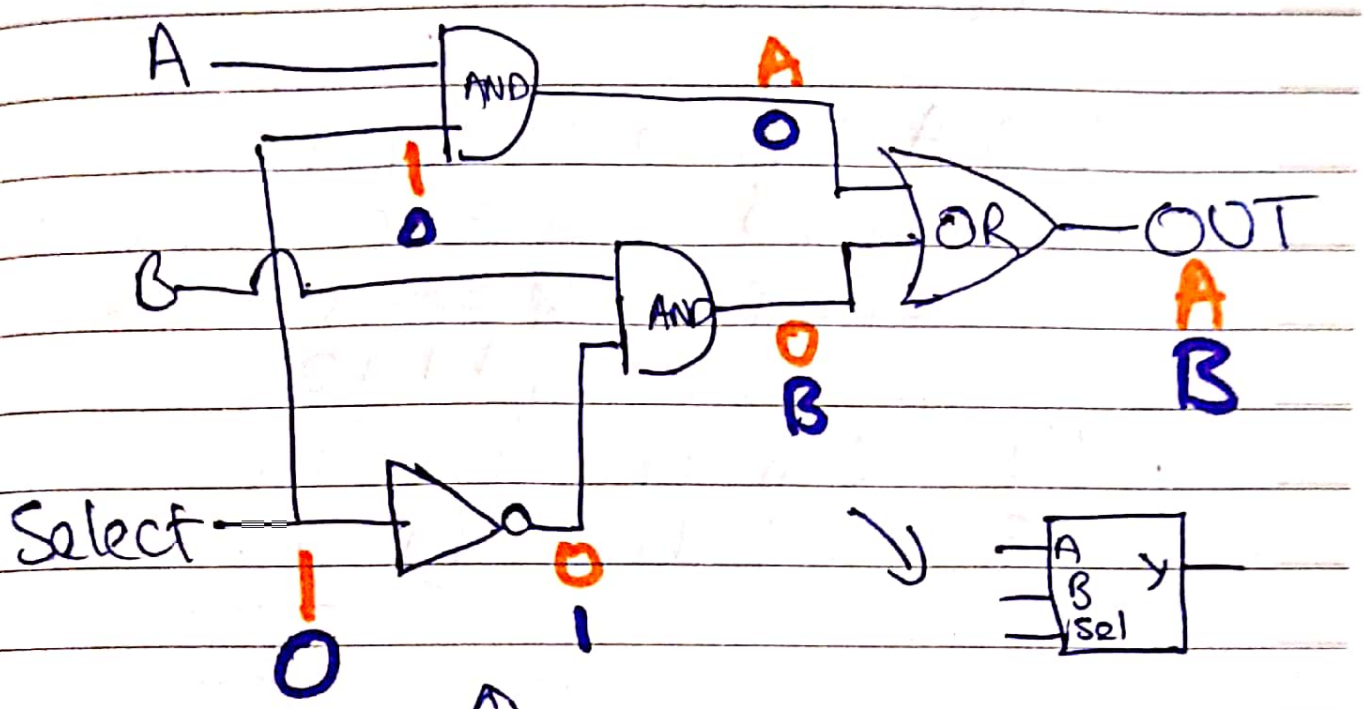
Haq, ek behtar zindag

Scanned with CamScanner

We want to be able to switch between manual / bus input for address.

A → manual

B → bus.



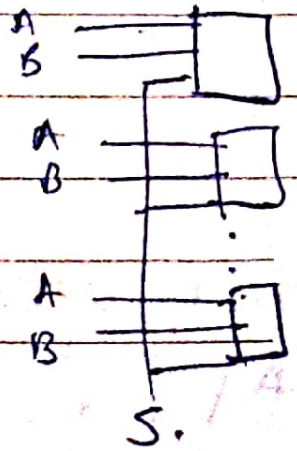
there is a chip for this.

Quad-2-line to 1-line Data Selector / Multiplexers.

DM74LS157.

DM74LS158

Logic Diag.



Haq, ek behtar zindagi ka.

