New Paltz
STATE UNIVERSITY OF NEW YORK
Division of Engineering Programs

EGC442 Introduction to Computer Architecture

Test #2

First Name: Key **Last Name:** ______________________

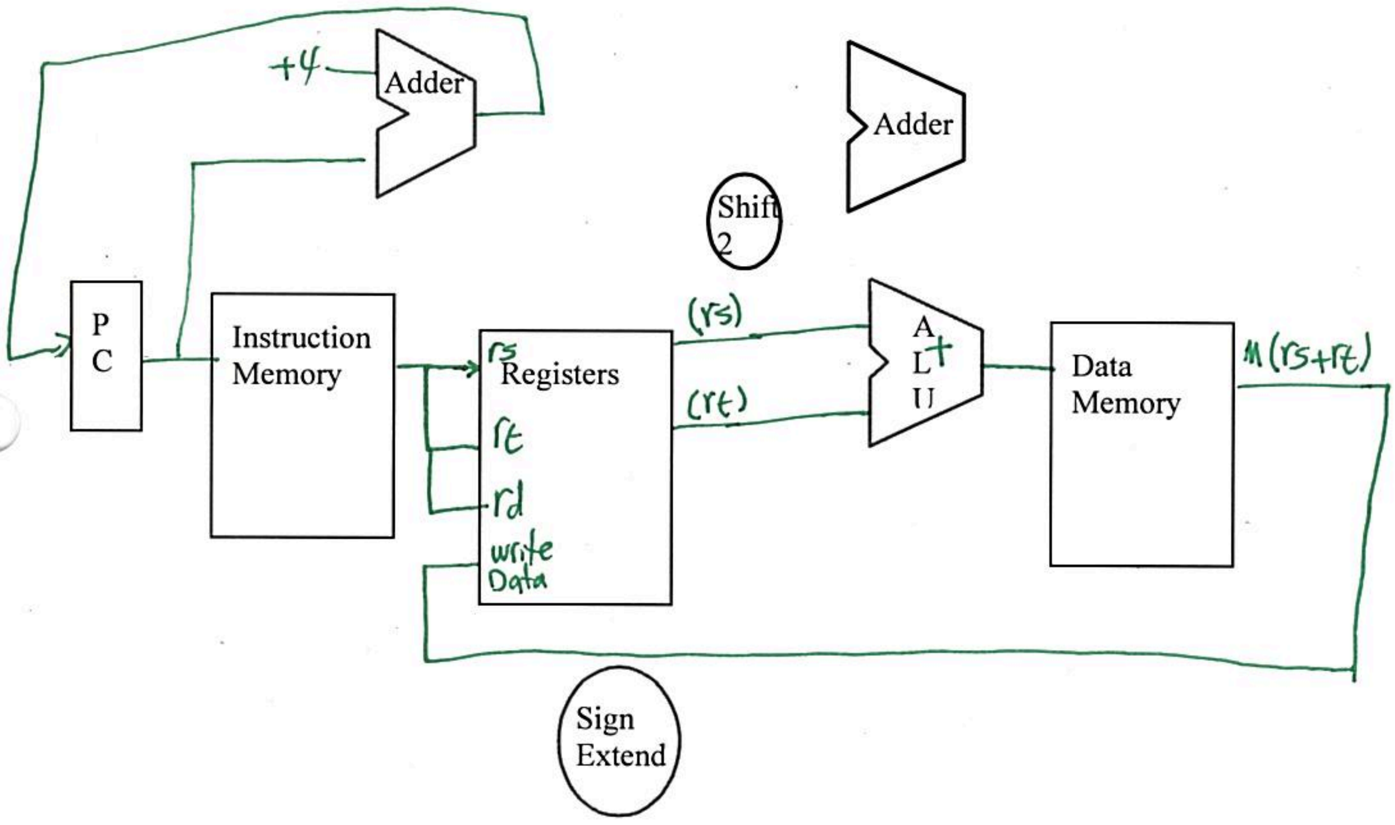
- For full credit you need to show complete work and answer the questions as directed.
- Your submission must be in a single PDF file
- Make sure you submit before the deadline of 12:15 PM. I will not accept late submission by email.
- You must adhere to the honor code. Any evidence of misconduct will be dealt with strictly per syllabus.

