

Quiz #1

EEC 485, Fall 2008

Date: Sep. 16 (T)

Name:

1. Consider a 6-stage pipelined processor. Each stage takes 5.8ns, 2ns, 1.9ns, 3ns, 2.8ns and 3.8ns.
 - (a) Assuming that the pipeline is initially empty, calculate the execution time for a sequence of four instructions. (Hint: The slowest cycle determines the cycle time of a processor.)

 - (b) What is the maximum possible frequency for this processor?

 - (c) A designer found that the 5.8ns stage can be subdivided into two stages of 3ns and 2.8ns. What is the maximum possible frequency for the modified processor?

2. Answer "True" or "False" to the following statements on pipelined architecture.
 - (a) Pipeline hazards can mostly be resolved by waiting (stalling).

 - (b) Ideal speedup is equal to the number of pipeline stage.

 - (c) Ideal speedup cannot be achieved due to hazards.

 - (d) Pipelining helps reduce the execution time of an instruction.

 - (e) Structural hazards occur when two instructions in two different pipeline stages want to use the same resource at the same clock cycle. However, reading and writing on the same register (for example, reading and writing to \$t0) during the same clock cycle does not cause a hazard because writing is performed during the first half of the cycle and reading is performed during the second half of the cycle.

 - (f) Reading and writing on the different registers (for example, reading \$t0 and writing to \$s1) during the same clock cycle does not cause a hazard in the pipelined MIPS.

 - (g) Writing on two different registers (for example, writing to \$t0 and \$s1) during the same clock cycle does not cause a hazard in the pipelined MIPS.

 - (h) Reading from instruction memory and writing to data memory during the same clock cycle does not cause a hazard in the pipelined MIPS.

 - (i) Reading from data memory and writing to data memory during the same clock cycle does not cause a hazard in the pipelined MIPS.

Quiz #2

EEC 485, Fall 2008

Date: Sep. 23 (T)

Name:

1. The following sequence of MIPS instructions is executed in the pipelined architecture.

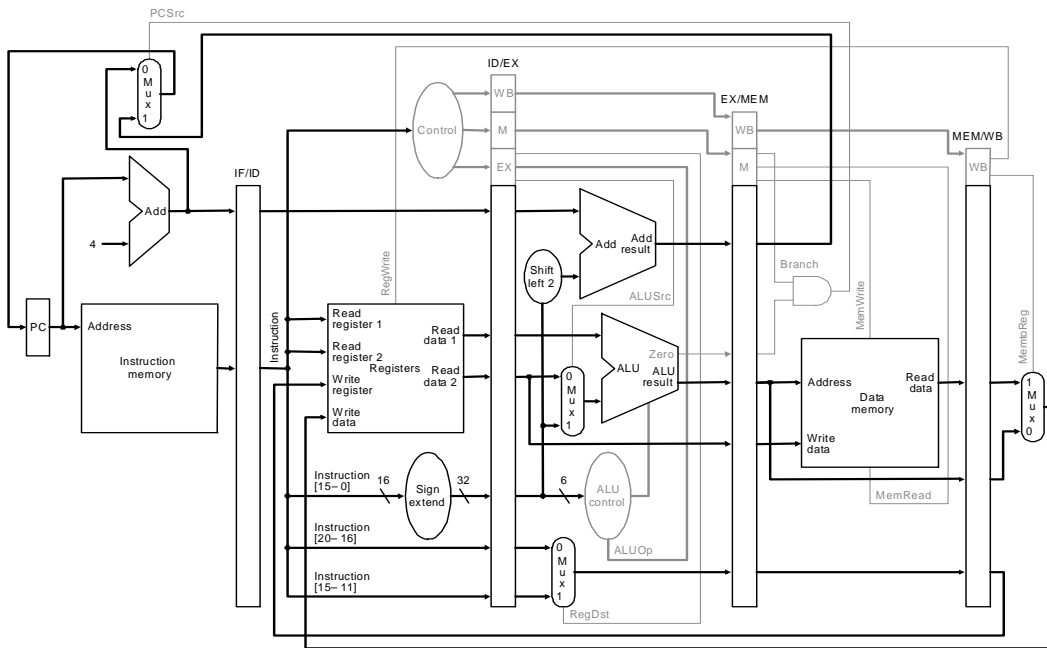
```

add  $8, $2, $3
sub  $10, $5, $6
lw   $7, 100($8)
sw   $7, 200($1)
or   $13, $14, $15
    
```

During the fifth cycle of execution, determine the followings.

