Single Cycle Architecture

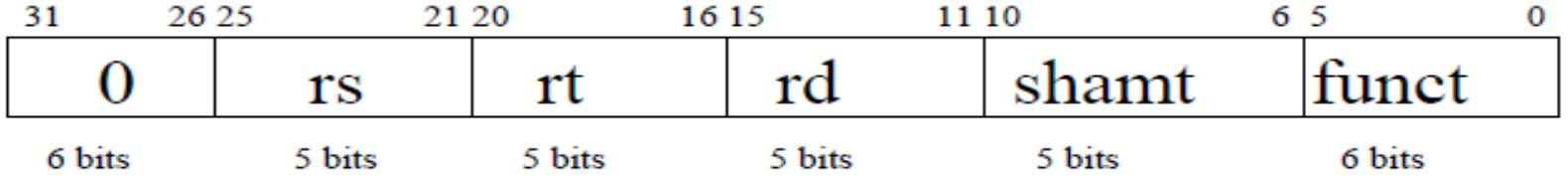
Part 4

A3 Due Friday 10pm
*upload pdf

- Office Hours Today Wed : 1-2
- Thursday : 12-1:30
- Sean (no office hours tomorrow)

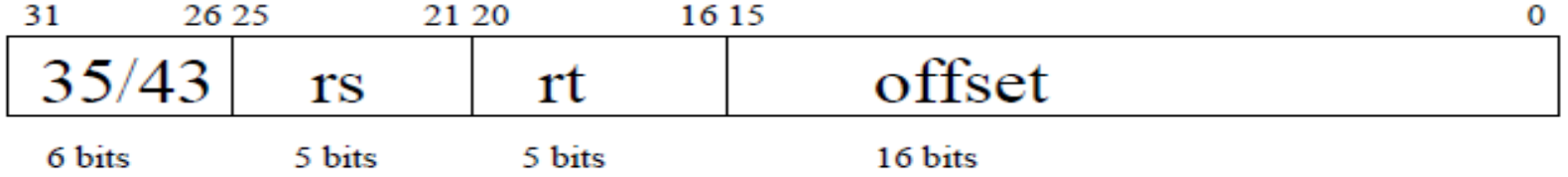
How the Instruction Bits are Broken up:

R-format: `add $t1, $t2, $t3` ⇒ `add rd,rs,rt`



sll, sub

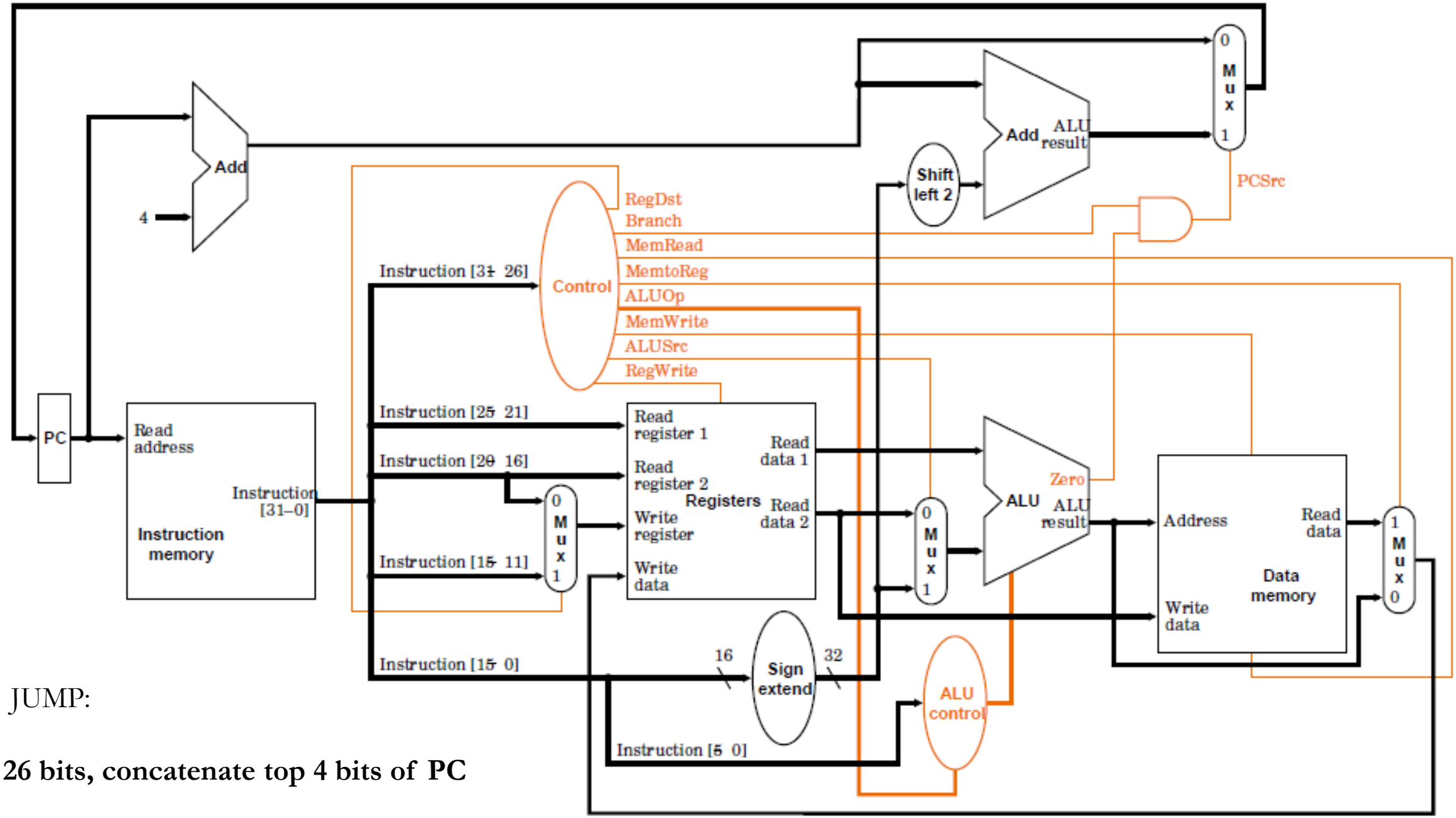
Load/store: `lw $t1, 100($t2)` ⇒ `lw rt,100(rs)`



