CprE 381: Computer Organization and Assembly Level Programming

Processor Design + Proj Part 2

Henry Duwe

Electrical and Computer Engineering

Iowa State University

Administrative

- HW4 Due tonight at 11:59pm
  - Assignment viewable now?

- Term Project Part 1 Demo in Labs
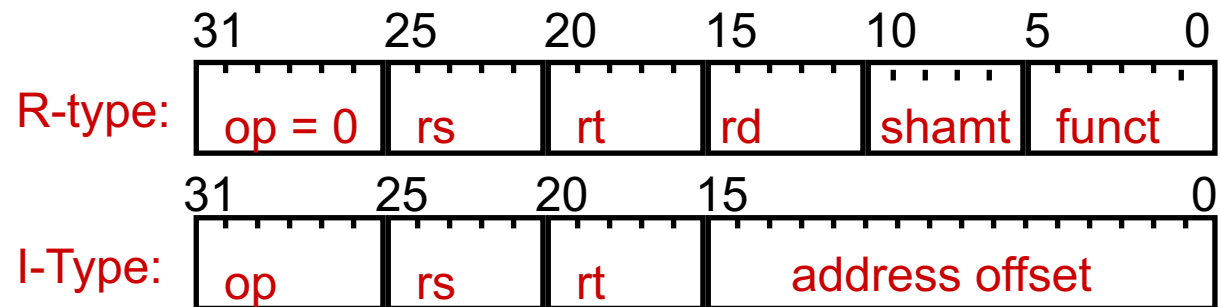
Administrative

- Academic Integrity Reminder
  - Term project must be entirely your groups work – all code, test programs (those not provided by instructors), reports, etc.
    - No sharing of VHDL code on term projects
    - No using past semesters VHDL
    - No using book added material VHDL
    - If caught, this will be escalated per ISU policy

# (Almost) Complete Single Cycle Processor
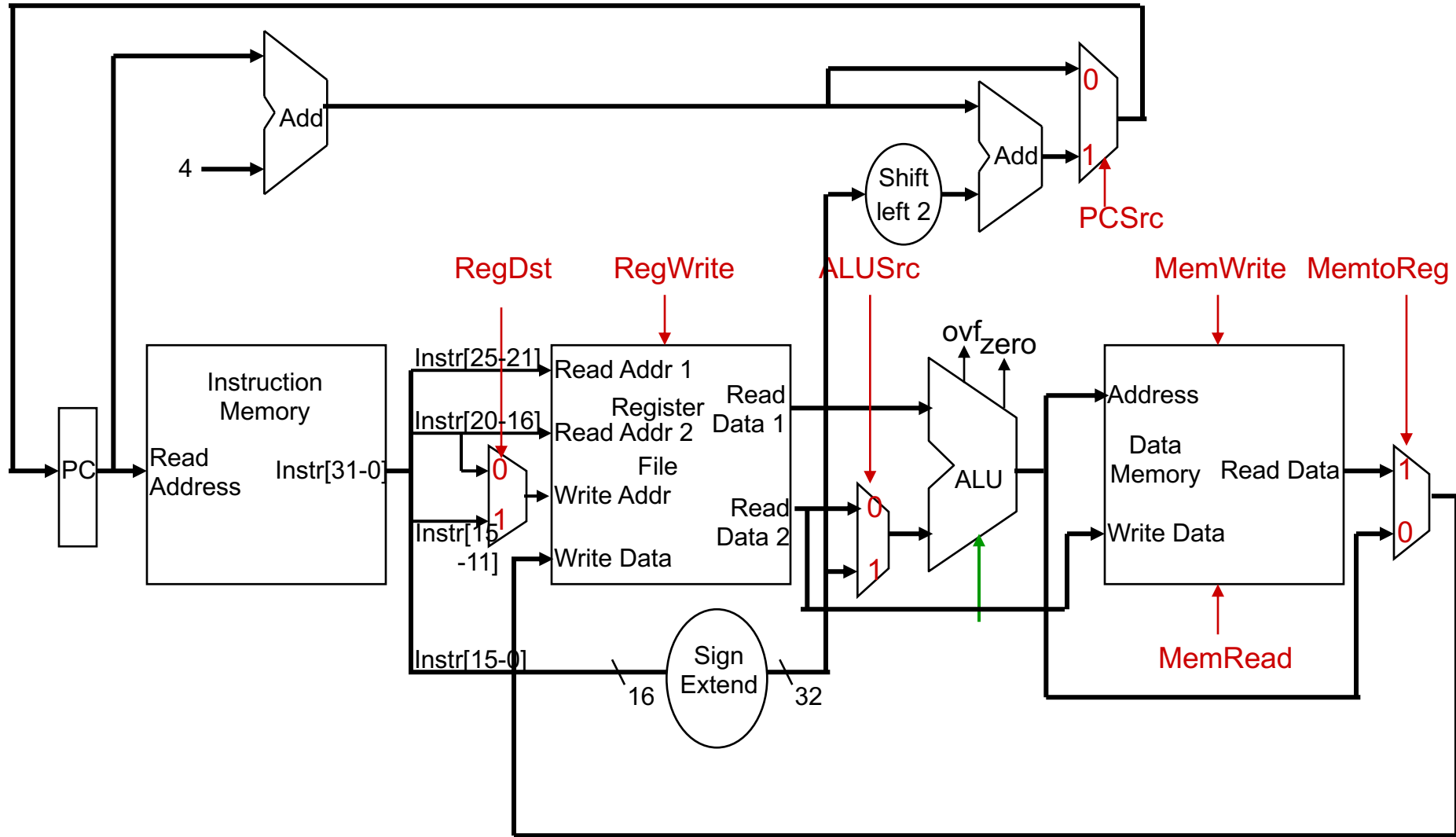
Adding the Control

- Selecting the operations to perform (ALU, Register File and Memory read/write)

- Controlling the flow of data (multiplexor inputs)

- Information comes from the 32 bits of the instruction

31	25	20	15	10	5	0

R-type:

op = 0	rs	rt	rd	shamt	funct

31	25	20	15	0

I-Type:

op	rs	rt	address offset

- Observations
  - op field always in bits 31-26
  - When op field is 0, funct field used from bits 5-0
  - Operand locations regular – rs always 25-20, rt always 20-15, etc.

(Almost) Complete Single Cycle Processor
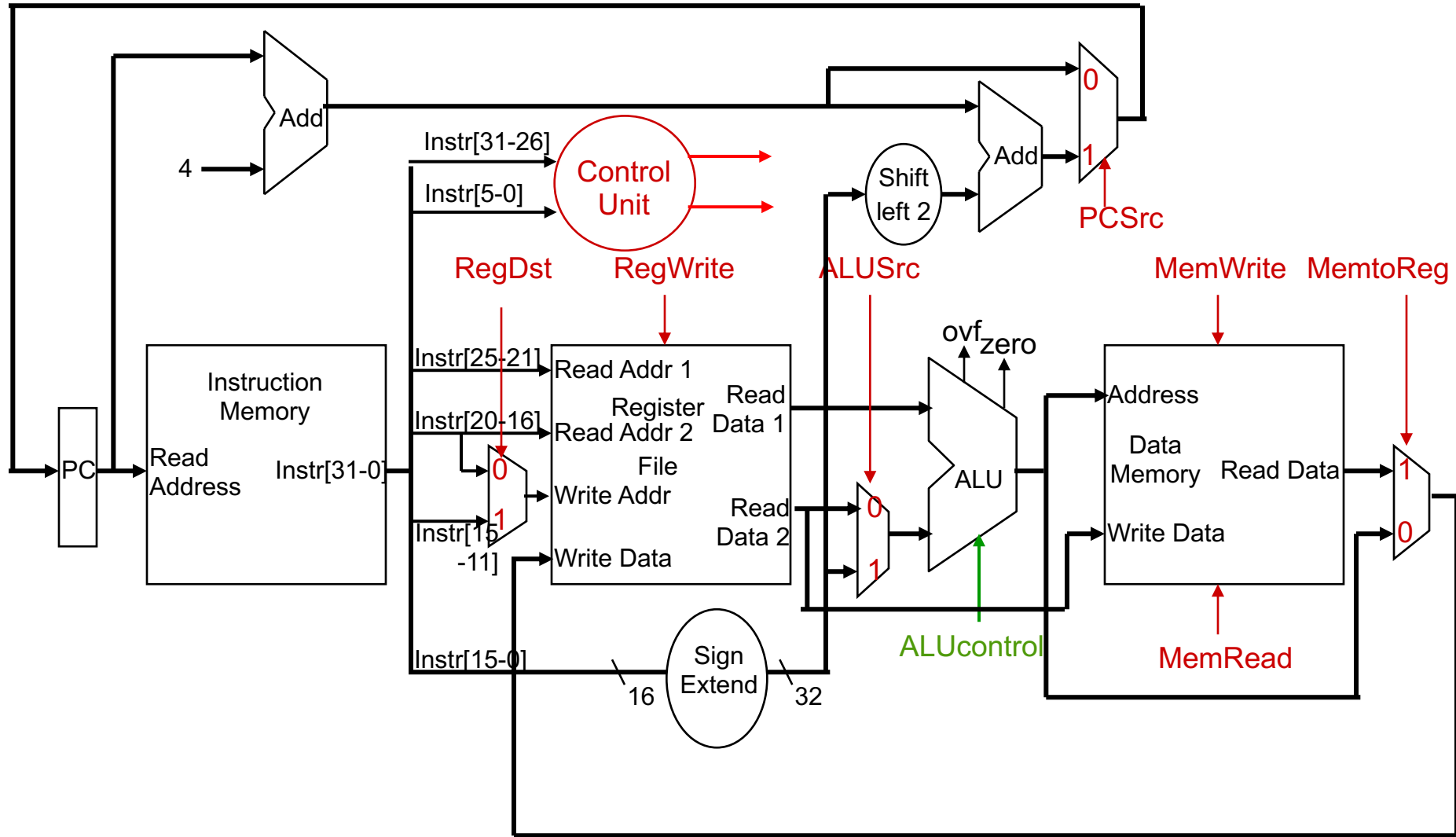
ALU Control

