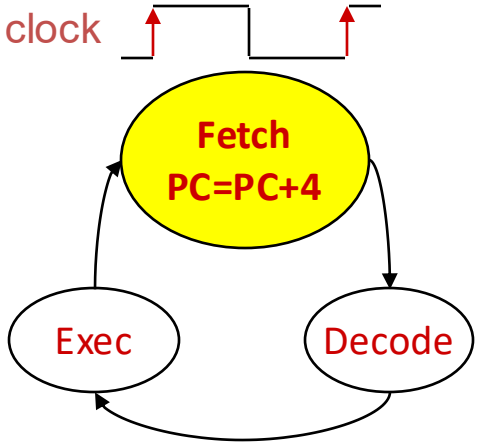
CprE 3810: Computer Organization and Assembly Level Programming

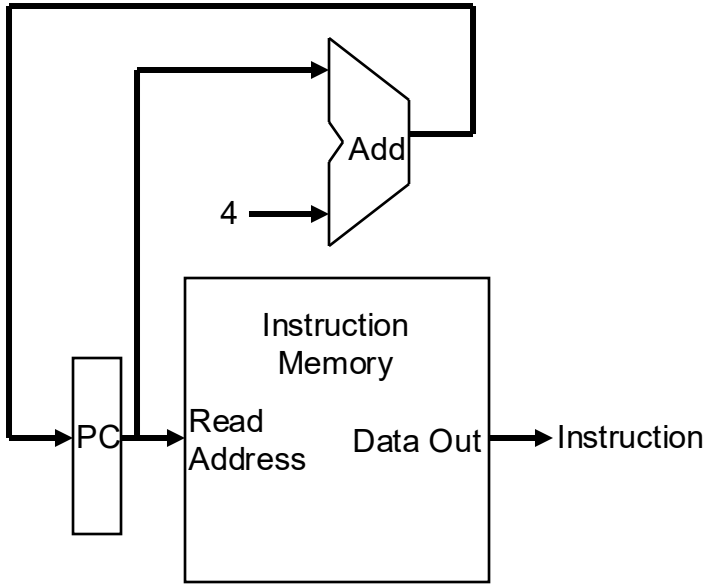
Processor Design

Henry Duwe
Electrical and Computer Engineering
Iowa State University

Fetching Instructions

- Fetching instructions involves
  - reading the instruction from the Instruction Memory
  - updating the PC value to be the address of the next (sequential) instruction

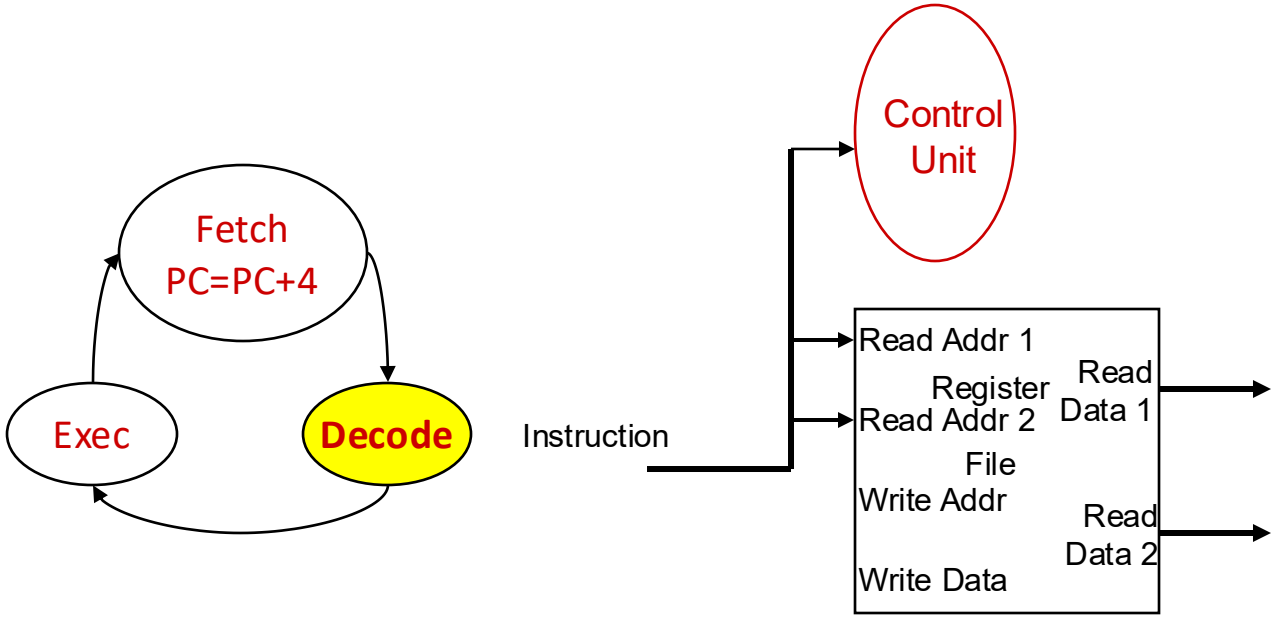




  - PC is updated every clock cycle, so it does not need an explicit write control signal – just a clock signal
  - Reading from the Instruction Memory is a combinational activity

Decoding Instructions

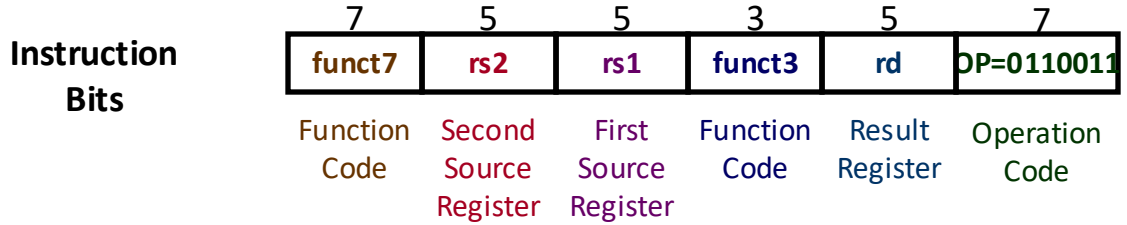
- Decoding instructions involves
  - Sending the fetched instruction's opcode and function field bits to the control unit



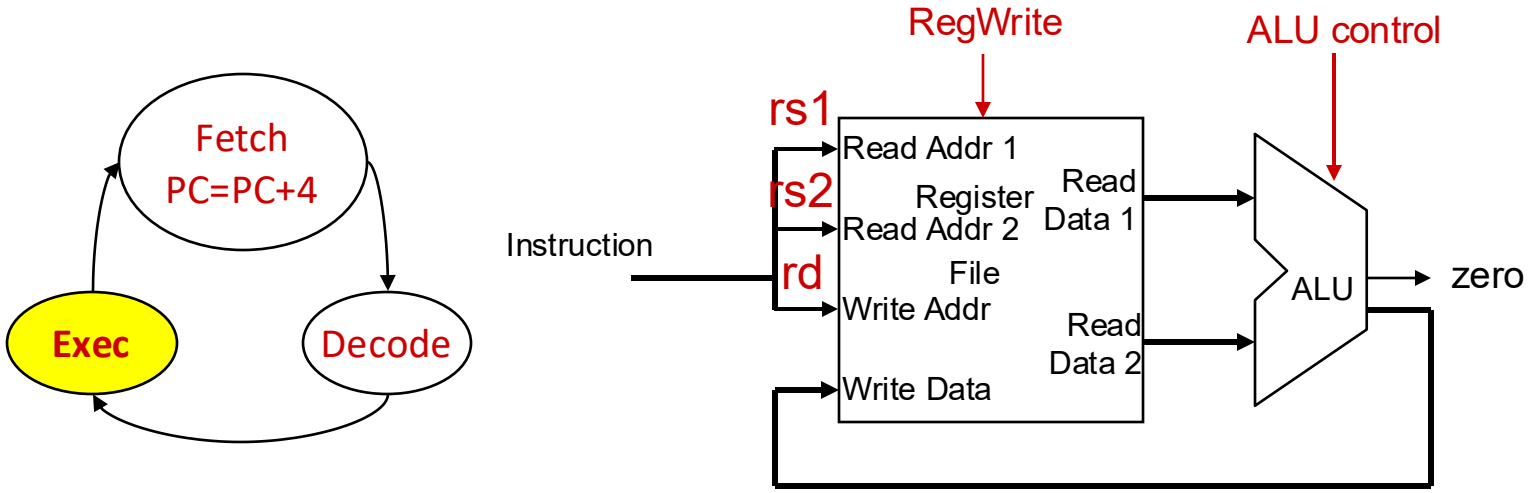
  - Reading two values from the Register File
    - Register File addresses are contained in the instruction

Executing R Format Operations

- R format operations (**`add, sub, slt, and, or`**)

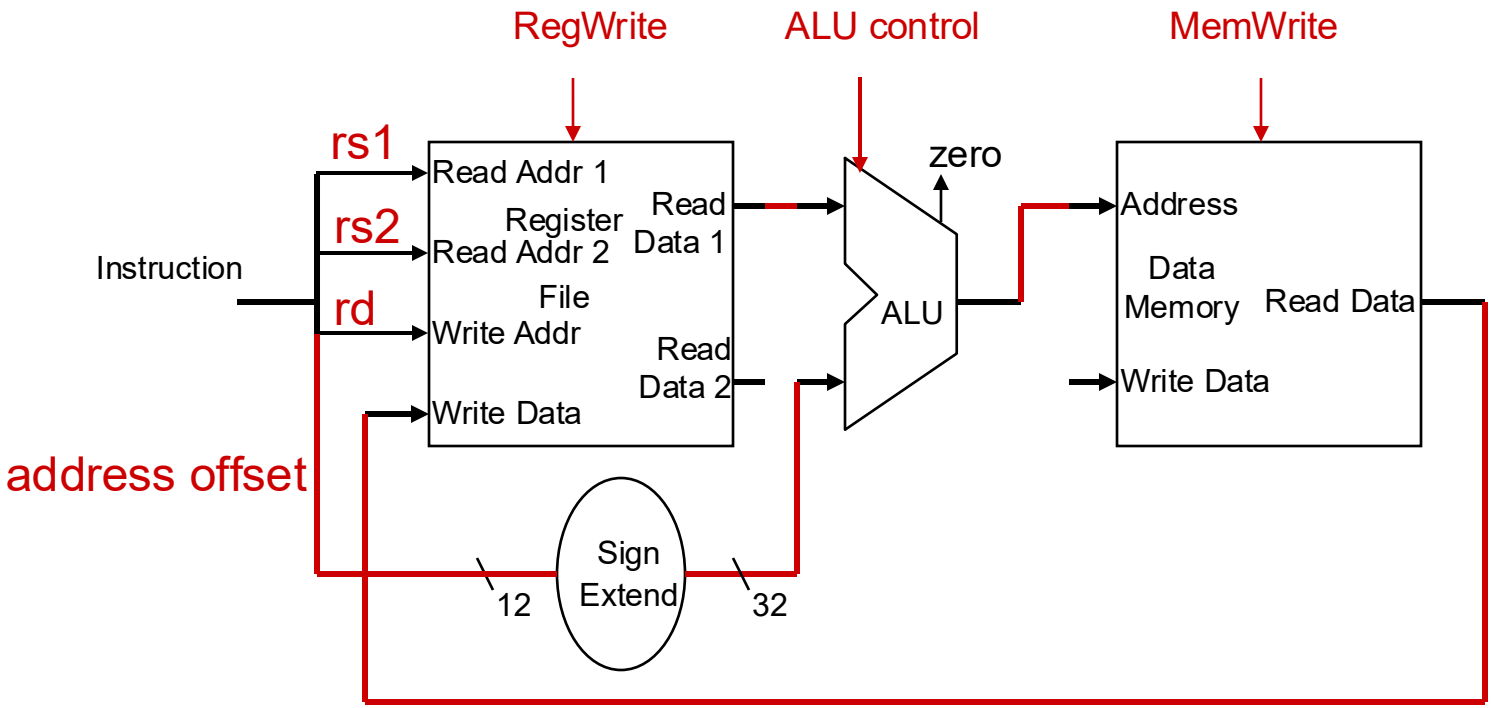


  - Perform operation (op and funct) on values in rs1 and rs2
  - Store the result back into the Register File (into location rd)

