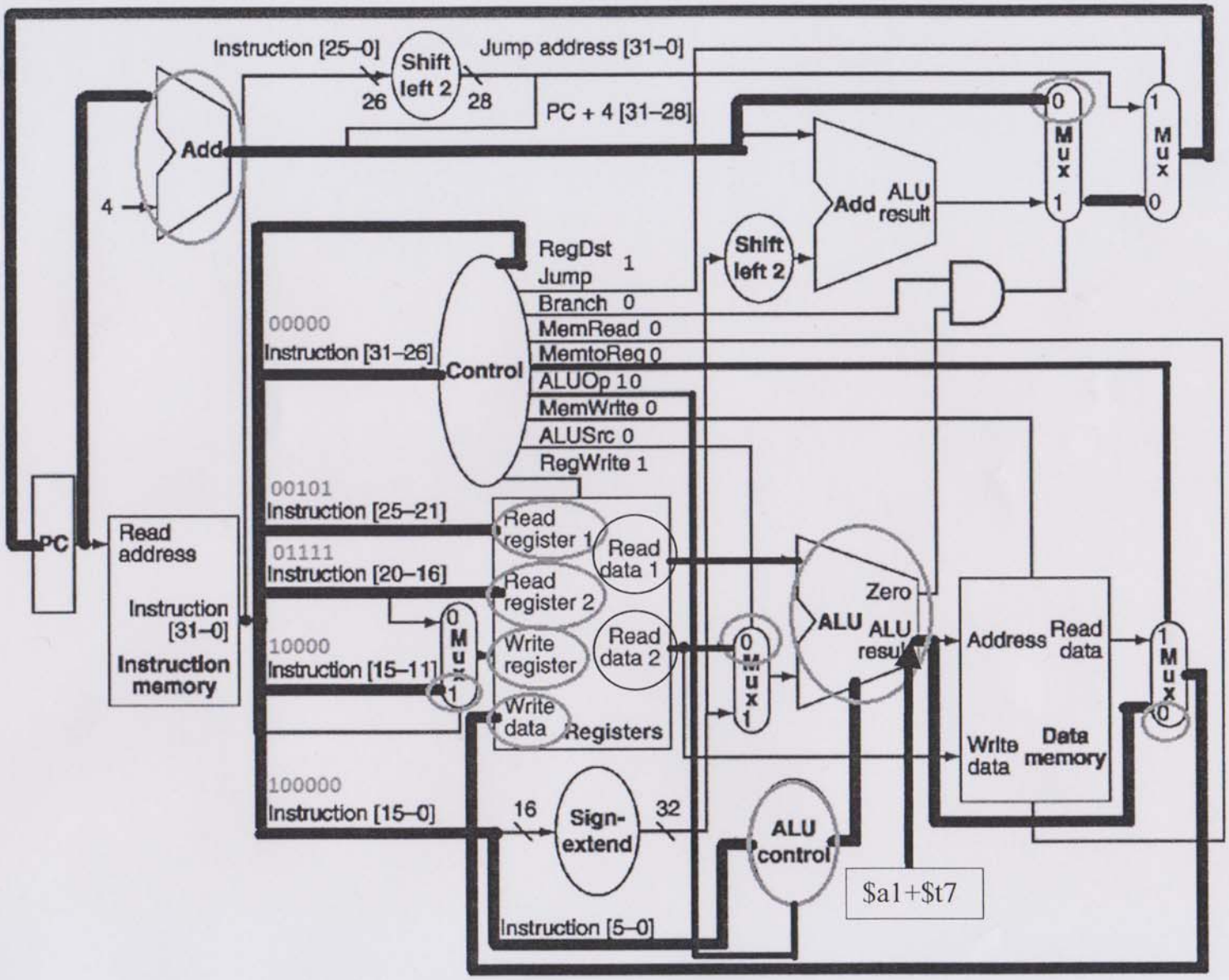
CSC 390 : HW 10

Q1)