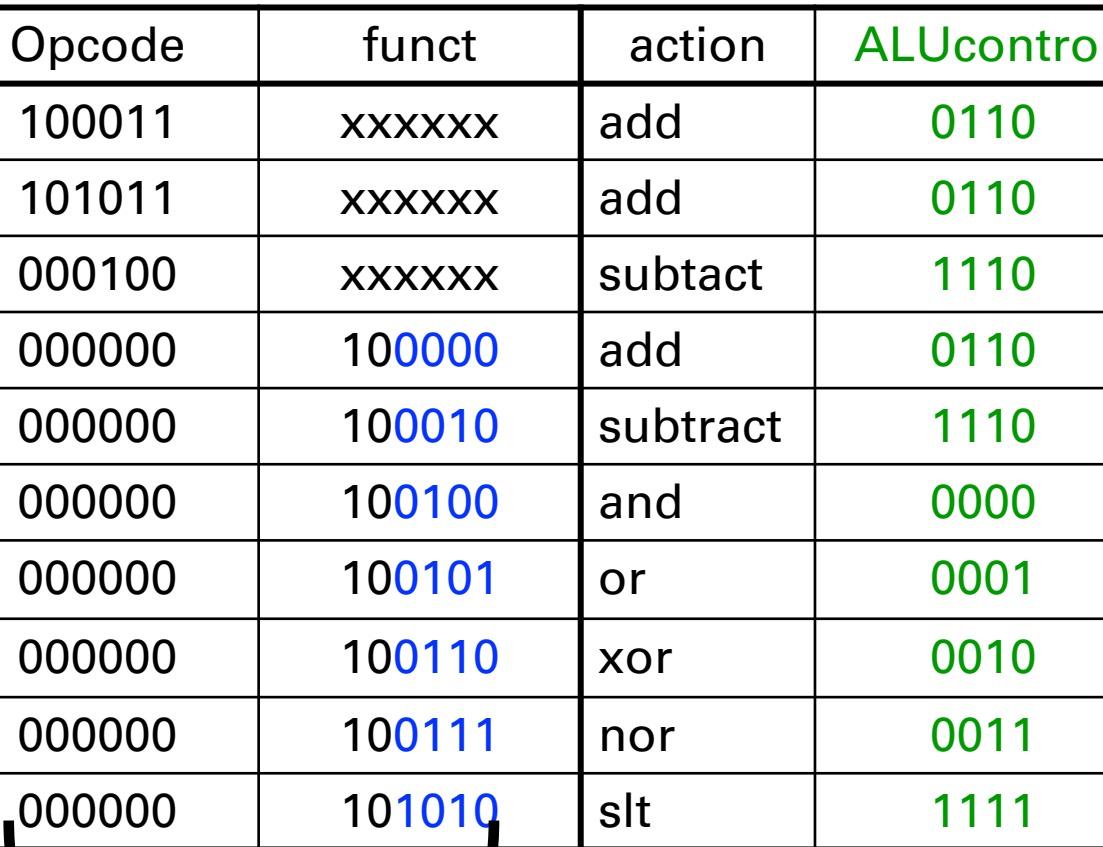- ALU's operation based on instruction type and function code:

ALU control input	Function
0000	and
0001	or
0010	xor
0011	nor
0110	add
1110	subtract
1111	set on less than

- Notice that we are using different encodings than in the book (and different than you have chosen for your project)

ALU Control (cont.)

- So describe with a truth table

Four truth tables

| Instr op	Opcode	funct	action	ALUcontrol
lw	100011	xxxxxx	add	0110
sw	101011	xxxxxx	add	0110
beq	000100	xxxxxx	subtact	1110
add	000000	100000	add	0110
subt	000000	100010	subtract	1110
and	000000	100100	and	0000
or	000000	100101	or	0001
xor	000000	100110	xor	0010
nor	000000	100111	nor	0011
slt	000000	101010	slt	1111

12 inputs (use don't cares: '-' in VHDL)