Question 1 (20 points)

Complete only the necessary (this includes updating the PC for the next instruction) data path to completely execute `lwr rd, rt(rs)` instruction, where *rs*, *rt*, and *rd* are taken from "instruction[25-21]"), ("instruction[20-16]", and ("instruction[15-11]", respectively, The instruction will result in copying the contents of memory pointed by *rs* + *rt* to register *rd*. For full credit, your diagram needs to be detailed. You may add additional gates and other devices such as multiplexers, if needed. You also need to cross out any functional unit that is not used by the instruction.

lwr rd, rt(rs)

write read read



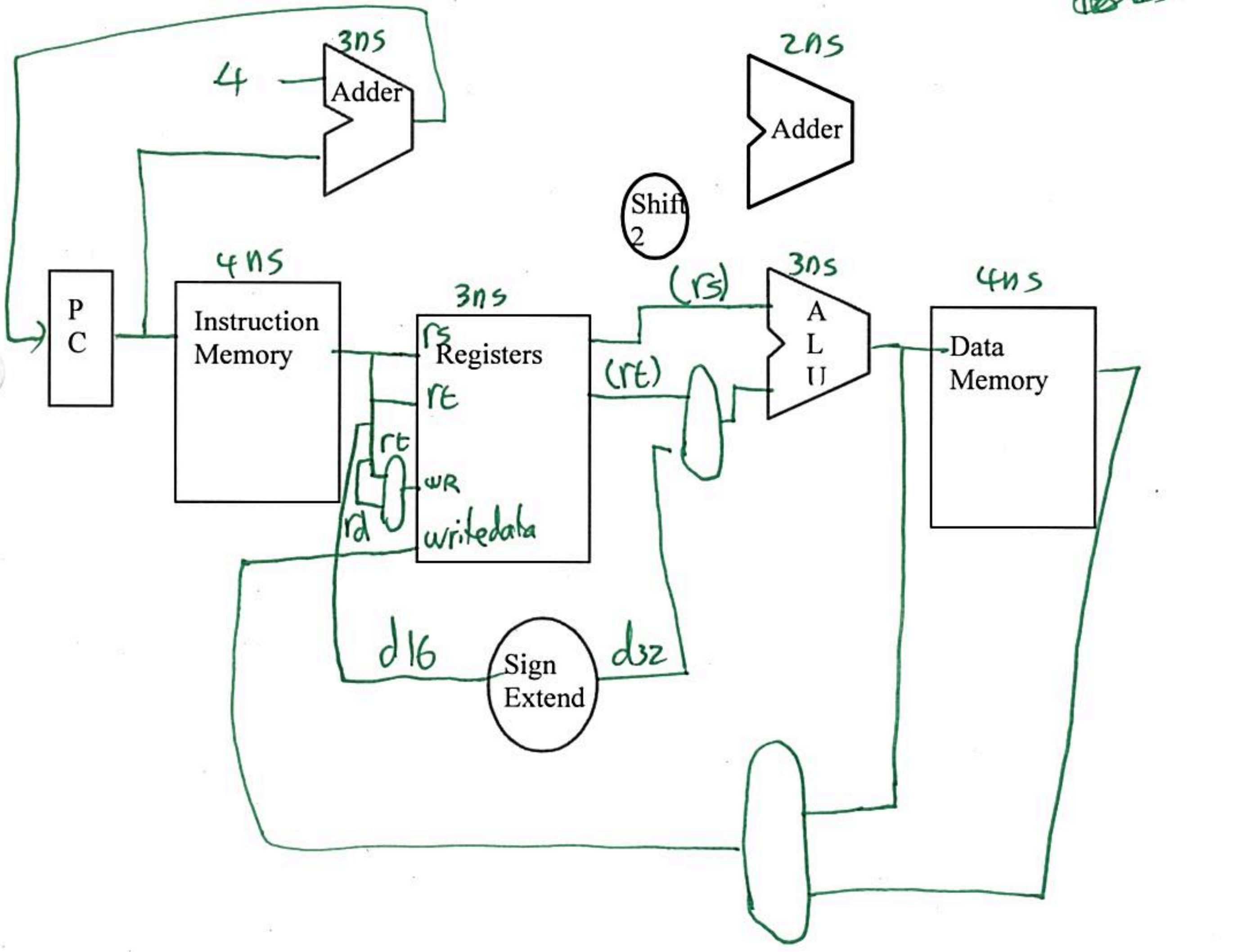
Question 2 (20 points)

Highlight <u>only</u> the active data paths and calculate the cycle time for a single cycle processor which can execute *sub* and *lw* instructions only. Assume negligible delays except 4ns for each memory access, 3ns for ALU, 2ns for each adder, and 3ns for register read and write. Justify your answer.