beq, subi, addi

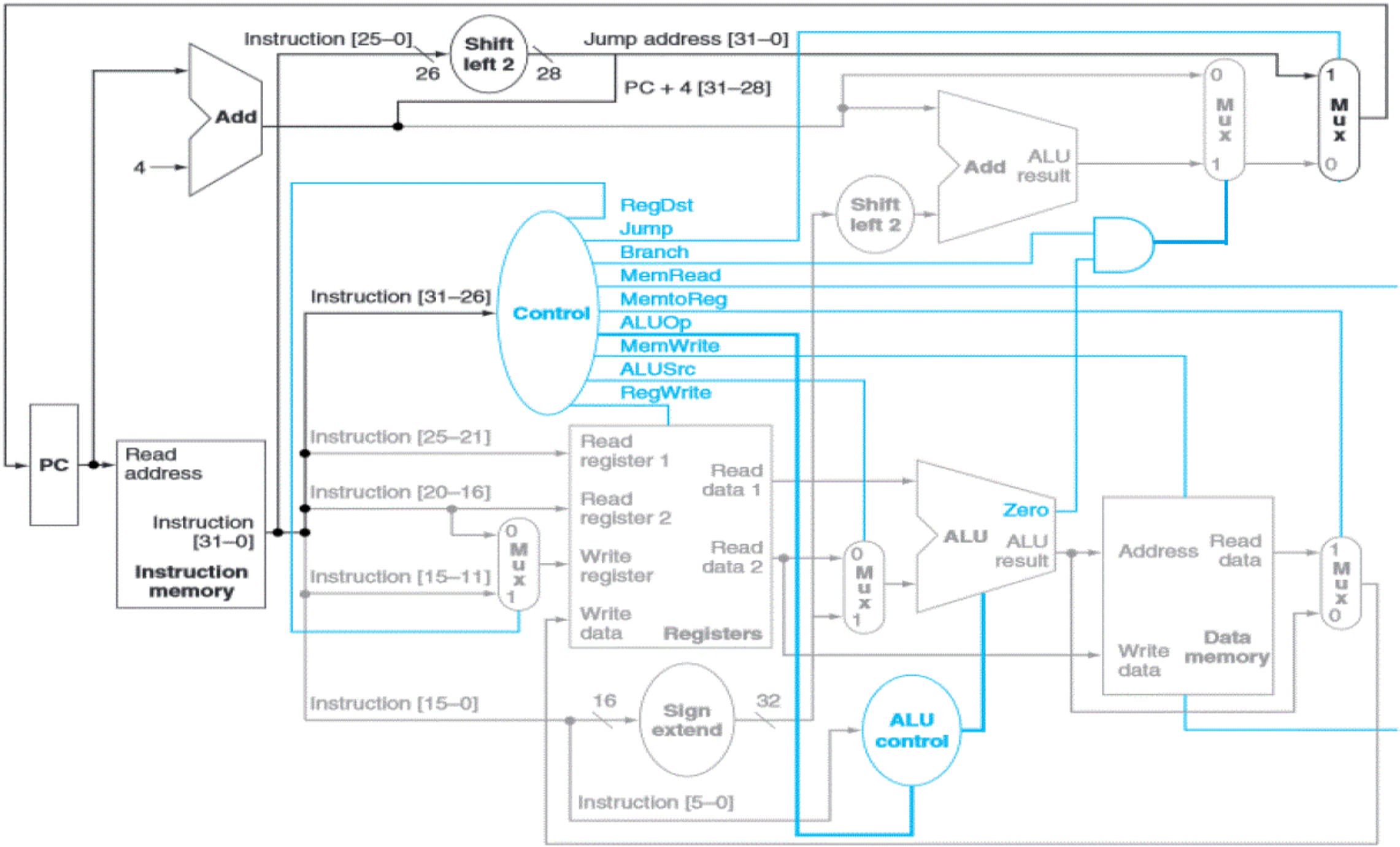
Jump: `j 3000`

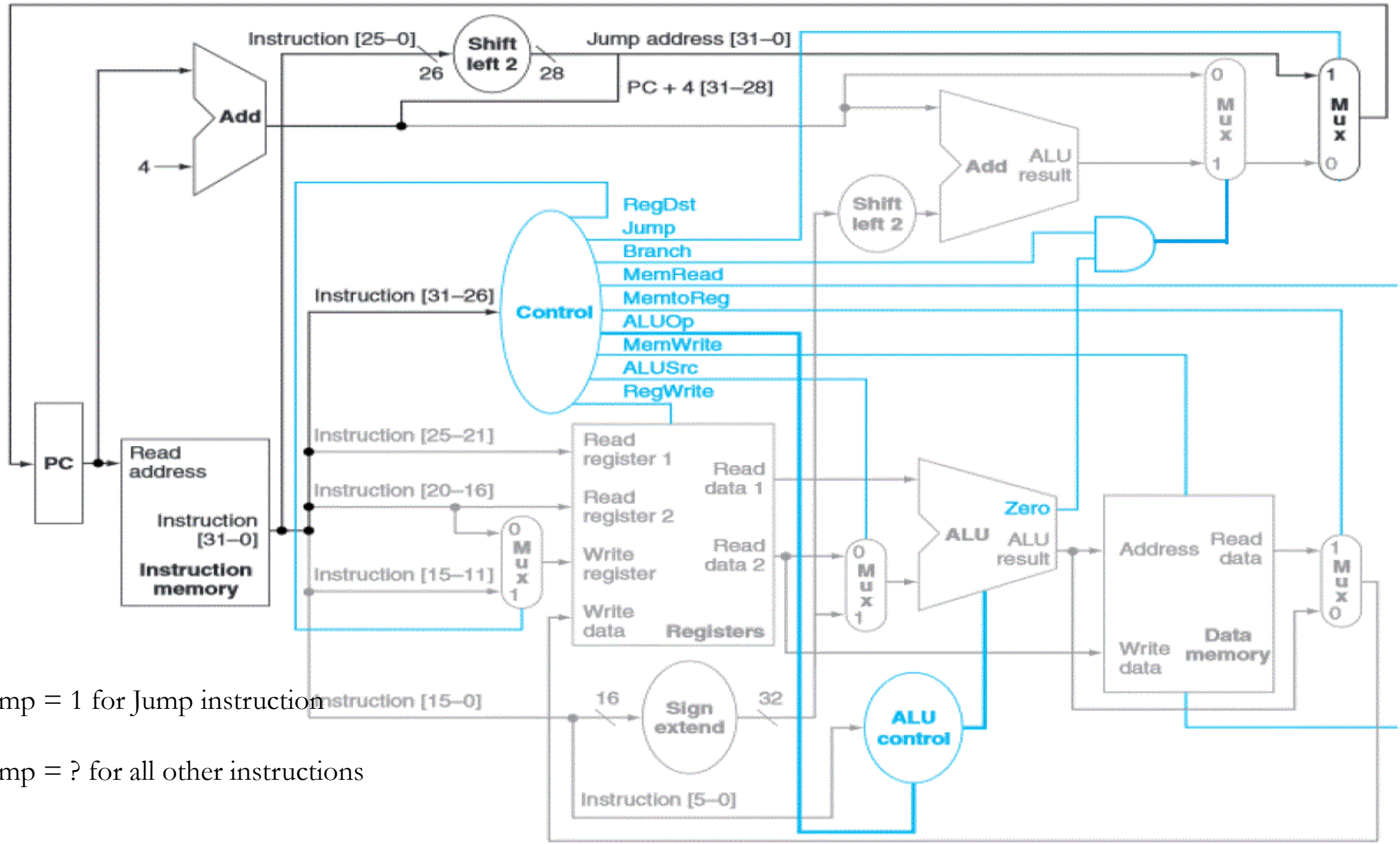
special



JUMP:

26 bits, concatenate top 4 bits of PC


Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC + 4 [31–28]
Add
4
M u x
0
1
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory