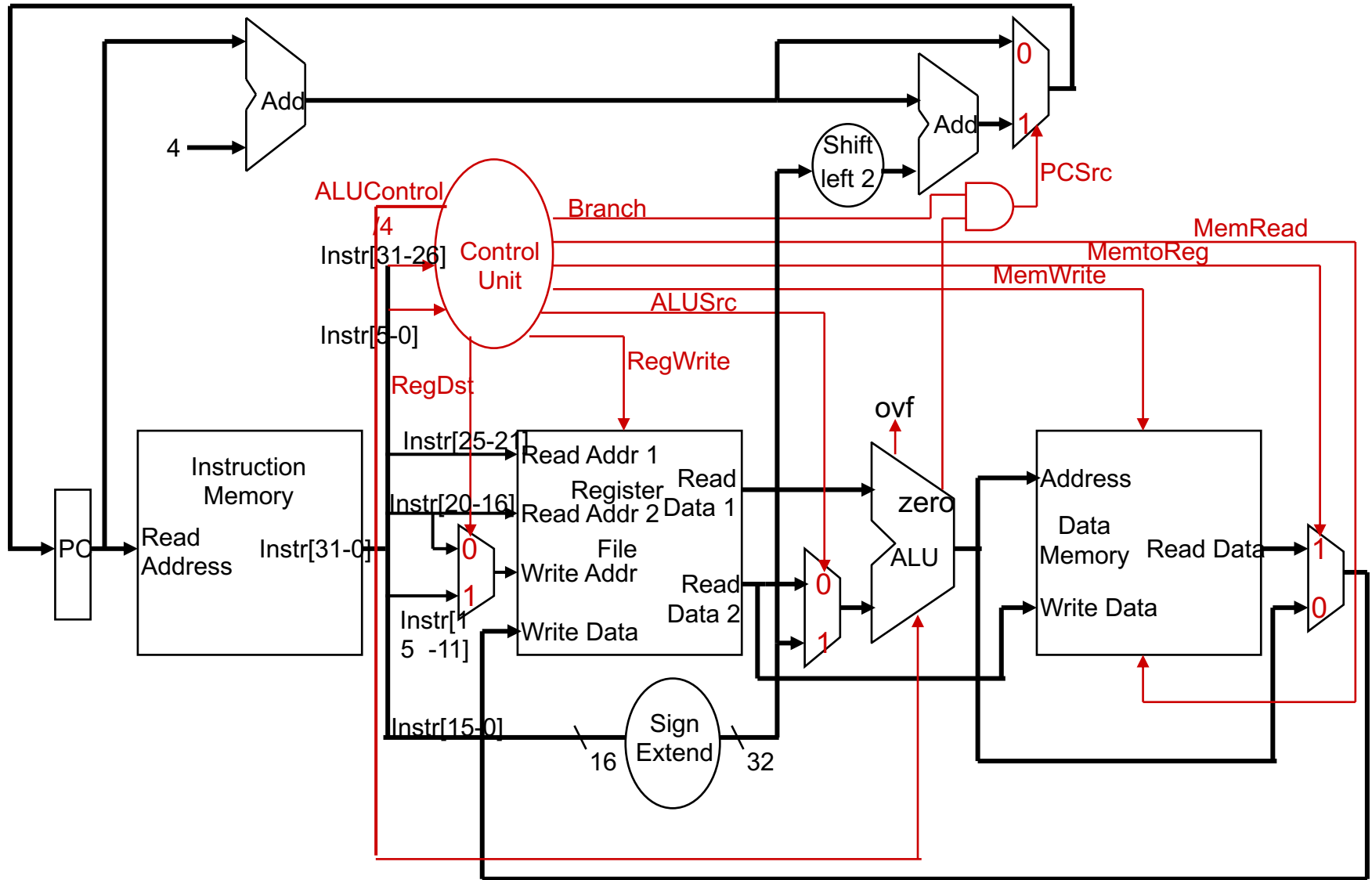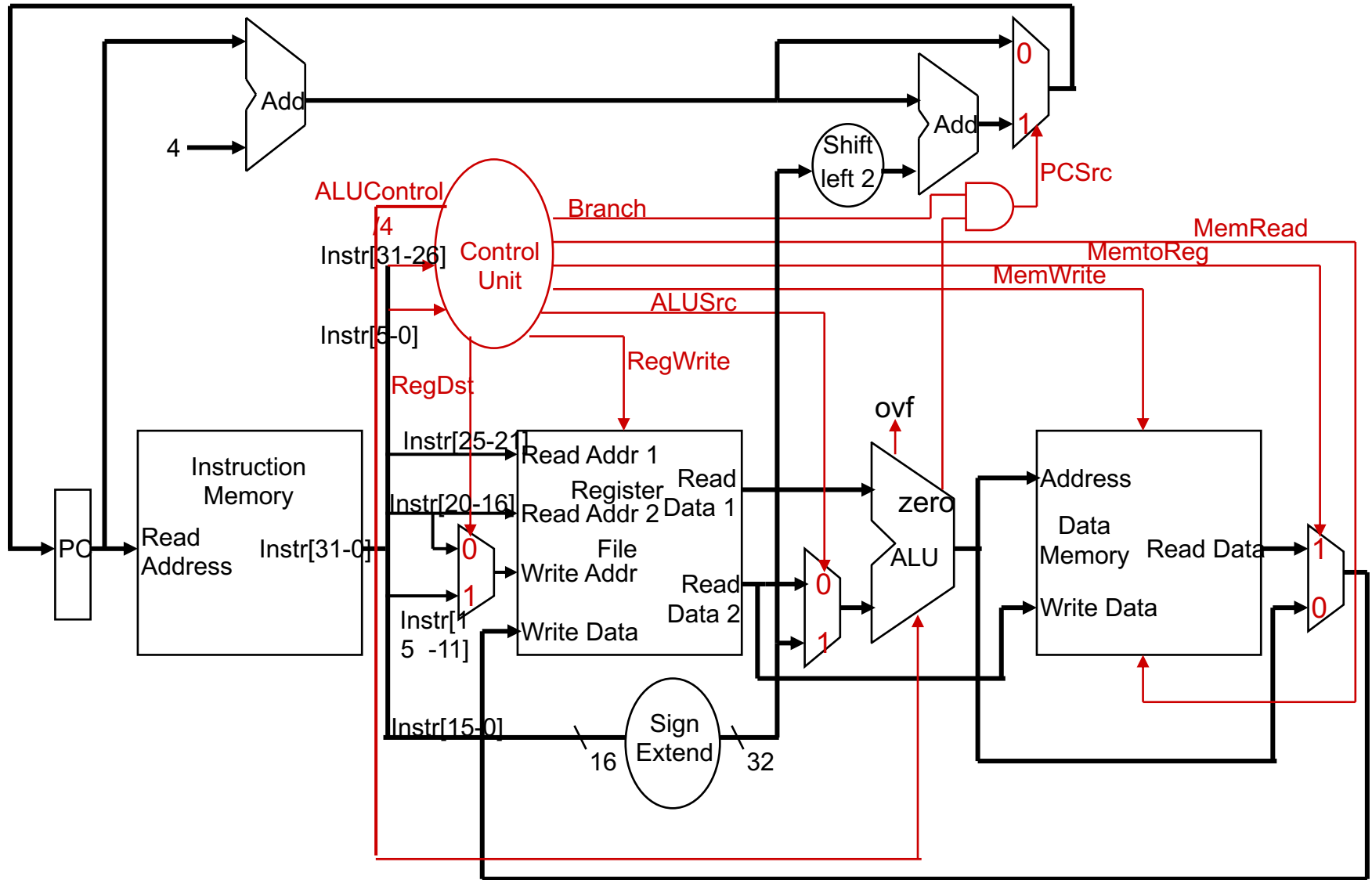# (Almost) Complete Datapath w/ Control
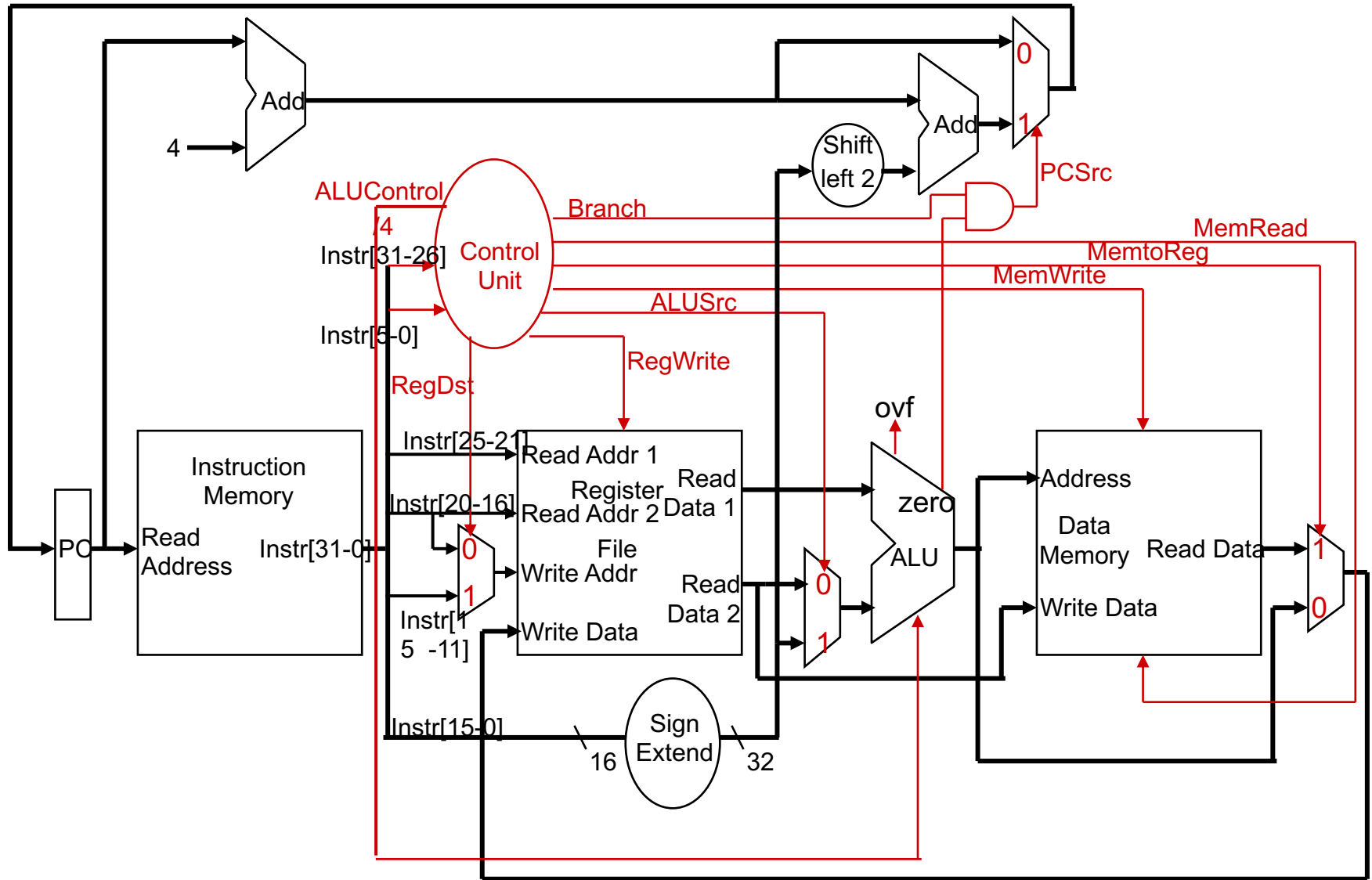
Main Control Unit

Instr	ALU control	RegDst	ALUSrc	MemReg	RegWr	MemRd	MemWr	Branch
R-type 000000								
lw 100011								
sw 101011								
beq 000100								

