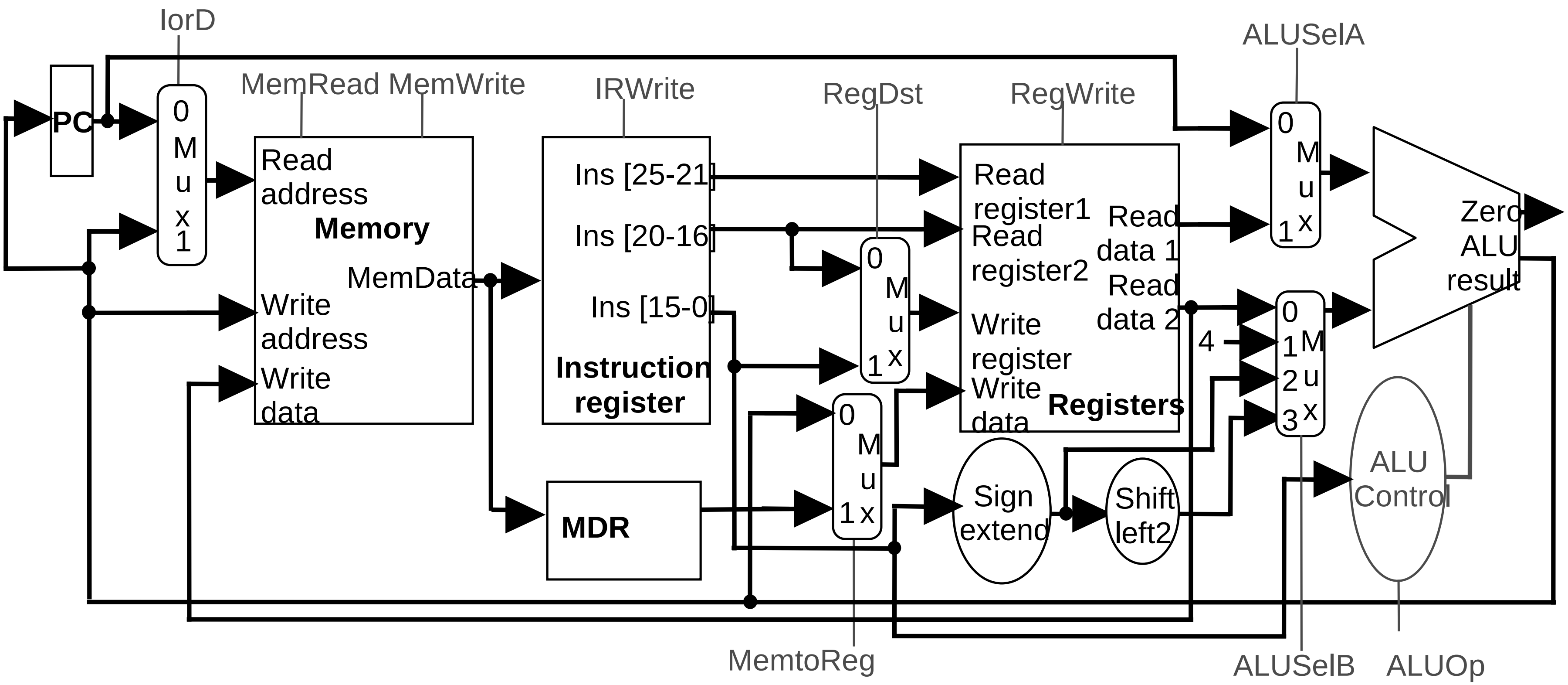
A Multiclock Design

A multiclock cycle design allows datapath components to be used multiple times in the execution of an instruction, saving hardware. More importantly, the design can be generalized (later) into a pipelined design for even greater speed.

The multiclock design could effectively realize the goals of the "variable clock length" design, postulated earlier.