ID/EX.Rs =
 ID/EX.Rt =
 EX/MEM.Rd =
 MEM/WB.Rd =

2. Consider the pipelined MIPS architecture and the control signals in the below.



Instruction	Execution/Address Calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Using the above control signal table, identify the value of control signals used in the pipeline at CC5 assuming that the following code sequence begins its execution at CC1.

lw	\$10, 20(\$1)
sw	\$11, 20(\$2)
sub	\$12, \$2, \$3
and	\$13, \$4, \$5
beq	\$14, \$8, Loop

- (a) RegDst at EX stage
- (b) ALUOp1 at EX stage
- (c) ALUOp0 at EX stage
- (d) ALUSrc at EX stage
- (e) Branch at MEM stage
- (f) MemRead at MEM stage
- (g) MemWrite at MEM stage
- (h) RegWrite at WB stage
- (i) MemtoReg at WB stage

Control signal table:

Instruction	Execution/Address Calculation stage control lines			Memory access stage control lines			Write-back stage control lines		
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Opcode Table

Instruction	Opcode/Function	Syntax	Instruction	Opcode/Function	Syntax
add	100000	ArithLog	slt	101010	ArithLog
addu	100001	ArithLog	sltu	101001	ArithLog
addi	001000	ArithLogI	slti	001010	ArithLogI
addiu	001001	ArithLogI	sltiu	001001	ArithLogI
and	100100	ArithLog	beq	000100	Branch
andi	001100	ArithLogI	bgtz	000111	BranchZ
div	011010	DivMult	blez	000110	BranchZ
divu	011011	DivMult	bne	000101	Branch
mult	011000	DivMult	j	000010	Jump
multu	011001	DivMult	jal	000011	Jump
nor	100111	ArithLog	jalr	001001	JumpR
or	100101	ArithLog	jr	001000	JumpR
ori	001101	ArithLogI	lb	100000	LoadStore
sll	000000	Shift	lbu	100100	LoadStore
sllv	000100	ShiftV	lh	100001	LoadStore
sra	000011	Shift	lhu	100101	LoadStore
srav	000111	ShiftV	lw	100011	LoadStore
srl	000010	Shift	sb	101000	LoadStore
srlv	000110	ShiftV	sh	101001	LoadStore
sub	100010	ArithLog	sw	101011	LoadStore
subu	100011	ArithLog	mfhi	010000	MoveFrom
xor	100110	ArithLog	mflo	010010	MoveFrom
xori	001110	ArithLogI	mthi	010001	MoveTo
lhi	011001	LoadI	mtlo	010011	MoveTo
llo	011000	LoadI	trap	011010	Trap

* The second column tells the function code for R-type instructions and opcode for other types of instructions. ArithLog in the last column denotes R-type instruction. Note that opcode for R-type instruction is 0. Note also that ArithLogI denotes I-type instruction.

- (a) Instruction in IF stage:
- (b) Instruction in ID stage:
- (c) Instruction in EX stage:
- (d) Instruction in MEM stage:
- (e) Instruction in WB stage:

IF/ID.Write

PCWrite

(c) Stalling the pipeline until a branch is complete is too slow. A common improvement over branch stalling is to assume that the branch will not be taken and thus continue execution down the sequential instruction stream. If the branch is taken, the instructions that are being fetched and decoded must be discarded.

Control signals such as () can be used for that purpose.

(d) Another form of control hazard involves exceptions because control must be transferred to the exception handling routine starting at 0x4000 0040 immediately after this instruction because we would not want the outcome of the exception to contaminate other registers or memory locations. In other words, we need to flush the instructions that follow the offending instruction from the pipeline and begin fetching instructions from the new address. For example, an arithmetic overflow exception is detected in EX stage and thus, we need to flush instructions in IF, ID, and EX stages.