- Note that a multiplexor whose control input is 0 has a definite action, even if it is not used in performing the operation
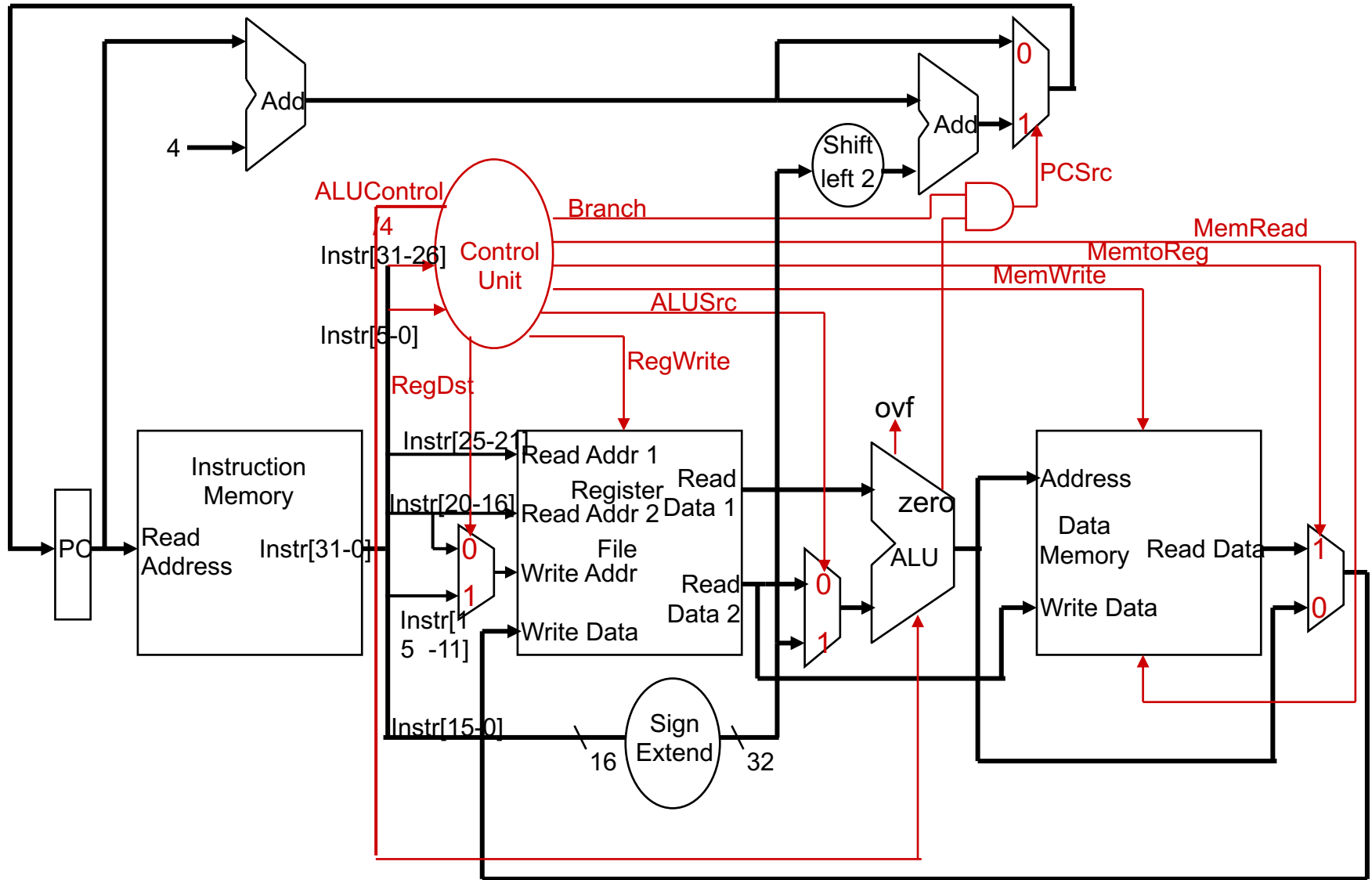
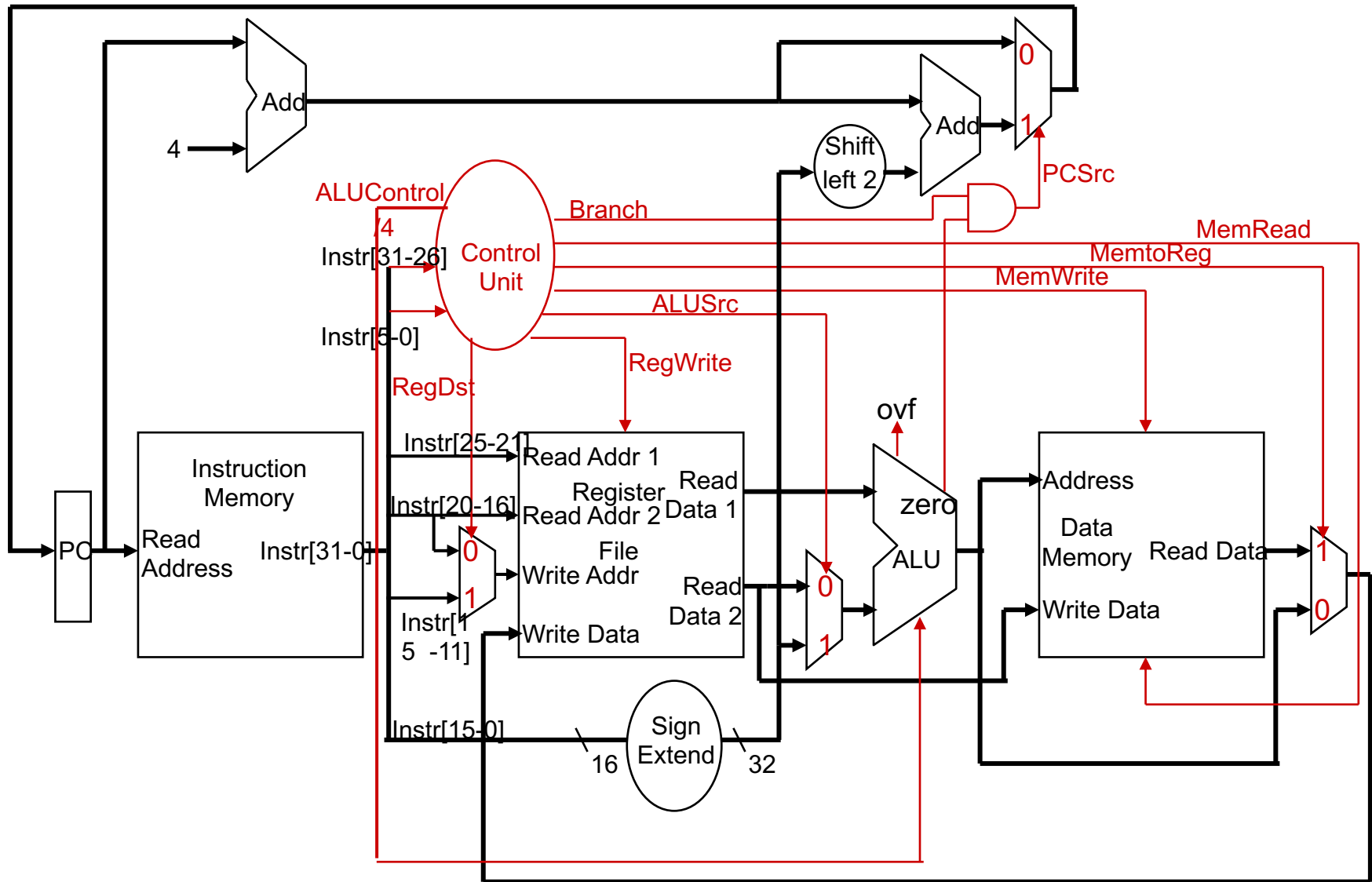# R-Type Instruction Data/Control Flow

lw Instruction Data/Control Flow

sw Instruction Data/Control Flow

beq Instruction Data/Control Flow

Main Control Unit (cont.)

