- Addpc rt, imm16 - (R[rt]= PC+4+SignExt(imm16))

- Which instruction does CPU execute when the control signals are as given below?

(AluOp=00 lw-sw AluOp=01 Rtype AluOp=10 branch)

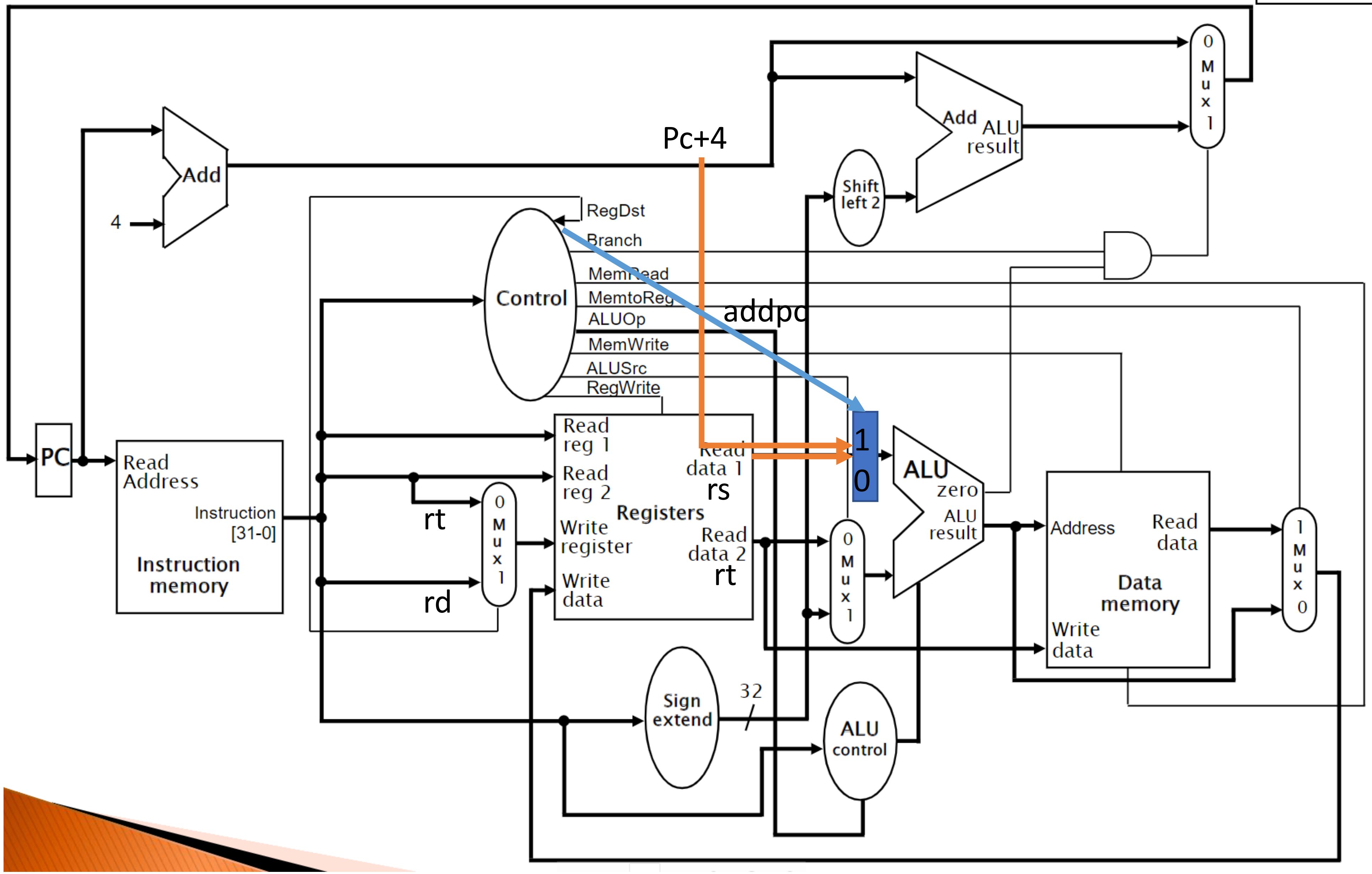
- RegDst=0 Branch=0 MemRead=0 MemtoReg=0 MemWrite=0 RegWrite=1 AluSrc=1 AluOp=11
- Answers for this question are in the next page.

- Another question: which instruction does CPU

execute when control signals are as given right?

Branch=0
MemRead=0
MemtoReg=0
MemWrite=0
RegDst=0
RegWrite=1
AluSrc=1 I TYPE
AluOp=00

R[rt]= PC+4+SignExt(imm16)



Alusrc=1
Regdst=0
Branch=0
Memread=0
Memwrite=0
Memtoreg=0
Regwrite=1

- LW $2, 100($5)
- SW $2, 200($6)

Show necessary modifications to the datapath (when lw is in WB stage & sw is in MEM stage)

Write equation for any control signal you use.

```
If (MEM/WEB.MemRead) && (EX/MEM.MemWrite)
        && (MEM/WEB.RegRd  == EX/MEM.RegRd)
        X = 1
Else
        X=0
```

1. Lw **$t2**, 0($t5)
2. NOP
3. NOP
4. Sll **$t3**, **$t2**, 1
5. Andi $t2, $t0, 34
6. NOP
7. Or $t0, **$t3**,$t4
8. Beq $t4, $s2, portakal
9. NOP
10. NOP
11. Add $s2, $s3, $s4