Control signals such as () can be used for that purpose.

Quiz #4

EEC 485, Fall 2008

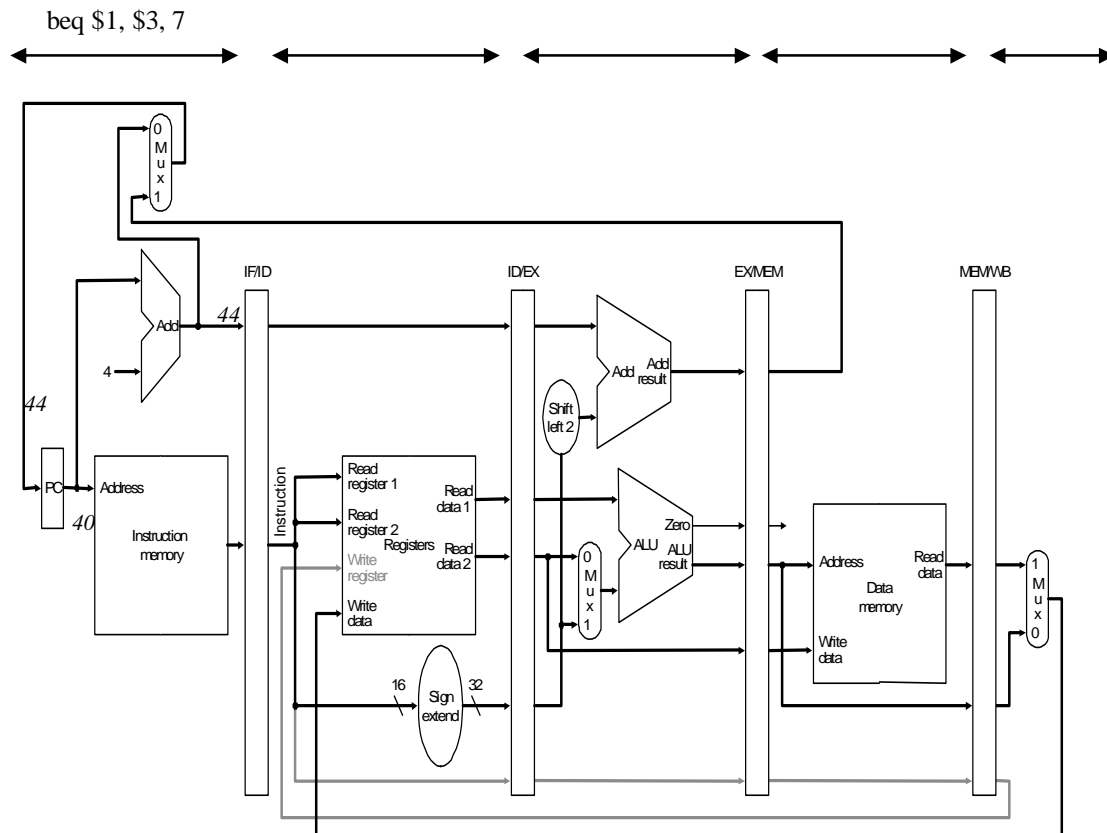
Date: Oct. 7 (T)

Name:

