Solutions

ECE 369A

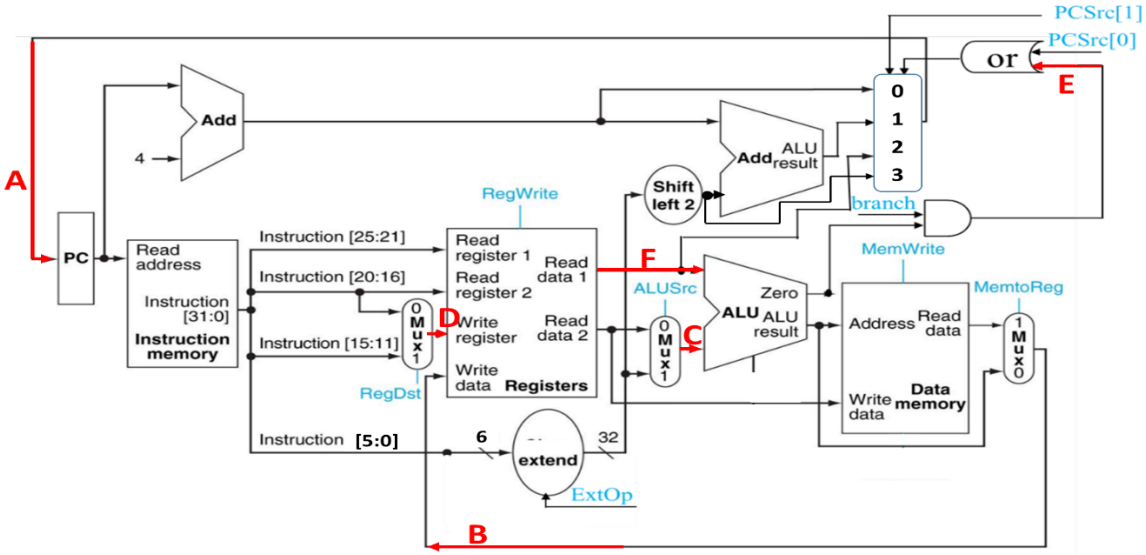
Fundamentals of Computer Architecture

Exam II – Type ABCD Solutions

Fall 2023

Solutions

Problem 1 (17 pts)



This datapath is designed to execute regular MIPS instructions following the MIPS specification, but it is somewhat different from the datapath introduced in class for a good reason. After benchmarking the vbsme.s code, your teammate found that values in the offset field of I-type instructions never exceeded 31. Therefore as an optimization, your teammate decided to use only six bits of the offset field in an I-type instruction as shown on the datapath. Assume that:

- The register file and memory both write on the rising edge of the clock
- The extender will zero extend if the ExtOp bit is 0 and sign extend when the ExtOp bit is 1
- The data memory reads asynchronously but has synchronous writes.
- Initially $t0 is 16, $t1 is 12 and PC is 204 and we are executing the **MIPS** instruction **slti $t0, $t1, -4**
- Control signals for this MIPS instruction are shown in the table below.

Opcode	RegDst	RegWrite	ExtOp	ALUSrc	ALU Operation	MemWrite	MemToReg	branch	PCSrc
slti	1	1	0	0	set less than	0	0	1	01

a) You are debugging this design. What values will you observe on wires labeled from "A" through "F"?

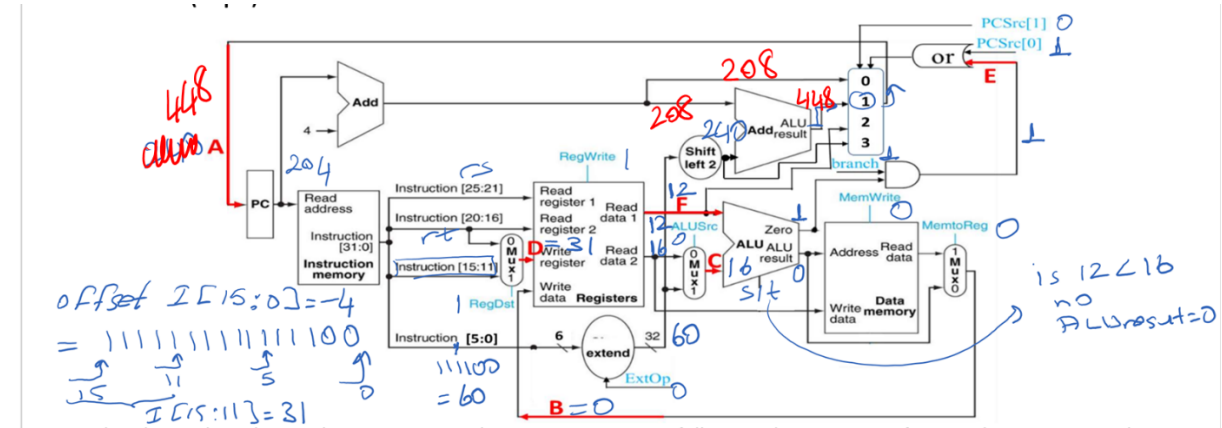
Wire	A	B	C	D	E	F
Value						

b) Indicate the error(s) in the control signal values and show the correct value(s) on the table. Take also into account value(s) that should be don't cares for full credit.

