CSE 112:
Computer Organization
Sujay Deb

Quiz III | Set-B Rubric

IIIT-Delhi, Winter 2025

Duration : 60 Minutes
Date: Apr 08, 2025
Points: 30

Guidelines

If you found any ambiguity in any of the questions or there appears to be a lack of information, then write an assumption on the answer sheet to explain your side interpretation of the problem and solve accordingly.

Instruction Set	Description	Operation
addi rd,rs, imm[11:0]	Add immediate	rd = rs + sext(imm[11:0])
sub rd, rs1, rs2	Subtract registers	rd = rs1 - rs2
add rd, rs1, rs2	Add registers	rd = rs1 + rs2
lw rd,#imm[11:0](rs1)	Load Word	rd=mem(sext(rs1+imm))
sw rd,#imm[11:0](rs1)	Store Word	mem(sext(rs1+imm))=rd
beq rs1, rs2, imm[12:1]	Branch if equal	PC = PC + sext({imm[12:1],1'b0}) if rs1 == rs2
beq x0,x0,#0	Branch if equal	**HALT**

Stage Description of a general Five Stage Processor

1. Fetch - Fetches the instructions from instruction memory
2. Decode - Decodes the instructions and type of operations. It also reads the input operands.
3. Execute - Perform the arithmetic and logical operations, determine branches.
4. Memory - Perform the read and write operations at data memory.
5. Writeback - Updates the CPU register value.

[CO3] Problem I: Control Hazards **[05 Points]**

Consider a processor following RISC-V ISA and having a four-stage pipeline with the stages Fetch (F), Decode (D), Execute (E), and **Update (U)**. The processor does not have any data forwarding hardware and branches are determined in the execute stage. You need to determine the CPI (Cycles Per Instruction) of the below-mentioned assembly program.

Update(U) - This stage is a combination of both Memory and Writeback stage. The data memory read and write operations along with writes on CPU registers are performed in this stage.

```
addi x1, x0, #5
addi x2, x0, #4
addi x3, x0. #1
sub x4, x1, x3
beq x4, x2, END
add x1, x3, x4
sub x4, x2, x3
END: beq x0, x0, #0
```

Solution

As the complete information is not provided. So, we are accounting for both the answers.

if the register write is performed on posedge and read is performed on negedge of clock.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Marks**
addi x1, x0, #5	F	D	E	U												
addi x2, x0, #4		F	D	E	U											
addi x3, x0. #1			F	D	E	U										
sub x4, x1, x3				F	D	D	E	U								1
beq x4, x2, end						F	D	D	E	U						1
add x1, x3, x4							F	F	D							1
sub x4, x2, x3									F							1
end: beq x0, x0, #0										F	D	E	U			1

CPI= (13/8)

If the register writes are read are performed on the posedge of clock as shown in figure[1].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Marks
addi x1, x0, #5	F	D	E	U												
addi x2, x0, #4		F	D	E	U											
addi x3, x0. #1			F	D	E	U										
sub x4, x1, x3				F	D	D	D	E	U							1
beq x4, x2, **END**						F	D	D	D	D	E	U				1
add x1, x3, x4							F	F	F	F	D					1
sub x4, x2, x3											F					1
END: beq x0, x0, #0												F	D	E	U	1

CPI= (15/8)

[CO3] Problem II: Branch Penalty **[02 Points]**

Consider two processors C, D following RISC-V ISA and having stage description as follows..

(a) Processor C has 27 stages as F1, F2, . . F5, D1, D2, ... D3, E1, E2, . . . E9, M1, M2, . . M5, W1, W2, … W5. The branch operations are determined in the stage E8.

(b) Processor D has 87 stages as F1, F2, … F30, D1, D2, … D10, E1, E2, … E20, M1, M2, … M15, W1, W2, … W12. And, the branch operations are determined in the stage E8.

You need to calculate the Branch penalty of both the processors.

Solution

Branch Penalty is the number of instructions to be squashed on a branch mispredict. In the processor microarchitecture that we are studying as in figure[2]. The Branch Penalty is the number instructions to be squashed which is equal to the number of pipeline stages before the stage that determines the branch. We have stage E8 in both the processors that determine the branch.