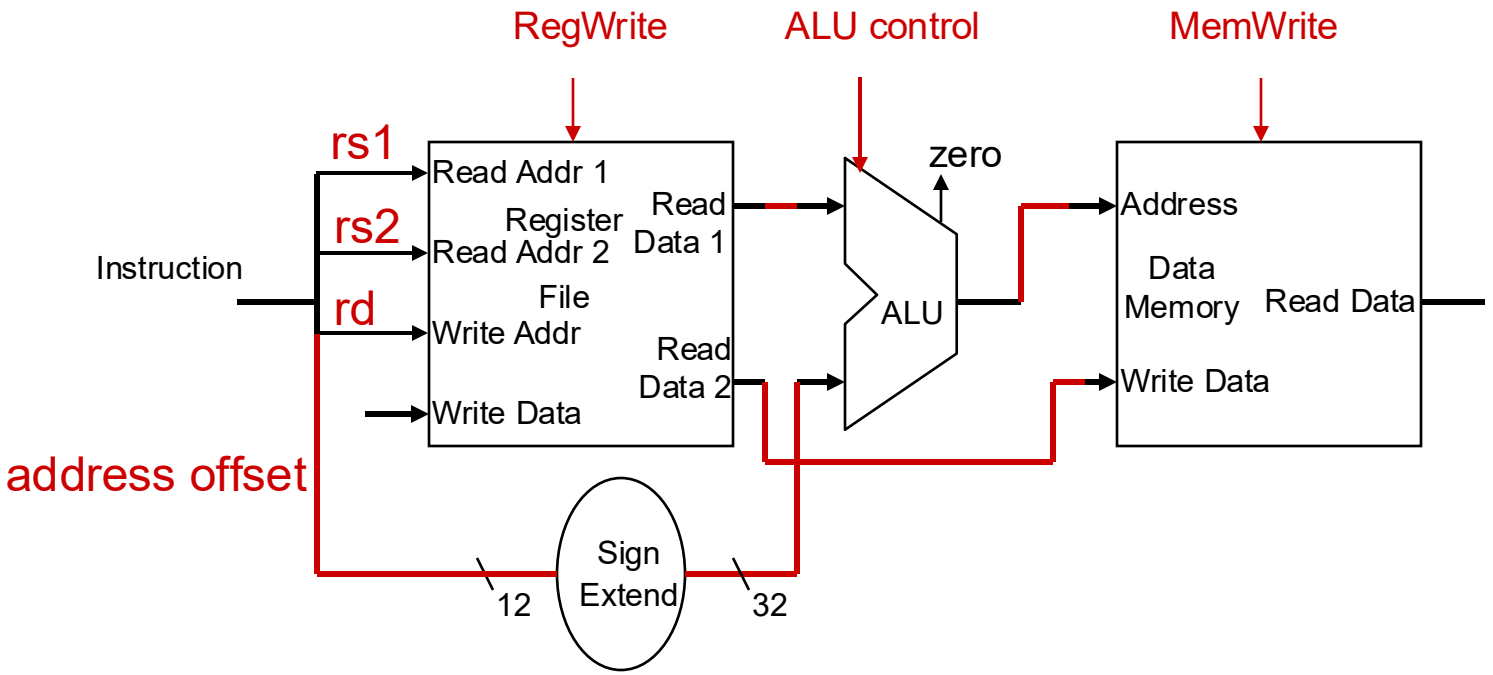


  - Note that Register File is not written every cycle (e.g., `sw` or `beq`), so we need an explicit write control signal for the Register File

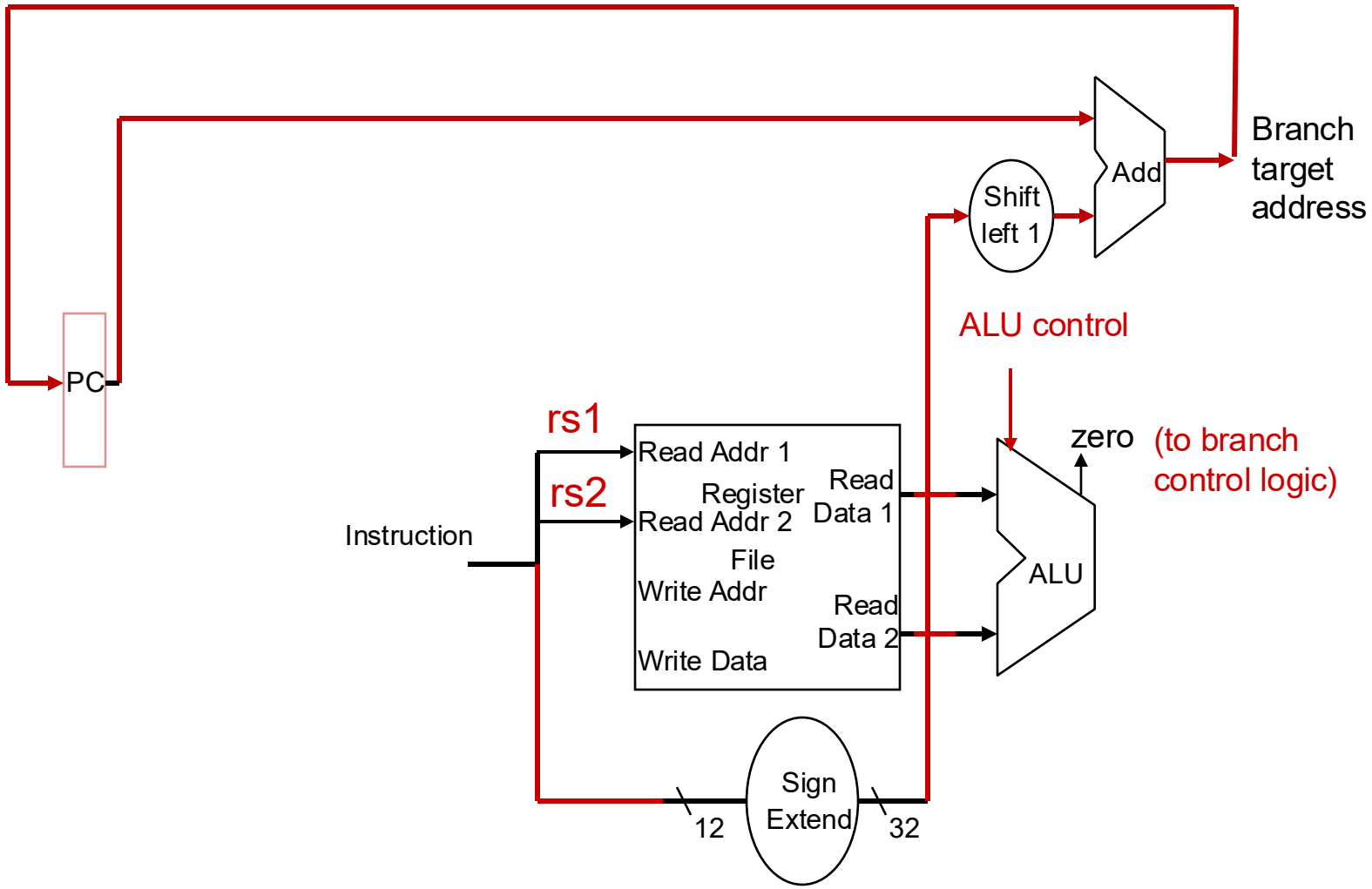
Executing Load Operations (cont.)



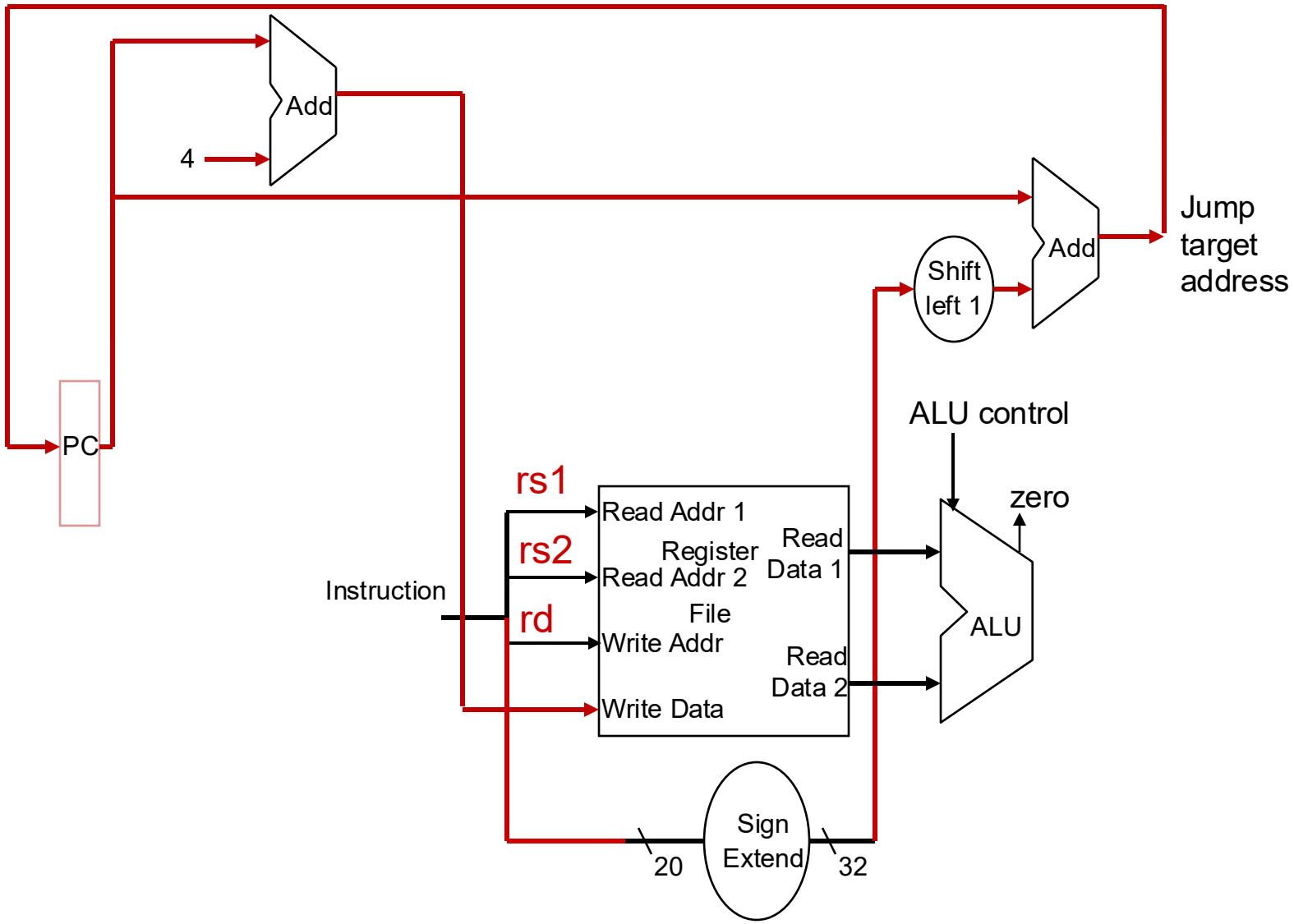
Executing Store Operations (cont.)



