CSM151B
Computer Systems Architecture

Week 4 Discussion

2/2/2018

Logistics

- HW3 due today

Agenda

- Single-cycle microarchitecture

Instruction Processing Viewed Another Way

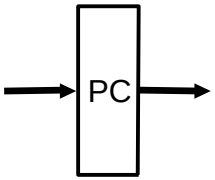
- An instruction processing engine consists of two components
  - **Datapath**: Consists of hardware elements that deal with and transform data signals
    - functional units that operate on data
    - hardware structures (e.g. wires and muxes) that enable the flow of data into the functional units and registers
    - storage units that store data (e.g., registers)
  - **Control logic**: Consists of hardware elements that determine control signals, i.e., signals that specify what the datapath elements should do to the data

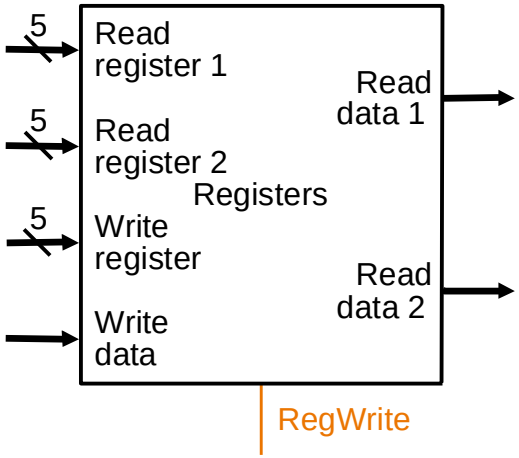
Many Ways of Datapath and Control Design

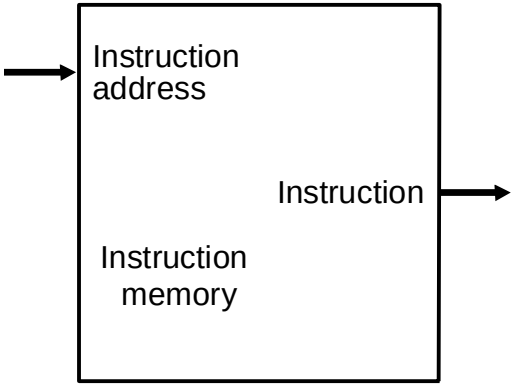
- There are many ways of designing the datapath and control logic
- Single-cycle, multi-cycle, pipelined datapath and control
- Hardwired/combinational vs. microcoded/microprogrammed control
  - Control signals generated by combinational logic versus
  - Control signals stored in a memory structure
- Control signals and structure depend on the datapath design

