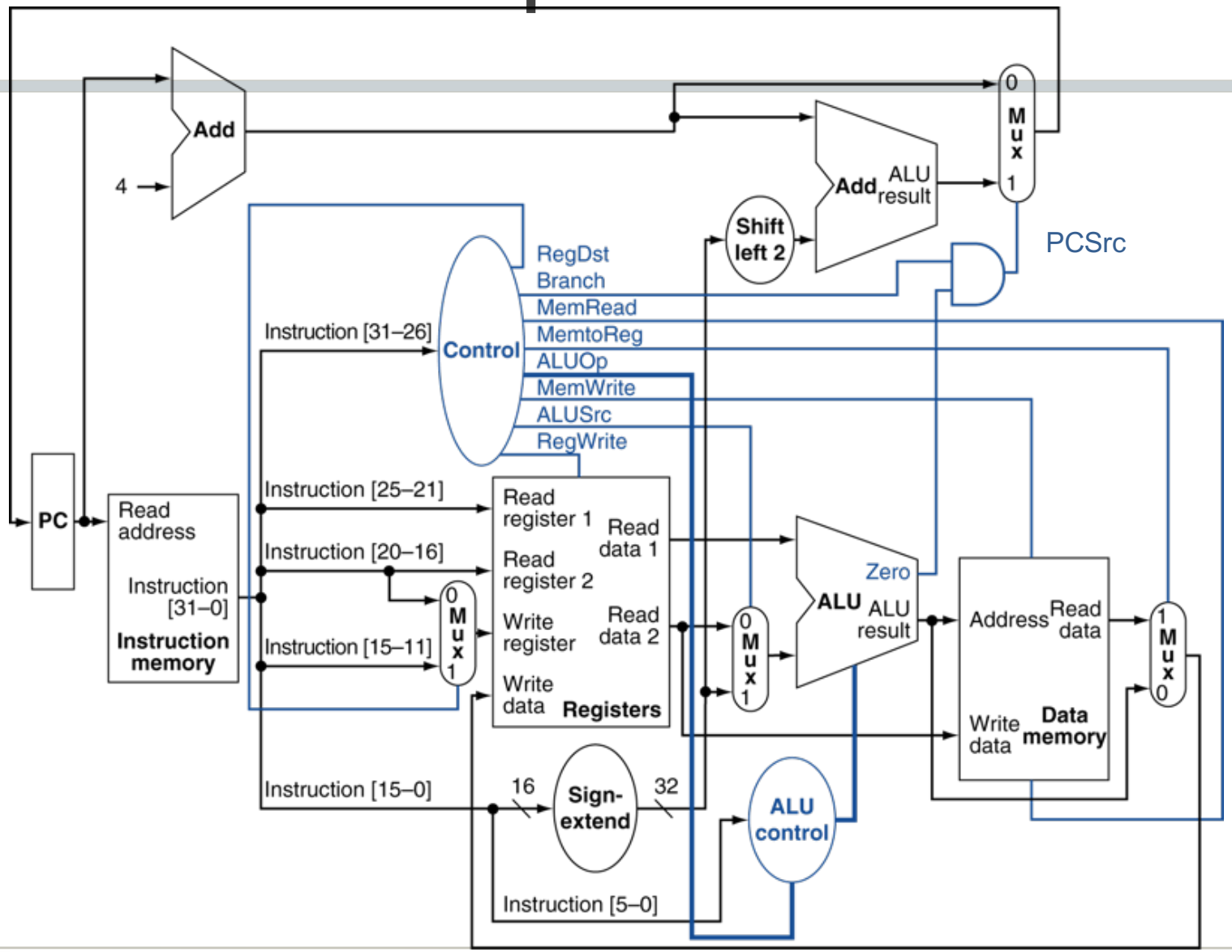
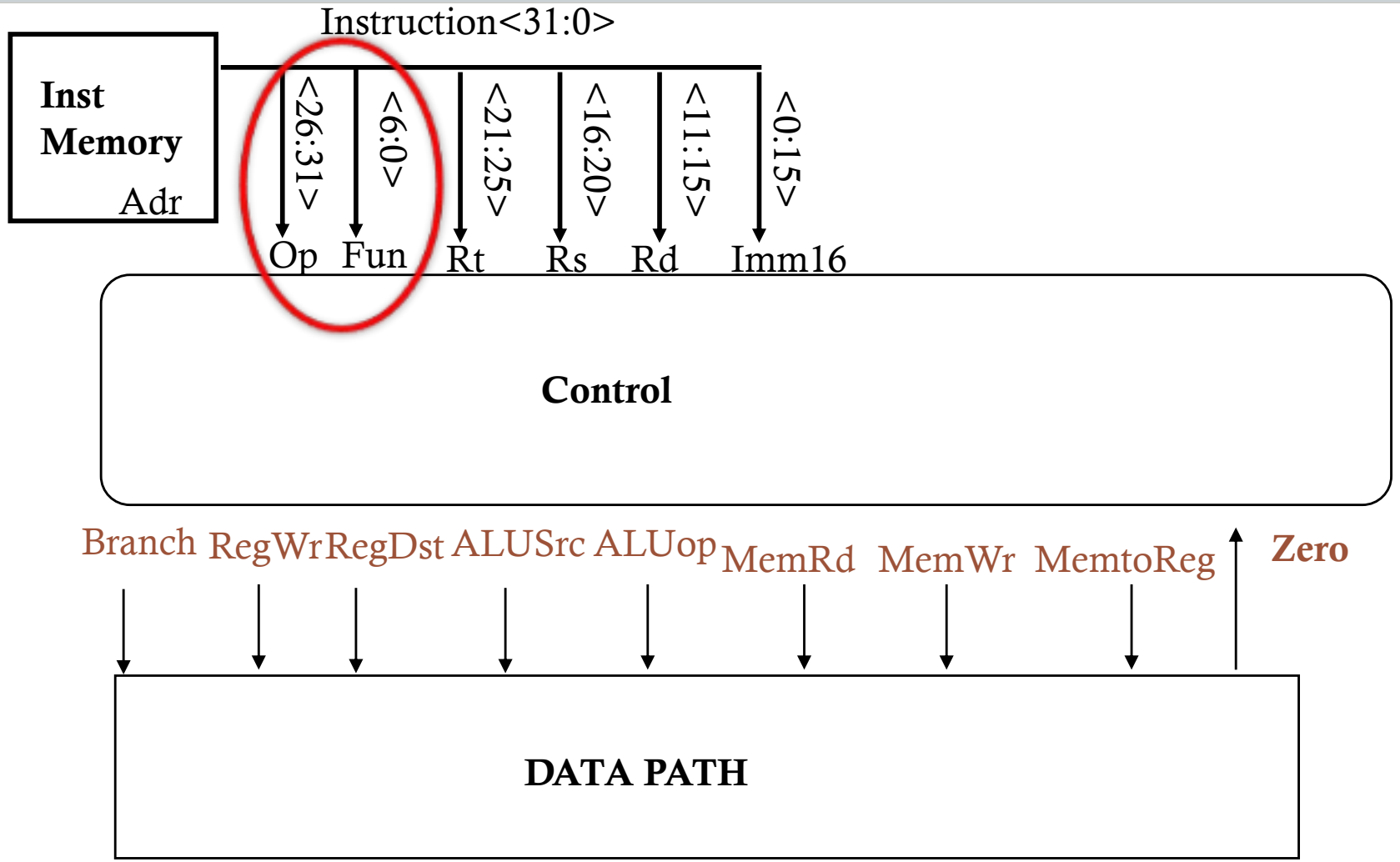
RoadMap