The branch penalty in Processor C is 5 + 3 + 7 = 15 [01]

The branch penalty in processor D is 30 + 10 + 7 = 47 [01]

[CO2] Problem III: Compiler Optimization and Load Delay Slot **[04 Points]**

The compilers are advanced enough to optimize the code to fill the Load Delay slots. Considering a compiler capable of compiling the RISC-V assembly program. You need to optimize and if needed reorganize the below-mentioned assembly code such that the code takes a minimum number of cycles to complete. Assume that the processor has the hardware to support data forwarding.

Assembly Program

0x00: Sub x1,x2,x0
0x04: Add x3,x2,x1
0x08: Sub x4,x2,x1
0x0C: Lw x5,8(x4)
0x10: Add x8,x5,x0
0x14: Sub x9,x8,x1
0x18: Add x11,x10,x8
0x1C: Add x13,x10,x8
0x20: Sub x14,x10,x8

Solution

The best way to execute the program as fast as possible is to fill all the pipeline stages such as there is no stall. The code given to us will lead to the stalling of instruction at address 0x0C, 0x10 due to load delay slot. We need to fill this delay slot with some instructions that are not dependent on Load instruction at address 0x08.

The instructions at addresses 0x14, 0x18, 0x18 are independent of the above instructions.
So, one possible optimized code by the compiler can be.

Assembly Program

0x00: Sub x1,x2,x0
0x04: Add x3,x2,x1
0x08: Sub x4,x2,x1
0x0C: Lw x5,8(x4)
0x10: **NOP** **// add x0,x0,x0**
0x14: Add x8,x5,x0
0x18: Sub x9,x8,x1
0x1C: Add x11,x10,x8
0x20: Add x13,x10,x8
0x24: Sub x14,x10,x8

Assembly Program **(Wrong as we are studying an Inorder Processor)**

0x00: Sub x1,x2,x0
0x04: Sub x4,x2,x1
0x08: Lw x5,8(x4)
0x0C: Add x3,x2,x1 **// filling load delay slot**
0x10: Add x8,x5,x0
0x14: Sub x9,x8,x1
0x18: Add x11,x10,x8
0x1C: Add x13,x10,x8
0x20: Sub x14,x10,x8

Note to TAs: The job of a student is just to remove the dependency among the instruction at the original address 0x08, 0x0c (in the question).

Marking Scheme:

1. [2 Points] If NOP is added without affecting the functionality and the instructions are being performed in order.
2. [2 Points] If the instruction NOP is designed instead of just writing the word **NOP**.

[CO3] Problem IV: Data Hazards **[03 + 03 Points]**

The timing diagram of a processor following FIRST Block Instructions from RISC-V ISA is shown in figure[1].You need to execute the below-mentioned assembly program on a five stage microprocessor for the two cases.

(a) There is no data forwarding hardware in the microprocessor.
(b) The microprocessor is designed to support data forwarding figure[2].

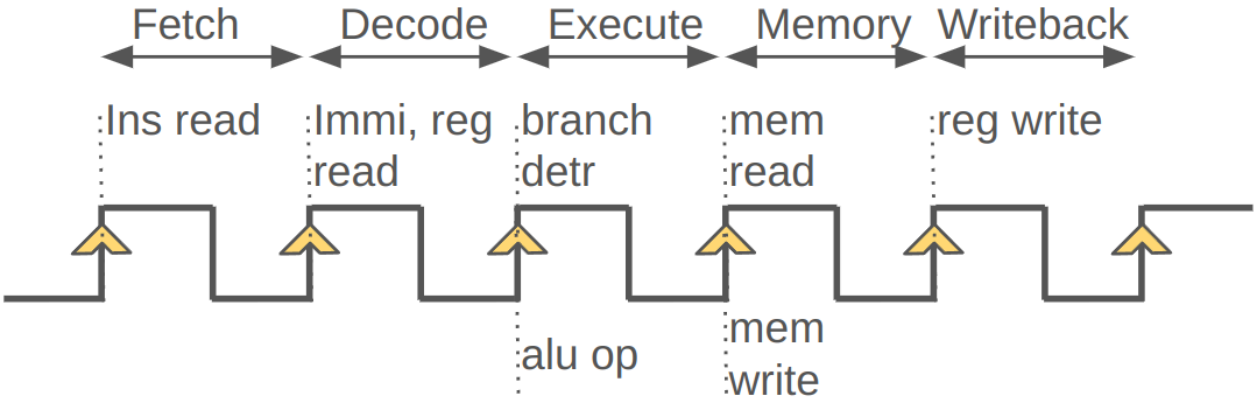

Figure 1: Timing diagram of a Five Stage RISC-V Processor following FIRST BLOCK Instructions.