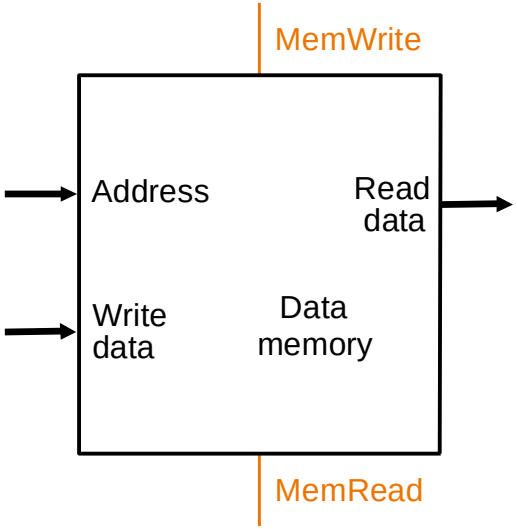
The State Elements

- Data and control inputs

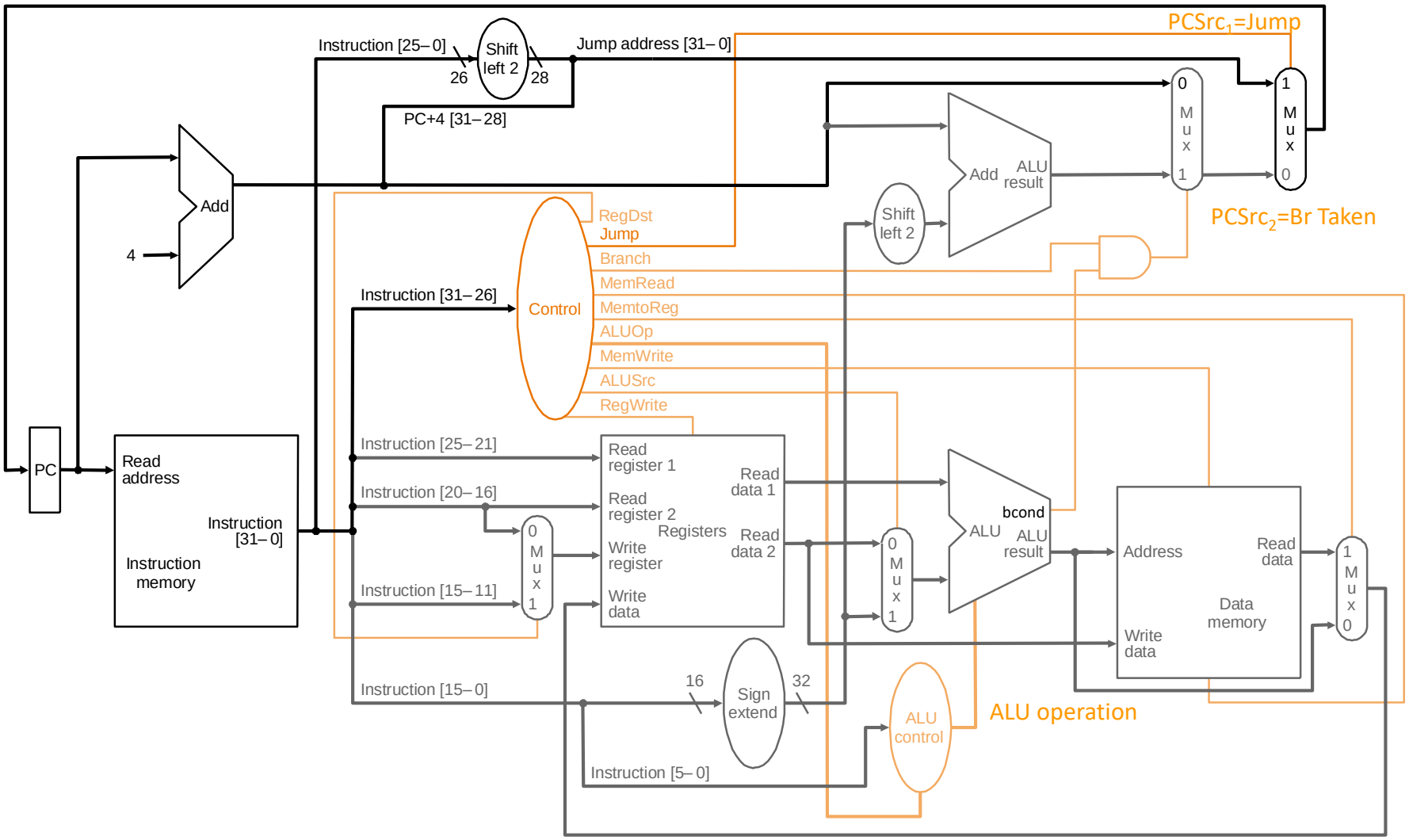




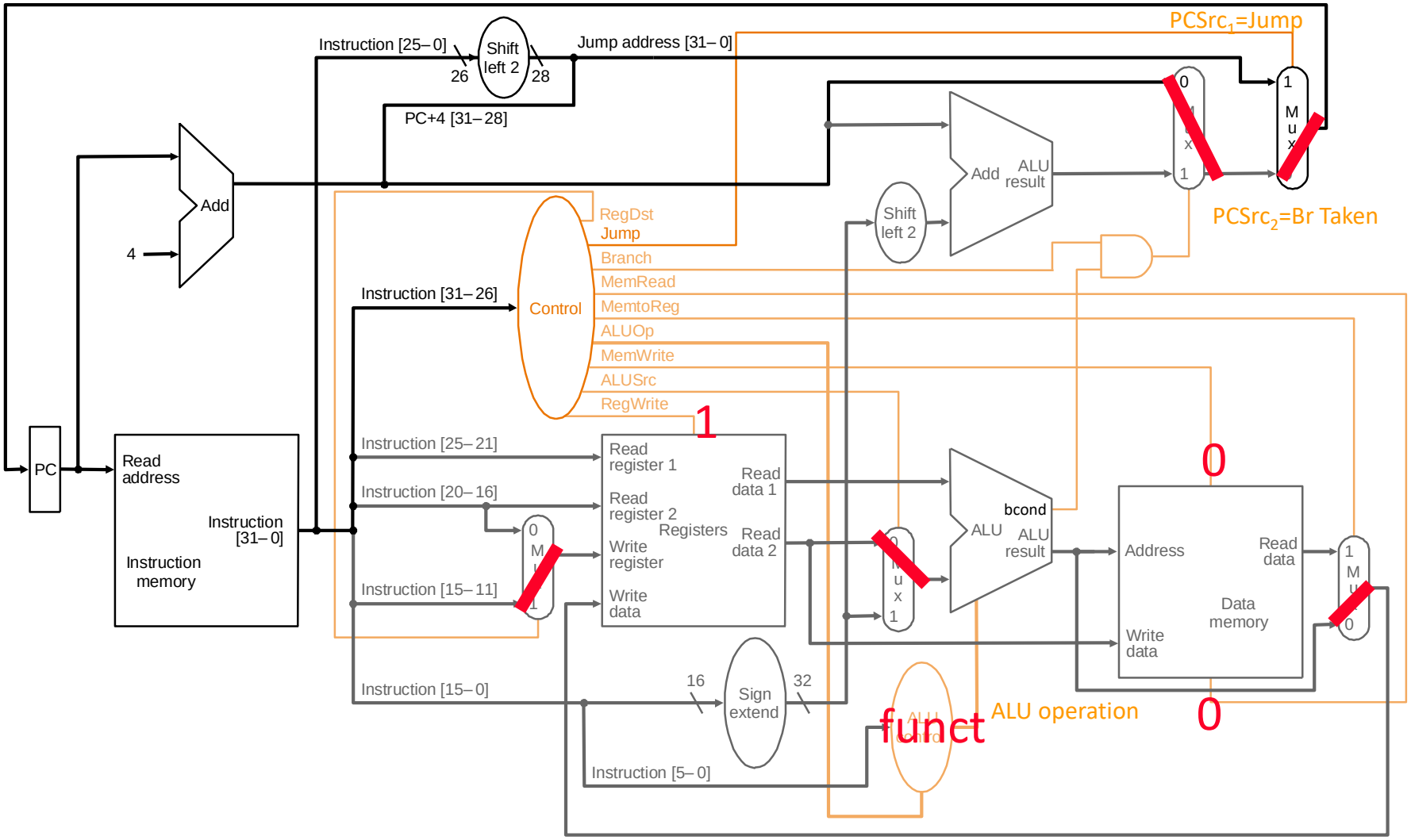




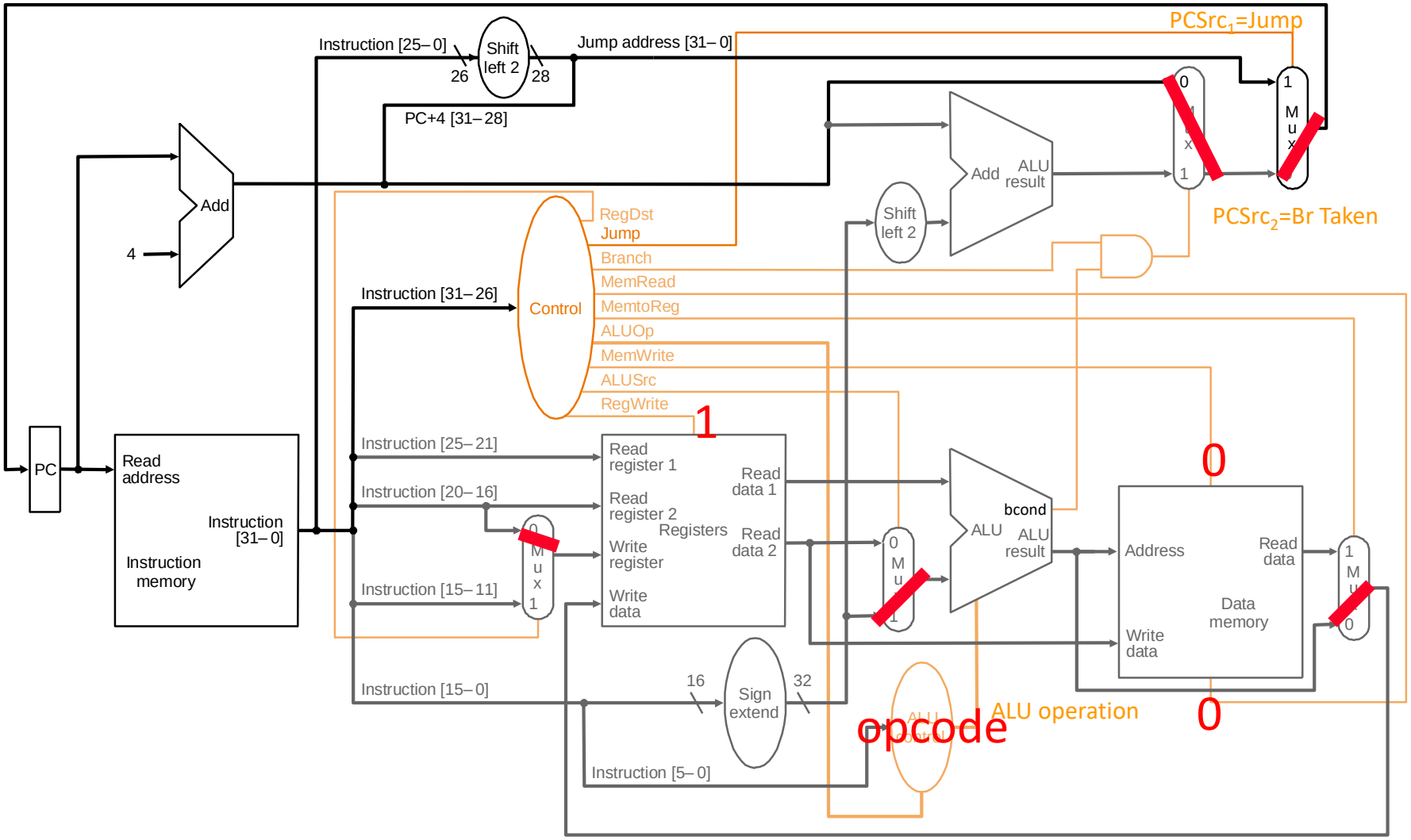
The Full MIPS Datapath



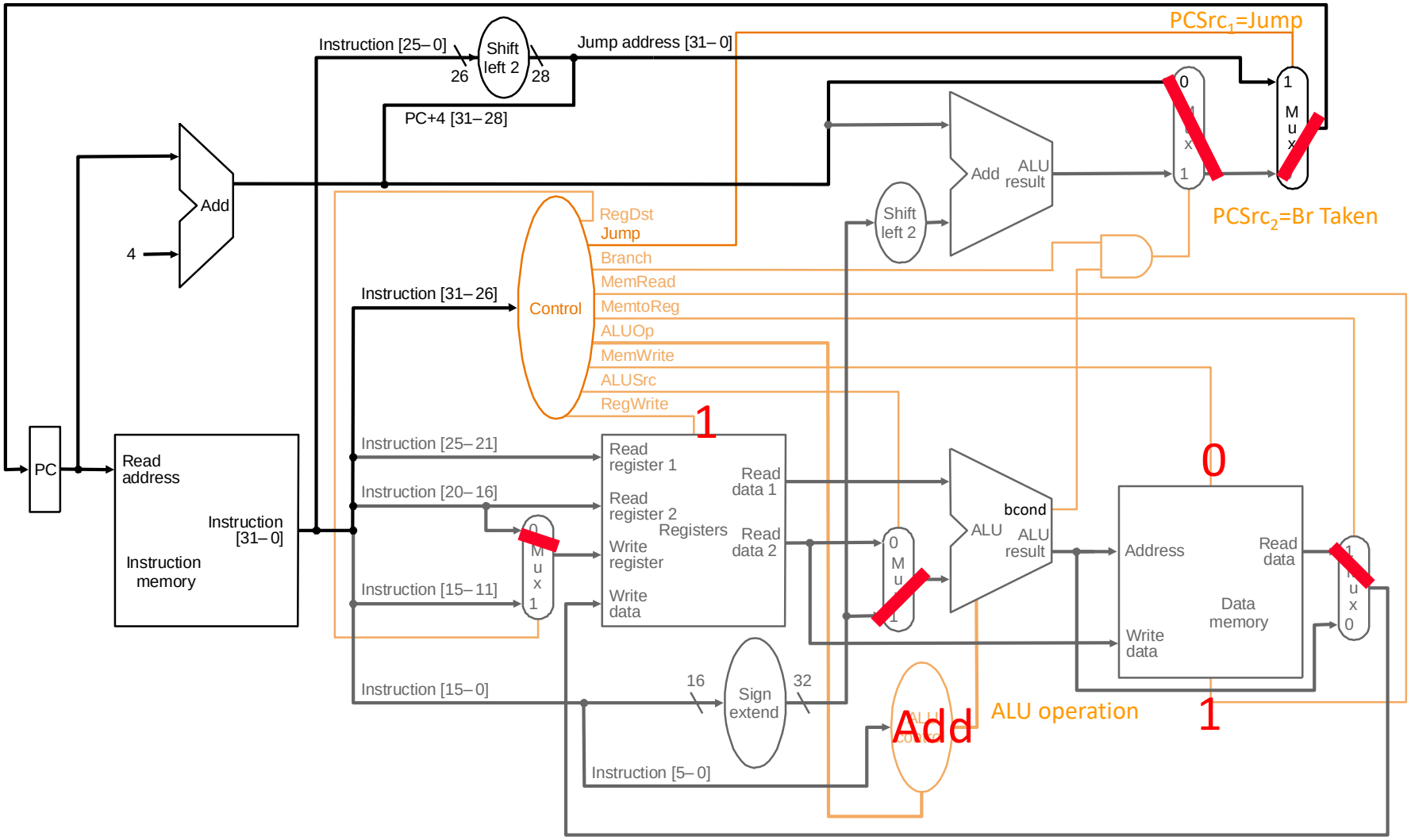
R-Type ALU


PCSrc1=Jump
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
Add
ALU result
Shift left 2
M u x
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
1
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
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
bcond
ALU result
0
Address
Read data
Data memory
Write data
funct
ALU operation

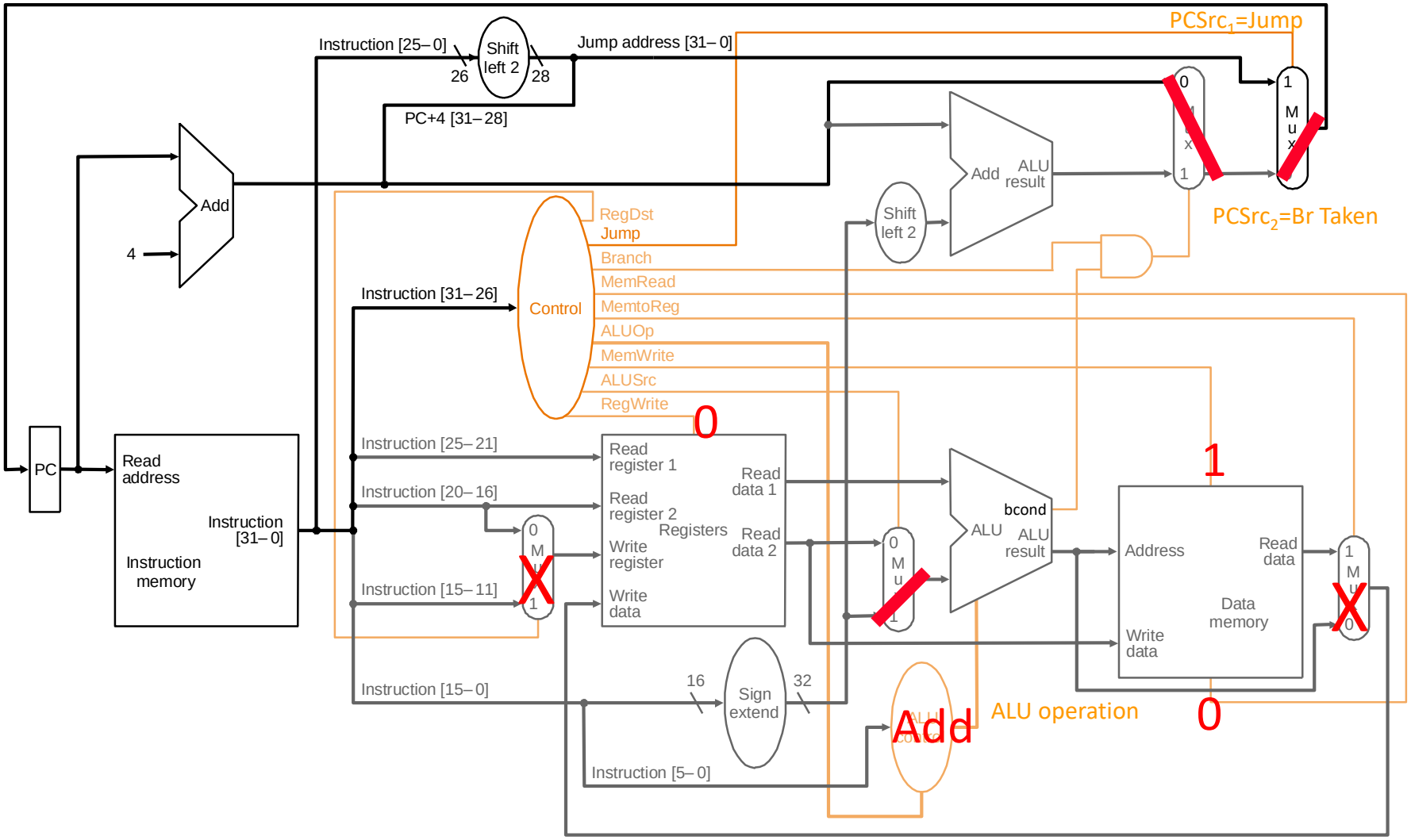
I-Type ALU


PCSrc1=Jump