- 1. Analyze instruction set ⇒ datapath requirements √
- 2. Select set of datapath components and establish clocking methodology √
- 3. Assemble datapath meeting the requirements √
- 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer ←
- 5. Assemble the control logic

Review: Datapath w/ Control



Review: Main Control



Design the Main Control Unit

- Seven control signals in addition to ALUOp

  RegWrite

  MemRead

  MemWrite

  RegDst *

  ALUSrc *

  MemtoReg *

  Branch(PCSrc*)

All derived from 6-bit Opcode

* is to a mux

Control Signals

1. RegDst = 0 => Register destination number for the Write register comes from the rt field (bits 20-16)

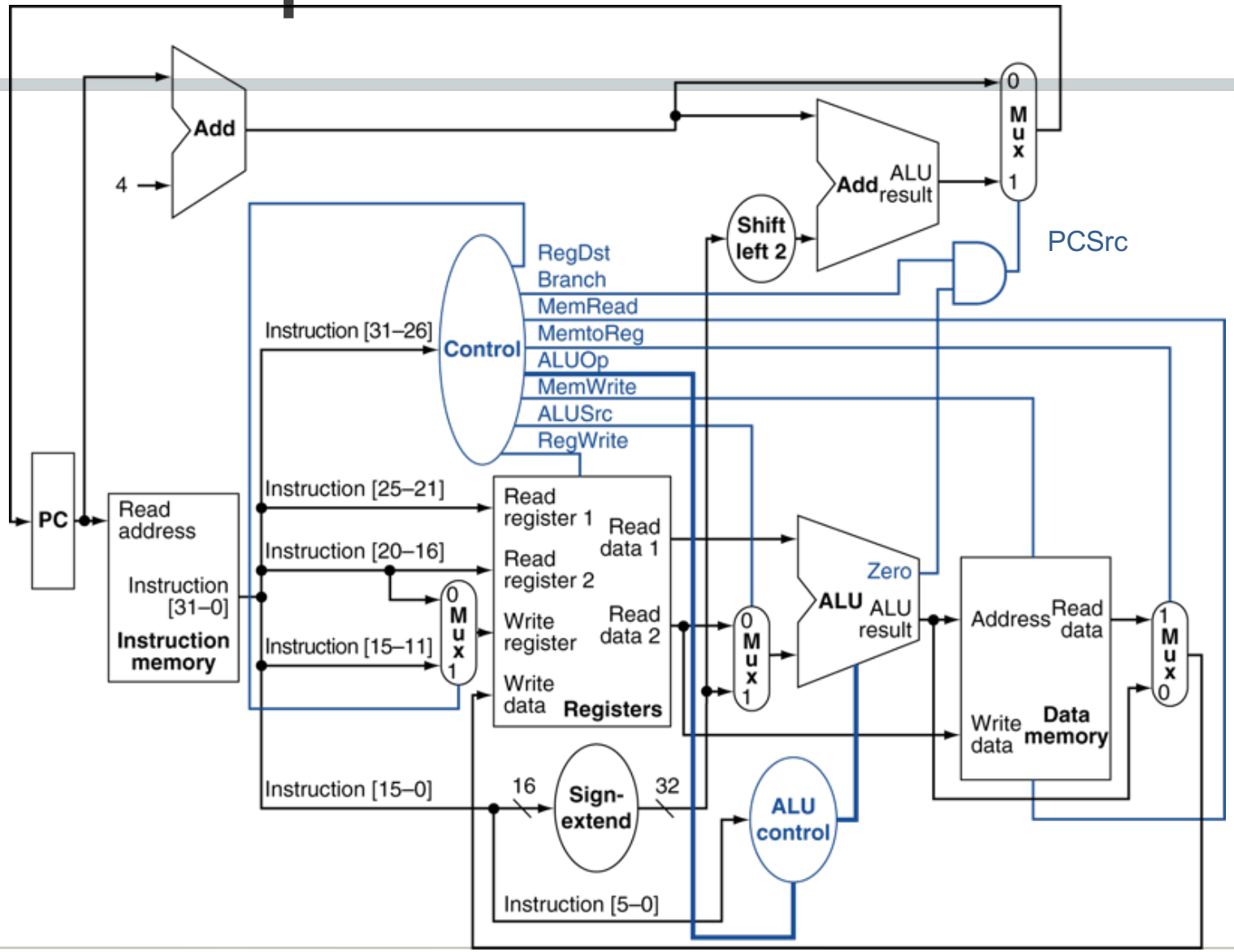
   RegDst = 1 => Register destination number for the Write register comes from the rd field (bits 15-11)
2. RegWrite = 1 => The register on the Write register input is written with the data on the Write data input (at the next clock edge)
3. ALUSrc = 0 => The second ALU operand comes from Read data 2

   ALUSrc = 1 => The second ALU operand comes from the sign-extension unit
4. Branch = 1 => branch instruction; PCSrc = 0 => The PC is replaced with PC+4