Scanned with CamScanner

Connection Diag.
<See the data sheet>
<straightforwards>

Control signal

LDA 14 → 0001 1110
ADD 15 0010 1111
OUTPUT ~~0011~~ xxxx
1110

14: 0b (1101) = 13
15: 19 (11001) = 19
32

PC
LDA 14: ~~MI~~ CO MI
RO II

0001 0001
16 1 = 17

	LDA 14	ADD 15	OUT
T0	CO MI	CO MI	CO MI
T1	RO II CE	RO II CE	RO II CE
T2	IO MI	IO MI	AO OI
T3	RO AI	RO BI	
T4		EO AI	

Haq, ek behtar zindg

Control
INSTR.
DA
ADD
OUT
HLT

Scanned with CamScanner

Date: 27/06/

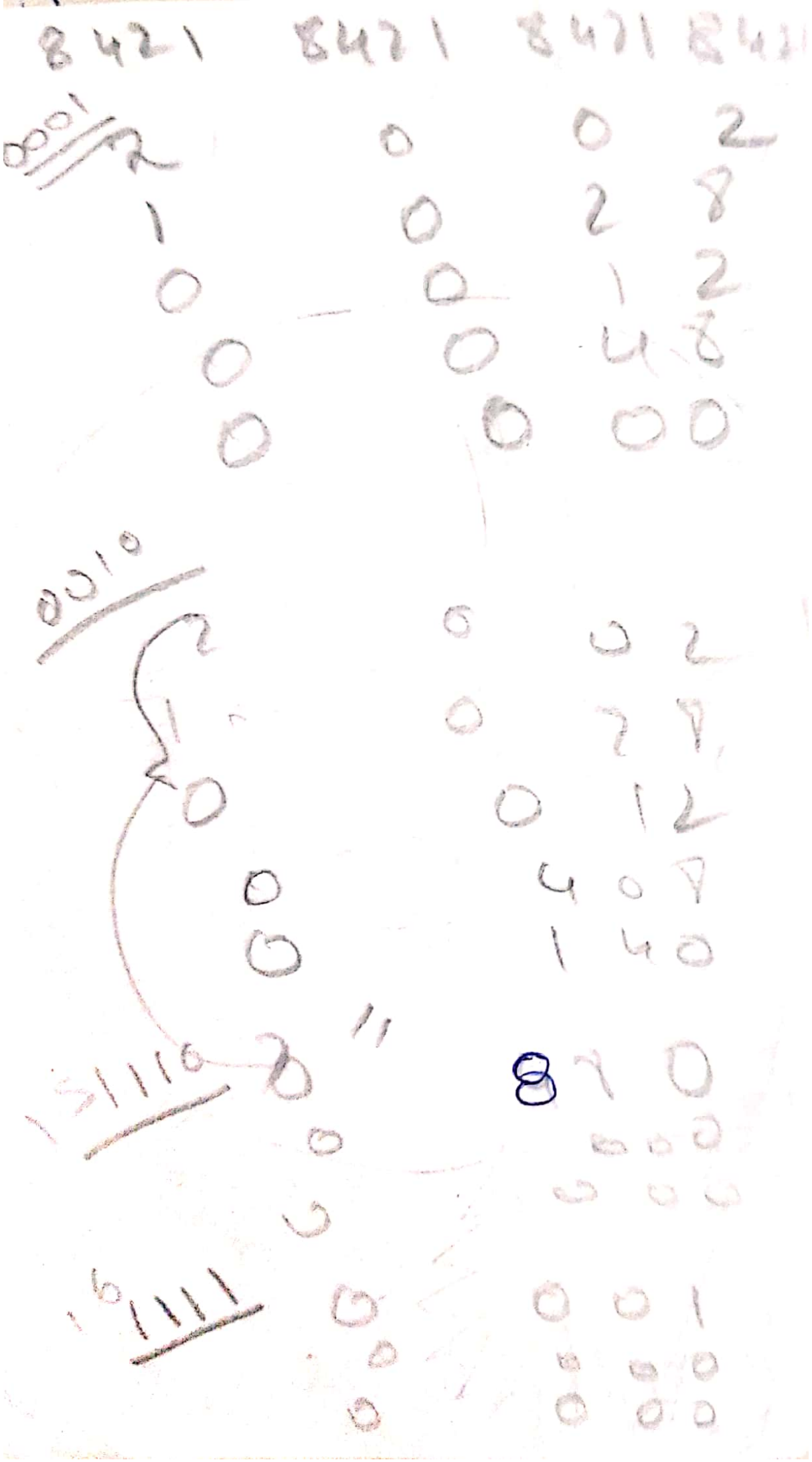
Control logic EEPROM.

