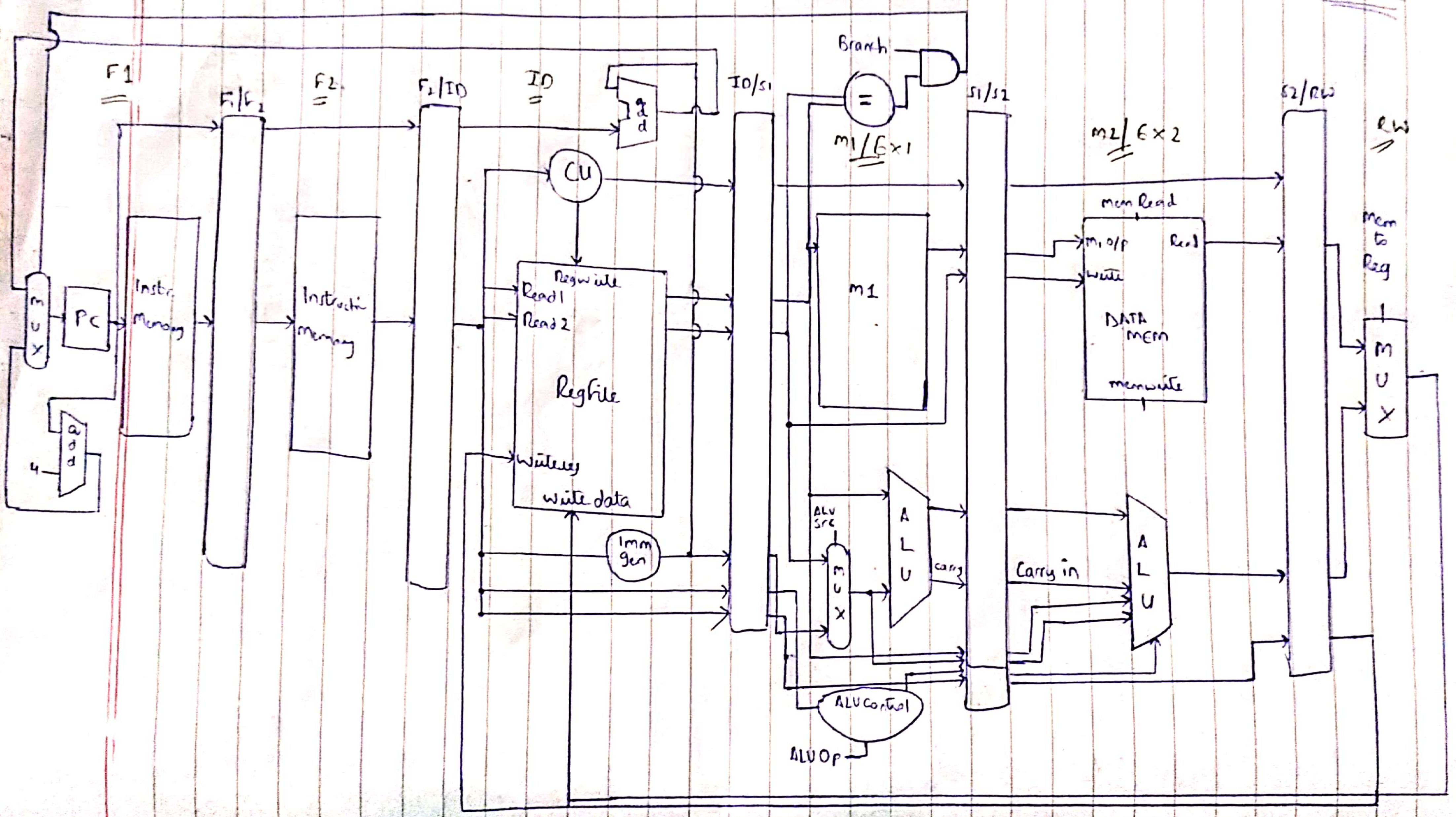
Q1)