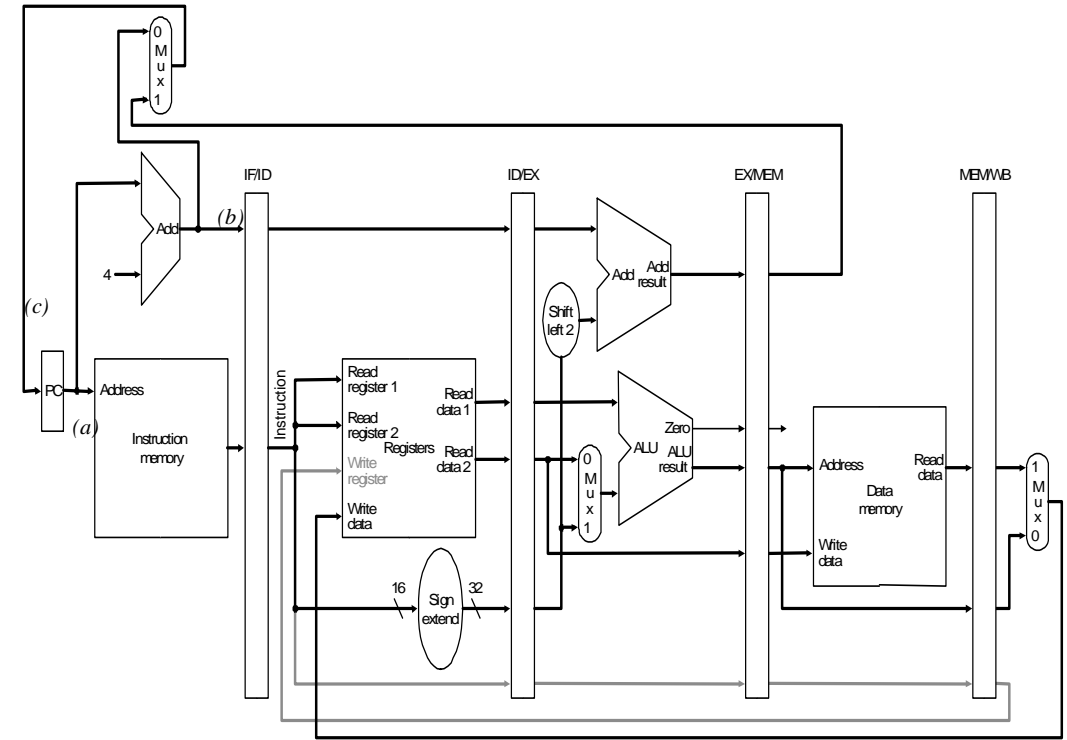
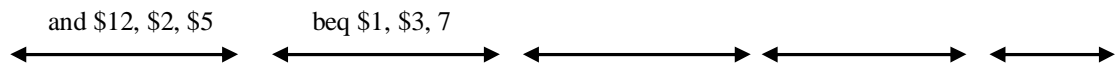
1. The following sequence of code is executed in two implementations of 5-stage pipelined MIPS: One without considering the branch hazards and the other one with modifications. The first 5 pages show the 5 cycles of execution of the first implementation, and the next 3 pages show the 3 cycles of execution of the second implementation. Identify addresses on the (a)-(x) at the end of each cycle. Note that “beq” instruction in memory address 40 is taken and the addresses shown are decimal numbers. (Hint: Branch target address is calculated by $40 + 4 + 7 \ll 2 (=28) = 72$.)

40 beq \$1, \$3, 7
 44 and \$12, \$2, \$5
 48 or \$18, \$6, \$2
 52 add \$14, \$2, \$2
 ...
 72 lw \$4, 50(\$7)