sub rd, rs, rt — rd: write, rs: read, rt: read   lw rt, d16(rs) — rt: write, rs: read

Instruction	Instr. Memory	Register Read	ALU Op.	Data Memory	Adder	Reg. Write	Total
lw	4ns	3ns	3	4ns	—	3ns	~~18ns~~ 17ns
sub	4ns	3ns	3	—	—	3ns	~~[illegible]~~ 13ns

~~18ns~~



Question 3 (15 points)

For the code below,

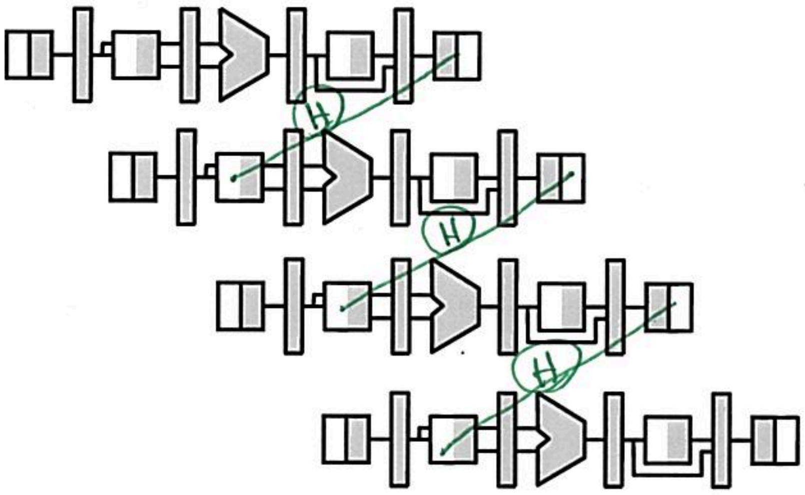
a. On the diagram, mark and identify all the data dependencies in the code given below and identify which dependencies will cause data hazards without forwarding hardware.

```
lw   $t4, 0x10($s2)

add  $s2, $t4, $t6

sub  $t4, $t3, $s2

and  $t5, $t4, $t6
```



b. Assuming there is no special hardware that is added for forwarding. Add "nop" instructions to the code to avoid the data hazards.

```
lw    $t4, 0x10($s2)
nop
nop
nop
add   $s2, $t4, $t6
nop
nop
nop
sub   $t4, $t3, $s2
nop
nop
nop
and   $t5, $t4, $t6
```

c. How many clock cycles does it take to execute the code in part b.

17

Question 4 (15 points)

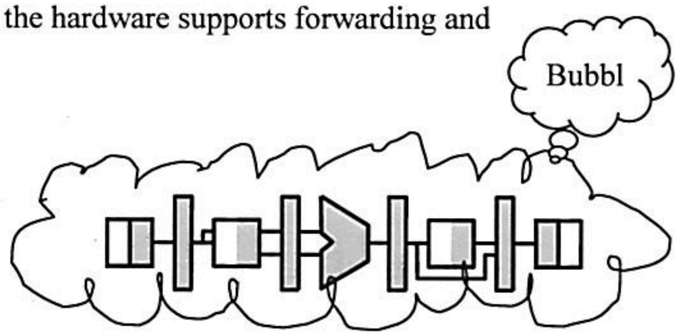
For the following sequence of instructions, assume that the hardware supports forwarding and stalling.