add $s0, $a1,$t7

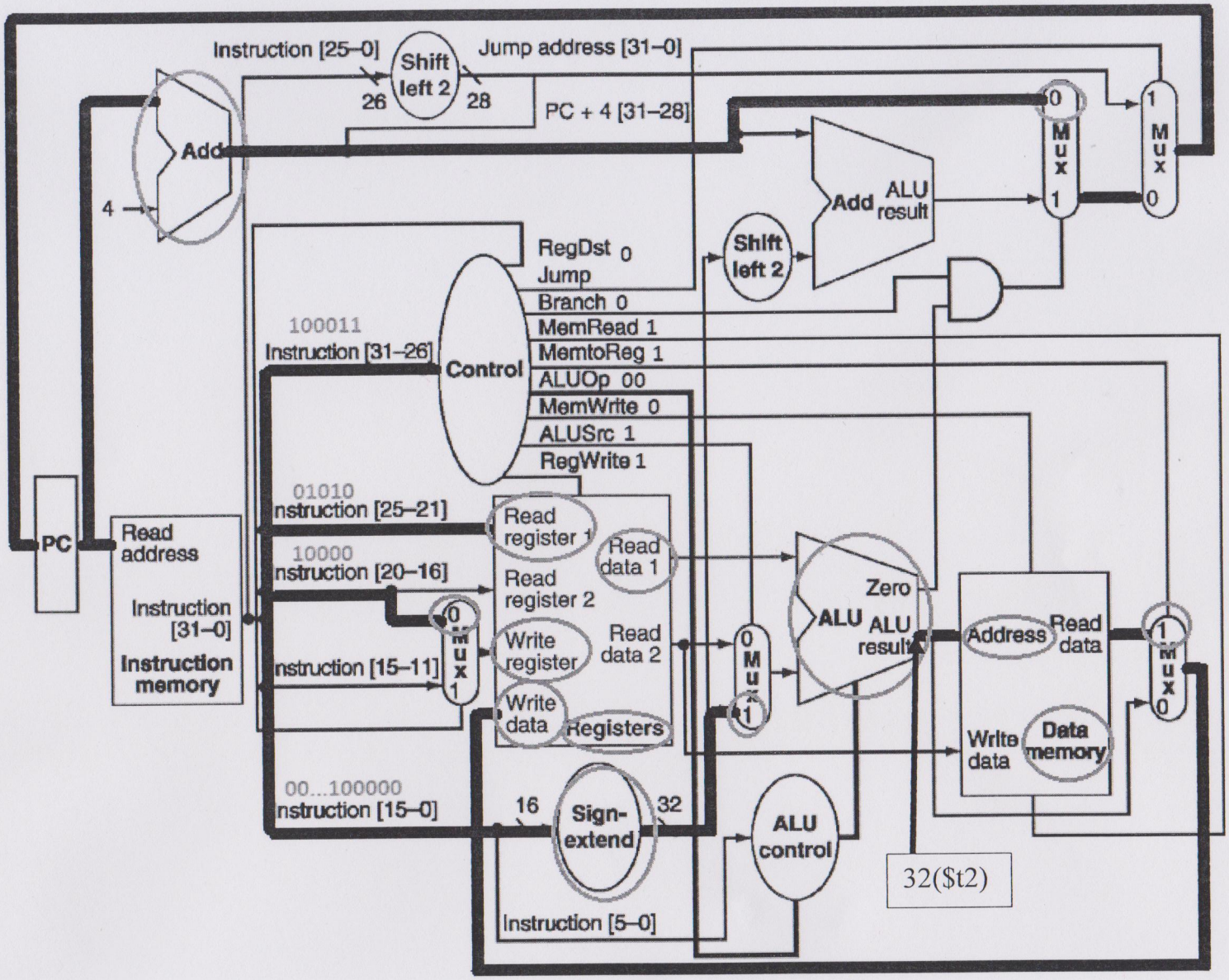
The result from the ALU is written into the register file using bits 15:11 of the instruction to select the destination register ($s0)



Q2)

lw $s0, 32($t2)

+ The ALU computes the sum of the value read from the register fle and the sign-extended, lower 16 bits of the instruction (offset).
+ The sum from the ALU is used as the address for the data memory.
+ The data from the memory unit is written into the register fle; the register destination is given by bits 20:16 of the instruction ($s0).