Original implementation: Cycle 1

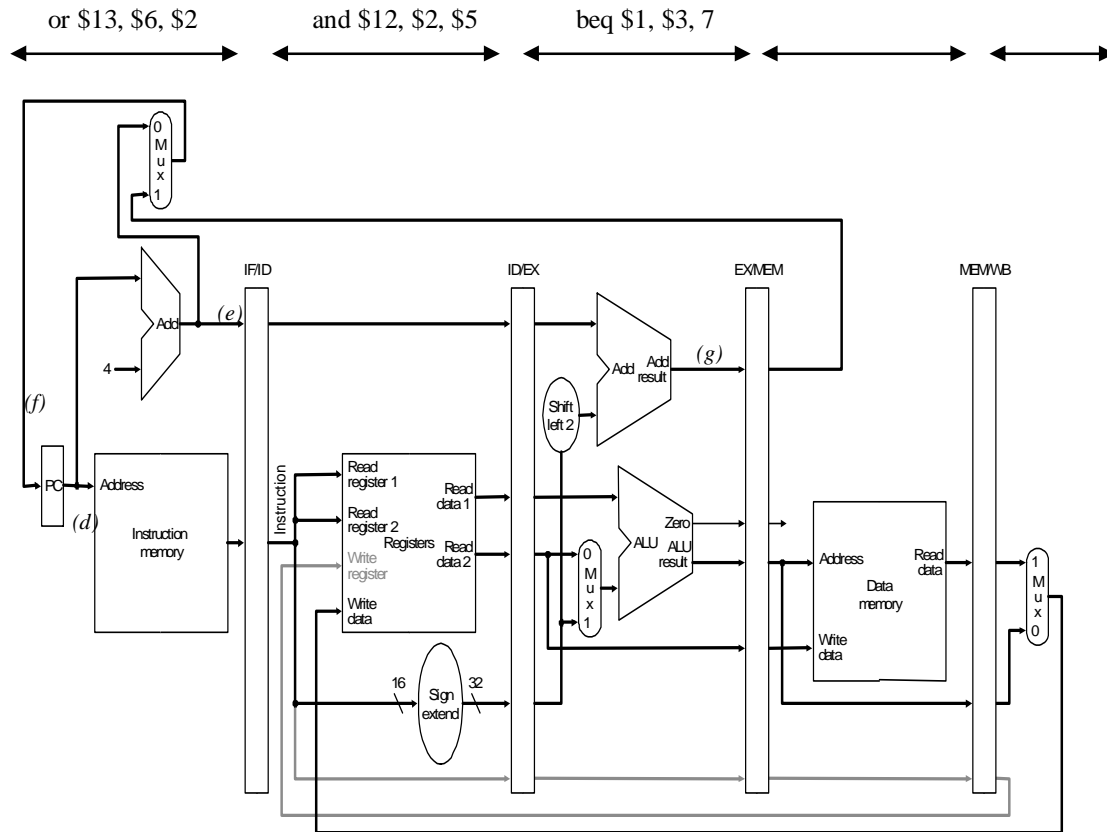


Original implementation: Cycle 2



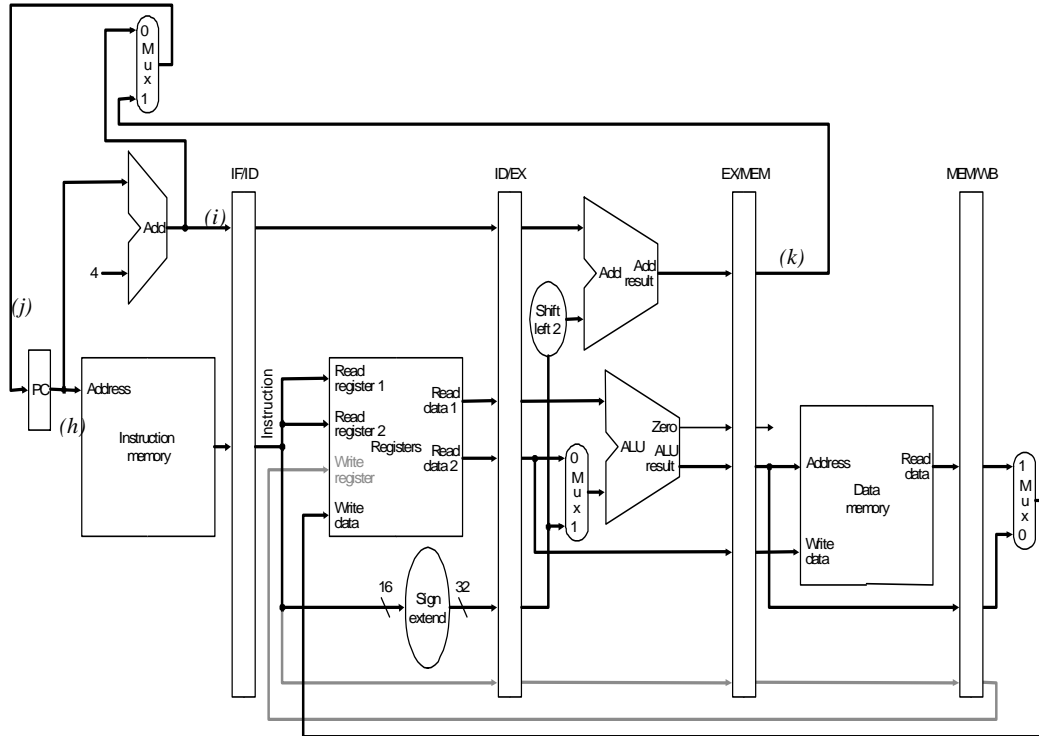
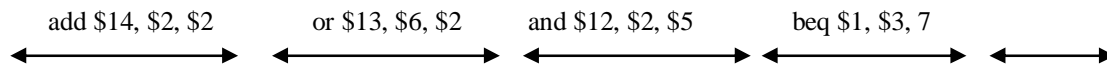
- (a) _____
- (b) _____
- (c) _____