lw *\$t4, 10(\$s2)*

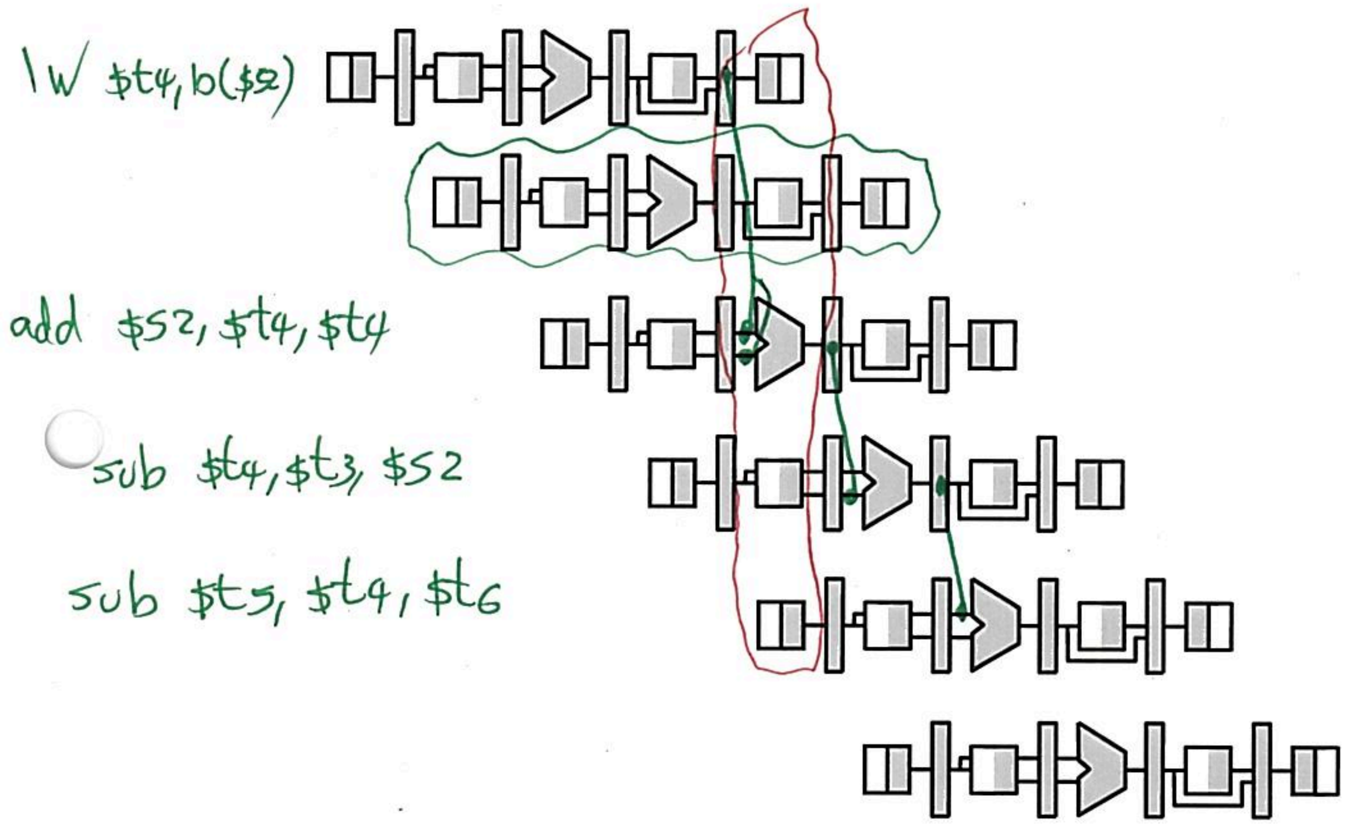
add *\$s2, \$t4, \$t4*

sub *\$t4, \$t3, \$s2*

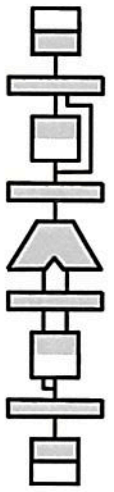
and *\$t5, \$t4, \$t6*



a. Show proper data forwarding and stall. Indicate from which pipe the data is taken from and where it is forwarded.



b. Indicate what each stage will do during the 5th clock cycle.



→ update (write) $t4

→ nop

→ add $t4 & $t4

→ read $t3, & $s2

→ upcode fetch sub

c. How many cycles will it take to execute this code using forwarding?

9

Question 5 (15 points)

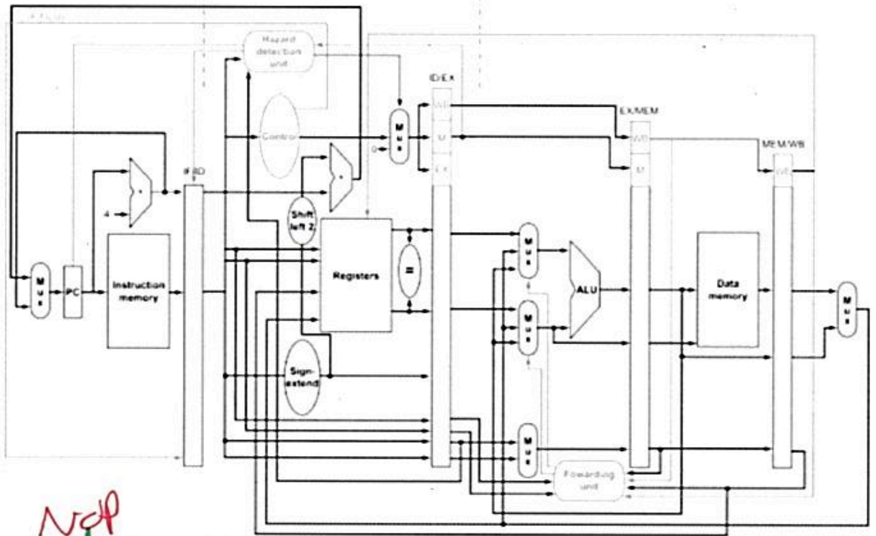
Assume the following sequence of instructions.

```
32   sub   $10, $4, $8
36   and   $4, $8, $6
40   bne   $2,  $4, 2
44   add   $8, $2, $5
48   xor   $9, $2, $6
52   lw    $11, 0x4($2)
56   slt   $15, $6, $7
```

