

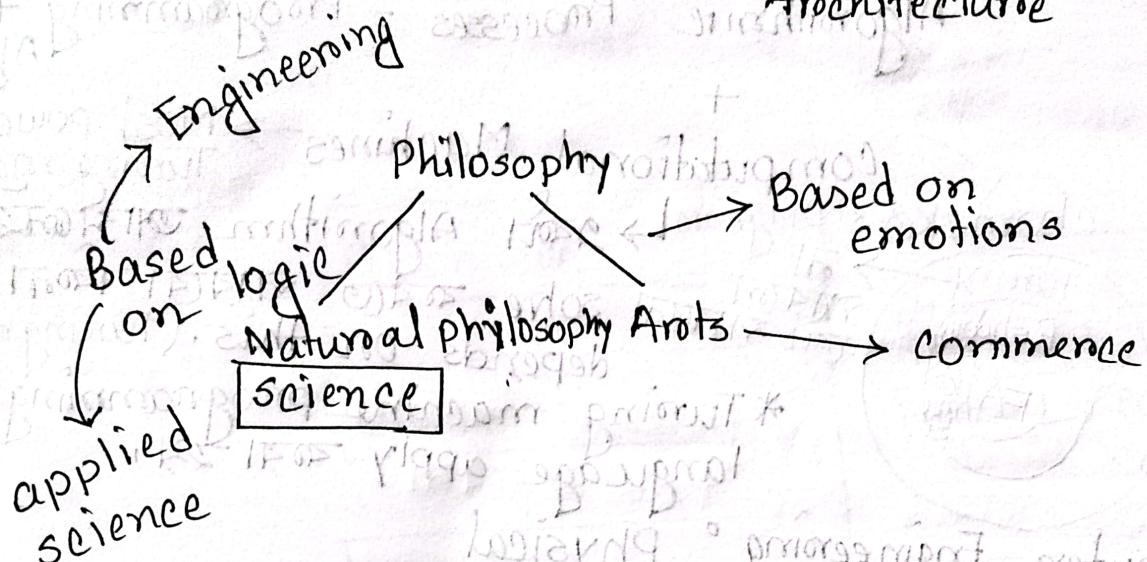
Sat  Sun  Mon  Tue  Wed  Thu  Fri

Date: 18/09/2023

Sub:

## CSE 3203 - Computer

### Architecture



\* Engineering → Science এর application  
\* specific on best way এরে problem solve কৰবলৈ.

\* Building block তৈরি কৰবলৈ → The main idea.

\* প্রযোগ concept হৈবলৈ কৰতে, কাম কৰবলৈ so that বড় তিনিম তৈরি কৰিব।

\* Transistors - ফুটি computer তৈরি, প্রযোগ হৈবলৈ transistor থেকে computer তৈরি।

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

## Computer Science : Abstract

Algorithmic Processes = Programming + Algorithm

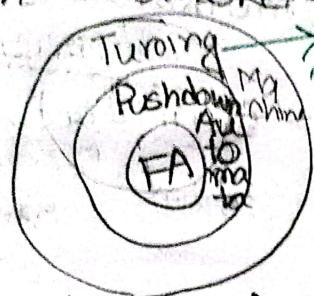
Problems  
Step by Step  
solve করা

+

Computational Machines

Most powerful  
Turing machine

Noam Chomsky's Diagram → একটি Algorithm আসলেই



না  
সম্ভব  
কর্ম করতে পারবে নি।  
যেটি সাধারণ। solve depends on this. (Turing machine)

\* Turing machine programming language apply করা যাব।

## Computer Engineering : Physical

\* Computer engineering এর বাইরে Turing machine বানানো,

\* Turing machine actually বানানো possible না  
কারণ infinite memory নাই,

\* Turing machine এর concept use করে finite  
memory use করে Turing compatible (CPU)  
বানানো যায়।

Relay → Vacuum Tube → Transistor

Mechanical

1st use  
কর্ম হয়েছিল।

(Reliable না)

Mechanical

Relay র পর

আড়া।

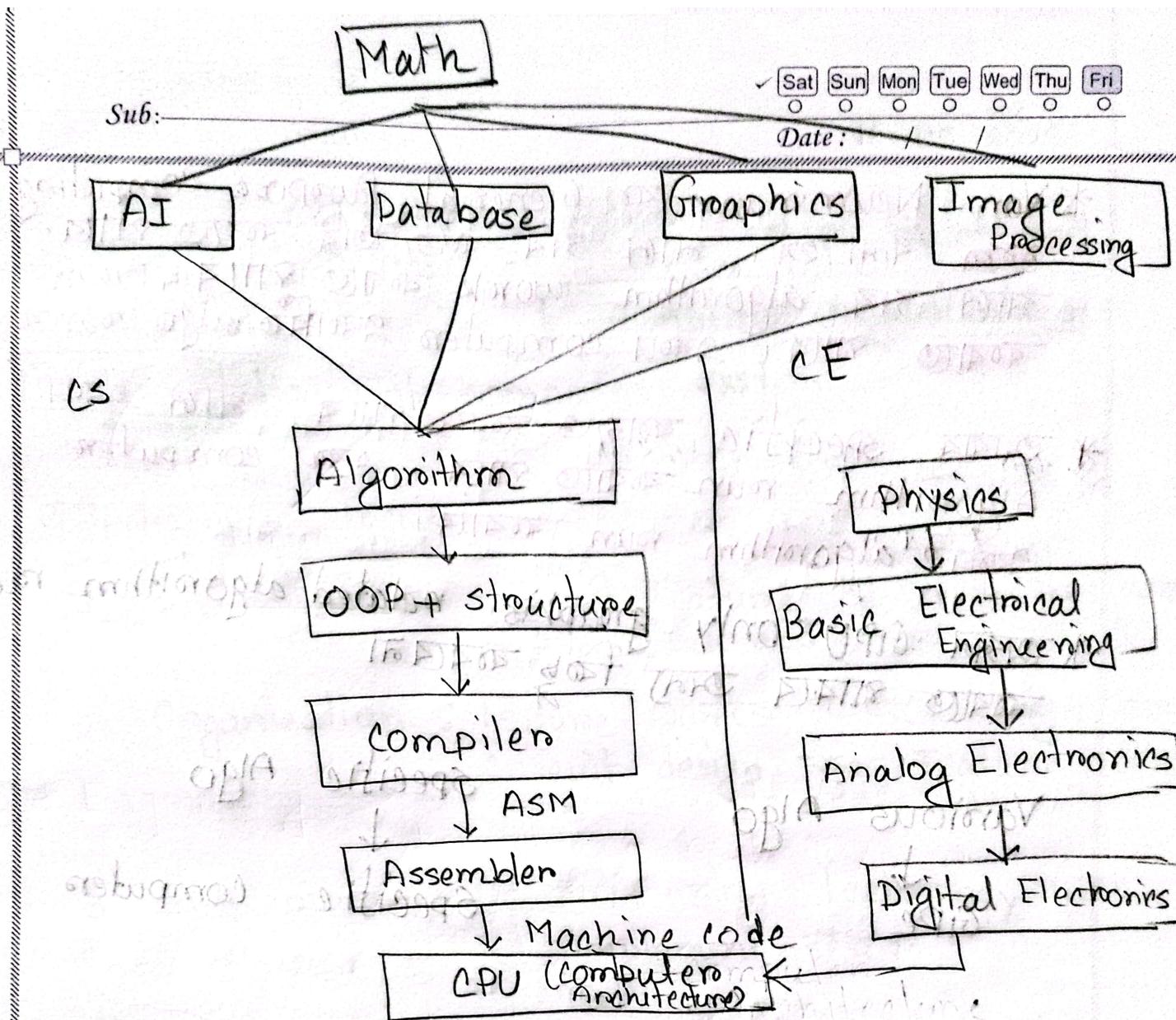
(blast হয়ে  
যায়।)

আধুনিক।

অনেক ছোট

size (nm)

পোর্টেন।



\* Building block : Transistor যা ট্রান্সিস্টর বানাতে

\* Computer science Math based

\* Computer Engineers Physics based

\* Computer Science এ building block নাই বা dependencies নাই প্রস্তুতি engineering এ আছে।

\* CPU ট্রান্সিস্টর Von Neumann Architecture ফলে use হয়। Von Neumann Architecture Turing compatible

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

\* Von Neumann দ্বাৰা General Purpose Computing  
জনকৈ বলা হয়, মান সব কাজ তাৰি কৰতে পাৰে,  
মান সব algorithm work কৰাতে পাৰে বা run  
কৰাতে পাৰে, একটা computer একাধিক algo run কৰে,

\*আবার specific কাজ কৰতে পাৰে, মান একটা  
Algorithm run কৰাতে পাৰে, একটা computer  
একটা algorithm run কৰায়।

\*অন্য GPU only graphics related algorithm run  
কৰাতে পাৰবে অন্য কিছু কৰবেনা

Various Algo  
↓  
GPU

specific Algo  
↓  
specific Computer

Sub:-

✓ Sat Sun Mon Tue Wed Thu Fri

Date: 20 / 09 / 2023

Computer Architecture is a set of rules and methods that describe functionality, organizations & implementation of computer system.

প্রার্থিতিগ্রহ Focus :

1. Functionality : কী কী Feature বা instruction আছে  
16 bit or 32 bit, add or sub, word size

2. Organization : Feature গুলোকে কীভাবে design করবেন  
circuit design. কোরা Feature ইয়েসে

3. Implementation : কোন এজন technology দিয়ে  
implement করবে.

CA = Microarchitecture + Instruction Set Architecture (ISA)

\* প্রার্থিত এবং ISA আছে অল্পাদ। ISA  
থেকে Functionality বৃক্ষ আছে,

VLSI → কোনা function কে digital circuit  
convert

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

\* ISA টা হলো Functionality

\* Microarchitecture হলো Organization & implementation.

\* কী কী Feature তথ্য এটা জানা যাবে ISA র সাথীয়ে, ISA বলে দেখা machine code অথবা কোড হবে। ".exe" File এ run করাতে হলো ক্ষেত্র। "মোড়ার তন্ত্র" specific ISA use ISA প্রাক্তন হবে। মোড়ার তন্ত্র result পাওনা, না করলে result পাওনা।

Intel Core i3  
Core i5      } ISA  
Core i7      } Same

\* "Hello.exe" File core i3 তে create করলে core i7 এর কী run হবে ?

— Run হবে, বরাবর এদের ISA same. অহ সংজ্ঞা হবে as machine code same.

\* ARM থেকে generate করে "Hello.exe" File core i7 and Ryzen এ চলবে ?

— Intel & Ryzen আলাদা company হলেও Run করবে।

— বরাবর IC same মধ্যে, কিন্তু ARM এ চলবেন।

— বরাবর ARM এর IC আলাদা।

Sub:

✓  Sat  Sun  Mon  Tue  Wed  Thu  Fri

Date: / /

\* 8086 এর কোড কিরে core i3 তে run  
হবে ?

— 8086 16 bit processor ও core i3 64 bit  
processor, ISA এন্ড করে design করা ছিল  
Future predict করে যেন 8086 এর  
code core i3 তে চালানো যাবে।

X86 (32 bit)

X86-64 (64 bit)

কিন্তু windows OS করার পর  
যায়না, তখন restrictions আসবে,

\* 8086 এ অনেক কিছু  
নাই, কিন্তু স্বাক্ষর করা  
অনেক কষতি মাঝে  
যেন Future এ support  
থাকে, আব এই  
support থাকে বলে হেলতো,

যোগী program এ Feature add করলে যোঁ use করা  
হবে, যাতে Feature support থাকে, যা না থাকলে  
run করতো না,

\* IS64 use করেনি কারণ Feature support নাই  
তাই পুরানো  
করতে চাইনি, তবে C/C++ recompile করলে করা  
যাতো, কিন্তু যোঁ ক্ষমতার অনেক।

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Sub: \_\_\_\_\_

Date: / /

\* Intel ২০লে, amd ৬৪ follow করে যেখানে এই Feature support এর সমাজেলা নাই।

\* Deprecated এসব Function থেকে যেনে Feature remove করতে চাই ~~কিন্তু~~ but remove করতে দয়না, কিন্তু সমাজেলা আছে।

8086 ADD AX, BX → 16 bit

come i3 ADD RAX, RBX → 64 bit

\* নতুন processor এ নতুন কী ?

— নতুন কিছি Feature provide করে,

Implementation fasten. Faster করে কীভাবে, instruction এর extension থাকে যা নতুন নতুন instruction করে, কিন্তু কীভাবে better performance provide করে, যেটা matter করে।

যানো 1920 x 1080 quality র video যখন 8086 খালি চালালো এবং তাহলে 2x106 বার Function করবে pixel wise change হতে অনেক time লাগব। মাত্র hardware support থাবার না।

Sub: \_\_\_\_\_

Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○

Date: / /

2x106

\* মনে এখন একটা way থের  
বরবরে এন অবজ্যুলো function  
parallelly process করবার প্রা  
Faster result থিব।

+ instruction এখন but implementation 100%  
Optimized ২৫%, performance ১০০%

ADD100 R<sub>2</sub>, R<sub>1</sub>

$$\begin{bmatrix} R_1 \\ \vdots \\ R_1 \end{bmatrix} + \begin{bmatrix} R_2 \\ \vdots \\ R_2 \end{bmatrix} = \begin{bmatrix} R_2 \\ \vdots \\ R_2 \end{bmatrix} \times 100$$

\* মেগানে কাজের জন্য hardware support লাগবে  
মনে নতুন processor এর performance optimize  
হবে, 4k video. ইধাও।

\* core i3, i5, core i7 এর ISA same but design  
আলাদা, তার hardware এর সাথে communication  
যা ISA same কিন্তু কৃটীর architecture আলাদা  
এবং microarchitecture.

Computer Arch



ISA = Function

Microprocessor = organization +  
architecture

n Implementation

- \* .exe file run হতে কিনা depends on ISA
- \* ISA same হলে microprocessors microarchitecture আলোনি
- \* সব version এ কৃতি function same না,
- \* Transistor এর সংখ্যা জো transistor বেশি রাখা মাত্র করে একটি বড়া করা যাবে, performance ডালো হবে,
- \* এখনকার CPU এর transistor ক্ষেত্র nm: 512 এবং হচ্ছে
- \* Transistor size ছোট হলে speed বেশি,
- \* Post Moore Era :  
Moore's Law প্রযোজ্য না, Moore's law অনুসৰী transistor size কিন্তু হলে clockspeed বাড়বে.

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: \_\_\_\_\_

Date: / /

\* Clock speed বাড়ানোর এখন আর possible না।  
কারণ হ্রাস করতে করতে এতে বনিয়েছে এই আর  
হ্রাস করা possible না transistor দ্বা।

Relationship between software and hardware:

'exe'  
'a.out' { Machine code , ISA part of hardware  
follows ISA  
'dmg'

\* যেকোনো code run করাতে হলে ISA follow  
বুবুতে হবে।

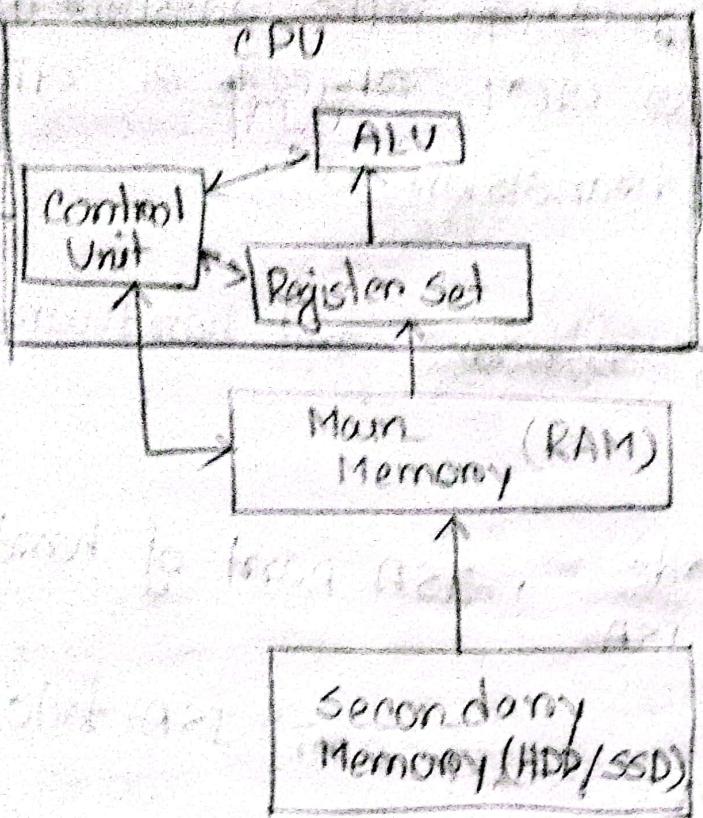
High Level Code → Compiler

Assembly code → Assembler

Machine code → CPU runs

CPU কীভাবে code run করায়?

General Purpose CPU follows Von-Neumann architecture



(Sat) (Sun) (Mon) (Tue) (Wed) (Thu)  
Date: / /

ALU  $\rightarrow$  MOV  
~~ADD~~ XOR  $\rightarrow$  ~~MOV~~  
~~XOR AX, AX~~  
~~ADD AX, 2~~

$\rightarrow$  MOV AX, 2  
 $\rightarrow$  UMOV a, AX

---

MOV BX, 3  
 MOV b, BX

---

ADD AX, BX  
 MOV c, AX

0011 01 0010  
 0011 10 0011  
 0001 01 10XX

$a = 2$        $\rightarrow$  MOV AX, 2  
 $b = 3$        $\Rightarrow$  MOV BX, 3  $\Rightarrow$   
 $a = a + b$       ADD AX, BX

\* CPU chip এর RAM এর chip আলান।

\* Data registers এর ALU রে মোভ তেন data RAM  
 এ স্টার্ট মাত্র এর operation ALU রে রেজ।

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 26/09/2023

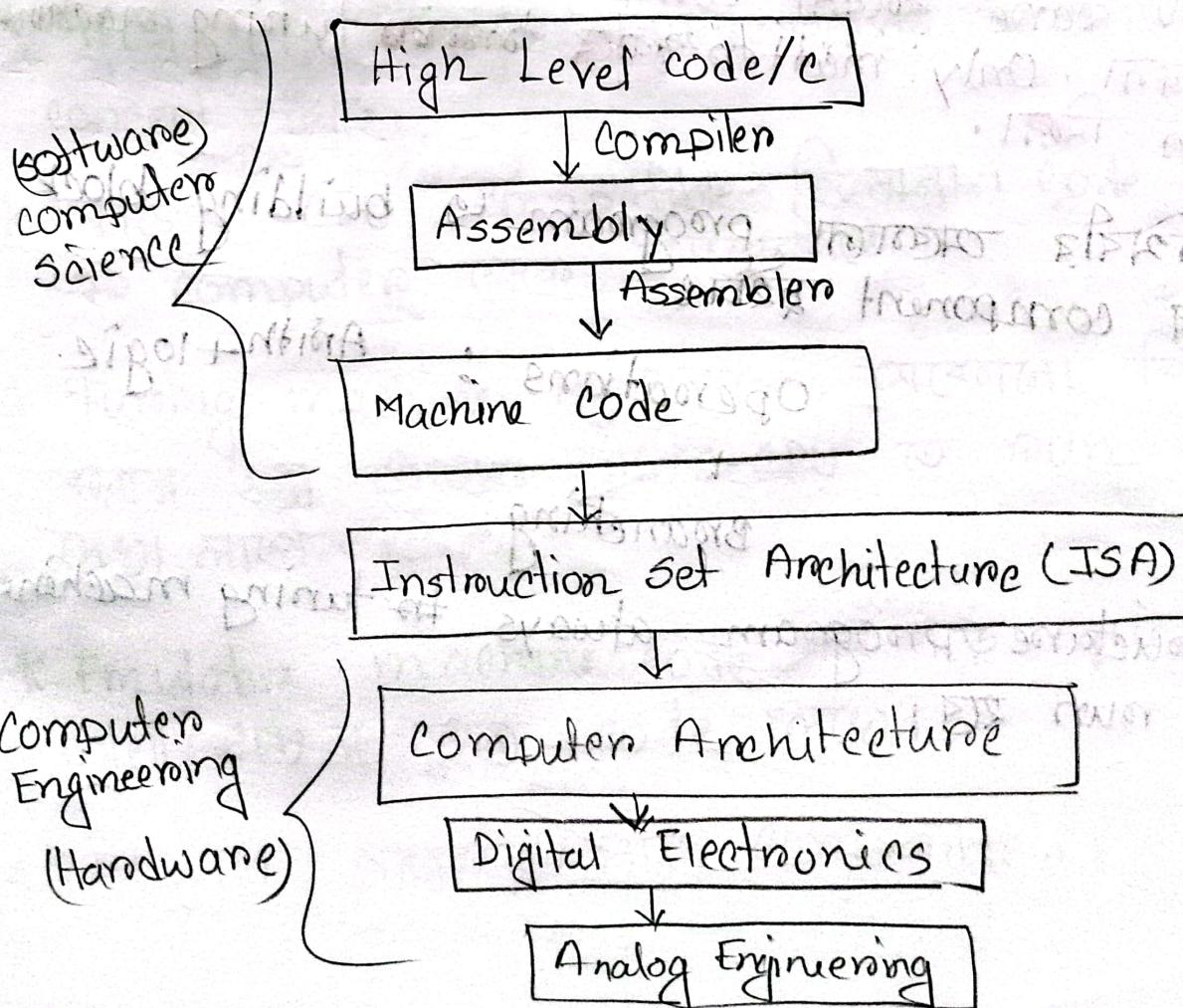
\* CPU Text কুমোনী

\* CPU থেকে চারিয়ে মেডিয়েশন দিতে হবে।

Compilers → High level code translate to machine  
মেমন: C, C++ Code  
Vs

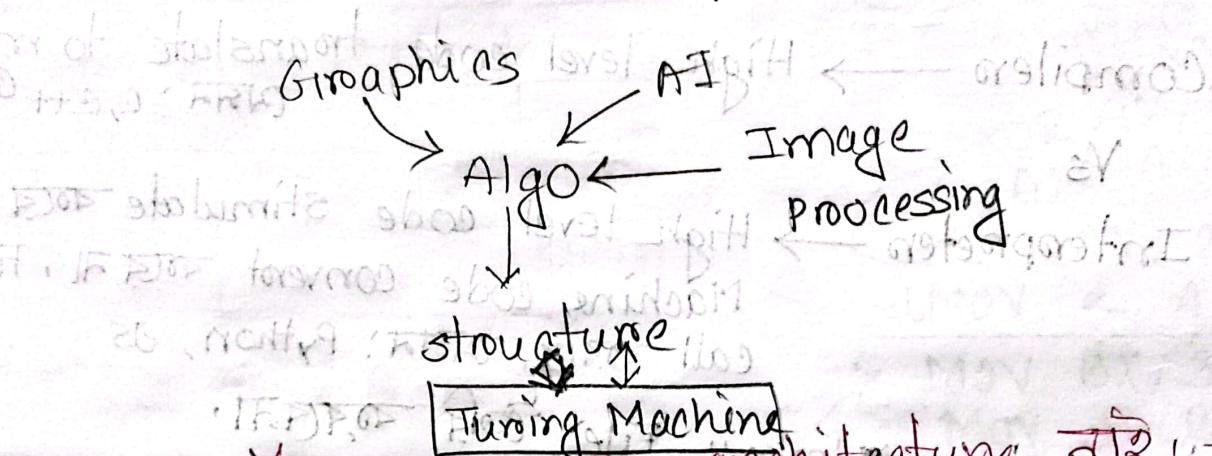
Interpreters → High level code stimulate করে  
Machine code convert করেনা, Function  
call করে, মেমন: Python, JS

\* Interpreter "exe" File তেরি কুরশনা।



\* যখন code run হবে transistor এ, ~~প্রসেসর~~

\* RAM and CPU always ~~প্রসেসর~~ সালাদ থাবলৈ।



\* CS এ Von-neumann architecture নাই, এবং  
CPU করেনা, computer architecture matter  
করেনা, Only matter করবে ~~turing machine~~  
চলে যাবে।

\* প্রমিকীয় মেণ্টেন program এ building block  
এবং component হবে।

Operations : Arith + logic  
+  
Branching

\* structure program always ~~to~~ turing machine  
এ run হবে।

✓  Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub:

Date: / /

\* Real life  $\leftrightarrow$  turing machine  $\rightarrow$  Gold standard  
বিদ্বান হয়।

\* CPU turing machine মেন চলে, আবে Von Neumann turing machine follow করে, কিন্তু in real turing machine তত্ত্ব করা possible না কারণ infinite memory নাই, এজন্য CPUকে Turing complete CPU বলা হয়, CPU turing machine concept follow করে।

\* আগে CPU হিসেবে মানানো তাৰপৰ turing machine concept অসমুছে।

\* আগের CPU compatible ছিলো না। Code টলতোলা এক computer থেকে আরেকটা থেকে।

\* Turing machine না মাজলে, মেধেনো বিষ্ট run কৰ্যার জন্য প্রয়োজন হিসেবে CPU টে run করে দেখা লাগে।

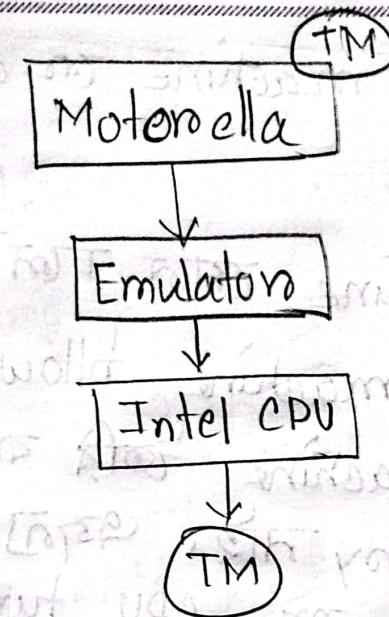
\* Emulators machine code সেক্ষেত্রে equivalent intell CPU-র জন্য code সেথে।

\* একটা TM আজক্ষণ্য সাথে compatible।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: \_\_\_\_\_

Date: / /



\*Turing machine ରେ ଚାଲାନେ ତା କେବୁ ଅଛି  
ଚାଲାନେ CPU କେତେ ଚାଲାନେ ଥାଏନା,

\*CPU ର ଅଟେକ୍ସର ବାଟୁ best building block  
and gate, or gate.

Transistor  
Switching

Amplify

Analog Electronics:

transistor

switch  
computer / IC (use)

Amplify  
communication (use)

Sub: \_\_\_\_\_

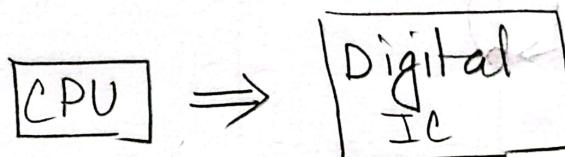
Sat Sun Mon Tue Wed Thu Fri

Date: / /

CPU বানালো স্মিথ CE র কথা না,

- IC: → 1. Analog IC (555 timer)  
2. Digital IC (CPU)

অবচেম্পে sophisticated IC → CPU → millions of transistors



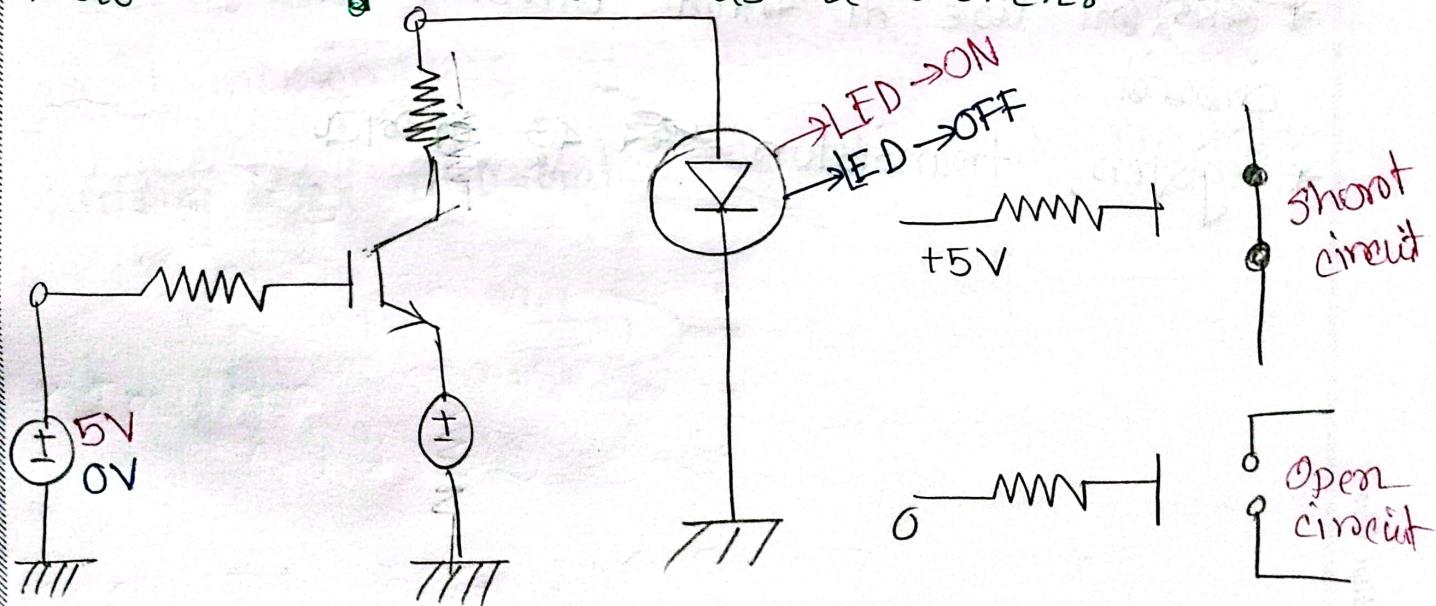
প্রতিটি chip design করা  
CE এর কথা

90% of the chip → Digital IC

10% " " " → Analog IC + Mixed

Transistor কীভাবে switch হিসেবে কাজ করে  
switch এর ২টা stage: ① ON  
② OFF

How transistor work as a switch?