Solutions

Solution

a-type

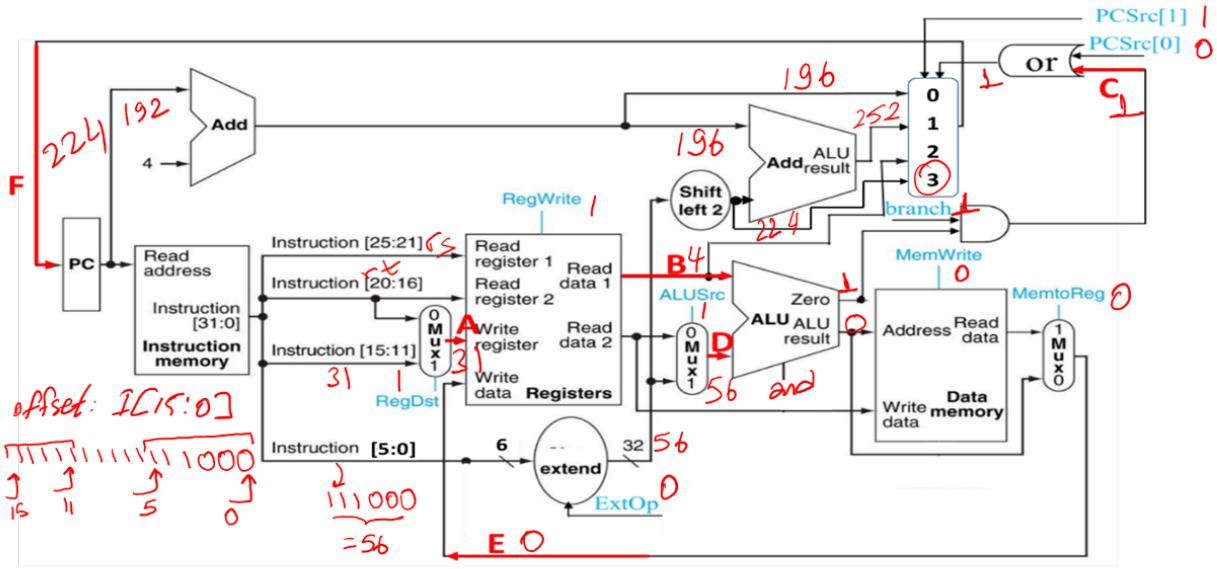


This datapath is designed to execute regular MIPS instructions following the MIPS specification, but it is somewhat different from the datapath introduced in class for a good reason. After benchmarking the vbsme.s code, your teammate found that values in the offset field of I-type instructions never exceeded 31. Therefore as an optimization, your teammate decided to use only six bits of the offset field in an I-type instruction as shown on the datapath. Assume that:

- The register file and memory both write on the rising edge of the clock
- The extender will zero extend if the ExtOp bit is 0 and sign extend when the ExtOp bit is 1
- The data memory reads asynchronously but has synchronous writes.
- Initially $t0 is 16, $t1 is 12 and PC is 204 and we are executing the **MIPS** instruction **slti $t0, $t1, -4**
- Control signals for this MIPS instruction are shown in the table below.

Opcode	RegDst	RegWrite	ExtOp	ALUSrc	ALU Operation	MemWrite	MemToReg	branch	PCSrc
slti	1 0	1	0 1	0 1	set less than	0	0	1 0	01 00

b-type



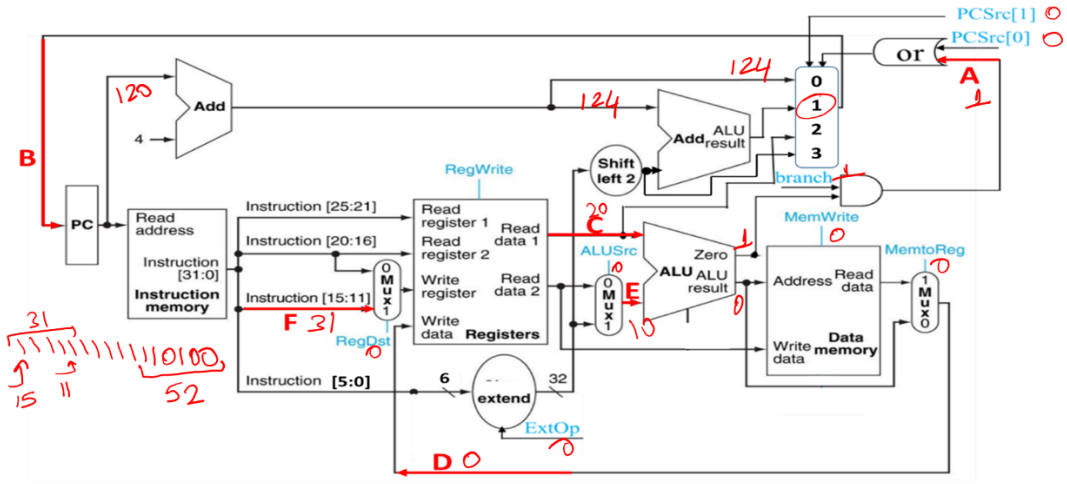
Opcode	RegDst	RegWrite	ExtOp	ALUSrc	ALU Operation	MemWrite	MemToReg	branch	PCSrc
andi	1 0	1 0	0 1	1	and	0	0	1 0	00

andi $t0, $t1, -8

PC = 192

Solutions

c-type

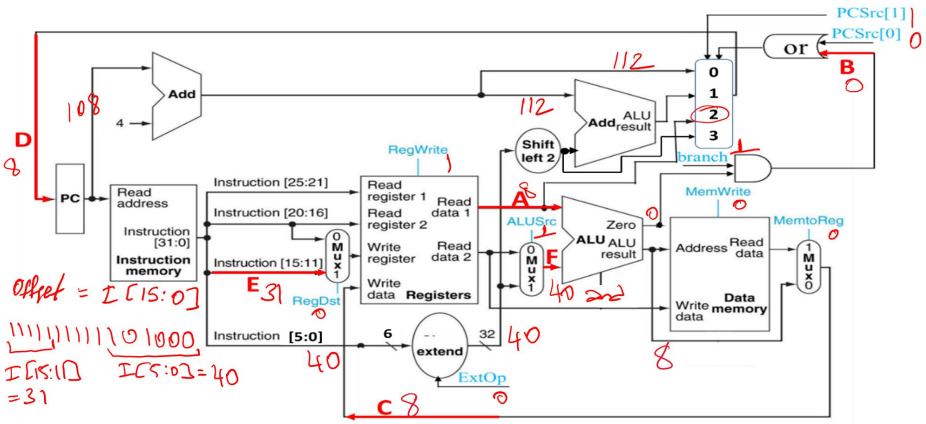


Opcode	RegDst	RegWrite	ExtOp	ALUSrc	ALU Operation	MemWrite	MemToReg	branch	PCSrc
slti	0	1	1	1	set less than	0	0	0	00