CS-102

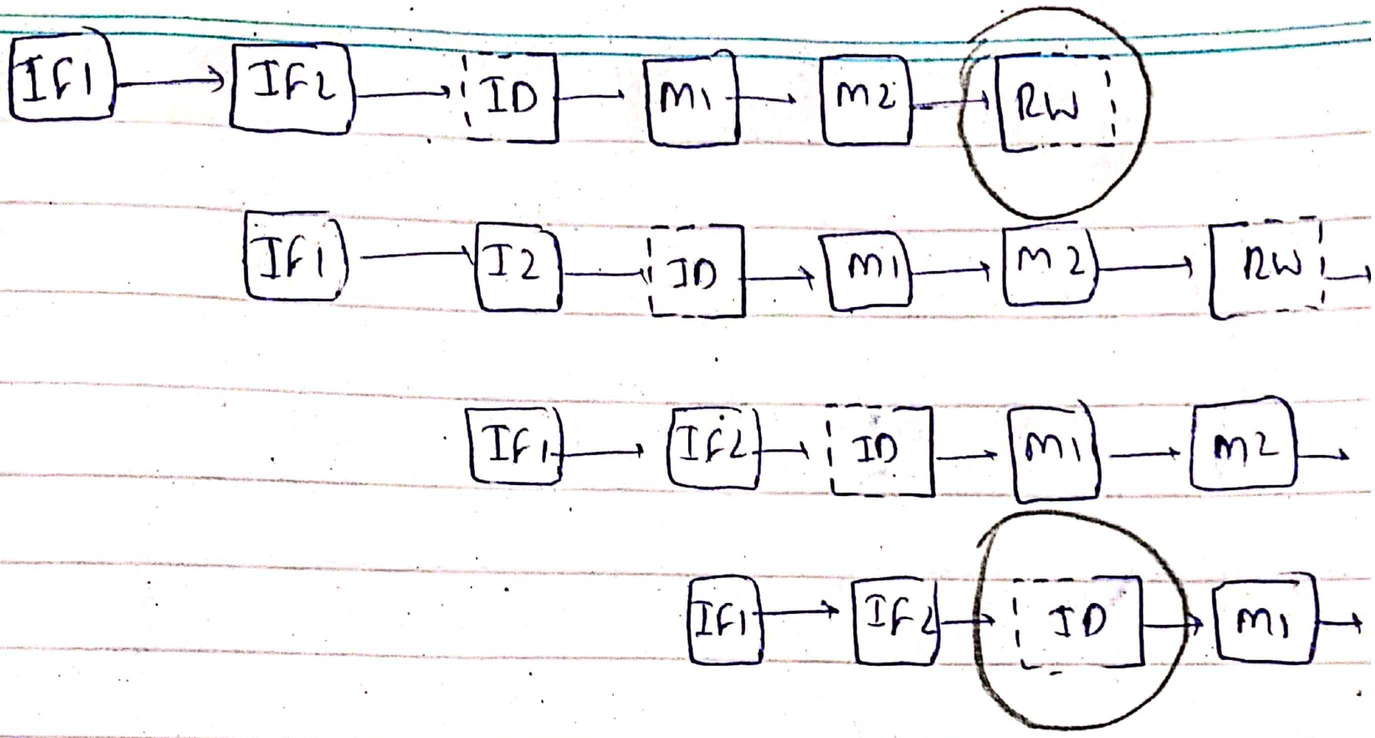


F1
F1/F2
F2
F2/ID
ID
ID/S1
S1/S2
S2/RW
RW
M1/EX1
M2/EX2
Branch
=
add
4
add
MUX
PC
Instr Memory
Instruction Memory
CU
Regwrite
Read1
Read2
Regfile
Write reg
write data
Imm gen
M1
ALU src
MUX
ALU
carry
ALU control
ALU Op
Carry in
ALU
mem Read
M1 o/p
Read
write
DATA MEM
memwrite
Mem to Reg
MUX

QUESTION 2 CS-102

Let us assume that IF1 always fetches instruction and IF2 always fetches operands. Also assume that M1 and M2 are always have their own memory systems. So IF1, IF2, M1 and M2 overlapping will not cause any structural hazard, bcoz instruction memory and data memory are different.

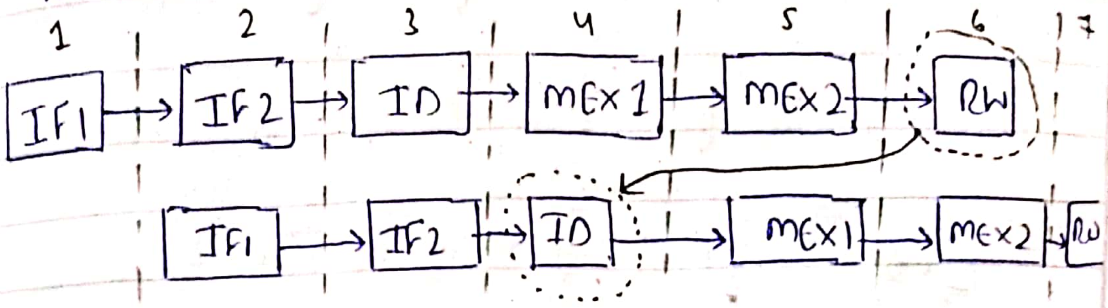
The only hazard that may occur is on RW and ID as they both requires Register access. However this hazard can be solved easily. Since RW only writes the register and ID only reads the register therefore ensuring that ~~RW performs~~ register performs write only function in upper half cycle (for RW) and read only in lower half cycle (for ID).