\* Conditional switching.

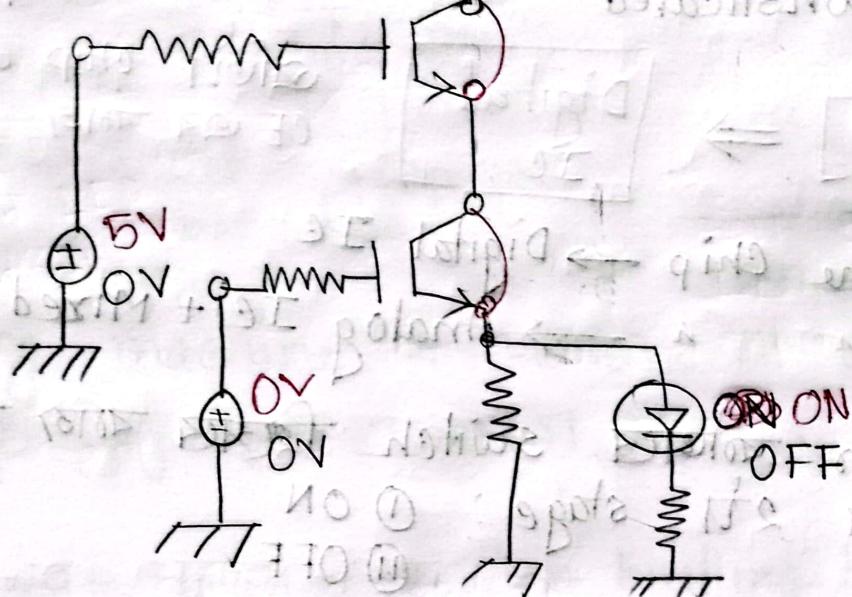
✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

Condition দিয়ে circuit on বা off করা যাবে।

AND Gate:



BJT Transistor.

\* অধিকৃতে we না করলে circuit হিটে যাবে।  
ক্ষেত্রে

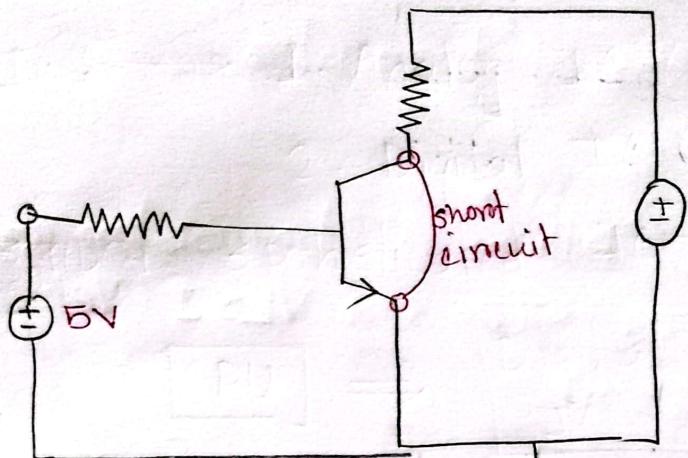
\* Registers, transistors এবং কৃত ক্ষেত্রে

Sub: \_\_\_\_\_

✓ Sat Sun Mon Tue Wed Thu Fri

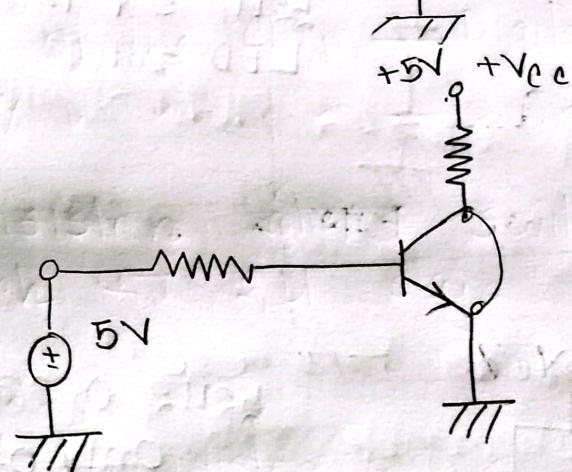
Date: 27 / 09 / 2023

IC



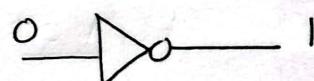
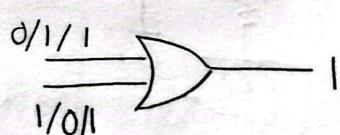
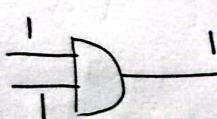
৫V দিলে বর্ণনা পূর্ণ  
হবে কোলো-

০V দিলে open circuit  
হবে



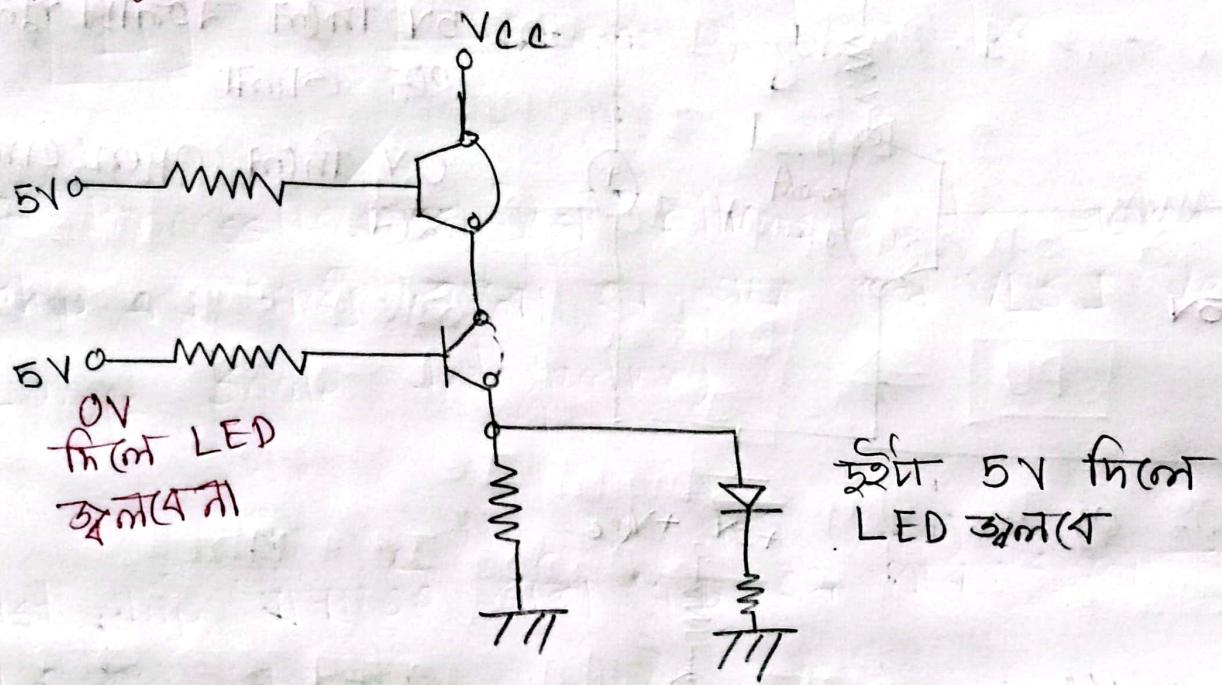
এই transistor use করে building block বানাবে হবে ,

Building block: AND Gate, OR Gate and NOT gate

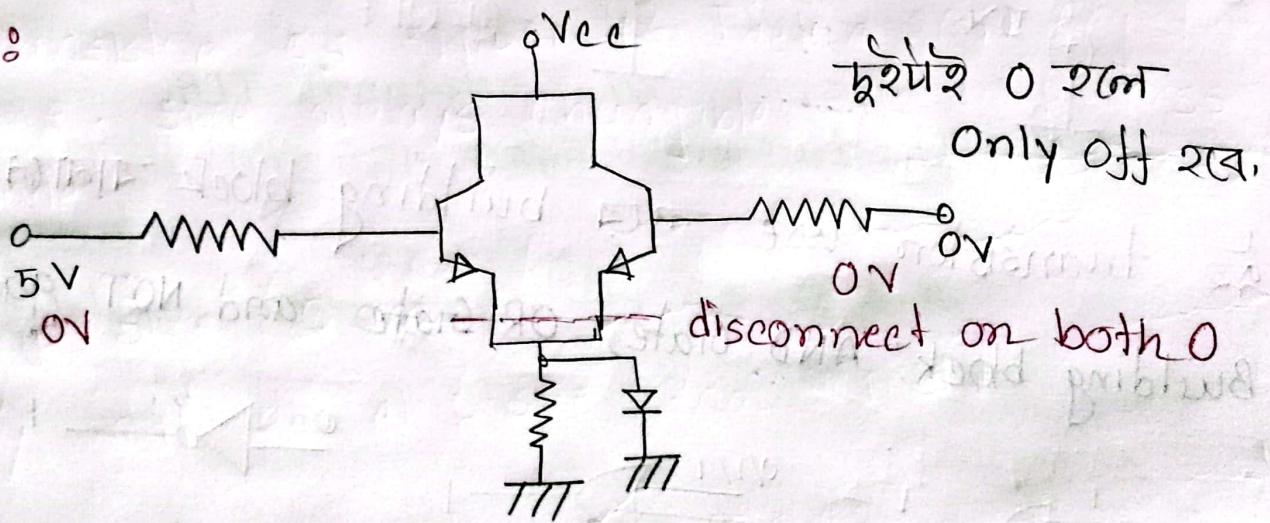


*Sub:* \_\_\_\_\_

~~AND~~ °  
°



OR:

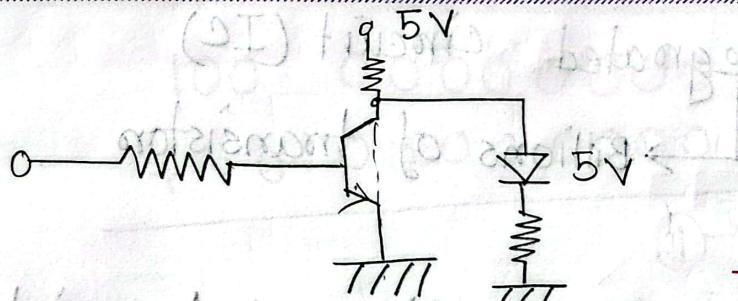


Sat  Sun  Mon  Tue  Wed  Thu  Fri

Date: / /

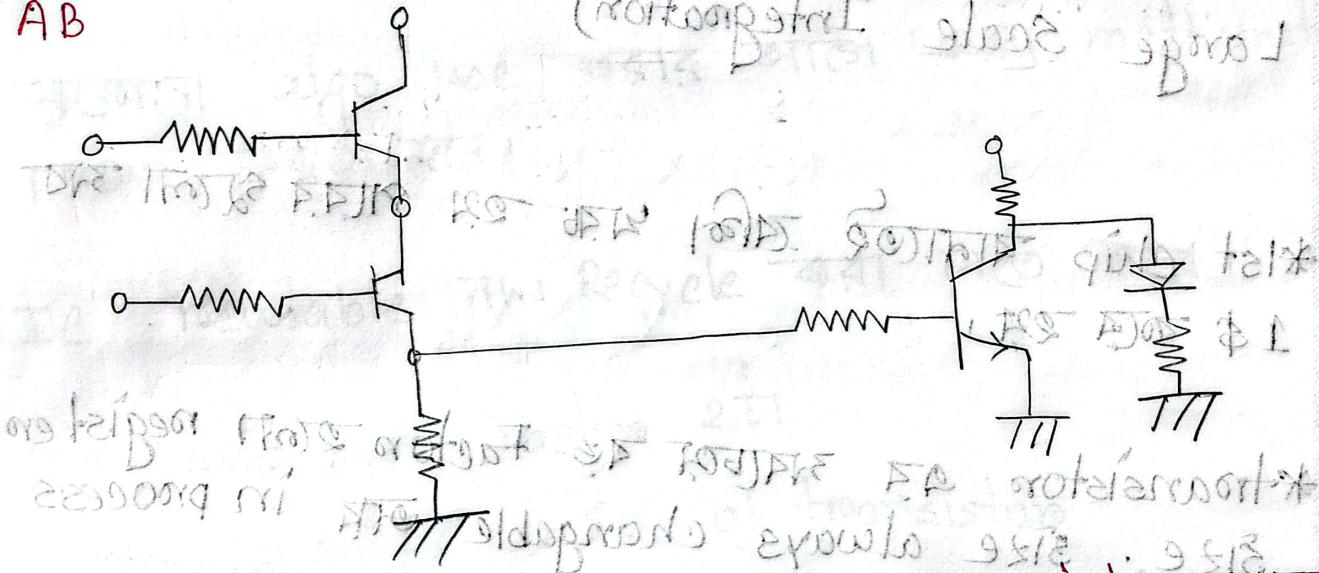
Sub:

NOT:



5V দিলে তা  
ground এর মাঝে  
connect রে, ground  
যেখানে powerful তারি-  
ground এতে সাধে

AB

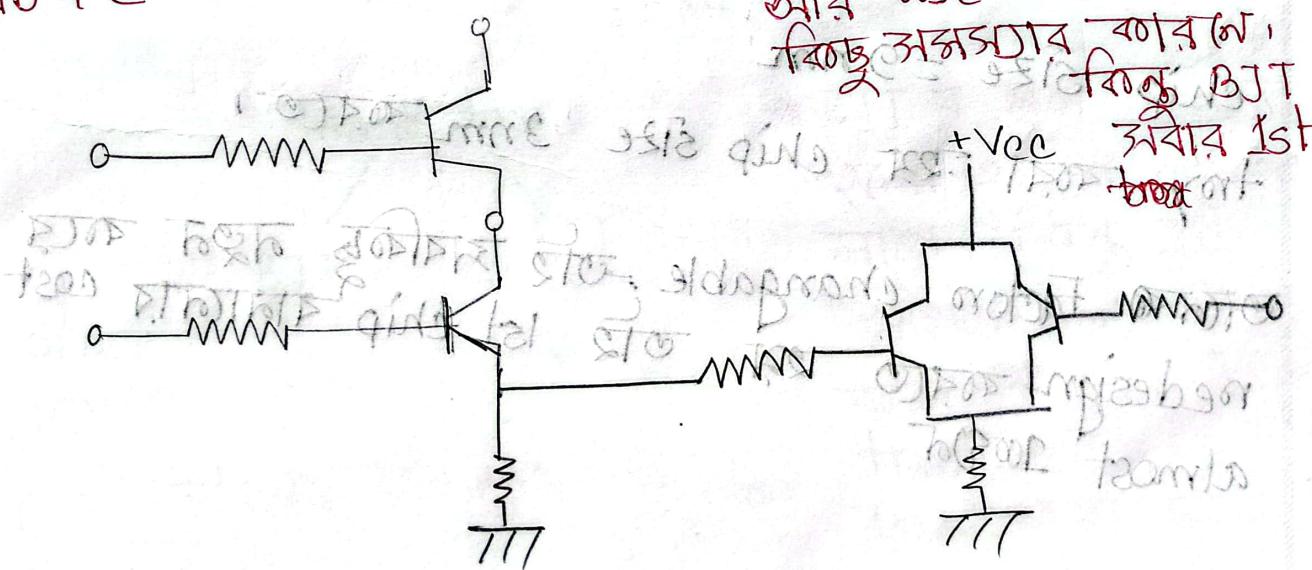


BJT Transistor এখন  
আর উব্বে করা হচ্ছে।

বিষ্টু অঙ্গসংকার করার জন্য।

বিষ্টু BJT মুকাবলী।

AB + C

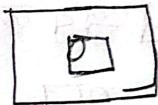


Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub: \_\_\_\_\_

Date: / /

## Integrated circuit (IC)



→ Billions of transistors

2<sup>nd</sup> Process এর মাধ্যমে billions of transistors কে chip এর মধ্যে দুরণ্তরো হয় আবে বলে VLSI (Very Large Scale Integration)

\* 1st chip বানাতেই অক্ষি খরচ হয় পারের ছিলো সব 1\$ কর্তৃ হয়।

\* transistors এর অবচেতন বড় Factors হলো registers size. size always changeable অব in process mode.

$$\text{chip size} = 9 \text{ nm}$$

troy বর্ষা হয় chip size 3nm করতে।

অনেক factors changeable তাই অবশিষ্ট নহুন করে redesign করতে হয় তাই 1st chip বানানোর cost almost 100\$।

Sub: \_\_\_\_\_

Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○

Date: / /

$$\begin{array}{r}
 100.00000000 \\
 + 10.00000000 \\
 \hline
 \end{array}$$

①  $\rightarrow$  দুটি অংশে।

দুটি অংশে দুরোহ calculation এ দ্বারা IC design এ দুটি হয়, এর জন্য ক্ষতিপূরণ নিয়ে হবে।

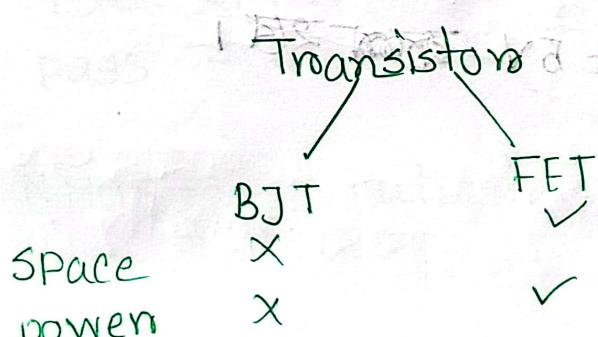
\* যেভাবে algo use করার আগে একে mathematically prove করতে হবে।

\* IC reusable না, Recycle করা মতে পারে।

IC design এর Factors ২টি

(a) Space  $\rightarrow$  size of transistors

(b) Power  $\rightarrow$  কী পরিমাণ power consume  
করতে।



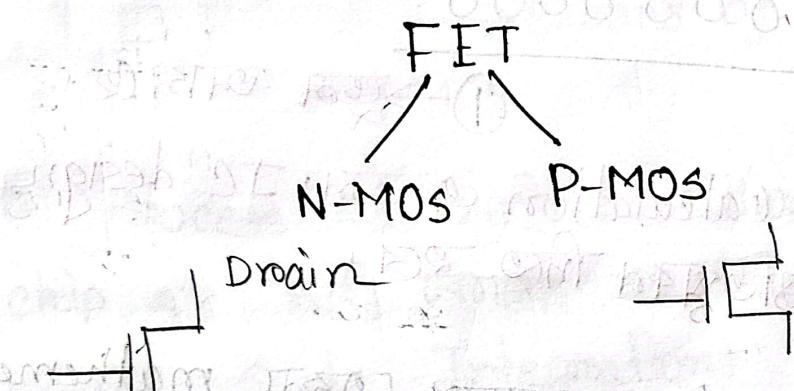
IC র সবচেয়ে বড় ক্ষেত্র  $\rightarrow$  register

\* BJT র দ্রুত অবস্থা হলো এখানে register লাগে, কিন্তু IC register নিয়ে টেক্সি করা যাবেনা, একে FET transistor এ FET use করা হয়।

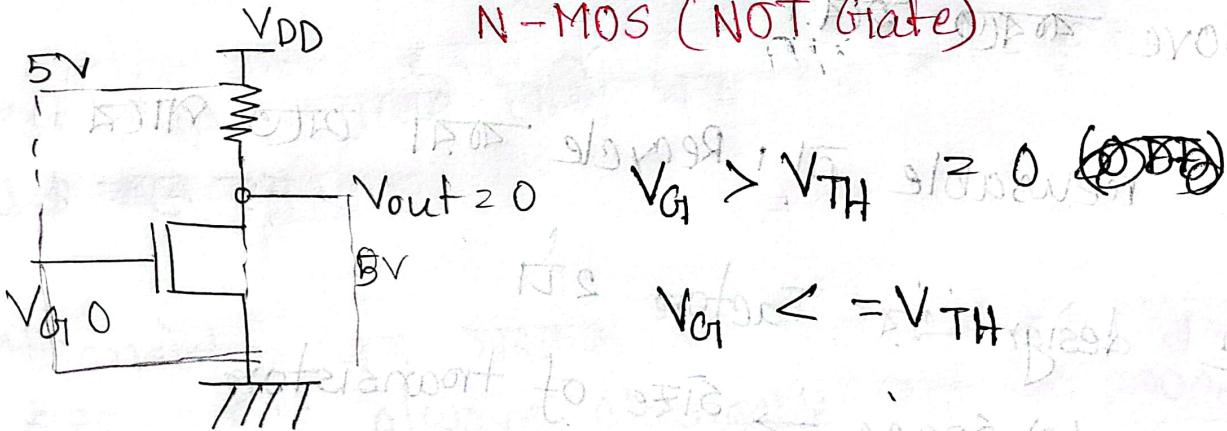
Sat    Sun    Mon    Tue    Wed    Thu    Fri  
 Date: / /

Sub:

FET to registers লাভে না, ০০১



N-MOS (NOT Gate)



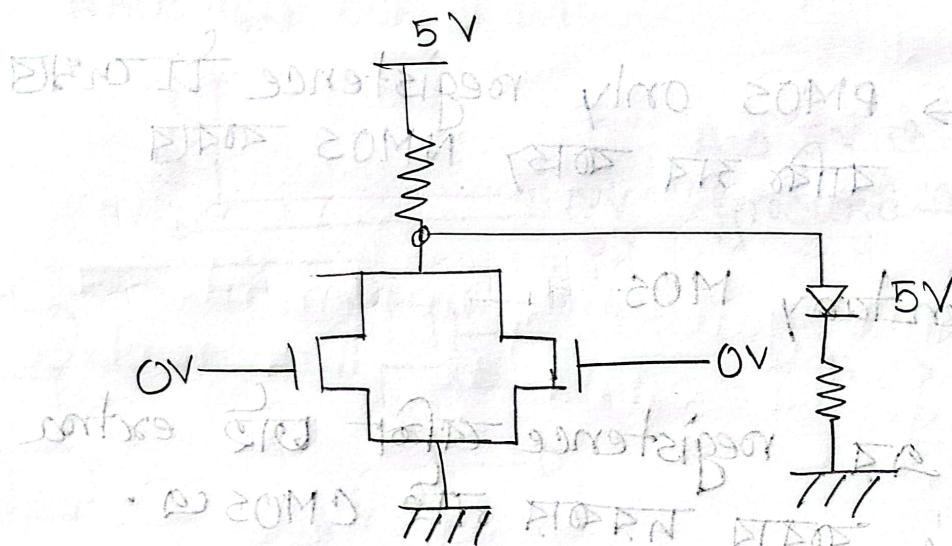
\*  $V_{G1} \rightarrow 5V$  হলে  $V_{out} = 0V$  ২৫

\*  $V_{G1} = 0V$  হলে  $V_{out} = 5V$  ২৫

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /



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

\* Only ~~কোনো~~ ON ফিল্টের output এ 5V পাবো, তাই  
~~কোনো~~ NAND gate এর property follow কৰবো।

\* NAND বা NOR gate use কৰতে transistor create  
 কৰবো হয়, FET ~~কৰবো~~ property follow কৰবো। AND/OR  
 Gate বানাতে NAND / NOR কৰে NOT gate কৰতে  
 pass কৰবাবো হয়।

NMOS → Pull down Network (Only ground)

NAND / NOR → Universal gate

\* BJT কৰতে AND/OR gate বানাবো যথে easy.

Sub:

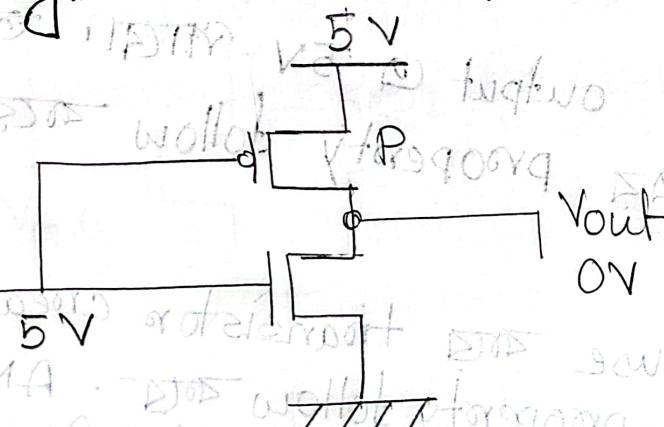
✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

\* PMOS এর registerance N-MOS এর টেক্স  
বেশি।

CMOS  $\rightarrow$  PMOS only registerance দিচ্ছে  
বাকি অব বাই N-MOS ক্ষয়তি  
↓  
Complementary MOS

As PMOS এর registerance বেশি তাই extra  
register use করার দরকার নাই CMOS এ।



\* NO power ক্ষয়তি  
উপরের short circuit  
হবে নিচের di open  
circuit

\* 5V use করলে Vout  
0V

\* NMOS, P-MOS এর টেক্স বেশি power consume  
করবে।

\* CMOS circuit এর separate করে আবার একে  
register ও নাই, তাই use করা mostly.

Sub: \_\_\_\_\_

Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○

Date: 02 / 10 / 2023

**A.B + C.D****NMOS**

5V

Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

How to design

100 millions

200 কোটি

transistor

1st IC design বর্ততে হবে যখন  
chip-এ 100 million transistors

IC  
↑  
VLSI

270 nm

transistor

পাইকা,

IC টো NAND/NOR

use করিব, কারণ NAND/NOR  
convert করা easy.  
পুরো NAND/NOR টো transistor  
দিয়ে replace.