Assembly Program

0x00: Addi x1,x0,#2
0x04: Sub x2,x1,x0
0x08: Lw x2,48(x0)
0x0C: Add x4,x2,x0
0x10: Add x5,x4,x0
0x14: Add x6,x5,x4

Solution

No Data Forwarding	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Marks
Add x1,x0,#2	F	D	E	M	W																		0.5
Sub x2,x1,x0		F	D	D	D	D	E	M	W														0.5
Lw x2,48(x0)			F	F	F	F	D	E	M	W													0.5
Add x4,x2,x0							F	D	D	D	D	E	M	W									0.5
Add x5,x4,x0								F	F	F	F	D	D	D	D	E	M	W					0.5
Add x6,x5,x4												F	F	F	F	D	D	D	D	E	M	W	0.5

With Data Forwarding	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Marks
Add x1,x0,#2	F	D	E	M	W																	0.5
Sub x2,x1,x0		F	D	E	M	W																0.5
Lw x2,48(x0)			F	D	E	M	W															0.5
Add x4,x2,x0				F	D	D	E	M	W													0.5
Add x5,x4,x0					F	F	D	E	M	W												0.5
Add x6,x5,x4							F	D	E	M	W											0.5

[CO3] Problem V: Hazard Control Unit **[09 Points]**

You need to draw the pipeline diagram of the below mentioned assembly program executed on a processor having the micro architecture shown in figure[2]. Thereupon, you need to highlight by creating **SQUARE** on the stage for data forwarding path **1A** being utilized and **CIRCLE** on the stages where the data forwarding path **1B** are being utilized for data forwarding.

Assembly Program

Addi x1,x0,#2
Add x2,x1,x1
Add x3,x2,x1
Lw x4,8(x3)
Add x5,x4,x0
Add x6,x5,x4
Lw x7,8(x6)
Lw x8,8(x7)
Add x9,x8,x1

Solution

1B = Circle
1A = Square

With Data Forwarding	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Marks
Add x1,x0,#2	F	D	E	M	W												1
Add x2,x1,x1		F	D	E	M	W											1
Add x3,x2,x1			F	D	EE	M	W										2
Lw x4,8(x3)				F	D	E	M	W									1
Add x5,x4,x0						F	D	E	M	W							1
Add x6,x5,x4							F	D	E	M	W						1
Lw x7,8(x6)								F	D	E	M	W					2
Lw x8,8(x7)									F	D	D	E	M	W			
Add x9,x8,x1										F	F	D	D	E	M	W	

Note: As, the microarchitecture does not contain the stall logic hardware. So, these two red colored instructions will not be evaluated.