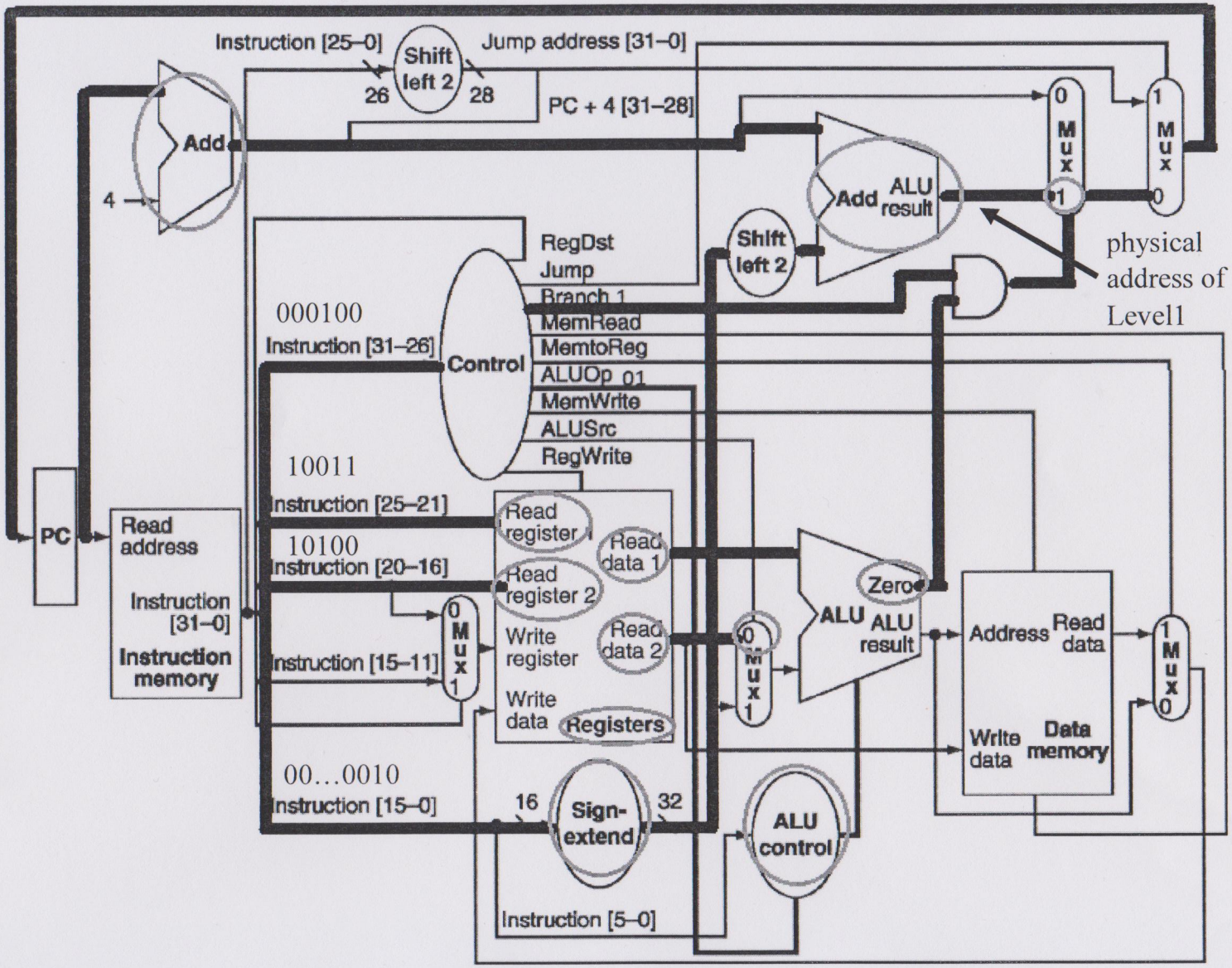


Q3)

beq $s3, $s4, Level1

+ Assume that value of
PC = 0x10000008
+ After increment by 4
PC = 0x1000000C

+ physical address of Level1
= 0x00000008
+0x1000000C
= 0x10000014



Q4)

1. and **$t2**, $s1, $s2
2. sub $s4, **$t2**, $s3
3. or $s7, $s6, **$t2**
4. add $t7, $t2, $t5
5. beq $t2, $s8, exit
6. sw $s9, 100($t2)

Data hazards will occur at instruction (2) and (3) because their register "$t2" depend on the preceding "and" instruction.