slti $t0, $t1, -12

PC = 120

d-type



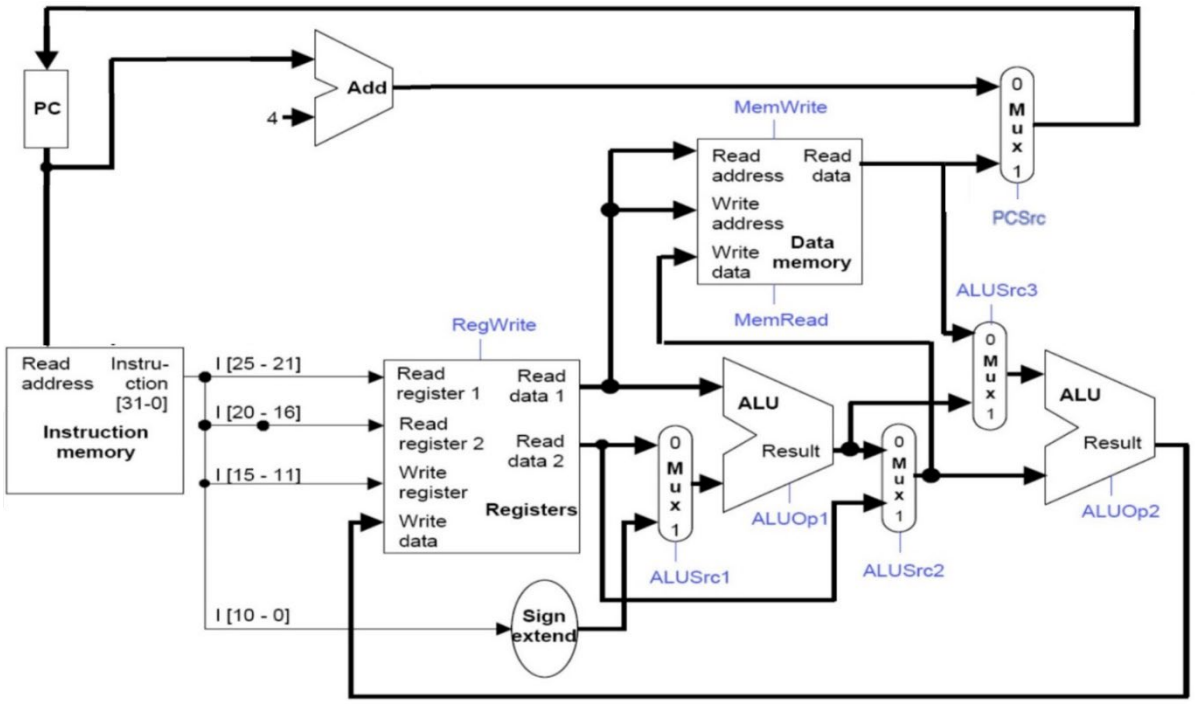
Opcode	RegDst	RegWrite	ExtOp	ALUSrc	ALU Operation	MemWrite	MemToReg	branch	PCSrc
andi	0	1	0	1	and	0	0	0	00

andi $t0, $t1, -24

PC = 108

Solutions

Problem 2 (18 pts):



Part (a) Your goal is to introduce a "pop" instruction specification for the provided datapath. All instructions use the same format given below. Two of the instruction fields are filled for you. The "opcode" is the integer value 64 and "rs" field is labeled as $sp for the "pop" instruction.

opcode I[31:26]	rs I[25:21]	rt I[20:16]	rd I[15:11]	imm I[10:0]
64	$sp			

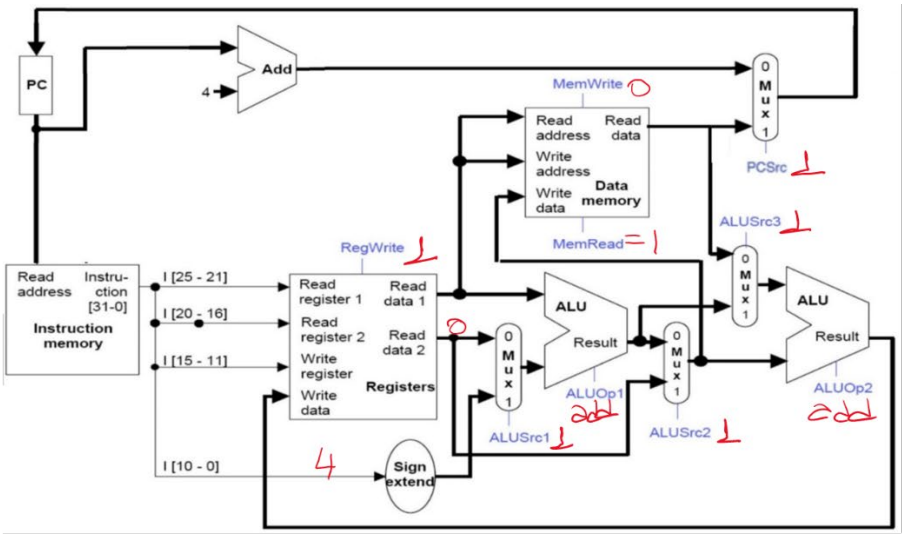
"pop" instruction involves the following operations:

```
# PC = Memory[GPR[$sp]]  where $sp is GPR[29] in the register file
# GPR[[$sp] = GPR[[$sp] + 4,
# GPR refers to general purpose register (register file)
```

i) Finalize the "pop" specification by utilizing the unused fields in the instruction so that it can be executed without having to make any changes to the datapath in a single clock cycle. What should be the values in rt, rd and imm fields of the pop instruction? Indicate on the table above.

ii) What should be the values for each control signal? On the datapath figure, indicate the value of each control signal in order to realize the "pop". You must Use X for don't care when applicable. ALUOp can be one of the following operations: add, sub, mul, sll, and srl. You are **not** allowed to modify the datapath.