(F) 8086 এর transistor → 17000-27000

\* IC-কে NAND gate দিয়ে implement করা easy,  
NAND gate কে আবার transistor দিয়ে replace  
করা যায়।

\* circuit বানানোর জন্য ডিম্পলগুলি fixed. তাহি এমনভাবে  
process বর্ততে হবে যেন তা size এ জায়গা হয়।

HDL → Hardware Descriptive Language

High level এর circuit লাগাও  
তা details এ descriptive করলে সেওয়ে  
circuit টা করে দিব।

assign Y = A^n B;

1st এ XOR বানাও  $\Rightarrow \text{D}\text{O}$

এবং NAND gate করো  $\Rightarrow \text{D}\text{O}$

Transistor দিয়ে replace করে  
এভাবে বানাই দিব :  $\begin{array}{c} \text{A} \\ \text{B} \\ \hline \end{array} \Rightarrow \text{D}\text{O}$

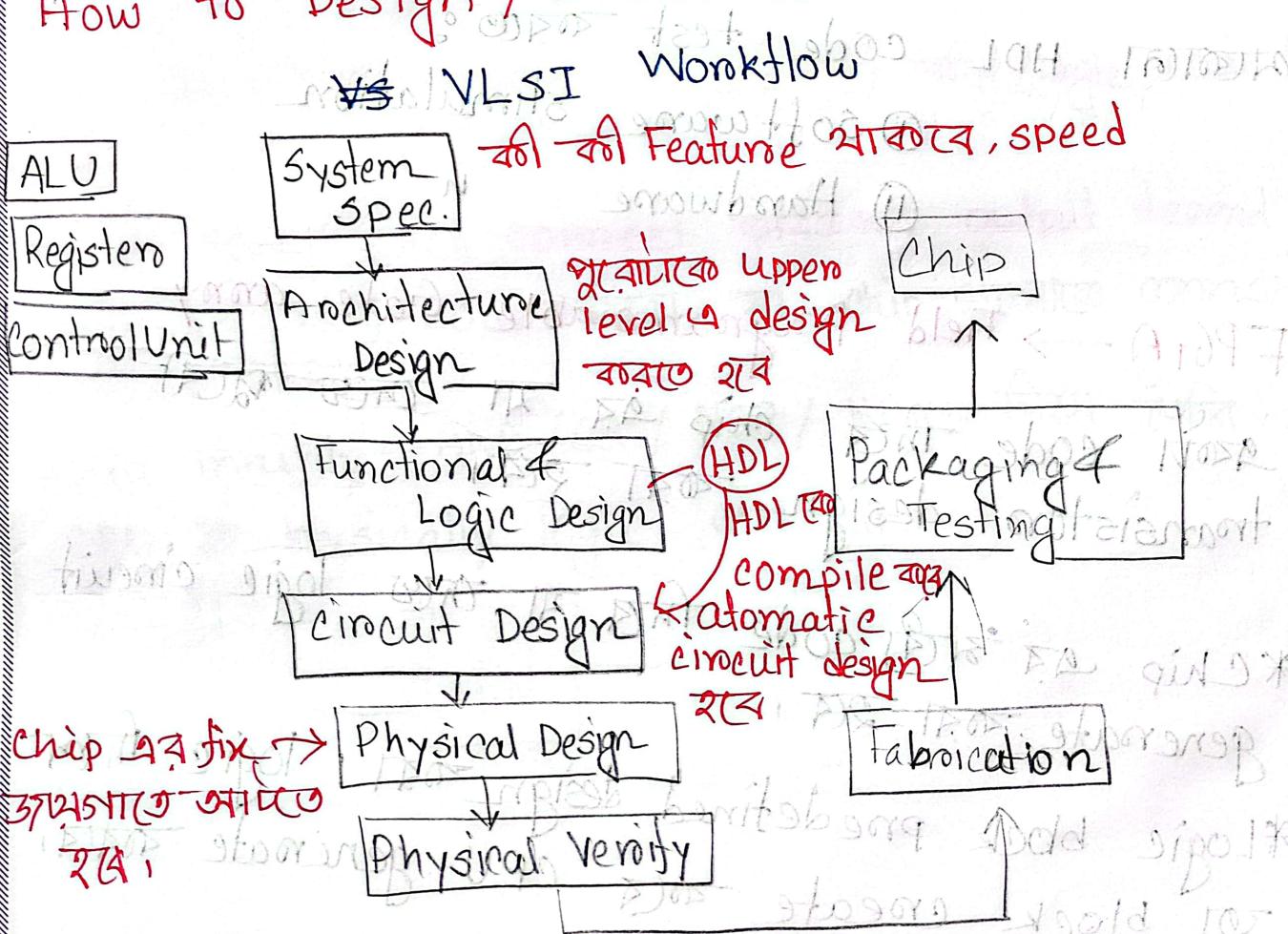
Sub: \_\_\_\_\_

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
-------------------------------------	-----	-----	-----	-----	-----	-----	-----

Date: / /

- \* HDL programming Language না, description দিয়ে circuit design করা হবে।
- \* বিভিন্ন Test case design করা হয়ে থাকে তা দিয়ে জানা যাবে circuit কিরণ করা work করতে পারে।
- \* Mathematically prove করে দেখবে।
- \* Chip তৈরি হলে তা এখন নিতেই test করবে prove করা must. না হলে তুলি হয়ে যাবে।

### How to Design / Create a chip:

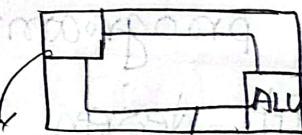


Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub: \_\_\_\_\_

Date: / /

## Physical Design:



CLK  $\rightarrow$  Routing  
WE  $\rightarrow$  Routing

করতে

(i) Partitioning

(ii) Planning

(iii) Placement

(iv) Routing  $\rightarrow$  তারগুলো কীভাবে Flow হবে

(v) Timing

মজানে HDL code test করতে :

① Software stimulation

② Hardware

## FPGA $\rightarrow$ Field Programmable Gate array

এখন code ফিলে chip এর মা দিয়ে প্রো  
transistor design করা হবে।

\* Chip এর জন্যে code ফিলে মা দিয়ে logic circuit  
generate করা হবে।

\* Logic block predefined design করা। logic block  
তা block create করে CPU generate করবে।

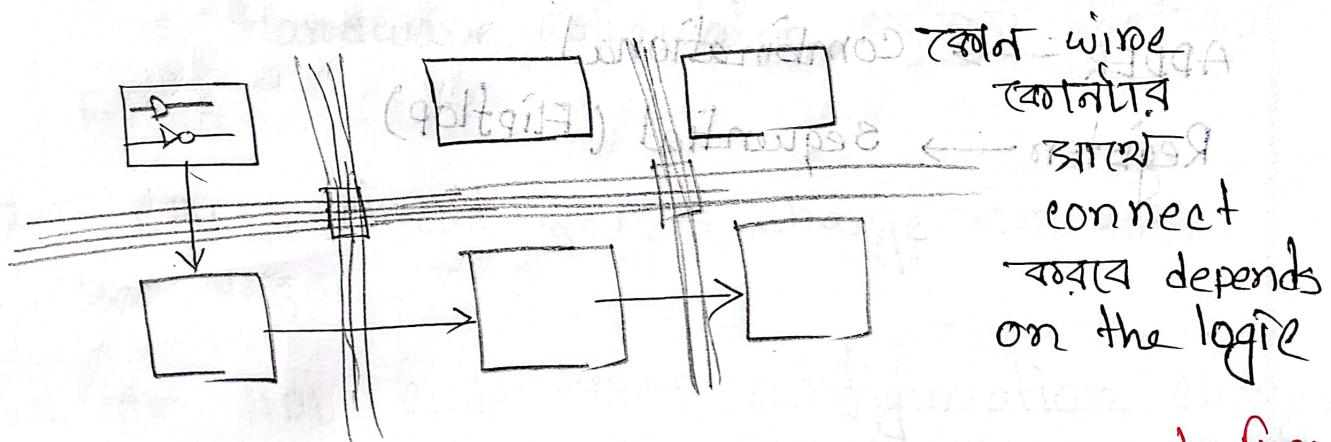
✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

\* CPU যেকোনো block আছে এ অথবা কোনো configuration কোরা থাকে only logic input  
ব্যবহারে block connect হয়ে থাকে।

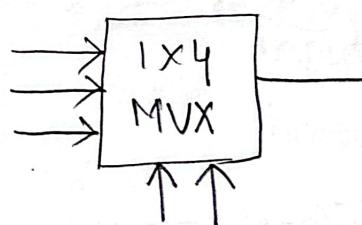
\* block দ্বিতীয়ে flip flop logic নিলে block দ্বিতীয়ে  
resistor create করবে



1st এ অবগুলো block disconnect করে logic code নিয়ে  
২nd connect হবে।

wire দ্বালো কীভাবে connect তা রেখাগুরুত্বে output depend করতে  
chip এর মধ্যে code নিয়ে, যা chip দ্বালোকে connect করে  
circuit বানাবে।

chip mainly AND / OR / NOR / NAND নিয়ে টার্ভি,  
CPU কে basically logic circuit



✓  Sat  Sun  Mon  Tue  Wed  Thu  Fri

*Sub:-*

Date :      /      /

Circuit 2 types

① Sequential circuit  $\rightarrow$  Flipflop দিয়ে  
Save করে বাধ্যতে  
পারবো

১১) Combinational  $\rightarrow$  logic ফিল্টের  
পারবে,

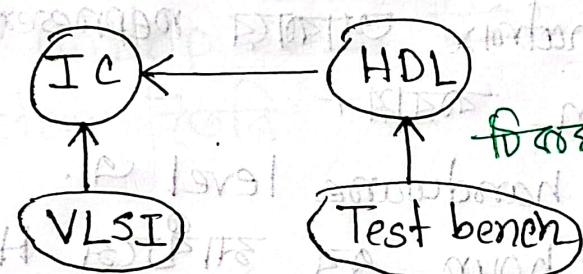
ADDER → Combinational

Registers → sequential (Flipflop)

✓ Sat Sun Mon Tue Wed Thu Fri

Date: 03 / 10 / 2023

Sub:

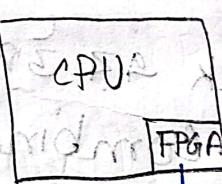


HDL code hardware simulation এর জন্য we ব্যবহার করি।  
যথে FPGAs.

চাইলে CPU র circuit পুরোপুরি change করা যাবে,  
HDL use করে,

FPGA তে HDL code ফল configuration change  
করা যাবে।

\*FPGA অনেক expensive তাই এটা Feasible না। FPGA  
software ও hardware both update করে, তাই-  
মুখ্য জনপ্রিয় হয়ে উঠে।



থাকতে,  
input ফলে আরামাদি (update) ফলে।

hardware তাই Faster হবে।

\*FPGA তে এখন sophisticated company শৈলো  
নুন করায়।

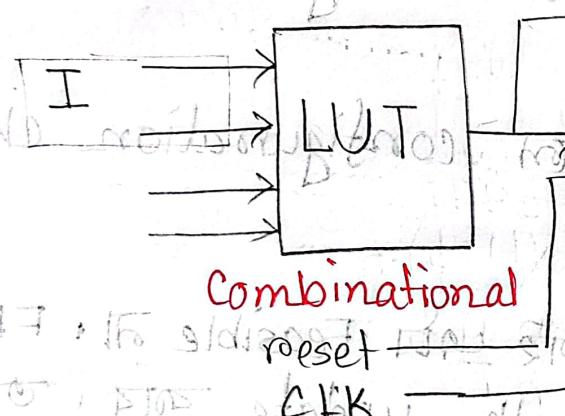
✓ Sat Sun Mon Tue Wed Thu Fri

Sub: \_\_\_\_\_  
Date: / /

\* GPU কোনো data matrix আবশ্যক রূপে  
হলে অনেক বারে run করাম।

\* compression হয় hardware level ৭.

এজন 1GB video 1 hour এর জার্জে HD  
কোড ফিল্টে, এইটি compression এর জার্জে  
circuit এবং সম্পর্ক data ফিল্টে, FPGA র  
তন্ত্র এই video upload 1000 মিলি seconds



Combinational

Sequential

computational or sequential বানানো যাবে।

\* computational circuit CLK এর উপর depend করেনা,  
instant result ফিল্টে, Combinational logic

অনেক fast and calculation করে।

sequential Memory (1)  
circuit (0)

✓ Sat Sun Mon Tue Wed Thu Fri

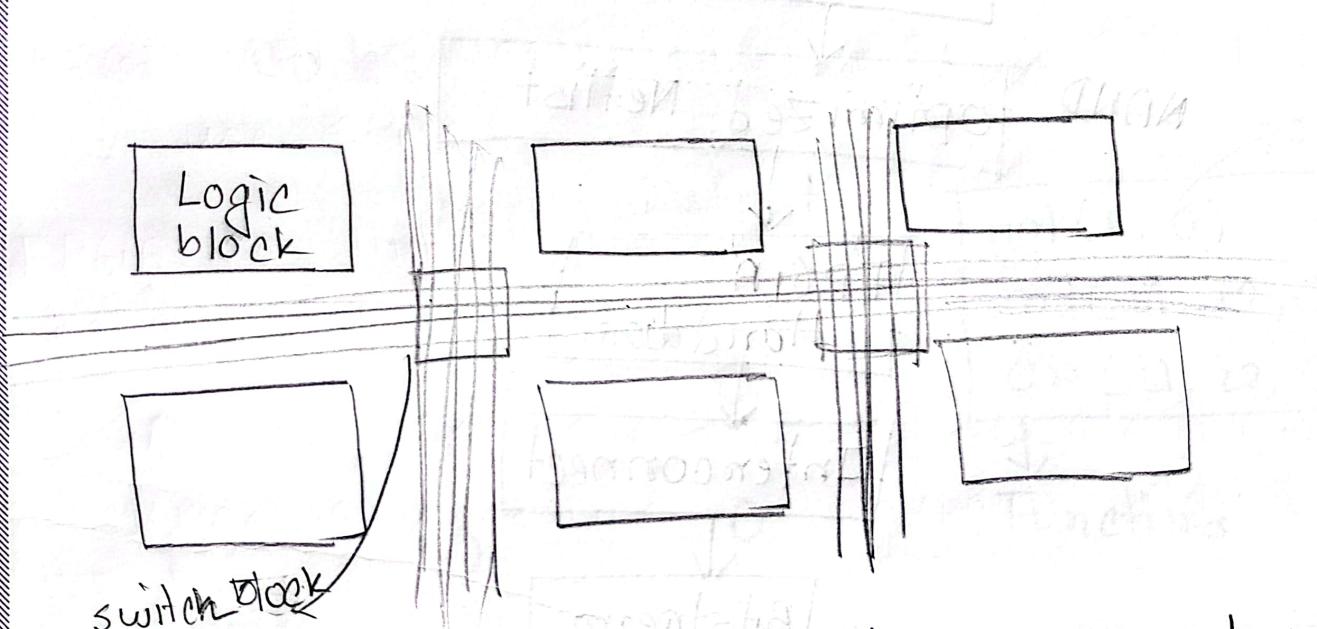
Sub:

Date: / /

3DI Flipflop আজ ফিল্ট কোডের 3 bit  
registers টেবি রয়।

LUT → Look Up table → computational circuit  
implement কৰা।

Flipflop → sequential circuit.



HDL code অনুমানে logic block connect কৰে।

switch block এলো connect কৰে তাৰে connection

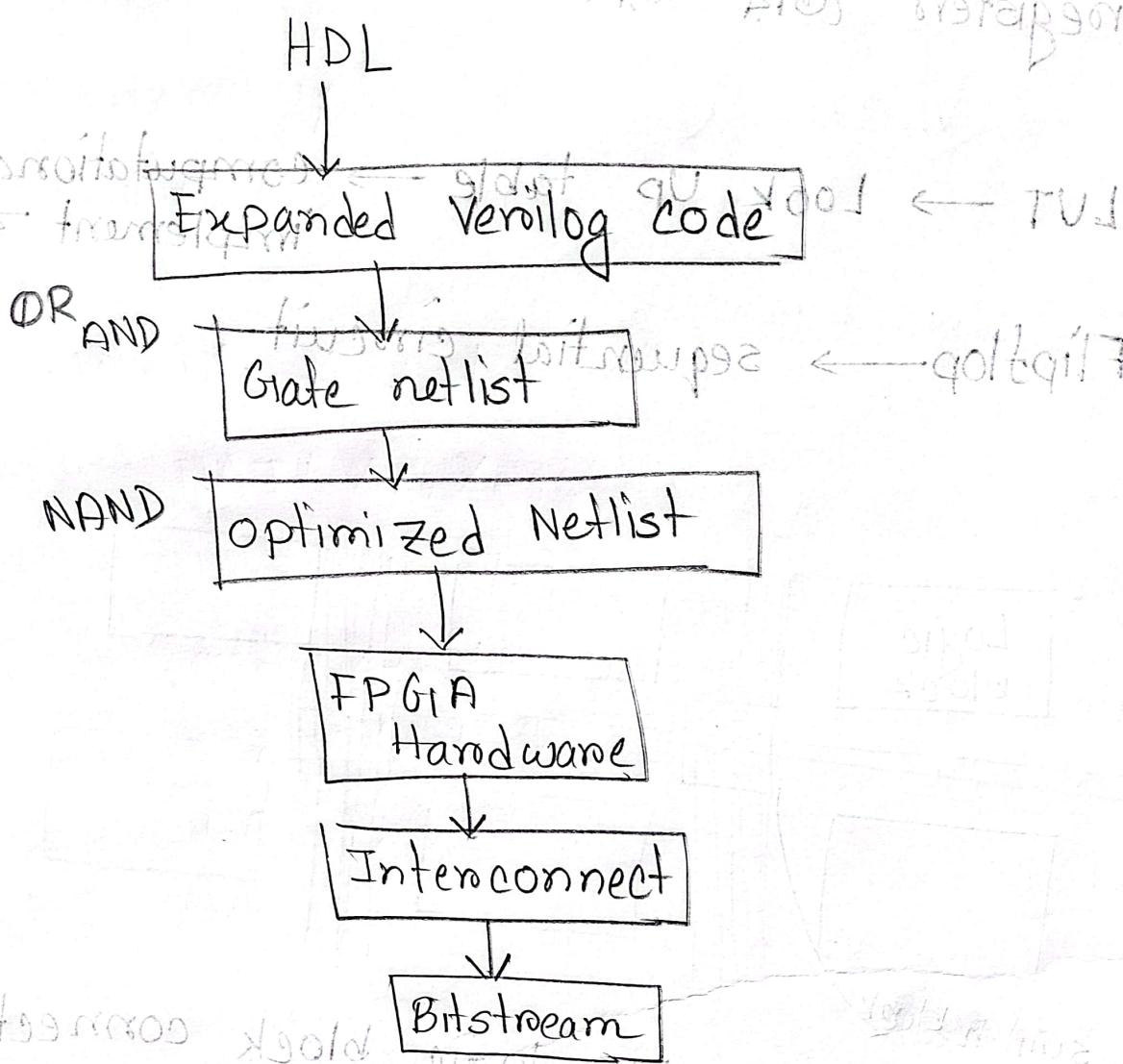
switch block এলো connect brodes কৰে।

switch কৰে, ৩DI CPU টে ফিল্ট অনুকৰণ কৰে।

আৱার cost ও আছে।

FPGA → costly & slow

# Verilog/VHDL



HDL কে expand করা কোন A+B, adder circuit  
এ replace করলো, এবাব upper level এর code  
গুলোকে gate এ convert করা, কোন gate কে  
NAND/NOR এ convert, এবাব check করবে  
কতোগুলো sequential memory and কতোগুলো  
computational ইয়ে তা FPGA board এ দিব।

✓ Sat Sun Mon Tue Wed Thu Fri

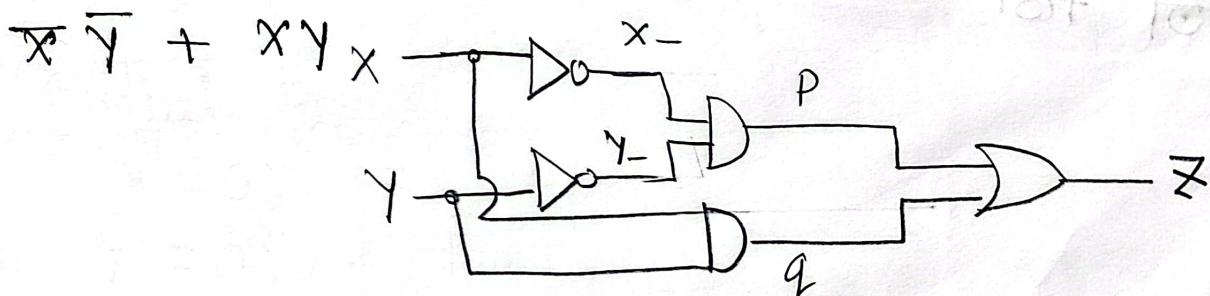
Sub: \_\_\_\_\_

Date: / /

FPGA hardware এ configure করে আগে যেকোন যোগ্য block আছে। FPGA block interconnect করবে। পারে তা bitstream এ convert করবে। bitstream chip configure করার code. bitstream প্রত্যেক FPGA র জন্য সন্তোষিত হয় depends on each chip. যা পারে FPGA code generate করবে।

HDL code:

X	Y	Output	Functions
0	0	$(P, X, X)$	$\text{not}(i, 0)$
0	1	$(P, X, X)$	$\text{and}(i_1, i_2, 0)$
1	0	$(P, X, X)$	$\text{or}(i_1, i_2, 0)$



<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Sub:

Date: / /

## Module Comparators

input X,

input Y,

Output Z

wire wire X-, Y-, P, Q

NOT(X-, X-);

NOT(Y-, Y-);

and (X-, Y-, P);

and (X, Y, Q);

OR (P, Q, Z);

end module

circuit test করাতে আবশ্যিক code লিখতে হবে  
 at test bench.

Sub:-

✓ Sat	Sun	Mon	Tue	Wed	Thu	Fri
-------	-----	-----	-----	-----	-----	-----

Date: / /

## Test bench:

timescale 1ns/1ps → মৰনি কোণা  
 module stimulus(); → Test bench এ  
 input/output থানা,

data store reg x;  
 কোরতো reg y;

computational circuit

data saved কোরতো পাবেনা

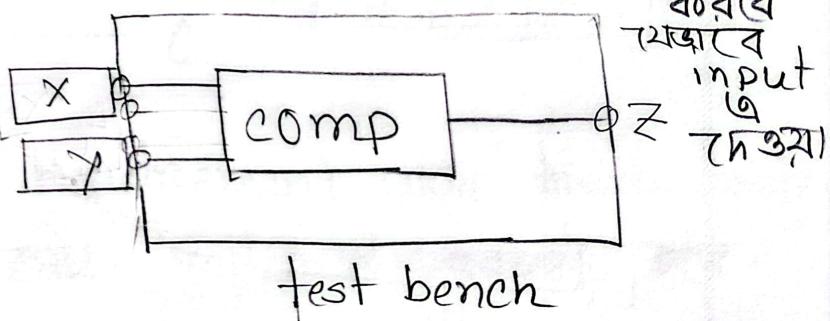
wire z; → save  
 ব্যবহার দৰকার  
 নাহি output  
 তাৰে wire

তাৰে register use কোৱা  
 হয়, registers টা থাবলো  
 উচিত পাবলো হতা

comparator unit( .x(x), .y(y), .z(z));

unit under|input  
test(x, y, z) connect  
serially connect

\* test bench কিম  
 কোৱতো কোৱা input  
 input কিমে, এস  
 upper level এর  
 circuit



initial begin → New block এখান input দেওয়া ছী

 $x = 0;$  $y = 0;$ #20;  $x \geq 1$ ; → 20ns এৰ পৰি  $x \geq 1$  কোৱা,  $y = 0$ #20  $y = 1$ ; →  $y = 1$ ,  $x = 1$

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

# 20  $x = 0;$

X Y

0 0  
|  
0

# 40;

end

0 1

initial begin  $\rightarrow$  result

0 0

\$monitors ("x=%d, y=%d, z=%d\n",  
x, y, z);

circuit monitor

বাটৰ x, y, z এৰ value

end.

end module

Sat    Sun    Mon    Tue    Wed    Thu    Fri  
 Date: 04/10/2023

Sub:

## HDL - 1 combinational circuit

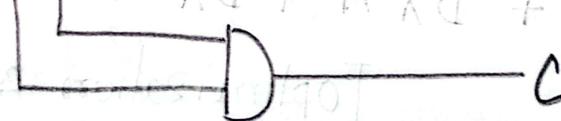
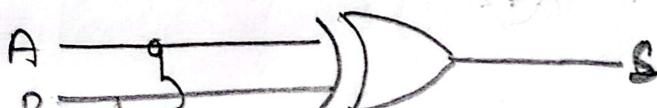
### Half Adder :

~~A + B~~

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

$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$C = AB$$



\* Truth table must requirement তার ট্রুথ টেবিল  
বানাতে হবে, তারপর সেটা implement করলেও হবে।

Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub: \_\_\_\_\_

Date: / /

D	X	A	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$L = D + A$$

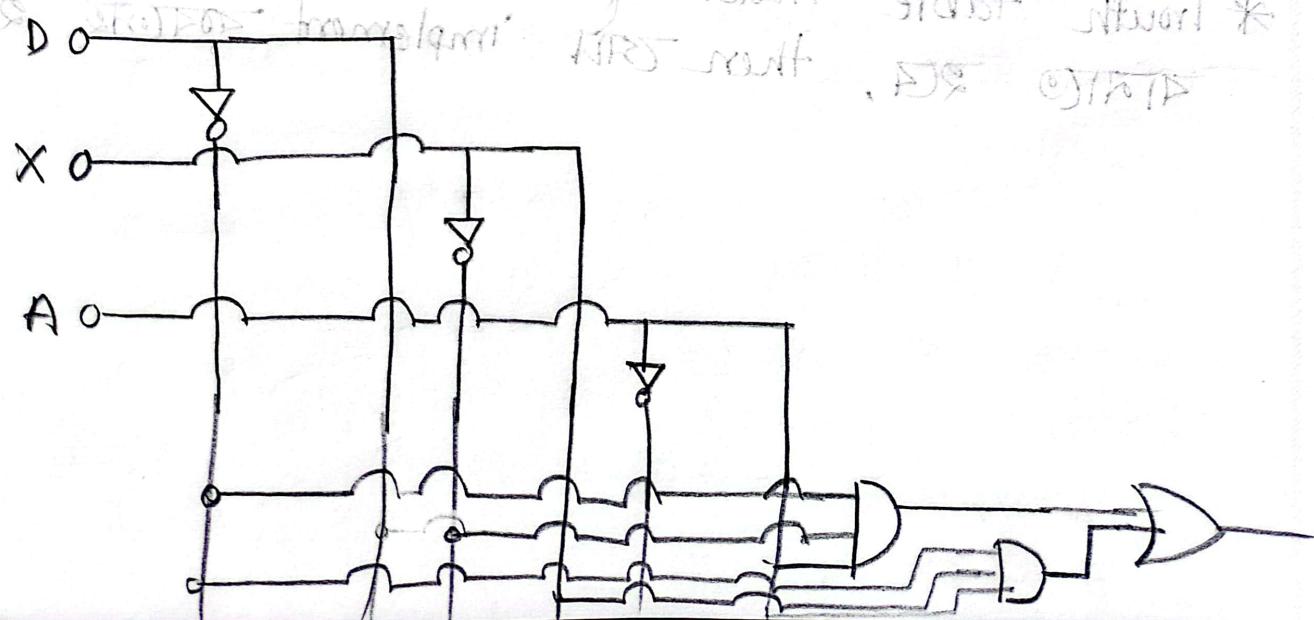
Logic diagram

$$\bar{D} \bar{X} A + D \bar{X} A$$

$$= \bar{D} \bar{X} A + D \bar{X} A$$

$$L = \bar{D} \bar{X} A + \bar{D} X \bar{A} + D \bar{X} \bar{A} + D X \bar{A} + D \bar{X} A + D X A$$

[Optimization এর সময়  
নাই]



✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

module comb1

```
( input D,
  input X,
  input A,
  input Z,
  output L
```

);

assign L = ( $\sim D$ )  $\wedge$  ( $\sim X$ )  $\wedge$  A | ( $\sim D$ )  $\wedge$  (X)  $\wedge$  ( $\sim A$ )
 $| (D) \wedge (\sim X) \wedge (\sim A) | (D) \wedge (\sim X) \wedge (A)$

~~| (D)  $\wedge$  ( $\sim X$ )  $\wedge$  (A) | D  $\wedge$  X  $\wedge$  A;~~

~~DECIMAL (A) A. (X) X. (D) D.~~

end module

~~Test~~ \* And, or, not operation বার বার না লিখে

~~Direct~~ অত্যবে \* symbolic অথে লেখা মাত্র

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

Test bench: → এর output

timescale 1ns/1ps

module comb1\_tb;

reg D;

reg X;

(A) (reg A);

(A) (wire L);

comb1 (A)

unit(D, X, A, L); → এটা ক্ষেত্রে  
নাম দিত হবে.