Executing Branch Operations (cont.)



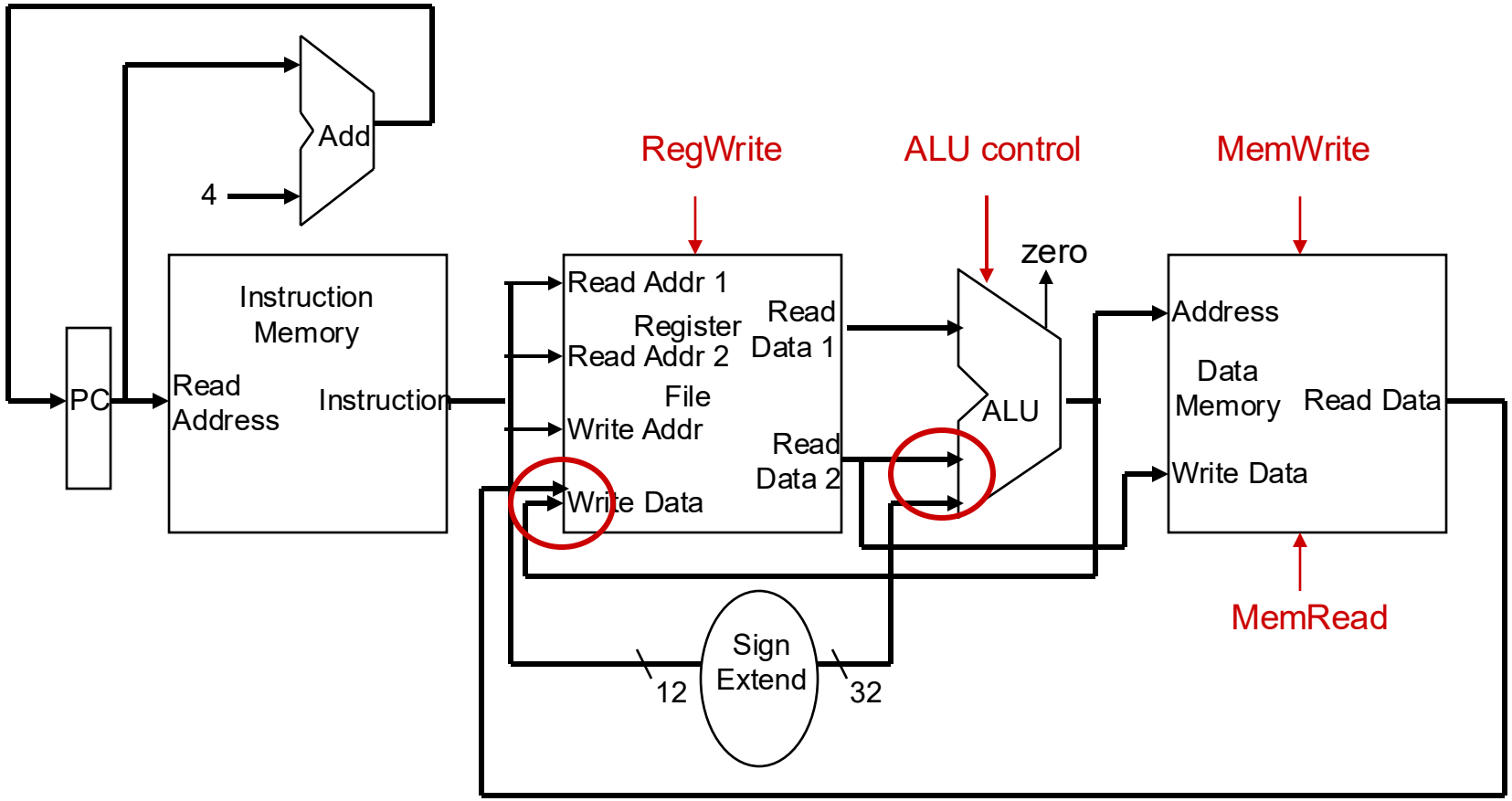
Executing Jump Operations (cont.)



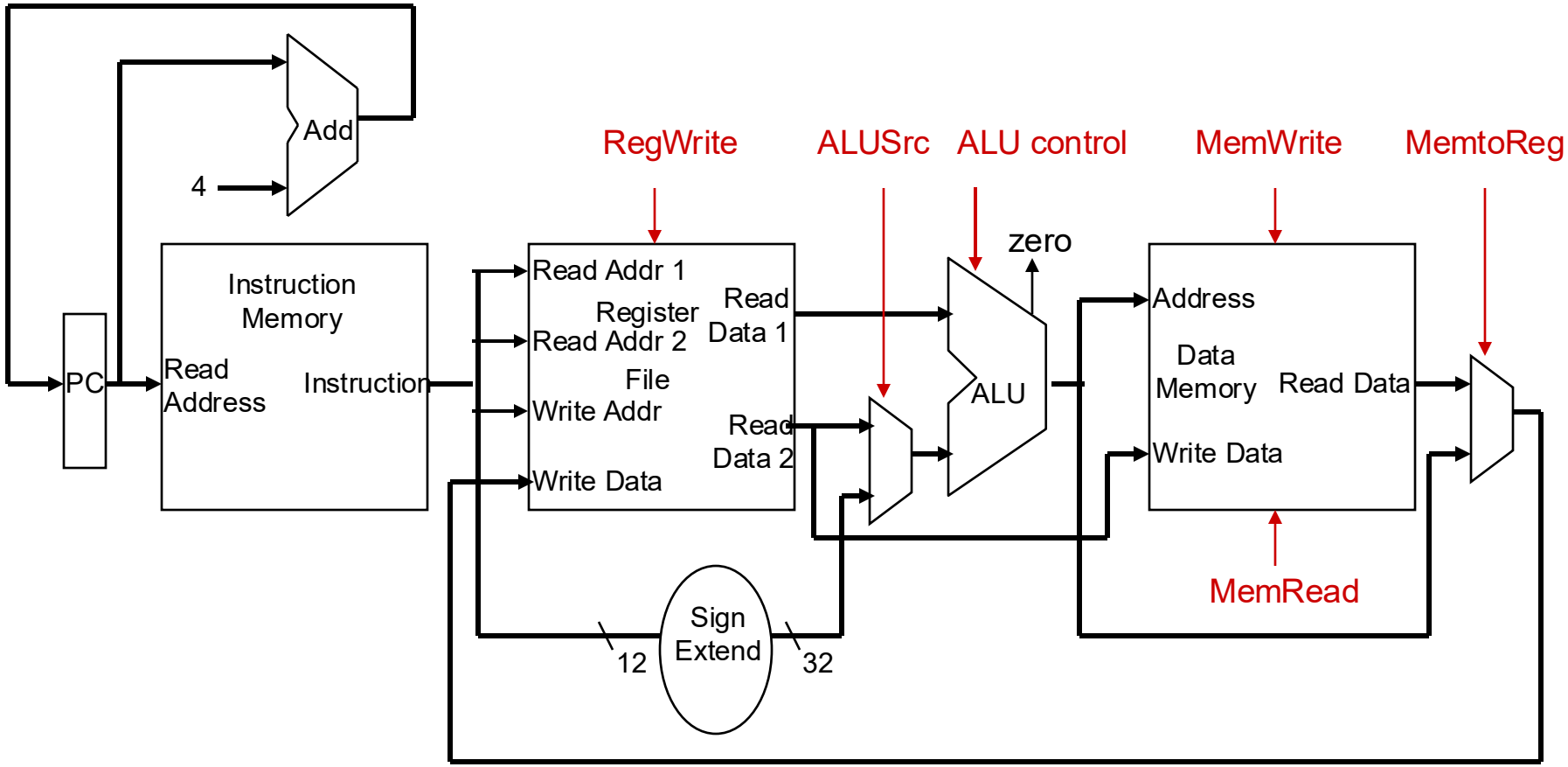
Creating a Single Datapath from the Parts

- Assemble the datapath elements, add control lines as needed, and design the control path
- Fetch, decode and execute each instructions in one clock cycle – single cycle design
  - One instruction can't use same resource/structure twice (ergo Harvard split memory architecture)
  - Two different instructions need multiplexors at the input of the shared elements with control lines to do the selection
- Cycle time is determined by length of the longest path