Jump = 1 for Jump instruction

Jump = ? for all other instructions

Add Jump control line

Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0	Jump
R-format	1	0	0	1	0	0	0	1	0	
`lw`	0	1	1	1	1	0	0	0	0	
`sw`	X	1	X	0	0	1	0	0	0	
`beq`	X	0	X	0	0	0	1	0	1	
Jump	**X**	**X**	**X**	**0**	**0**	**0**	**__**	**X**	**X**	**1**

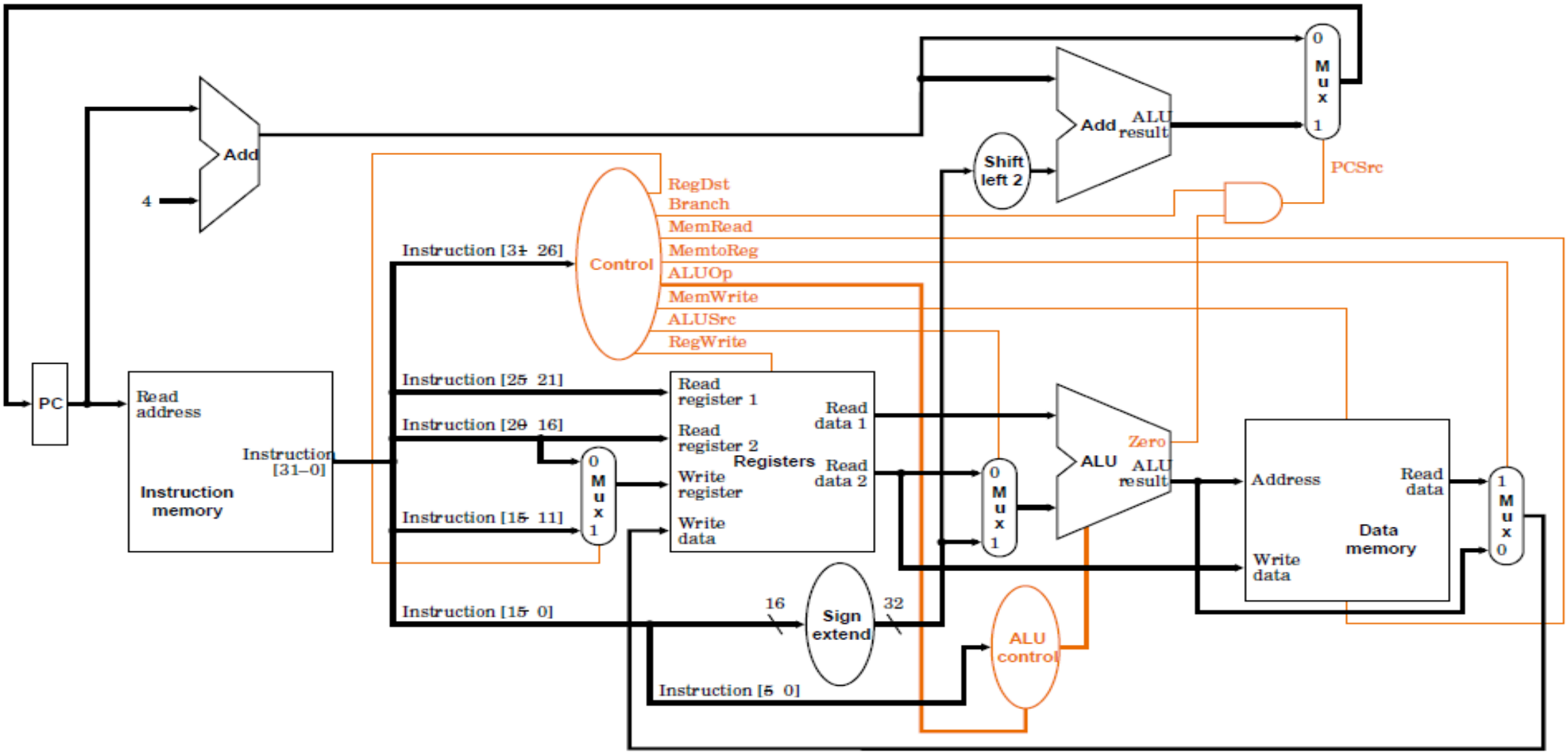
What is value of Branch Bit Control Line for the JUMP Instruction

A) 1 B) 0 C) X

Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0	Jump
R-format	1	0	0	1	0	0	0	1	0	
lw	0	1	1	1	1	0	0	0	0	
sw	X	1	X	0	0	1	0	0	0	
beq	X	0	X	0	0	0	1	0	1	
Jump	**X**	**X**	**X**	**0**	**0**	**0**	**__**	**X**	**X**	**1**