D(D), X(X), A(A),

L(L) → এটা সিধুল

sequence এর  
সমস্যা নাই

initial begin

D = 0;

X = 0;

A = 0;

#20;

A = 1; → 001

#20;

X = 1; → 010

#20;

D = 1 → 111

\* চাইলে মোলান্ট

করে অব combination  
run করে করে

চালি মাত্রে।

sequentially

এটা করলে sequence  
maintain করবেন।

Sub:

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Date: / /

# 20;

A = 0; → 110

# 20;

X = 0; → 100

end

initial begin

\$monitor function

\$monitor ("D=%d, X=%d, A=%d, L=%d\n",  
D, X, A, L);

end

end module

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

## Vernor Operations

Multi bit এর তাত্ত্বিক

input [3:0] A,

module comb2

(

input [3:0] A,

input [3:0] B,

output [3:0] out

3 bit যোগ

assign out = A & B;

end module

Test bench:

timescale 1ns/1ps

module comb2\_tb;

reg [3:0] A;

reg [3:0] B;

wire [3:0] out;

Sub: \_\_\_\_\_

✓	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Date: / /

comb2 unit (A, B, Out)

initial begin

$$A = 4'b0000; \rightarrow [3:0]$$

4 bit binary (0, 1, 2, 3)

$$B = 4'b0000;$$

এতোবেগে লেখা মাত্র,

$$A = 4'b1100; \rightarrow \text{এসর decimal } 12.$$

$$B = 4'b0110;$$

লেখা মাত্র 12.

$$\#20;$$

end

$$\begin{array}{r}
 & 1100 \\
 & 0110 \\
 \hline
 & 1010
 \end{array}$$

overflow এসে বাদ  
মাদে,5th bit এসে হচ্ছে।  
তাই হারিয়ে যাও, but  
এতোবেগে code করো  
যাবে যেন 5th bit না

হারায়

initial begin

\$monitor( . . . )

end

endmodule

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
	○	○	○	○	○	○	○

Date: / /

Sub: \_\_\_\_\_

# VHDL operators

## Arithmetic

$+$  → adder circuit

\*

$\downarrow$  0000 dip

shift

Arithmetic Left shift

$>>$ ,  $<<$ ,  $<<<$ ,  $>>>$

Arithmetic right shift

## Relational

$>$ ,  $<$ ,  $>=$ ,  $<=$

→ A < B compare

বরবর

## Equality

$=$ ,  $!=$

case wise bit by bit check

## Bit wise

$\&$ ,  $1$ ,  $\sim$

bit by bit  
ফরম  
চেক

## Logical

$ff$ ,  $11$ ,  $!$

<u>X</u>	<u>Y</u>	<u>Z</u>
0	0	0
1	1	1

module comp

( input wire X,  
input wire Y,  
input)

$$Z = \overline{X} \overline{Y} + XY$$

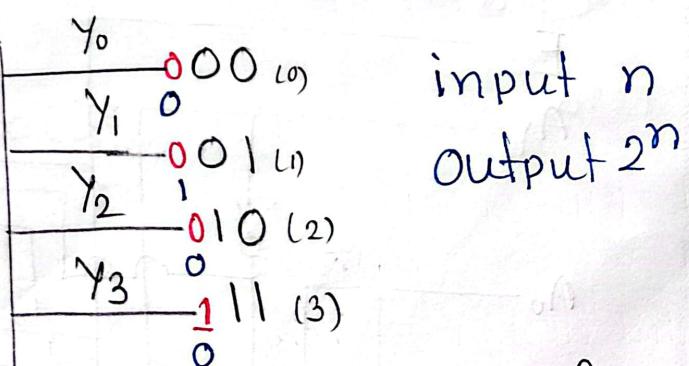
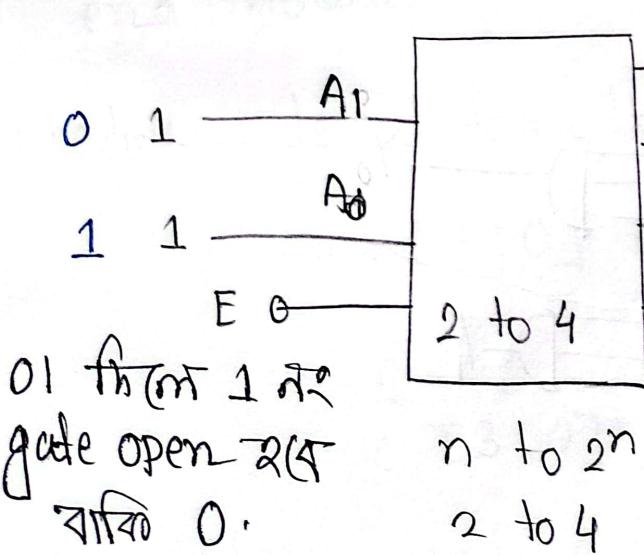
iverilog → install in VS code → From slide.

HDL → Very Very  
important  
18 marks in SF



assign anith = A + C; → addero.

## Combinational circuit:



11 input from 3 NO  
gate open হবে (1)  
বাবি অব 0

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: 8200101

Date: / /

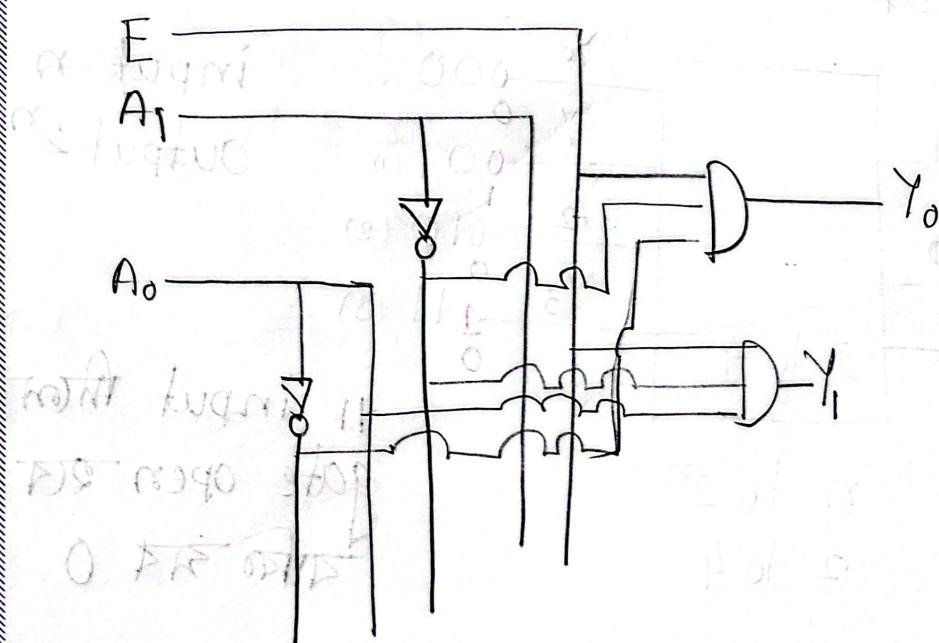
E	$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$	X
1	0	0	0	0	0	1	0
1	0	1	0	0	1	0	1
1	1	0	0	1	0	0	1
1	1	1	1	0	0	0	$X + \bar{Y} = S$

$$Y_0 = E \cdot \overline{A_1} \cdot \overline{A_0}$$

$$Y_1 = E \cdot \overline{A_1} \cdot A_0$$

$$Y_2 = E \cdot A_1 \cdot \overline{A_0} \quad 0 + 1 = 1$$

$$Y_3 = E \cdot A_1 \cdot A_0$$



Sub:

✓	Sat	Sun	Mon	Tue	Wed	Thu	Fri
	○	○	○	○	○	○	○

Date: / /



module decod2to4

C input [1:0] A → Array

input E,

Output [3:0] Y

;

assign Y[0] = E &amp; (~A[1]) &amp; (~A[0]);

assign Y[1] = E &amp; (~A[1]) &amp; (A[0]);

assign Y[2] = E &amp; (A[1]) &amp; (~A[0]);

assign Y[3] = E &amp; (A[1]) &amp; (A[0]);

end module

module decod2to4\_tb();

reg [1:0] A;

reg E;

wire [3:0] Y;

decod2to4 uut(A, E, Y);

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: \_\_\_\_\_

Date: / /

initial begin

A[1] = 1'b0;

$A = 2'b00; \rightarrow A_0, A_1$  হইত্বে ক্ষয়গ্রস্ত

#20

A = 2'b01; E = 1'b1;

#20

(EJA)  $\rightarrow$  (EJA) এর জৰা combination

$A = 2'b10;$  মাঝতে ইম্বে

#20

(EJA)  $\rightarrow$  (EJA)  $\Rightarrow E = 1'b0;$  নথিপেস

$A = 2'b11; E = 01'b0;$  নথিপেস

(EJA)  $\rightarrow$  (EJA)  $\Rightarrow E = 1'b1;$  নথিপেস

(EJA)  $\rightarrow$  (EJA)  $\Rightarrow E = [E];$  নথিপেস

end module

end

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

always @ ({sensitivity-list})

begin

end

if (E == 1'b0)

$y = 4'b0000;$

else if (E == 1'b1)

else if (A == 2'b00)

$y = 4'b0001;$

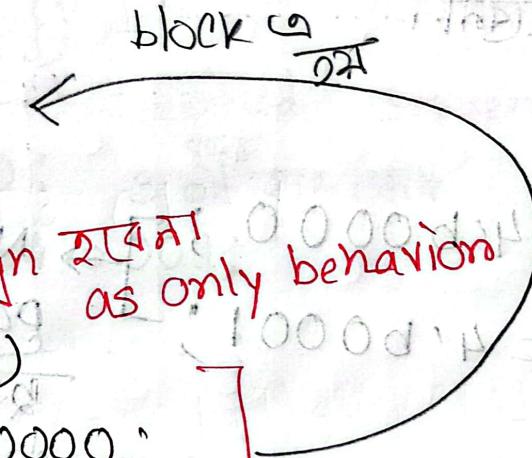
else if (....)

Var = exp  $\rightarrow$  blocking assignment

Var <= exp  $\rightarrow$  non blocking assignment

input এর জন্য

always @ (A, E) \* সিধেই all input কুকাবে,



→ মেরুদণ্ড  
সময় এবং  
use করা  
যাবে। আরো  
কানেক্ট  
লেভেল High  
লেভেল হবে।

→ এখানে only logic  
change করে ফেলে  
হবে নিচে নিচে  
circuit বানাবে তব  
combination use  
করে, যদিও প্রয়োগের  
decoder হবে।  
behavior থাকবে

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

blocking assignment জাতে sequentially

যত্থেক্ষণেই, একসাথে execute না করে আবশ্যিক

যাওয়া পথ:

$y <= 4'b0000;$  ] Non-blocking

$x <= 4'b0001;$  parallelly করবে

কোরি sequence  
doesn't matter

always @(\*)

begin

if ( $E_2 == 1'b0$ )

assign  $y = 4'b0000;$  non-blocking

কোরি  
করুন

else if ( $A == 2'b00$ )

$y <= 4'b0001;$

end

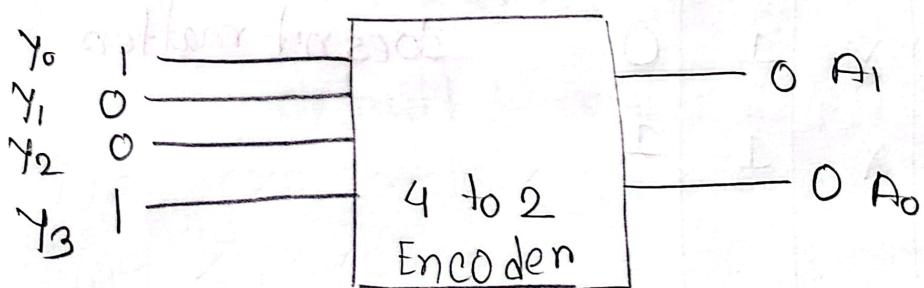
✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 10/10/2023

always @ \* block  
begin  
Case ({E,A})  
3'b100 : 3 bit রূপে CST  
 $Y = 4'b001;$   
3'b101 :  
3'b110 ;  
3'b111 ;  
default :  $Y = 4'b0000;$  End of if  
end

4 to 2 encoder



2<sup>n</sup> to n

encoder

Priority Encoder: priority ক্ষেত্রে হবে, 1001  
input & output কী আসবে তা  
জন্য

Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub: \_\_\_\_\_

Date: / /

$y_3$	$y_2$	$y_1$	$y_0$	$A_1$	$A_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0		
1	1	1	1	0	0

3R combination  
wise priority  
set বর্তমানে হবে,

$$A_1 = y_3 \bar{y}_2 + \bar{y}_1 + \bar{y}_0 + y_3 y_2 \bar{y}_1 \bar{y}_0$$

$y_3$	$y_2$	$y_1$	$y_0$	$A_1$	$A_0$
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

$x \rightarrow$  যা আবশ্যিক  
does not matter

১ প্রতিটো

বাকি যা

যাইকে ১। হবে।

সত্ত্বেও

$$A_1 = \bar{y}_3 y_2 + y_3$$

বাকি দুটো সহজ  
ইউ

Sub:

✓	Sat	Sun	Mon	Tue	Wed	Thu	Fri
	○	○	○	○	○	○	○

Date: / /

module pencode 4 to 2

C input [3:0] Y

output [1:0] A

);

always @(\*)

begin

case (Y)

4'b0001 : A = 2'b00;

4'b0001x : A = 2'b01;

4'b01xx : A = 2'b10;

4'b1xxx : A = 2'b11;

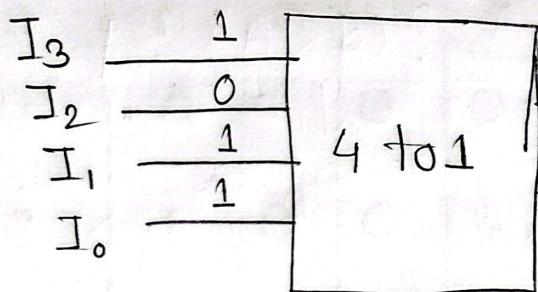
default : A = 2'b00;

end

end module.

## Multiplexers : (Most Imp)

$2^n$  to 1 MUX



4টি input থাকে কোনো নাগতি আৰু select কৰবে, যেন্তে select হবে depends on selection line.

$$\begin{array}{r} S_1 \quad S_0 \\ \hline 1 & 0 \\ \hline 2 & \end{array}$$

$\rightarrow I_2$  select হবে,  $I_2$  এর  $Y$  এর value পাবো।

$S_1 = 0$ ;  $S_2 = 0$  হলে  $I_0$  Line select হবে।

input  $S_1, S_0 \rightarrow$  Output  $Y$

6টি input দিলে 26 combination

যাৰচেন্দ্ৰ বেঁকি-এই block use কৰা হৈলো multiplexers

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

$$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$$

module pencode

```
( input [3:0] I,
  input [1:0] S,
  output reg Y
);
```

always @(\*)

begin

Case(S)

2'b00 : Y = I[0];

2'b01 : Y = I[1];

2'b10 : Y = I[2];

2'b11 : Y = I[3];

default:

end

end module

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Date: / /

Sub:

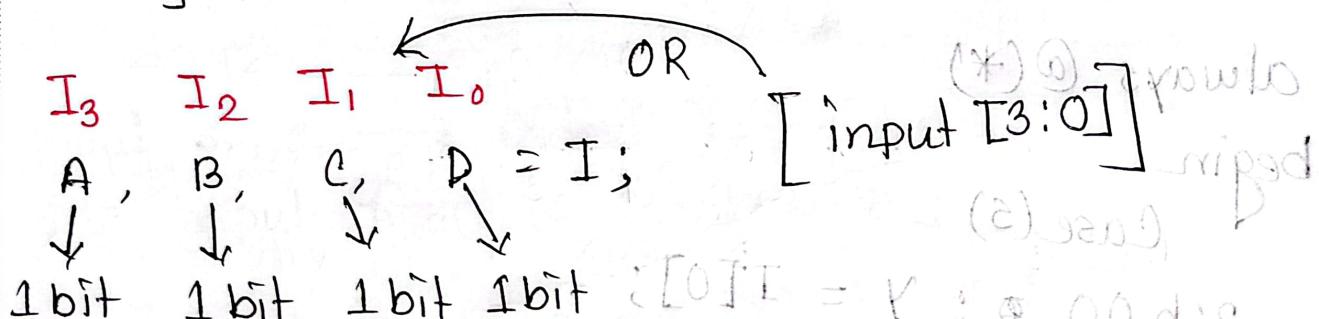
assign

$y = (A > B) ? A : B ; \rightarrow$  another operator  
 assign ternary operator

assign

$y = (S_2 = 2'b00) ? I[0] : ($   
 $(S_2 = 2'b01) ? I[1] : ($   
 $(S_2 = 2'b10) ? I[2] : ($

$\{A, B\} \rightarrow A, B$  ২টি অরকে combined



অথবা

$$A = I[0];$$

$$B = I[1];$$

$$C = I[2];$$

$$D = I[3];$$

more readable.

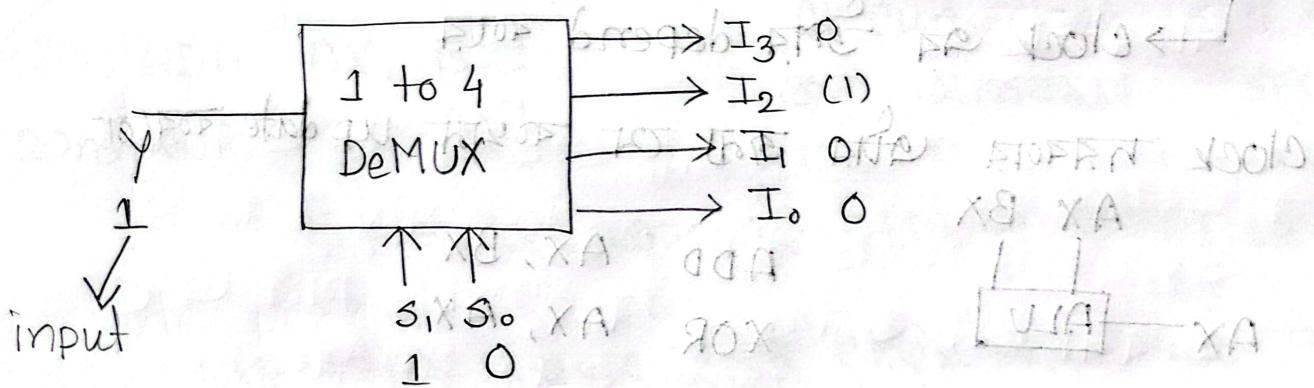
\* practice exercise problem

Sub:

ECE 01 II

De

## Demultiplexers:



$S_1$	$S_0$	$I_3$	$I_2$	$I_1$	$I_0$
0	0	0	0	0	Y
0	1	0	0	Y	0
1	0	0	Y	0	0
1	1	Y	0	0	0

4 to 2 multiplexers 3rd টপায়ে = বোরা যায়

1. assign ternary: output  $\rightarrow$  wire
2. always block:
  - i) if else { output
  - ii) case { (register) } practice

✓ Sat Sun Mon Tue Wed Thu Fri

Date: 11 / 10 / 2023

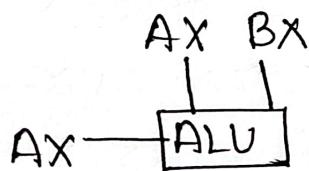
Sub:

## HDL 2

### Sequential Circuit

→ Clock এবং উপর নির্ভর করে

Clock দ্বারা এটির জন্য যে কোড কোড update করতে



ADD AX, BX

XOR AX, AX

একটি instruction কেবল হয়ে- আরেকটি instruction execute করবে, কোড যেননটা execute করবে  
dependent on clock speed

3.2 GHz

$$\Rightarrow 1s \longrightarrow 3.2 \times 10^9$$

∴ 1 ns বাজ করতে  $\frac{1}{3.2 \times 10^9}$  ns সময় লাগবে

Sequentially নির্ভর - clock speed পরপর instruction execute করে।

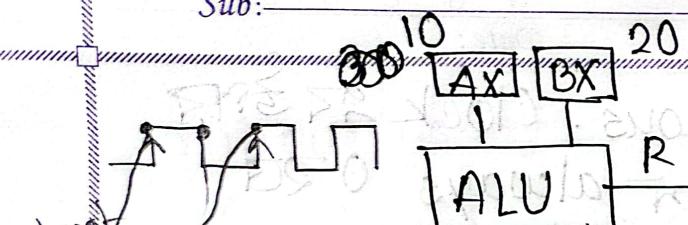
**clock synchronise** করে মান decide করে কোড

কোণ value update হয়।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: \_\_\_\_\_

Date: / /



$20\text{ns}[\text{ADD } AX, BX]$

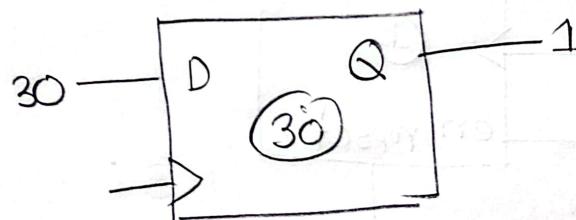
$20\text{ns}[\text{XOR } AX, AX]$

$\oplus \rightarrow \text{clock}$  এর এই অবস্থায় যাবলে update.

\* প্রতি cycle ক্ষেত্রে একবার clock এর pulse আজাব মানে update হবে।

\* মানে পরের instruction এর ক্ষেত্রে value update হবে।

\* clock না যাবলে synchronization করতে পারা যাবেন। যখন দ্বিতীয় তখন update হবেন। clock নিশ্চিত যখন দ্বিতীয় তখন update করবে। করে - ২৫



\* updated value র দ্বিতীয় পরের instruction execute করবার আগে update হবে।

এর ক্ষেত্রে

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
02	0	0	0	0	0	0	0

Sub: \_\_\_\_\_

Date: 01 / 1 / 1

\* reset pin asynchronous. Clock এর ক্ষেপণ  
depend করবেনা। 1 হলে always 0 হবে  
যায়ে সাধে,

DFF Reset to 0

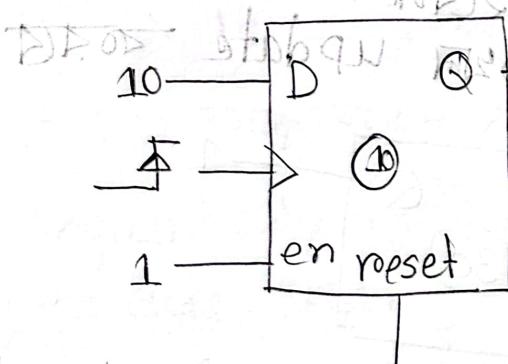
\* enable pin depends on clock. এটা একটা  
synchronous pin. অর্থাৎ only 1 টি

enable হবেনা। 1 টি হলে enable pin এ ও  
reset pin 0 টিতে হবে। enable pin clock  
positive edge থাবাতে হবে, data update হবে।

\* enable pin, positive edge এ, থাবাসতে

মনে 0 হয় ও reset pin 0 হয় তাহলে \*

data update হবেনা



\* current off করে দিলে Flipflop এর ক্ষেপণ

value 0 হয় যাবে, কারণ Flipflop volatile  
হাবে value update এ current on নাই,

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date:

module d-ff

```
( input wire clk,
  input wire d
  output reg q
);
```

always @ (posedge clk)

begin

```
  q1 <= d;
  q2 <= d;
```

memory ত্বক্ষয়ি-  
জানে আবার use  
করা যাবে,  $q = d$   
memory ত্বক্ষয়না।

একসাথে execute হবে,  
parallelly. দুইটা আলাদা  
memory তাই যাবে, কিন্তু  
combination এ একটুই  
memory তাই যাবনা ও আবার  
এটা বলবা।

$q \leftarrow d$

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
	○	○	○	○	○	○	○

Date: / /

Sub:

module d-ff

→ Reset add রোলেন্স

```
( input wire reset; রোলেন্স ফার্ম
  input wire clk; সোন্ন ফার্ম
  input wire d; ডিপস ফুর্টি
  output reg q; কাইড ফুর্টি
);
```

always @ (posedge clk, posedge reset)

begin

if (reset)

q <= 1'b0;

else

q <= d;

end

end module

b = p

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

module d-ff

(

input wire (reset);

input wire CLK;

input wire en;

input wire q\_d;

); output reg q;

enable add  
বর্ণনা

বিবরণ

always @ (posedge CLK, posedge reset) begin

if (reset) q <= 1'b0;

else if (en)

q <= d;

end

end module.

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

timescale 1ns / 1ps

module d-ff-fb();

reg clk,

reg reset,

reg en,

reg d,

wire q;

d-ff unit (reset, clk, en, d, q);

always @  
begin

clk = ~clk;

#10

end

initial begin

clk <= 0;

reset <= 1;

en <= 0;

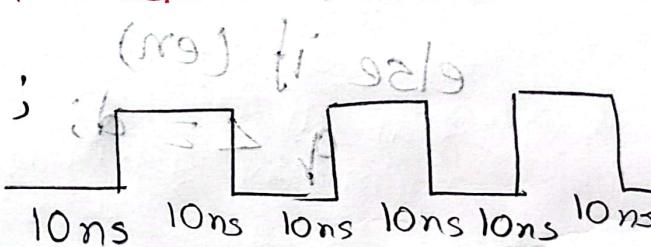
d <= 0;

initial

begin

till the end  
বন্ধ করে দিব

কারো উপর depend না



10ns এর পর 1,0 হত

মানে,

শুধুমাৎ 10ns

Sat  Sun  Mon  Tue  Wed  Thu  Fri

Date: / /

Sub:

১০০১ প

#20

reset <= 0;

end

end module.

\*initial begin block এ সবগুলো একবার করে  
execute হবে

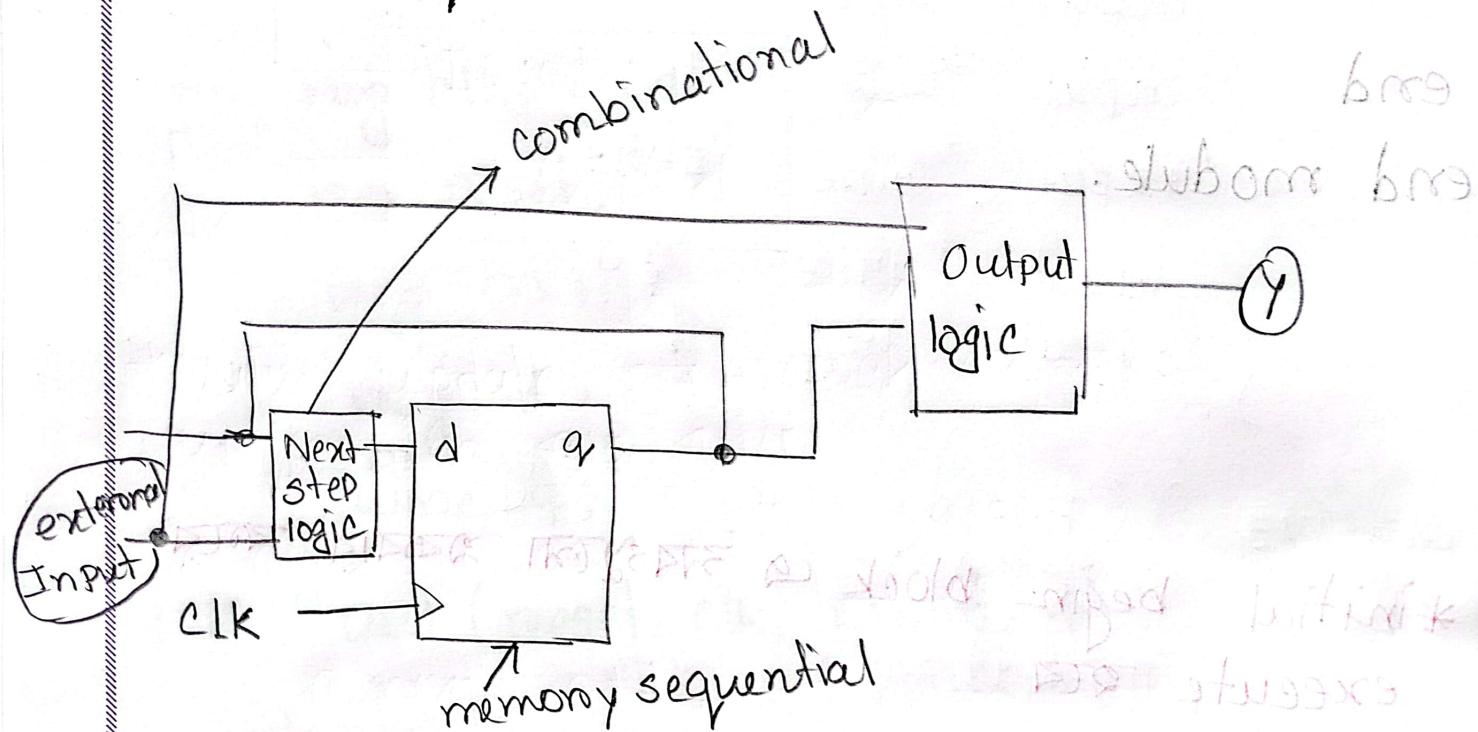
✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 17 / 10 / 2023

## Synchronous Design Methodology:

sequential + combinational



\* Next step logic decide করে data update  
হবে কিনা

~~modular~~ - off

FF এ always block এ output register এ

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

Sub:

module d-ff

( input clk,

input d,

output reg q;

d-FF  
[3:0];

reg r-next, r-neg;

always @ (posedge CLK)

begin

$$q \leftarrow d \rightarrow \text{হওনা}$$

$$r\_neg \leftarrow r\_next; \quad r\_next \leftarrow b$$

end

output logic

assign q = r-neg;

next step logic

assign r-next = d;

Assign এর কাজ always block value use করা হলো]

Ans:

always @ (\*)

begin

q = r-neg;

end

input [3:0] d, ] → For 4 D FF  
output [3:0] q ]

next step maintain  
r-next, r-neg maintain করা (data set  
করা)

বিধয়ে যদি q ও d  
একসাথে প্রক্রিয়া

[best practice memory  
তে direct করা দরকার  
এজন জাতে temporary

always @ (\*)