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
Add
ALU result
Shift left 2
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
1
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
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
bcond
ALU result
0
ALU operation
opcode
Address
Read data
Data memory
Write data
0
M u x
1

LW

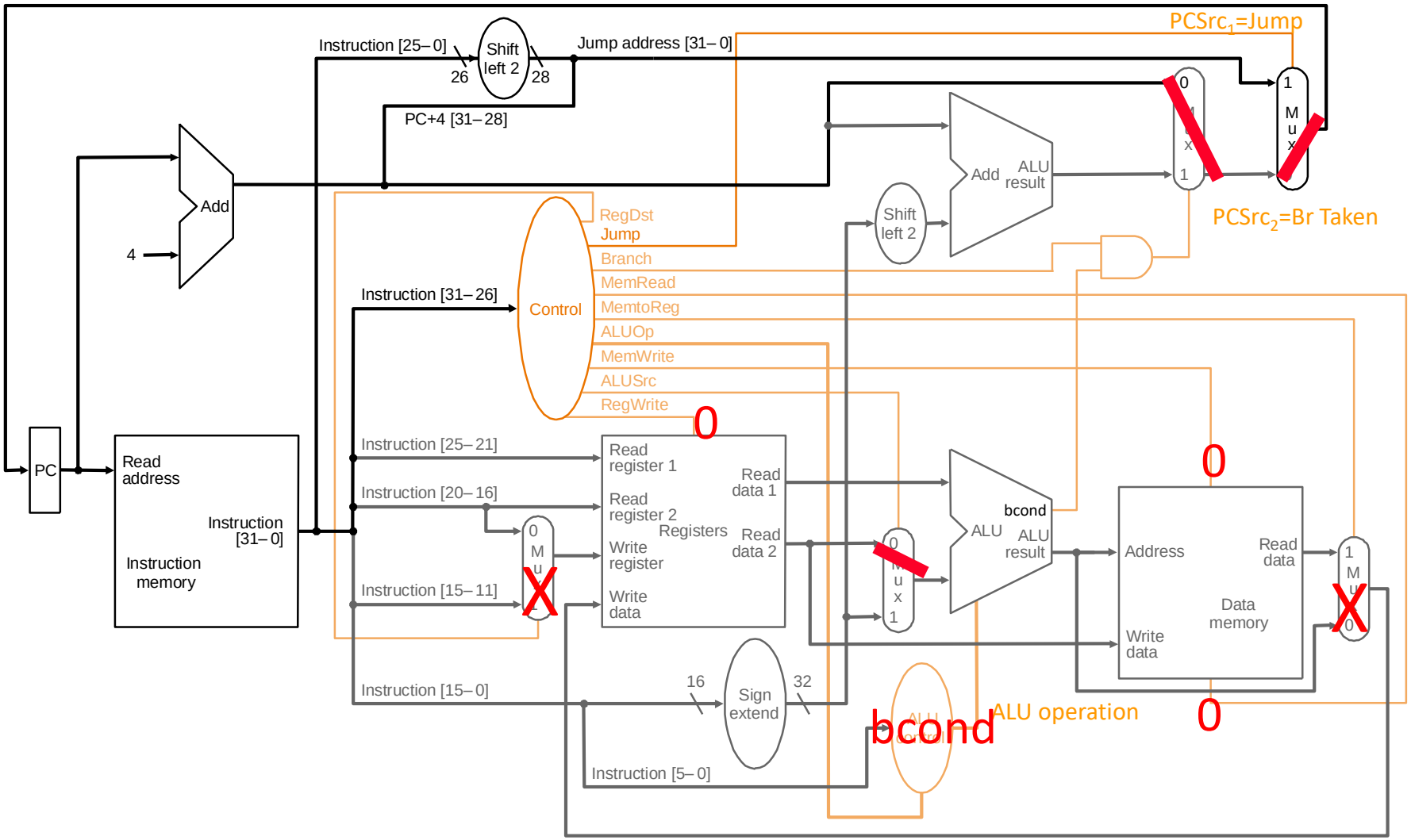

PCSrc1=Jump
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
0
M u x
1
1
M u x
Add
ALU result
Shift left 2
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
1
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0
M u x
1
16
Sign extend
32
ALU
bcond
ALU result
Add
ALU operation
0
Address
Read data
Data memory
Write data
1
M u x
0
1