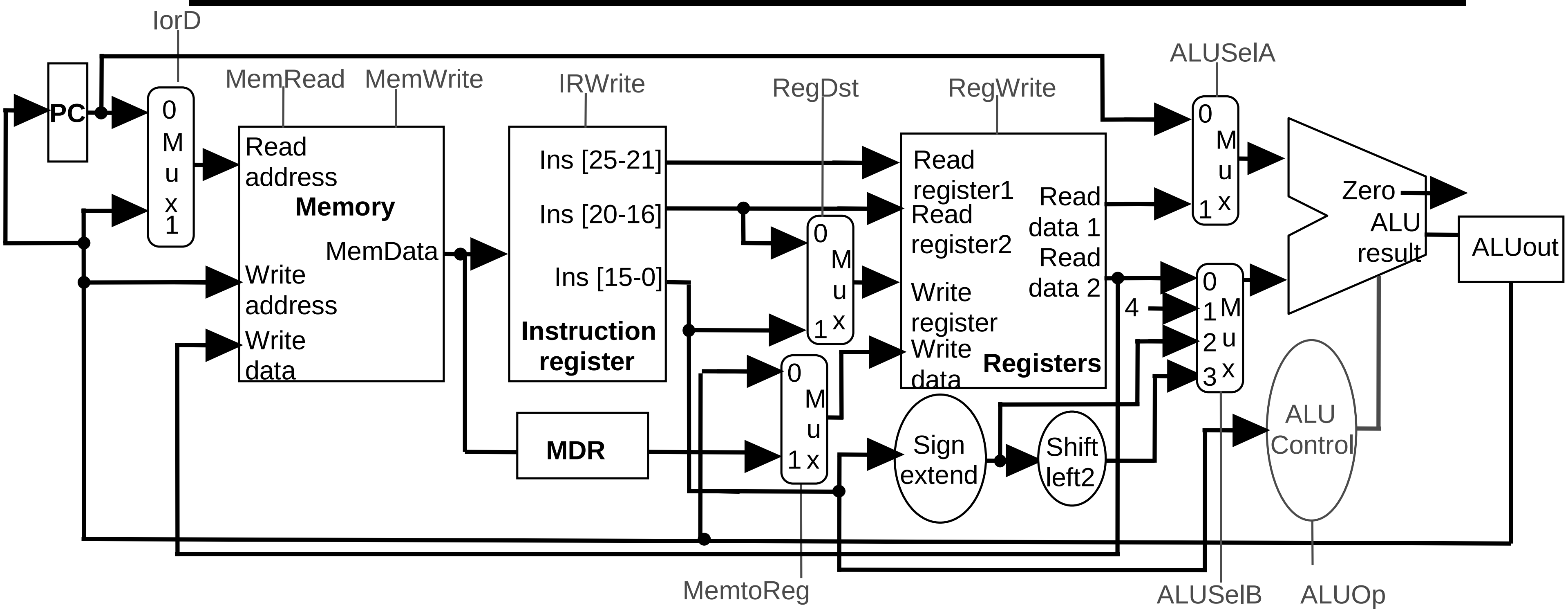
The goal is to balance the work taking place during each cycle so that the tick can be as small as possible.