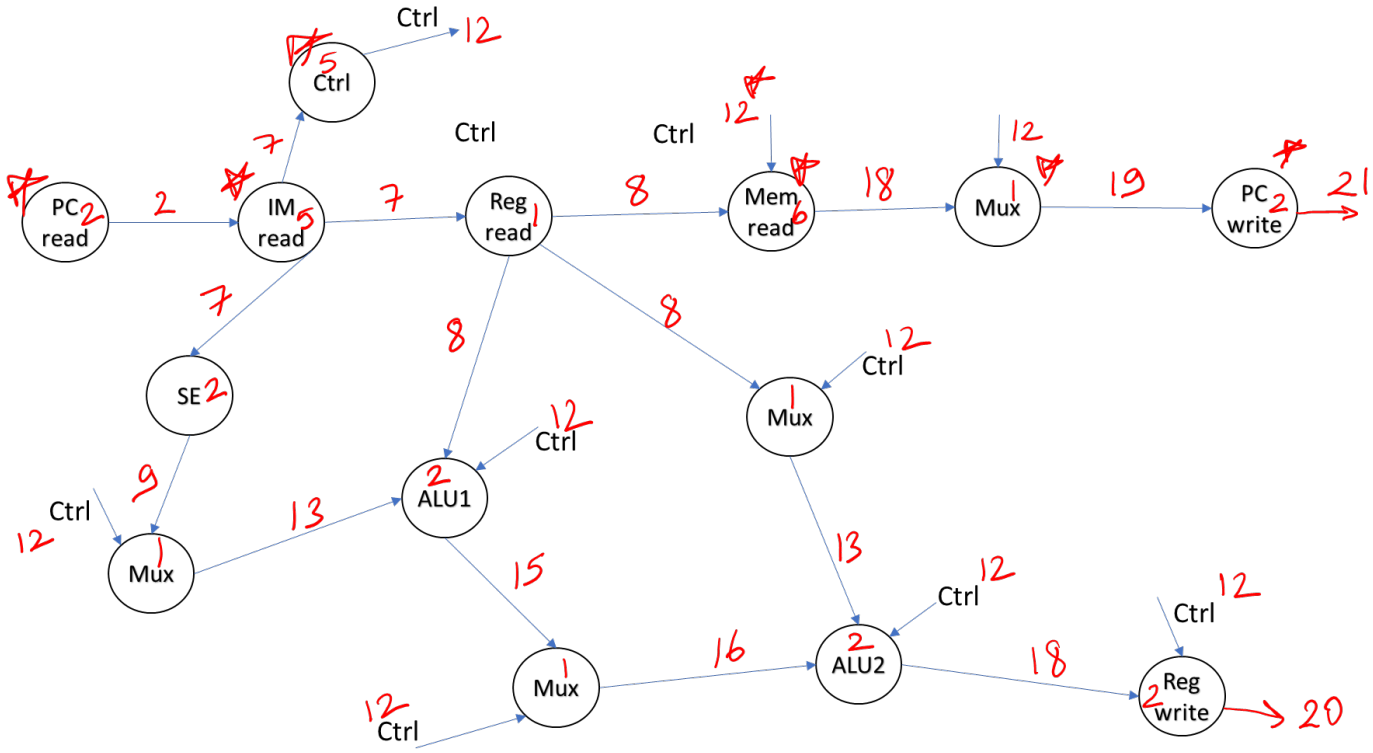
Solutions



opcode I[31:26]	rs I[25:21]	rt I[20:16]	rd I[15:11]	imm I[10:0]
64	$sp	0	29	4

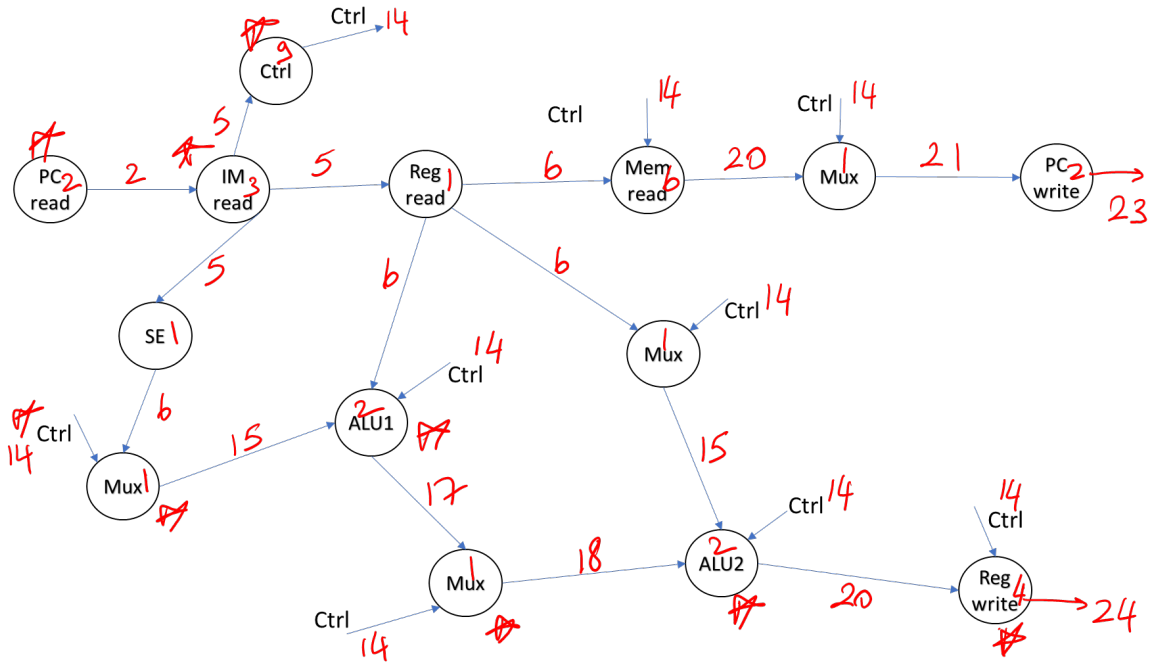
```
# PC = Memory[GPR[$sp]]  where $sp is
GPR[29] in the register file
# GPR[[$sp] = GPR[[$sp] + 4,
```