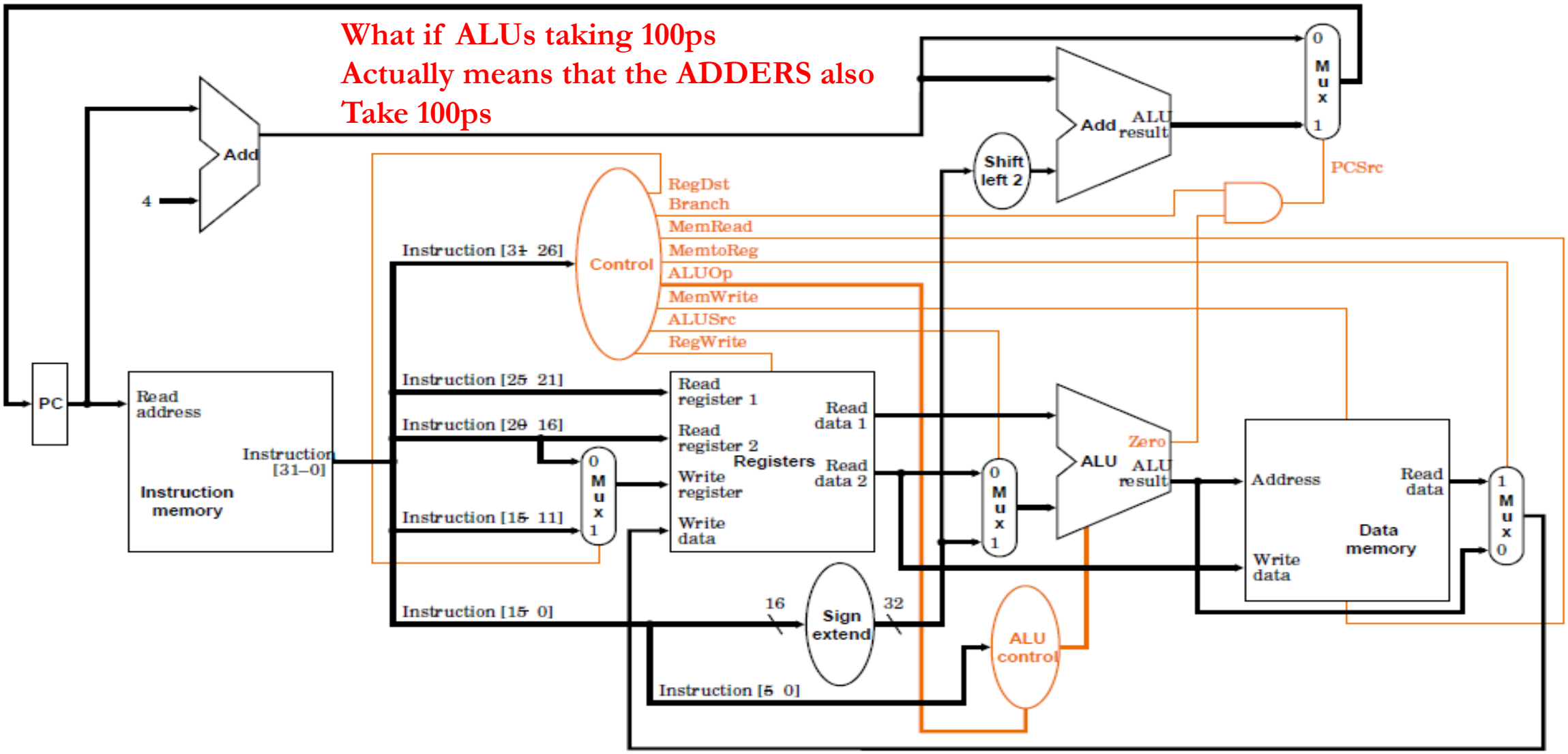
Add Jump control line

Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0	Jump
R-format	1	0	0	1	0	0	0	1	0	**0**
`lw`	0	1	1	1	1	0	0	0	0	**0**
`sw`	X	1	X	0	0	1	0	0	0	**0**
`beq`	X	0	X	0	0	0	1	0	1	**0**
Jump	**X**	**X**	**X**	**0**	**0**	**0**	**X**	**X**	**X**	**1**



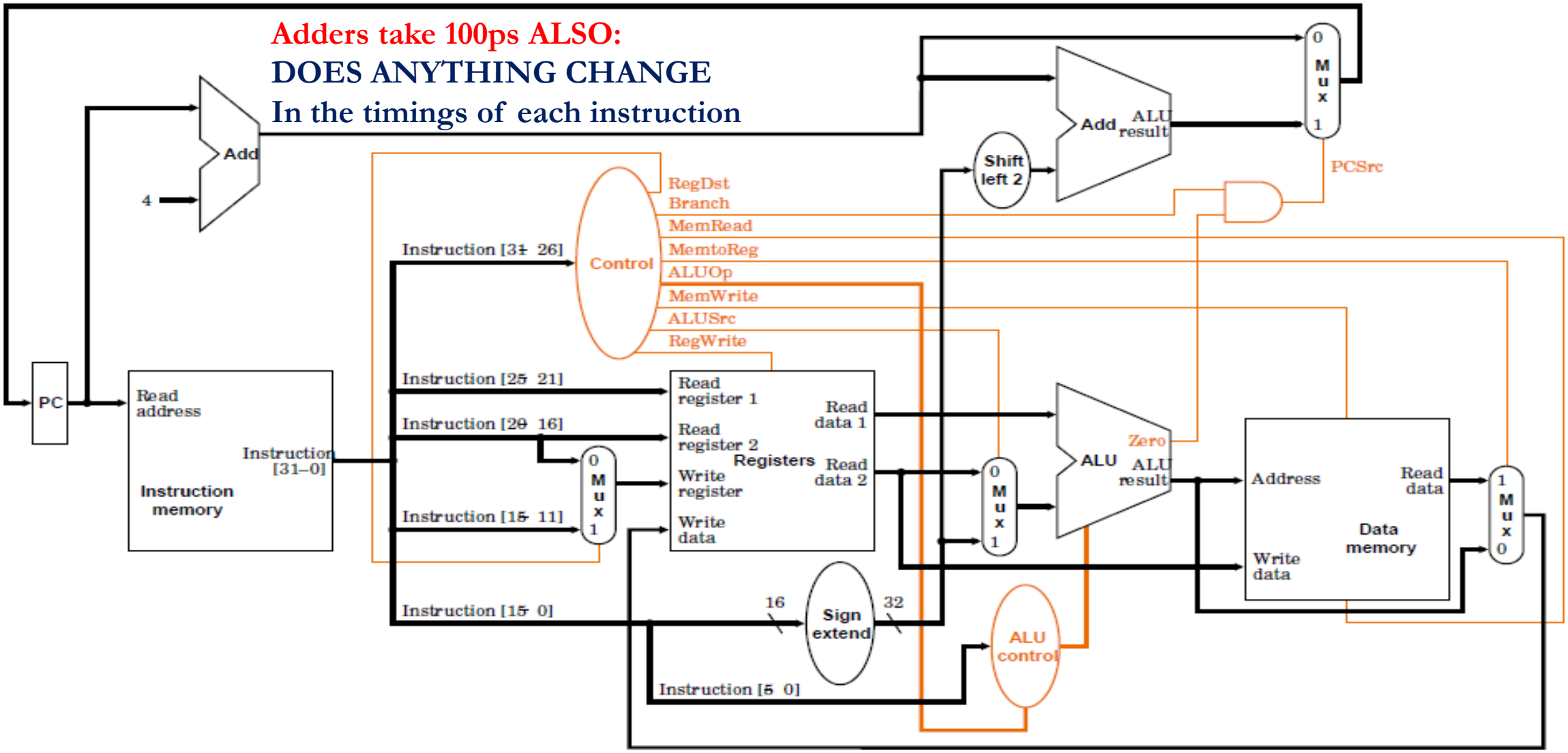
- Suppose memory units take 200 ps (picoseconds), ALUs 100 ps, register files 50 ps, no delay on other units
- Jumps take 200 ps, branches take 350 ps, R-format instructions 400 ps, stores 550 ps, loads 600 ps.

