Asawari Badkundri
ab10445

Q1) (i)
```
bge   X6, X5, ELSE
addi  X5, X5, 1
add   X7, X0, X0
beq   X7, X0, STOP
ELSE: addi  X6, X6, -1
END
```
branch if h >= g

(ii)
```
START: blt   X6, X5, ELSE
       andi  X6, X6, 0
       beq   X6, X0, STOP
ELSE : andi  X5, X5, 0
STOP
```
branch if h < g.

Aswari Badkundri
ab10445

Q2. swap contents of X5, X6

eg: X5 : 1 X6 : 2.

```
add  X5, X5, X6    // X5 : 3.
sub  X6, X5, X6    // X6 : 1 → value of X5 now in X6.
sub  X5, X5, X6    // X5 : 2 → value of X6 now in X6
STOP
```

Here, the values of X5 and X6 have been swapped without using a 3rd register.

Asawari Badkundri
ab10445

3.

3.1) addi:

- Imem to ~~read~~ fetch ~~instruction~~ instruction : 140
- Read registers x2 : 140 (70 x 2)
- Write register : 60

340 pJ

→ addi reads 2 registers, writes in 1 for both single and 5-stage pipeline

3.2) ld:

- Imem to read instruction : 140
- Read registers x 2 : 140 (70 x 2)
- ~~Read~~ D-Mem Read : 140
- Register write : 60

480 pJ

→ ld reads from memory as well as registers

3.3) beq:

- Imem to fetch instruction : 140
- Read registers : 140 (70 x 2)

280 pJ

→ beq does not write to registers or memory, doesn't read from memory either.

Asawari Badkundri
ab10445

4) add (X1), X2, X3 IF ID ALU1 ALU2 MEM WB
add X5, X4, (X1) X X X IF ID ...

If no forwarding is used, 3 NOPs will have to be inserted after instruction 1 to avoid data hazards.

4.1. Here, forwarding from ALU2 from the 1st addi instruction to the 2nd ALU1 of the 2nd addi instruction can be used to reduce the number of stalls and NOPs

add X1, X2, X3: IF ID ALU1 ALU2 MEM WB
X IF ID → ALU1 ALU2 MEM WB

Here, the 'X' indicates a bubble.

∴ The ~~instruction~~ code with full forwarding (ALU2 → ALU1) becomes:

add X1, X2, X3
nop
add X5, X4, X1

The initial code show an EX to 1 data hazard.

Arawari Badkundri
ab1044s

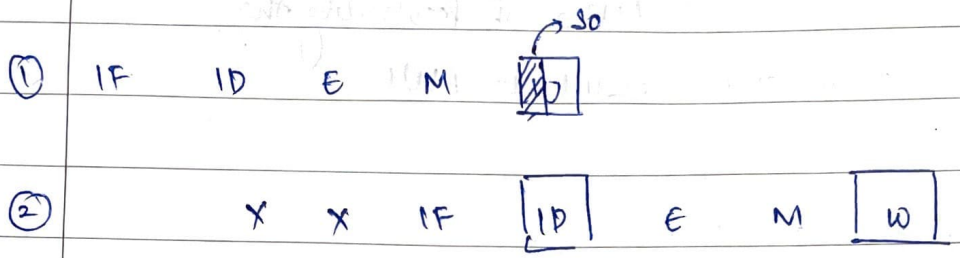
4.2

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
add X1, X2, X3	IF	ID	ALU1	ALU2	MEM	WB			
nop									
~~add X5, X4, X1~~									
add X5, X4, X1				IF	ID	ALU1	ALU2	MEM	WB

ALU 2 if forwarding the Result to ALU1

Asawari Badkundri
ab10445

5) addi s0, zero, 5 ① //1
L1: bge zero, s0, DONE ② //6
~~addi s0, s0~~ addi s0, s0, -1 ③ //5
j L1 ④ //5
DONE //1



Iterations: s0 : 5 s0: 4 s0: 3 s0: 2 s0: 1
s0 : 4 s0: 3 s0: 2 s0: 1 s0: 0

Here, the loop will run 5 times
∴ number of cycles:

addi will run 1 time
L1 : will run 6 times
addi will run 5 times
j will run 5 time
END : 1 time
18 instructions

Aswani Badkundri
ab10445

6) ∴ Assuming no data hazards and no structural hazards, the program will run in

$k + n - 1$ cycles

k: no. of stages

n: no of instructions

∴ $5 + 18 - 1$

= 22 cycles

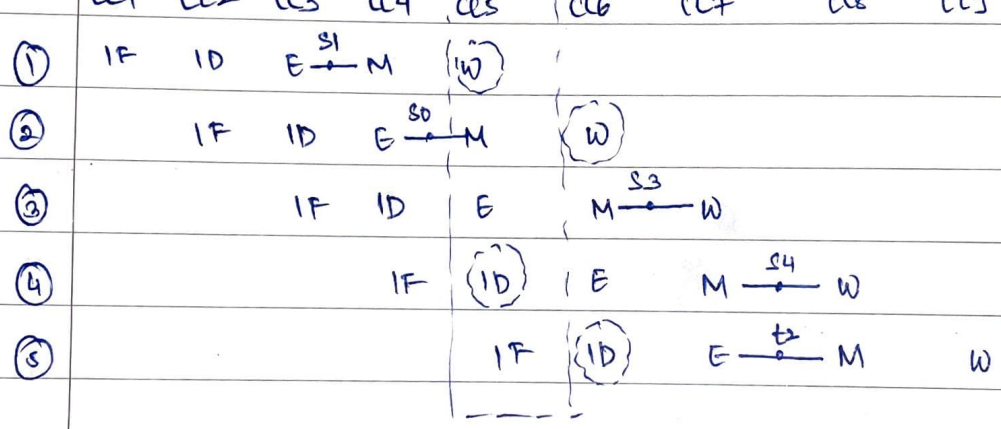
∴ CPI : $\frac{\#\text{cycles}}{\#\text{instructions}}$

