Name: (as it would appear on official course roster)	
UCSB email address:	**@ucsb.edu**
Lab Section Time:	
Optional: name you wish to be called if different from above	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone")	

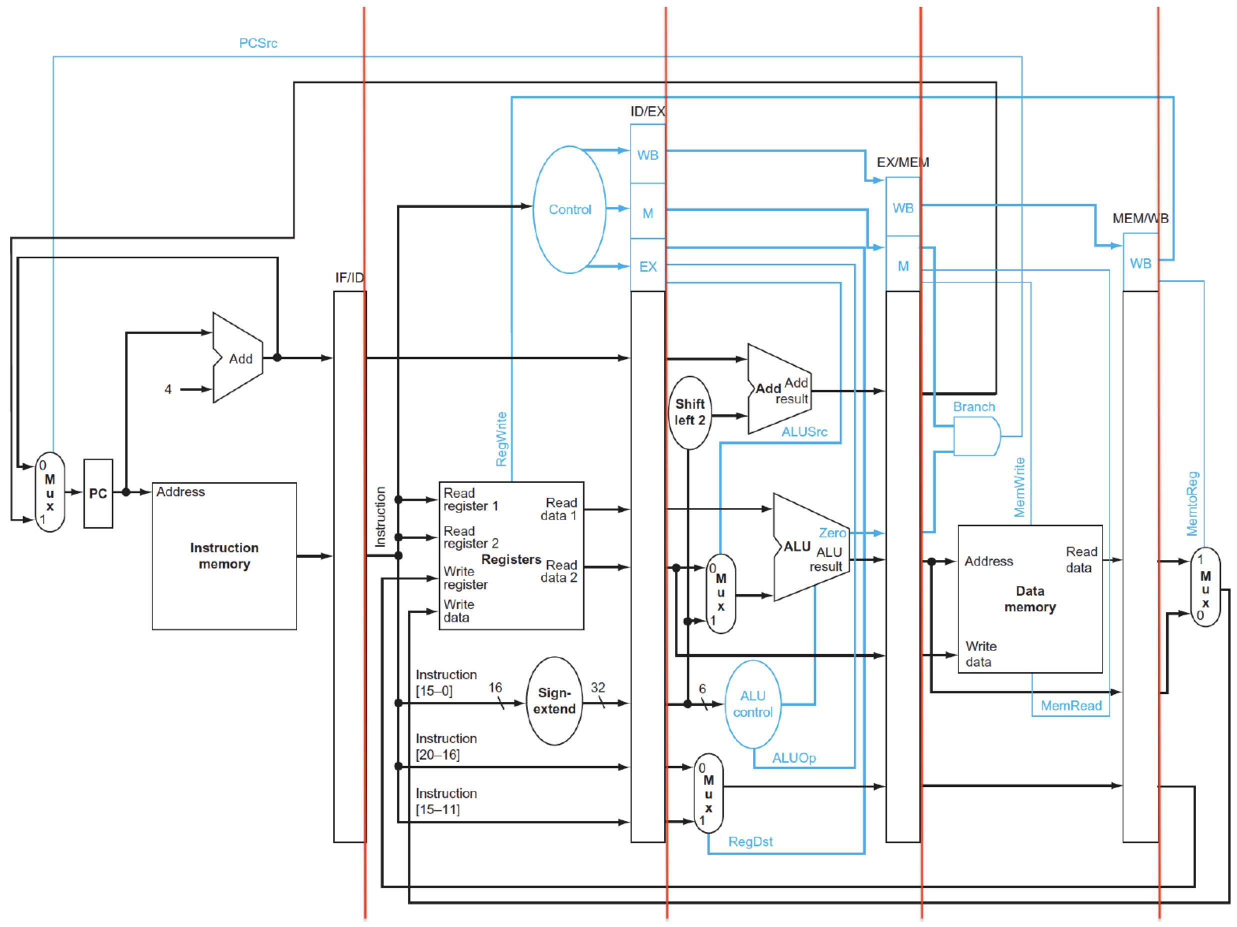
Assignment 08: Pipelined CPU Architecture

Assigned: *Friday, March 6th, 2020*
Due: *Thursday, March 12th, 2020*
Points: *100*

- You may collaborate on this homework with AT MOST one person, an optional "homework buddy".
- MAY ONLY BE TURNED IN ON **GRADESCOPE as a PDF file** (see instructions in online lab01 description).
- There is NO MAKEUP for missed assignments.
- We are strict about enforcing the LATE POLICY for all assignments (see syllabus).

Only use the space provided for answers. Use clear and clean handwriting (or typing). You will get penalized if you are asked to show your calculations and do not do so.
<u>ALWAYS SHOW YOUR WORK!</u>

Consider the data path figure given below (it's also reproduced on the last page, only bigger)



Name:
(as it would appear on official course roster)

Consider the following sequence of 7 instructions for the pipelined MIPS. Assume the pipeline is initially empty. There is an enlarged figure on the next page which you will have to use for question 4 below.

```
lw $t0, 0($s0)
lw $t1, 4($s0)
lw $t2, 8($s0)
addi $t0, $t0, 1
add $t1, $t1, $t2
sw $t0, 0($s0)
sw $t1, 4($s0)
```