What if ALUs taking 100ps
Actually means that the ADDERS also
Take 100ps



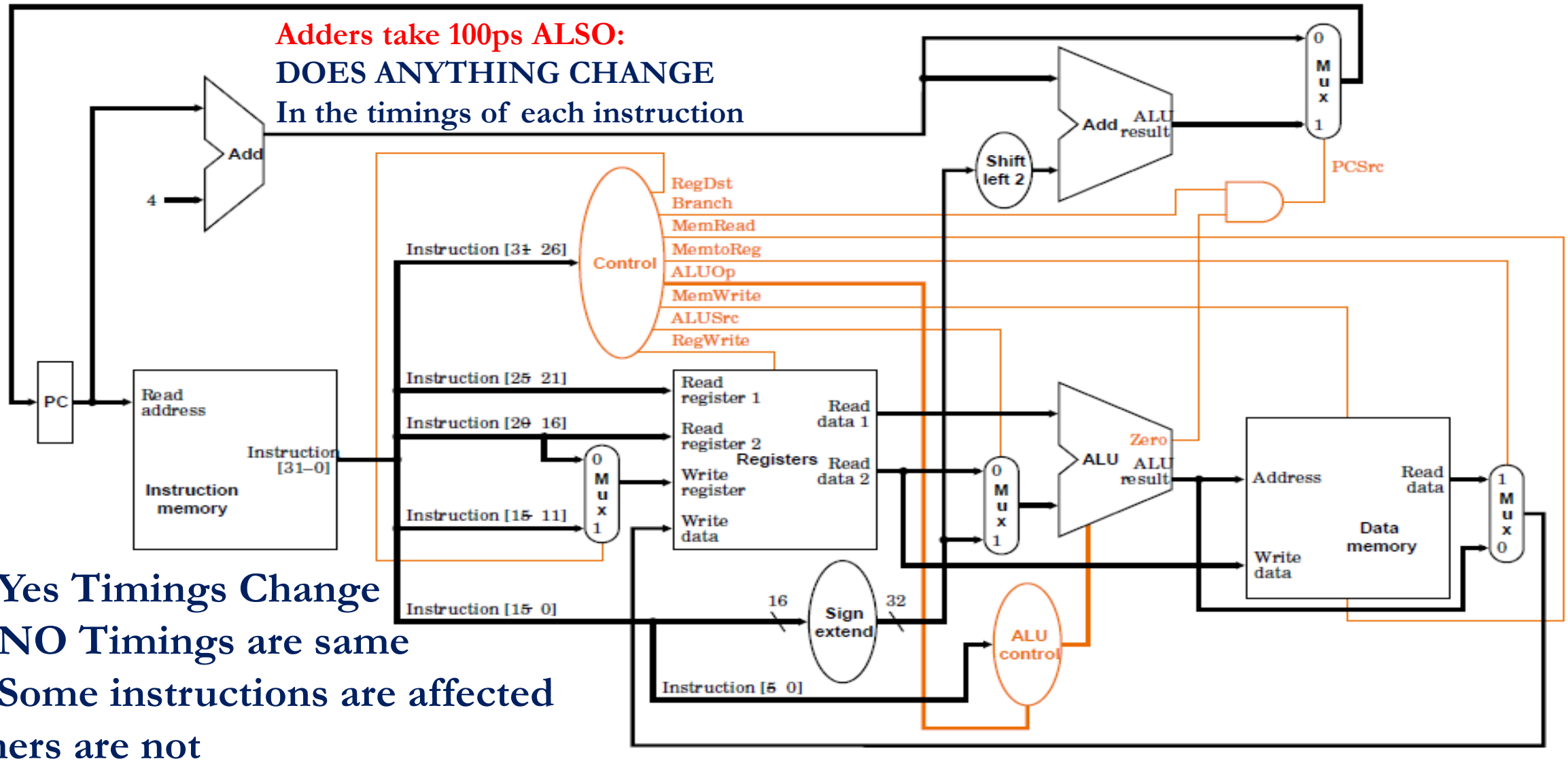
- Suppose memory units take 200 ps (picoseconds), ALUs 100 ps, register files 50 ps, no delay on other units
- Jumps take 200 ps, branches take 350 ps, R-format instructions 400 ps, stores 550 ps, loads 600 ps.

Adders take 100ps ALSO:
DOES ANYTHING CHANGE
In the timings of each instruction



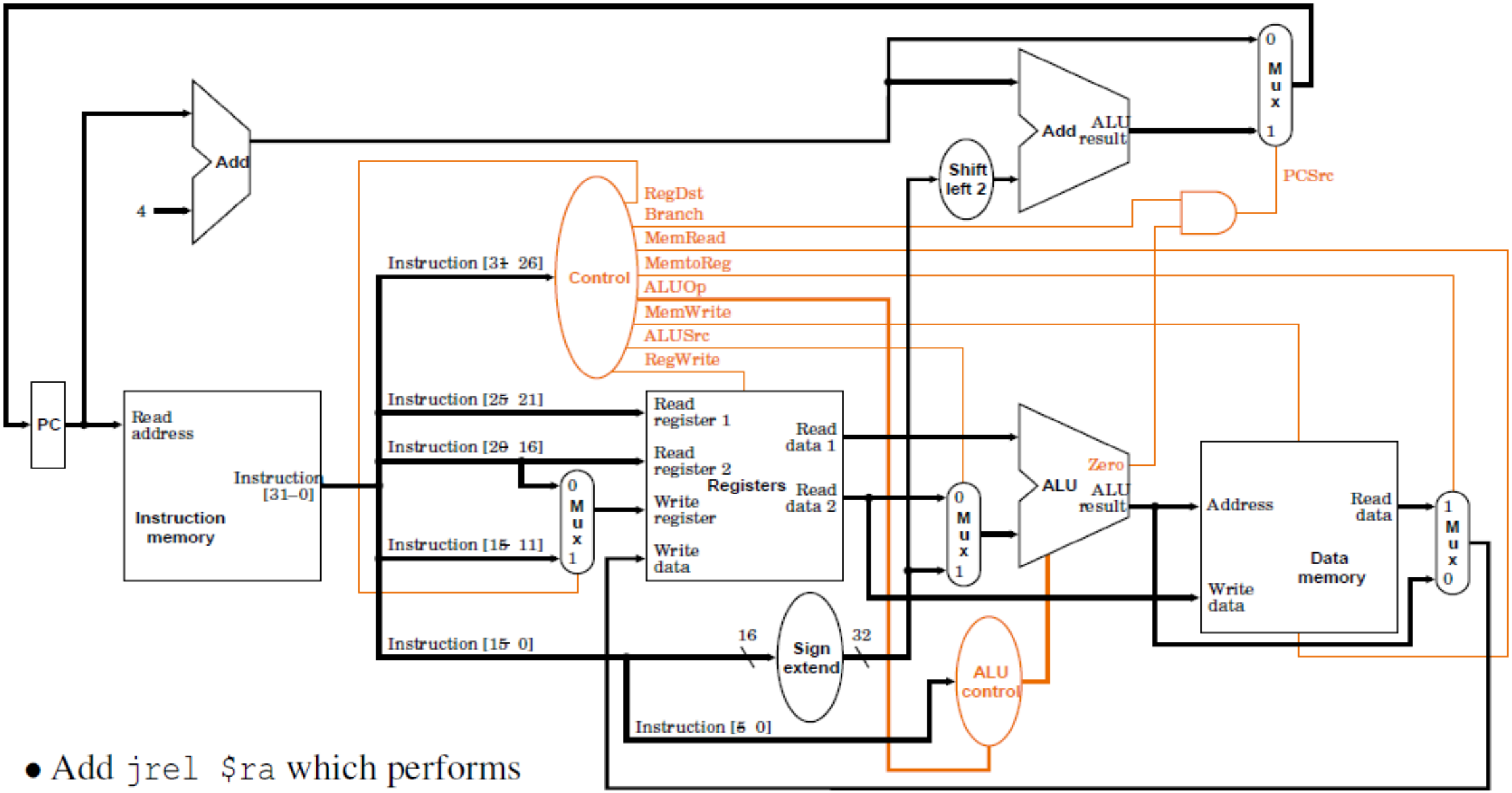
- Suppose memory units take 200 ps (picoseconds), ALUs 100 ps, register files 50 ps, no delay on other units
- Jumps take 200 ps, branches take 350 ps, R-format instructions 400 ps, stores 550 ps, loads 600 ps.