	INSTR.	STEP	HLT	MI	RI	RO	IO	II	AI	AO	EO	SU	BI	OI	CE	CO	J
FETCH	xxxx	000	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
	xxxx	001	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0
LDA	0001	010	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
	0001	011	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	0001	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADD	0010	010	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
	0010	011	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
	0010	100	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
OUT	1110	010	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	1110	011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1110	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HLT	1111	010	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1111	011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1111	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RAM: 1e 2f e0 f0

Haq, ek behtar zindagi ka.

Scanned with CamScanner



Scanned with CamScanner

JMP 0011 (0000)(0000)(0000)(0001)

STA 0100 OP: 1011 M1 (0100)(1000)

0011 A0/R1 (0010)(0001)

LDI → puts from IR instead of RAM: 0101 1011 A1

(0000) (0010)

SUB → 0110 M1 1.0 0012

R0 B1 1 SUB 0608 1 A

00011110 LDA .1 4 SUM ALLEO 0340

00101111 ADD STA 15 1 AU

STA 15 0 B

01001111 STA 14 ADDN

OUTPUT

STA 14 // 14

JMP 0001 // 0

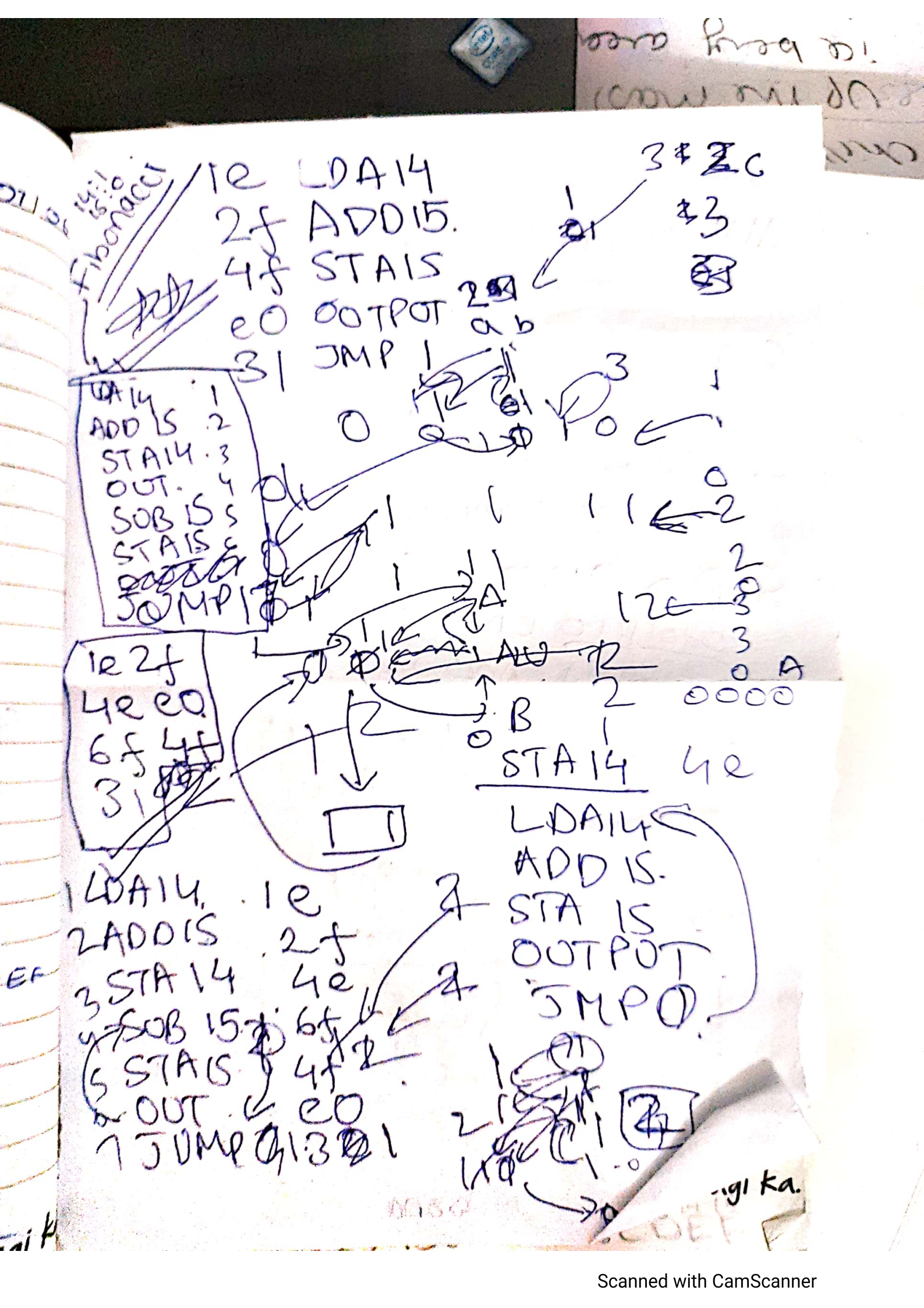
00110001 LDA. 1

1e2f4fe0e310

1 2 3 4 5 6

Fibonacci code.