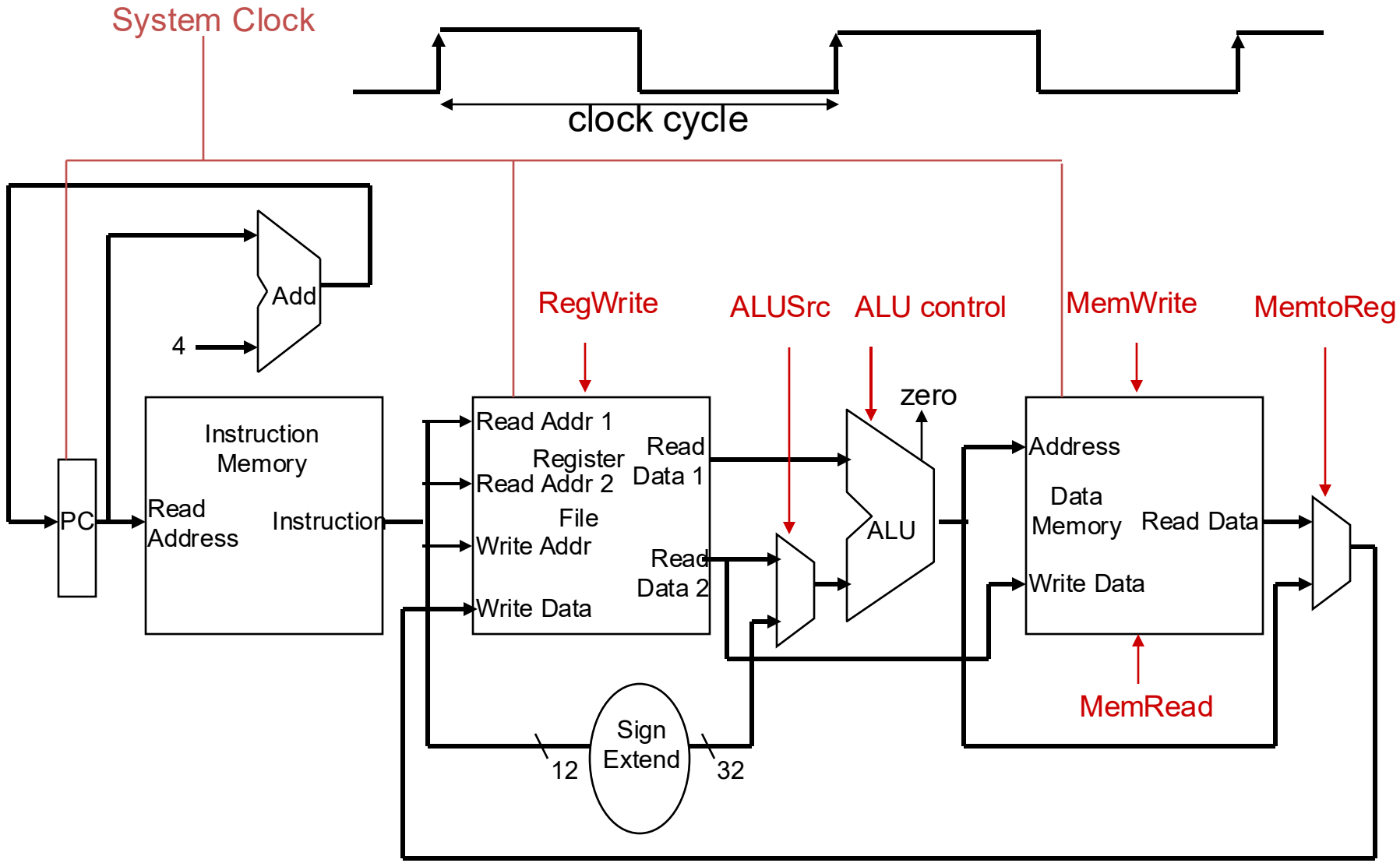
Fetch, R, and Memory Access Portions



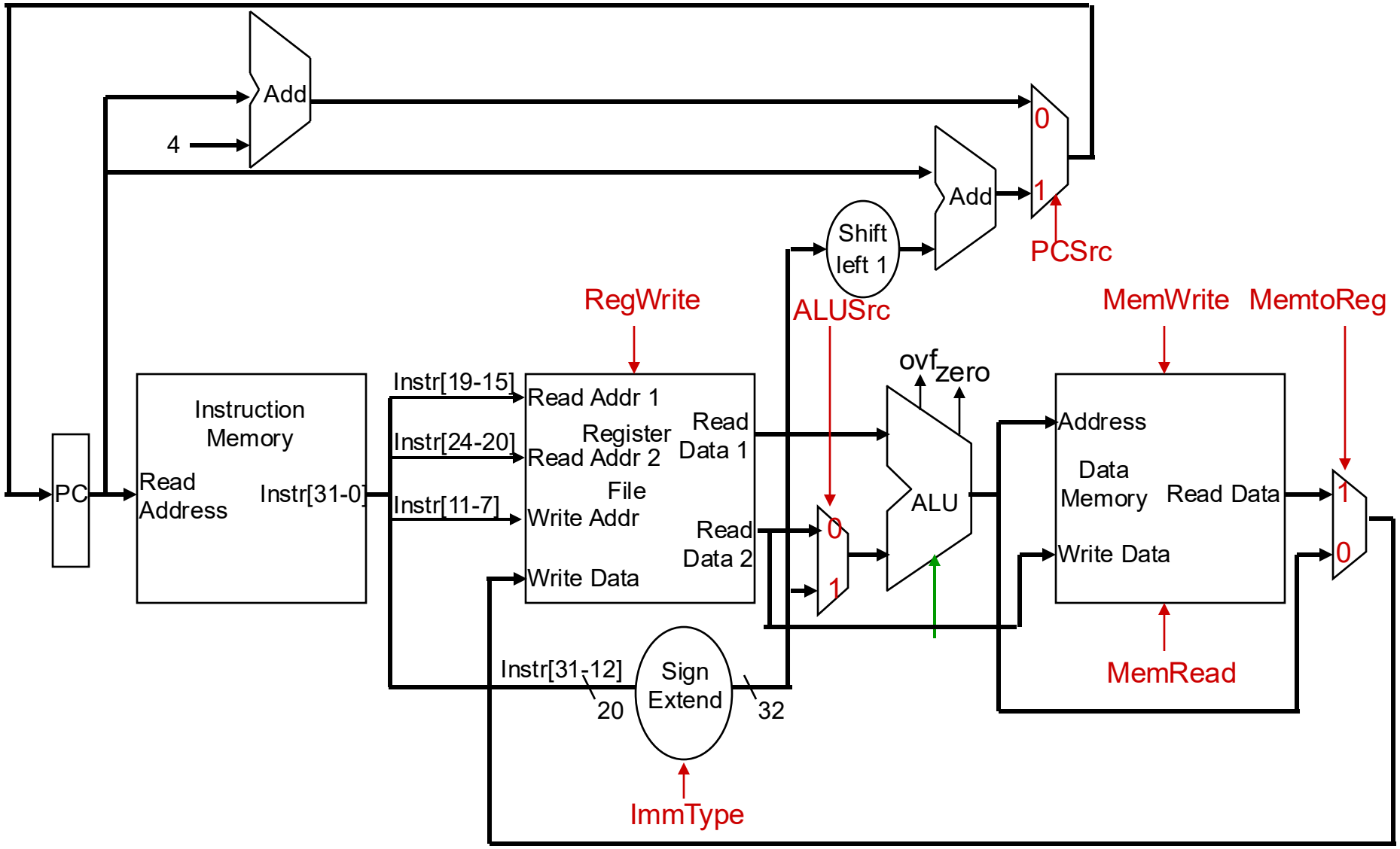
Multiplexor Insertion



Clock Distribution

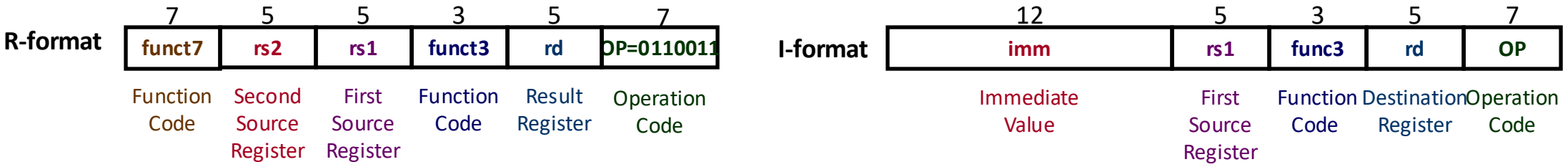


(Almost) Complete Single Cycle Processor



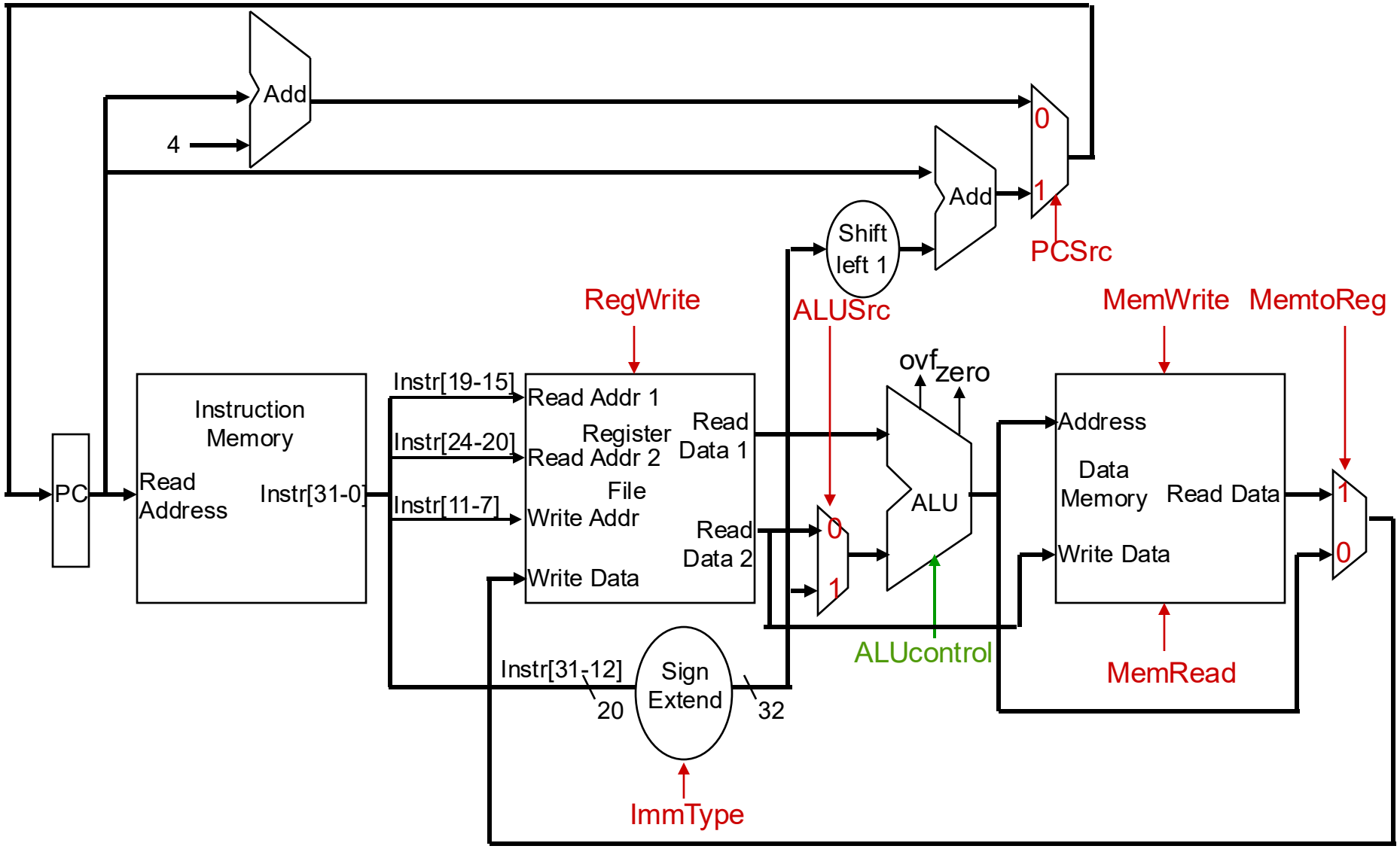
Adding the Control

- 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



- Observations
  - Opcode field always in bits 6-0
  - When op field is 0110011, funct3 field (14-12) used (and sometimes a bit from funct7 – bit 30)

(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.)