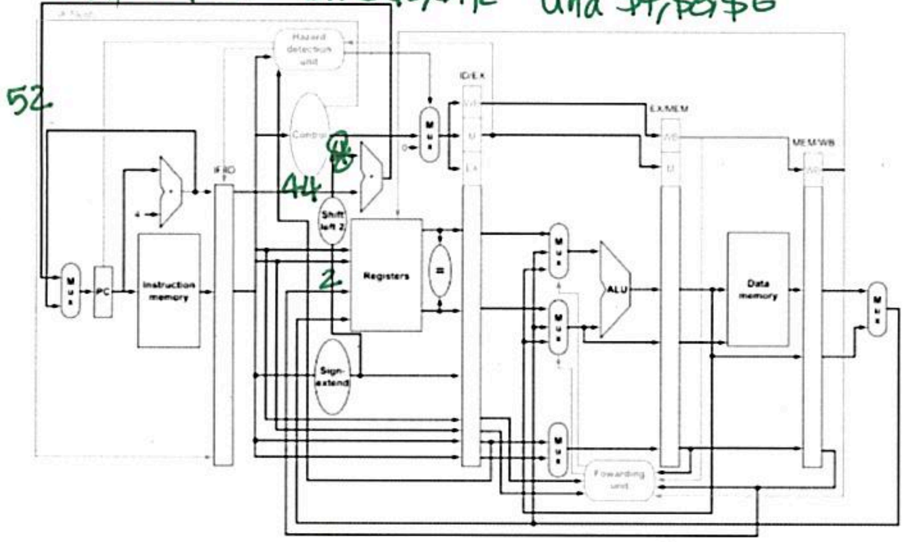
a. Using the following diagram, assuming ($2)= 0x26 ($4)= 0x3E, complete the next four cycling steps in detail.