Basic Datapath



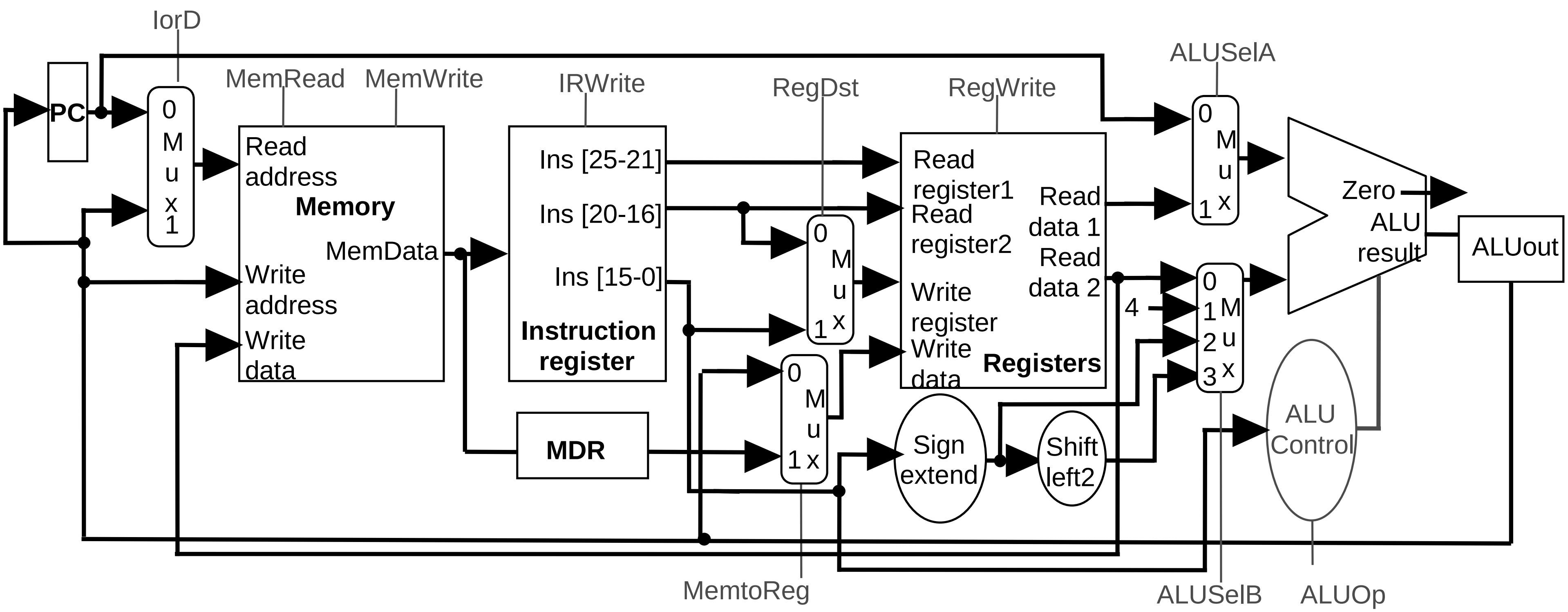
Control Signals

Signal	Effect Deasserted	Effect Asserted
MemRead	None	Contents at address read out
MemWrite	None	Contents at address replaced
ALUSelA	PC is first ALU operand	Reg is first ALU operand
RegDst	Reg Destination from rt	Reg Destination from rd
RegWrite	None	Contents of reg replaced
MemtoReg	Data to reg from ALU	Data to reg from memory
IorD	Mem Address from PC	Mem Addr from ALU
IRWrite	None	Value from memory to IR