IF1
IF2
ID
M1
M2
RW
IF1
I2
ID
M1
M2
RW
IF1
IF2
ID
M1
M2
IF1
IF2
ID
M1

(Q3) QUESTION 3
Let us take 2 instructions ;

ADD x1 , x2 , x3
SUB x3 , x1 , x5

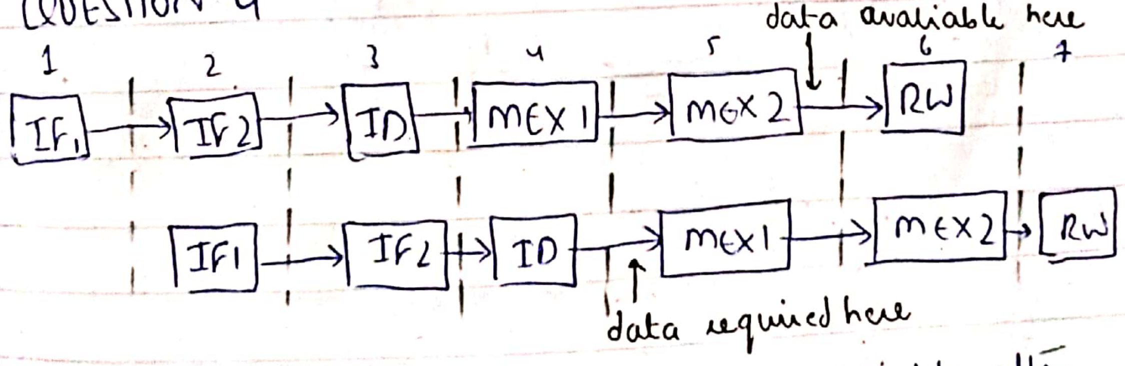
a data hazard occurs X1 is not loaded yet when the instruction 2 is decoded.



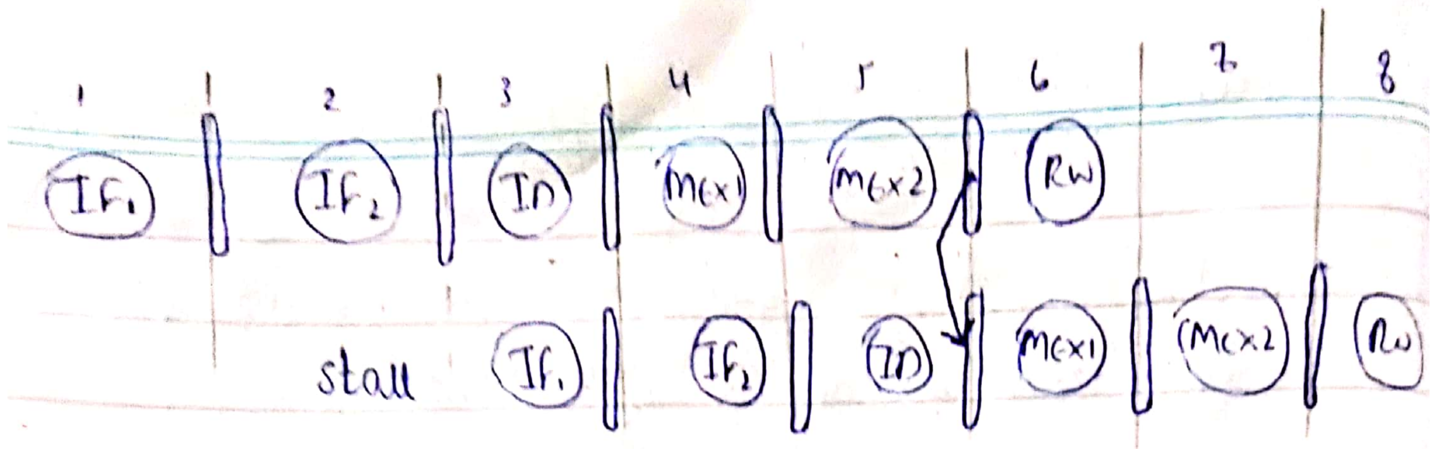
The value of X1 is required at the cycle number 4 , however the X1 will not be avaliable until 6th cycle. This results in data hazard.