NE

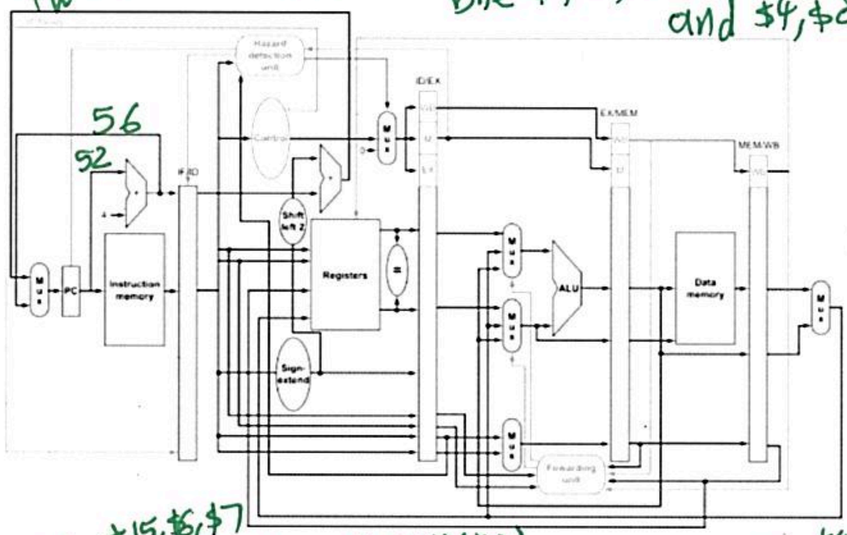
bne $2,$4,2 *and $4, $8, $6*



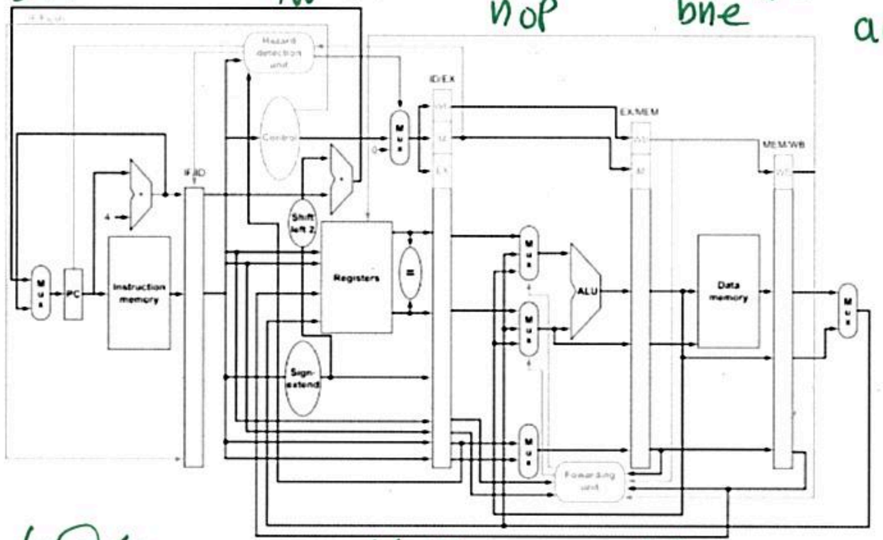
Nop
~~add $8, $2, $5~~ bne $2,$4,2 and $4,$8,$6



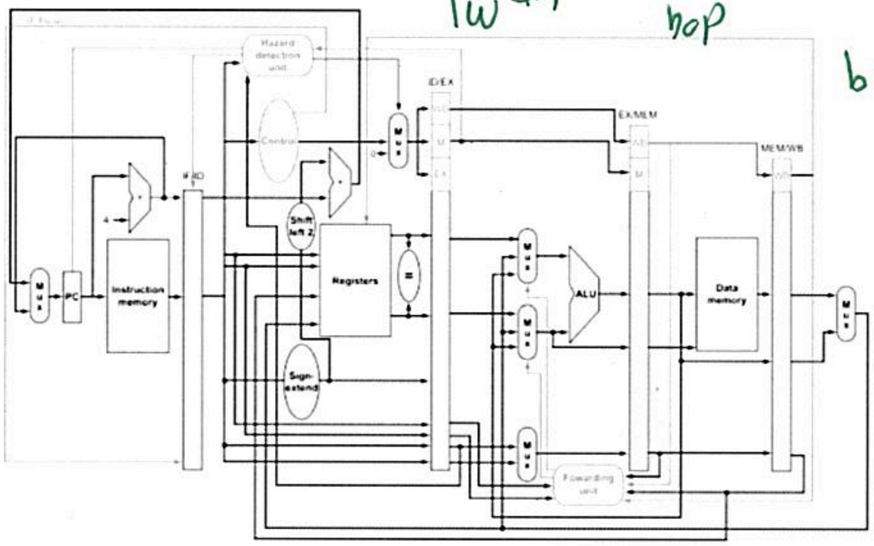
lw $11, 0x4($2) nop bne $2, $4, 2 and $4, $8, $6



slt $15, $6, $7 lw $11, 0x4($2) nop bne $2, $4 2 and $4, $8, $6



Instr @ 60 slt $15, $6, $7 lw $11, 0x4($2) nop bne $2, $4, 2



Problem 6 (15 Points)