Instr	ALU control	RegDst	ALUSrc	MemReg	RegWr	MemRd	MemWr	Branch
R-type 000000	Depends on Funct	1	0	0	1	0	0	0
lw 100011	0110	0	1	1	1	1	0	0
sw 101011	0110	X	1	X	0	0	1	0
beq 000100	1110	X	0	X	0	0	0	1

- Setting of the MemRd signal (for R-type, sw, beq) depends on the memory design (could have to be 0 or could be a X (don't care))

Control Unit Logic

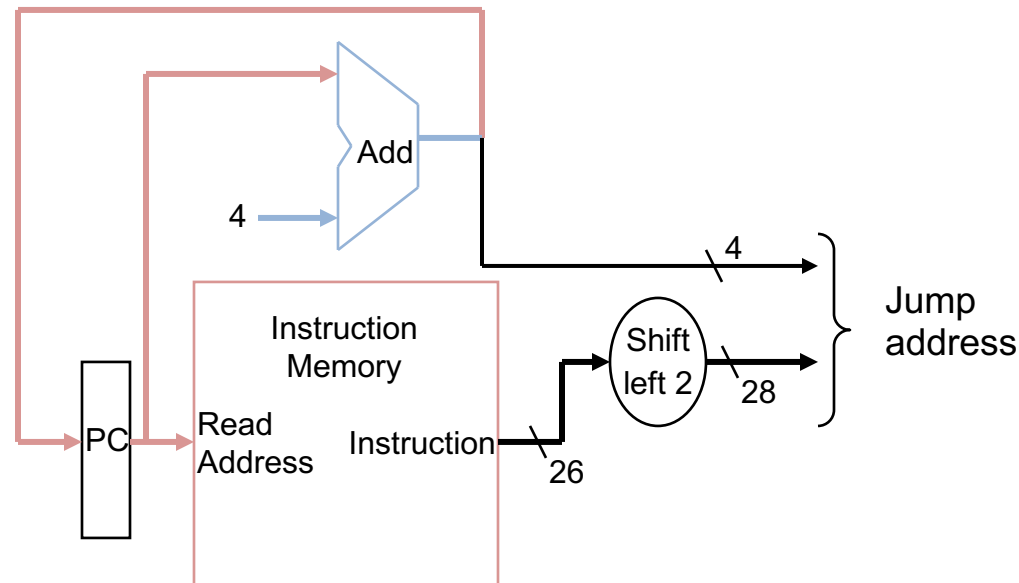- From the truth table can design the Main Control logic
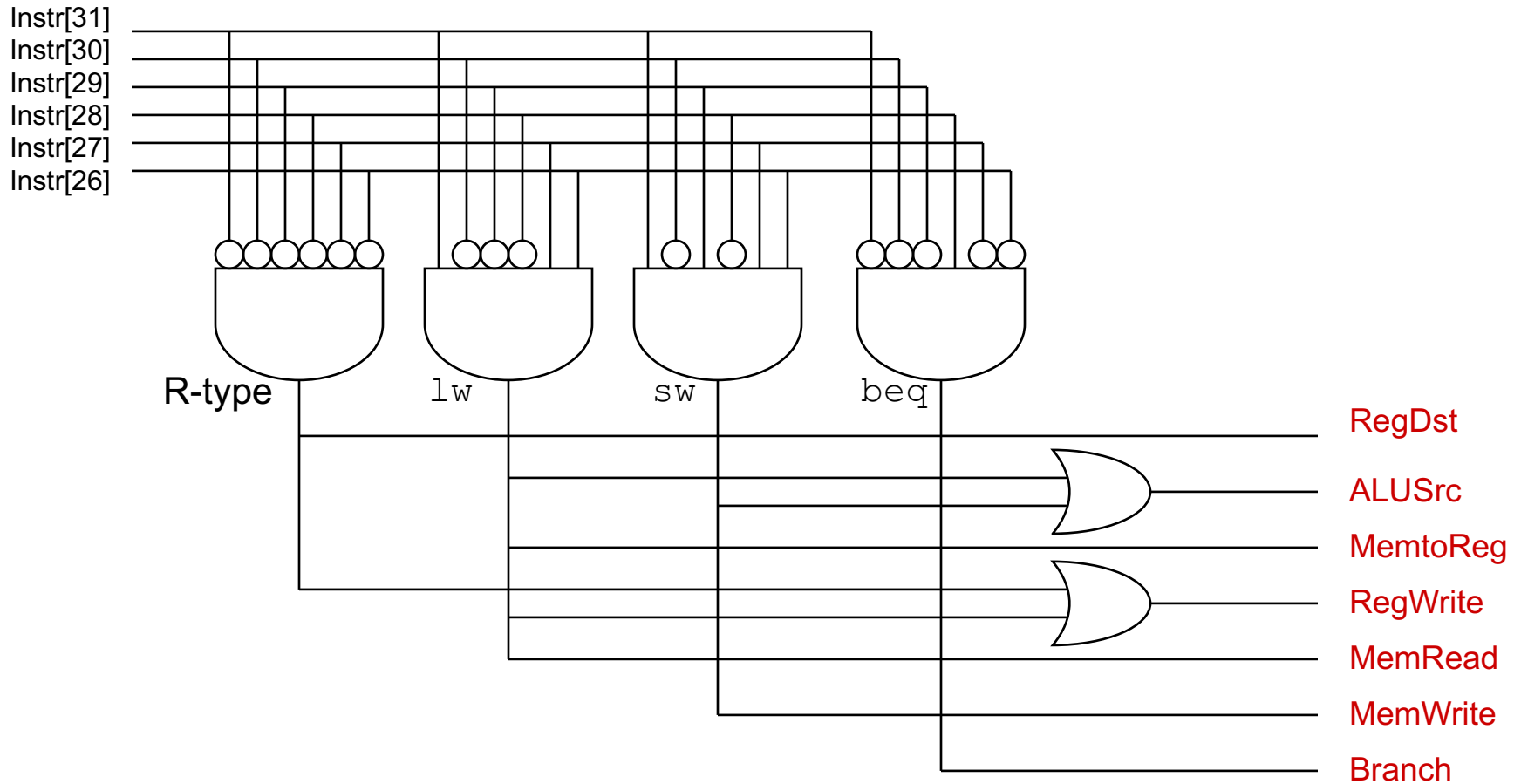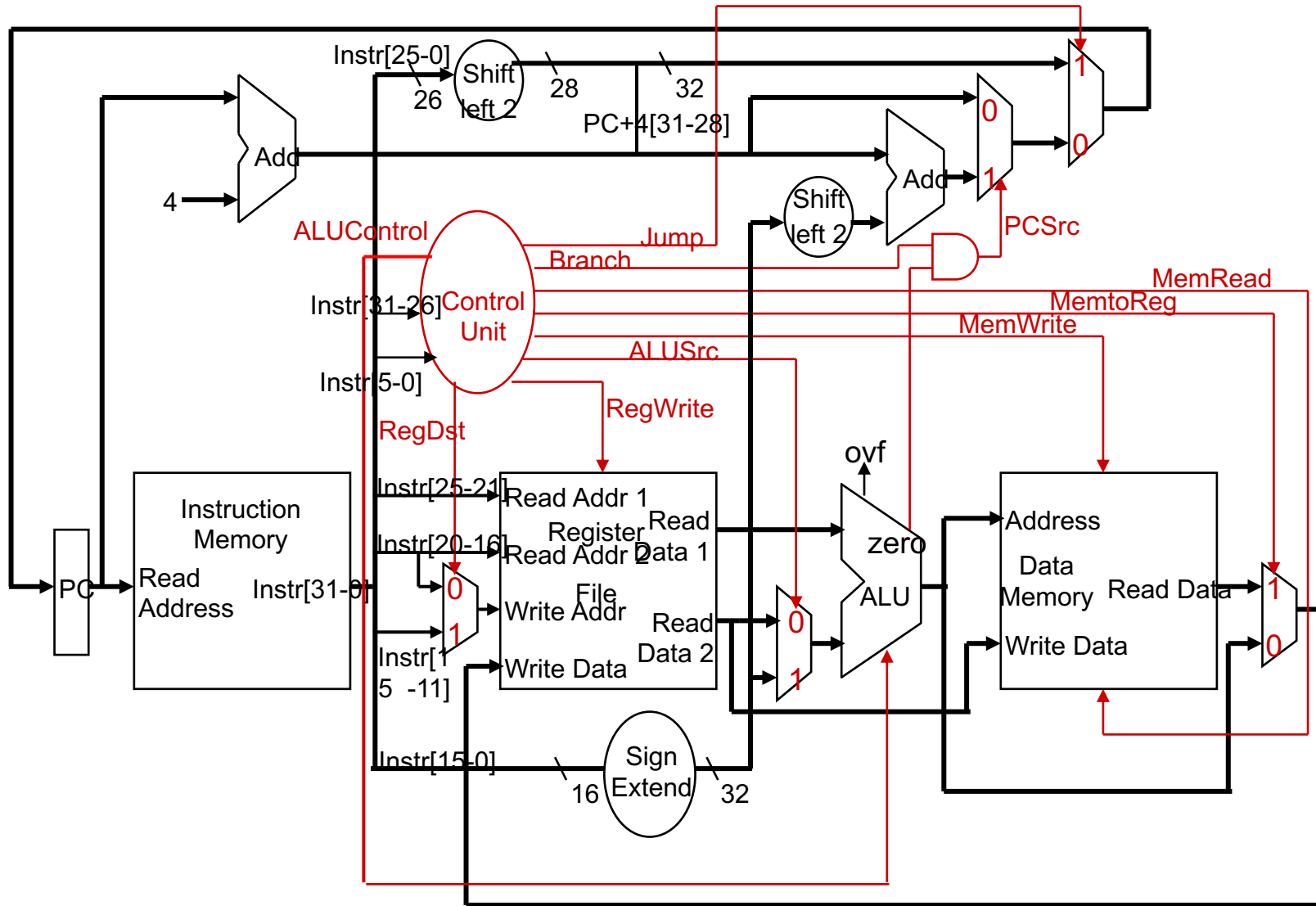
Review: Executing Jump Operations
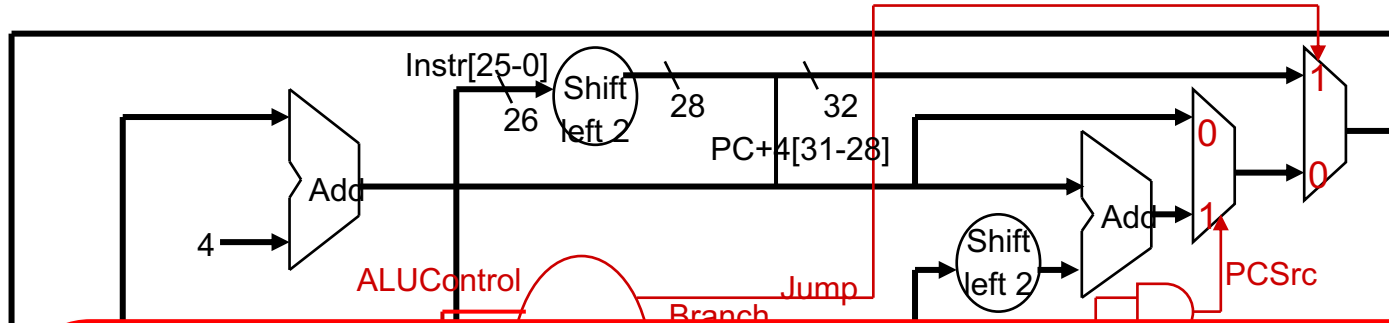
- Jump operations have to