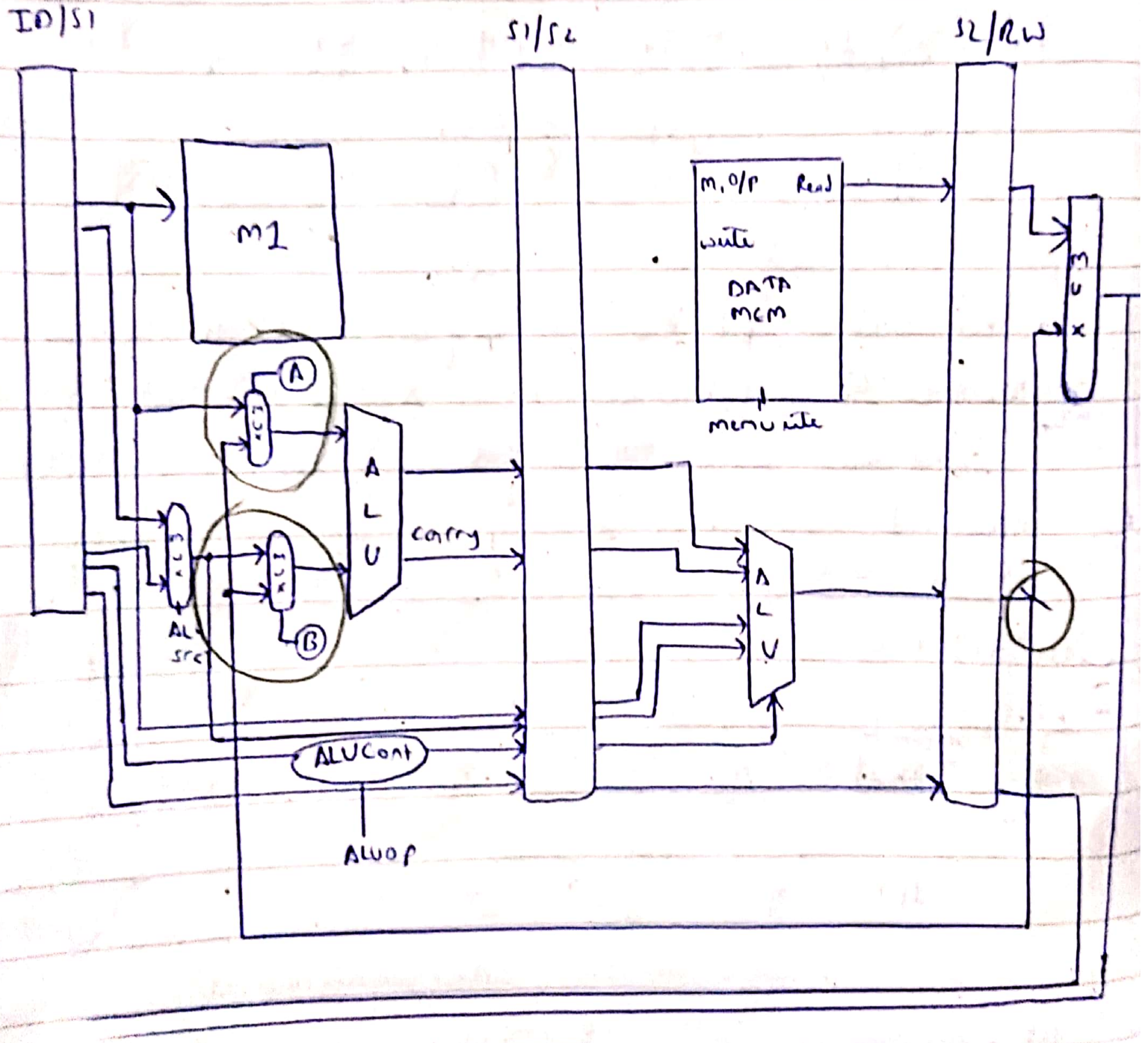
QUESTION 4

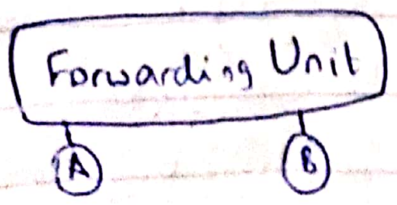
CS-102



at maximum extent the data is avaliable after the end of 5th cycle as soon as possible and will be required at the beginning of 5th cycle. We can stall the secondary instruction by 1 cycle then use a forwarding unit.