a-type timing analysis

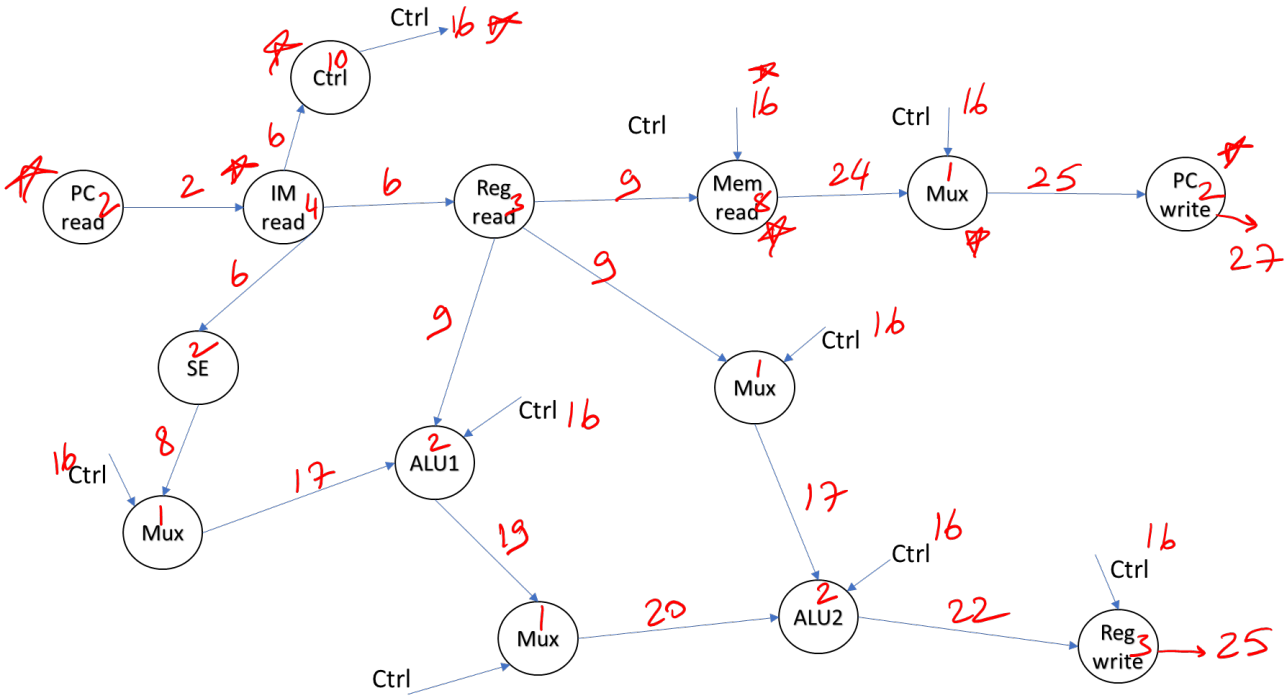


Solutions

B type

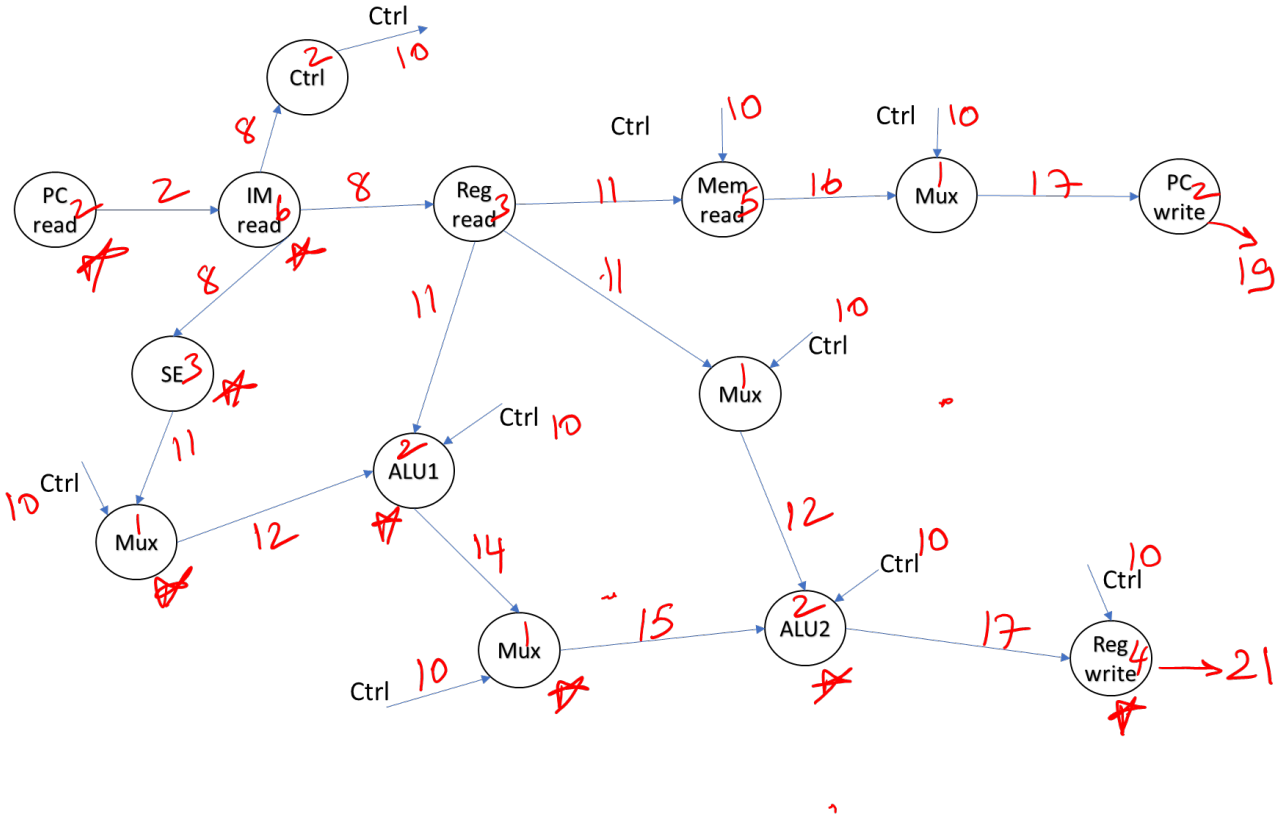


C type

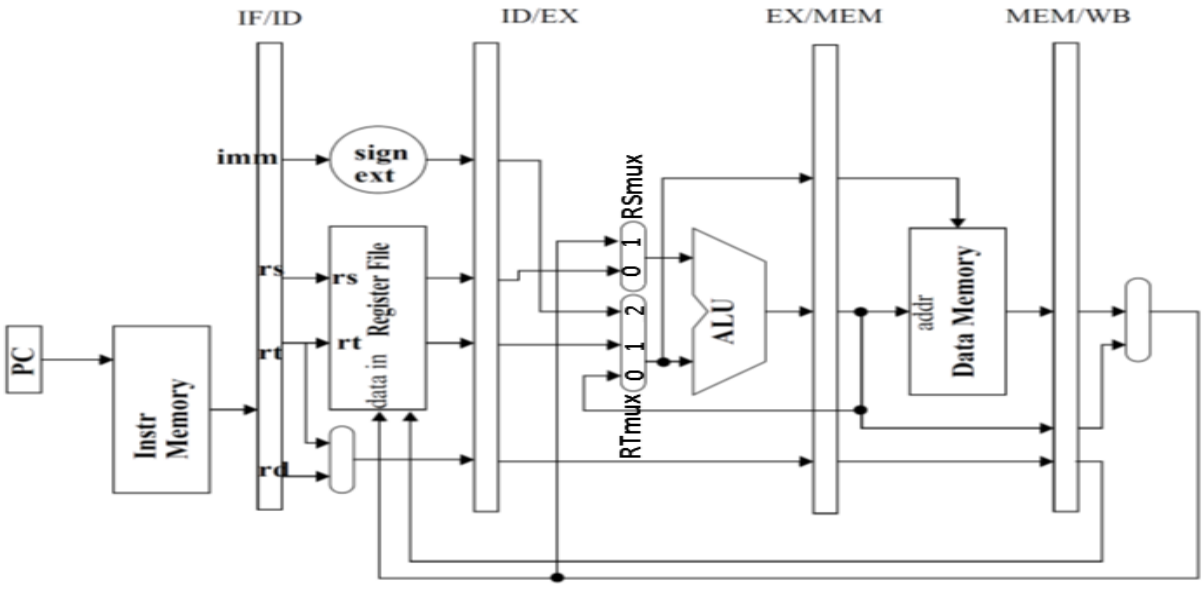


Solutions

D type



Problem 3



Solutions

A type

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13
lw $1, 8($3)	F	D	X	M	W								
sub $4, $2, $1		F	D	D	D	E	M	W					
or $8, $1, $4					F	D	E	M	W				
add $5 ,$1, $4						F	D	D	E	M	W		

a) What is the CPI for this code sequence when executed on the datapath given above? Show your work.
CPI = 11/4

b) What would be the CPI of this code sequence if we ran it **on a single cycle datapath**?