Original implementation: Cycle 3



- (d) _____
- (e) _____
- (f) _____
- (g) _____

Original implementation: Cycle 4



(h) _____

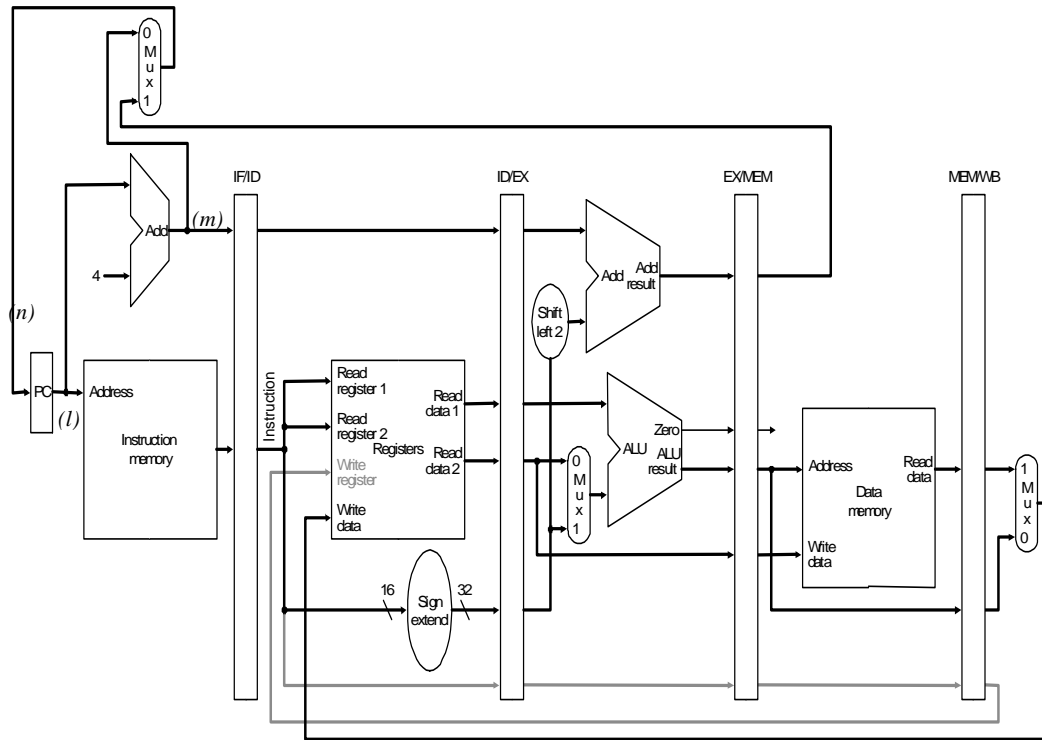
(i) _____

(j) _____

(k) _____

Original implementation: Cycle 5

← lw \$4, 50(\$7) ← add \$14, \$2, \$2 ← or \$13, \$6, \$2 ← and \$12, \$2, \$5 ← beq \$1, \$3, 7