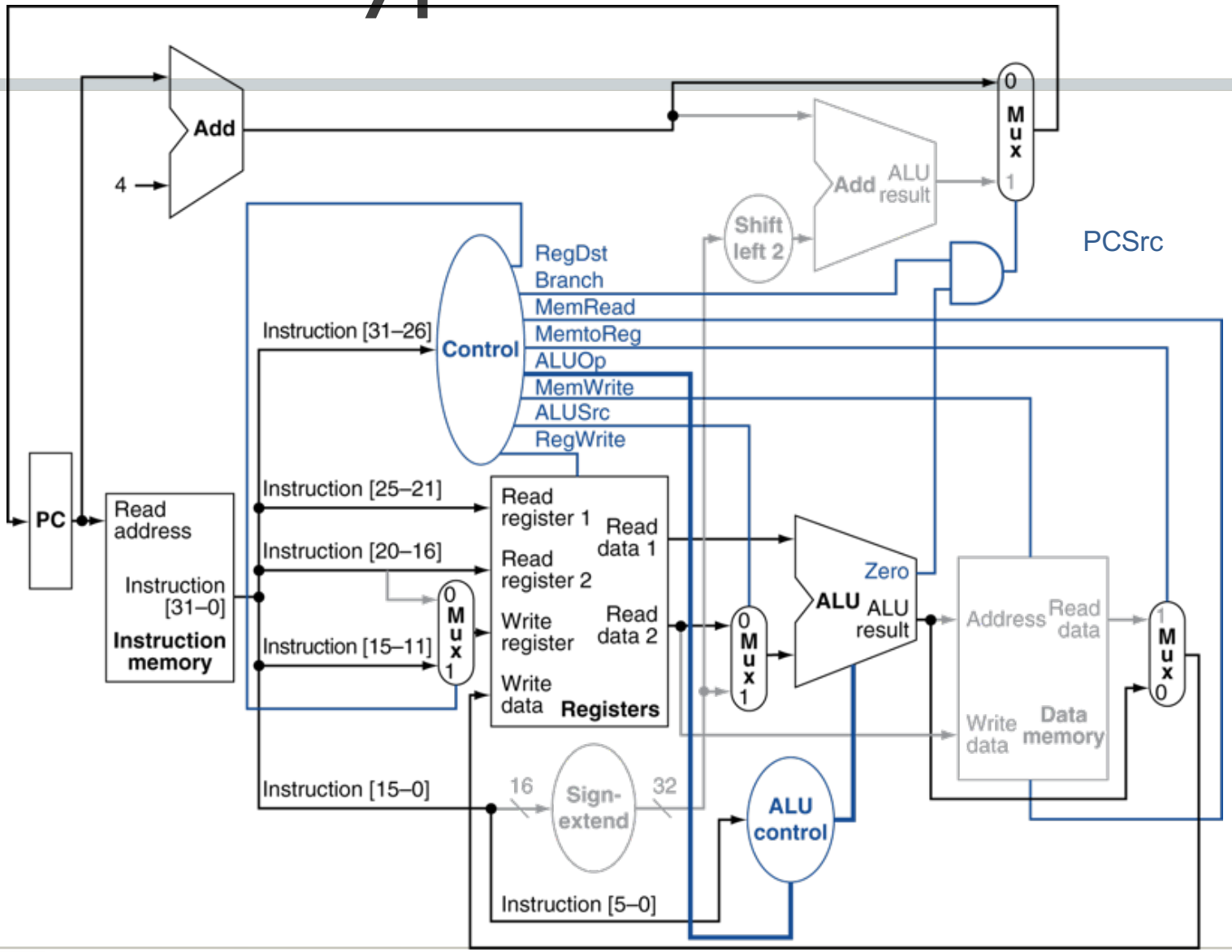
   PCSrc = 1 => The PC is replaced with the branch target address
5. MemtoReg = 0 => The value fed to the register write data input comes from the ALU

   MemtoReg = 1 => The value fed to the register write data input comes from the data memory
6. MemRead = 1 => Read data memory
7. MemWrite = 1 => Write data memory

