pbp2022
N10102389

Q.3)

1) Instruction memory is read, 2 registers are read & one is written.
Energy consumed for addi = 140pj + 2* 70pj + 60pj
= 340pj

2) Instruction memory is read, 2 registers are read & data memory is written onto.
Energy consumed for ld = 140pj + 2*70 pj + 120pJ
= 400pJ.

3) Instruction memory is read, 2 registers are read
Energ consumed for beq = 140pJ + 2* 70pJ
= 280pJ

Q.4) 1) Data forwarding alone does not suffice as by the time the first ALU instruction already ~~computes/executes~~ completes the second step, the second ALU instruction is already there in first ALU step. Thus, its too late.

Hence, NOP between 2 instructions as well as the forwarding between second ALU step & deco step is required.

pbp2022
N10102389

Q.4 2)

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
add x1, x2, x3	IF	ID	ALU1	ALU2	MEM	WB			
NOP									
add x5, x4, x1			IF	ID	ALU1	ALU2	MEM	WB	

Q.1) i) Code for g>h :

```
ble x5, x6, Else // goto else if g ≤ h
addi x5, x5, 1   // g = g+1
beq x0, x0, Exit  // if 0=0 goto Exit
Else: addi x6, x6, -1  // h= h-1
Exit.
```

ii) Code for g<= h

```
blt x6, x5, else // goto else if h<g
sub x5, x5, x5 // g = g -g = 0
beq x0, x0, Exit // if 0=0 goto exit
Else : sub x6, x6, x6  // h= h-h = 0
Exit.
```

pbp2022
N10102389

Q.7) i)

```
or   x13, x12, x11
ld   x10, 0(x13)        // EX to 1st RAW Hazard
ld   x11, 8(x13)        // EX to 2nd RAW Hazard
add  x12, x10, x11      // MEM to 1st RAW && MEM to 2nd RAW
subi x13, x12, 16       // EX to 1st RAW Hazard
```

```
or   x13, x12, x11
NOP
NOP
ld   x10, 0(x13)
ld   x11, 8(x13)
NOP
NOP
add  x12, x10, x11
NOP
NOP
subi x13, x12, 16.
```

pbp2022
N10102389

Q.7 2)

	1	2	3	4	5	6	7	8	9	10
or	IF	ID	EX	MEM	WB					
ld		IF	ID	EX	MEM	WB				
ld			IF	ID	EX	MEM	WB			
NOP										
add					IF	ID	EX	MEM	WB	
subi						IF	ID	EX	MEM	WB

→ Mandatory NOP for which no forwarding solution possible. Load = data + use.

1) A = x   B = x   // no instruction in EX stage yet.

2) A = x   B = x   // no instruction in EX stage yet.

3) A = 0   B = 0   // both operands of 'or' x11 & x12 come from register file.

4) A = 2   B = 0   // Base ~~address~~ RS1 of first ld x13 taken from MEM of previous instruction

5) A = 1   B = 0   // Base RS1 of second ld x13 taken from MEM of previous instruction

6) A = x   B = x   // no instruction in EX yet coz of NOP

7) A = 0   B = 1   // RS2 in add instruction is x11 which is

pbp2022
N10102389

Q.1 2) CONTD....

forwarded from MEM of 2nd ld, the result of 1st ld has already been written into reg file in CC6 so no forwarding for 1st operand

8) A=1 B=0 // RS1 of sub forwarded from EX/MEM of add.

Q.2) Consider following example:

a= 3 , b= 5
We need to swap values
a= a+b = 3+5 = 8.
b= a-b = 8-5 = 3
a = a-b = 8-3 = 5.
Thus, a= 5 && b=3.

Code is :
a= a+b
b= a-b
a= a-b.
Say a's value is in x5 && b's value in x6.

M10102389

Q.2) RISC V code-

```
add x5, x5, x6     // x5 = x5 + x6.     a = a+b
sub x6, x5, x6     // x6 = x5 - x6.     b = a-b
sub x5, x5, x6     // x5 = x5 - x6      a = a-b.
```

Pbp

N1010

Q.6)

XOR s1, s2, s3	IF	ID	EX	MEM	WB				
addi s0, s3, -4		IF	ID	EX	MEM	WB			
lw s3, 16(s7)			IF	ID	EX	MEM	WB		
sw s4, 20(s1)				IF	ID	EX	MEM	WB	
or t2, s0, s1					IF	ID	EX	MEM	WB
					↳ 5th cycle				

Result of XOR is written back to register ~~$~~ s1 in

At stage 5, XOR is in write back stage,
At stage 5, addi is in memory stage,
At stage 5, lw is in execute (ALU) stage.
At stage 5, sw is in decode stage.
At stage 5, or is in fetch stage.
Hence, only s1 is written in 5th cycle.

pbp2022
N10102389

Q.8)

	Instruct/Prog	CPI	Circuit
Replace the 2 op-erand ALU with a 3 operand one & add 3 operand register-register instructions to ISA (for ex: add rs1, rs2, rs3, rd	Decrease New instruction will replace any 2 instruction seq that achieved same such as, add rs1, rs2, rd add rs3, rd, rd	Same ALU will perform 3-way addition in one cycle. Acceptable increase because more RAW	Increase 3 operand ALU is more complex than 2 operand.
Use same ALU for instructions & for incrementing PC by 4	Same No difference to instructions	Increase All instruction use same ALU. ALU operations have to stall	decrease There is one less adder / ALU cycle.
Increase the number of user registers from 32 to 64.	The same or decrease If more register enable compiler to avoid loads & stores then decrease otherwise same.	Same does not affect actions of each instruction	decrease More register complicate register file.

pbp2022
N10102389

Q.5) $4 + (3+4+3) \times 5 + 3 = 57$ clock cycles

No. of instructions exectud $= 1 + (3 \times 5) + 1 = 17$

↳ 'if' runs 5 times

CPI $= 57/17 = 3.35$ CPI.

(CPI = Clock cycles/ no. of instruction)

1st instr

Last instr

for a 5 stage pipeline $= 5 + (17-1) = 21$

$\therefore$ CPI $= 21/17 = 1.23$