1
2
3
4
5
6
7
8
IF1
IF2
ID
Mex1
Mex2
RW
stall
IF1
IF2
ID
Mex1
Mex2
Rw


ID/S1
S1/S2
S2/RW
M1
A
B
MUX
MUX
MUX
ALU
ALU
AL src
carry
ALUCont
ALUOP
M.O/P
Read
write
DATA MEM
memwrite
MUX


Forwarding Unit
A
B

cond1. if (S2|RW. Reg-rd == ID|S1. Reg-rs1)
then forward A = 1

cond2 if (S2|RW. Reg-rd == ID|S1: Reg rs2)
then forward B = 1

Example.

| | | rd | rs1 | rs2 | |
|---|---|---|---|---|---|
| 1) | ADD | x1, | x2 | , x3 | => forward A = 1 |
| | SUB | x4, | x1 | , x5 | |

| | | rd | rs1 | rs2 | |
|---|---|---|---|---|---|
| 2) | ADD | x1, | x2, | x3 | => forward B = 1. |
| | SUB | x4, | x5 | , x1 | |

| | | rd | rs1 | rs2 | |
|---|---|---|---|---|---|
| 3) | LW · | x1, | x2 | | => forward A = 1 |
| | ADD | x3, | x1 | , x5 | |

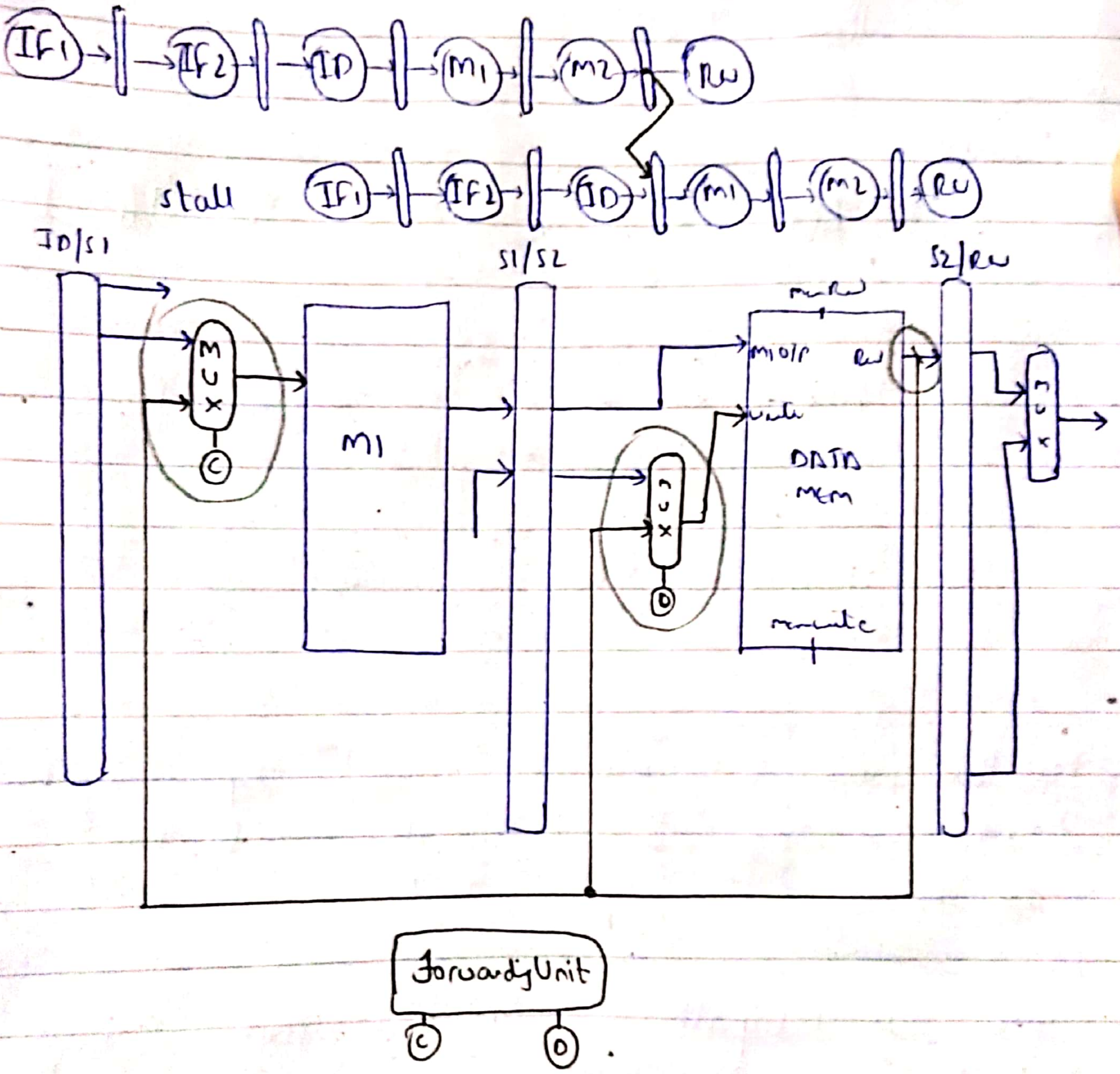
Yes this forwarding unit also solve the dependency b/w load and any other arithmatic instruction as shown in example no. (3).