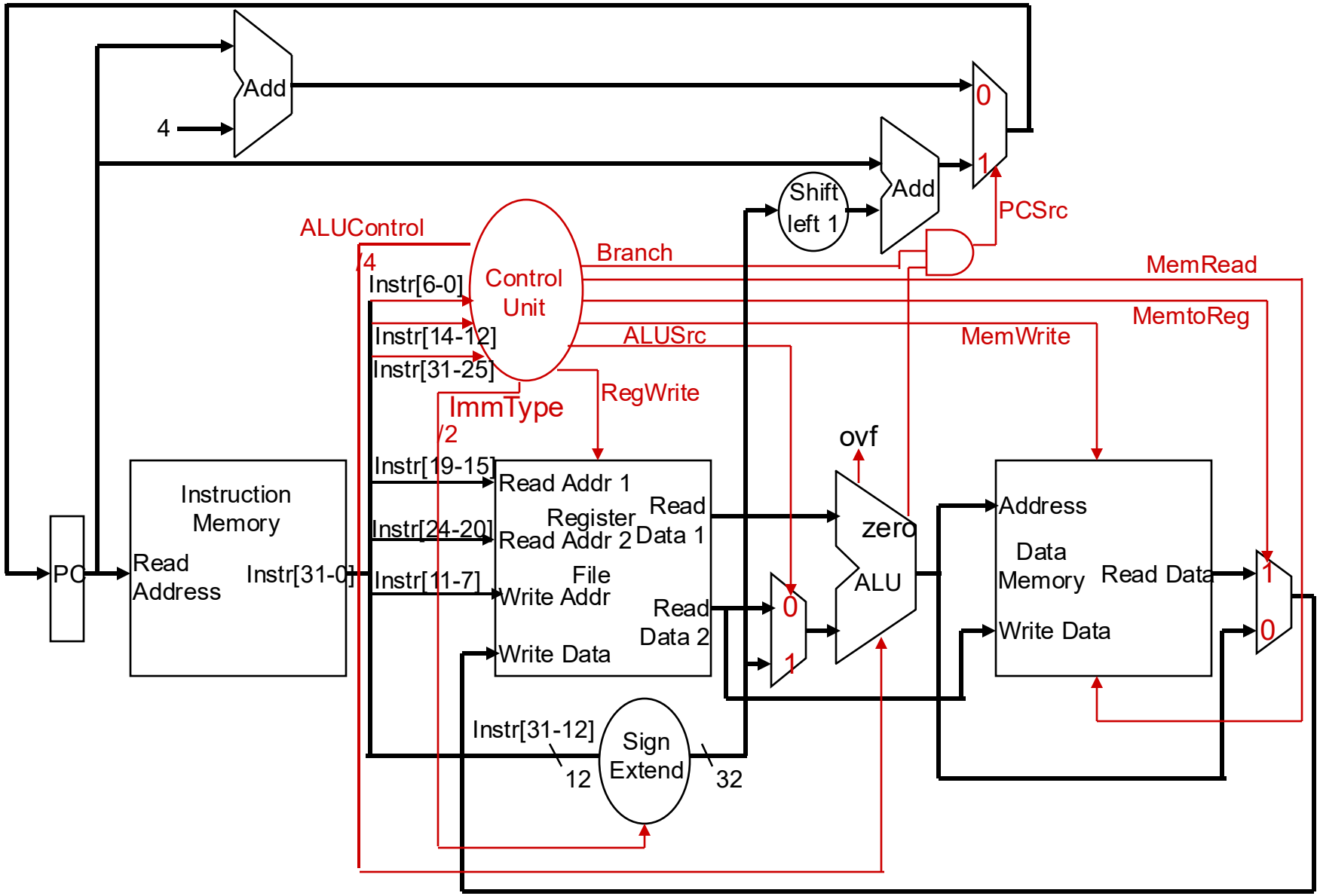
- So describe with a truth table

Four truth tables

Instr op	Opcode	Funct3	Funct7	Action	ALUcontrol
lw	0000011	010	xxxxxxx	add	0110
sw	0100011	010	xxxxxxx	add	0110
beq	1100011	000	xxxxxxx	sub	1110
add	0110011	000	0000000	add	0110
sub	0110011	000	0100000	sub	1110
and	0110011	111	0000000	and	0000
or	0110011	110	0000000	or	0001
xor	0110011	100	0000000	xor	0010
slt	0110011	010	0000000	slt	1111

17 input bits (although not all rows needed)

(Almost) Complete Datapath w/ Control

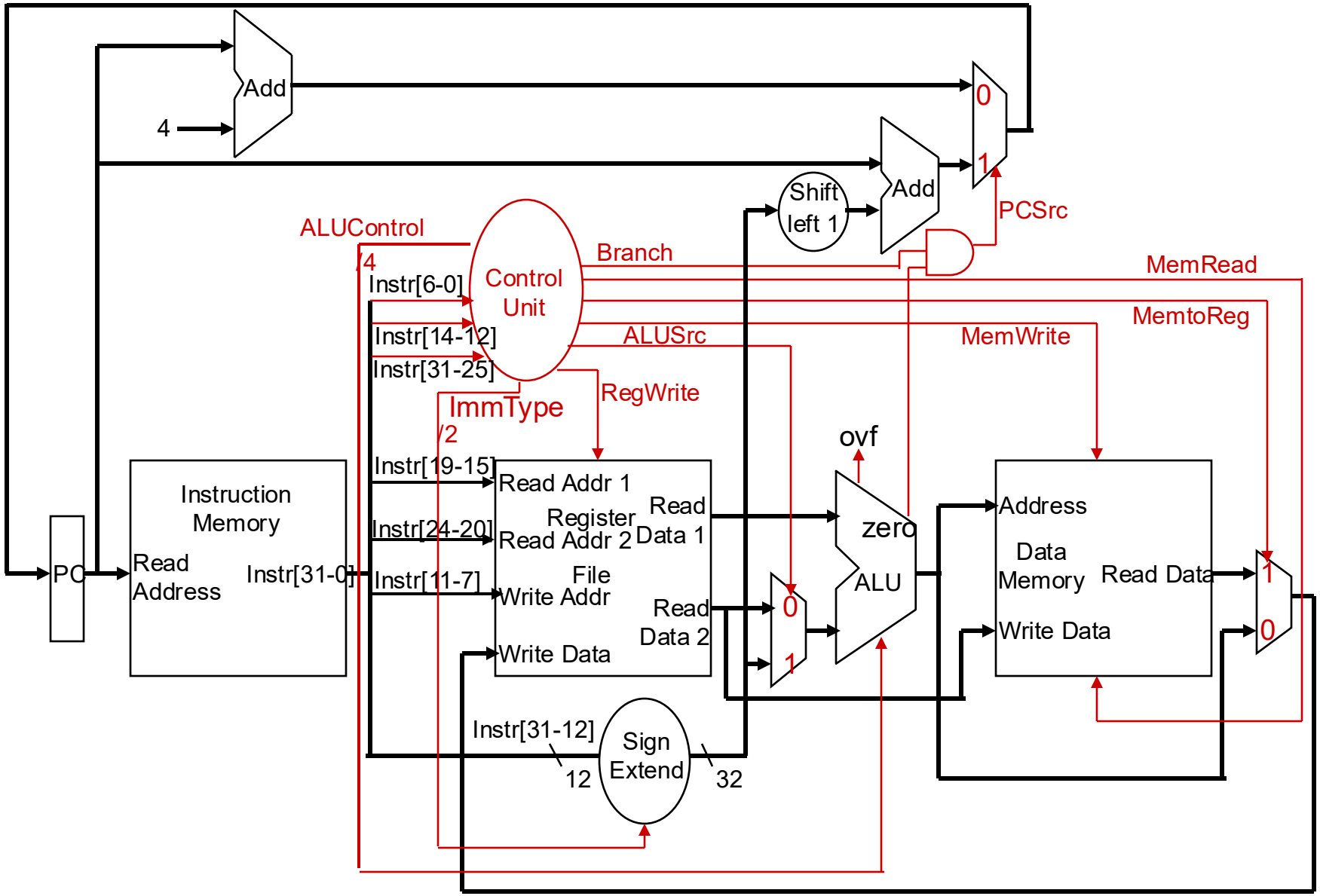


Main Control Unit

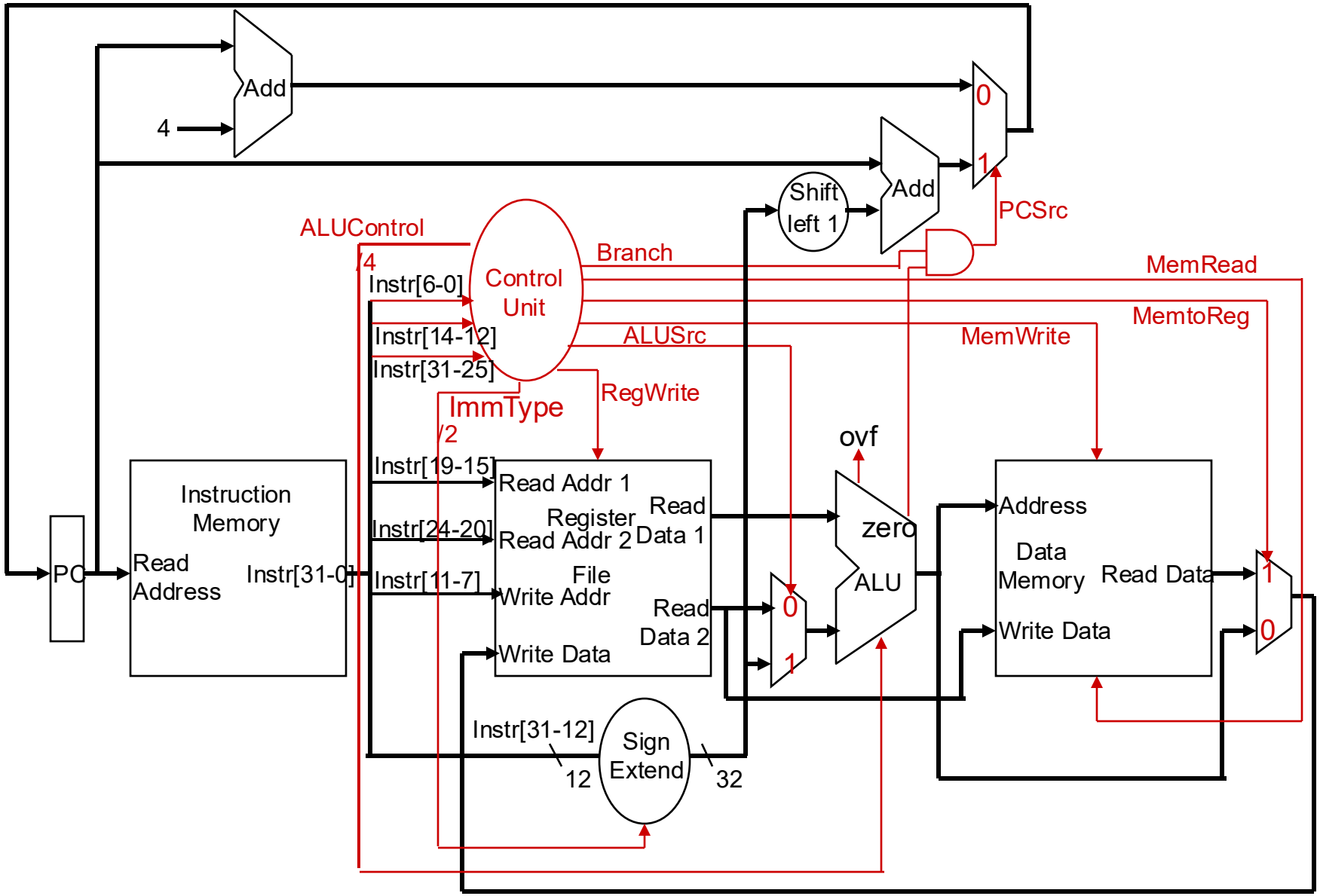
Instr	ALU control	Imm Type	ALUSrc	MemReg	RegWr	MemRd	MemWr	Branch
R-type 0110011								
lw 0000011								
sw 0100011								
beq 1100011								

- Note that a multiplexor whose control input is 0 has a definite action, even if it is not used in performing the operation