Takes 1 clock cycle to complete a single instruction on a single cycle datapath.

B

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13
add $1, $2, $3	F	D	EX	M	W								
sub $4, $5, $1		F	D	E	M	W							
or $8, $3, $1			F	D	D	E	M	W					
lw $4, 4($8)				F	F	D	D	E	M	W			

a) CPI = 10/4, b)CPI=1

C

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13
lw $1, 8($3)	F	D	EX	M	W								
sub $4, $2, $5		F	D	E	M	W							
or $8, $1, $4			F	D	E	M	W						
add $5 ,$8, $8				F	D	D	D	E	M	W			

a) CPI = 10/4, b)CPI=1

Solutions

D

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13
add $1, $2, $3	F	D	EX	M	W								
sub $4, $1, $5		F	D	D	E	M	W						
or $8, $1, $4			F	F	D	E	M	W					
lw $7, 4($8)					F	D	D	E	M	W			

10/4, 1

Solutions

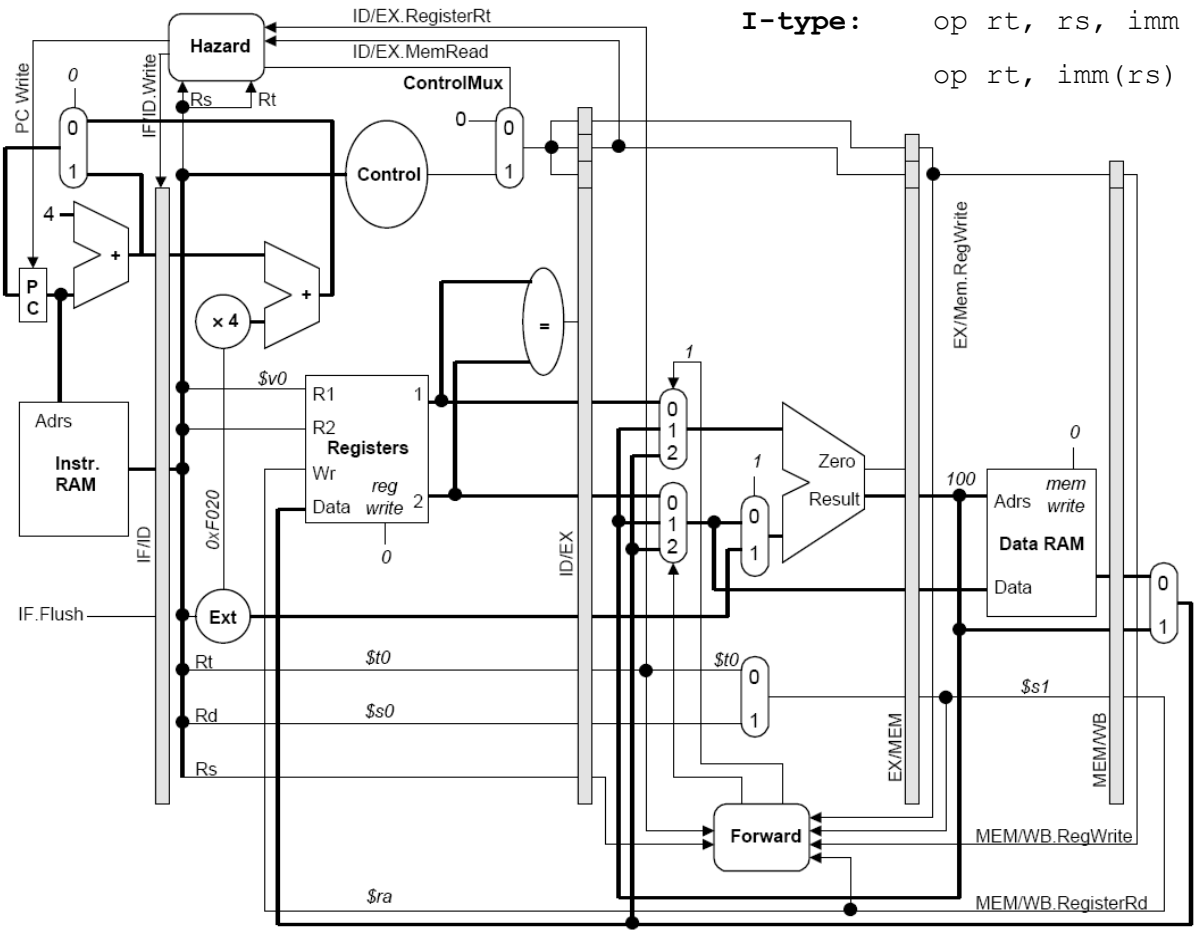
Problem 4 (extra credit)