J-Type:

31	25	0
op	jump target address	

- Replace the lower 28 bits of the PC with the lower 26 bits of the fetched instruction shifted left by 2 bits

Add

4

4

Jump address

Instruction Memory

Shift left 2

28

PC

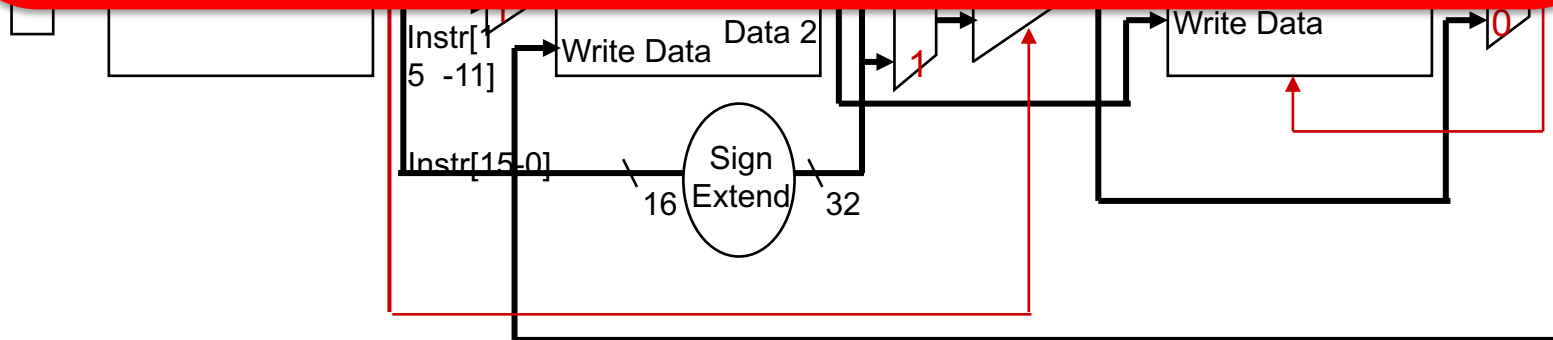Read Address

Instruction

26

Adding the Jump Instruction

Adding the Jump Instruction

In-class Assessment!
Access Code: Hop

Note: sharing access code to those outside of classroom or using access while outside of classroom is considered cheating

Main Control Unit (cont.)

| Instr	ALU control	RegDst	ALUSrc	MemReg	RegWr	MemRd	MemWr	Branch	Jump
R-type 000000	Depends on Funct	1	0	0	1	0	0	0	0
lw 100011	0110	0	1	1	1	1	0	0	0
sw 101011	0110	X	1	X	0	0	1	0	0
beq 000100	1110	X	0	X	0	0	0	1	0
j 000010	X	X	X	X	0	0	0	X	1

- Setting of the MemRd signal (for R-type, `sw`, `beq`) depends on the memory design