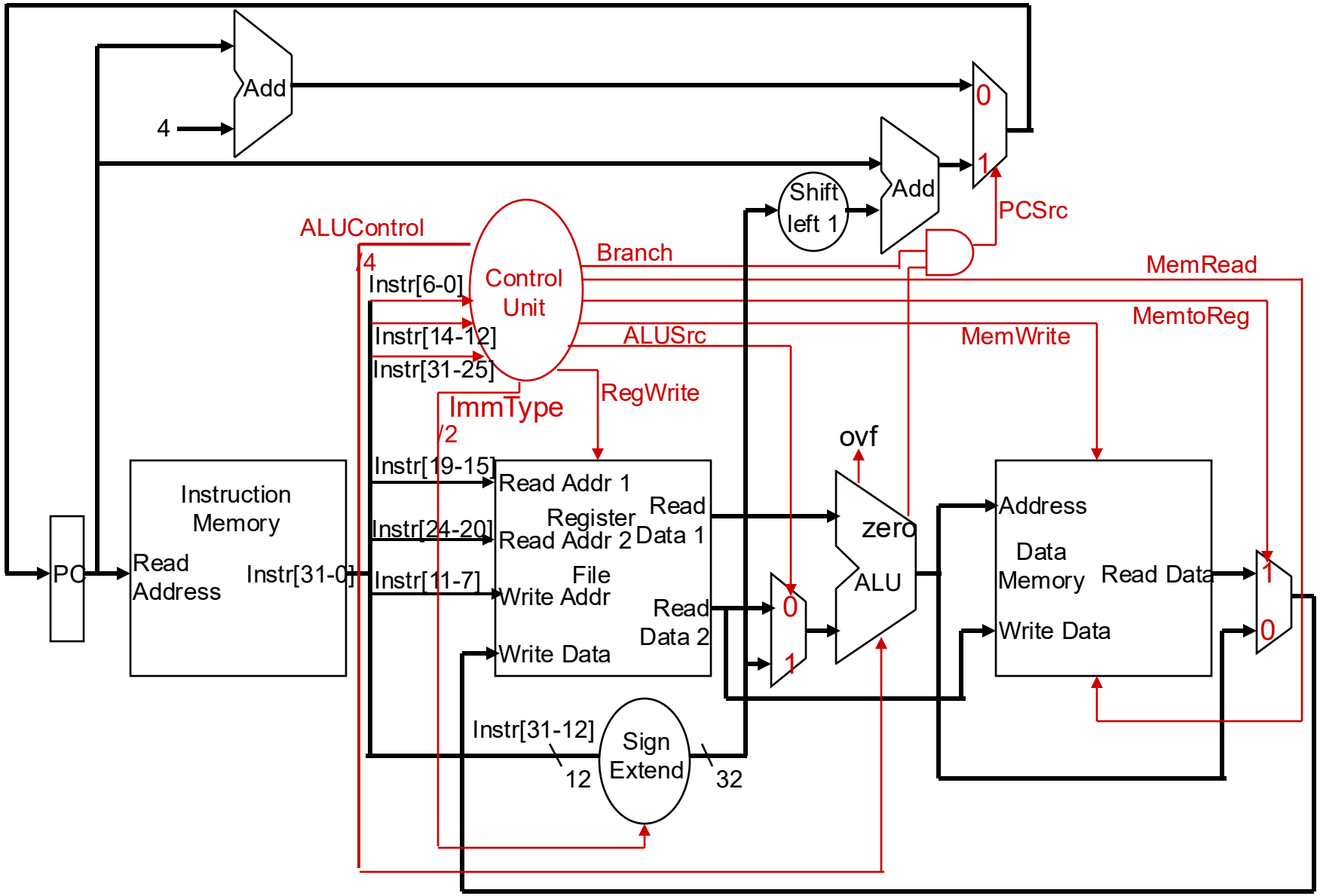
R-Type Instruction Data/Control Flow



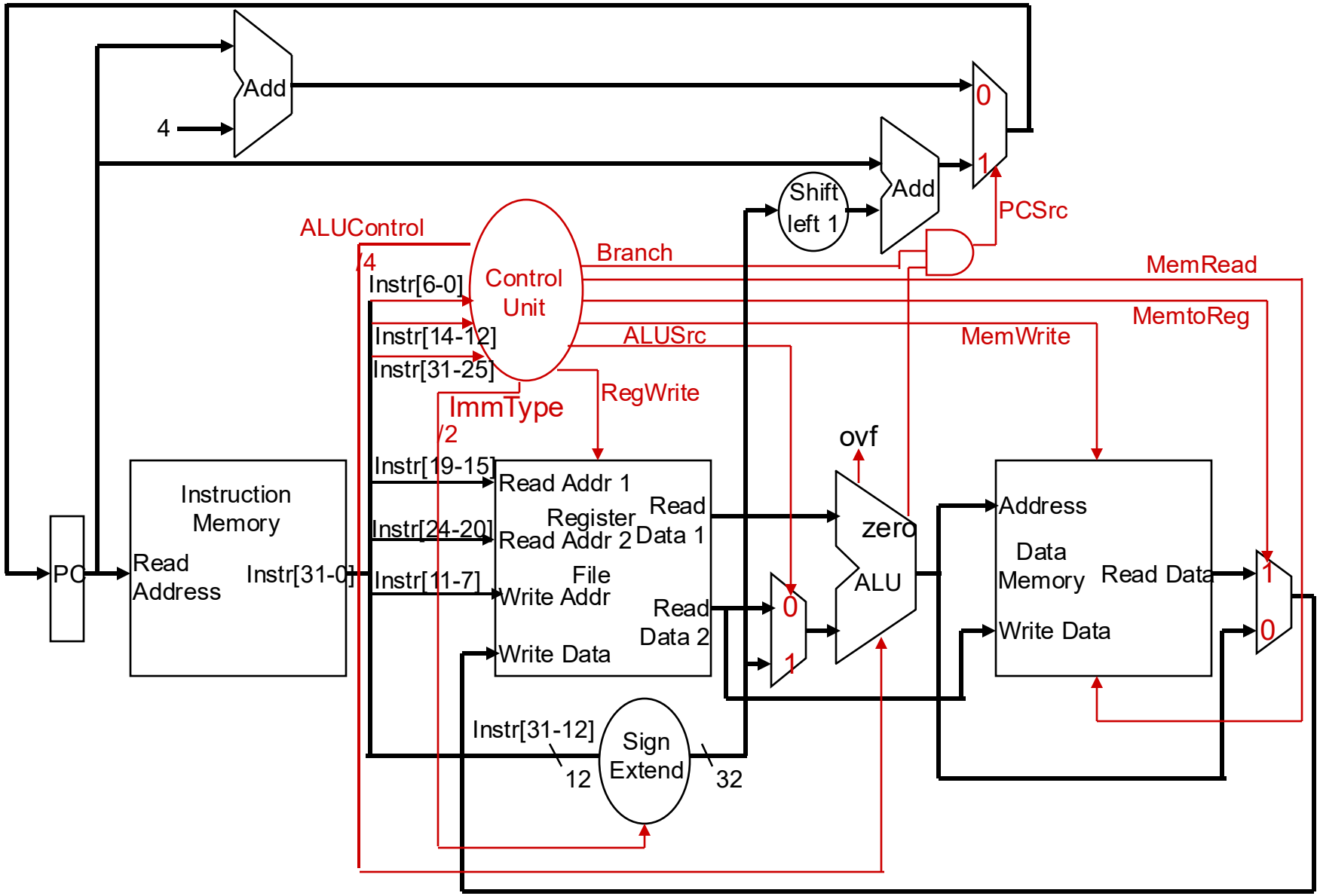
lw Instruction Data/Control Flow



sw Instruction Data/Control Flow



beq Instruction Data/Control Flow

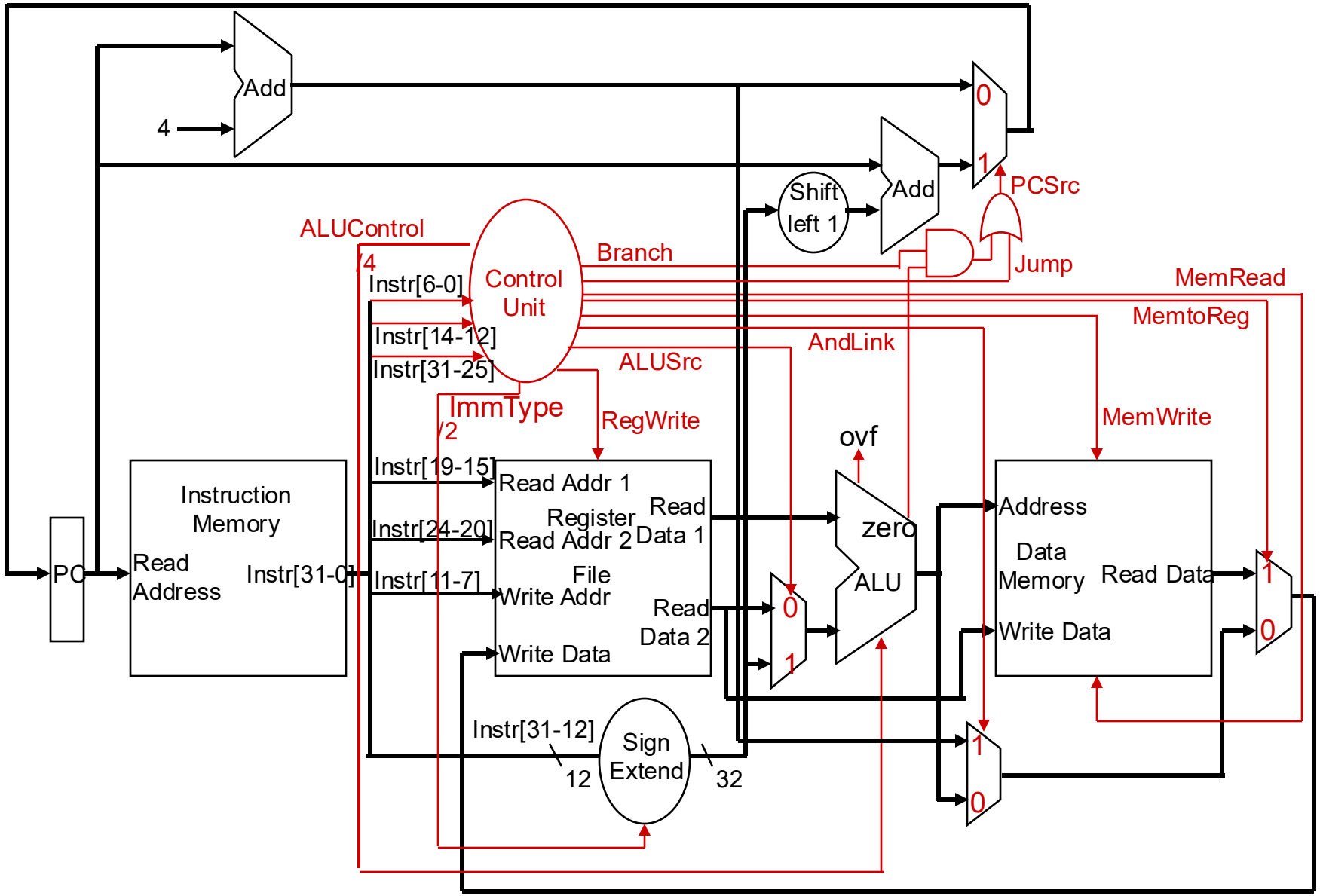


Main Control Unit (cont.)