A

R-type: `op rd, rs, rt`

I-type: `op rt, rs, imm`

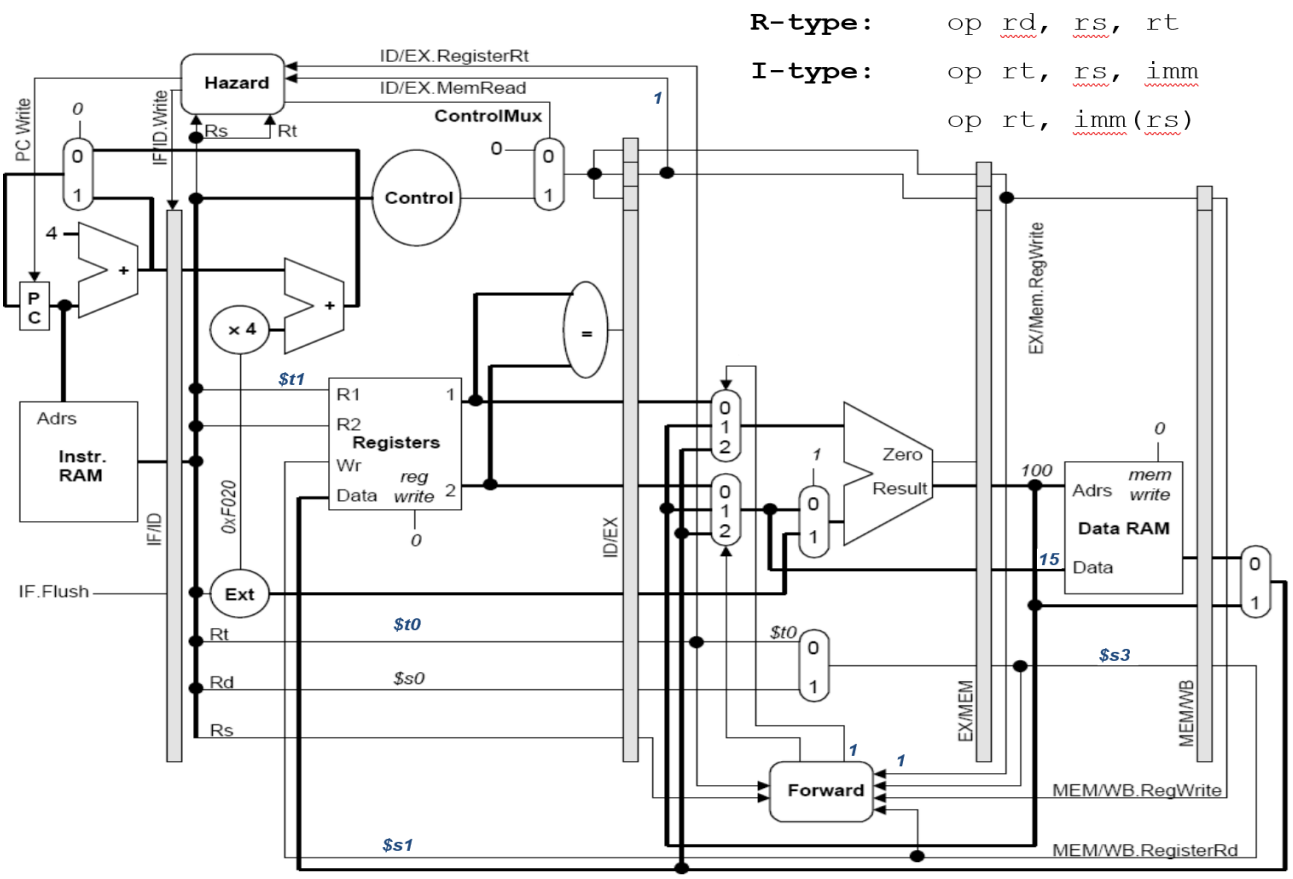
`op rt, imm(rs)`



	ID Stage (3pts)	EX Stage (3pts)	ME Stage (Bonus)	WB Stage (Bonus)
	a) bne $s0, $s3, label s0	a) sub $t0, $s1, $t0	a) lw $s0, 100($s1)	a) sw $t1, -4($v0)
	b) bne $t0, $v0, label	b) xori $t1, $s3, 100	b) sw $s1, 100($t1)	b) addi $s1, $v0, 100
	c) bne $s0, $s4, label	c) andi $s1, $t0, 100	c) slti $s1, $s2, 100	c) lw $a1, 16($s1)
	d) bne $t0, $t1, label	d) ori $t0, $s1, 80	d) beq $s1, $s3, loop	d) add $s1, $s1, t0
	e) add $s0, $v0, $t0	e) lw $t0, 4($s3)	e) addi $s1, $s2, 12	e) sub $s1, $s1, $a3
Justify Your Selection	Select line for the mux before the PC is 0. This is a branch instruction with rs = v0 and rt = t0	ALU first input forwarded from MEM stage. Destination register in MEM is s1 (= rs). Alu second input is the immediate field. Rt is t0.	memwrite = 0, not a sw adrs input is 100, not a slti since forwarding to ex stage, regwrite must be high, not a beq only option is addi.	regwrite is 0 only sw fits

Solutions

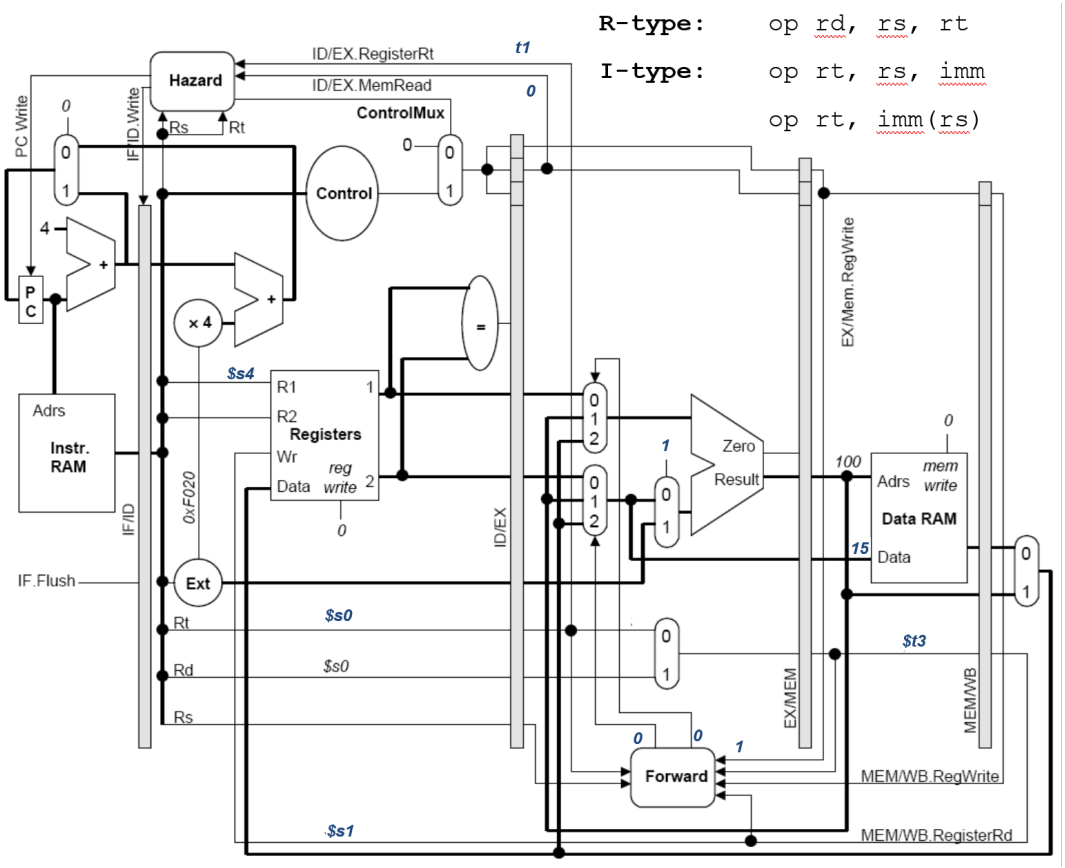
B



	ID Stage (3pts)	EX Stage (3pts)	ME Stage (Bonus)	WB Stage (Bonus)
	a) bne $s0, $s3, label b) bne $t0, $v0, label c) bne $t0, $v0, label d) bne $t0, $t1, label e) add $s0, $t1, $t0	a) sub $t0, $s8, $t0 b) xori $t1, $s3, 100 c) and $s1, $s3, t0 d) ori $t0, $s1, 80 e) lw $t0, 4($s3)	a) addi $s3, $t0, 8 b) sw $s3, 100($s1) c) slti $s3, $s2, 100 d) beq $s3, $s2, 100 e) addi $s1, $s2, 12	a) add $ra, $s1, $t0 b) add $s1, $t1, $t0 c) lw $s1, 16($s1) d) sw $ts3, -4($s0) e) and $s1, $ra, $a3
Justify Your Selection	Select line for the mux before the PC is 0. This is a branch instruction with rs = t1 and rt = t0	**MemRead is 1**	memwrite = 0, not a sw adrs input is 100, not a slti since forwarding to ex stage, regwrite must be high, not a beq only option is addi.	regwrite is 0 only sw fits