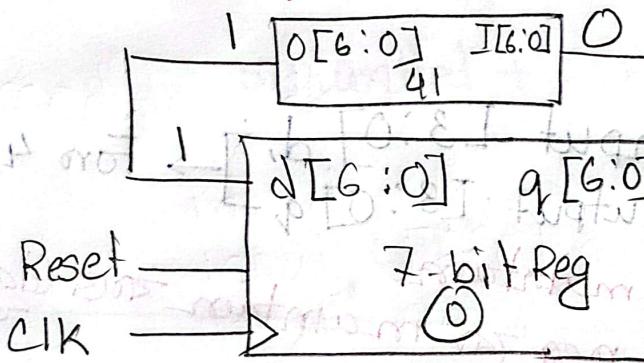
begin

r-next = d;

end

Sub: \_\_\_\_\_

## 7-bit up counter design



\* Value update হলে (+) clock ক্ষেত্রে, এখনে  
up counter বাতিল করতে।

$$r_{\text{reg}} \leftarrow r_{\text{next}} + 1;$$

→ update  
করতে এবং

$$r_{\text{next}} = d + 1;$$

change হয়।

\* output pin input pin same

module d-ff

(input CLK;

input reset,

output [3:0] q

):

reg [6:0] r\_reg  
r\_reg <= r\_next;

✓ Sat Sun Mon Tue Wed Thu Fri  
 Date: / /

Sub:

always @ (posedge clk)

begin

if (reset)

r\_reg <= 7'b0000\_000;

else

r\_reg <= r\_next

end

always @ (\*)

begin

q = r\_reg // 00\_0000 d

end

always @ (\*)

begin

r\_next = q + 1;

end

end module

Sat	Sun	Mon	Tue	Wed	Thu	Fri
✓	○	○	○	○	○	○

Date: / /

Sub:

7-bit FF to normal register: ④ expands  
7-bit

module d-lf

```
( input CLK,
  input [6:0] d,
  input reset,
  output reg [6:0] q
);
```

always @ (posedge CLK)

begin

if (reset) q = 00000000;

else q = d;

end

module Addone

```
( input [6:0] I,
  output [6:0] O,
);
```

assign O = I + 1

end module

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Sub:

Date: / /

## module 7bit - counters

( input clk,

input reset,

output [6:0] q

): wire [6:0] d\_t, d\_0;

register register1 (clk, d\_t, reset, d\_0);

→ memory circuit

add one → next step logic

Addone circ1 (d\_0, d\_t);

assign q = d\_0; → output logic

end module

Sat  Sun  Mon  Tue  Wed  Thu  Fri  
 Date: 18/10/2023

Sub:

ALU

(a) arithmetic

$+$ ,  $-$ ,  $*$ ,  $/$ , Comp

P [LOGIC] logic

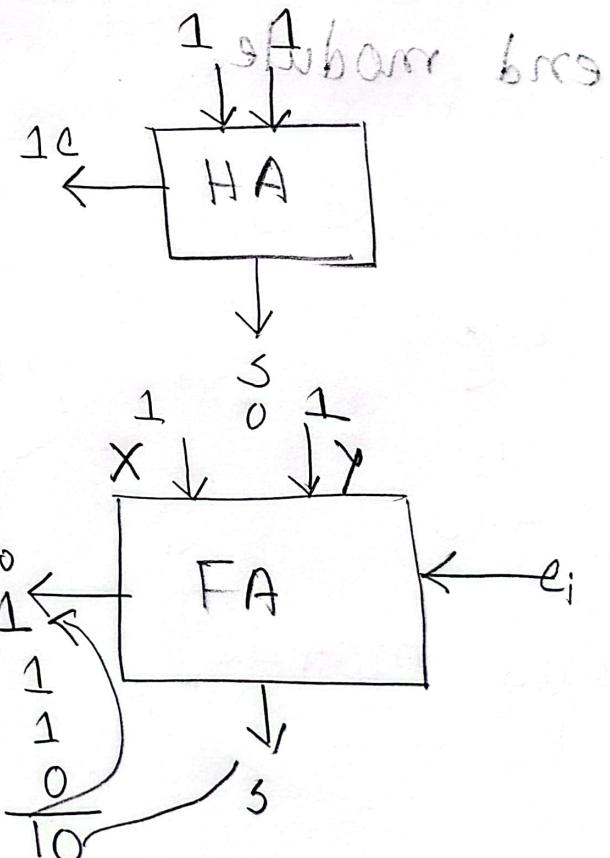
(b) logic

AND, OR, XOR

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\begin{aligned}
 S &= \bar{X}Y + X\bar{Y} \\
 &= X \oplus Y
 \end{aligned}$$

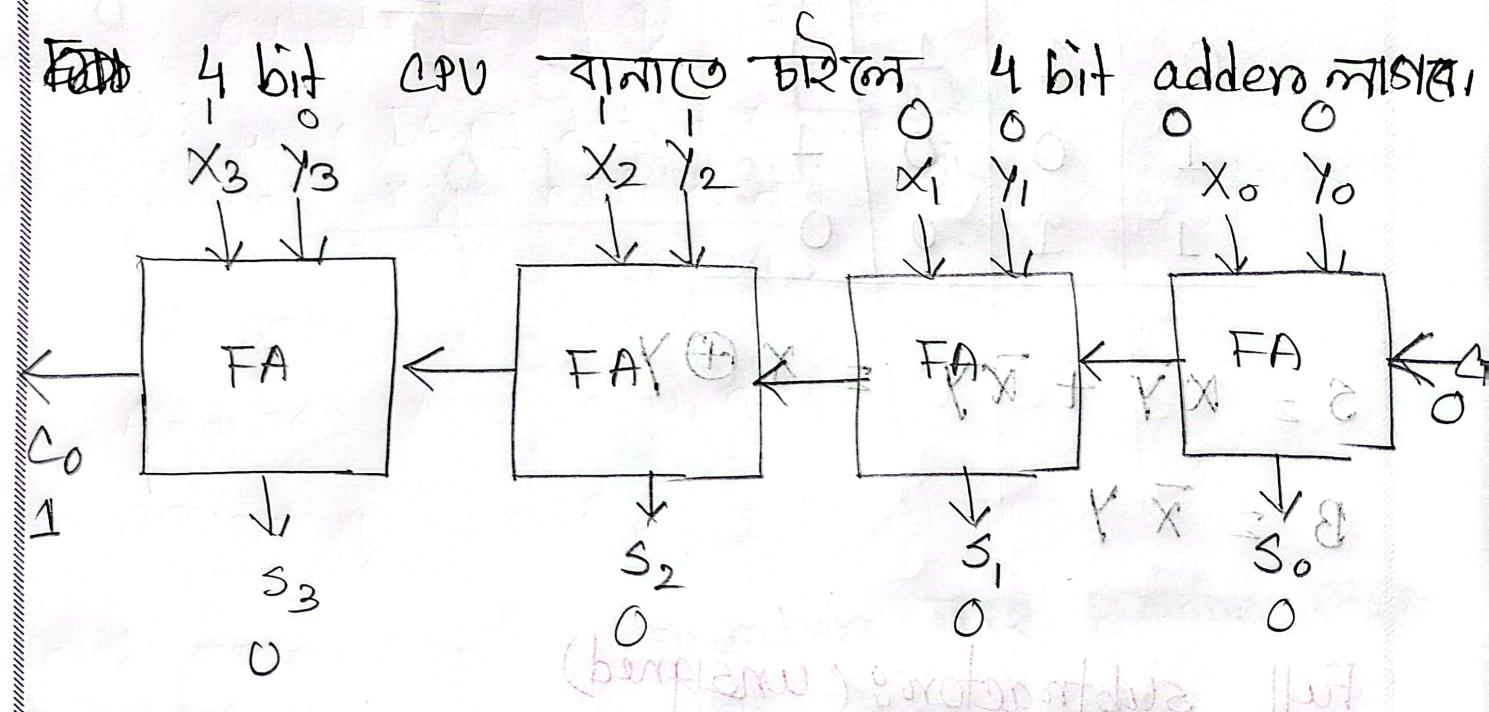
X	Y	Cin	C <sub>0</sub>	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Sub:

$$S = X \oplus Y \oplus C_i$$

$$C = XY + C_i(X \oplus Y)$$



$$2^4 - 1 = 15 \text{ bit represent করা যাবে।}$$

কিন্তু 16 bit আসল তখন overflow

## Subtraction:

1. Signed (2's complement)

2. Unsigned (Subtractors)

## Half subtractors: (Unsigned)

X	$\bar{Y}$	B	S
0	0	0	0
1	0	1	1
1	1	0	0

borrow  $\oplus Y$   $\oplus X = S$

$\frac{1}{0} \rightarrow \frac{1}{1} = 1$

$$S = X\bar{Y} + \bar{X}Y = X \oplus Y$$

$$B = \bar{X}Y$$

## Full subtractors (unsigned)

X	$\bar{Y}$	$B_1$	$B_0$	S
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$\frac{Y \downarrow \rightarrow \text{borrow}}{1 1 0 1 1 0 1 1} = 1 - 0$

Sub:

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Date: / /

unsigned হলে For 3 bit 0 to  $2^3 - 1 = 7$  পর্যন্ত  
represent কোনো  
যাবে।

Signed

0	0	0	1
0	0	1	2
0	1	0	3
<hr/>			
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

3 ~~পর্যন্ত~~ positive  
side এ represent  
কোনো যাবে।

\* basically negative numbers নাই positive মাত্রে  
positive numbers কোনো হচ্ছে।

$$\begin{array}{r}
 3 \\
 4 \\
 \hline
 -1
 \end{array}
 \quad
 \begin{array}{r}
 0 \ 11 \\
 1 \ 00 \\
 \hline
 ① \ 111
 \end{array}$$

→ represent

কোনো যাবেনা

$$\begin{array}{r}
 5 \\
 7 \\
 \hline
 -2
 \end{array}
 \quad
 \begin{array}{r}
 101 \\
 011 \\
 \hline
 ① \ 110
 \end{array}$$

\* unsigned এ negative  
represent কোনো  
যাবেনা

Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub:

Date: / /

2's Complement ~~করে~~ negative value

আনতে হবে।

$$0 \ 0 \ 0 = +0$$

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

$$0 \ 1 \ 0 = +2$$

$$0 \ 1 \ 1 = +3$$

$$\underline{1 \ 0 \ 0 = -0}$$

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

$$1 \ 1 \ 0 = -2$$

$$1 \ 1 \ 1 = -3$$

+3 - 0 আছে তাহুমে যানা, 2's complement

use করা যাব, এস দিয়ে easily adders

circuit use করে যোগ দিয়া করা

যাব, আলাদা করা লাগেন।

Full subtractor  $\rightarrow$  unsigned subtraction

$$\begin{array}{r} 0110 \\ - 0011 \\ \hline 11 \end{array}$$

$$\begin{array}{r} 0110 \\ - 0111 \\ \hline 11 \end{array} \rightarrow \text{আমরেনা}$$

sign subtraction  $\rightarrow$  2's complement

Signed

1	000
2	001
3	010
4	011
-4	100
-3	101
-2	110
-1	111

unsigned

$$\begin{array}{r} 6 = 110 \\ - 2 = 010 \\ \hline 4 = 100 \end{array}$$

ক্রি range এ মানো  $\leftarrow$  ~~পার্স~~  
 min minus (-) num বিয়োজন  
 করা মারেনা করণ ক  
 range এ নাই

\* unsigned করতে চাহলে যেকী bit লাগব।

$$\begin{array}{r} 010(2) \\ 011(3) \\ \hline 101 \rightarrow (-2) \end{array}$$

↳ signed overflow

Sub: CSE II 30

## Signed subtraction!

$$\begin{array}{r}
 & -3 \rightarrow 2\text{'s complement} \\
 \begin{array}{r}
 010(2) \\
 101(-3) \\
 \hline
 111(-1)
 \end{array} & \begin{array}{r}
 011(3) \\
 -100 \\
 +1 \\
 \hline
 101(-3)
 \end{array}
 \end{array}$$

↓  
signed

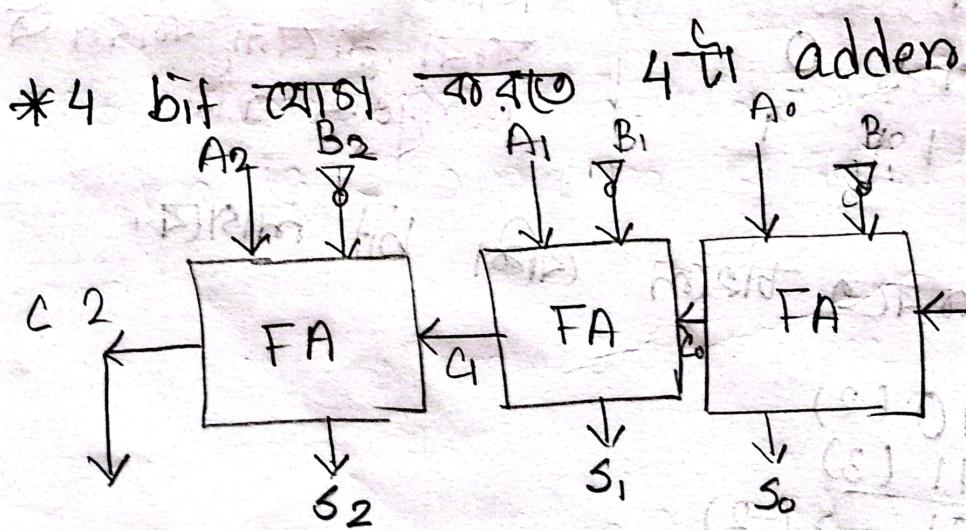
sign subtraction basically subtraction না

যোগ করবা :

$$\begin{array}{r}
 010 \\
 -101 \\
 \hline
 \end{array}$$

sign overflow  $\rightarrow$  representations support করতে

করতে, same sign যোগ করে result, opposite sign.



signed subtraction

Sub: \_\_\_\_\_

✓	Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○	○

Date: / /

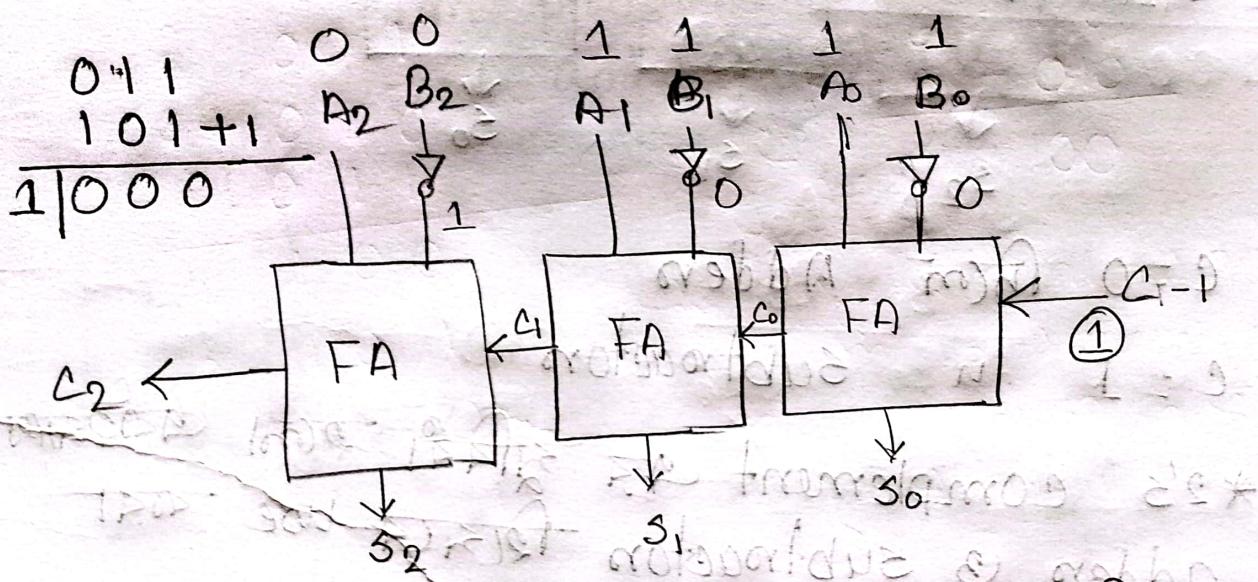
$$A - B$$

$$= A + (-B)$$

= A + 2's complement of B

= A + 1's complement of A + 1

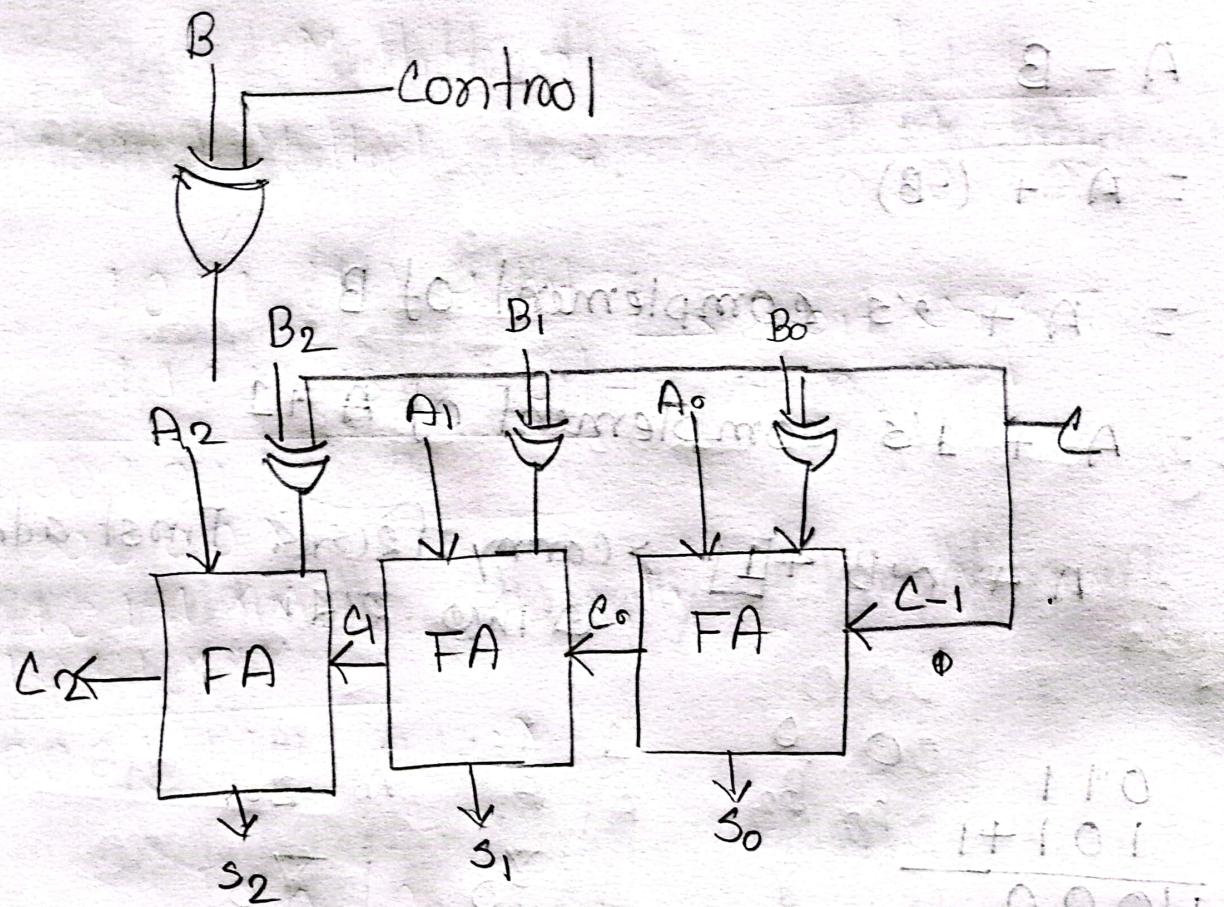
= A +  $\sim B + 1$  → carry হিসেবে First adder  
এন্টিপুল রয়ে।



~~একটি একটি তেই~~ Adders & subtractors হিসেবে  
use করা যাবে।

Sat Sun Mon Tue Wed Thu Fri  
Date: / /

Sub:-



$C=0$  টিপ্পনী Adder

$C \geq 1$  n Subtractor

\* 2's complement এর সুবিধা হলো একটি জাতে  
adders & subtractors হিসেবে we করা  
মাত্র।

\* কিন্তু এসে তা only 3 bit. Computers এ 64 bit  
we করা হয়।

\*  $C_1$  না আসলে বাধ্যতা 3rd FA করা।  
করবেন।

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

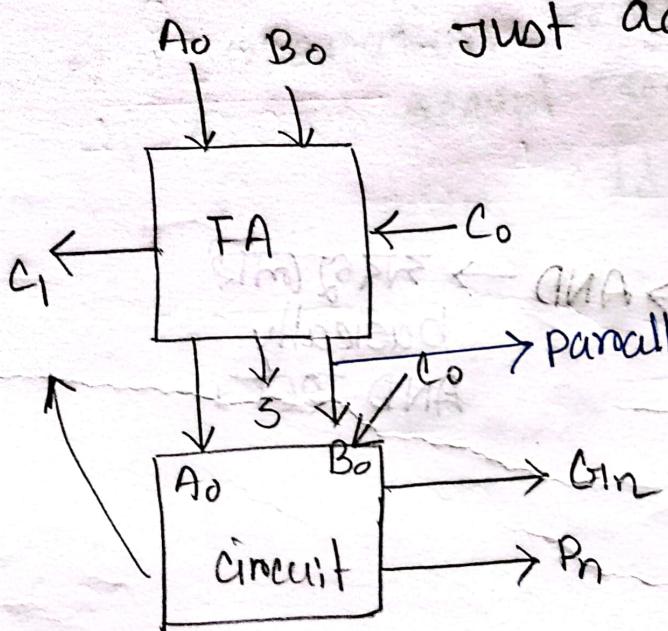
Sub:

\* Adders usually অনেক slow হয়। এটাকে  
Faster করার জন্য parallel adder ব্যবহার করা যায়।

$$C_n = \boxed{A_n B_n} + \boxed{(A_n + B_n)} C_{n-1}$$

G<sub>n</sub>      আজি calculate  
                করা থাবাব

\* parallelly calculate করা থাবাব, as  
sum problem না, carry আমল  
just add করা লাগব।



আগে 3 দীপে হবে বিট্ট  
এখন 2 দীপে

$$\begin{array}{r} 0A 1A \\ 0B 1B \\ \hline \end{array}$$

\* carry র জন্য wait করা  
লাগব but level ব্যবস্থা  
যাই। difference অনেক  
হচ্ছে হচ্ছে 64 bit এ  
impact হচ্ছে অনেক

$$\begin{array}{r} 0A 0A 0A 0A \\ 0A 0A 0A 0A \\ \hline 0A 0A 0A 0A \end{array}$$

✓ Sat Sun Mon Tue Wed Thu Fri

Sub: \_\_\_\_\_

Date: / /

## Multipliers

Unsigned Multiplication:

	1	0
1	1	0
0	0	0

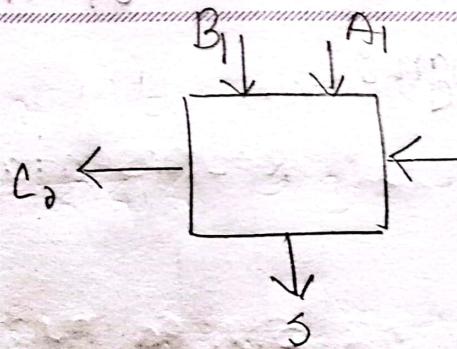
1011 (11) 4 bit ] → 8 bit আসবে  
 1101 (13) 4 bit after mul

$$\begin{array}{r}
 00001011 \\
 0000000X \\
 \hline
 001011XX \\
 01011XXX \\
 \hline
 10001111 (+143)
 \end{array}$$

$$\begin{array}{r}
 A_1 \quad A_0 \\
 B_1 \quad B_0 \\
 \hline
 0 \quad 0 \quad \overline{A_1 B_0} \quad \overline{A_0 B_0} \rightarrow \text{AND} \rightarrow \text{সবগুলোই} \\
 0 \quad A_1 B_1 \quad A_0 B_1 \quad X \qquad \qquad \qquad \text{basically} \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 A_2 \quad A_1 \quad A_0 \\
 B_2 \quad B_1 \quad B_0 \\
 \hline
 0 \quad 0 \quad 0 \quad \overline{A_2 B_0} \quad \overline{A_1 B_0} \quad \overline{A_0 B_0} \\
 0 \quad A_2 B_1 \quad A_1 B_1 \leftarrow A_0 B_1 \quad X \\
 A_2 B_2 \quad A_1 B_2 \quad A_0 B_2 \quad X \quad X \\
 \hline
 \end{array}$$

Sub:



first এ সবশুলো AND দ্বারা করতে হবে। carry  
যাকে যাজ পাখে দিয়ে যোগ। করার পর।

\* AND gate ৩ Full adders নামায়।

\* AND operation করতে, এরপর উপর থেকে data  
আসলে তা accept করতে এবং ফিল্টের মাধ্যমে  
তা নিয়ে then add করা হবে।

✓ Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: 07/11/202

## Unsigned multiplier design:

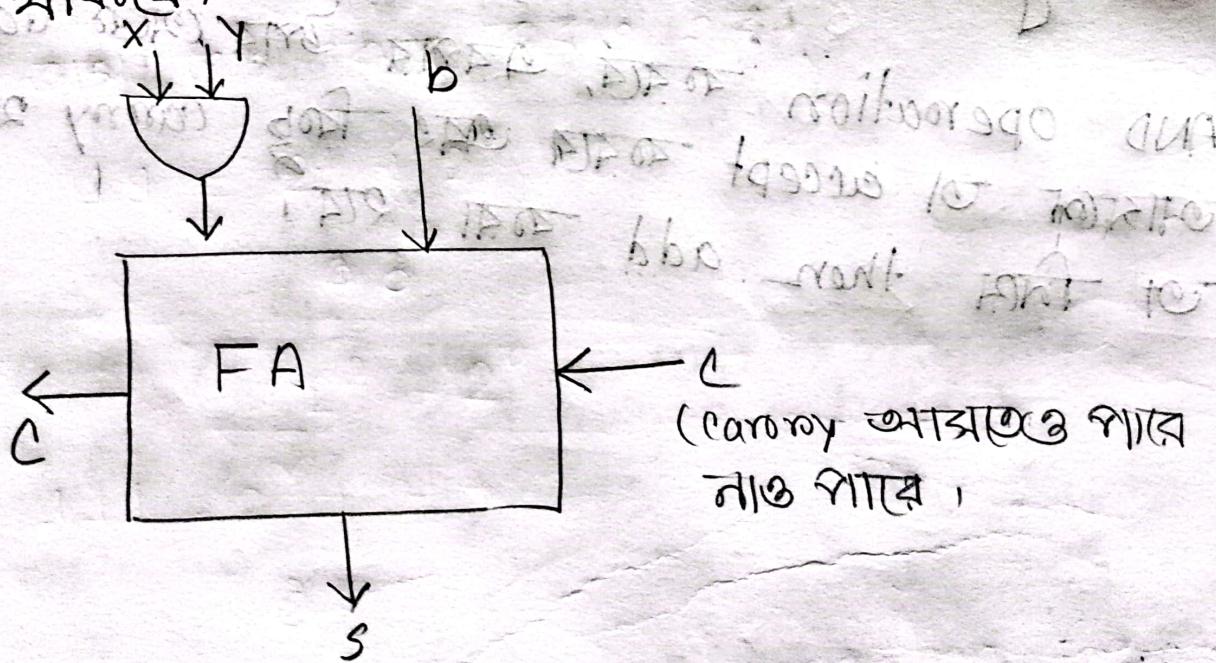
$x_1 \quad x_0$

$y_1 \quad y_0$

$$\begin{array}{r} & x_1 \quad x_0 \\ & y_1 \quad y_0 \\ \hline & (x_1 y_1) \leftarrow x_0 y_0 \\ & (x_0 y_1) \end{array}$$