Scanned with CamScanner



Scanned with CamScanner

NAND

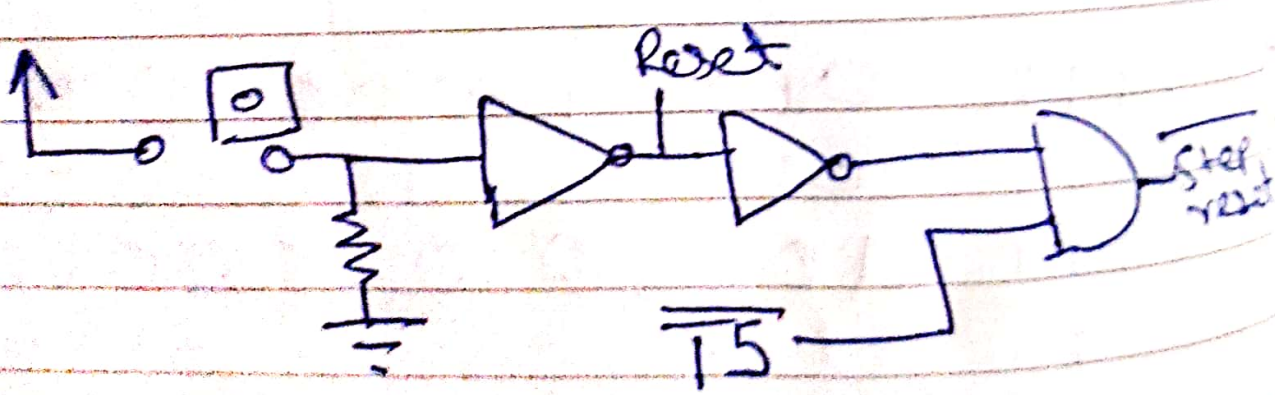
W1 CPU, Memory Sys, Hardware

P1: Logic gates (15 chips)
P2: ALU (adder chips)
P3: RAM (Reg's & Mem Unit)
P4: Mach. Lang.
P5: CPU
P6: Assembler.

Course1: Part1: Make Comp Sys Book
Course2: Part2: Jack, OS, VM, Compiler

Ben

Reset: (Circuit)