Model Truth Table in VHDL

Use selected signal statement (data flow)

```
-- input : 5-bit addr; output: 32-bit sel
  with addr select
     sel <= x"00000001" when b"00000",
            x"00000002" when b"00001",
            … -- more cases
            x"80000000" when b"11111";
```
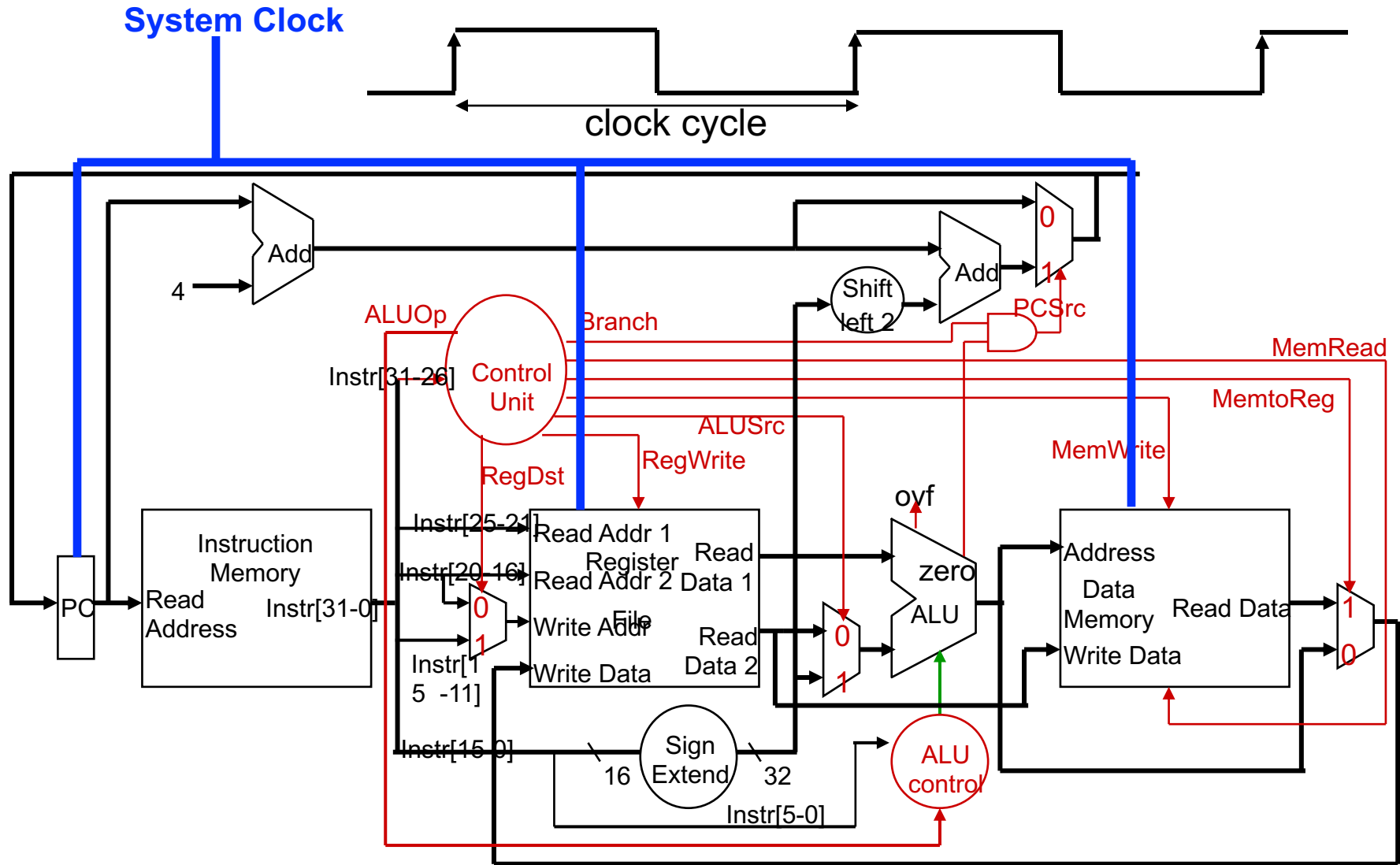
Clock Distribution
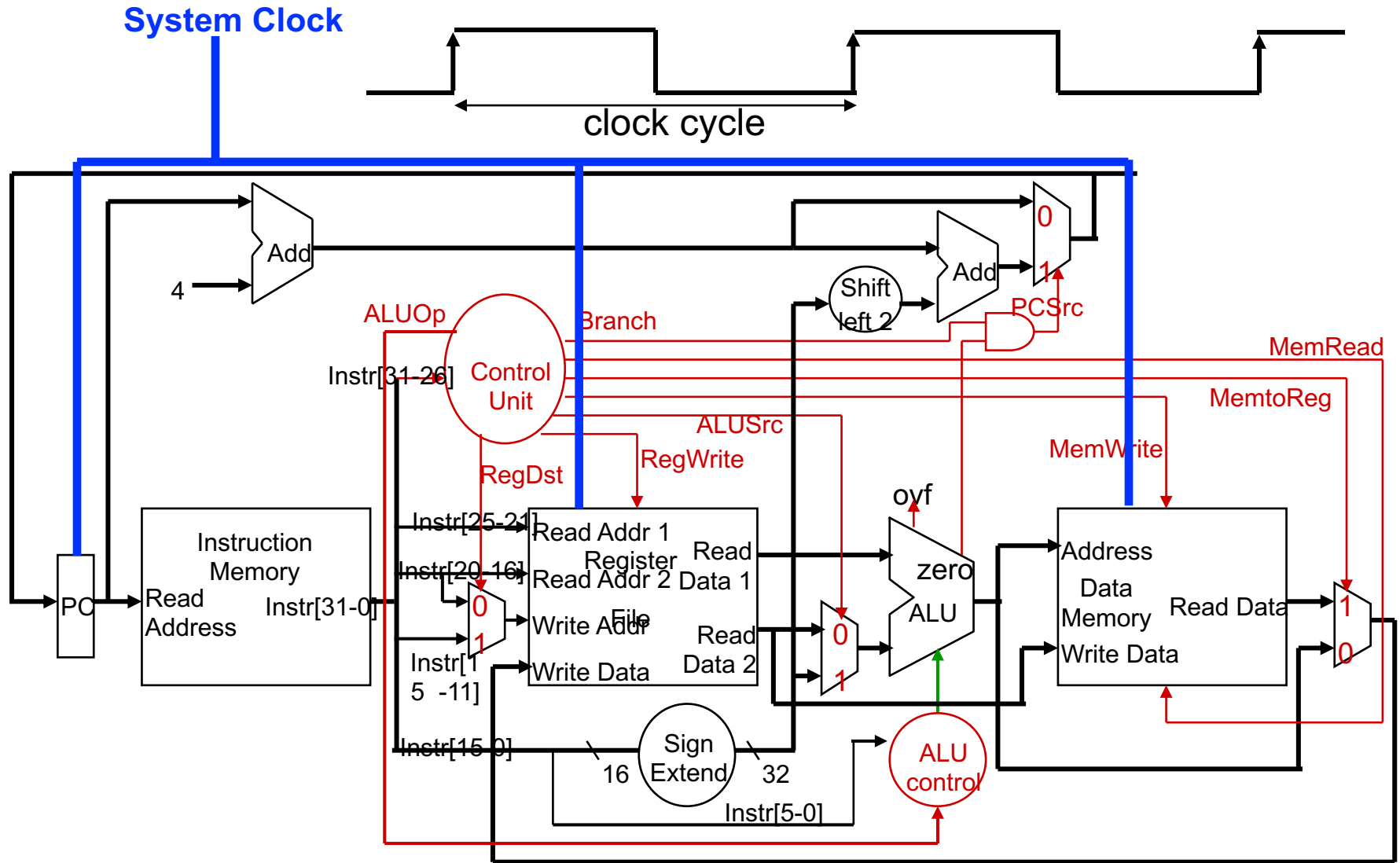
Operation

- ## We wait for everything to settle down

    - ### ALU might not produce "final answer" right away

    - ### Memory and RegFile reads are combinational (as are ALU, adders, muxes, shifter, signextender)

    - ### Use write signals along with the clock edge to determine when to write to the sequential elements (to the PC, to the Register File and to the Data Memory)

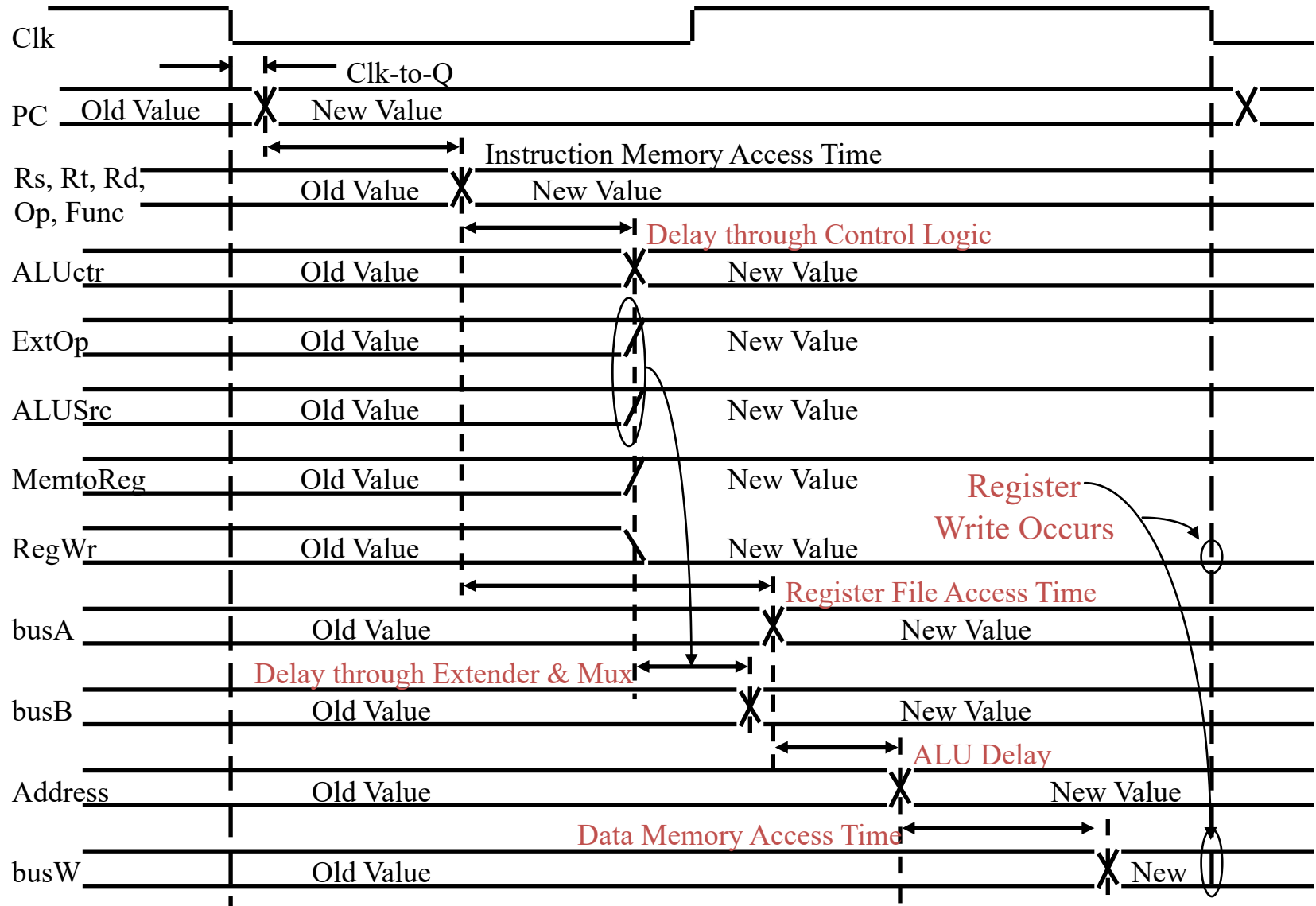- ## The clock cycle time is determined by the logic delay through the longest path