= $\frac{22}{18} = 1.22$

6) xor (s1), s2, s3 (1)
addi (s0), s3, -4 (2)
lw s3, 16(s7) (3)
sw s4, 20((s1)) (4)
or t2, (s0), s1 (5)

Assuming that the hazard detection supports full forwarding and can stall instructions:

- There are no data hazards in these instructions as after xor, there are 2 instructions before the s0 instruction and similarly, after the addi instruction, there are 2 instructions before the or instruction.



In cycle 5, register s1 is being written into
s1 = s2 + s3

Here, the register that is being read is also s1
Hence As the both read and write are happening in the same register, this is a

Aswani Badkundni
ab1044s

6) structural hazard

∴ ~~A nop will~~ A nop will have to be inserted before instruction (4).

→ This will resolve the structural hazard taking place in cycle 5 and 6.

∴ New code:

```
xor   s1, s2, s3
addi  s0, s3, -4
lw    s3, 16(s7)
nop
sw    s4, 20(s1)
or    t1, s0, s1
```

Asawari Badkundri
ab10445

7) 7.1

```
or   X13, X12, X11
ld   X10, 0(X13)      EX to 1 → X13
ld   X11, 8(X13)      X11 : MEM to 1
add  X12, X10, X11    X10 : MEM to 2
subi X13, X12, 16     X12 : EX to 1
```

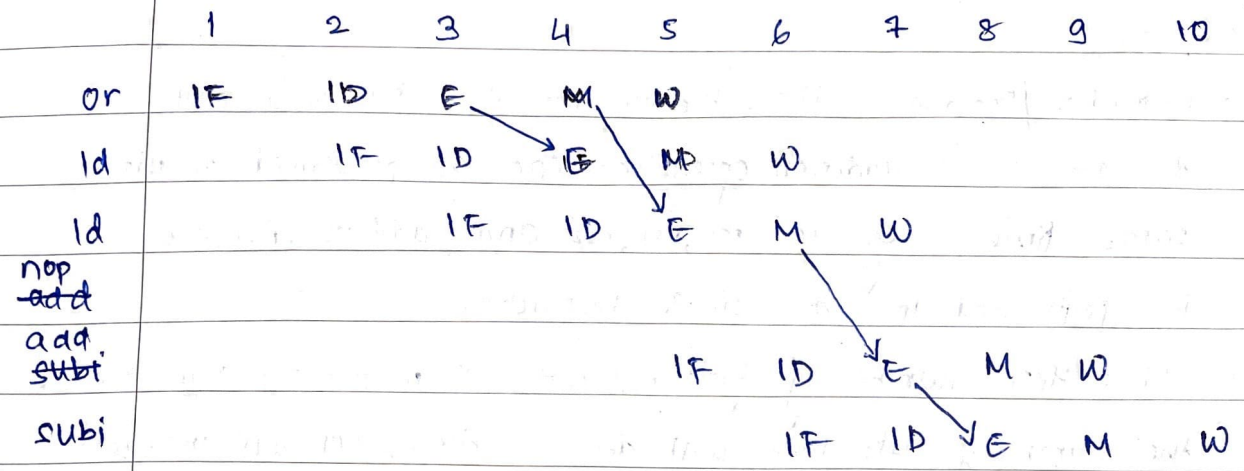
∴ Resolution :

```
or   X13, X12, X11    IF ID E M
nop
nop
ld   X10, 0(X13)      X X IF ID E M
ld   X11, 8(X13)      IF ID E M
nop
nop
add  X12, X10, X11    X X IF ID E M
nop
nop
subi X13, X12, 16     X X IF ID
```

The 'X' in to the right denote bubbles.

As there is no forwarding and no hazard detection, 6 nops will need to be added.

Asawari Badkundri
ab10445



E: Ex , M : MEM W: WB , The arrows show forwarding

Cycle	Forward A	Forward B	
1	X	X	Here, nothing is in execute stage
2	X	X	
3	0	0	
4	2	0	
5	1	0	
6	X	X	
7	0	1	
8	1	0	

Asawari Badkundri
ab10445

8) A

→ Instructions/Programs : Here, the number of instructions will decrease as 2 addition operations can be performed at the same time. eg add rs1, rs1, rs2 and add rs1, rs1, rs3 can be performed in a single instruction

→ CPI : Here, number of cycles required will remain the same but the number of instructions will decrease. Hence, CPI will increase.

→ Circuit complexity : Circuit complexity will increase as we need to read an extra register in this case.

B

→ Instructions/Programs : Here, number of instructions will decrease as we will not have to perform PC = PC + 4 separately. The PC will point to the next instruction after ~~automa~~ execution in ALU.

→ CPI : Here, number of cycles will remain the same but no of instructions will decrease. Hence, CPI will increase.

→ Circuit complexity : It will decrease as the adder for PC will not be required

C

→ Instructions/programs : No change as changing the numbers will not affect the instruction size.

→ CPI : no change as increasing the number of register will have no effect of number of cycles, neither number of instructions.

→ Circuit complexity : This will increase as more registers have been added to the circuit.