SW


PCSrc1=Jump
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
0
M u x
1
1
M u x
Add
ALU result
Shift left 2
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
0
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
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
bcond
ALU result
Add
ALU operation
1
0
Address
Read data
Data memory
Write data

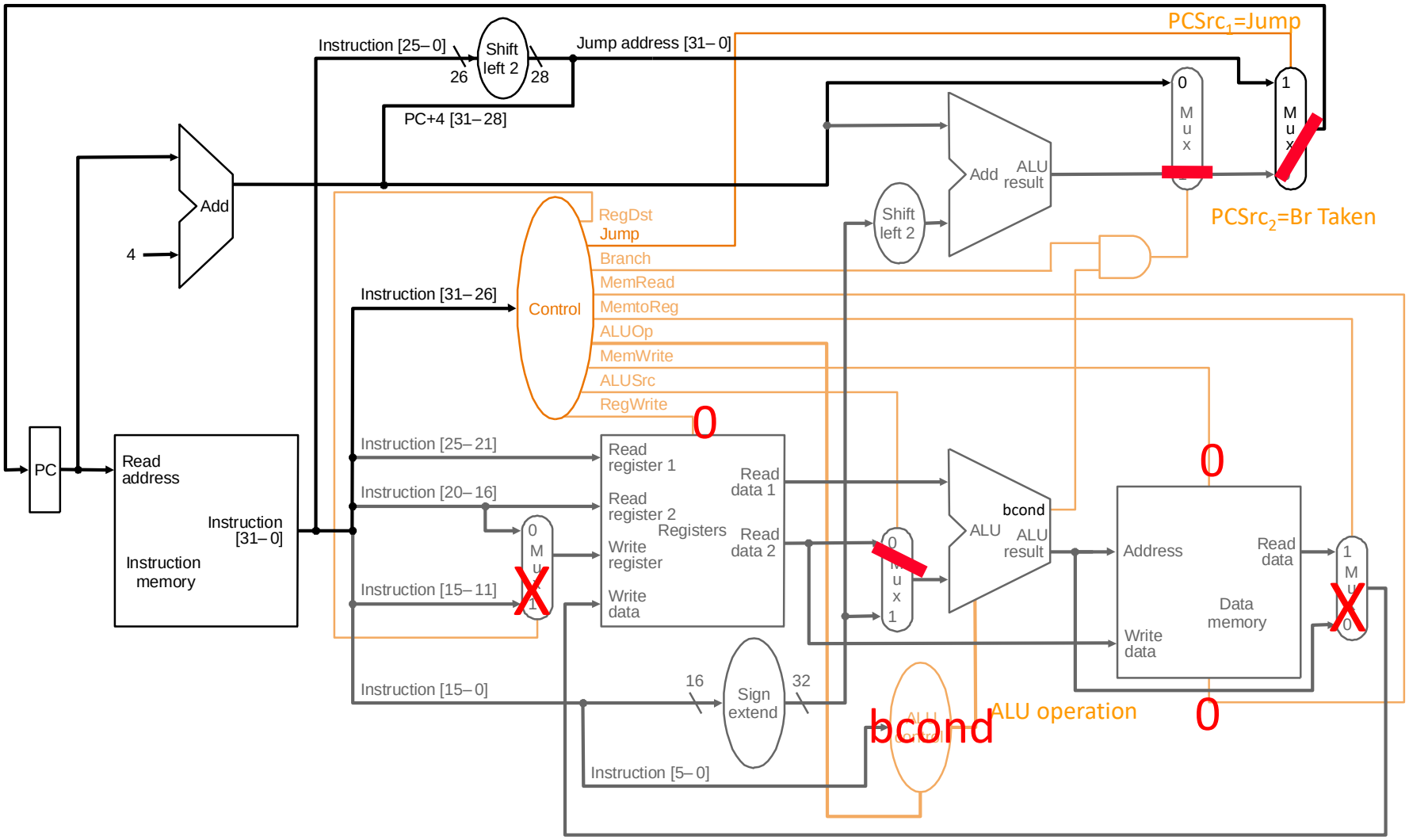
Branch (Not Taken)

Some control signals are dependent on the processing of data

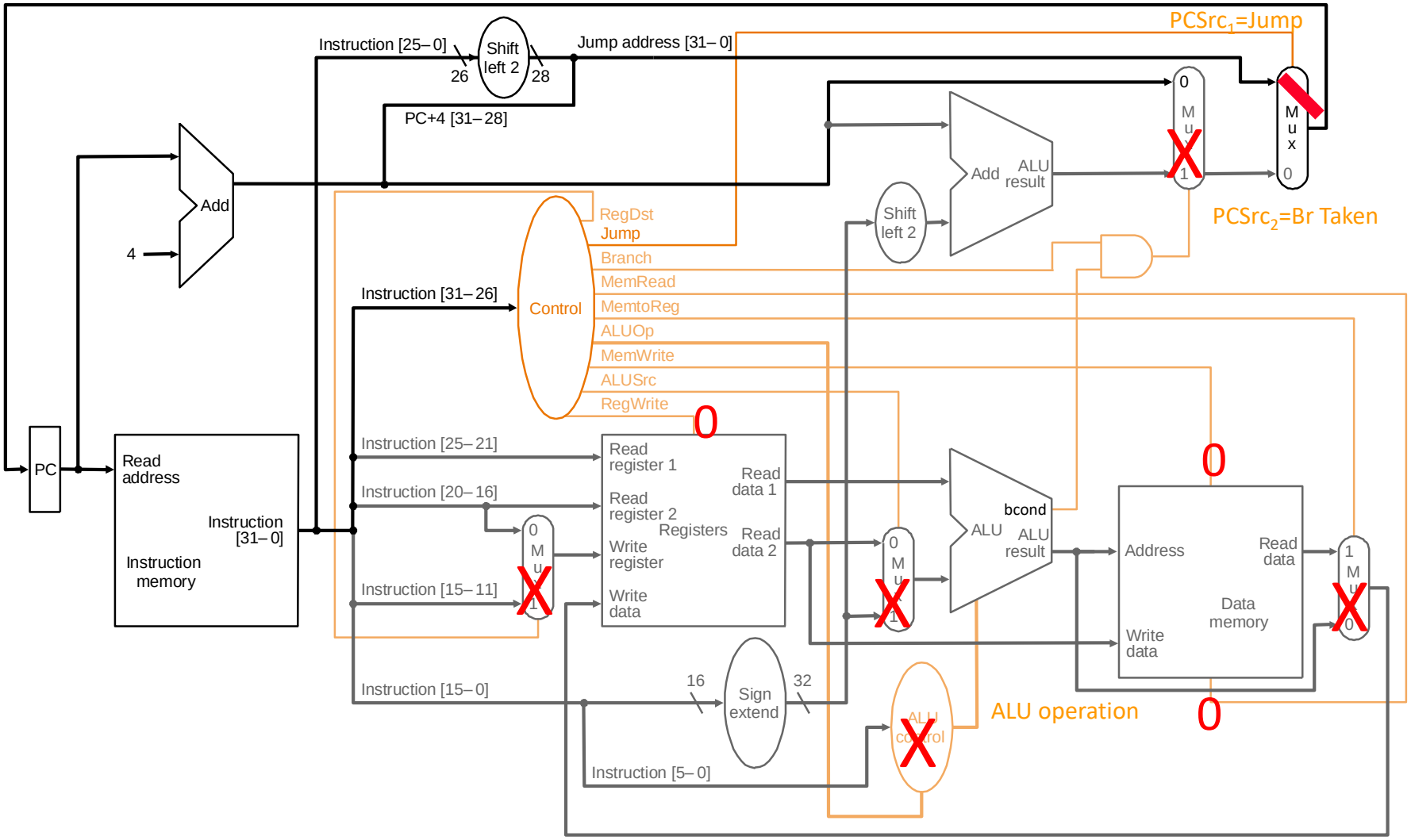


Branch (Taken)

Some control signals are dependent on the processing of data



Jump


PCSrc1=Jump
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
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
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
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
bcond
ALU result
ALU control
ALU operation
Address
Read data
Data memory
Write data
0
0
0

What is in That Control Box?