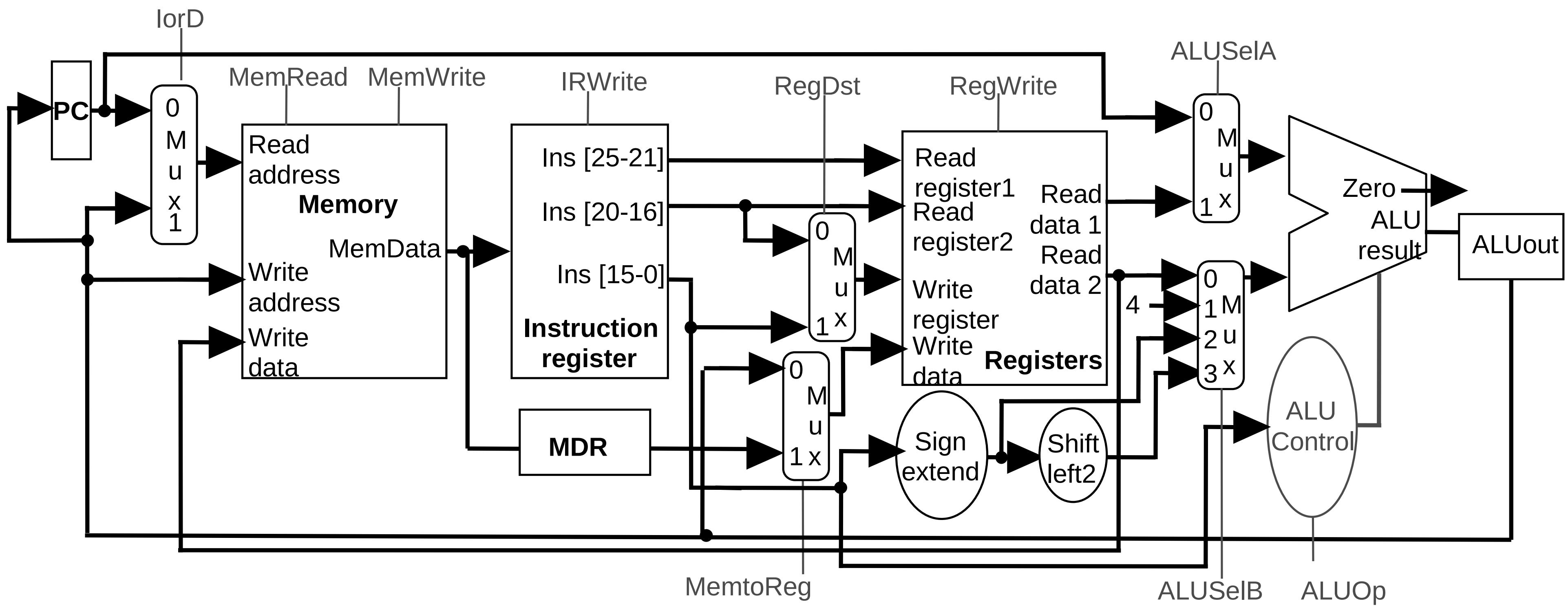
More Control Signals

Signal	Value	Effect
ALUSelB	00	Second ALU input from register rt
	01	Second ALU input is constant 4
	10	Second ALU input is sign extended 16 lsb of IR
	11	Second ALU input is sign extended 16 lsb of IR shifted
ALUOp	00	ALU performs addition
	01	ALU performs subtraction
	10	ALU operation comes from function code of instruction