Control hazards will occur at instruction (6).

instruction (2)	instruction (3)
EX hazard	MEM hazard
EX/MEM.RegWrite = true EX/MEM.RegisterRd ≠ 0 EX/MEM.RegisterRd = ID/EX.RegisterRs	MEM/WB.RegWrite = true MEM/WB.RegisterRd ≠ 0 MEM/WB.RegisterRd = ID/EX.RegisterRt

Q5)

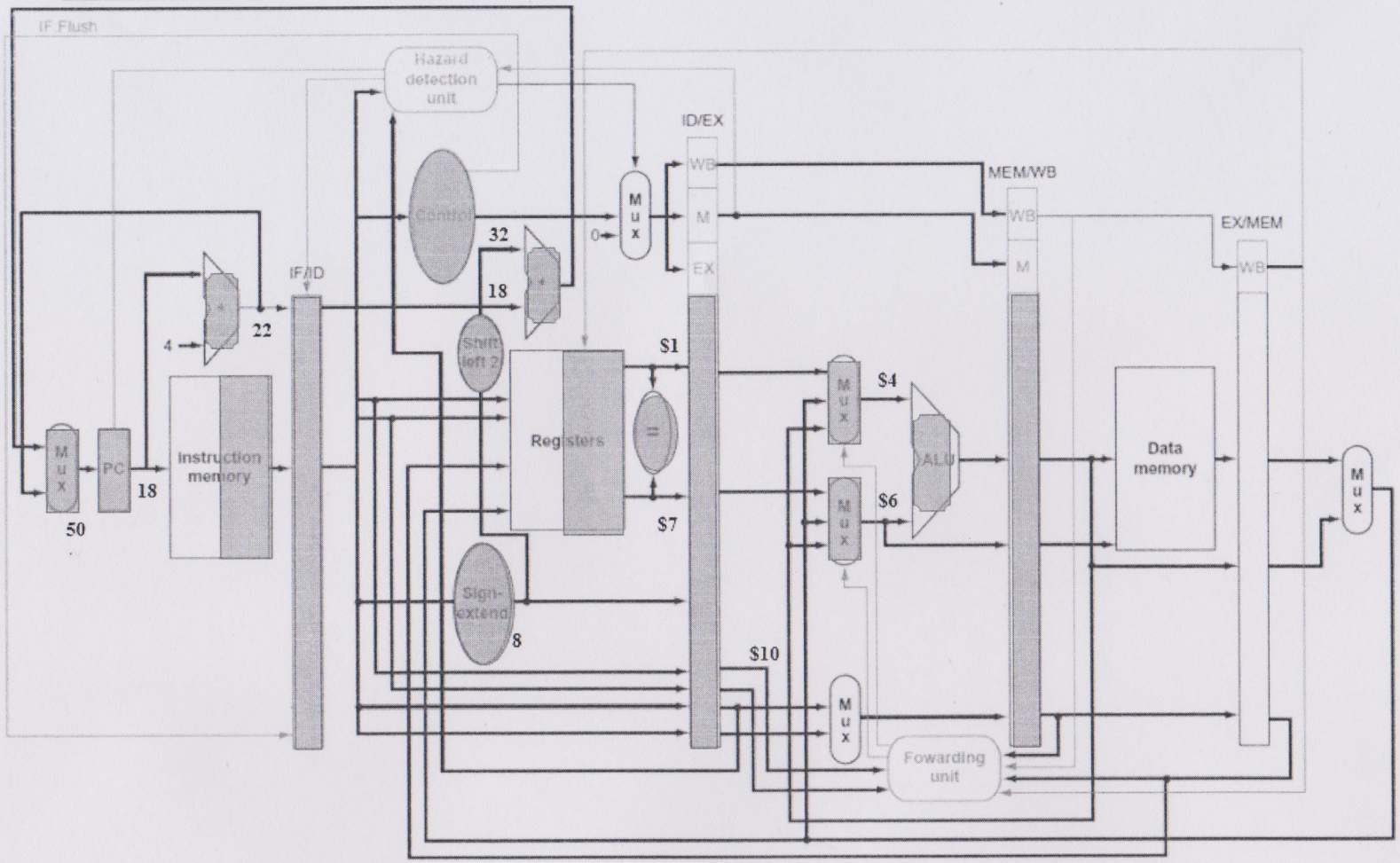
+ Moving the branch decision up requires two actions to occur earlier: computing the branch target address and evaluating the branch decision. The easy part of this change is to move up the branch address calculation. We already have the PC value and the immediate feld in the IF/ID pipeline register, so we just move the branch adder from the EX stage to the ID stage; of course, the branch target address calculation will be performed for all instructions, but only used when needed.
+ The harder part is the branch decision itself. For branch equal, we would compare the two registers read during the ID stage to see if they are equal. Equality can be tested by frst exclusive ORing their respective bits and then ORing all the results. Moving the branch test to the ID stage implies additional forwarding and hazard detection hardware, since a branch dependent on a result still in the pipeline must still work properly with this optimization. For example, to implement branch on equal (and its inverse), we will need to forward results to the equality test logic that operates during ID