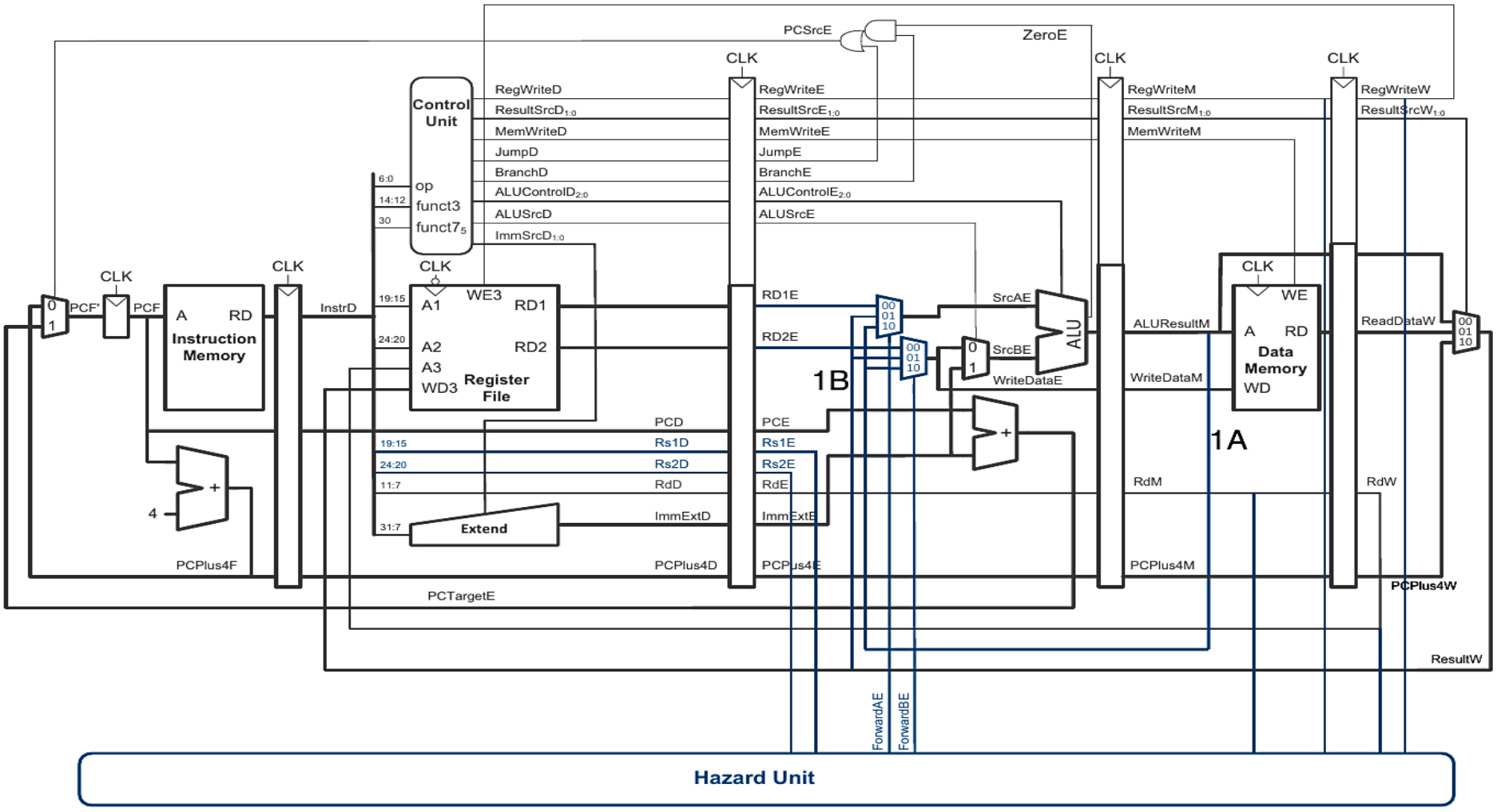

Figure 2: Microarchitecture of Five Stage RISC-V Processor following FIRST BLOCK INSTRUCTIONS.

Image Source: S. L. Harris and D. Harris, "Digital Design and RISC-V Computer Architecture Textbook," *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*, Raleigh, NC, USA, 2021

[CO3] Problem VI: CPI Analysis **[03+01 Points]**

You need to calculate the speedup in the execution of the mentioned benchmark program when executed on two different processors A and B. Both processors have five stages and operate as per the clock cycle diagram mentioned in figure[1].

Processor_A Miss Predicts 20% of branches and carries no data forwarding hardware capability.

Processor_B Miss Predicts 7% of branches and contains hazard control as shown in figure[2].

Assume 20% of the LOADs in the benchmark program are used by the next/subsequent instruction.

Specification of the Benchmark Program

30% LOAD
20% STORE
10% BRANCH
40% ALU Operation

Solution

(a) Every branch miss predict leads to squash of 2 instructions or two additional cycles per branch miss predict for the complete program execution.
(b) Every subsequent instruction using the load variable leads to one cycle stall.

Total Number of Cycles Consumed by Processor_A

Additional cycles in comparison to CPI of 1=>

Due to branch miss predicts = Branch_percentage * Miss_predict * no_of_stages_earlier_to_that = 0.1 * 0.2 * 2 [0.5 Points]

Due to load delay slot = Load_percentage * load_dependance_percentage * no_of_extra_cycle = 0.3 * 0.2 * 2 [1 Point]

Total = 0.04 + 0.12 = 0.16

Total Number of Cycles Consumed by Processor_B

As there is no hardware designed to handle load delay slots. So, this processor will also face the same issue of load delay slots.