Datapath With Control



R-Type Instruction



R-format Instructions

ALUOp = 10

MemRead = 0

MemWrite = 0

RegWrite = 1

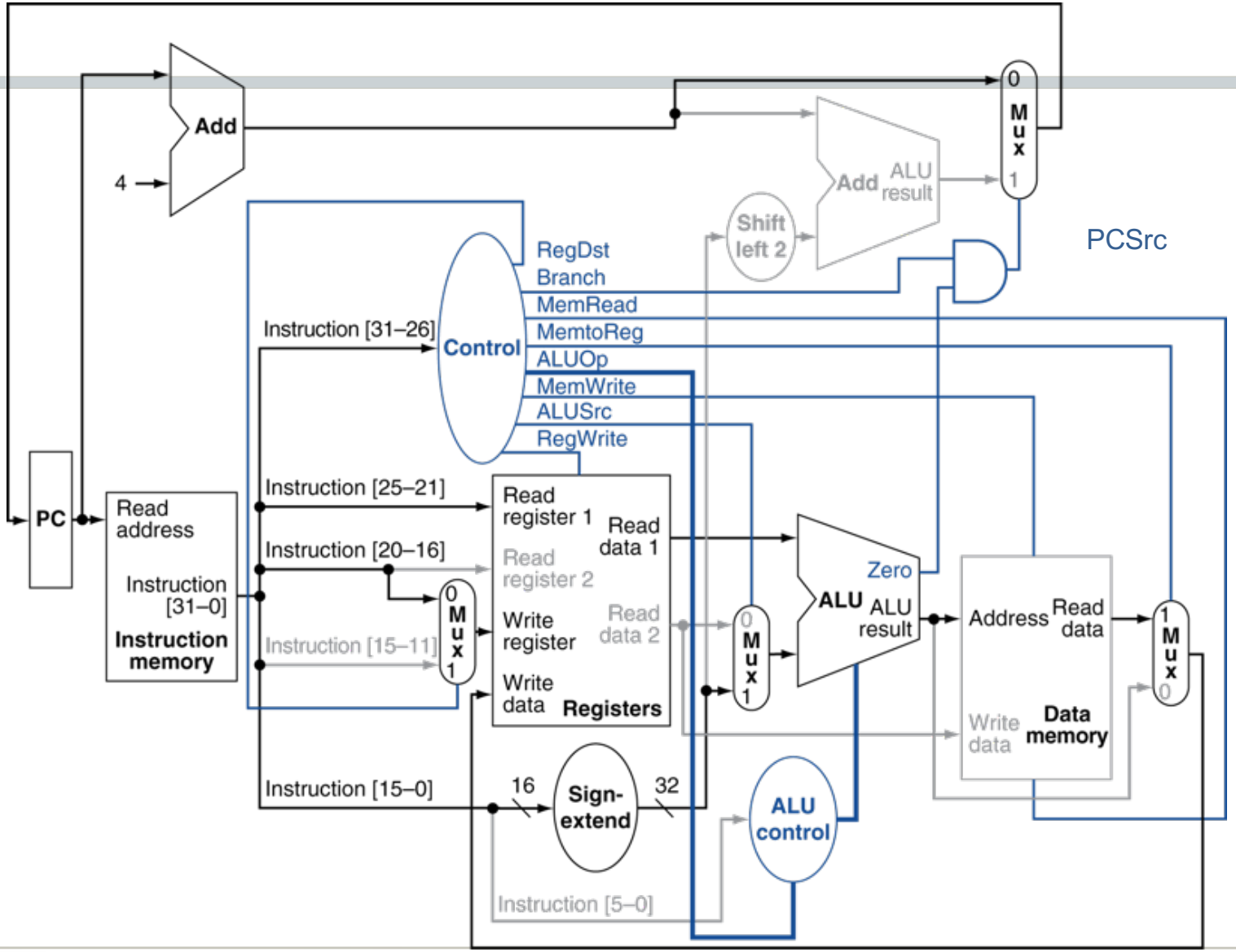
Branch = 0

RegDst = 1

ALUSrc = 0

MemtoReg = 0

Load Instruction



Memory Access Instructions

Load word

ALUOp = 00

MemRead = 1

MemWrite = 0

RegWrite = 1

Branch = 0

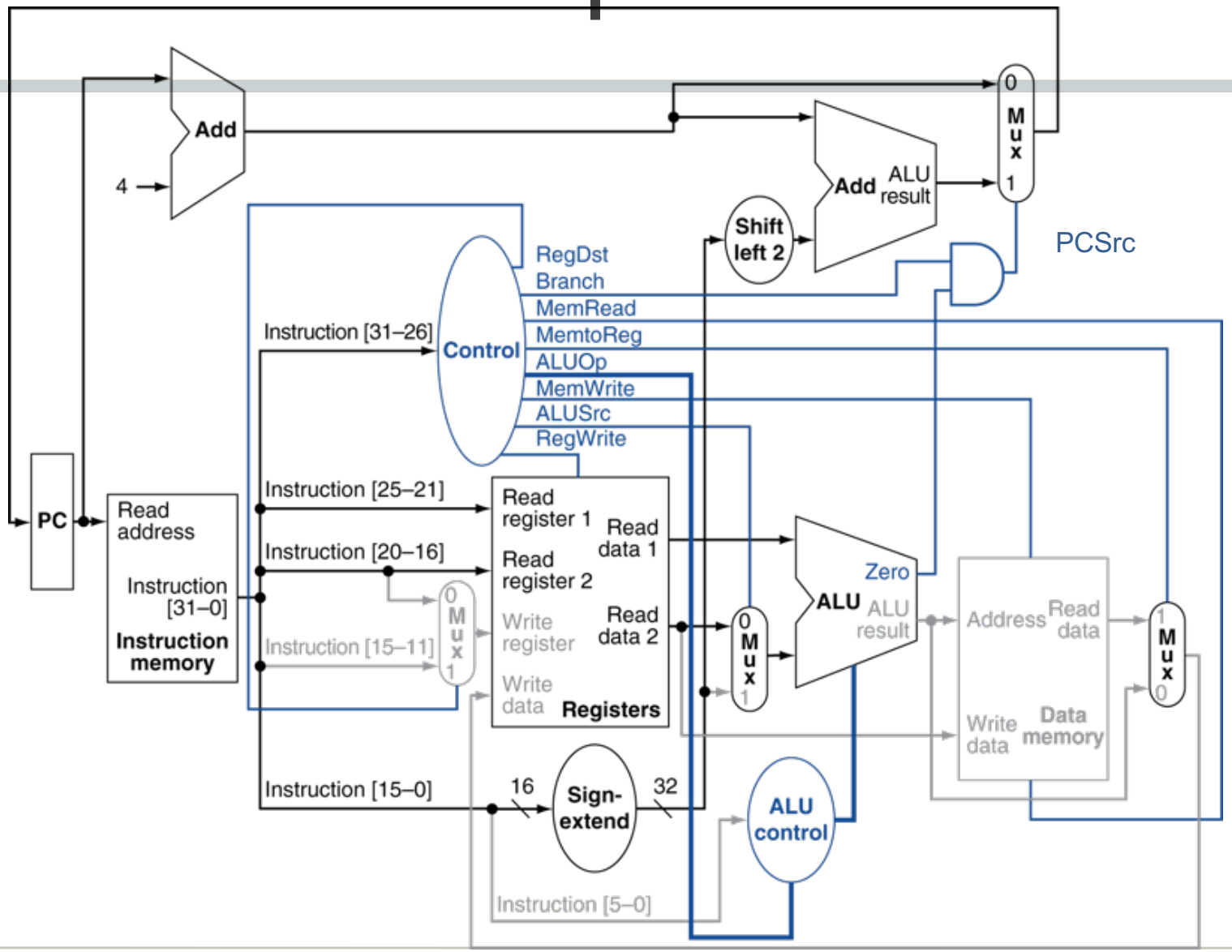
RegDst = 0

ALUSrc = 1

MemtoReg = 1

Store Word?

Branch-on-Equal Instruction



Branch on Equal

ALUOp = 01

MemRead = 0

MemWrite = 0

RegWrite = 0

Branch = 1

RegDst = X

ALUSrc = 0

MemtoReg = X

X indicates the signal is not used (don't care)

Step 5: Implementing Control

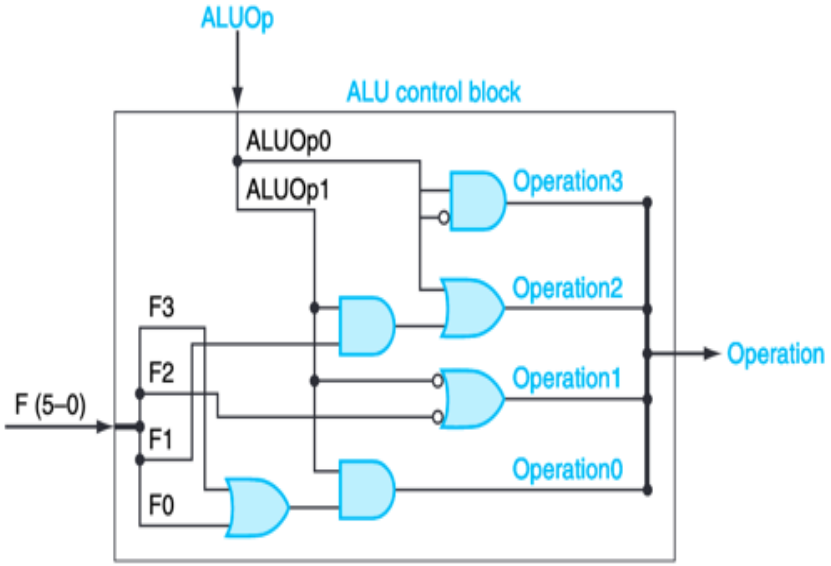
- Truth table based on instruction analysis