Carry একটি স্টেপ আয়োজনের পথে অ্যাড হত

যা বলতে,

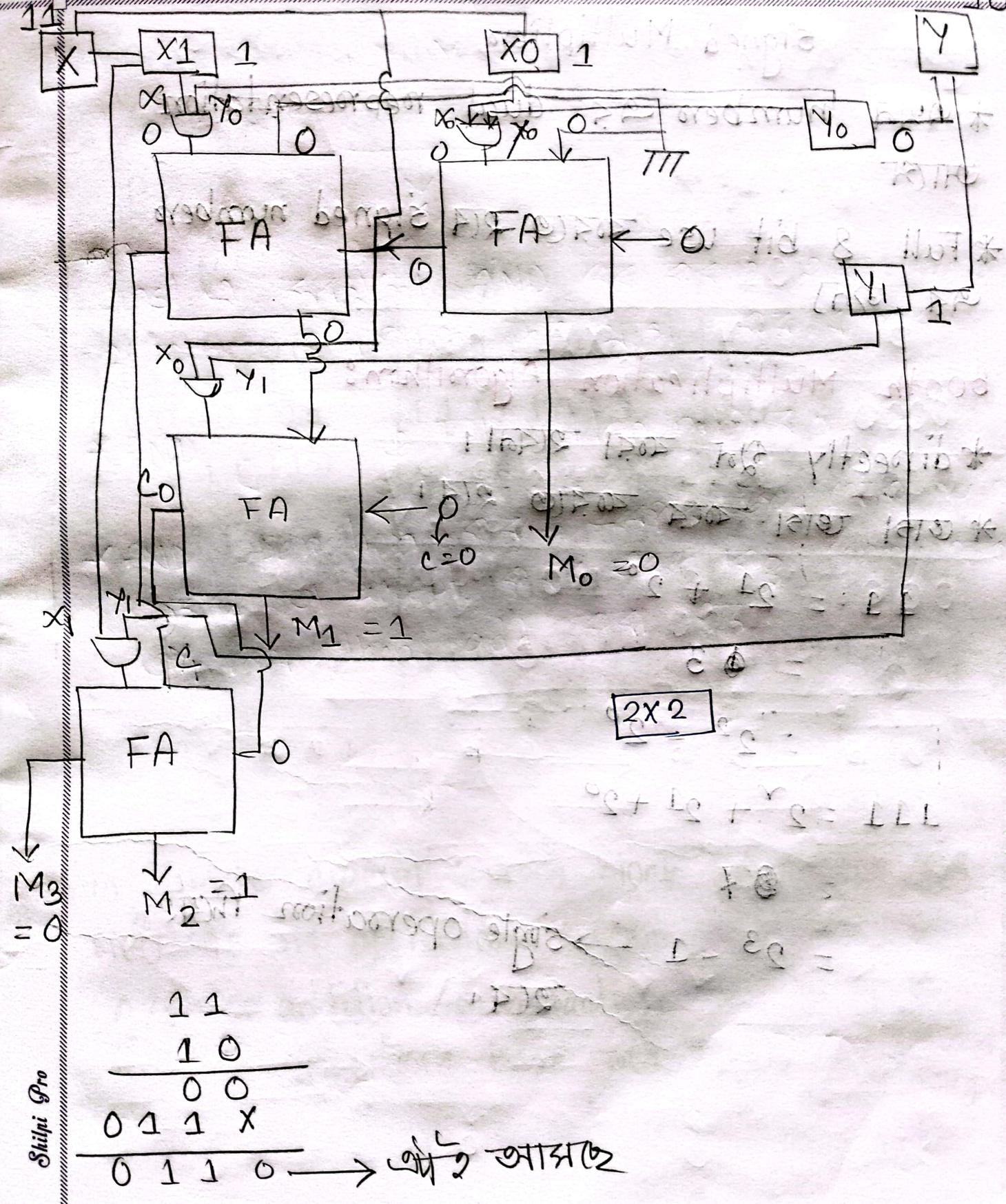


(Carry আয়োজনে পার  
নাও পারে,

Sat Sun Mon Tue Wed Thu Fri

Date: / / 10

Sub:



## Signed Multipliers

- \* এখন numbers এর dual representation আছে
- \* Full 8 bit use করতে হবে signed numbers এর তার

## Booth Multiplication Algorithm:

- \* directly ছুল করা হবে।
- \* তার জন্য করে ব্যবহৃত হবে।

$$11 = 2^1 + 2^0$$

$$= \textcircled{4} 3$$

$$= 2^2 - 2^0$$

$$111 = 2^2 + 2^1 + 2^0$$

$$= \textcircled{8} 7$$

$= 2^3 - 1 \rightarrow$  single operation পিসের  
হবে।

Sub: \_\_\_\_\_

Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○

Date: / /

$$X = 1101$$

 $X$  এর ছেফে,  $\begin{matrix} 00 \\ 11 \end{matrix}$  } No operation

$$Y = 1011$$

$$10 \rightarrow P - Y$$

$$P_2 X * Y$$

$$01 \rightarrow P + Y$$

$(1-1) \rightarrow$  last  $\Rightarrow$  extra add

P	Y	$XX_{-1}$	Operation
0000000000 11111011	111111011	11010	
00000101	11111011	<u>11010</u>	$P = R - Y$
11110110	<del>1101011</del> <del>11110110</del> shifted left (unsigned)	<del>11010</del>	<del>11010</del>
11111011	11110110 11101100	<u>11010</u>	$P = P + Y$ shift Y
00001111	11101100 11011000	<u>11010</u>	$P = P - Y$ shift Y
		<u>11010</u>	No operation

result

to create circuit: এমন block তৈরি করতে হবে যার  
জন্মে এই সব operation থাকবে।

A/S  $\rightarrow$  addition / subtraction

H, D  $\rightarrow$  এন্দের উপর base করে হয় যোগ / বিয়োগ / No operation

Sub: \_\_\_\_\_

Date: / /

figure of booth multiplier  $\rightarrow$  HxD just data bypass

বরাবর:

$$\begin{array}{r}
 11(-1) \\
 11(-1) \\
 \hline
 01
 \end{array}$$

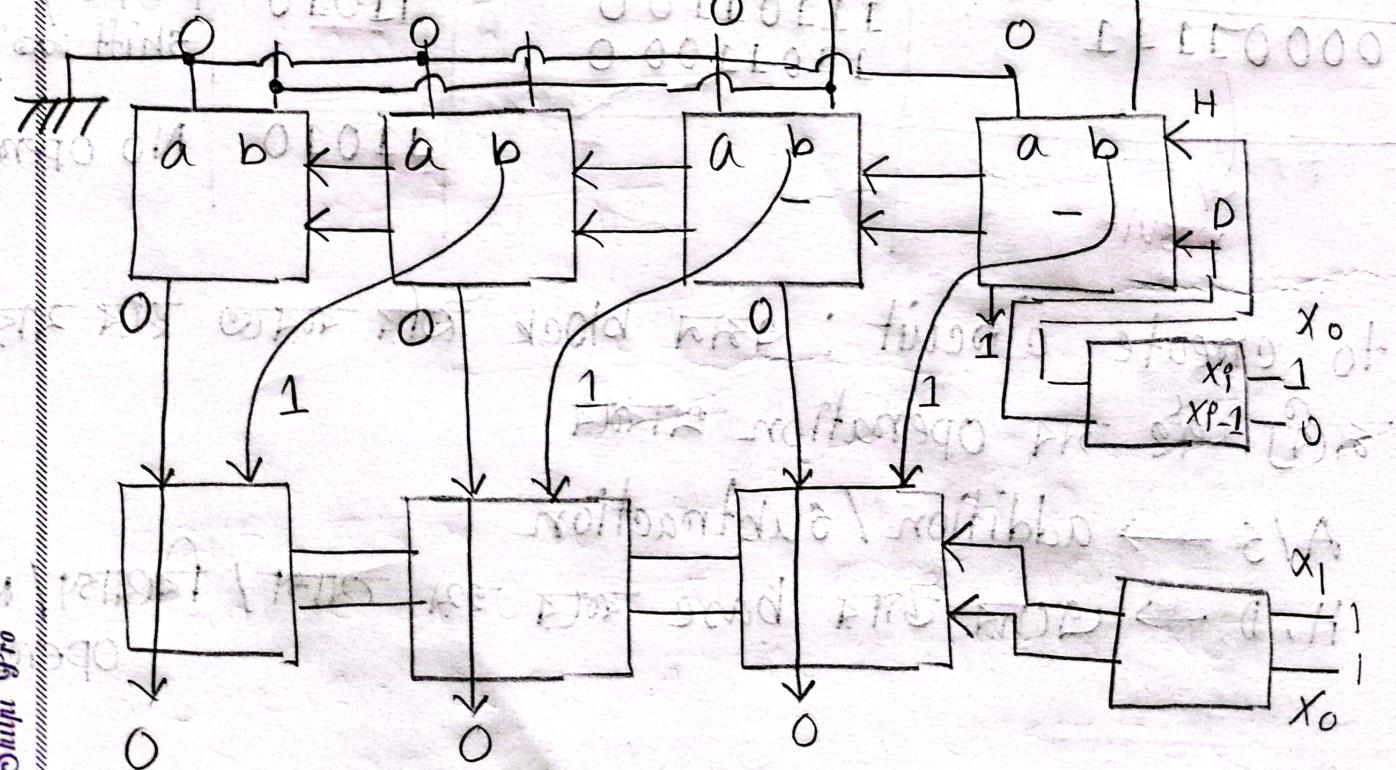
$$Y-9 \leftarrow CL$$

$$Y+9 \leftarrow LO$$

$$LLLOL = Y$$

$$Y + X = 9$$

P	0	01Y	LLXX+1	Operation	00000000
0000	0	1111	LL0L LLLL	LL0L LLLL	10000000
0001	1	1111	0110	P=P <sub>Y</sub> shift Y	0LL0 LL11
0001	1	1110	110	No operation	LL0L LLLL



Sub: \_\_\_\_\_

Sat	Sun	Mon	Tue	Wed	Thu	Fri
-----	-----	-----	-----	-----	-----	-----

Date: 08/11/2023

$$\begin{array}{r} 1011 \\ - 1101 \end{array} \quad (-5)$$

$$\begin{array}{r} 1101 \\ - 1011 \end{array} \quad (-3)$$

$$1011 = -\frac{3}{2} + 2 + 2^0$$

$$\begin{array}{r} 1101 \\ - 1011 \end{array} \quad \begin{array}{l} -8 + 2 + 1 \\ -5 \end{array}$$

0-11-1  $\rightarrow$  convert করে দুটি করবে

$$\begin{array}{r} 1101 \\ - 1011 \end{array} \quad \begin{array}{l} -2^2 + 2^1 - 2^0 \\ = -3 \end{array}$$

$$\begin{array}{r} 1101 \\ - 1011 \end{array} \quad \begin{array}{l} \text{last } 0 \text{ add} \\ \text{জন্মের } 0 \end{array}$$

$$\begin{array}{r} -2^2 + 2^1 - 2^0 = -5 \\ 1011 \end{array}$$

$$1101 \rightarrow -2^3 + 2^1 + 2^0$$

$$Y * X = (-5) * (2^2 + 2^1 - 2^0)$$

$$\begin{array}{r} 1101 \\ - 1011 \end{array} \quad \begin{array}{l} -2^2 * (-5) + 2^1 * (-5) - 2^0 * (-5) \\ \hline 5 \end{array} \quad \begin{array}{l} \rightarrow 0 \\ \hline 5 \end{array}$$

$$0000 0000$$

$$\begin{array}{r} 1111 1011 \\ (-5) \end{array}$$

$$0000 0101 (5)$$

$$\begin{array}{r} 1111 0110 \\ (-10) \end{array}$$

shift Y (1 bit shift)

$$1111 1011 (-5)$$

$$\begin{array}{r} 1110 1100 \\ (-20) \end{array}$$

shift Y (2 bit shift)

$$0000 1111 (15)$$

\* অসমক 1 ধারণে bypass করে দিতে হবে। Only transition period এ ধোতি বিদ্যুৎ রয়ে।

Sat Sun Mon Tue Wed Thu Fri

Sub: ECE 1180

Date: / /

## Unsigned Binary Division

$$0111) \overline{1111} (0010$$

$$\begin{array}{r} 11 \\ 00 \\ \hline 11 \end{array}$$

$$\begin{array}{r} 1101 \\ \hline 0001 \end{array}$$

$\rightarrow$  Not possible

$$\begin{array}{r} 0000 \\ \hline 001 \end{array}$$

$\rightarrow$  remainder

$$15/7$$

$$\begin{array}{r} 22 \\ R1 \end{array}$$

1111 এর মাঝে 000 bit add করে

4 bit বাই বাই

$$0111) \overline{00001111} (0010 \text{ carry easy}$$

$$\begin{array}{r} 0000 \\ \hline 0011 \end{array}$$

$$\begin{array}{r} 0000 \\ \hline 0111 \end{array}$$

$$\begin{array}{r} 0111 \\ \hline 0001 \end{array}$$

$$\begin{array}{r} 0000 \\ \hline 001 \end{array}$$

বিয়োগ করনে borrow আসলে ① বিয়োগ

শাবেনা তখন 0 বাইবো, কম্পিউট সিরকিট

design এ borrow করে আসলে NOT

গতে পাঠিত হবে, borrow 1 আসলে 0 ও  
0 আসলে 1.

Sub: \_\_\_\_\_

 Sat  Sun  Mon  Tue  Wed  Thu  Fri

Date: / /

A 0000000

A	PA	B	PA - B	Bo	Q
0001111	<u>000111</u> 0111 <u>1010</u>	0111	1010	1	0
0001111	<u>000111</u> 0111 <u>1010</u> 1100	0111	1100	1	0
	<u>000111</u> 0111 <u>0000</u>	0111	0000	0	1
	0000001	0111	1010	1	0

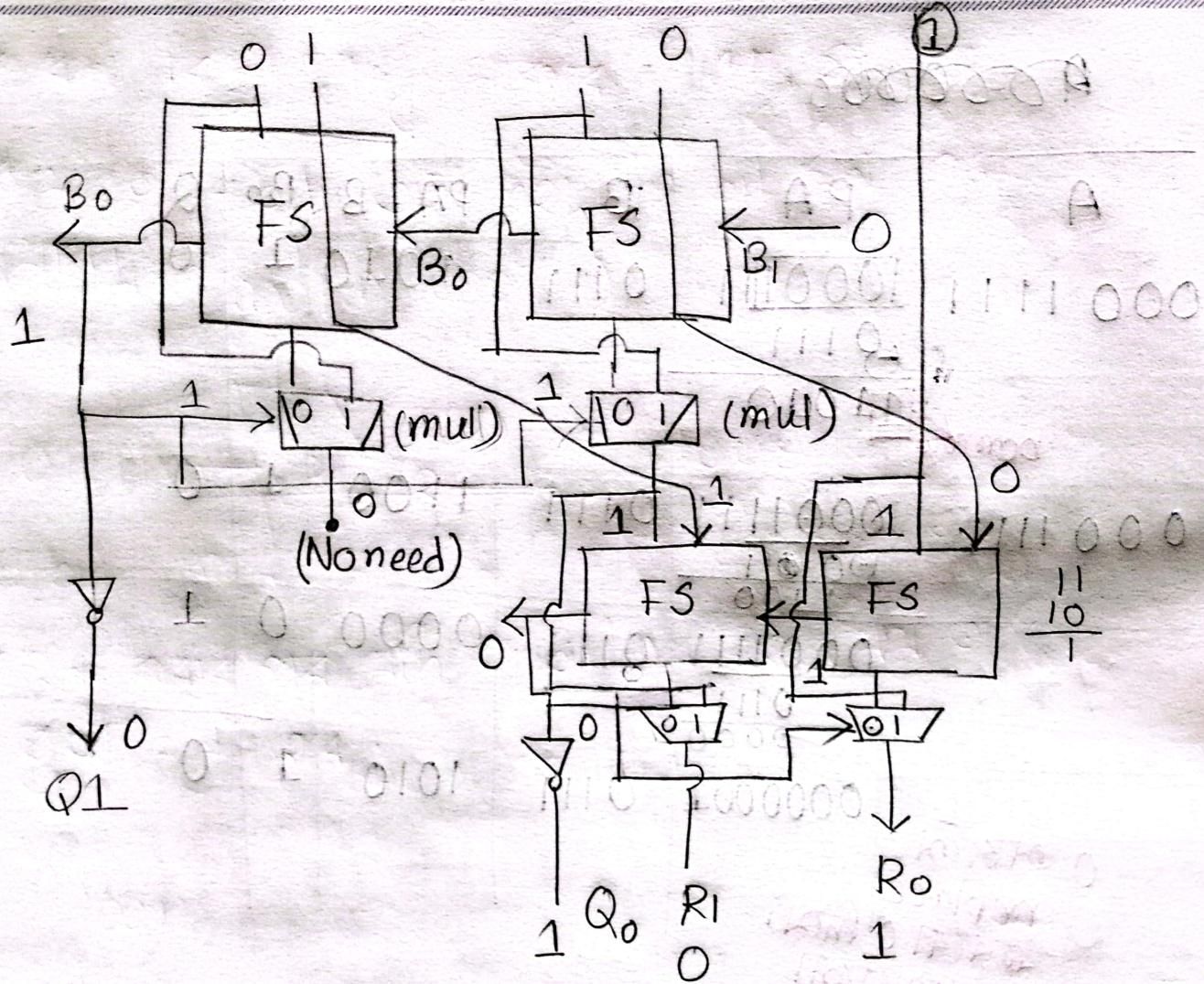
0 টান্ডেন  
replace  
যোর দো 0 ফিল  
ডেকা / ডাক্ষিণ  
বড়ে - ২ মিন  
৩ সেকেন্ডে বার্দ্ধ

$$10) \overline{011(01)} \\ \underline{10} \\ \overline{01}$$

সবগুলো unsigned তাই circuit design ৭  
subtractors লাগব।

Sat    Sun    Mon    Tue    Wed    Thu    Fri  
 Date: / / /

Sub: \_\_\_\_\_



exception করে crash করে দেওয়া হয়।  
 নাহলে তুল result আসবে, তোম result এক্ষার  
 আগলে শুরো program নষ্ট করে দিবে।

Sub: \_\_\_\_\_

Sat Sun Mon Tue Wed Thu Fri

Date: 14 / 11 / 2023

## Final 4 bit ALU Design

ALU

Arithmetic & Logical Arithmetic Operation

Addition (ADD)

Multiplication (MUL)

Subtraction (SUB)

Division (DIV)

shift

Left shift

Right "

Left Rotate

Right "

Logical operation

AND

OR

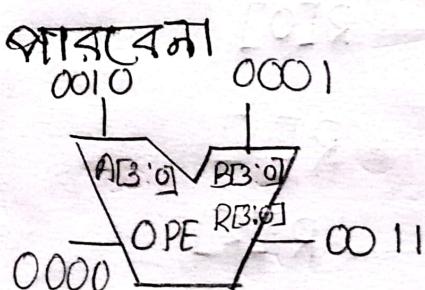
XOR

NOT

\* যতো 4 bit CPU ততো 4 bit maximum memory.

4 bit CPU মালে 4 bit এর সমস্যা calculation

বর্তমানে



\* OPE তে কি করবে  
দিব্যে কোন operation  
হবে, Addition এর অন্য  
0000 দিব্যে নিলাম,  
এলেকে 2<sup>4</sup> বা 16 DI  
Operation করবেনো যাক

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

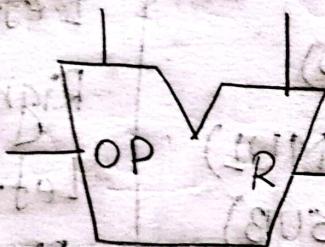
Sub:

\* যেকোনো Flag থাবলে, CF, SF, ZF.

\* যেহেতু 4 bit তাই ট্র্যাজি 2 bit দ্বারা বচা

যাবে।

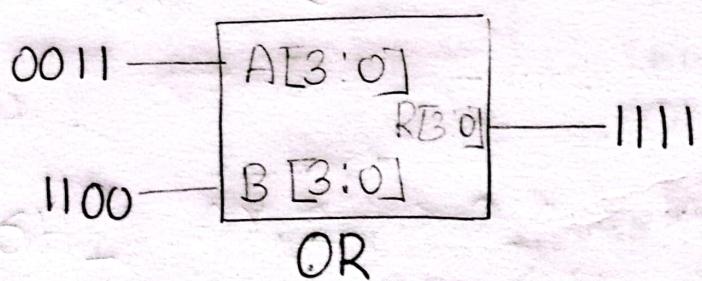
$0001 \leftrightarrow 1000 (-4)$



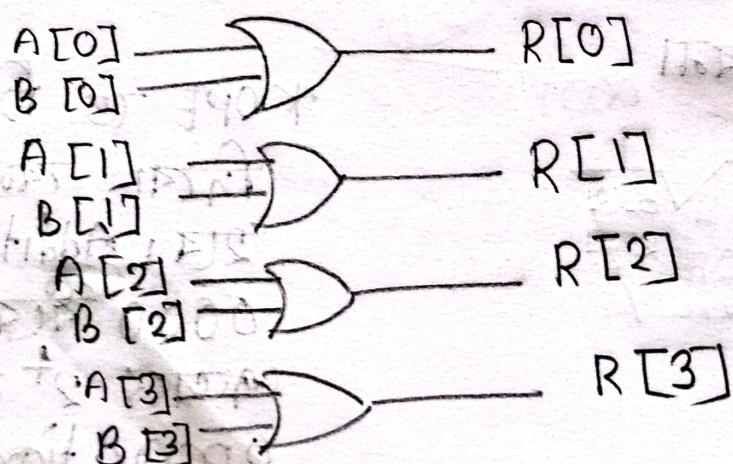
আমার  
বস্থা

\* কোনো বচার পর যদি 4 bit এর বাইবে মাঝে তাহলে

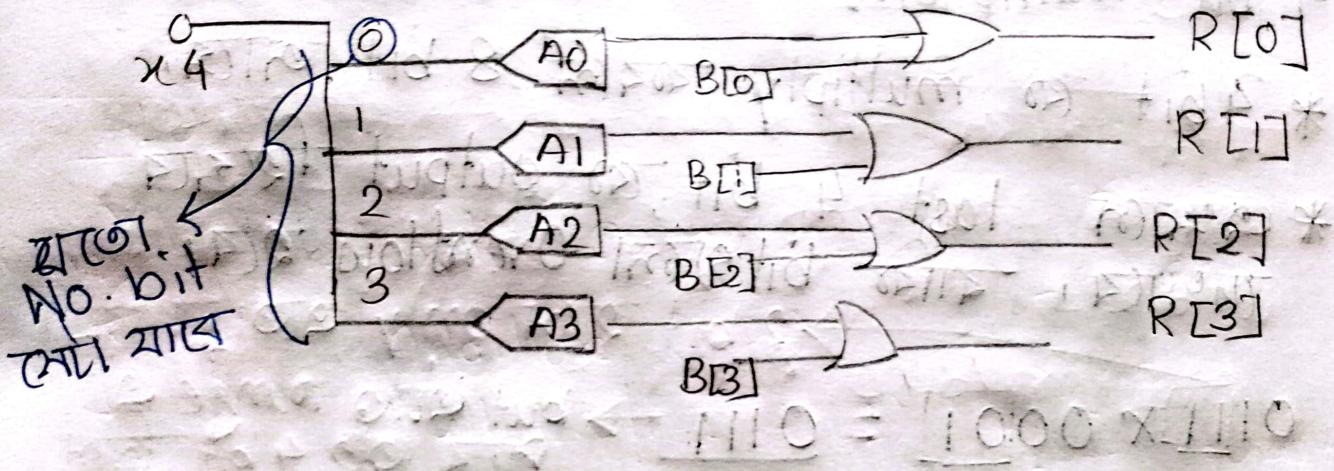
১) SF = 1 হবে।



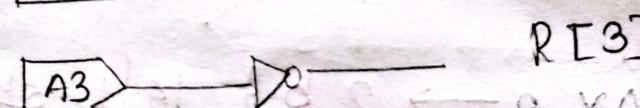
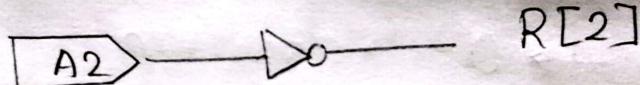
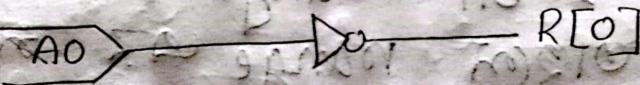
\* bit wise OR operation রিপ



Sub: \_\_\_\_\_ Date: / /



NOT এর জন্য:



4 bit Adder/Subtractors:

\* Selection line ৰ ০ দিলে ০ দিয়ে XOR কোর্যাবলী  
যদি input দিব্বা হয়েছিলো অস্টার্ট পাবে;

From → slide (মুখ্যত হবে)

<input checked="" type="checkbox"/>	Sat	Sun	Mon	Tue	Wed	Thu	Fri
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Sub: \_\_\_\_\_

Date: / /

## 4 bit Multiplier:

\* 4 bit কে multiply করলে 8 bit আয়ে,

\* এখানে last 4 bit কে output হিসেবে

দখাবে, বাকি bit জুলা Overflow হবে।

দখাবে, বাকি 2 bit হলে only এতাদুর ক্ষমতা।

$$0111 \times 0001 = 0111 \rightarrow \text{but এটা } 2 \text{ bit range এ। তাই } 4 \text{ bit পরে অতি } 4 \text{ bit পরি।}$$

\* 2 bit করে রাখলে only 3 bit ওজুলাই result হবে।  
 দিতো যা 2 bit range এ। তাই 4 bit  
 করা হয়- তাহলে range এর মধ্যে থাকলে  
 তা result show করবে। এবাবে করলে  
 অনেকদুর পর্যন্ত করা যাবে।

## Shift :

SHL AX, 2  $\rightarrow$  2 bit left shift

$$1011 \rightarrow 1100$$

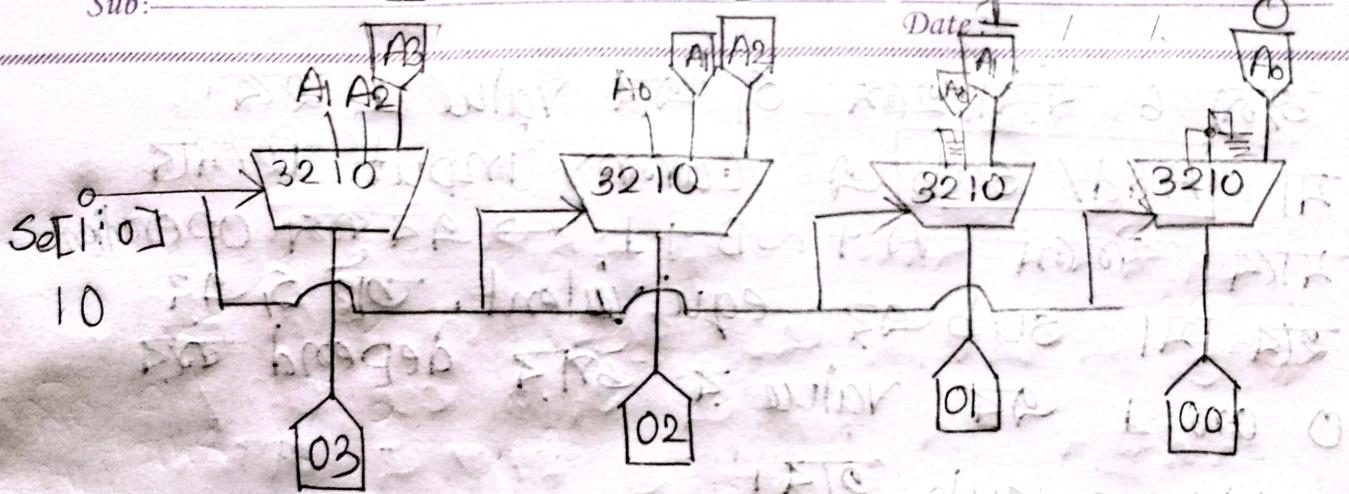
Sub: \_\_\_\_\_

1

0

Sat	Sun	Mon	Tue	Wed	Thu	Fri
-----	-----	-----	-----	-----	-----	-----

Date: \_\_\_\_\_



\* Selection line এ 00 টিকে মাত্র input আ সাবে

\* " " " 01 " connect হবে,

A0 এর multiplexer ground এ connect হবে,

\* আবার selection line 10 টিকে A0 এর 1112

ও A1 এর 2 ground এ connect হবে।

\* এসবে shift work করে।

বাকিটুলো from slide

\* Selection line decide করে যেগুলো operation হবে ALU তে।

\* একজনাথে যোগ / বিয়োগ করতে selection line S<sub>1</sub> এর value র উপর depend করে। প্রথম 5 টি add এর জন্য S<sub>1</sub> এ তাহলে 0 আবাবে যোগ addition এ হস্তানো। carry নাই, আবাবে sub এর

Sat	Sun	Mon	Tue	Wed	Thu	Fri
0	0	0	0	0	0	0

Sub:

Date: / /

তাণ্ডু 6 বিংশ টেল 3, এর value 1 হবে।  
মা **Add/ sub** এ carry input দিয়েও

মাধ্যে যদি  $A + \sim B + 1 \rightarrow$  এর কোন operation  
হবে মা sub এর equivalent. তাই এই  
0 on 1 এর value র উপর depend করে  
Add on sub হবে।

\* subtraction ও comparison also same

কুটীর তাণ্ডু  $s_1$  র 1 আমরে হবে।

যেটা ensure করতে হবে  $s_1$  র 1 আমরে

subtraction ও comparison এ

Must ensure  $s_1 \rightarrow 0 \rightarrow$  addition  
 $s_1 \rightarrow 1 \rightarrow$  subtraction / comparison

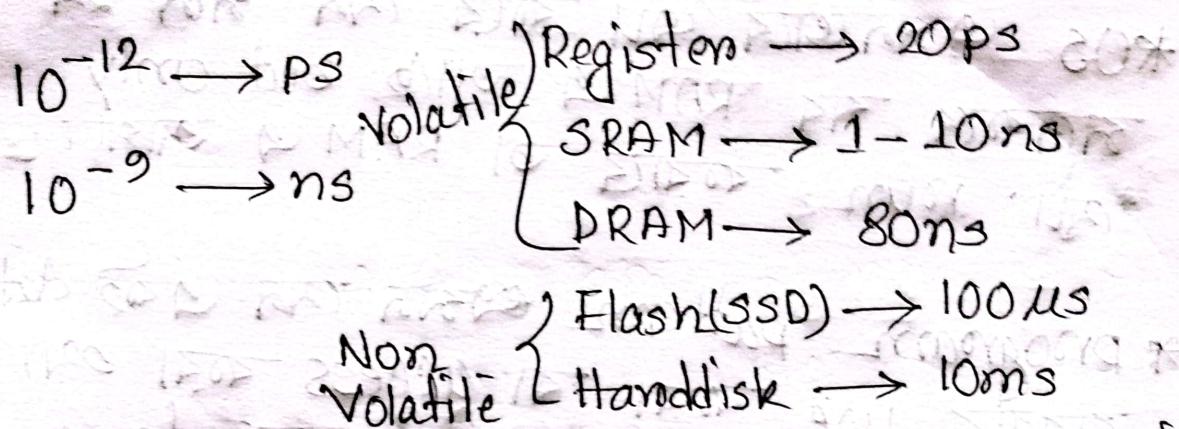
\* ZF check করতে 4th bit এর জর্দে OR  
করে NOT gate ফিলেই হবে।

Sub: \_\_\_\_\_

Sat Sun Mon Tue Wed Thu Fri  
Date: 15 / 11 / 2023

## Memory Hierarchy

\* অতি ধূমৰ Faster memory অতি বেশি cost.