Additional cycles in comparison to CPI of 1=>

Due to branch miss predicts = 0.1 * 0.07 * 2 [1 Point]

Due to load delay slot = 0.3 * 0.2 * 1 [1 Point]

Total = 0.014 + 0.06 = 0.074

Speedup = (1 + 0.076) / (1 + 0.16) [1 Point]

Note: As, nothing is mentioned in the questions about the dependency among the ALU operation. Or dependency among other possible situations. So, if someone has accounted for these things and wrote their assumption appropriately. They will be rewarded accordingly.

CSE 112:
Computer Organization
Sujay Deb

IIIT-Delhi, Winter 2025

Quiz III | Set-A Rubric

Duration : 60 Minutes
Date: Apr 08, 2025
Points: 30

Guidelines

If you found any ambiguity in any of the questions or there appears to be a lack of information, then write an assumption on the answer sheet to explain your side interpretation of the problem and solve accordingly.

Instruction Set	Description	Operation
addi rd,rs, imm[11:0]	Add immediate	rd = rs + sext(imm[11:0])
sub rd, rs1, rs2	Subtract registers	rd = rs1 - rs2
add rd, rs1, rs2	Add registers	rd = rs1 + rs2
lw rd,#imm[11:0](rs1)	Load Word	rd=mem(sext(rs1+imm))
sw rd,#imm[11:0](rs1)	Store Word	mem(sext(rs1+imm))=rd
beq rs1, rs2, imm[12:1]	Branch if equal	PC = PC + sext({imm[12:1],1'b0}) if rs1 == rs2
beq x0,x0,#0	Branch if equal	**HALT**

Stage Description of a general Five Stage Processor

6. Fetch - Fetches the instructions from instruction memory
7. Decode - Decodes the instructions and type of operations. It also reads the input operands.
8. Execute - Perform the arithmetic and logical operations, determine branches.
9. Memory - Perform the read and write operations at data memory.
10. Writeback - Updates the CPU register value.

[CO3] Problem I: Control Hazards **[05 Points]**

Consider a processor following RISC-V ISA and having a four-stage pipeline with the stages Fetch (F), Decode (D), Execute (E), and **Update (U)**. The processor does not have any data forwarding hardware and branches are determined in the execute stage. You need to determine the CPI (Cycles Per Instruction) of the below-mentioned assembly program.

Update(U) - This stage is a combination of both Memory and Writeback stage. The data memory read and write operations along with writes on CPU registers are performed in this stage.

```
addi x1, x0, #5
addi x2, x0, #4
addi x3, x0. #1
sub x4, x1, x3
beq x4, x2, END
add x1, x3, x4
sub x4, x2, x3
END: beq x0, x0, #0
```

Solution

If the register write is performed on posedge and read is performed on negedge of clock.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Marks
addi x1, x0, #5	F	D	E	U												
addi x2, x0, #4		F	D	E	U											
addi x3, x0. #1			F	D	E	U										
sub x4, x1, x3				F	D	D	E	U								1
beq x4, x2, end						F	D	D	E	U						1
add x1, x3, x4							F	F	D							1
sub x4, x2, x3									F							1
end: beq x0, x0, #0										F	D	E	U			1

CPI= (13/8)

If the register writes are read are performed on the posedge of clock as shown in figure[1].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Marks
addi x1, x0, #5	F	D	E	U												
addi x2, x0, #4		F	D	E	U											
addi x3, x0. #1			F	D	E	U										
sub x4, x1, x3				F	D	D	D	E	U							1
beq x4, x2, end						F	D	D	D	D	E	U				1
add x1, x3, x4							F	F	F	F	D					1
sub x4, x2, x3											F					1
end: beq x0, x0, #0												F	D	E	U	1

CPI= (15/8)

[CO3] Problem II: Branch Penalty **[02 Points]**

Consider two processors C, D following RISC-V ISA and having stage description as follows..

(c) Processor C has 37 stages as F1, F2, . . F10, D1, D2, ... D5, E1, E2, . . . E9, M1, M2, . . M7, W1, W2, … W6. The branch operations are determined in the stage E5.

(d) Processor D has 95 stages as F1, F2, … F30, D1, D2, … D20, E1, E2, … E20, M1, M2, … M15, W1, W2, … W10. And, the branch operations are determined in the stage E5.

You need to calculate the Branch penalty of both the processors.

Solution

Branch Penalty is the number of instructions to be squashed on a branch mispredict. In the processor microarchitecture that we are studying as in figure[2]. The Branch Penalty is the number instructions to be squashed which is equal to the number of pipeline stages before the stage that determines the branch. We have stage E5 in both the processors that determine the branch.