Since, our datapath is designed in such a way That EX2 and M2 is avaliable at same clk cycle therefore correcting for execution (ALU) instructions will also correct for memory instructions.

QUESTION 5 CS-102

for e.g:

| | rd | rs1 |
|---|---|---|
| Load | x6, | x5 |
| Store | x6, | x7 |



after 1 stall cycle if following conditions are met then:

if (S2/RW.Reg_rd == ID/S1.Reg_rs1 && ID/S1.Reg_rs2==

then forward A = 1

forward B = 1

Scanned with CamScanner

QUESTION 6 CS-102

fom Q4 and Q5 we saw that 1 stall cycle is needed even with forwarding units, therefore we design a HDU that detects hazard and then insert a stall.

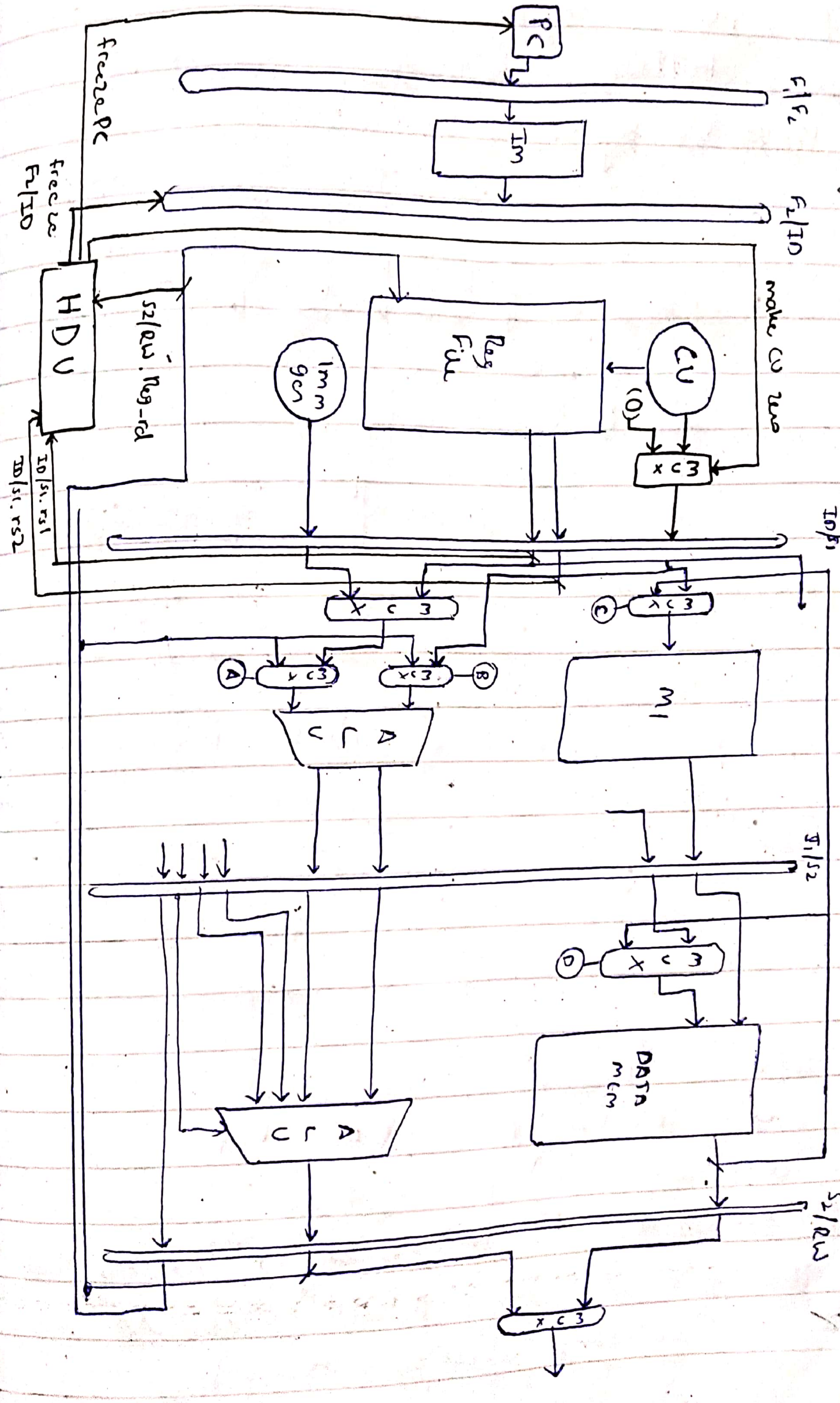
Following conditions are required;