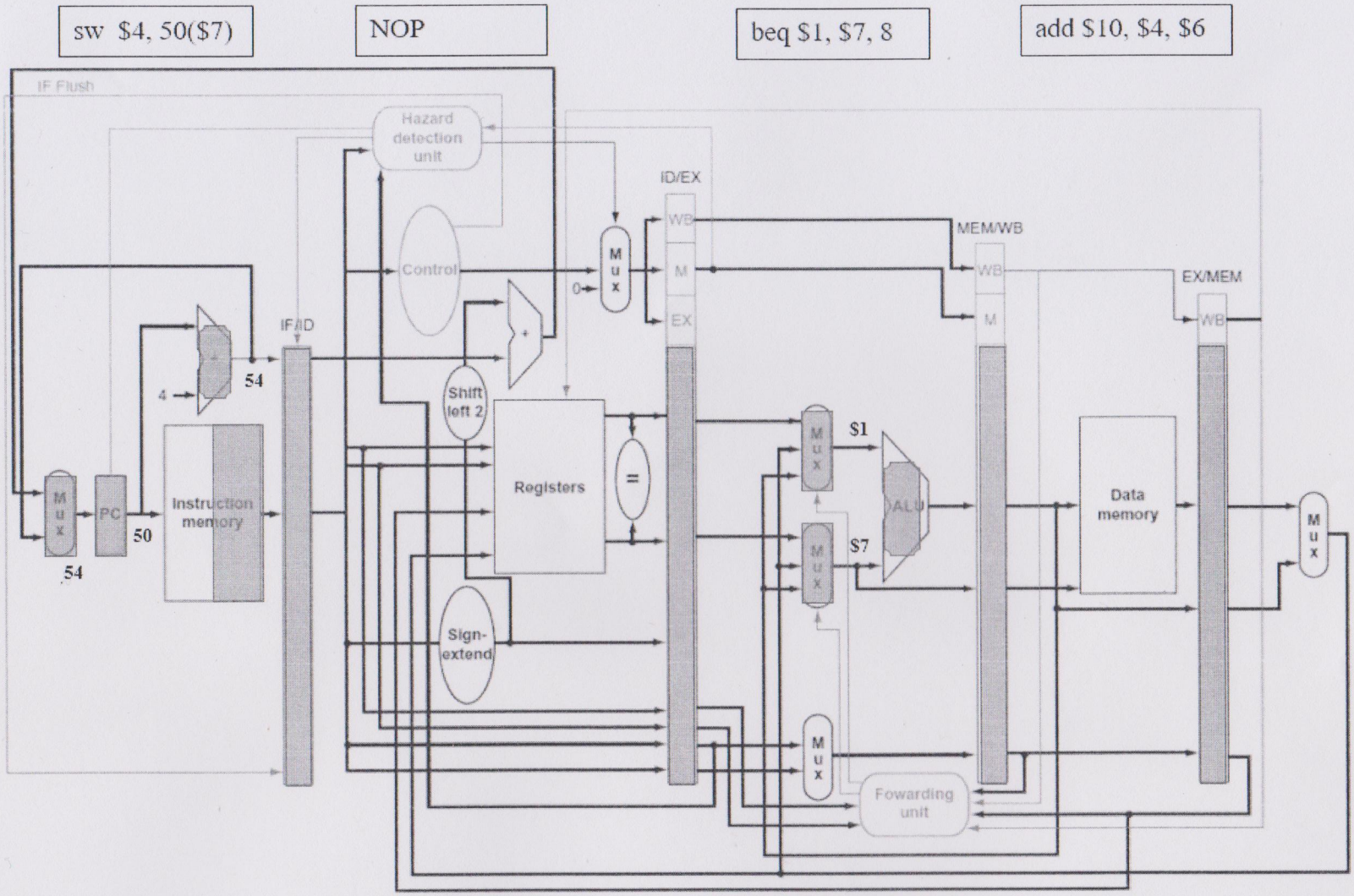
or $13, $2, $6

beq $1, $7, 8

add $10, $4, $6

Before "add"..




sw $4, 50($7)
NOP
beq $1, $7, 8
add $10, $4, $6
IF Flush
Hazard detection unit
Control
ID/EX
MEM/WB
EX/MEM
WB
M
EX
IF/ID
Shift left 2
Registers
Instruction memory
PC
Sign-extend
ALU
Data memory
Fowarding unit
Mux
$1
$7
4
54
50