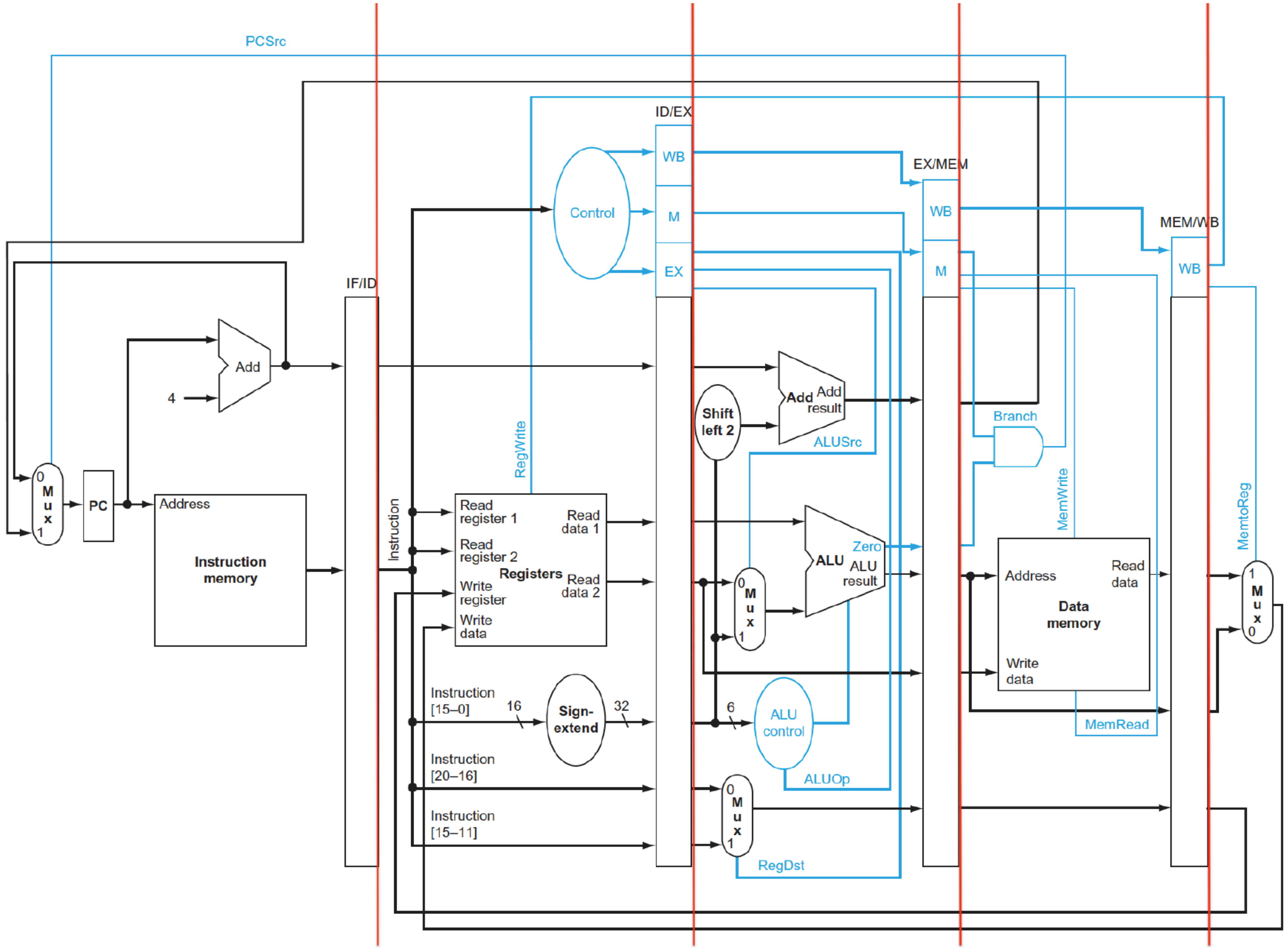
1. How many clock cycles are needed to push these instructions through the CPU? Assume NO forwarding and NO stalling (not a realistic scenario, but just figure out the "minimum" number of clock cycles needed).

2. List all hazards **and** their types in this code, if any. For each hazard, say how it will be addressed/fixed (i.e. forwarding, or something else, etc…).

3. Given your answer in part 2 above, how many *actual* clock cycles are needed to push these instructions through the CPU?

4. Show the values of **ALL data**, **address**, and **control line/bus** values as the instructions follow through the stages of pipeline using the figure (i.e. for every clock cycle). Use a copy of the ***enlarged figure*** on the last page and make as many copies as needed (how many copies will be needed?). Follow the format used in **Figure 4.62**, on **Page 320** in the book and write the answers neatly on the page. Show the data and address lines more compactly using hexadecimal values. The control lines/buses of interest here are shown in the figure.
   They are:
   PCSrc, RegWrite, ALUSrc, ALUOp, RegDst, Zero, Branch, MemWrite, MemRead, MemtoReg

5. Draw a multiple-clock-cycle diagram (similar to **Figure 4.52**, on **Page 305** in the book) to point out the cycles and units through which **forwarding** happens.


PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
WB
M
EX
Control
Add
4
Mux
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
Instruction [15–0]
16
Sign-extend
32
Instruction [20–16]
Instruction [15–11]
Shift left 2
Add
Add result
ALUSrc
Branch
ALU
Zero
ALU result
6
ALU control
ALUOp
RegDst
MemWrite
MemRead
Read data
Data memory
Write data
MemtoReg