- Combinational Logic → **Hardwired Control**
  - Idea: Control signals generated combinationally based on instruction
  - Necessary in a single-cycle microarchitecture…
- Sequential Logic → **Sequential/Microprogrammed Control**
  - Idea: A memory structure contains the control signals associated with an instruction
  - Control Store

A Single-Cycle Microarchitecture: Analysis

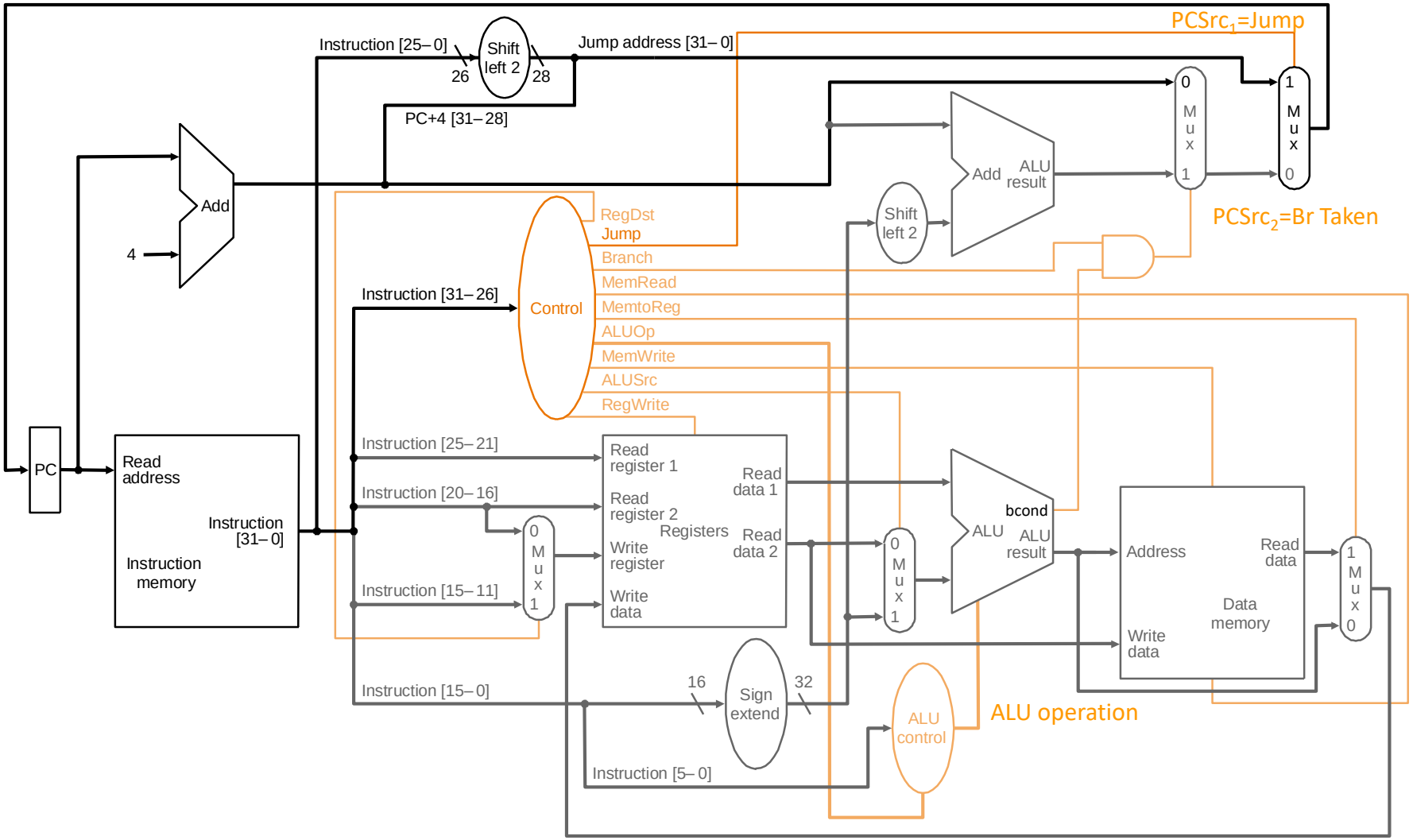
- Every instruction takes 1 cycle to execute
  - CPI (Cycles per instruction) is strictly 1
- How long each instruction takes is determined by how long the slowest instruction takes to execute
  - Even though many instructions do not need that long to execute
- Clock cycle time of the microarchitecture is determined by how long it takes to complete the slowest instruction
  - Critical path of the design is determined by the processing time of the slowest instruction

Single-Cycle Datapath Analysis

- **Assume**
  - **memory units (read or write): 200 ps**
  - **ALU and adders: 100 ps**
  - **register file (read or write): 50 ps**
  - **other combinational logic: 0 ps**

steps	IF	ID	EX	MEM	WB	Delay
resources	mem	RF	ALU	mem	RF	
R-type						
I-type						
LW						
SW						
Branch						
Jump						