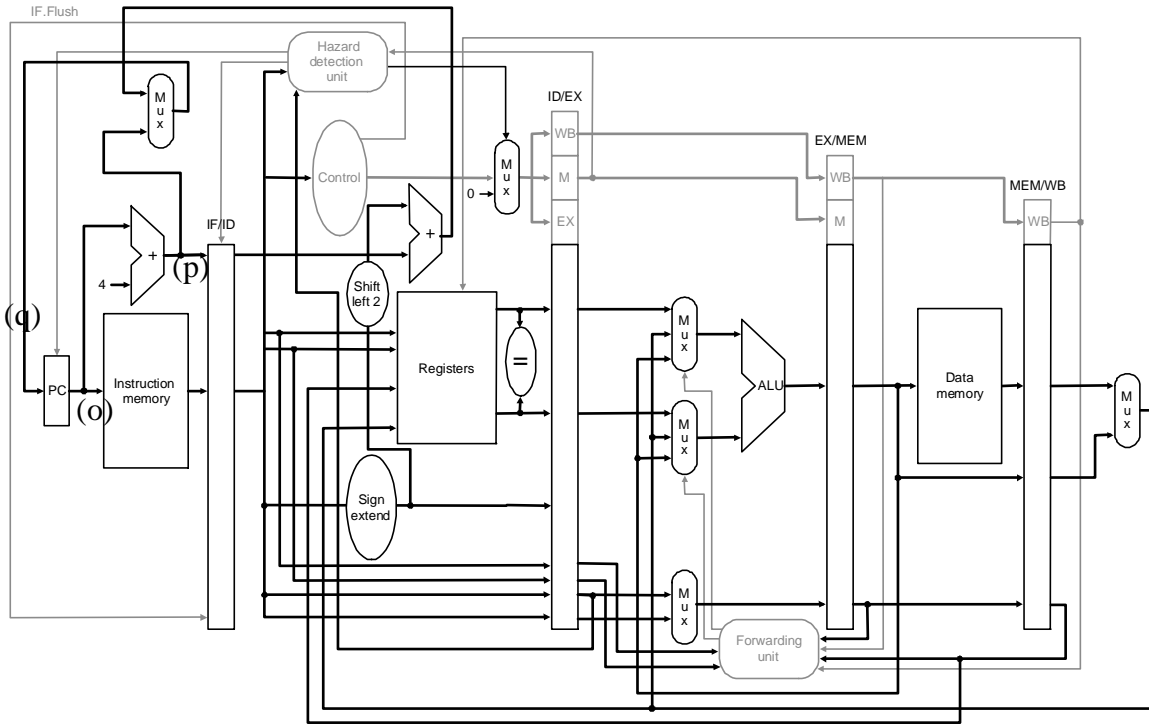
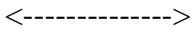
(l) _____