1) Given this instruction sequence,

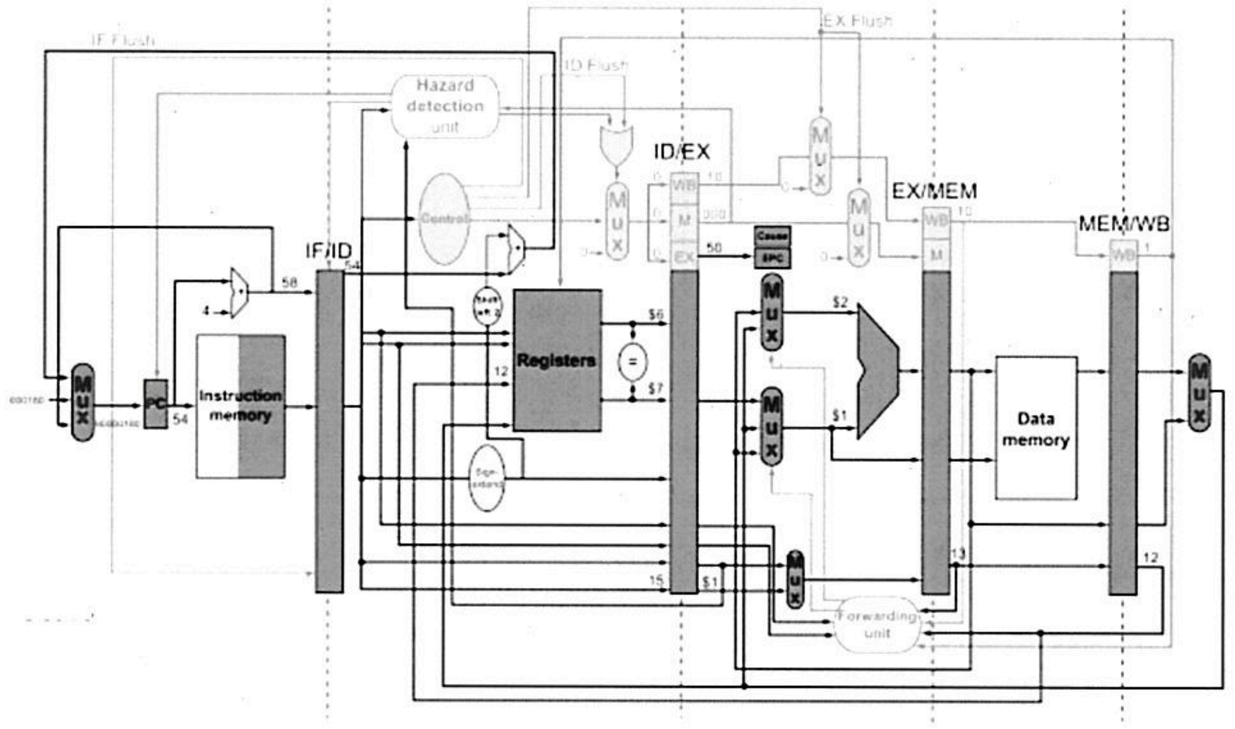
32	*sub*	*$10, $4, $8*
36	*add*	*$4, $8, $6*
40	*bne*	*$2, $4, 2*
44	*nor*	*$8, $2, $5*
48	*and*	*$9, $2, $6*
52	*lw*	*$11, 0x4($2)*
56	*slt*	*$15, $6, $7*

Assume the instructions to be invoked on an exception begin like this:

```
80000180hex   sw      $26, 1000($0)
80000184hex   sw      $27, 1004($0)
...
```

Show what happens in the pipeline if an overflow exception occurs in the *add $4, $8, $6* instruction.