Let's Find the Critical Path

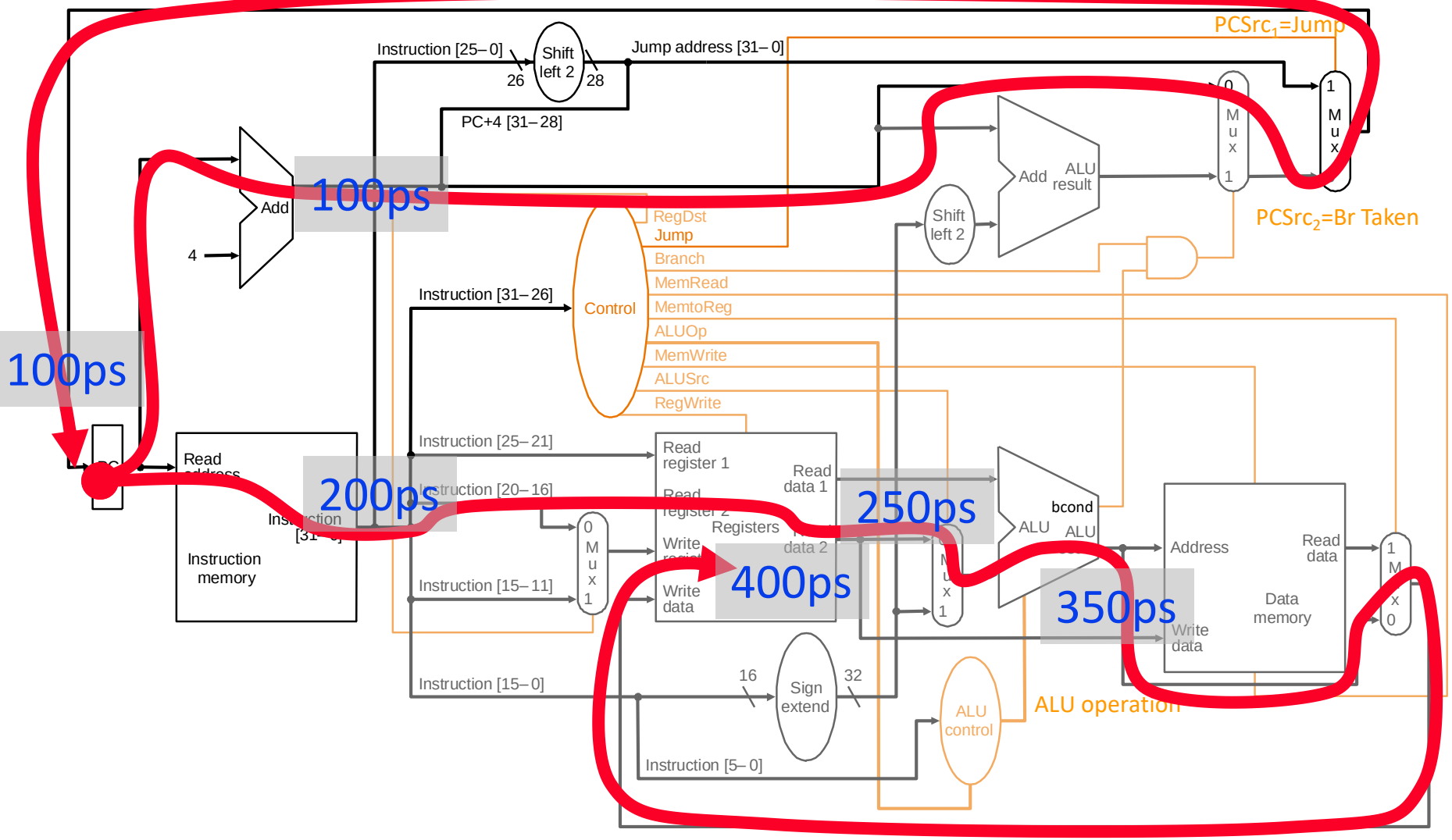


Single-Cycle Datapath Analysis

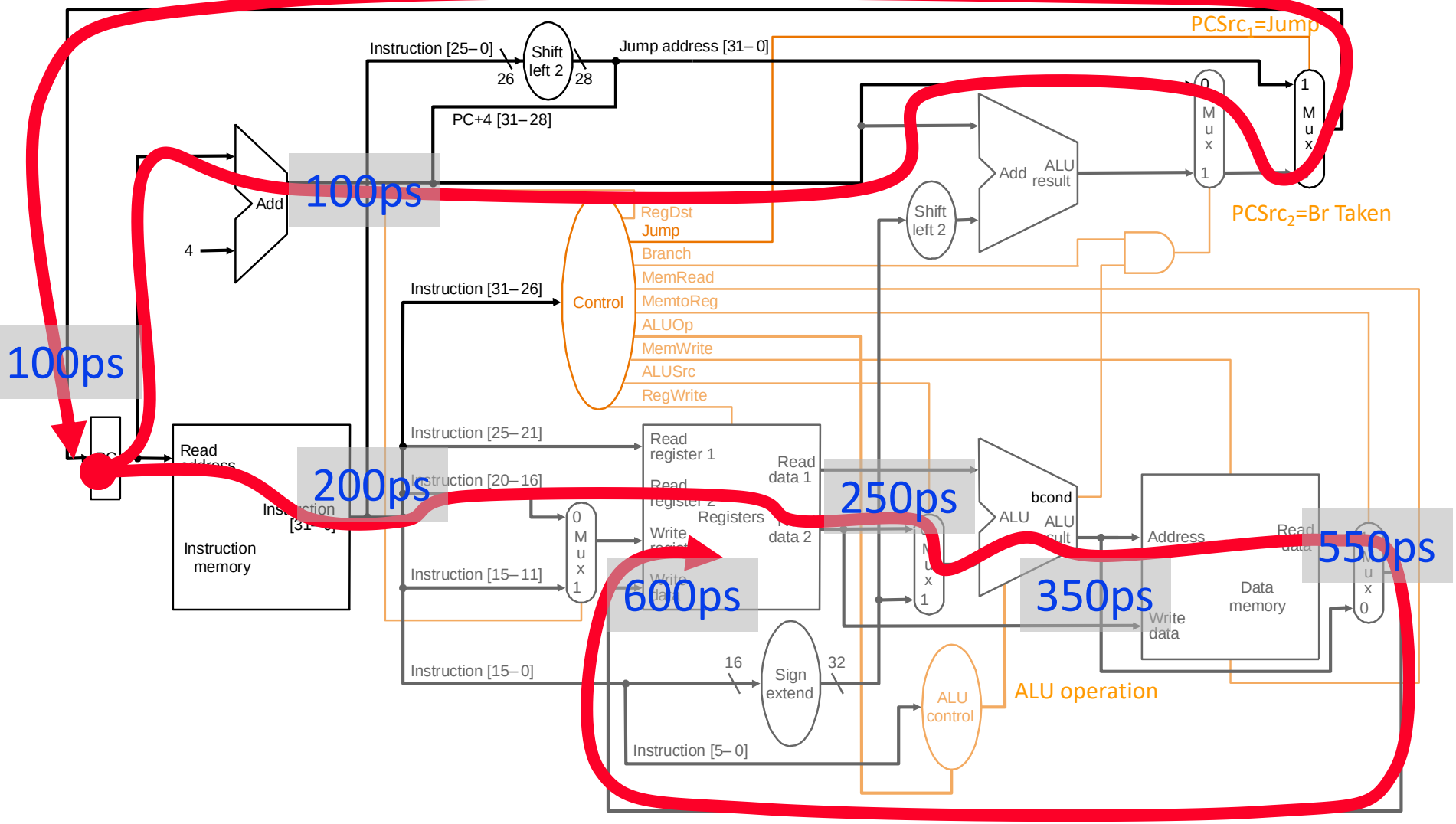
- **Assume**
  - memory units (read or write): 200 ps
  - ALU and adders: 100 ps
  - register file (read or write): 50 ps
  - other combinational logic: 0 ps

steps	IF	ID	EX	MEM	WB	Delay
resources	mem	RF	ALU	mem	RF	
R-type	200	50	100		50	400
I-type	200	50	100		50	400
LW	200	50	100	200	50	600
SW	200	50	100	200		550
Branch	200	50	100			350
Jump	200					200