Solutions

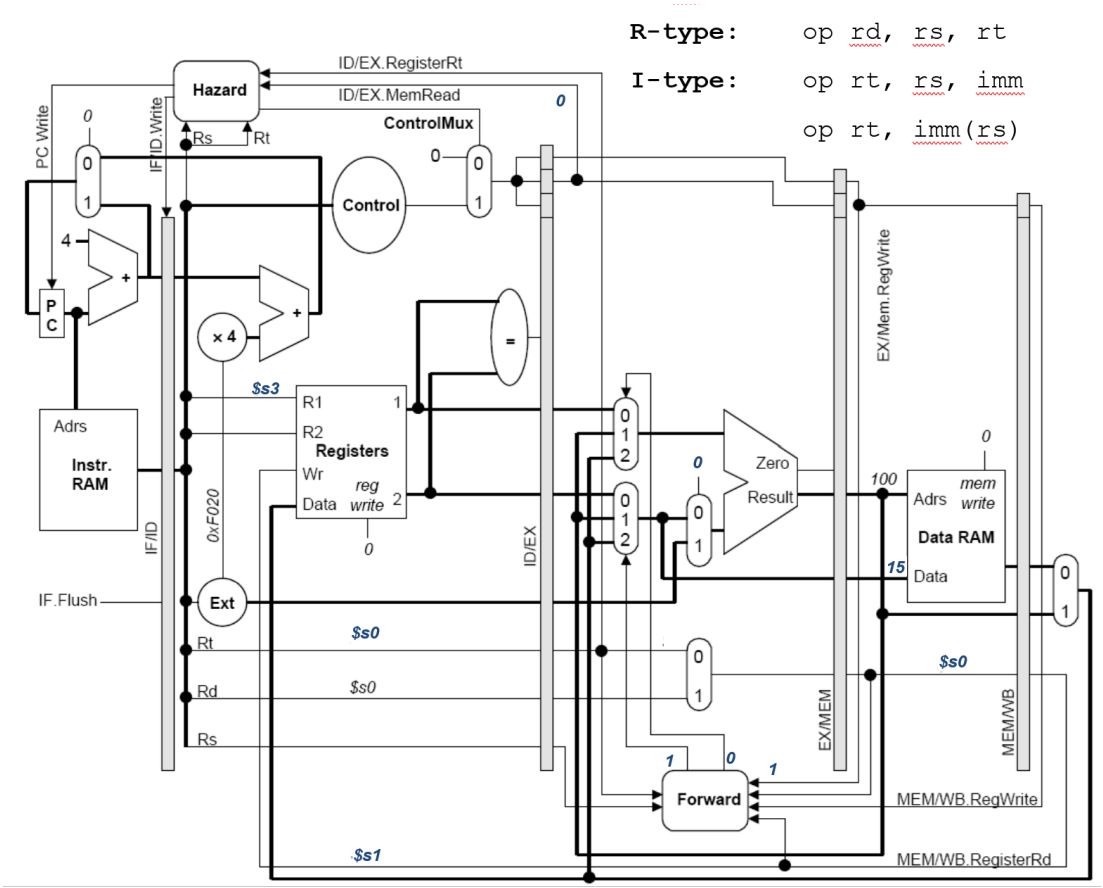
C



	ID Stage (3pts)	EX Stage (3pts)	ME Stage (Bonus)	WB Stage (Bonus)
	a) bne $s0, $s3, label	a) sub $t0, $s8, $t0	a) lw $s1, 100($t3)	a) add $ra, $s1, $t0
	b) bne $t0, $v0, label	b) xori $t1, $s3, 100	b) sw $t3, 100($s1)	b) and $s1, $t1, $t0
	c) bne $s0, $s4, label	c) andi $t1, $t0, 100	c) slti $t3, $s2, 100	c) sw $s7, 16($s3)
	d) bne $t0, $t1, label	d) ori $t0, $s3, 80	d) addi $t3, $s0, 4	d) lw $s1, -4($s0)
	e) add $s0, $s0, $s4	e) lw $t0, 4($s3)	e) beq $t3, $s2, loop	e) or $s1, $ra, $a3
Justify Your Selection	Select line for the mux before the PC is 0. This is a branch instruction with rs = s4 and rt = s0	Alu second input is the immediate field. Rt is t1.	memwrite = 0, not a sw adrs input is 100, not a slti since forwarding to ex stage, regwrite must be high, not a beq only option is addi.	regwrite is 0 only sw fits

Solutions

D



	ID Stage (3pts)	EX Stage (3pts)	ME Stage (Bonus)	WB Stage (Bonus)
	a) bne $s0, $s3, label	a) sub $t0, $t0, $s0	a) slti $s0, $s2, 100	a) add $ra, $s1, $t0
	b) bne $t1, $t0, label	b) xori $t0, $s0, 100	b) sw $s0, 100($s1)	b) or $s1, $t1, $t0
	c) bne $s0, $s4, label	c) andi $s0, $t0, 100	c) addi $s0, $s1, 8	c) lw $s1, 16($s0)
	d) bne $t0, $t1, label	d) ori $s0, $s3, 80	d) lw $s1, 100($s0)	d) and $s1, $ra, $a3
	e) add $s0, $s3, $s0	e) lw $t0, 4($s0)	e) beq $s0, $s2, loop	e) sw $t0, -8($s2)
Justify Your Selection	Select line for the mux before the PC is 0. This is a branch instruction with rs = s3 and rt = s0	ALU second input forwarded from MEM stage. Destination register in MEM is s0 (= rt). Alu second input is not immediate field.	memwrite = 0, not a sw adrs input is 100, not a slti since regwrite is high, not a beq only option is addi.	regwrite is 0 only sw fits