The branch penalty in Processor C is 10 + 5 + 4 = 19 [01 Point]

The branch penalty in processor D is 30 + 20 + 4 = 54 [01 Point]

[CO2] Problem III: Compiler Optimization and Load Delay Slot **[04 Points]**

The compilers are advanced enough to optimize the code to fill the Load Delay slots. Considering a compiler capable of compiling the RISC-V assembly program. You need to optimize and if needed reorganize the below-mentioned assembly code such that the code takes a minimum number of cycles to complete. Assume that the processor has the hardware to support data forwarding.

Assembly Program

0x00: Addi x1,x2,x0
0x04: Sub x3,x2,x1
0x08: Lw x4,8(x2)
0x0C: Add x5,x4,x0
0x10: Sub x6,x5,x1
0x14: Add x11,x10,x8
0x18: Add x13,x10,x8
0x1C: Sub x14,x10,x8

Solution

The best way to execute the program as fast as possible is to fill all the pipeline stages such as there is no stall. The code given to us will lead to the stalling of instruction at address 0x0C, 0x10 due to load delay slot. We need to fill this delay slot with some instructions that are not dependent on Load instruction at address 0x08.

The instructions at addresses 0x14, 0x18, 0x18 are independent of the above instructions.
So, one possible optimized code by the compiler can be.

Assembly Program

0x00: Addi x1,x2,x0
0x04: Sub x3,x2,x1
0x08: Lw x4,8(x2)
0x0C: Add x11,x10,x8
0x10: Add x13,x10,x8
0x14: Sub x14,x10,x8
0x18: Add x5,x4,x0
0x1C: Sub x6,x5,x1

Assembly Program **(Wrong as we are studying an Inorder Processor)**

0x00: Sub x1,x2,x0
0x04: Sub x4,x2,x1
0x08: Lw x5,8(x4)
0x0C: Add x3,x2,x1 **// filling load delay slot**
0x10: Add x8,x5,x0
0x14: Sub x9,x8,x1
0x18: Add x11,x10,x8
0x1C: Add x13,x10,x8
0x20: Sub x14,x10,x8

Note to TAs: The job of a student is just to remove the dependency among the instruction at the original address 0x08, 0x0c (in the question).

Marking Scheme:

3. [2 Points] If NOP is added without affecting the functionality and the instructions are being performed in order.
4. [2 Points] If the instruction NOP is designed instead of just writing the word **NOP**.

[CO3] Problem IV: Data Hazards **[03 + 03 Points]**

The timing diagram of a processor following FIRST Block Instructions from RISC-V ISA is shown in figure[1].You need to execute the below-mentioned assembly program on a five stage microprocessor for the two cases.

(c) There is no data forwarding hardware in the microprocessor.

(d) The microprocessor is designed to support data forwarding figure[2].

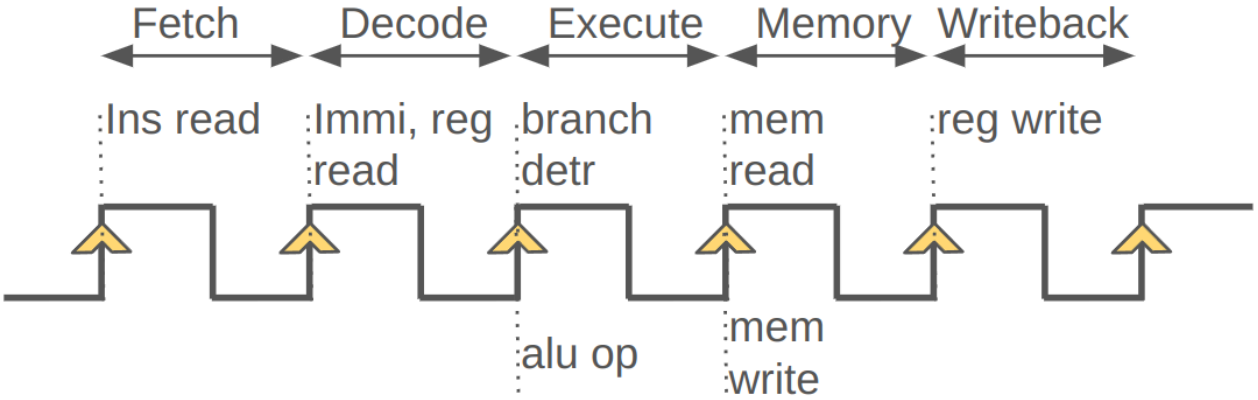

Figure 1: Timing diagram of a Five Stage RISC-V Processor following FIRST BLOCK Instructions.