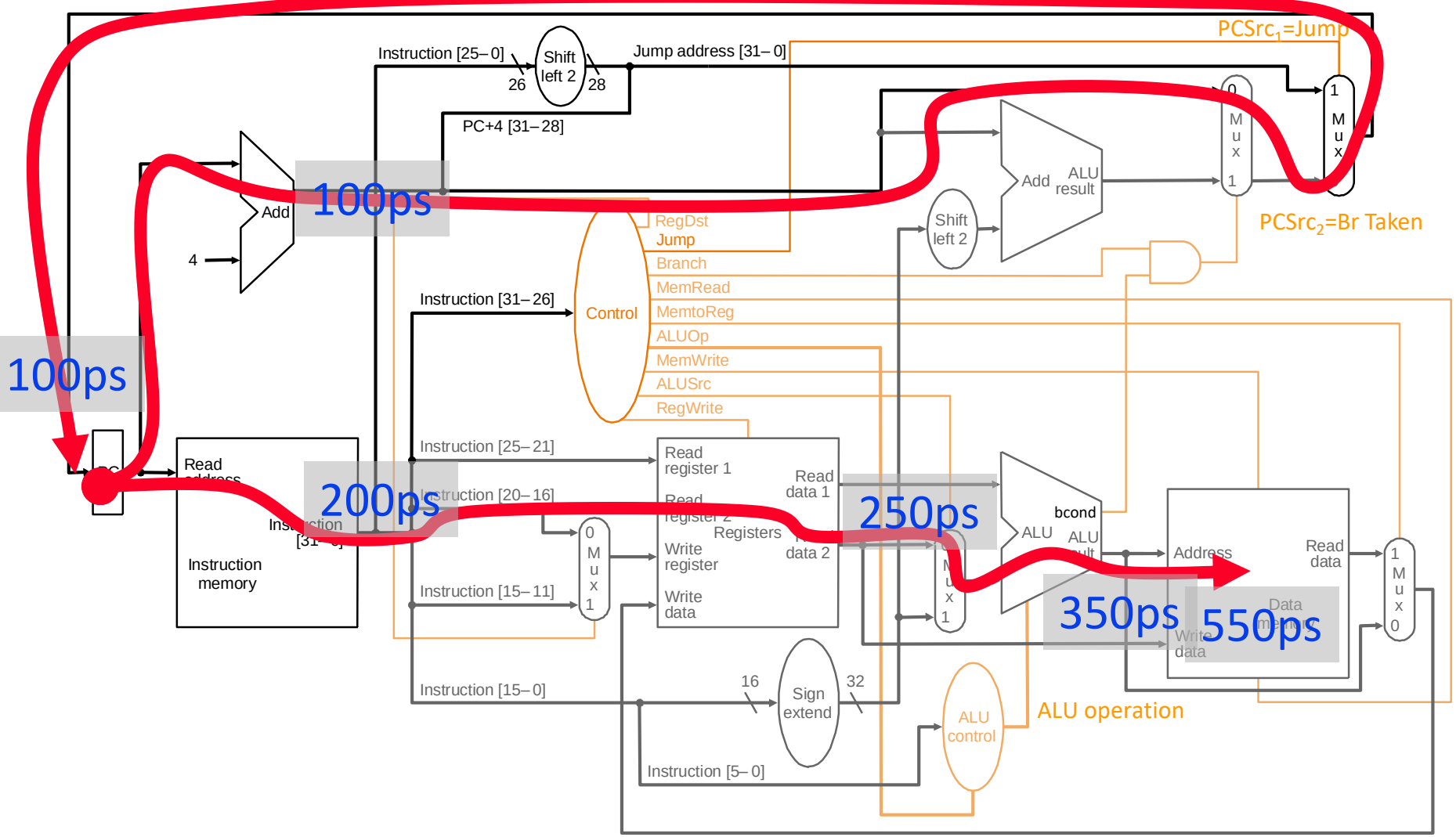
R-Type and I-Type ALU



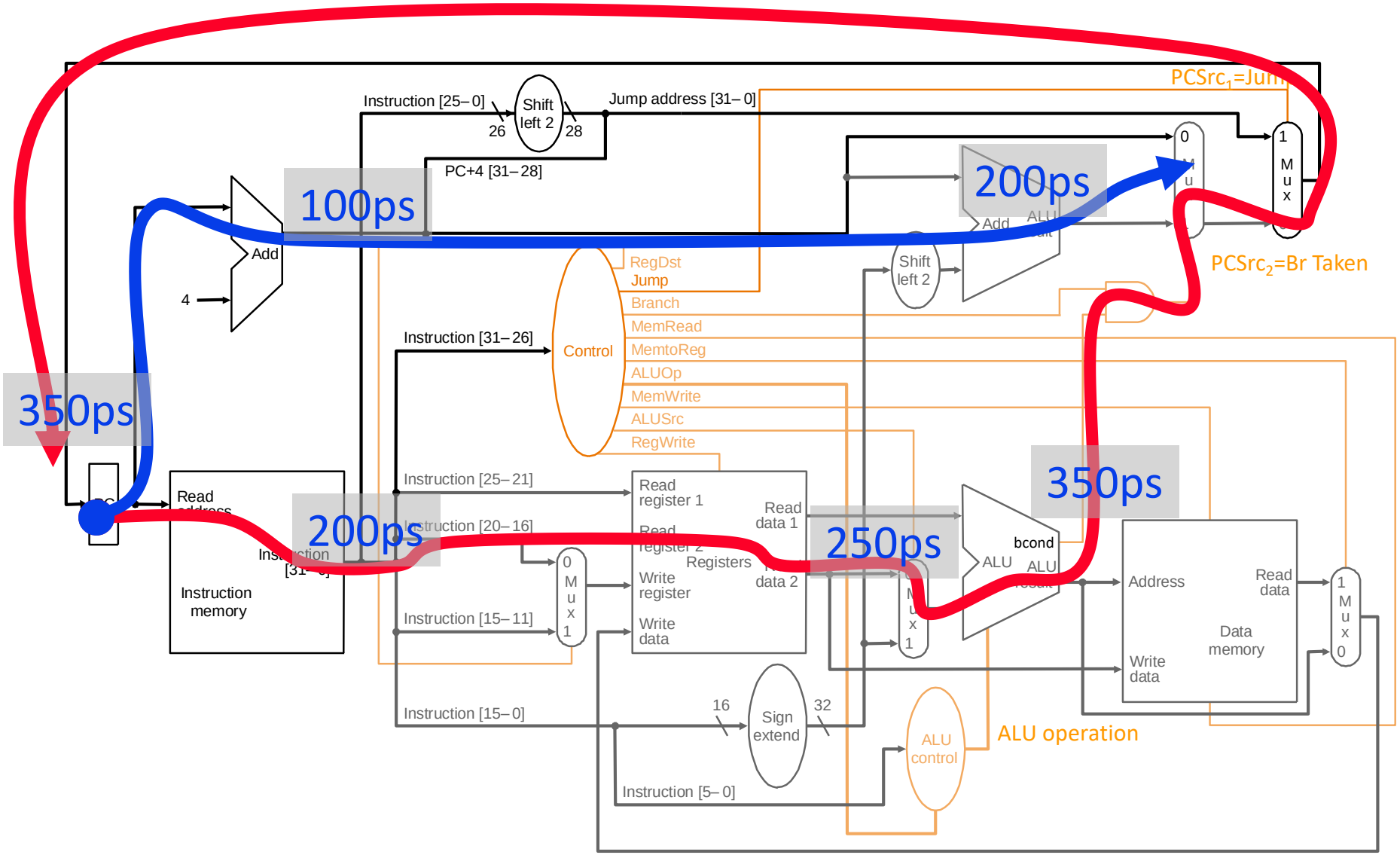
LW


PCSrc1=Jump
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
100ps
0
M u x
1
1
M u x
Add
ALU result
Shift left 2
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
100ps
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
200ps
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0
M u x
1
600ps
250ps
ALU
bcond
ALU result
M u x
1
350ps
Address
Read data
Data memory
Write data
M u x
0
550ps
16
Sign extend
32
ALU control
ALU operation

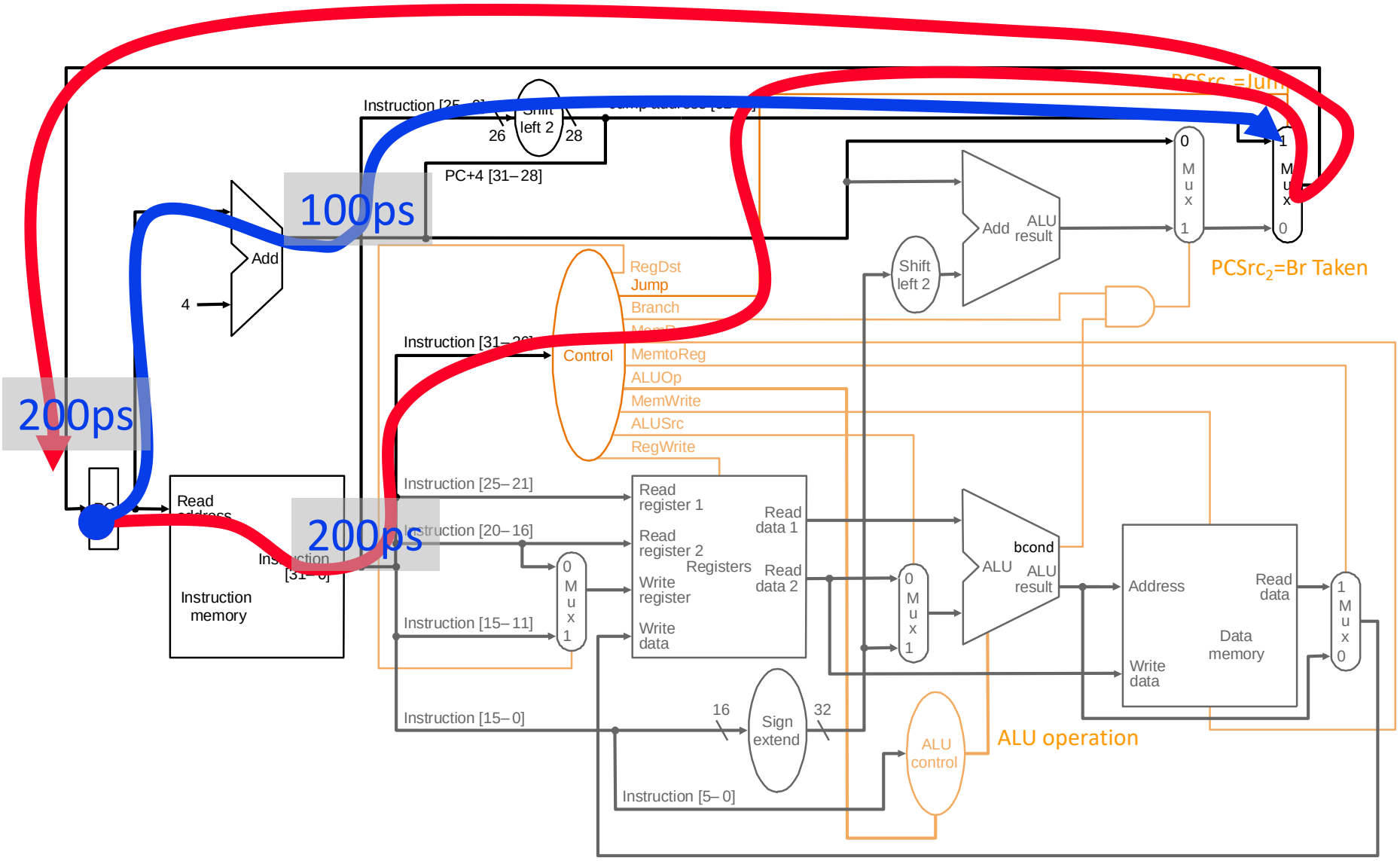
SW


PCSrc1=Jump
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
100ps
0
M u x
1
1
M u x
Add
ALU result
Shift left 2
PCSrc2=Br Taken
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
100ps
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
200ps
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0
M u x
1
250ps
ALU
bcond
ALU result
Address
Read data
Data memory
Write data
350ps
550ps
16
Sign extend
32
ALU control
ALU operation

Branch Taken



Jump


100ps
200ps
200ps
PCSrc =Jump
Instruction [25–0]
Shift left 2
26
28
PC+4 [31–28]
Add
4
0
M
u
x
1
Add
ALU result
Shift left 2
PCSrc2=Br Taken
RegDst
Jump
Branch
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
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
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
ALU
bcond
ALU result
ALU control
ALU operation
Address
Read data
Data memory
Write data