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

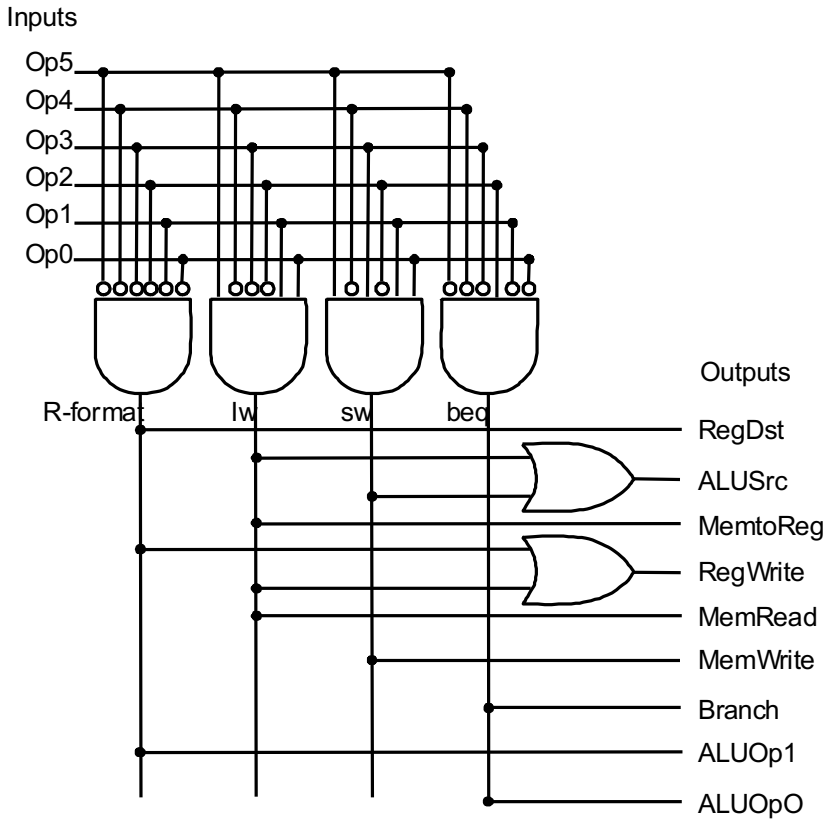
Fig 4.13 The truth table for the 4 ALU control bits (called Operation).