Adders take 100ps ALSO:
DOES ANYTHING CHANGE
In the timings of each instruction

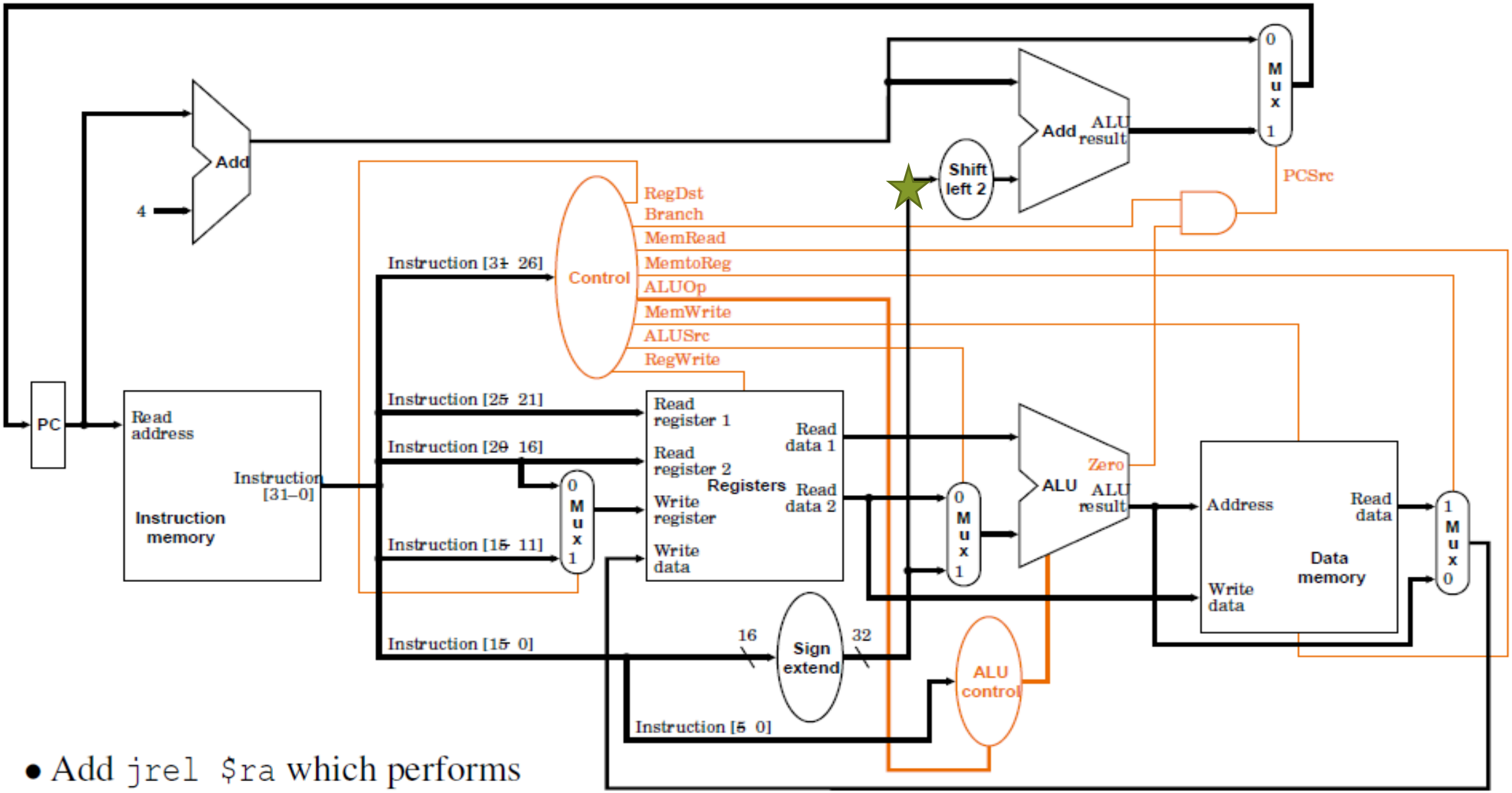


A) Yes Timings Change
B) NO Timings are same
C) Some instructions are affected others are not

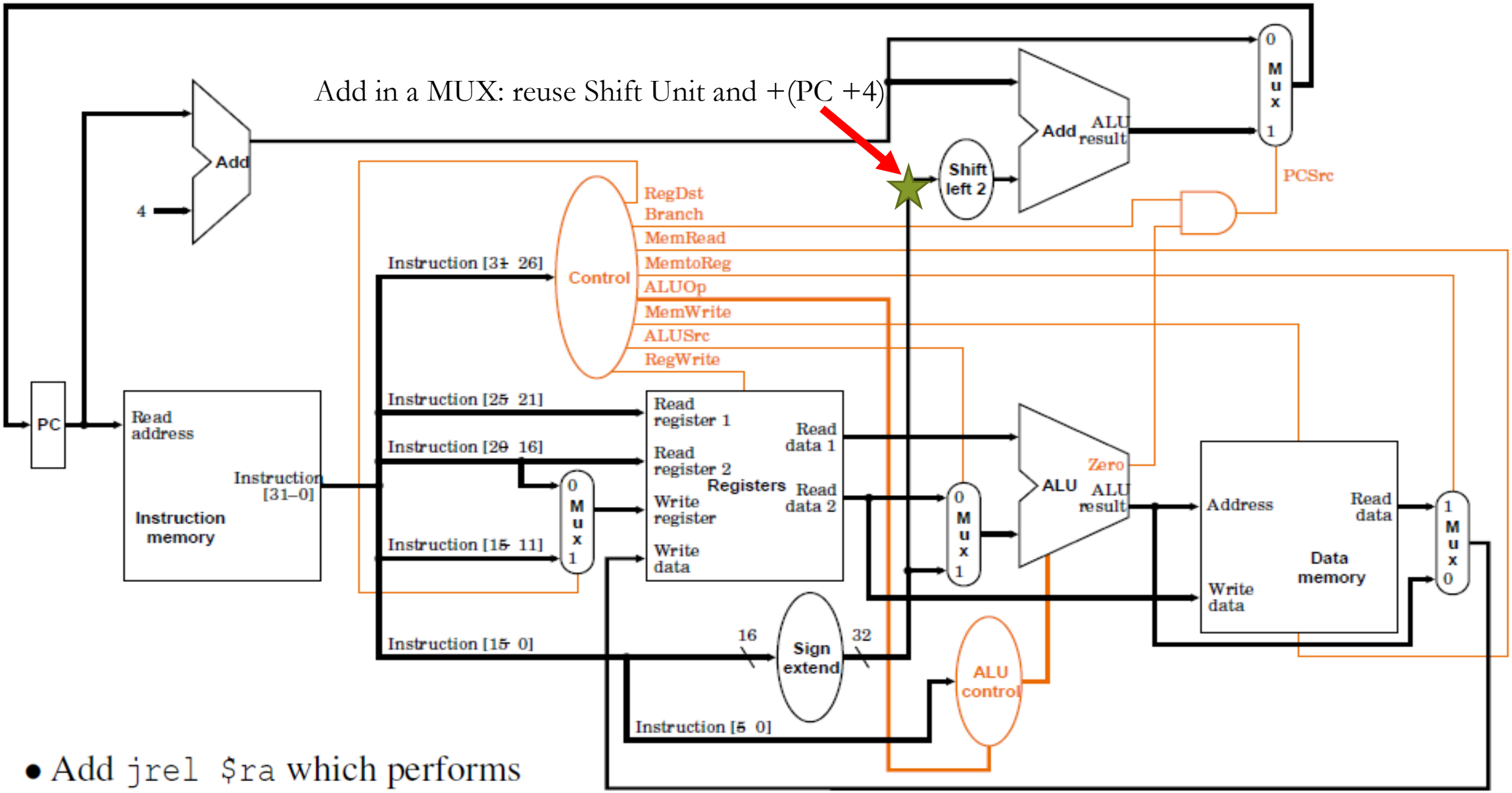
- Suppose memory units take 200 ps (picoseconds), ALUs 100 ps, register files 50 ps, no delay on other units
- Jumps take 200 ps, branches take 350 ps, R-format instructions 400 ps, stores 550 ps, loads 600 ps.