5 Clock Cycle Execution

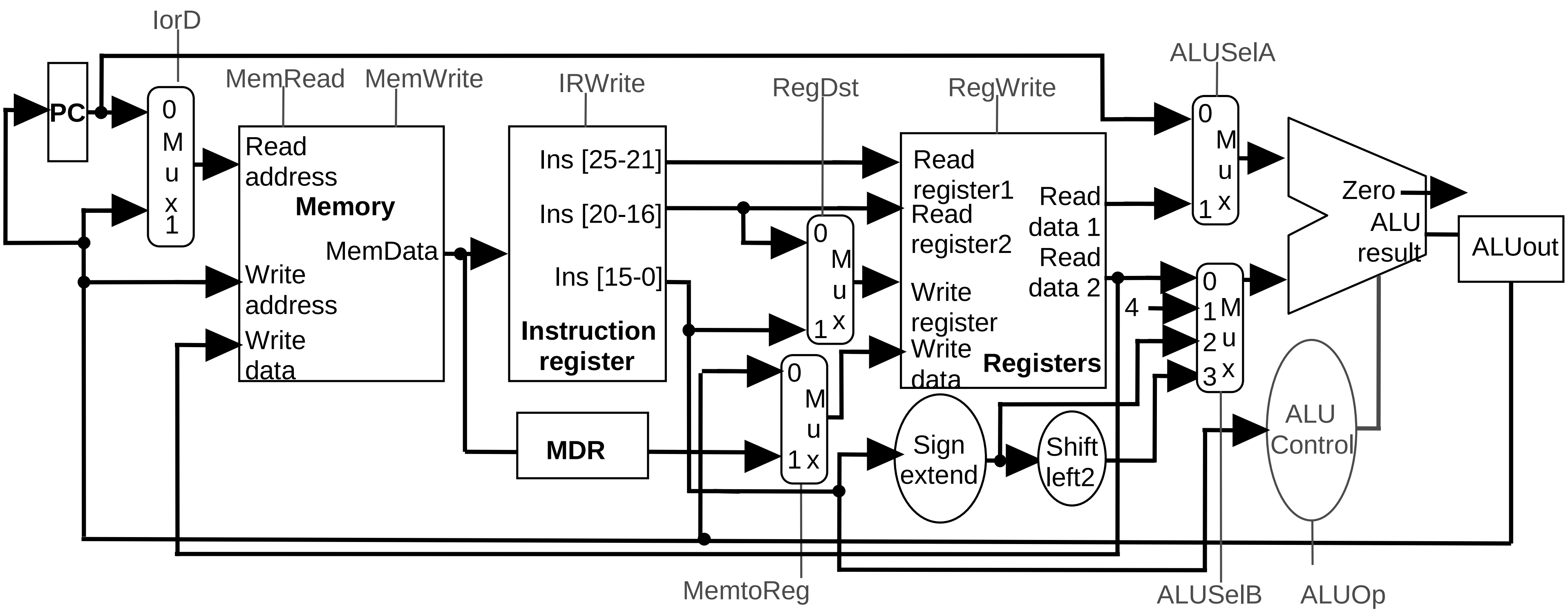
- Instruction Fetch, including PC = PC + 4
- Instruction Decode and Register Fetch
- Execution, Memory Address Computation, Branch Completion
- Memory Access or R-type Instruction Completion
- Write-back



Instruction Fetch

IR = Memory[PC]
PC = PC + 4

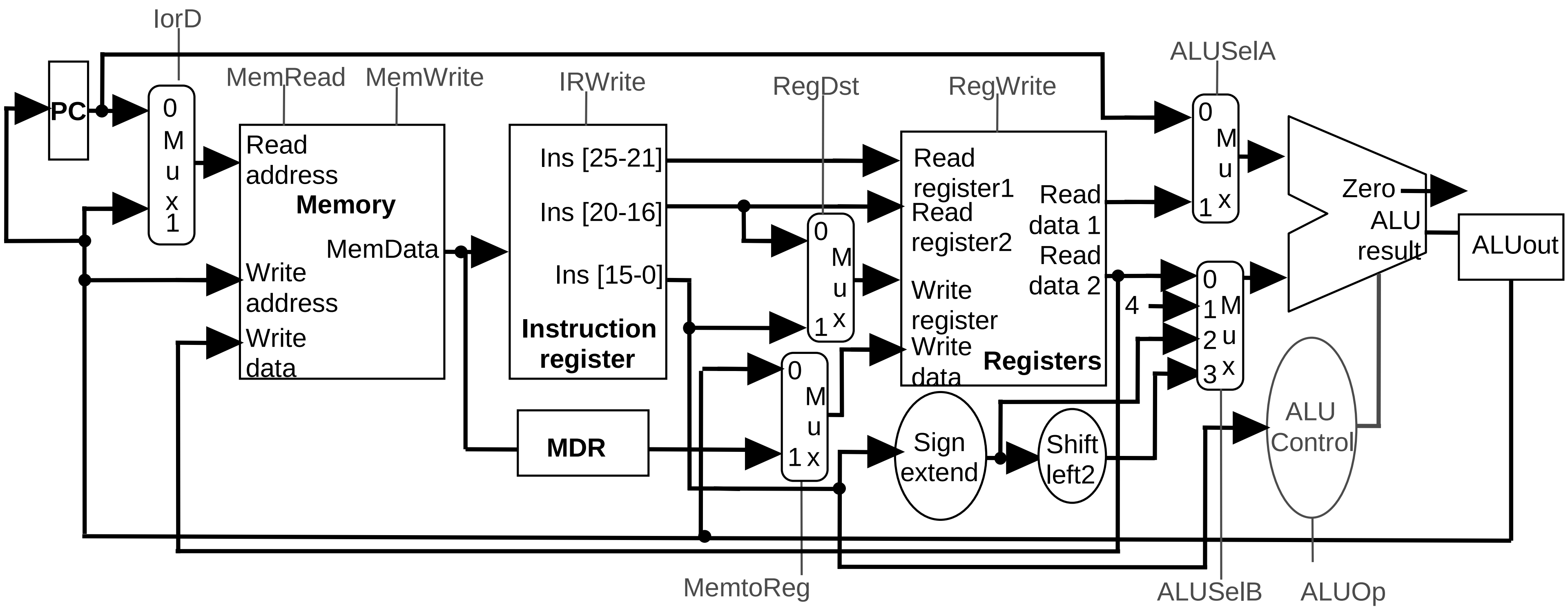
MemRead
IRWrite
IorD = 0
ALUSelB=01
ALUSelA=0
ALUOp=00