if { [(S2|RW.Reg-rd == ID|S1.Reg-rs1) || (S2|RW.Reg-rd == ID|S1.Reg-rs2) || (S2|RW.Reg-rd == ID|S1.Reg-rs1 && ID|S1.Reg-rs2 == 0) ~~|| (S2|RW.Reg-rd == (ID|S1.Reg-rs1 || ID|S1.Reg-rs2)) && (secondary instruction is branch~~)) }

then stall.

QUESTION 6 - Diagram

CS-102



1) Freeze PC
2) Freeze $F_2$/ID
3) Make CV signals zero.

## QUESTION 7

A/c to given condition;
3 cycles for BTA calculation and 1 cycle to calculate branch condition.

we will insert stall until branch condition is known

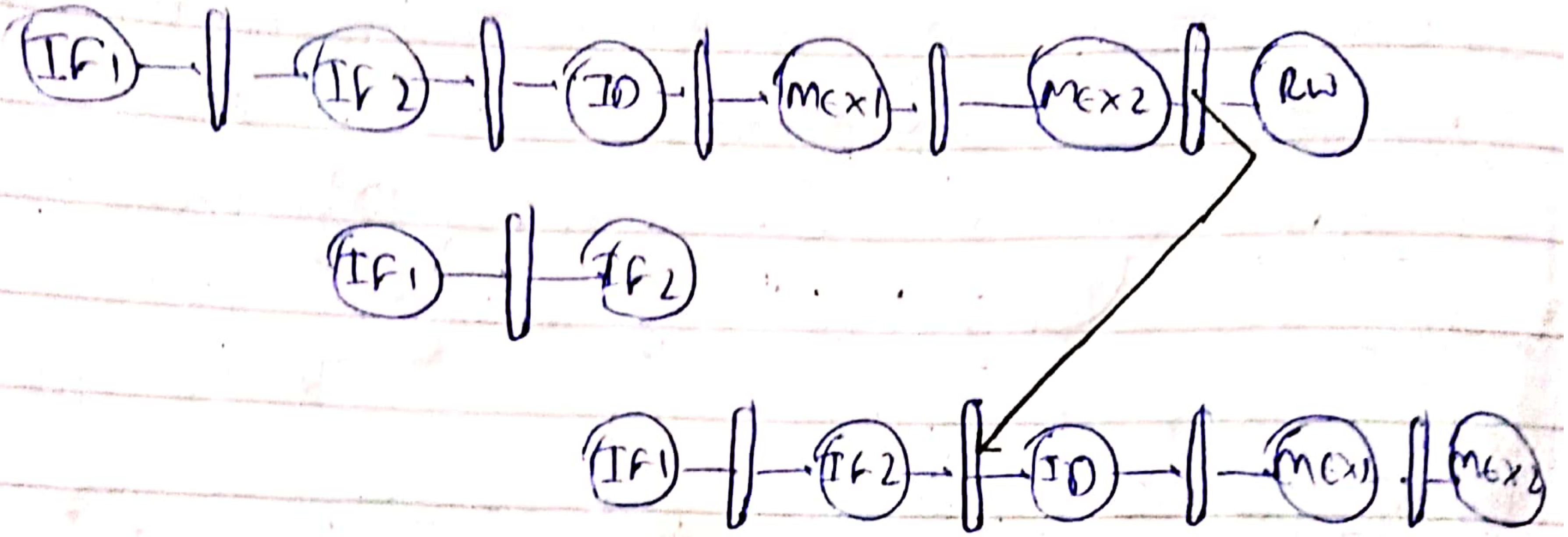
our branch would be confirmed after EX1 stage therefore we may start the next instruction at EX2 stage. So $\underline{\underline{3}}$ stall cycles are needed.

| | | | | | | |
|---|---|---|---|---|---|---|
| IF1 → | IF2 → | ID → | M/EX1 → | M/EX2 → RW | | |
| | stall | stall | stall | IF1 → | IF2 → | ID → M/Ex1 |