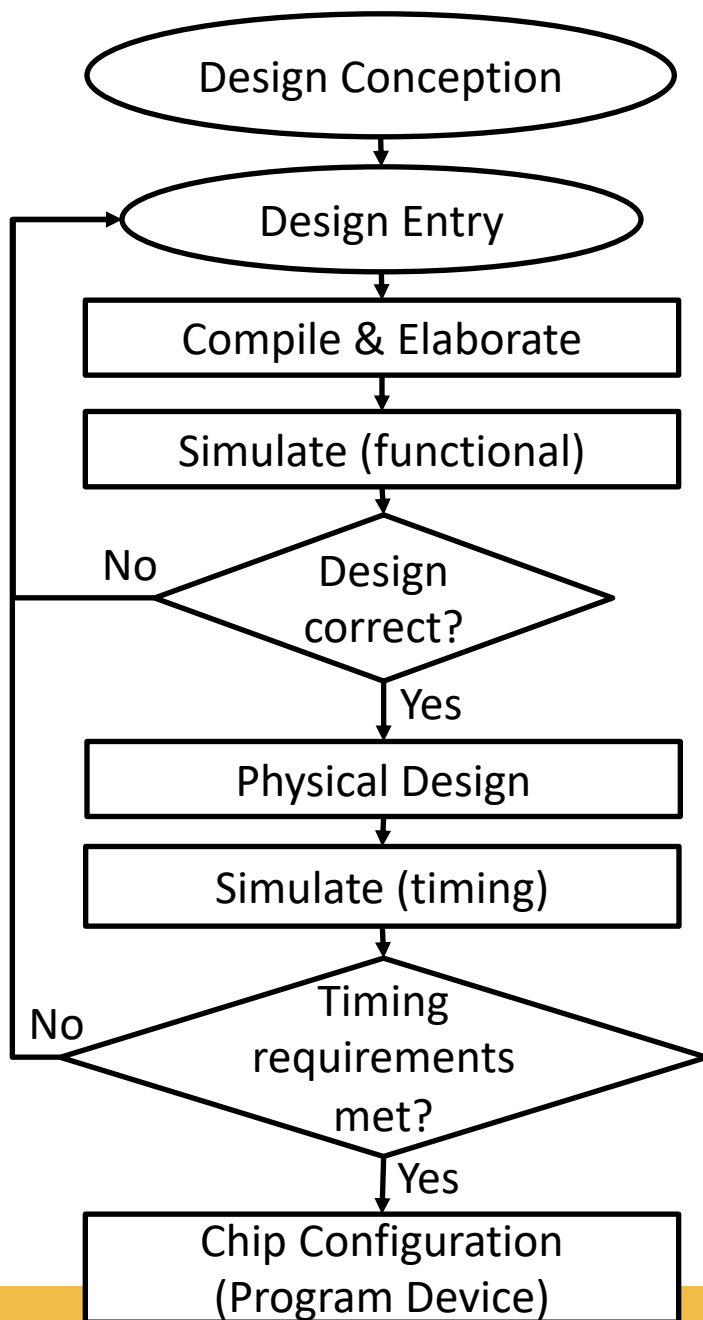### We are ignoring some details like register setup and hold times
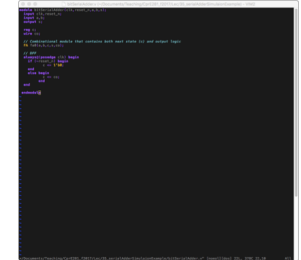
Worst Case Timing (Load Instruction)

System Clock

clock cycle

Add

4

ALUOp

Control Unit

Instr[31-26]

Branch

Shift left 2

Add

PCSrc

MemRead

ALUSrc

RegWrite

MemtoReg

RegDst

MemWrite

ovf

PC

Instruction Memory

Read Address

Instr[31-0]

Instr[25-21]

Instr[20-16]

Instr[15-11]

Instr[15-0]

Read Addr 1

Read Addr 2

Write Addr

Write Data

Register File

Read Data 1

Read Data 2

zero

ALU

Address

Data Memory

Write Data

Read Data

Sign Extend

16   32

Instr[5-0]

ALU control

Worst Case Timing (Load Instruction)

Tools: **ASMs**

HDLs
(VHDL)

ModelSim

QUARTUS

ModelSim

QUARTUS
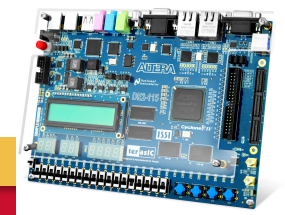
[MODIFIED Figure 2.35 from the 281 textbook]

Automated Testing Framework
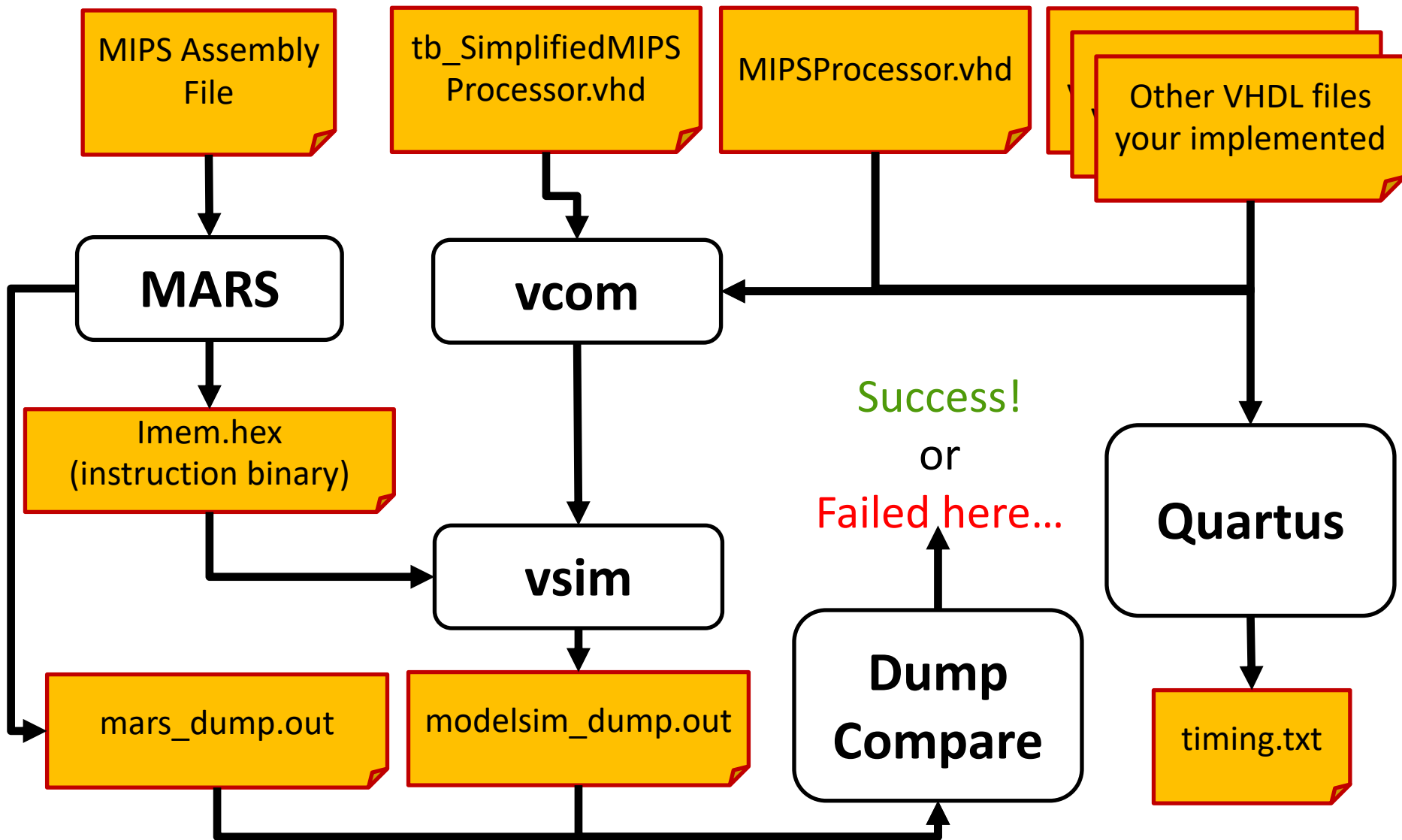
Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)