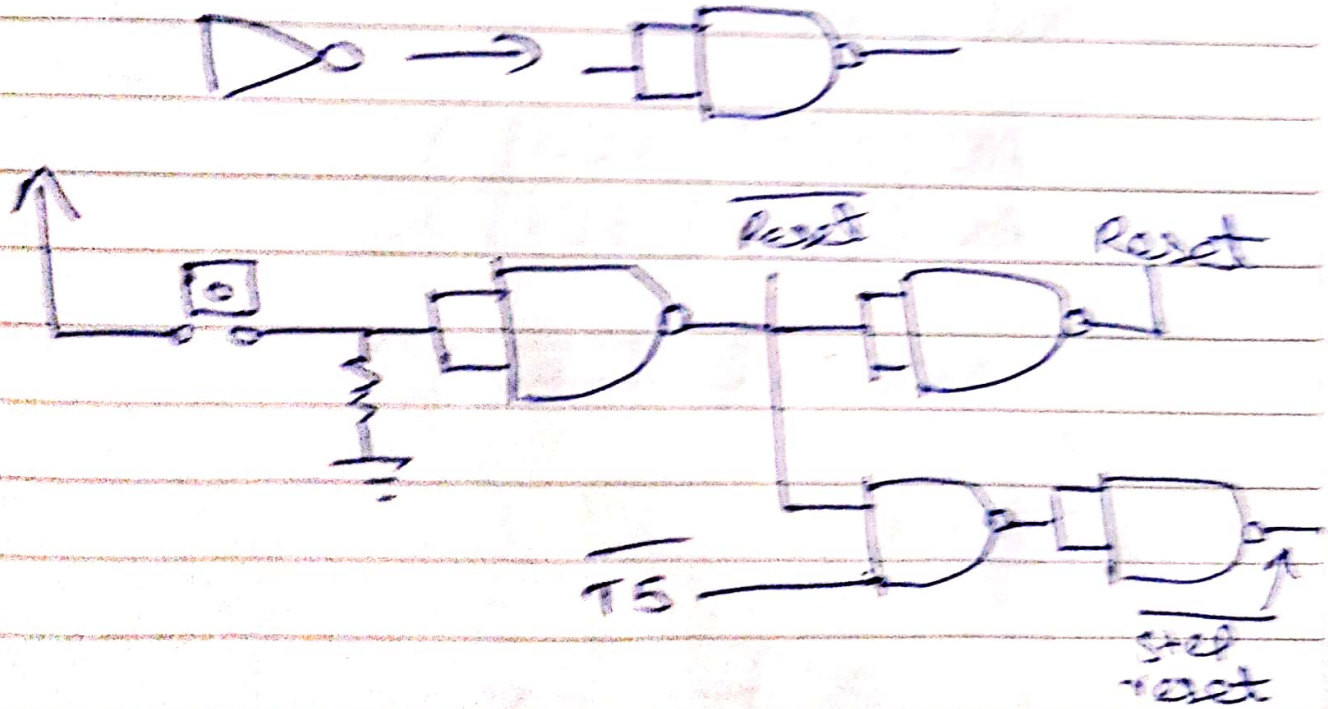


Haq, ek behtar zindg

Scanned with CamScanner

Date: 10/05/

NAND eq. circuit is used in one chip



Add ALU functions in simplest logic

HACK-E2 (Proj 1 (chips))

Scanned with CamScanner

HACKW2 (Still E2)

Date: ___/___/___

Dec → Binary.

eg 99

get the biggest pow of 2.

$2^6 = 64$

& sub

99

$-\underline{64}$

35

$\underline{32}$ $2^5 = 32$

3 $2^1 = 2$

$\underline{2}$

1 2^0

make indexes

1 1 0 0 0 1 1

6 5 4 3 2 1 0

Overflow happens (carry bit ignored and result gets truncated)

Adders:-

1. Half adder: adds 2 bits
2. full adder: adds 3 bits
3. Adder : Adds 2 no.'s

Scanned with CamScanner

Negative no's

2's Complement

0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	-8
1 0 0 1	-7
1 0 1 0	-6
1 0 1 1	-5
1 1 0 0	-4
1 1 0 1	-3
1 1 1 0	-2
1 1 1 1	-1

(+) ve

(-) ve

★ $-x(\text{neg. no}) = 2^n - x$.

$\Rightarrow \quad -3 = 16 - 3 = 13$

$-5 = 16 - 5 = 11$

Scanned with CamScanner

Date: ___/___/___

In 2's complement

$$\begin{array}{r} -2 \\ +\ -3 \\ \hline -5 \end{array} \Rightarrow \begin{array}{r} 14 \\ +\ 13 \\ \hline 11 \end{array} \Rightarrow \begin{array}{r} 1110 \\ +\ 1101 \\ \hline 1011 \end{array}$$

gets thrown away.

$$\begin{array}{r} 7 \\ +\ -5 \\ \hline 2 \end{array}$$

same logic.

Computing (-x)

IN x ; OUT -x.

Cause $y - x = y + (-x)$

flips all bits of x

Idea: $2^n - x = 1 + (2^n - 1) - x$

adds 1 to flipped bit o/p.

$$\begin{array}{r} 111111\ldots n \\ -\ 101010 \leftarrow x \\ \hline 010101 \\ +\quad\ 1 \\ \hline 010110 \end{array}$$

flipped bit

2's complement

Haq ek behtar zindagi ka.

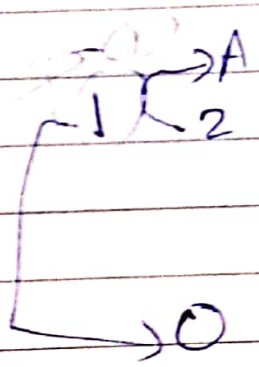
Scanned with CamScanner

Date: 13/06

Ben

Fibonacci 15:1

0 LDA 14
1 ADD 15
2 A → RAM15
3 A → OUTPUT
4 JMP 0.



14 is
1 0

0 1 1 2 3 5 8

1 0
15 → 14
14 → sum.

1 1
1 0
1 1 2
2 1 3
3 2 5
5 3 8
8 5

Scanned with CamScanner