Step 5: Implementing Control

- Simple combinational logic based on truth tables

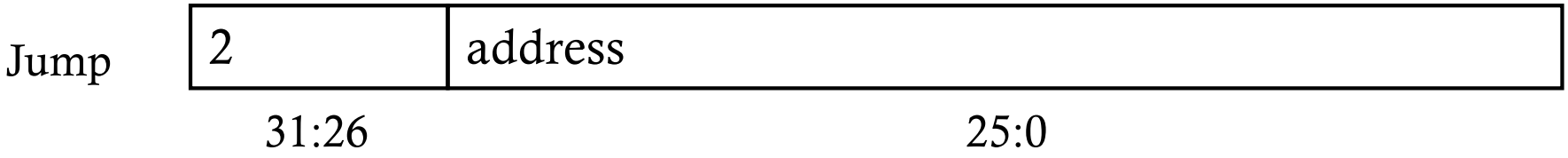


ALU Control Unit



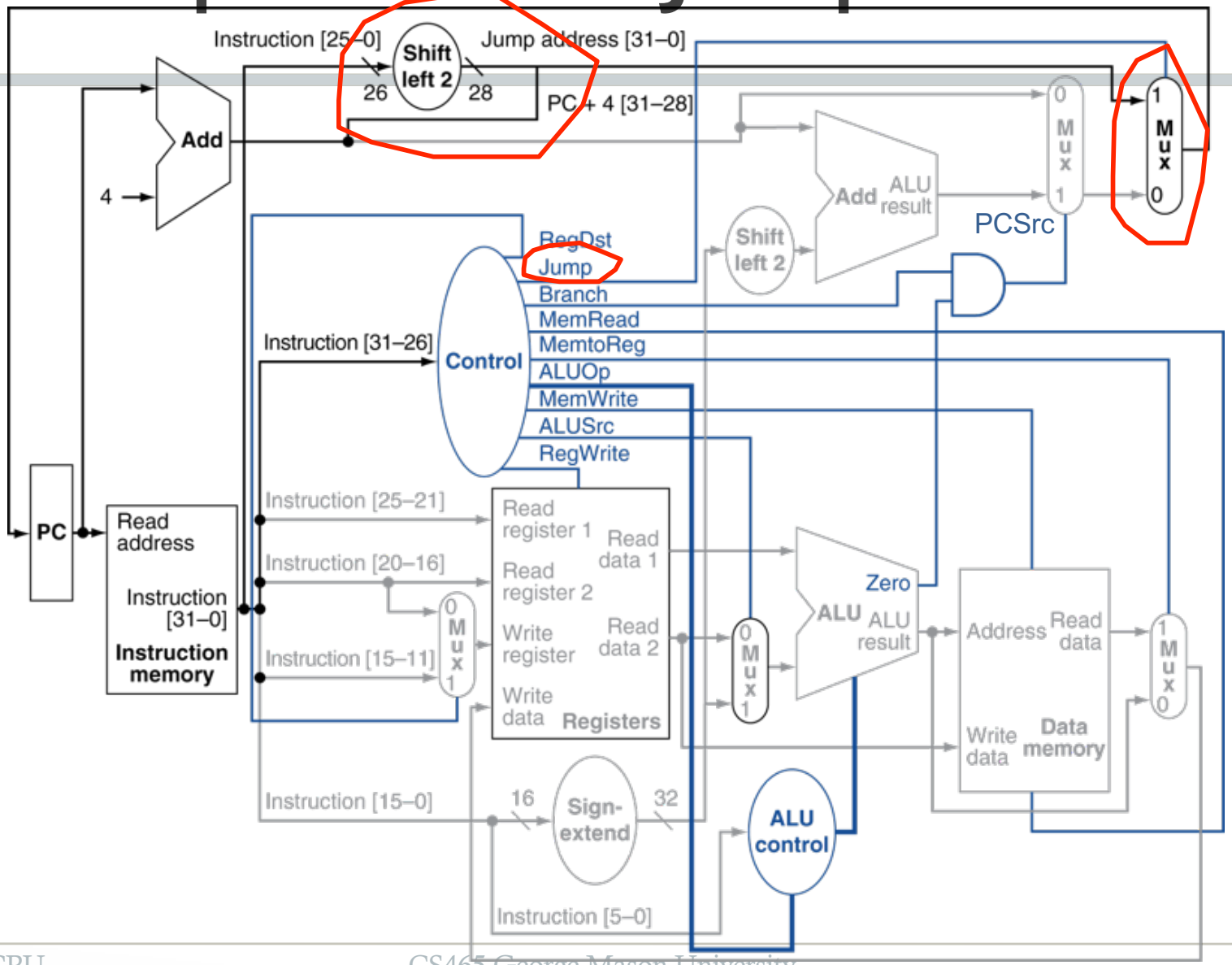
Main Control Unit

Implementing Jumps

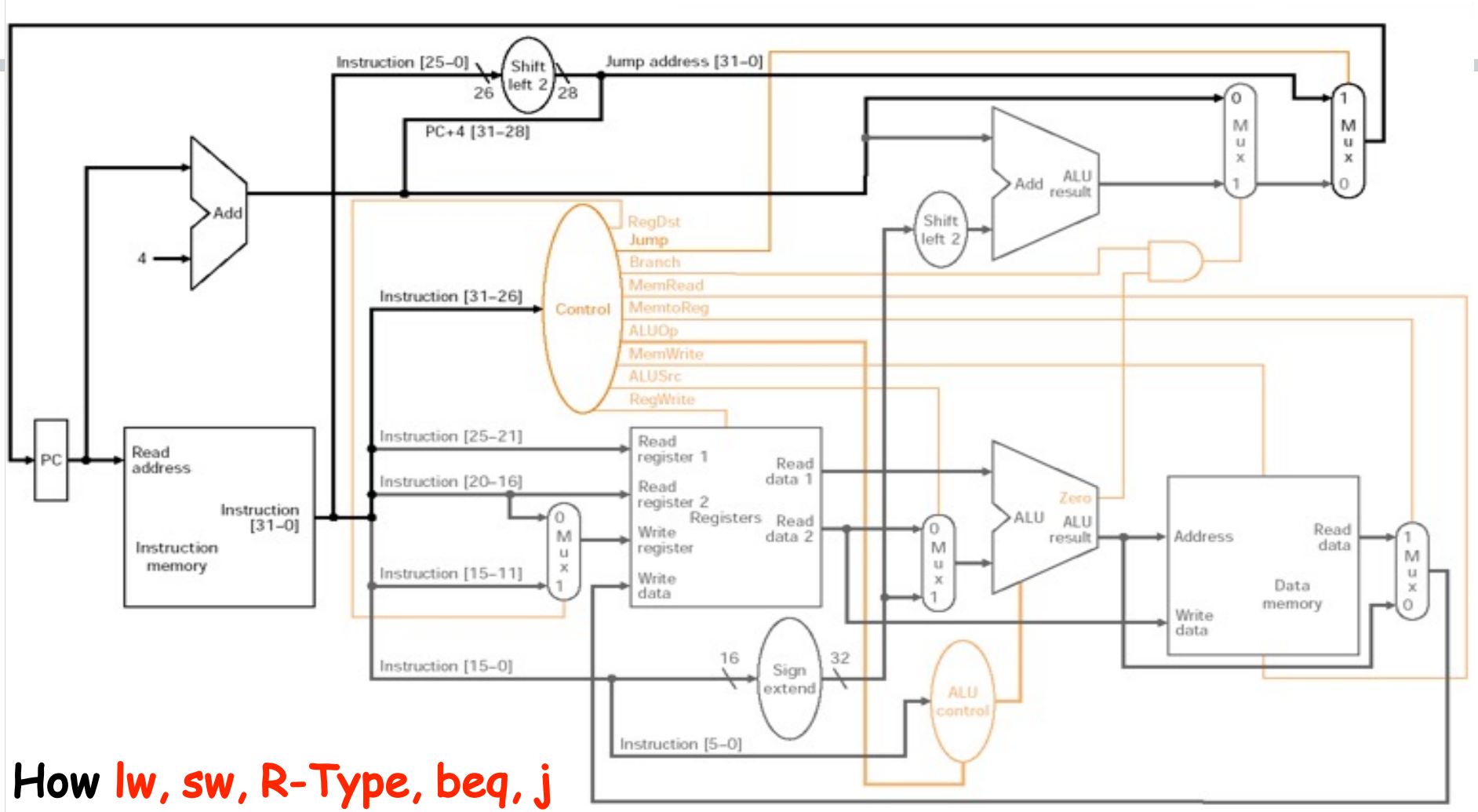


- Jump uses word address
- Need one more option (more datapath elements) to update PC as concatenation of
  - Top 4 bits of old PC
  - 26-bit jump address
  - 00
- Need an extra control signal decoded from opcode to choose which one to use