Portakal: addu $t5, $t2, $t3

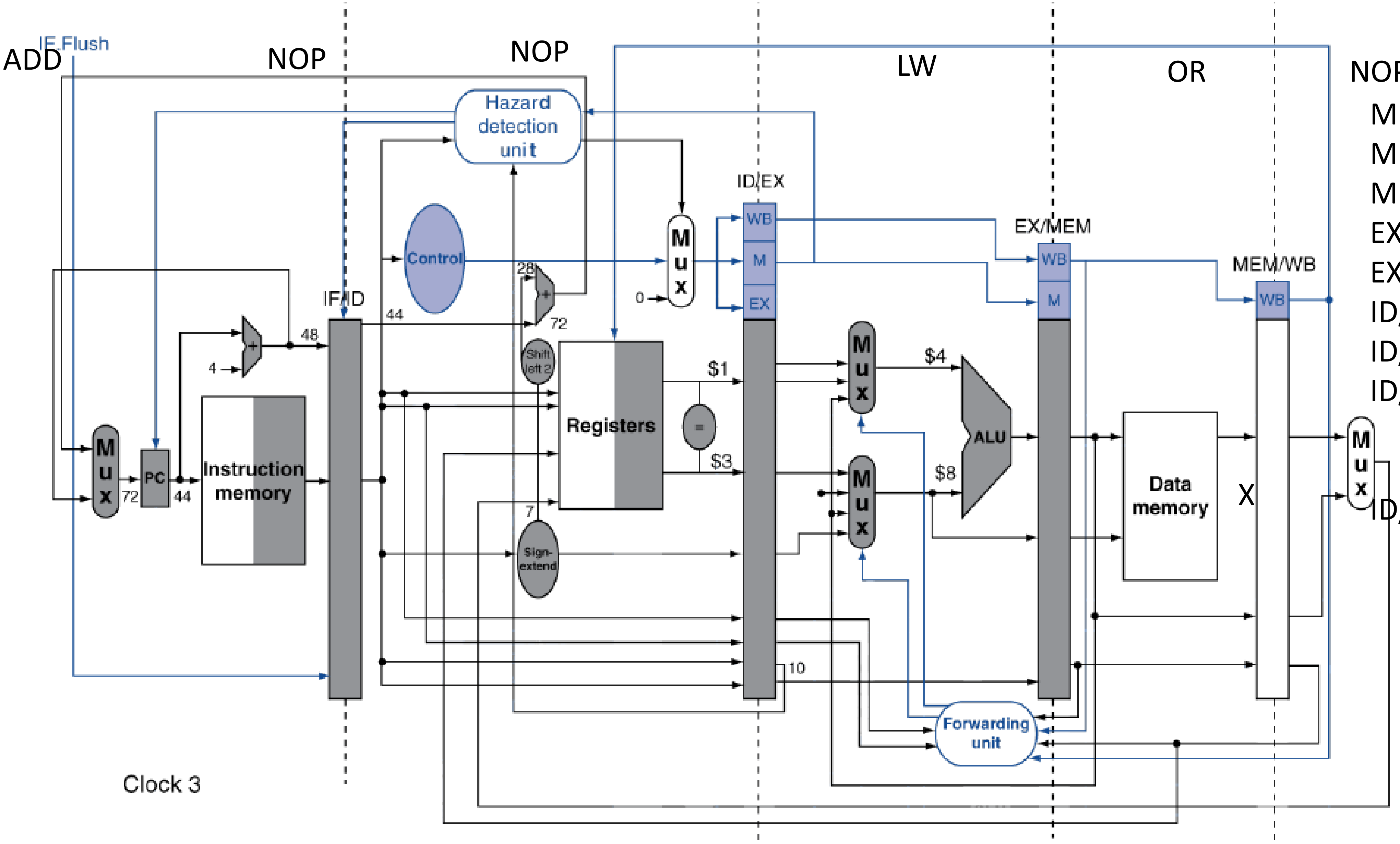
Hazard types? Any hazards must be resolved with nops

Line 1-2 : data hazard due to $t2- need to add 1 nop
Solution: forwarding unit & hazard detection unit
Line 2-4: data hazard due to $t3- forwarding unit solved this
Solution: : forwarding unit
Line 5-6: control hazard –need to add 1 nop
Solution: hazard detection unit

Which hazard that can not be solved without stalling even we use forwarding unit?
Without forwarding unit = 5 nops, forwarding unit = 2 nops

Hazard detection unit?

After the inserting nops, what’re the value of the control signals in binary at the 10th cycle.



ADD NOP NOP LW OR NOP

MEM/WB. RegDst= 0

MEM/WB. MemtoReg = X

MEM/WB. RegWrite=0

EX/MEM. MemWrite=0

EX/MEM. MemRead=0

ID/EX. Branch = 1

ID/EX. AluSrc=0

ID/EX. AluOp=10

X

ID/EX.RegWrite

9th = ANDI
8th = SLL
7th = NOP
6th =NOP
5th = LW

- At ID stage there is an equality check block (=) Design that block using combinational logic

A0 xor B0 = R0

……………. = R1……..R30 = nand(R0, …, R31) = equalSig = 1

A31 xor B31 = R31

Xor	and
00 0	00 0
01 1	01 0
10 1	10 0
11 0	11 1