Instruction Decode & Register Fetch

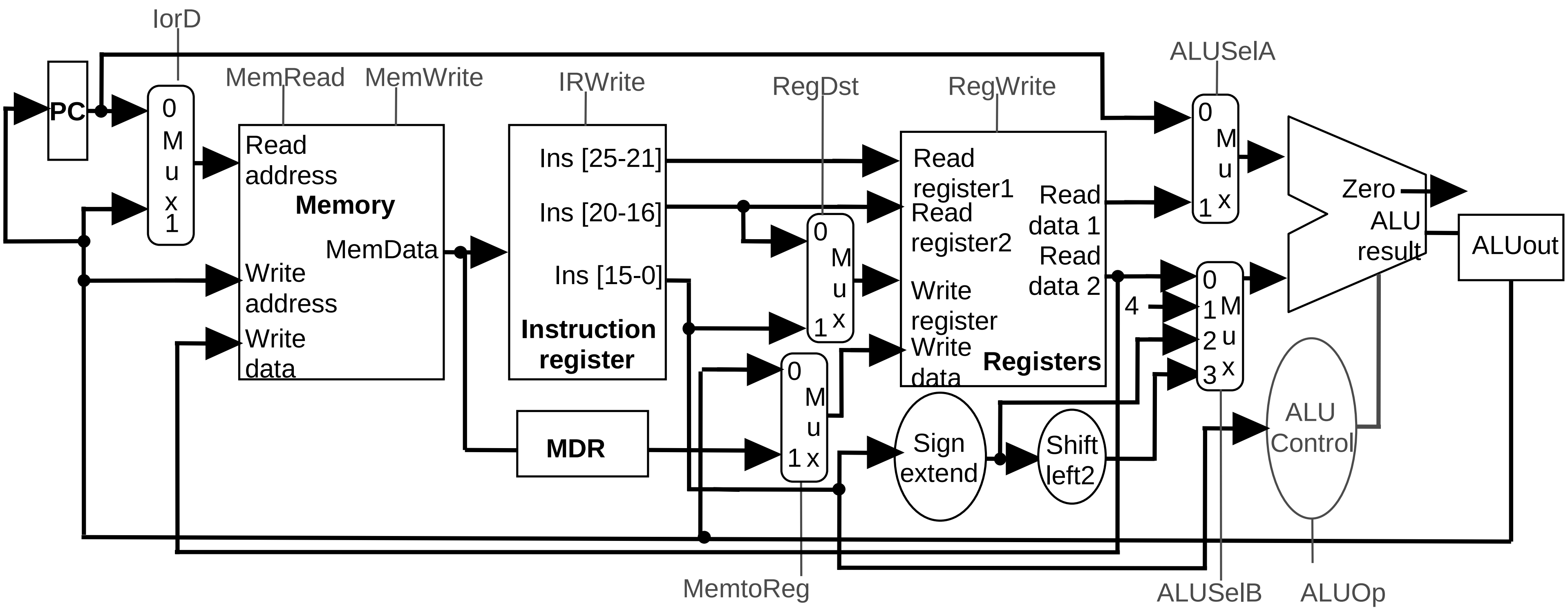
ALUSelB=11
ALUSelA=0
ALUOp=00

A = Register[IR[25-21]]
B = Register[IR[20-16]]
ALUout = PC + signextend(IR[15-0]) << 2)