Datapath With Jumps Added

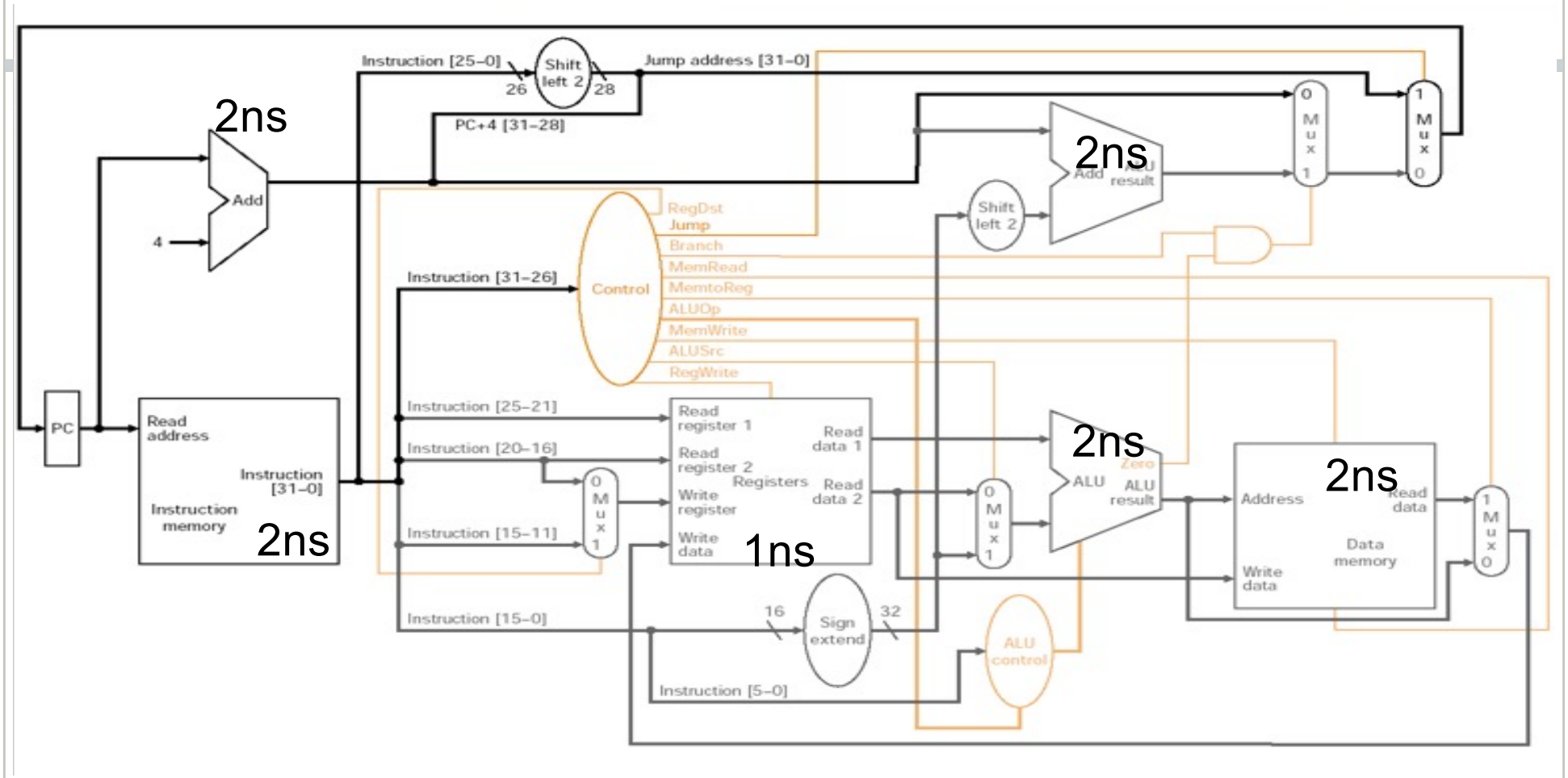


Complete Single Cycle Processor



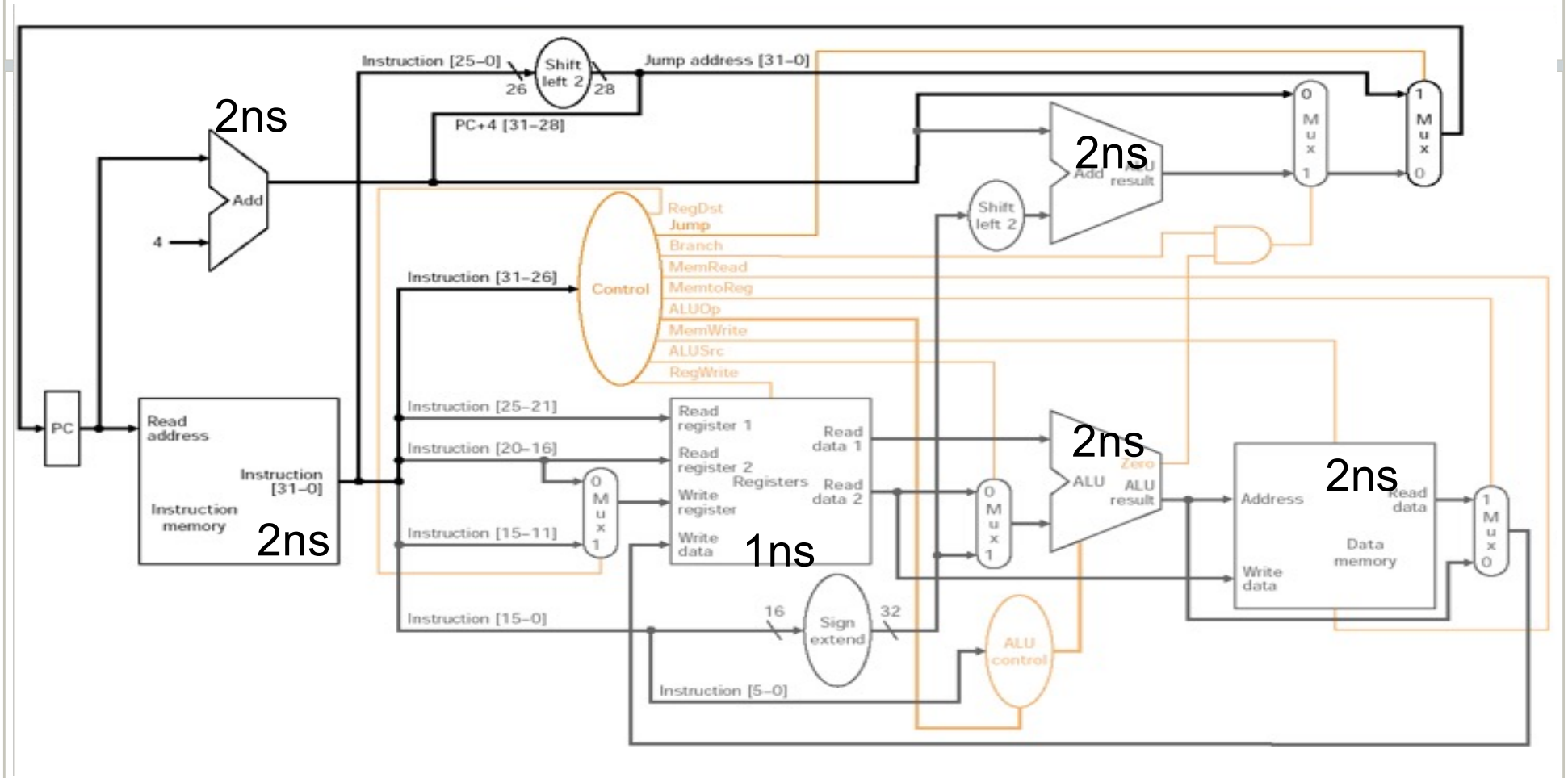
How lw, sw, R-Type, beq, j instructions work?

Delays in Single Cycle Datapath



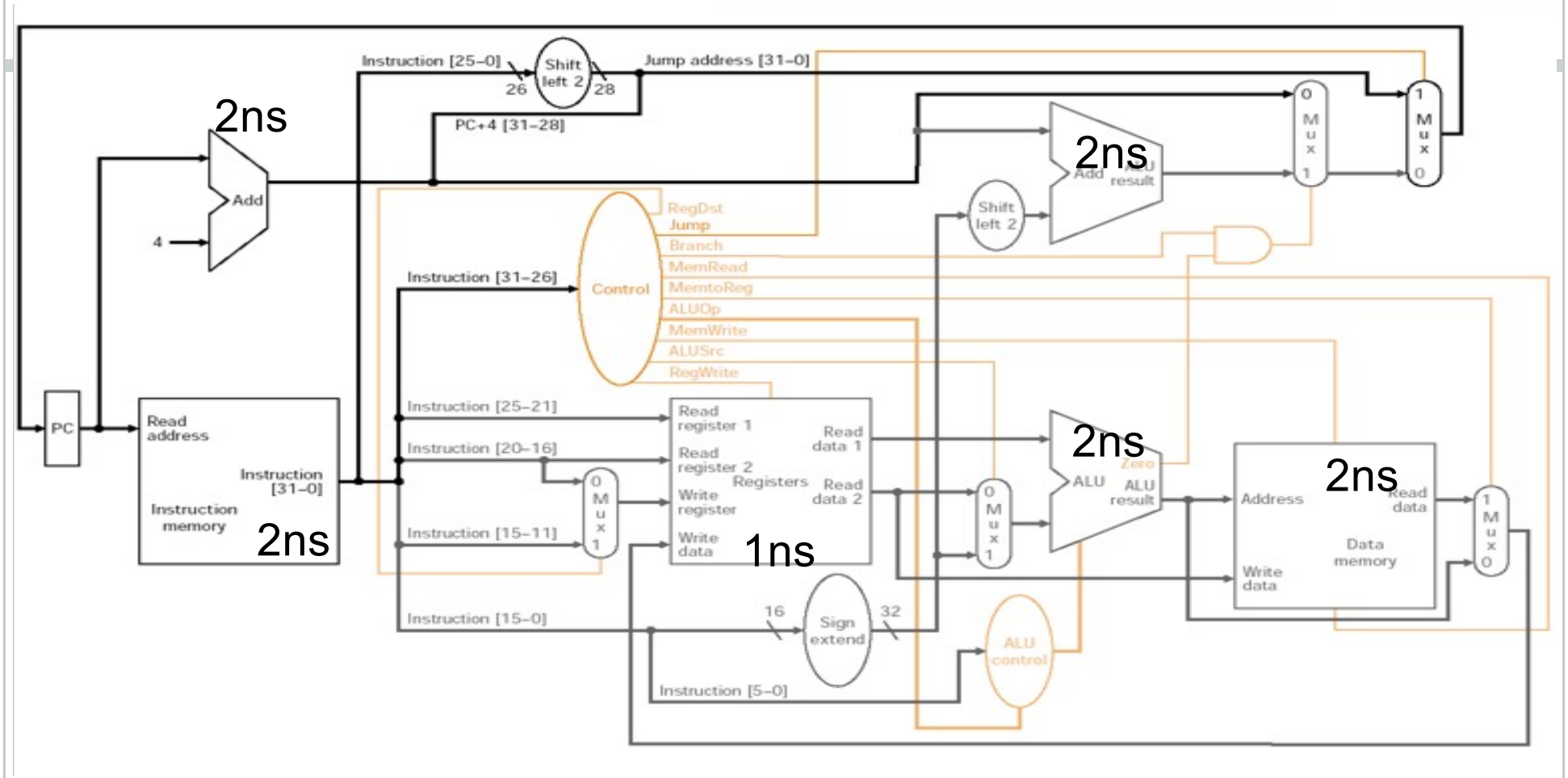
What are the delays for lw, sw, R-Type, beq, j instructions?