## QUESTION 8

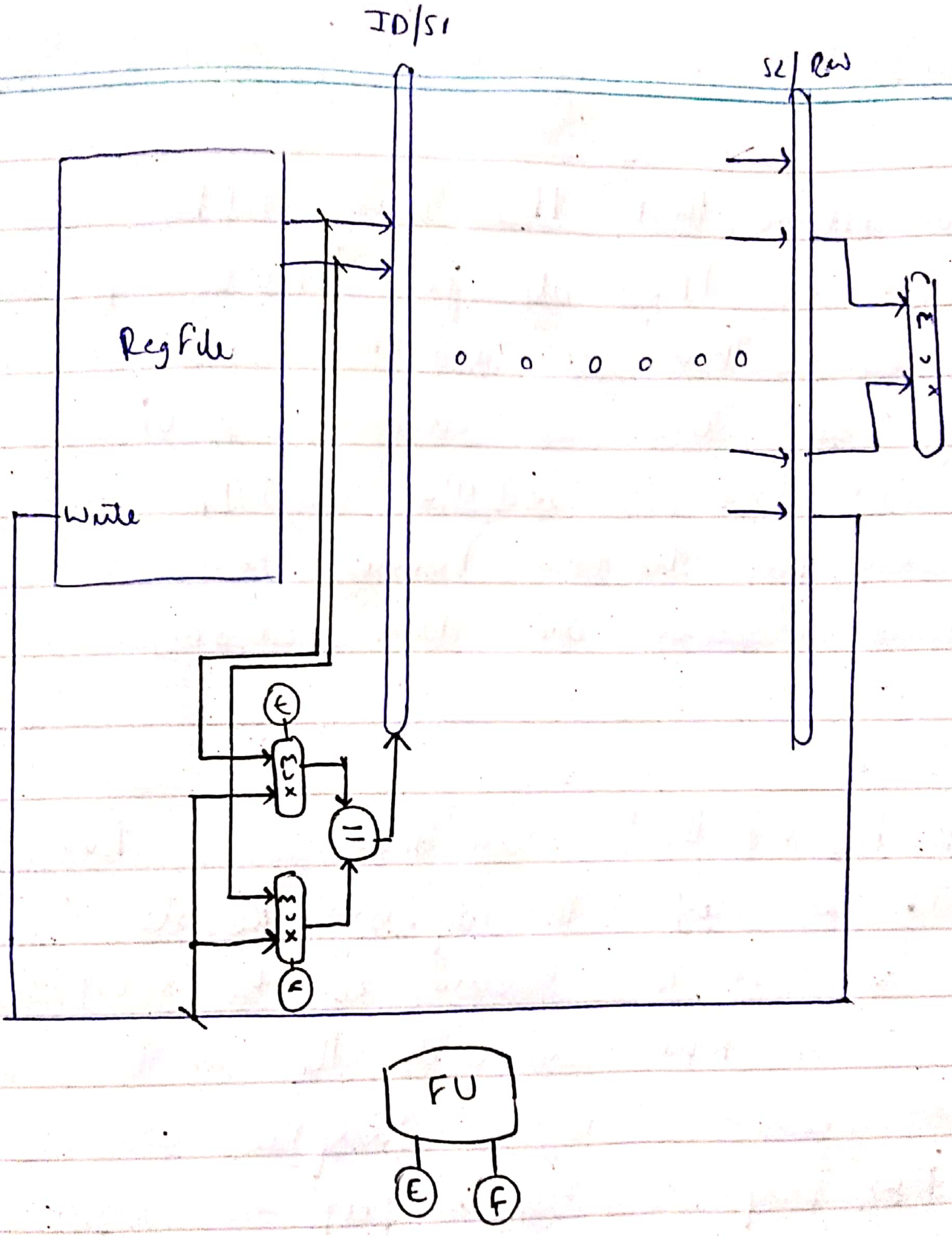
Comparator moved to ID stage therefore new data hazards arise.
Now both BTA and BC evaluated in ID stage

e.g. ADD x1, x2, x3
any instruction
BEQ x1, x5, Label



requires one stall

The new condition is ;
if (S2|RW.Reg_rd == (ID|S1.Reg_rs1||ID|S1.Reg_rs2))
&& secondary instr. is branch)
then stall.


ID/S1
S2/RW
Reg File
Write
MUX
MUX
MUX
=
E
F
FU
E
F

The HDU will become;

if {(~~S2|Reg~~ S2|RW. Reg_rd == ID|S1. Reg_rs1) ||
(S2|RW. Reg_rd == ID|S1. Reg_rs2) ||
(S2|RW. Reg_rd == ID|S1. Reg.rs1 && ID|S1. Reg_rs2} == 0) ||
((S2|RW. Reg_rd == (ID|S1. Reg_rs1 || ID|S1. Reg.rs2)) &&
secondary instruction is branch) }

Then stall