Sample Question

- The basic single-cycle MIPS implementation can only implement some instructions. New instructions can be added to an existing ISA, but the decision whether or not to do that depends, among other things, on the cost and complexity such an addition introduces into the processor datapath and control. The new instructions are shown below:

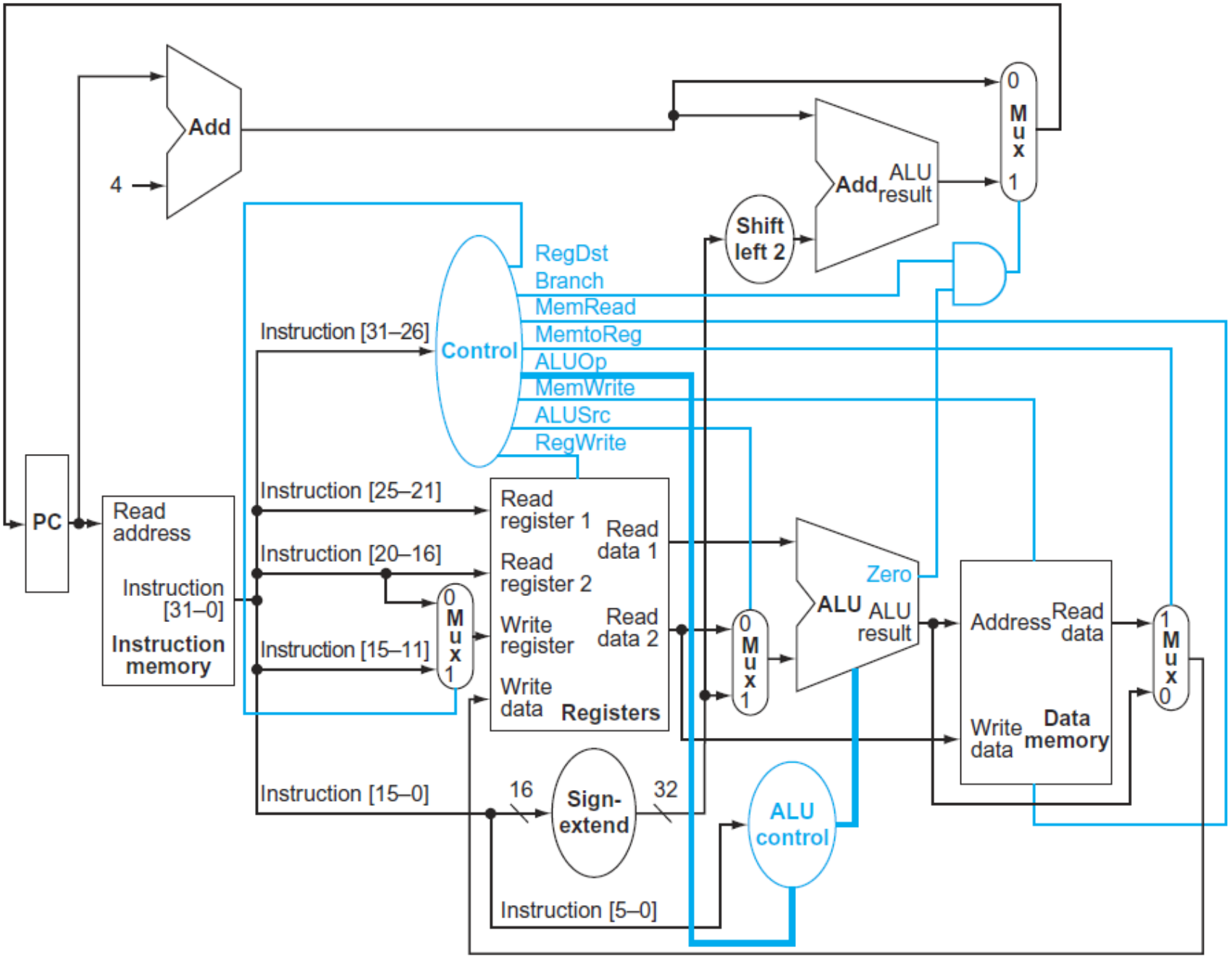
- jri: jump register + immediate
  - jri $s0 IMM
  - PC = R[rs] + sign-extend(IMM)

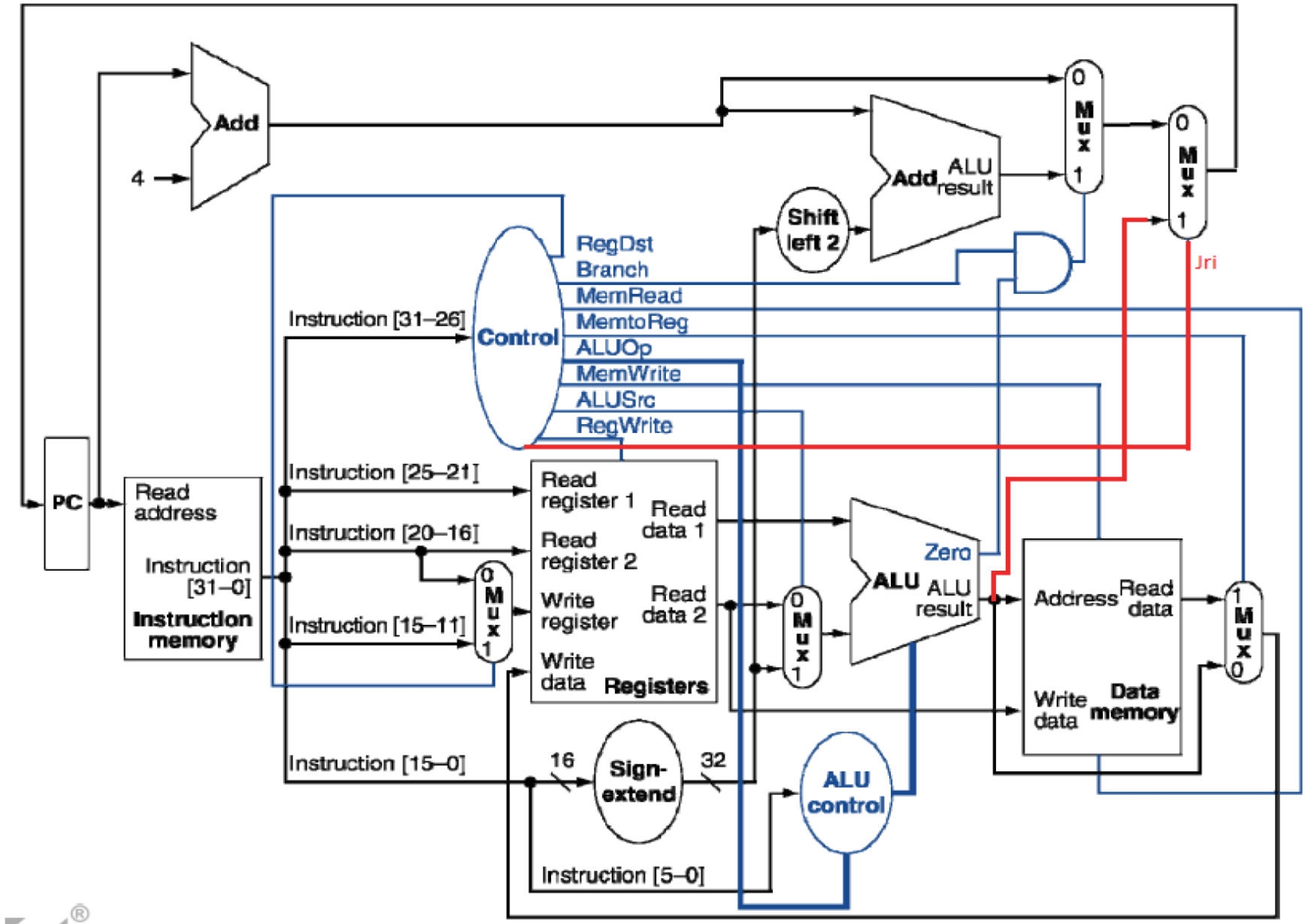
Sample Question

- PC = R[rs] + sign-extend(IMM)
- What type of instruction should this be?
- We need a register and an immediate. I-type fits the bill.
  - We'd have a unused register if we implement it as an I-type
- We could possibly optimize by defining a new instruction type that has one register and one immediate
  - Trade off?

Sample Question

- PC = R[rs] + sign-extend(IMM)
- What new connections must be made?
- Additional connections
  - Already hooked up in such a way as to do addition of register value and a sign extended immediate
  - From output of ALU to PC
- How do we choose the output of the ALU as the input to the PC
  - Need a MUX somewhere before the PC
  - Also a new control signal to selection input


Add
4
0
M
u
x
1
Add
ALU result
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
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 control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory


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

Slide credits

- Onur Mutlu