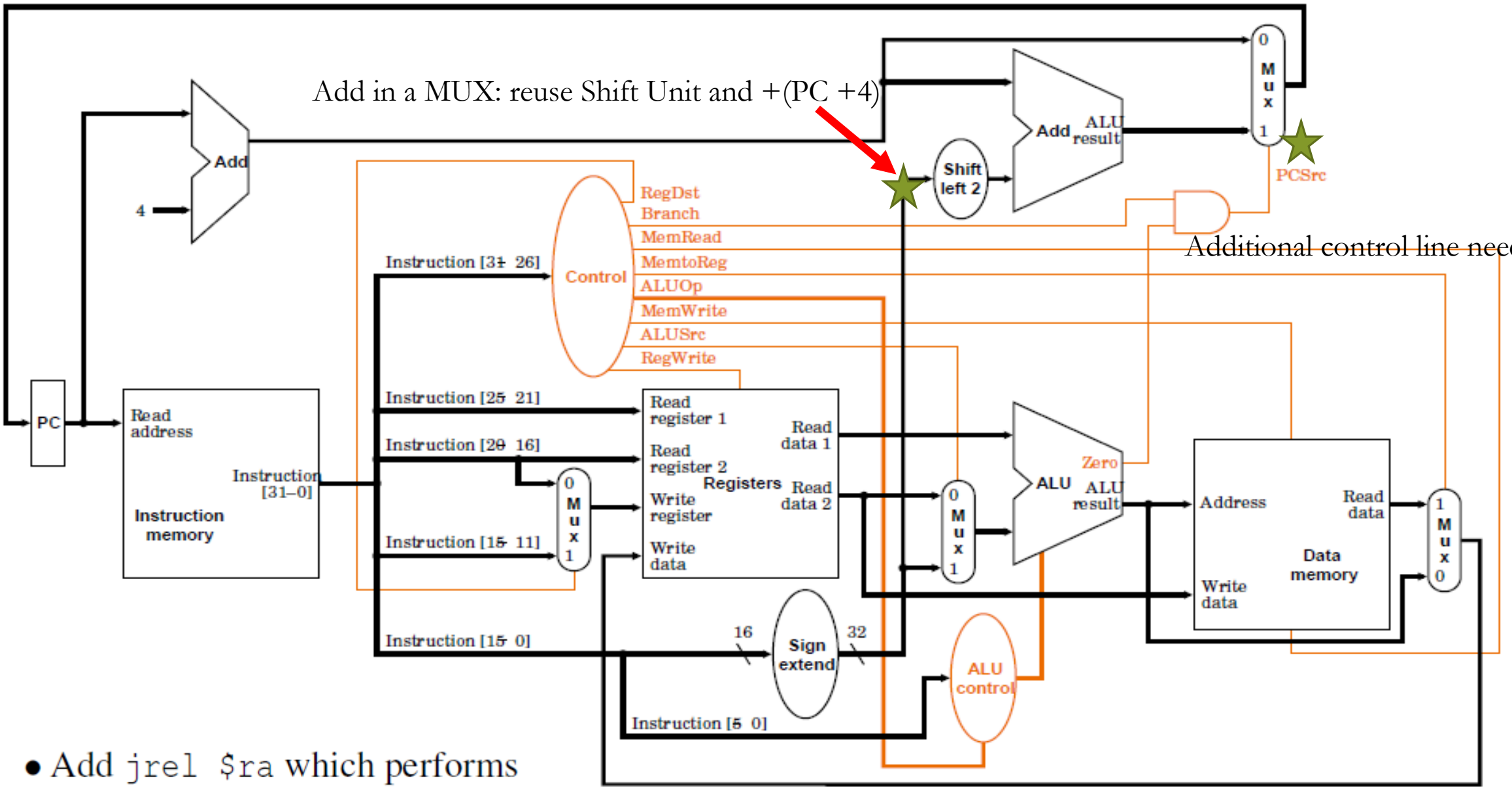
- Add `jrel $ra` which performs
  PC ← PC + 4 + 4*$ra



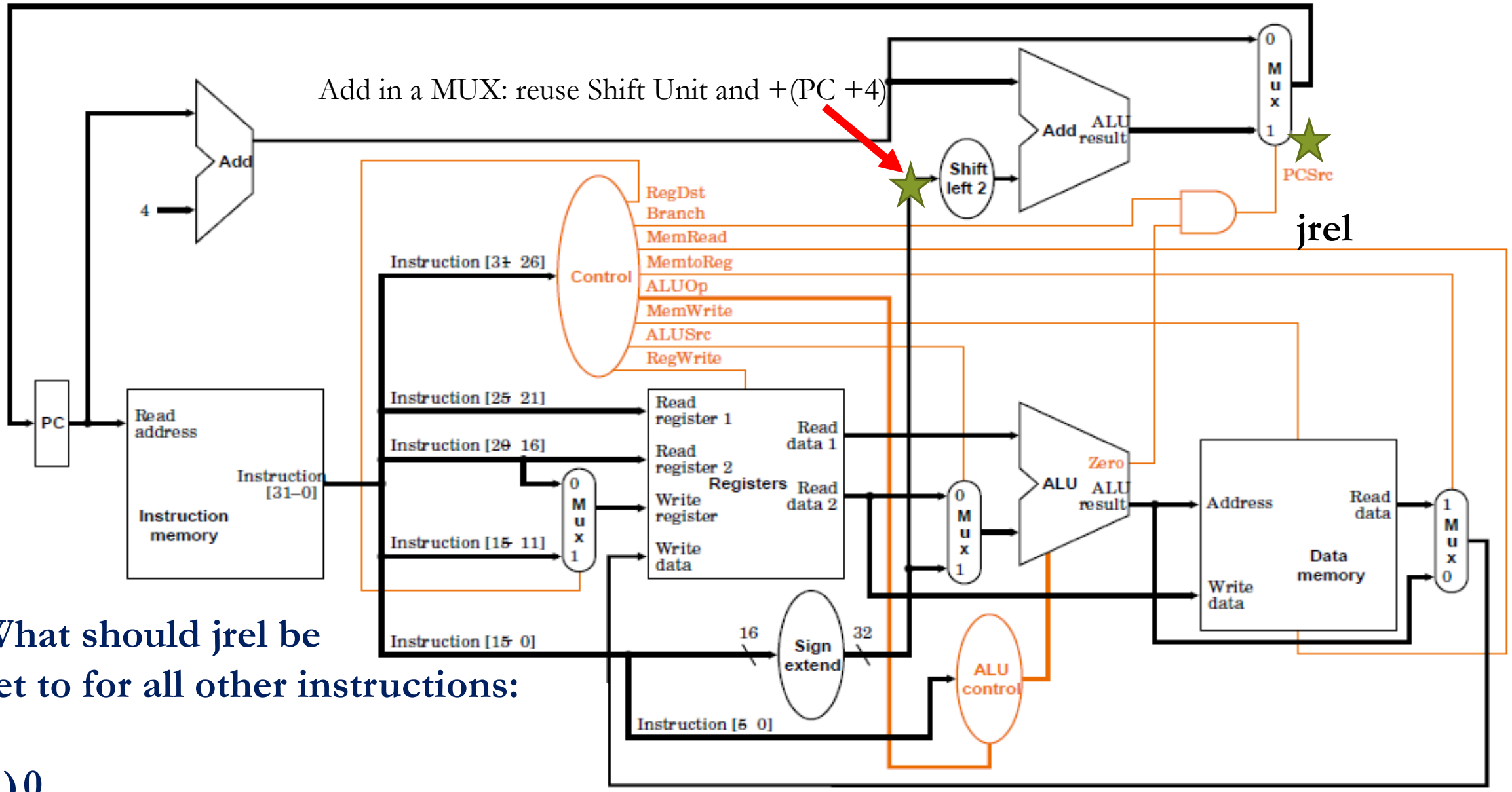
- Add `jrel $ra` which performs
  PC ← PC + 4 + 4*$ra