নিচের দিকে গোলা time বাড়বে, cost বজাবে, size কমবে  
ক্ষয়ক্ষতি বাড়বে।

\* At least 1st memory non-volatile লাগবে, নাহলে  
Data save করা যাবেনা, তাই SSD on harddisk  
যেখানে 1st use করতেই হবে।

\* Large, fast & cheap is not possible all together

\* যখন program করি বীরে নেই memory একটাই,  
but Computer নিতৃপক্ষ মধ্যে আস করে নেই,

\* Memory বেশি বড় কিছু কি বাধ্যতে না পারলে  
Virtual memory কৈ বীরে নেই, এটা main  
programming memory রে,

Sat Sun Mon Tue Wed Thu Fri

Sub:

Date: / /

\* বিবে নিবে 1tb জায়গা আছে, যখন  
program run করবে মেটা RAM এ  
আসবে, যাবিং harddisk এ থাকবে,

\* OS এজনভাবে এসে করবে যেন সব সব  
সময়েই সব RAM এ আছে but only  
মেটা · run করাতে তা RAM এ আসবে,

\* program pattern এজন যেন এক্ষেত্রে data  
ব্যাবহার করা বা access করা হয়,  
তাই program এর সুবিধির তল্য frequent  
used data save করা হবে cache  
memory.

Data Flow:

Harddisk → RAM → Cache → Registers

\* OS প্রয়ো memory কে virtual memory বিবে,  
OS বলবে আপনি 1tb জায়গা আছে, কিন্তু  
actually তা না, এখন program 200GB  
জায়েছে তার বিছু part RAM এ রাখবে  
বাকি জায়েছে HDD, তে থাকবে।

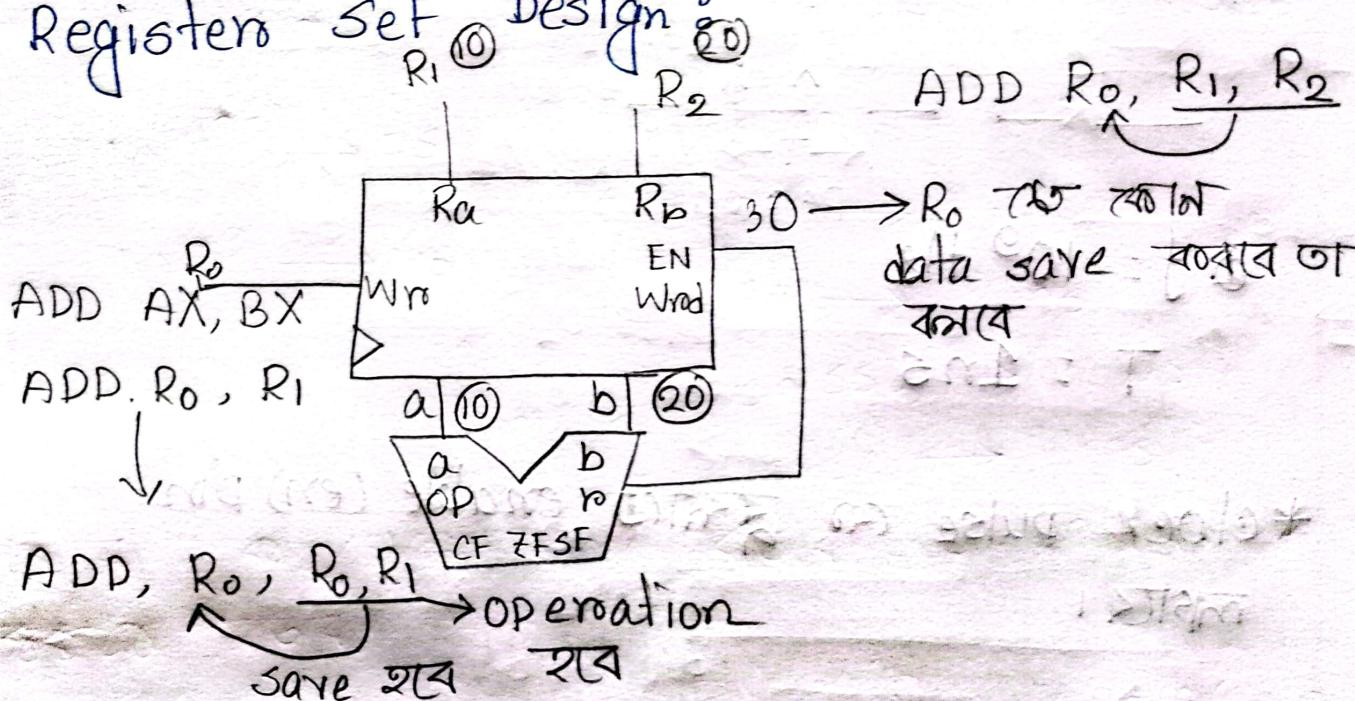
✓ Sat Sun Mon Tue Wed Thu Fri

Date: 1/1/2023

Sub:-

\* Registers এ ALU operation এর data যাবে,

Registers Set Design



ADD R0, R1, R2

R0 টত কোন  
data save কৰবো তা  
কলবে

\* Ra ও Rb তে R1 ও R2 এর value দেওয়া আ-

পরবর্তীতে a ও b তে চলে আবে।

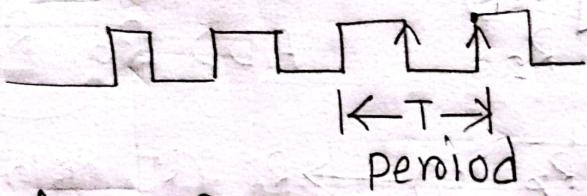
\* As 4 bit CPU, Registers 4 bit এর মতো হবে।

\* write কৰতে চাহলে EN 1 হতে হবে না হবে

write হবেনা।

\* CLK positive edge এ write কৰতে পারবেনা।

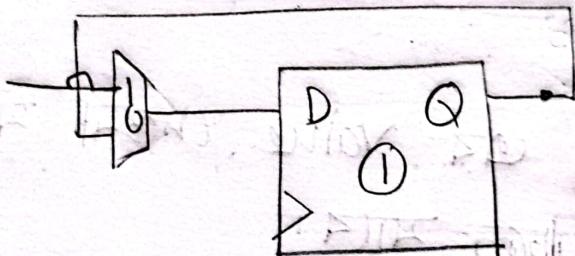
১ns এ  $10^9$  হারে চালাতে পারে এজন 1ns  
পর data update হবে।



$$\textcircled{1} f = 10^9 \text{ Hz}$$

$$T = 1\text{ns}$$

\* clock pulse এর ব্রহ্মাণ্ড enable (en) pin  
লাগবে।

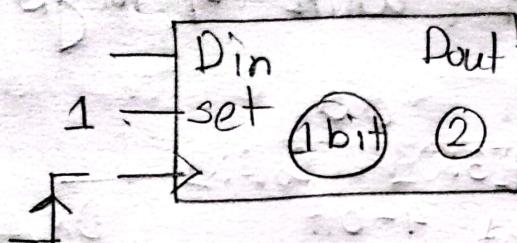


en  $\rightarrow$  0 জান যা output টা input, change  
হবেনা, value স্ব ছিলো তা।

en  $\rightarrow$  1 হল, data save কৰবে।

\* CLK pulse আমলেই থাকবে, অবসর্জন data accept  
কৰবো এবন না, তাই en এর value র উপর  
depend কৰবে হবে।

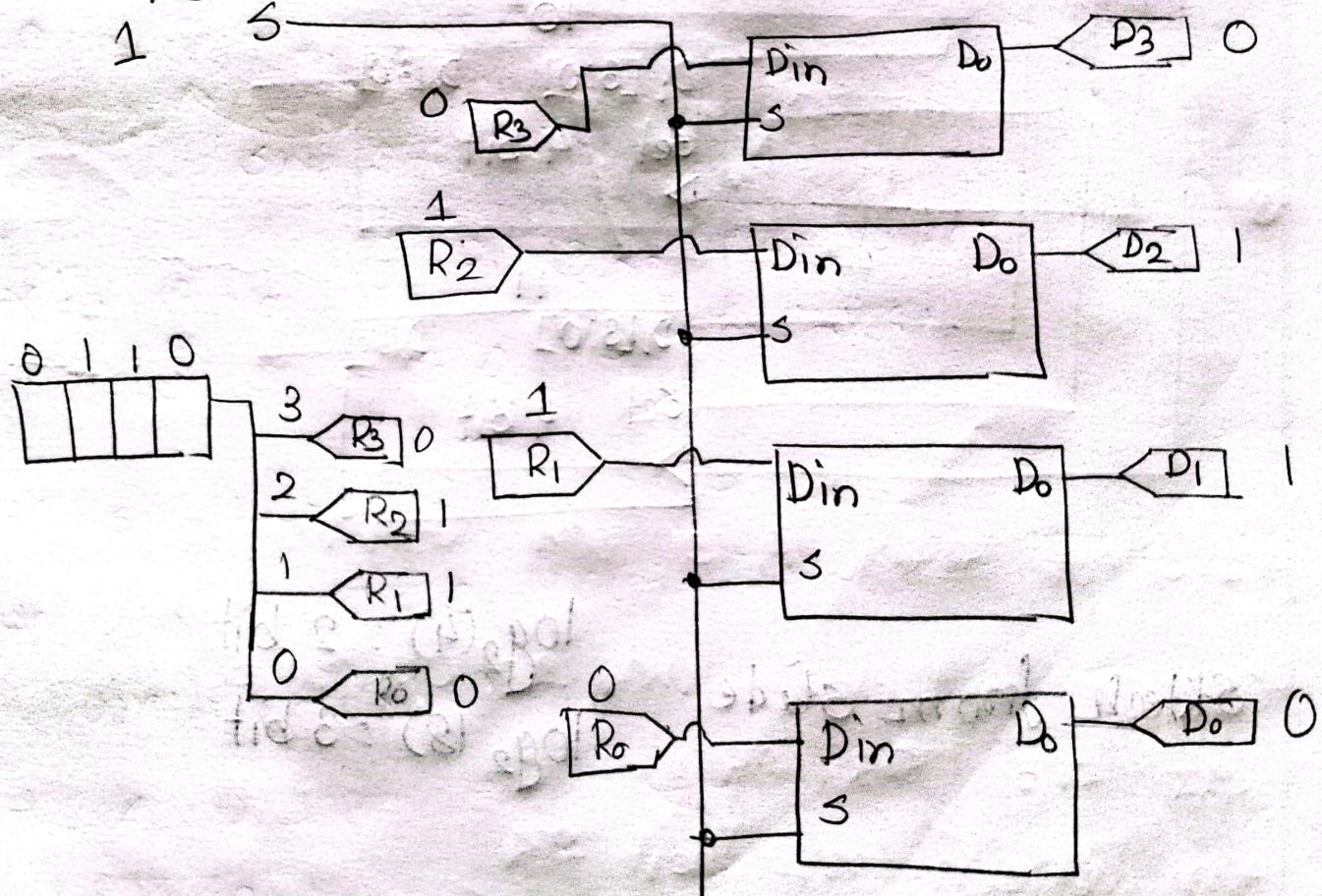
Sub:



1 bit register তাই only 0/1 value  
accept করবে।

\* 4 bit register বানাতে 4টি 1 bit registers use

ব্যবহার হবে।

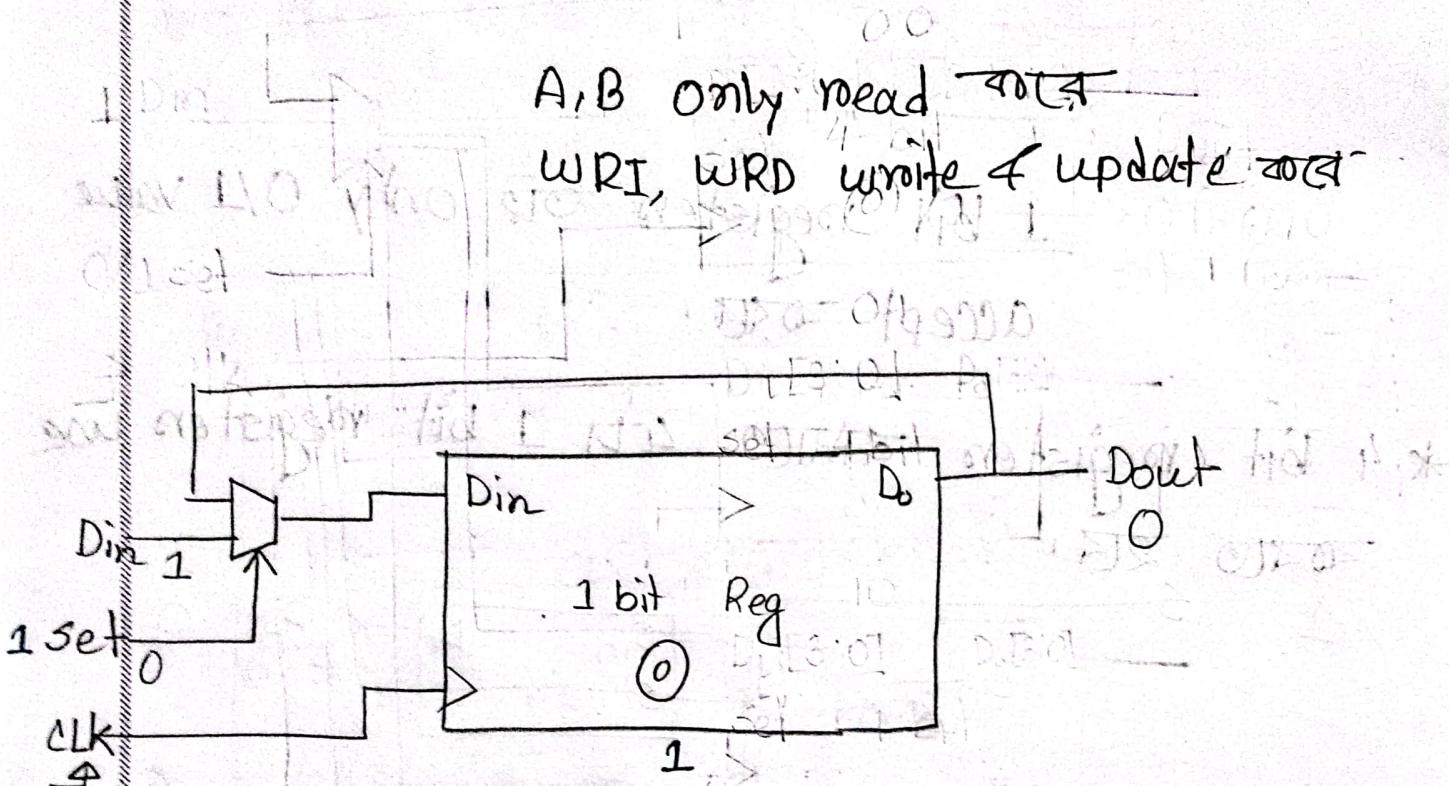


Sat  Sun  Mon  Tue  Wed  Thu  Fri

Sub:

Date: 20 / 11 / 2023

## 4 bit Registerset with 4 registers



A, B only read

WRI, WRD write & update

set = 0  $\Rightarrow$  Dout = 0 when Din = 1

(selection = 0)

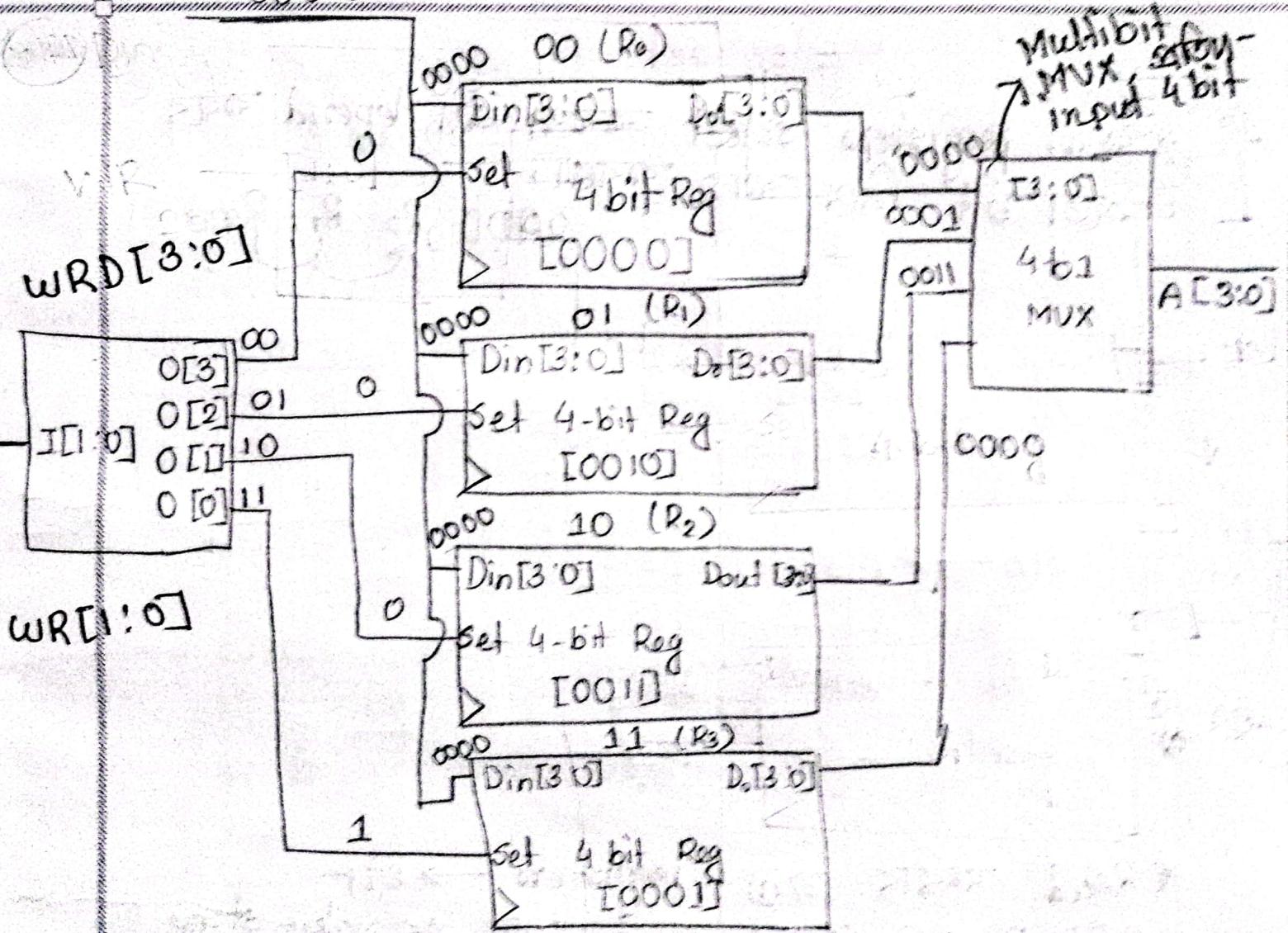
Din = 1 Set = 1, save = 1 Dout = 1, clk = (+)ve

Sketch from slide

$$\log_2(4) = 2 \text{ bit}$$

$$\log_2(8) = 3 \text{ bit}$$

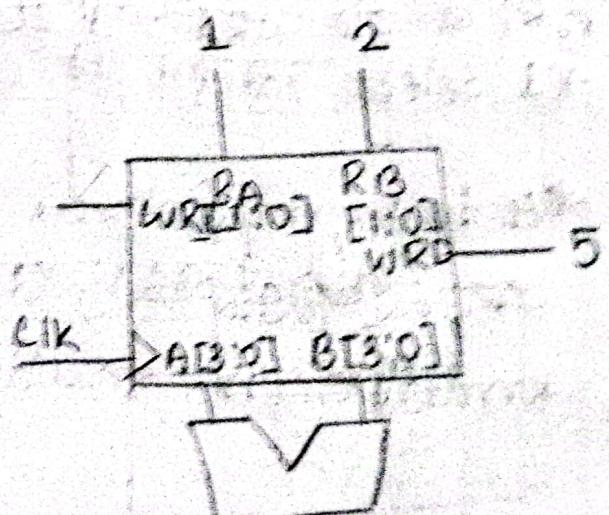
Sub: \_\_\_\_\_ Date: / /



set হলো decoders এর

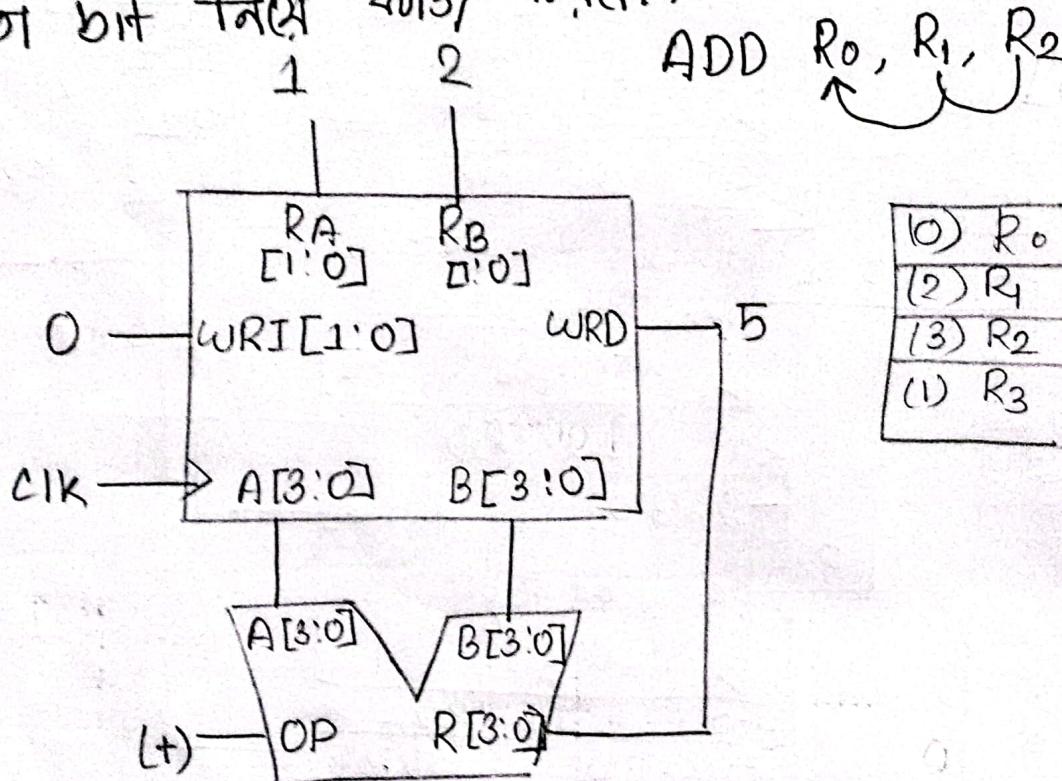
line select করবে

সেপার উপর depend করবে



## Register Set

\* কয়েটা registers select কৰবো এমনি depend কৰবে  
কঠো bit নিয়ে বলত কৰবো,



\* Read কৰাব তাহা register  $\rightarrow 2^2$

\* ADD  $R_1, R_2$  হওয়া সাথে WRI এর value store

$R_1$  এ ও  $WRI = 1$  হবে

\* 1 select কৰলে A টো চলে আসব।

\* 4 to 1 multiplexers দিয়ে এই output selection  
এবং বলত কৰা হবে।

\* multiplexer multibit হাব

Sub:

Sat Sun Mon Tue Wed Thu Fri  
Date: / /

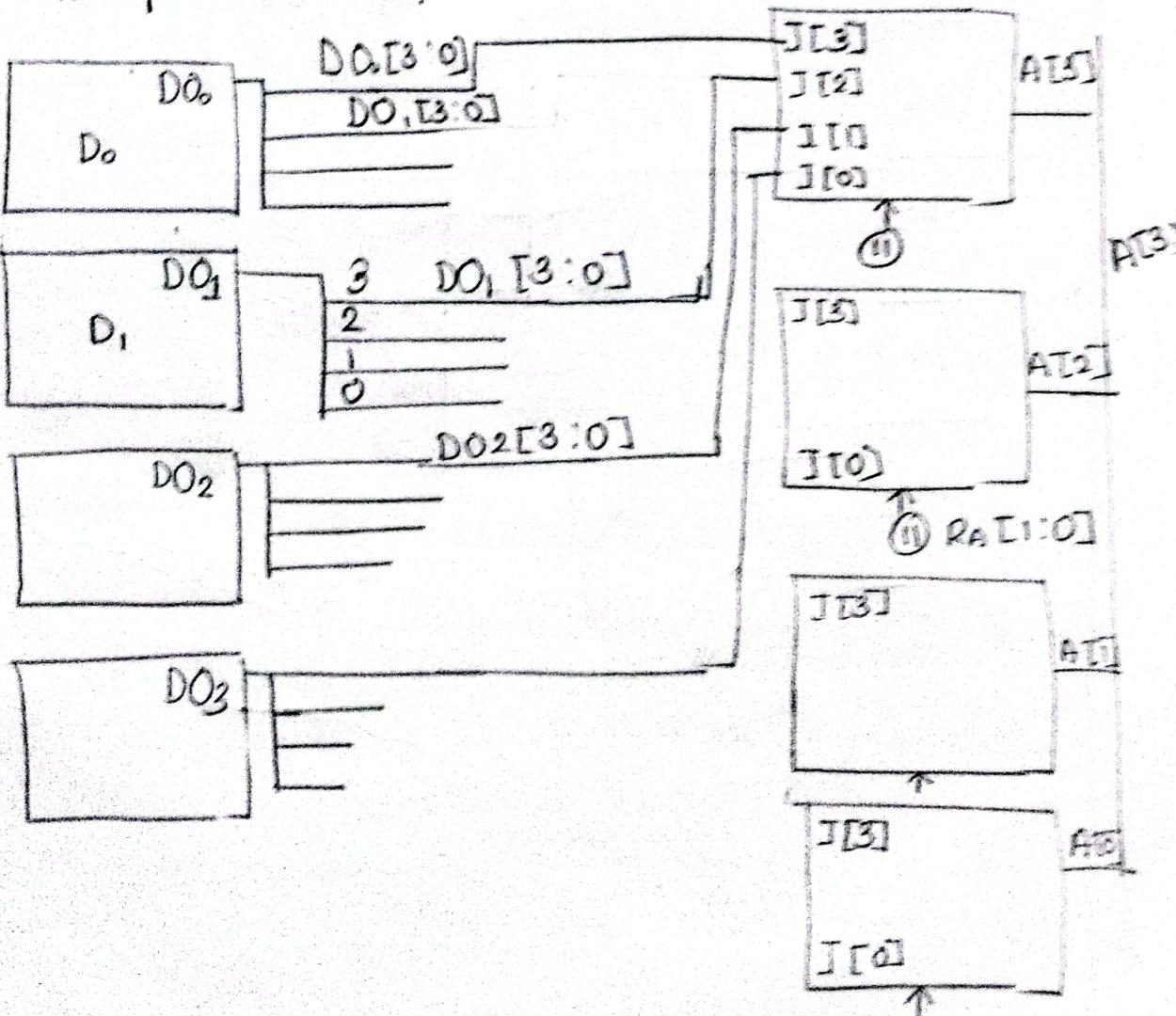
\* 4 bit multiplexers 4'ଟି କେବଳ ଏହା ଯାତ୍ରା ପାଇଁ, ତା  
1 bit multiplexers ହୁଏ ।