Assembly Program

0x00: Add x1,x0,#2
0x04: Lw x3,8(x0)
0x08: Add x4,x3,x0
0x0c: Add x5,x4,x0
0x10: Add x6,x5,x4

Solution

No Data Forwarding	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Marks
Add x1,x0,#2	F	D	E	M	W														0.5
Lw x3,8(x0)		F	D	E	M	W													0.5
Add x4,x3,x0			F	D	D	D	D	E	M	W									0.5
Add x5,x4,x0				F	F	F	D	D	D	D	D	E	M	W					0.5
Add x6,x5,x4							F	F	F	F	F	D	D	D	D	E	M	W	1

With Data Forwarding	1	2	3	4	5	6	7	8	9	10					Marks
Add x1,x0,#2	F	D	E	M	W										0.5
Lw x3,8(x0)		F	D	E	M	W									0.5
Add x4,x3,x0			F	D	D	E	M	W							0.5
Add x5,x4,x0				F	F	D	E	M	W						0.5
Add x6,x5,x4						F	D	E	M	W					1

[CO3] Problem V: Hazard Control Unit **[09 Points]**

You need to draw the pipeline diagram of the below mentioned assembly program executed on a processor having the micro architecture shown in figure[2]. Thereupon, you need to highlight by creating **CIRCLE** on the stage for data forwarding path **1A** and **SQUARE** on the stages where the data forwarding path **1B** are being utilized for data forwarding.

Assembly Program

Addi x1,x0,#2
Add x2,x1,x1
Add x3,x2,x1
Lw x4,8(x3)
Add x5,x4,x0
Add x6,x5,x4
Lw x7,8(x6)
Lw x8,8(x7)
Add x9,x8,x1

Solution

1A = Circle

1B = Square

With Data Forwarding	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Marks
Add x1,x0,#2	F	D	E	M	W												1
Add x2,x1,x1		F	D	E	M	W											1
Add x3,x2,x1			F	D	EE	M	W										2
Lw x4,8(x3)				F	D	E	M	W									1
Add x5,x4,x0						F	D	E	M	W							1
Add x6,x5,x4							F	D	E	M	W						1
Lw x7,8(x6)								F	D	E	M	W					2
Lw x8,8(x7)									F	D	D	E	M	W			
Add x9,x8,x1										F	F	D	D	E	M	W	

Note: As, the microarchitecture does not contain the stall logic hardware. So, these two red colored instructions will not be evaluated.