- Add `jrel $ra` which performs
  PC ← PC + 4 + 4*$ra



- Add `jrel $ra` which performs
  PC ← PC + 4 + 4*$ra

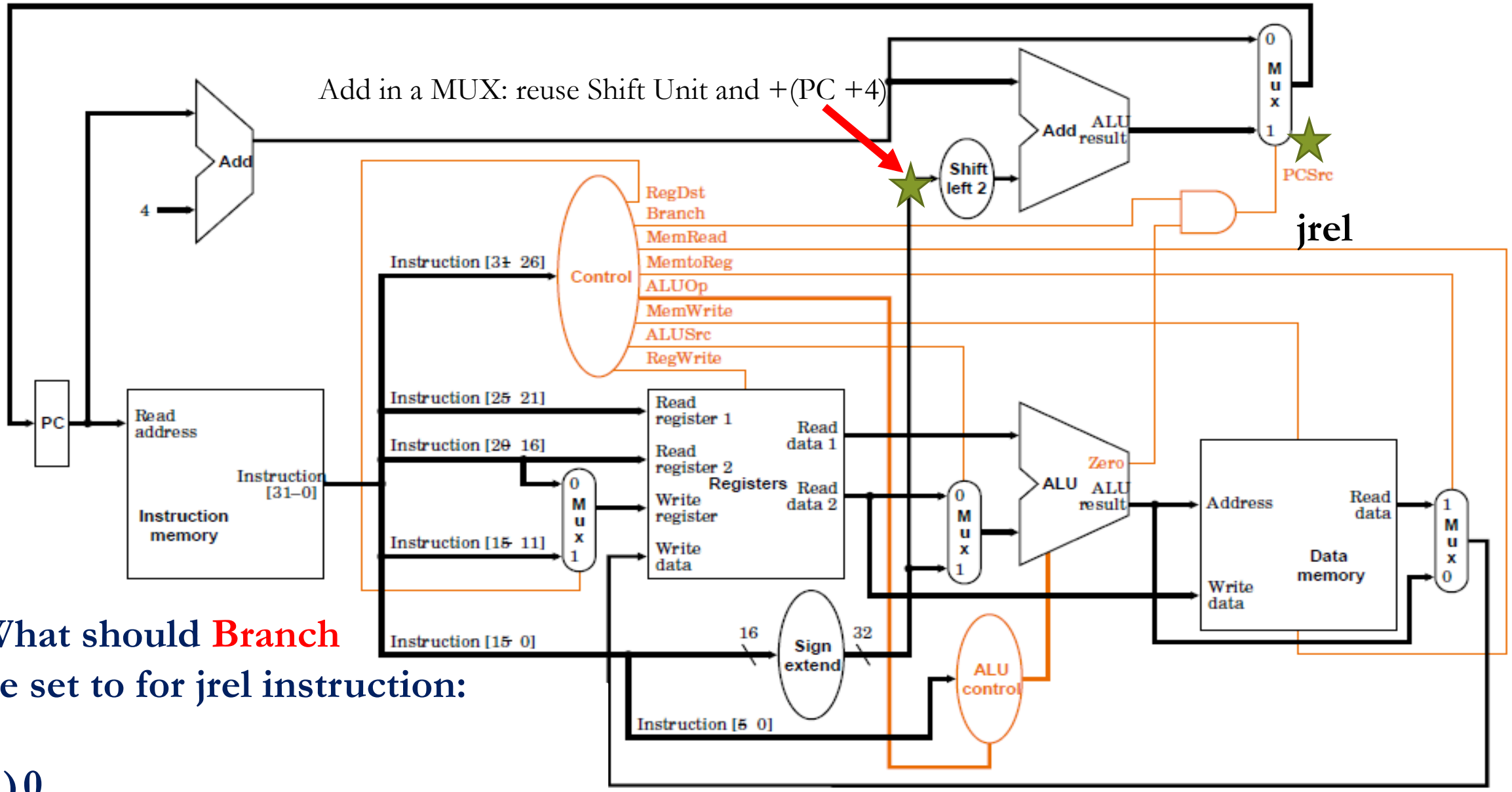


What should jrel be set to for all other instructions:

A) 0

B) 1

C) 1 or X for some instruction



What should Branch be set to for jrel instruction:

A) 0
B) 1
C) X

Add Jrel control line

Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0	Jrel
R-format	1	0	0	1	0	0	0	1	0	0
`lw`	0	1	1	1	1	0	0	0	0	0
`sw`	X	1	X	0	0	1	0	0	0	0
`beq`	X	0	X	0	0	0	1	0	1	0
Jrel	**X**	**X**	**X**	**0**	**0**	**0**	**X**	**X**	**X**	**1**

Performance of Single Cycle Machines

- Suppose memory units take 200 ps (picoseconds), ALUs 100 ps, register files 50 ps, no delay on other units
- Jumps take 200 ps, branches take 350 ps, R-format instructions 400 ps, stores 550 ps, loads 600 ps.
- Clock period must be increased to 600 ps or more
- Even worse when floating-point instructions are implemented
- Idea: use multicycle implementation and R format

Modifying the datapath

- Normally design complete datapath for all instructions together.
- Various ways to modify datapath. The following is one approach for adding a new assembly instruction:

1. Determine what datapath is needed for new command
2. Check if any components in current datapath can be used
3. Wire in components of new datapath into existing datapath
   Probably requires MUXes
4. Add new control signals to Control units
5. Adjust old control signals to account for new command