Instr	ALU control	Imm Type	ALUSrc	MemReg	RegWr	MemRd	MemWr	Branch
R-type 0110011	Depends on Funct3 and Funct7	XX	0	0	1	0	0	0
lw 0000011	0110	00	1	1	1	1	0	0
sw 0100011	0110	01	1	X	0	0	1	0
beq 1100011	1110	10	0	X	0	0	0	1

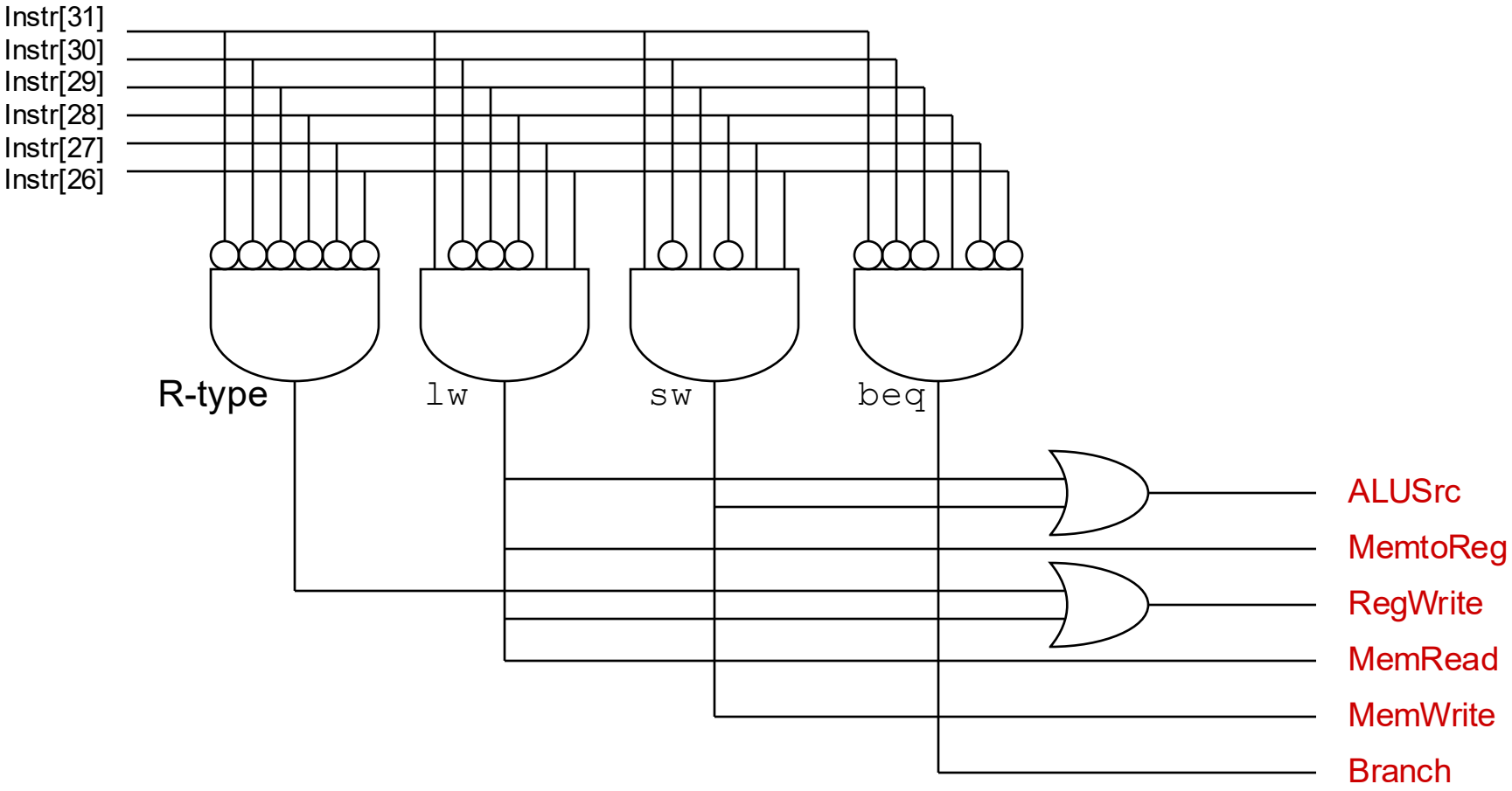
- Setting of the MemRd signal (for R-type, sw, beq) depends on the memory design (could have to be 0 or could be a X (don't care))

jal Instruction Data/Control Flow



Control Unit Logic

- From the truth table can design the Main Control logic

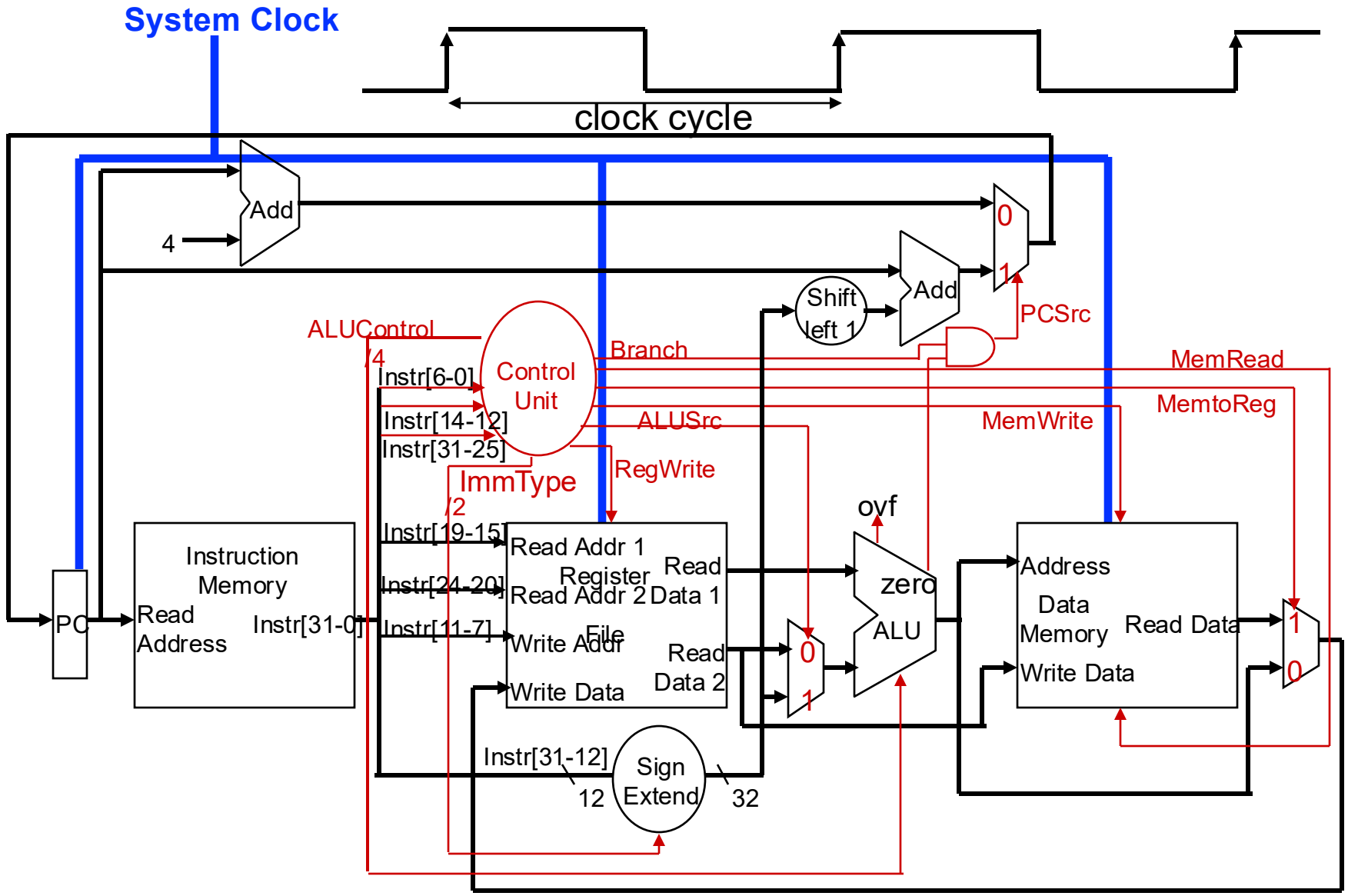


Model Truth Table in VHDL

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



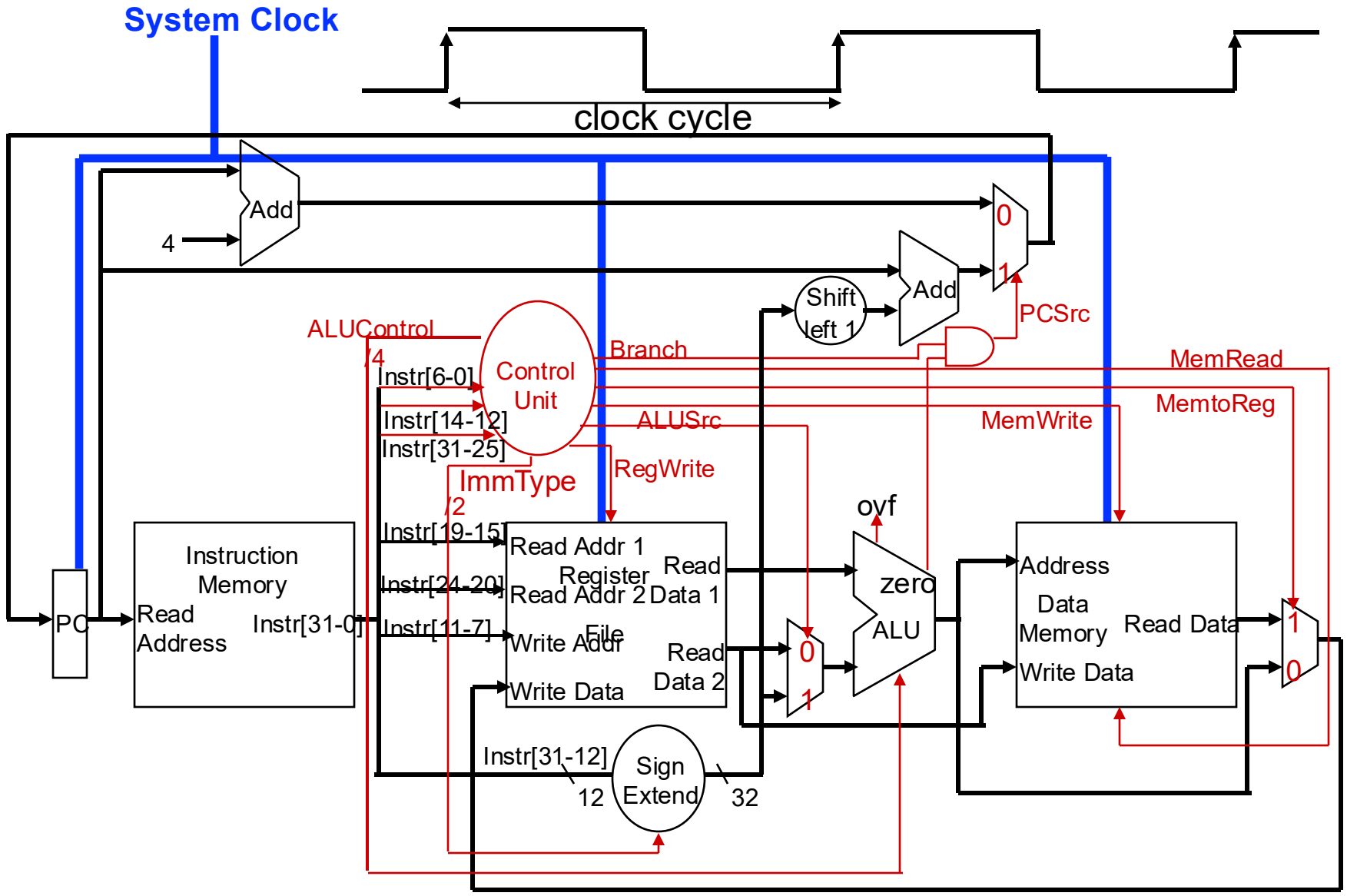
Operation

- 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)