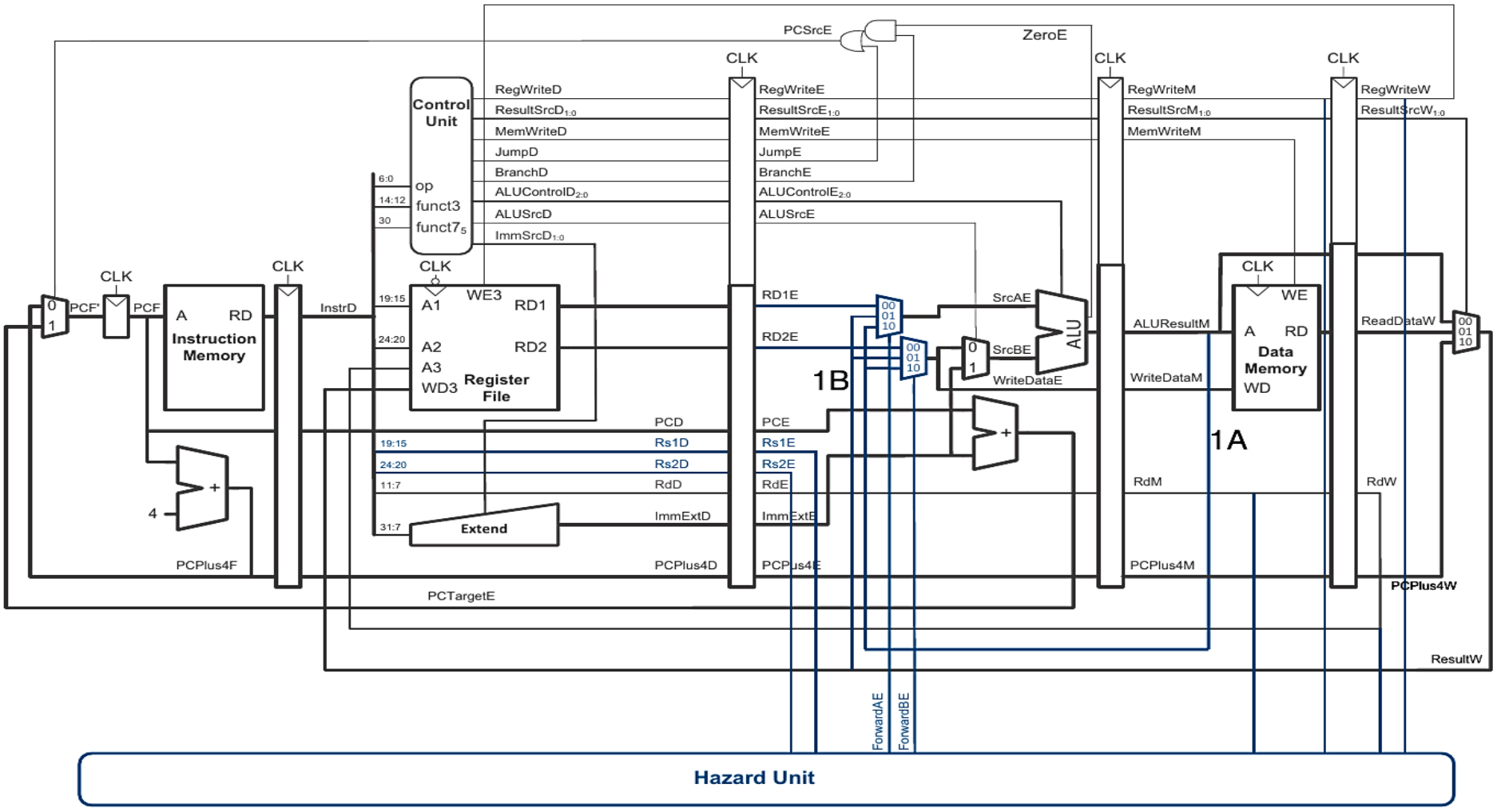

Figure 2: Microarchitecture of Five Stage RISC-V Processor following FIRST BLOCK INSTRUCTIONS.

Image Source: S. L. Harris and D. Harris, "Digital Design and RISC-V Computer Architecture Textbook," *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*, Raleigh, NC, USA, 2021

[CO3] Problem VI: CPI Analysis **[03+01 Points]**

You need to calculate the speedup in the execution of the mentioned benchmark program when executed on two different processors A and B. Both processors have five stages and operate as per the clock cycle diagram mentioned in figure[1].

Processor_A Miss Predicts 30% of branches and carries no data forwarding hardware capability.

Processor_B Miss Predicts 5% of branches and contains hazard control as shown in figure[2].

Assume 20% of the LOADs in the benchmark program are used by the next/subsequent instruction.

Specification of the Benchmark Program

30% LOAD
20% STORE
10% BRANCH
40% ALU Operation

Solution

(c) Every branch miss predict leads to squash of 2 instructions or two additional cycles per branch miss predict for the complete program execution.
(d) Every subsequent instruction using the load variable leads to one cycle stall.

Total Number of Cycles Consumed by Processor_A

Additional cycles in comparison to CPI of 1=>

Due to branch miss predicts = 0.1 * 0.3 * 2 [0.5 Points]

Due to load delay slot = 0.2 * 0.3 * 2 [0.5 Points]

Total = 0.06 + 0.12 = 0.18

Total Number of Cycles Consumed by Processor_B

As there is no hardware designed to handle load delay slots. So, this processor will also face the same issue of load delay slots.

Additional cycles in comparison to CPI of 1=>

Due to branch miss predicts = 0.1 * 0.05 * 2 [1 Point]

Due to load delay slot = 0.2 * 0.3 * 1 [1 Point]

Total = 0.01 + 0.06 = 0.07

Speedup = (1 + 0.07) / (1 + 0.18) = 0.907 [01 Point]