(m) _____

(n) _____

New implementation: Cycle 1

beq \$1, \$3, 7



(o) _____

(p) _____

(q) _____

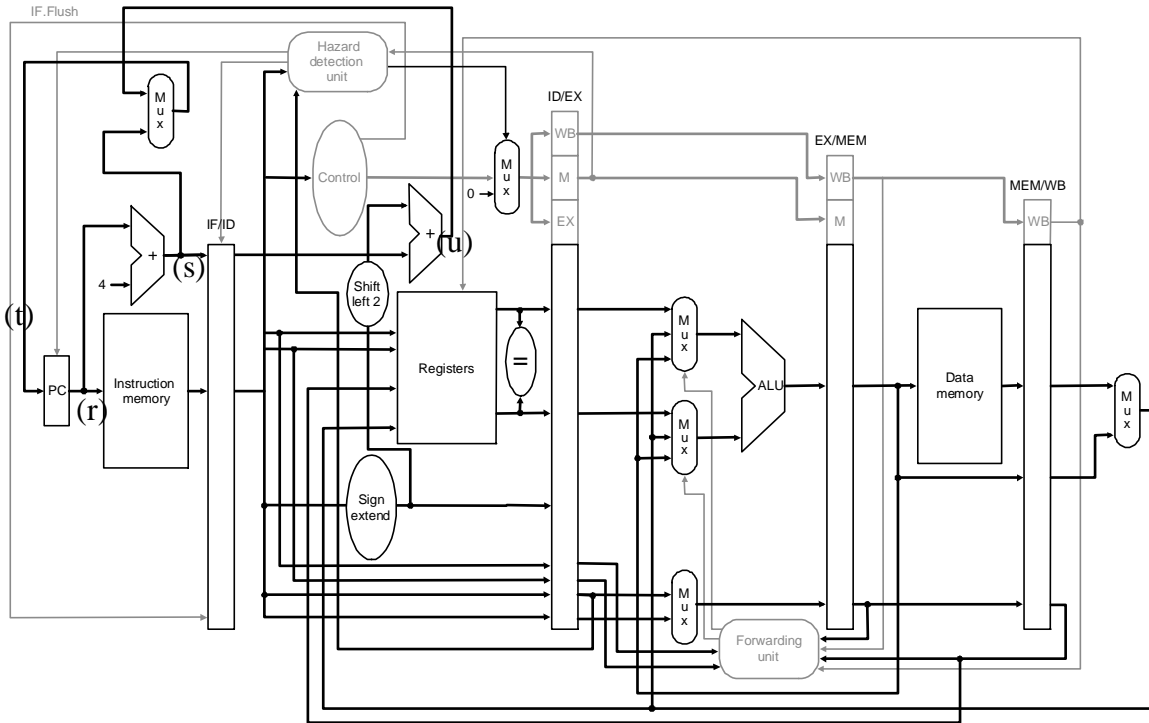
New implementation: Cycle 2

and \$12, \$2, \$5

beq \$1, \$3, 7

<----->

<----->



(r) _____

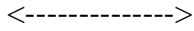
(s) _____

(t) _____

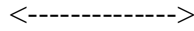
(u) _____

New implementation: Cycle 3

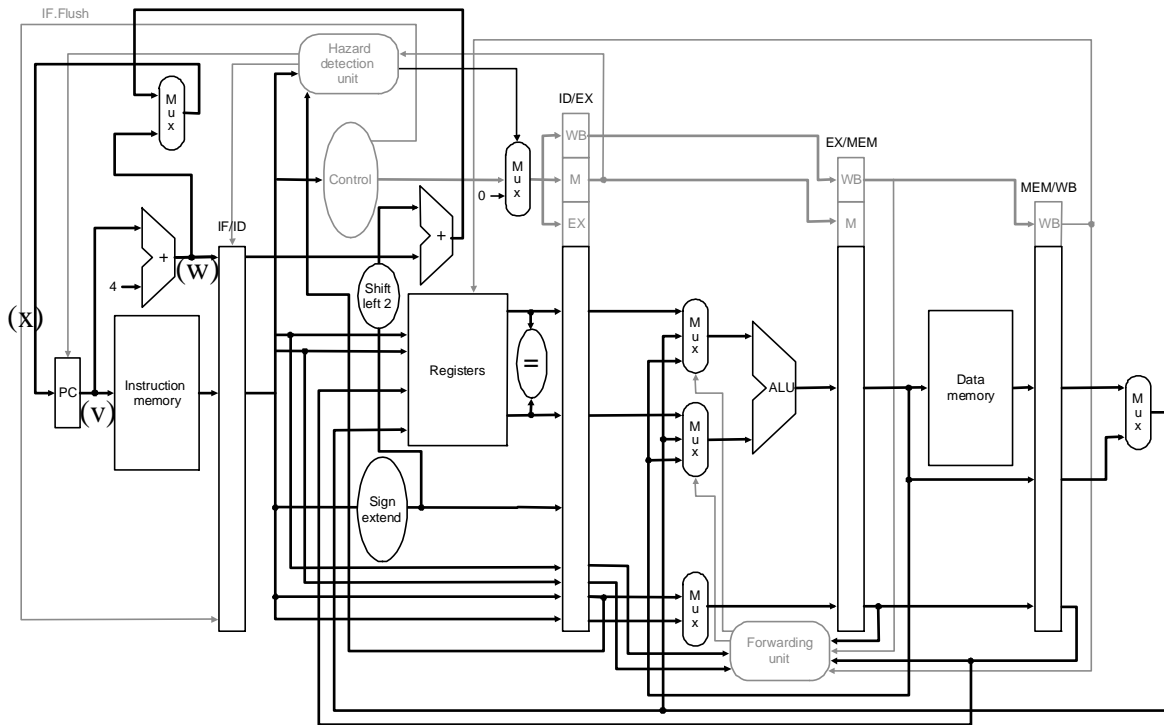
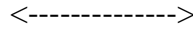
lw \$4, 50(\$7)



and \$12, \$2, \$5



beq \$1, \$3, 7



(v) _____

(w) _____

(x) _____