bne $2, $4, 2 *add $4, $8, $6* *sub $10, $4, $8*

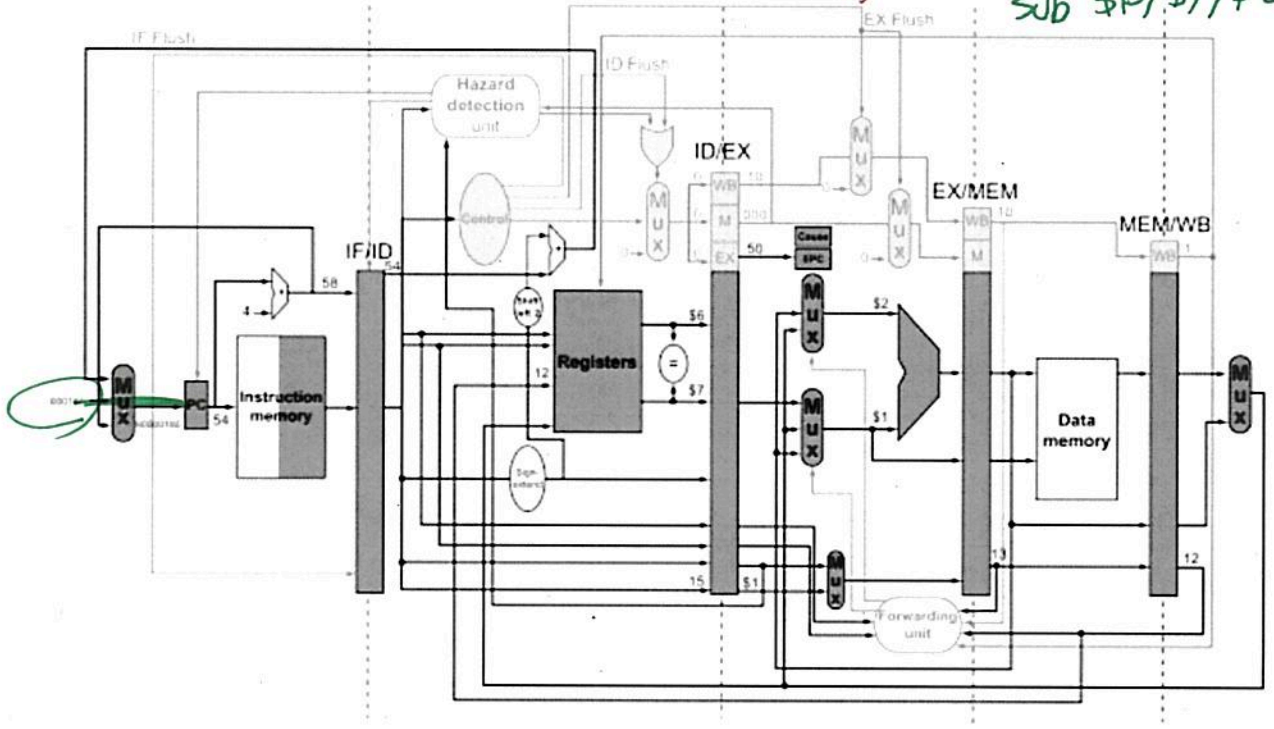


nop
nor ~~$8,~~$2,$5

nop
bne ~~$2,~~$4,2

nop
~~add $4,$8,$6~~

sub $10,$4,$8



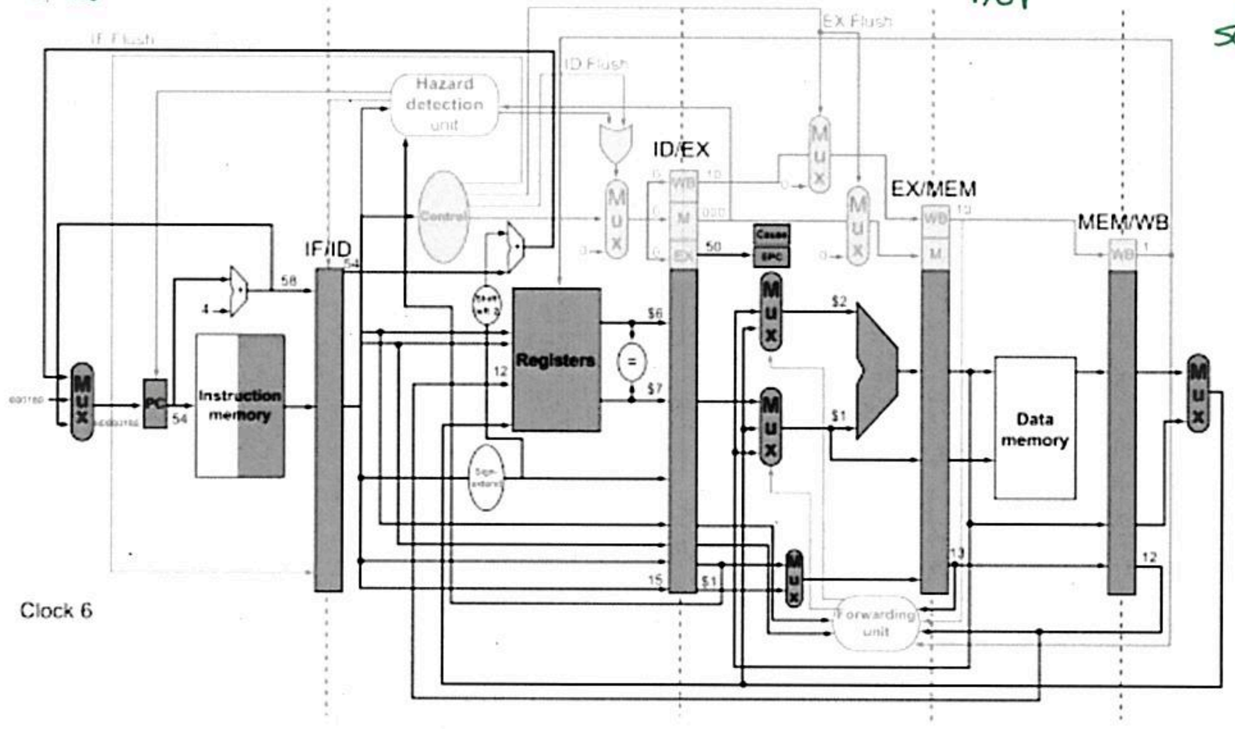
sw $26,1000($0)

nop

nop

nop

sub $10,$4,$8



Clock 6