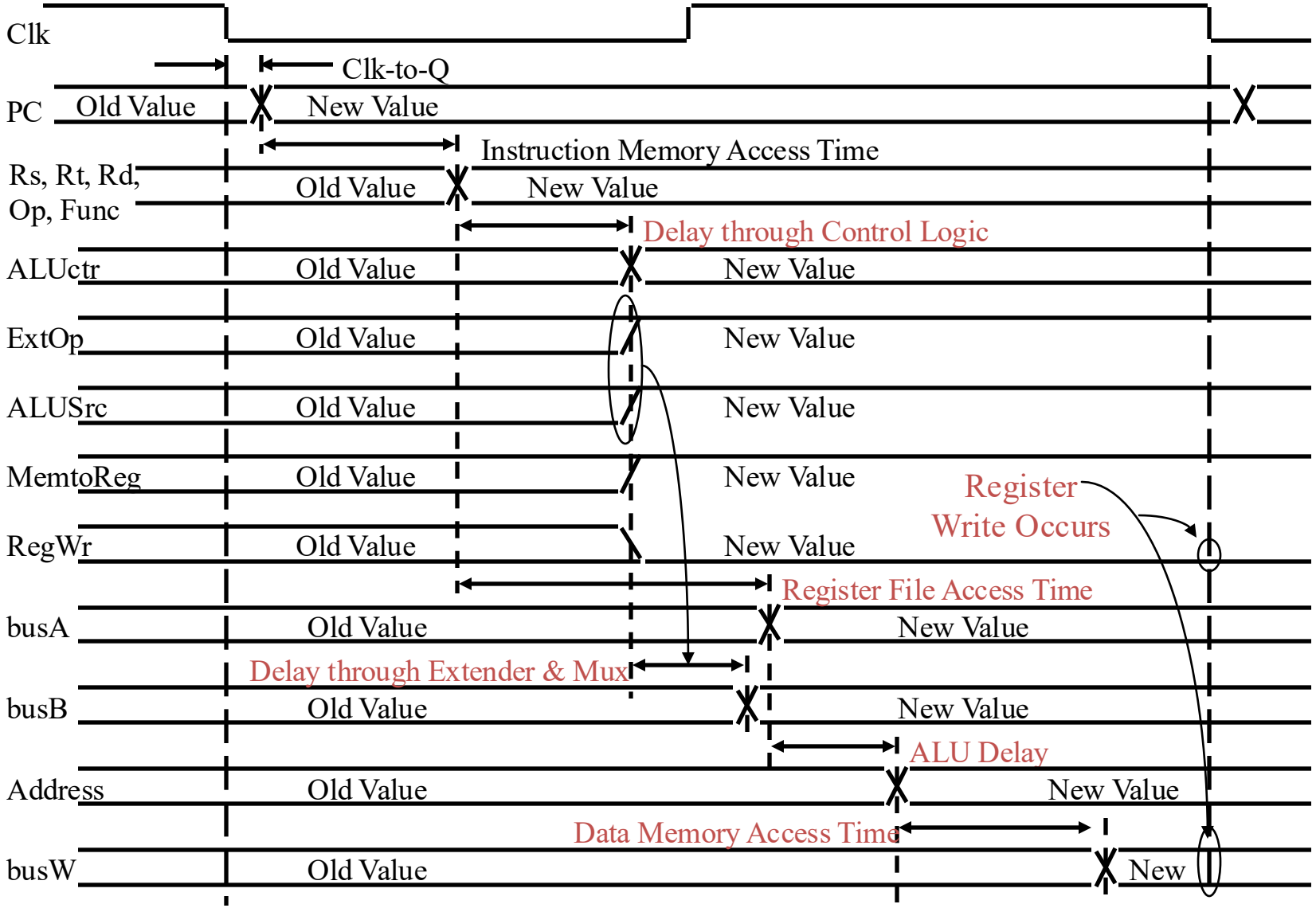
- The clock cycle time is determined by the logic delay through the longest path

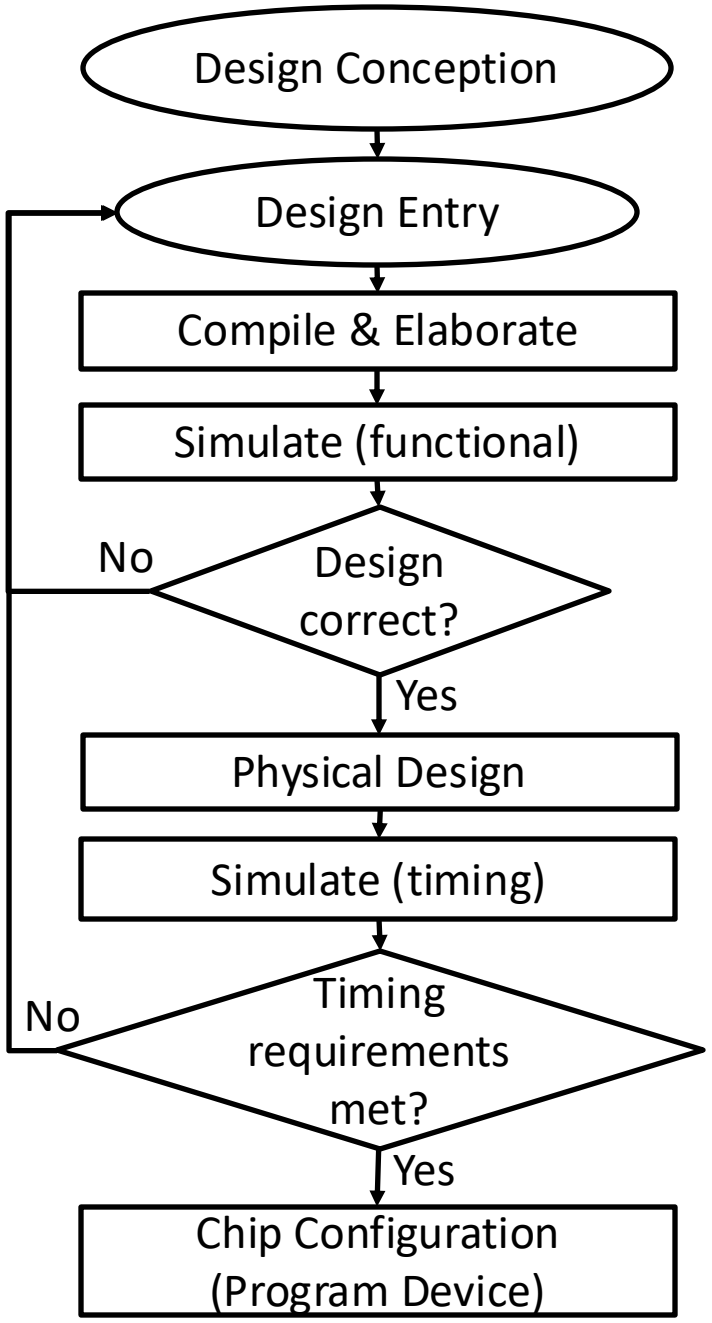
We are ignoring some details like register setup and hold times

Worst Case Timing (Load Instruction)



Worst Case Timing (Load Instruction)





[MODIFIED Figure 2.35 from the 281 textbook]

Tools: **ASMs**

HDLs

(VHDL)

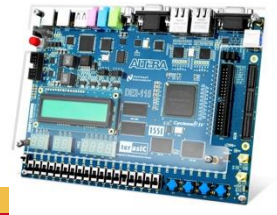
QUARTUS™

ModelSim

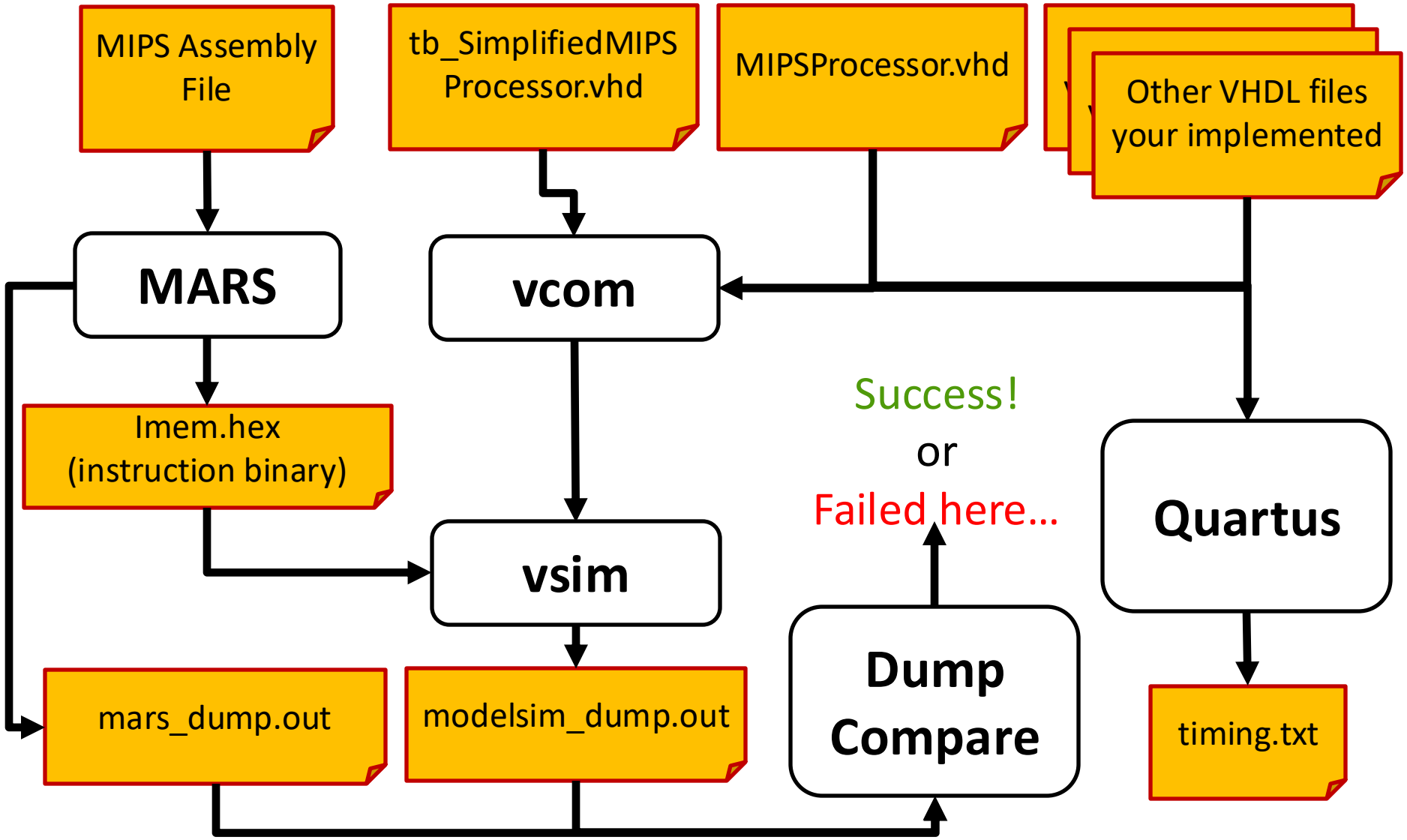
QUARTUS™

ModelSim

QUARTUS™



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)