Delays in Single Cycle Datapath



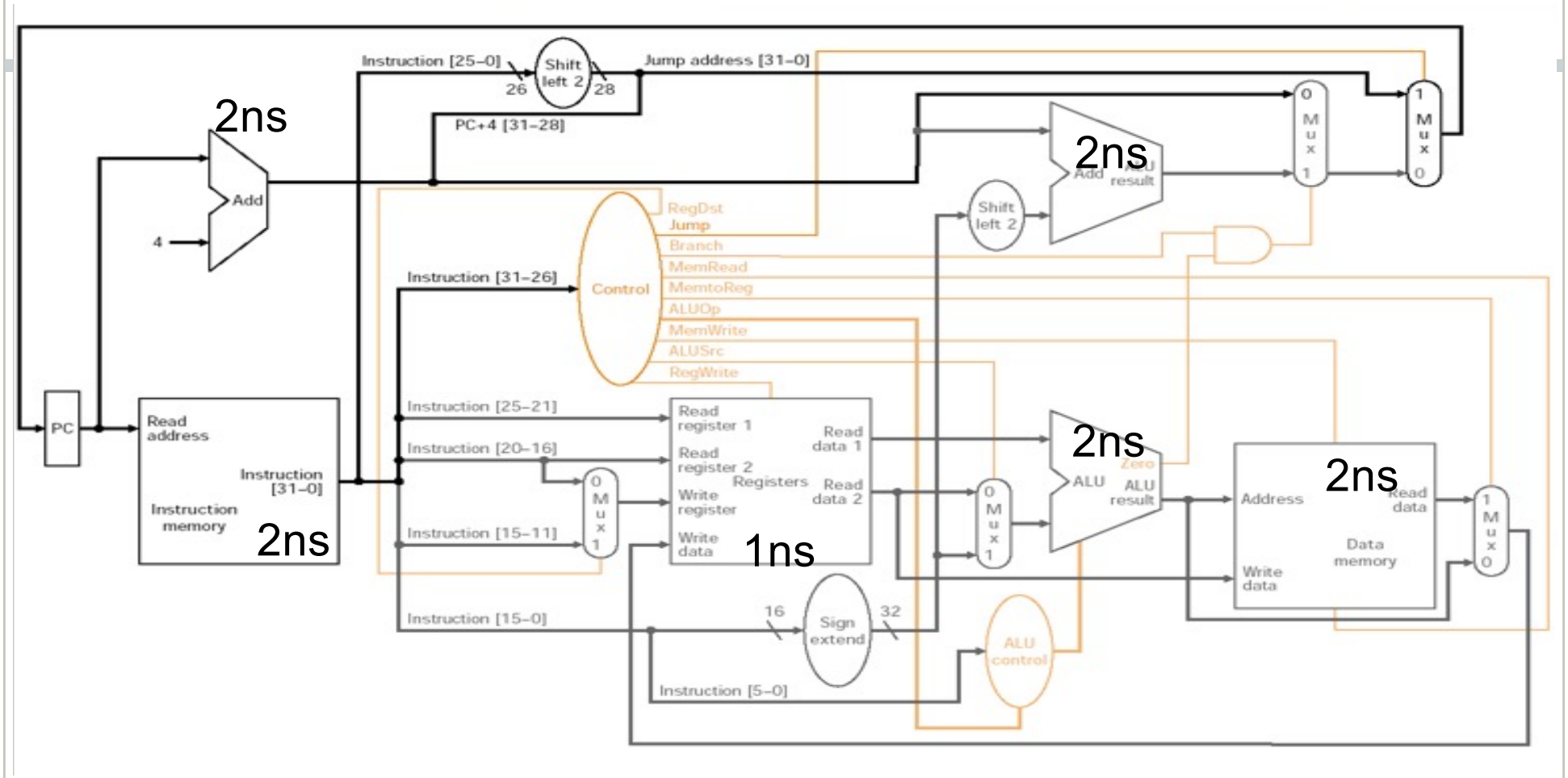
What is the delay for an R-Type instruction?

Delays in Single Cycle Datapath



What is the delay for a beq instruction?

Delays in Single Cycle Datapath



What is the delay for a lw instruction?

Delay in Single Cycle Datapath

- Calculate by assuming negligible delays except:
  - memory (2ns), ALU and adders (2ns), register file access (1ns)

Instruction class	Instruction Fetch (Memory)	Register Access	ALU	Memory Access	Register Access	
R-Type	X	X	X		X	6
Load	X	X	X	X	X	8
Store	X	X	X	X		7
Branch	X	X	X			5
Jump	X					2

Performance Issues

- Single cycle datapath => CPI=1, Clock Cycle Time => long
- Longest delay determines clock period
  - Critical path: load instruction
    - Instruction memory → register file → ALU → data memory → (register file)
- Violates design principle
  - Making the common case fast
- We will improve performance by pipelining

Summary

- 5 steps to design a processor
  - 1. Analyze instruction set => datapath requirements
  - 2. Select set of datapath components & establish clock methodology
  - 3. Assemble datapath meeting the requirements
  - 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer
  - 5. Assemble the control logic
- MIPS makes it easier
  - Instructions same size
  - Source registers always in same place
  - Immediates same size, location
  - Operations always on registers/immediates