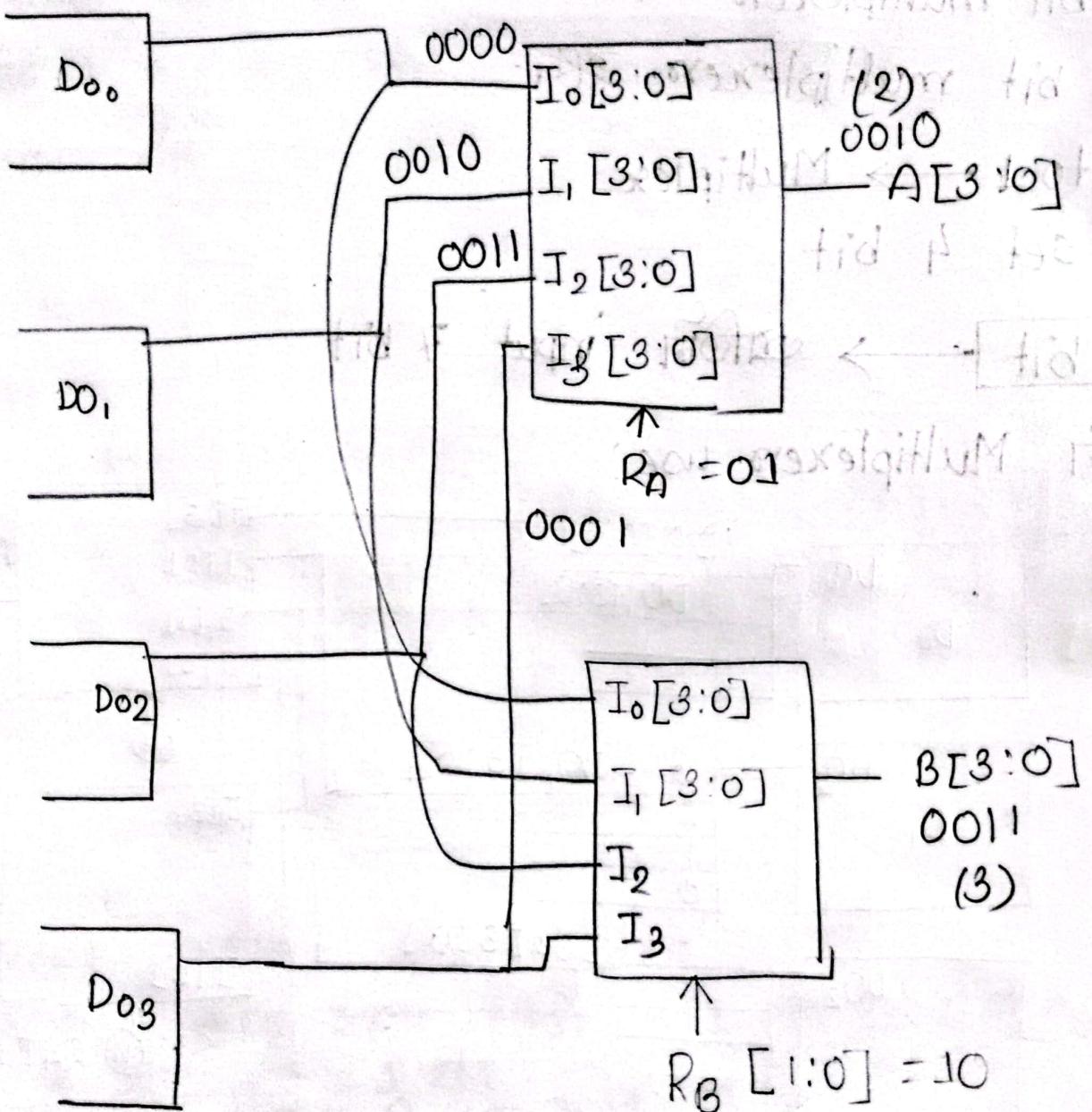
16 to 1 → Multiplex

set 4 bit

7 bit → ଏହିଟି ଅନ୍ତିମ input 7 bit

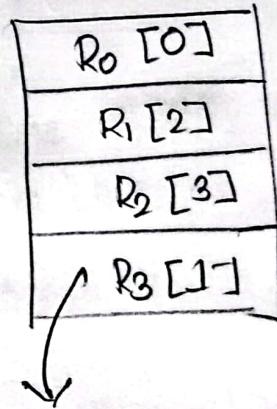
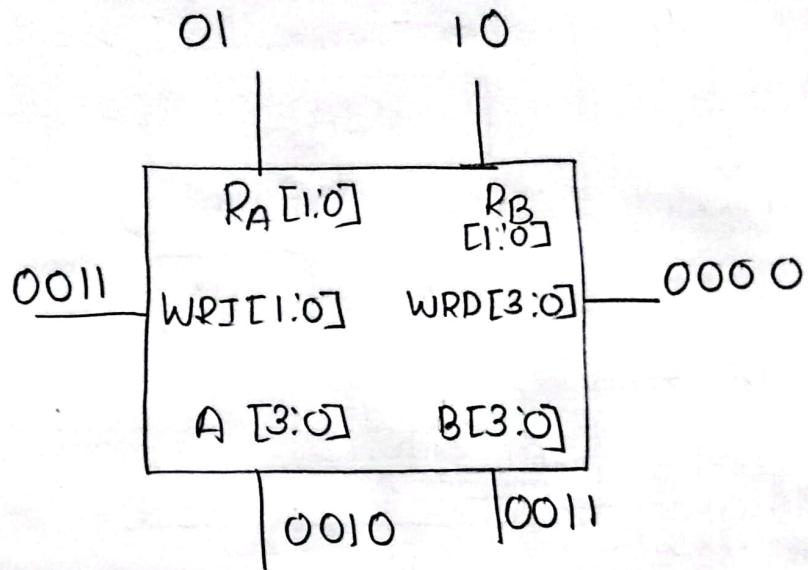
4'ଟି Multiplexers use:





Sat    Sun    Mon    Tue    Wed    Thu    Fri  
 Date :   /   /

Sub: \_\_\_\_\_



এখন  $R = 0$  save করতে  
কৈ হো?

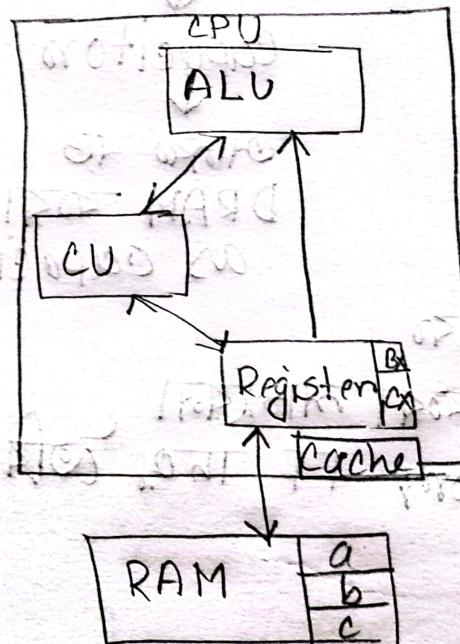
Sub: \_\_\_\_\_

Sat Sun Mon Tue Wed Thu Fri

Date: 21 / 11 / 2023

RAM

## Von Neuman Architecture

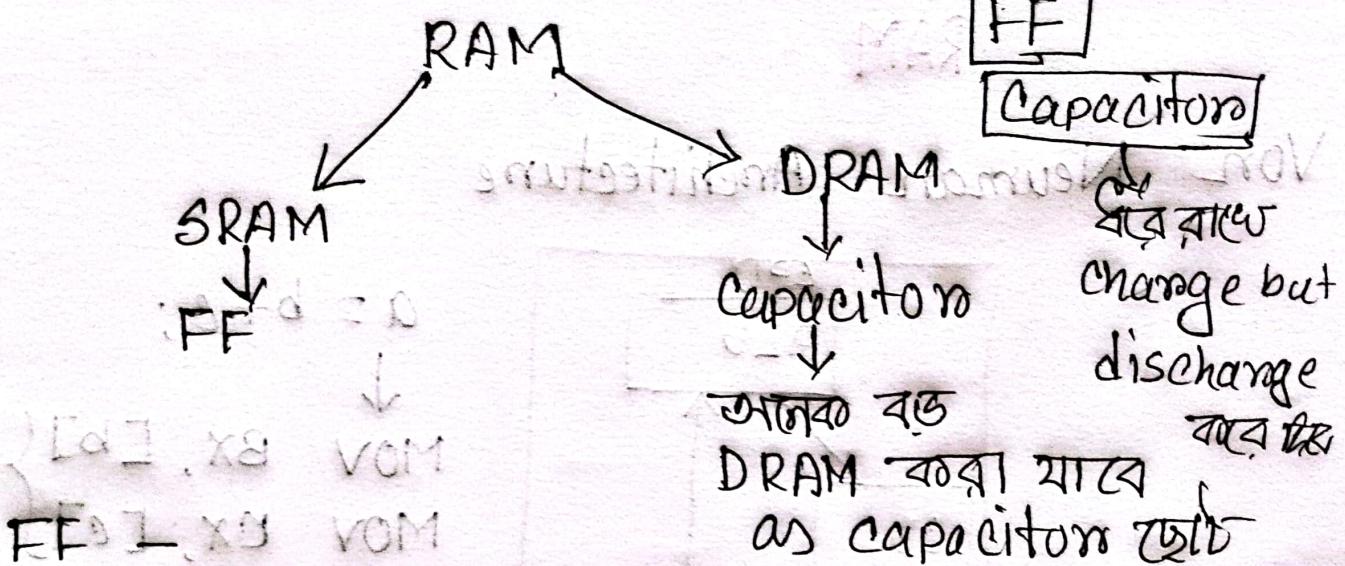


$a = b + c;$   
↓  
MOV BX, [b] { Register  
MOV CX, [c].  
ADD BX, CX → ALU  
MOV [a], BX → RAM  
→ 10MB বা কম চাইলে

- \* মেমোরি variable declare করি তখন RAM এ রয়ে  
operation চালান্তে registers এ মাঝে
- \* Registers ছুব হচ্ছে এজন data main memory  
তে রাখে,
- \* মেমোরি data RAM এই থাকবে,
- \* Randomly access করে RAM. আর time fix  
খেঙানো access করবা থেকে না কোন,

Sub:

Computer Architecture



\* Faster অনেক

\* Recharge করা লাগবেনা

\* Cache memory FF টিক্ক তৈরি,

Capacitors -

\* চোরে ছো

\* use করা more feasible ব্যবহৃত ছো

\* 100 instruction কে memory টে store

কমতে  $\log_2(100) \approx 7$  bit লাগবে।

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

000-0000

প্রতি column Representation  
13 bits as ISA 13 bit.

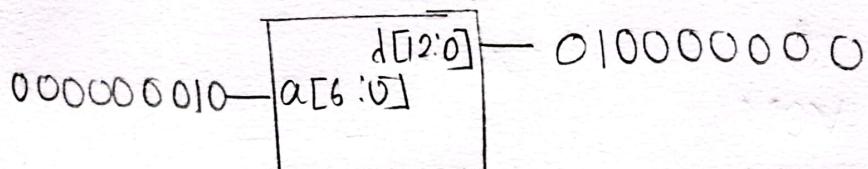
111-1111

Sub: \_\_\_\_\_

Sat	Sun	Mon	Tue	Wed	Thu	Fri
<input type="radio"/>						

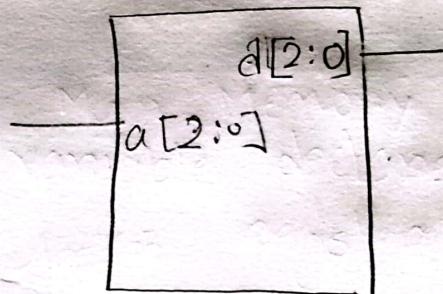
Date: / /

\* এই instruction এ লিখি আসা machine code এ convert করতে হবে, ultimately এই machine code দ্বারা RAM এ save কোরা করতে হবে,



single port SRAM

\* এখানে write করার option নাই, write করতে চাইলে extra একটা port লাগবে,



RAM এর size  $2 \times 2 = 4$

RAM chip  $4 \times 2$ ,  $\rightarrow$

\* write করার ক্ষমতা যদি এই write enable না মানু।

Sub:

Date: / /

1x1 SRAM%

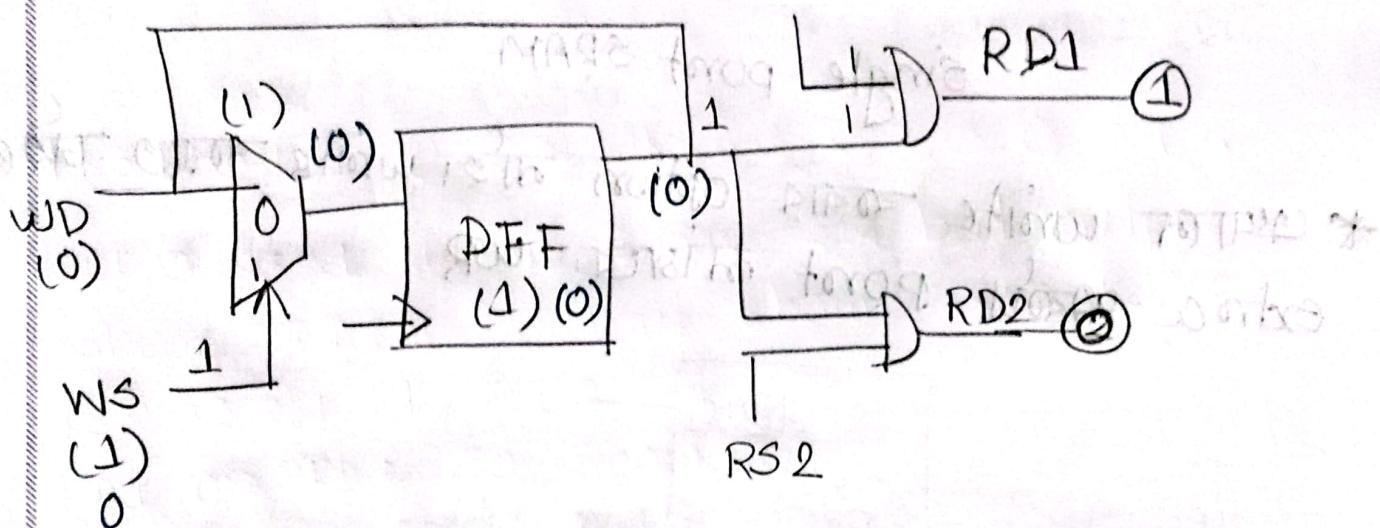
Column = 1

 $\boxed{1 \times 1} \rightarrow$  data 1 DT2

Row = 1

So FF needed = 1

RS1 = 1



Q: convert Assembly code to machine code?

RA, RB  $\rightarrow$  data read করবে

A, B  $\rightarrow$  RA, RB র value

WRD  $\rightarrow$  data output / write এর value

<input checked="" type="checkbox"/>	Sat	<input type="checkbox"/>	Sun	<input type="checkbox"/>	Mon	<input type="checkbox"/>	Tue	<input type="checkbox"/>	Wed	<input type="checkbox"/>	Thu	<input type="checkbox"/>	Fri
-------------------------------------	-----	--------------------------	-----	--------------------------	-----	--------------------------	-----	--------------------------	-----	--------------------------	-----	--------------------------	-----

Date: / /

Sub: \_\_\_\_\_

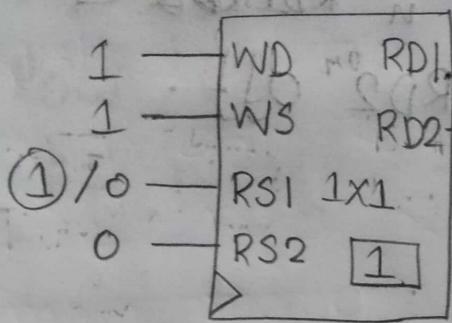
2<sup>nd</sup> read operation

$RS_1 = 1, RS_2 = 1$  point of read  
 $RS_2 = 0, RS_1 = 1$  RD1 RD2 & read only  
RD2 off.

Sub:-

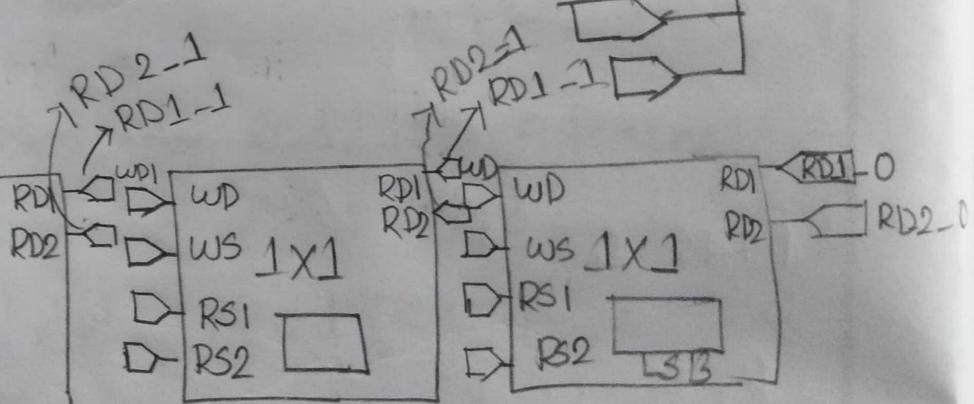
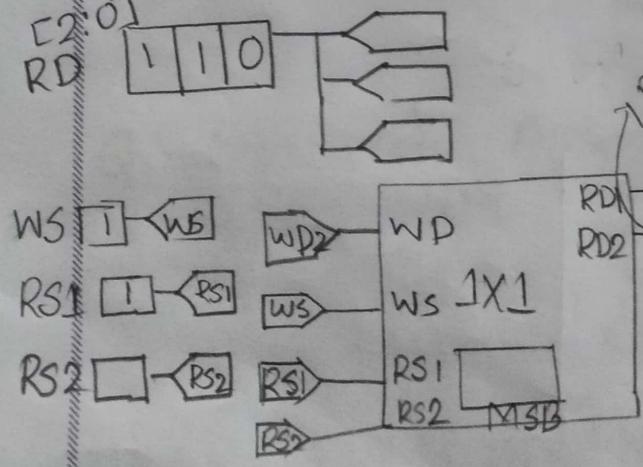
বাক্স

1x1 SRAM



\* RS1 ও RS2 ০ হলে RD1 ও RD2 ০ হবে।

\* WD ও WS ১ ফলে ১ save হবে but output show করবেনা। output show ব্যবহৃত RS1 on RS2 একটা ১ ব্যবহৃত হবে মেমোর respect আ output RD1 on RD2 তে show করবে।



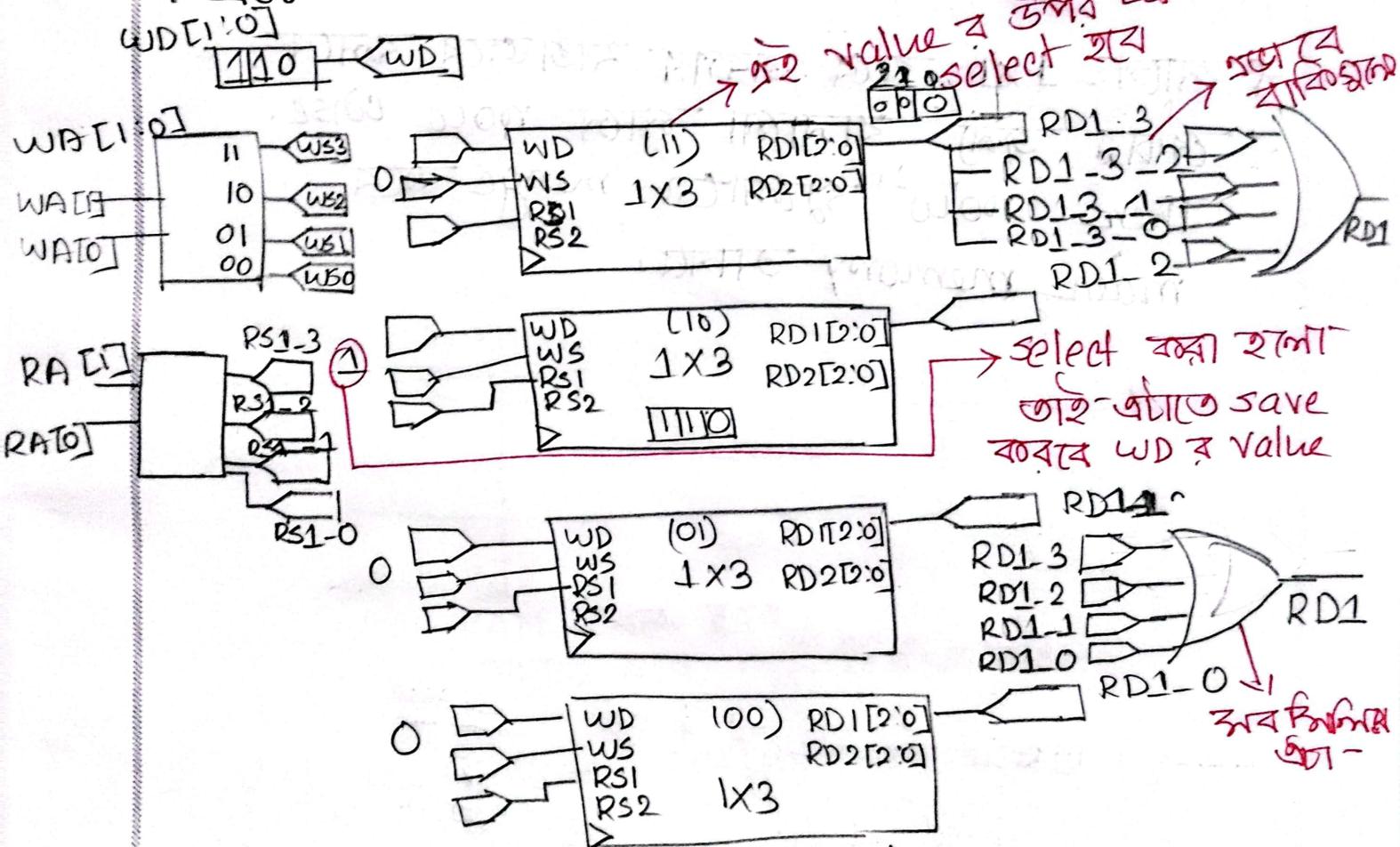
*Sub:-*

<input checked="" type="checkbox"/> Sat	<input type="checkbox"/> Sun	<input type="checkbox"/> Mon	<input type="checkbox"/> Tue	<input type="checkbox"/> Wed	<input type="checkbox"/> Thu	<input type="checkbox"/> Fri
---	------------------------------	------------------------------	------------------------------	------------------------------	------------------------------	------------------------------

Date :      /      /

4x3 static ROM

\* প্রত্যেকটি 3 bit বাটে save করে। বাটা depend করে



\* যদিগুলোই 3 bit বাটৰ যাবে, যেপে select কৰো only  
সেপ প্রাইট কৰবে।

\* Address বলে দেওয়া হবে, যাতা select করবো এবং

\* 2to4 decoders use 'enable' or select बटन का इष्ट।  
उदाहरण

Sub:

✓ Sat Sun Mon Tue Wed Thu Fri

Date: / /

\* depending on the value RD কেন্টা  
select করবো

\*আগত  $1 \times 1$  বারে তারপর যত্থাইবে বলবে  
মেমর ডিজি বানানো গানে now wise.  
Then now পুলোকে merge করবে  
main memory আসাব।

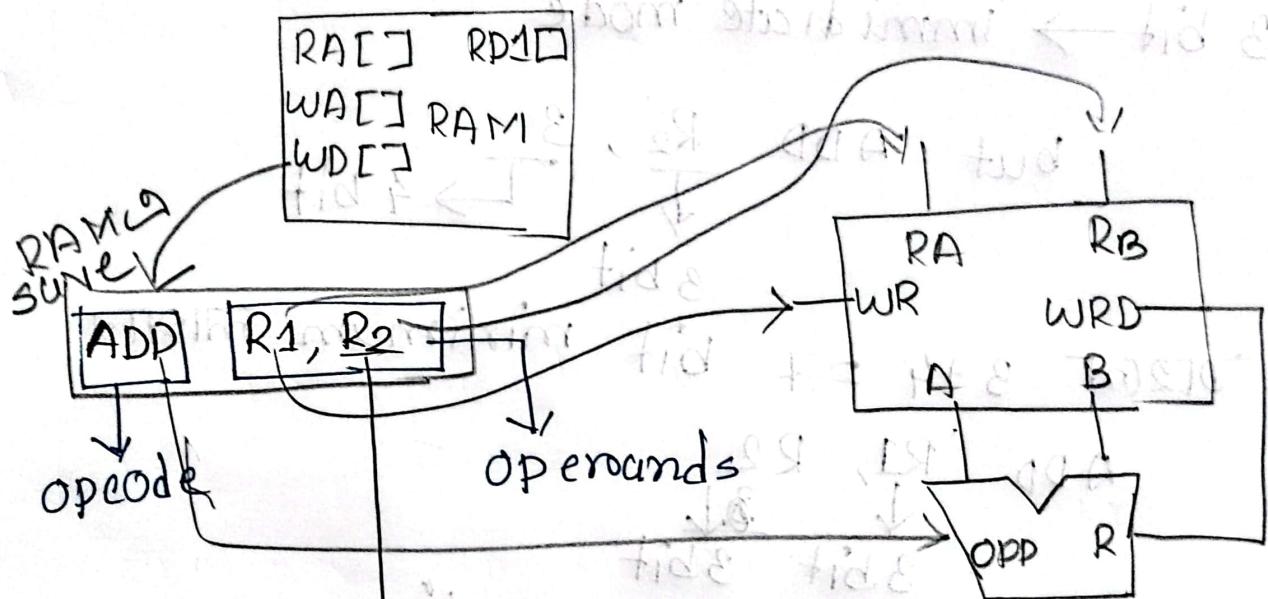
Sub: \_\_\_\_\_

✓ Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○

Date: / /

## ISA

\* Machine code mainly



1. 4st of opcode design করতে হবে,

Opcode 4 bit → For ALU operation  
2 bit → For 4 instructions

minimum 6 bit লাগবে -

6 bit → Opcode

## 2. For operand design

3 bit  $\rightarrow$  register 8 bit

3 bit  $\rightarrow$  immediate mode

but ADD  $\frac{R_2, 3}{\downarrow} \rightarrow 4 \text{ bit}$   
3 bit

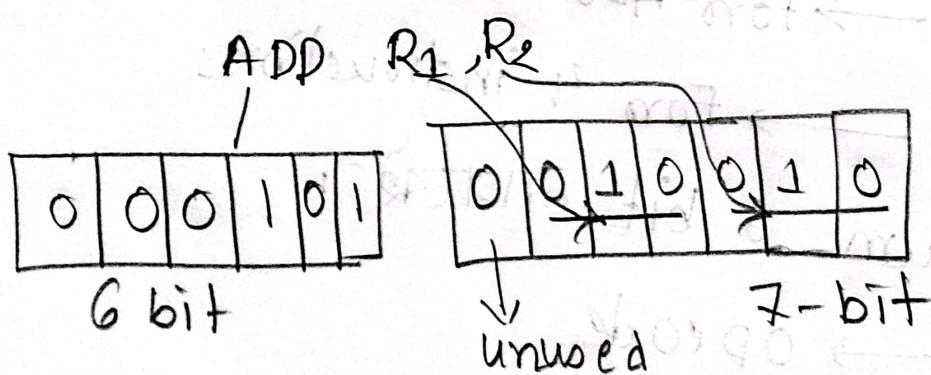
তাহলে  $3+4 = 7$  bit minimum লাগবে

ADD  $R_1, R_2$   
 $\downarrow$   
3 bit 3 bit

6 bit  $\rightarrow$  আজ 1 bit  $\xrightarrow{\text{কোরা থাকবে}}$

তাহলে total 6 bit + 7 bit = 13 bit

এটা যথেষ্ট লাগবে।



\* unused bit first/ Last bit depends on design