Execution, Mem Address Comp, Branch Comp

Execution: ALUOut = A op B
Mem Addressing Comp: ALUOut = A + signextend(IR[15-0])
Branch Completion: if (A == B) PC = ALUout



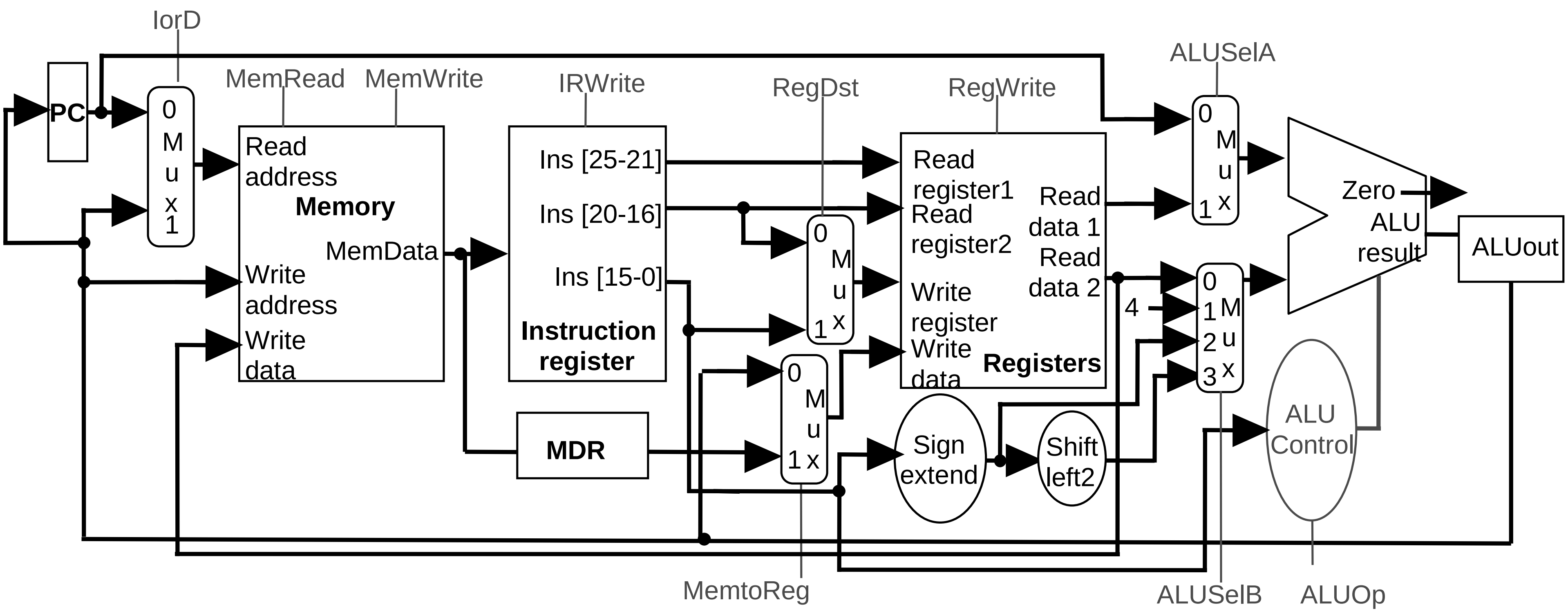
Mem Access or R-type Competion

Memory Ref: MDR = Memory[ALUout]

-or-

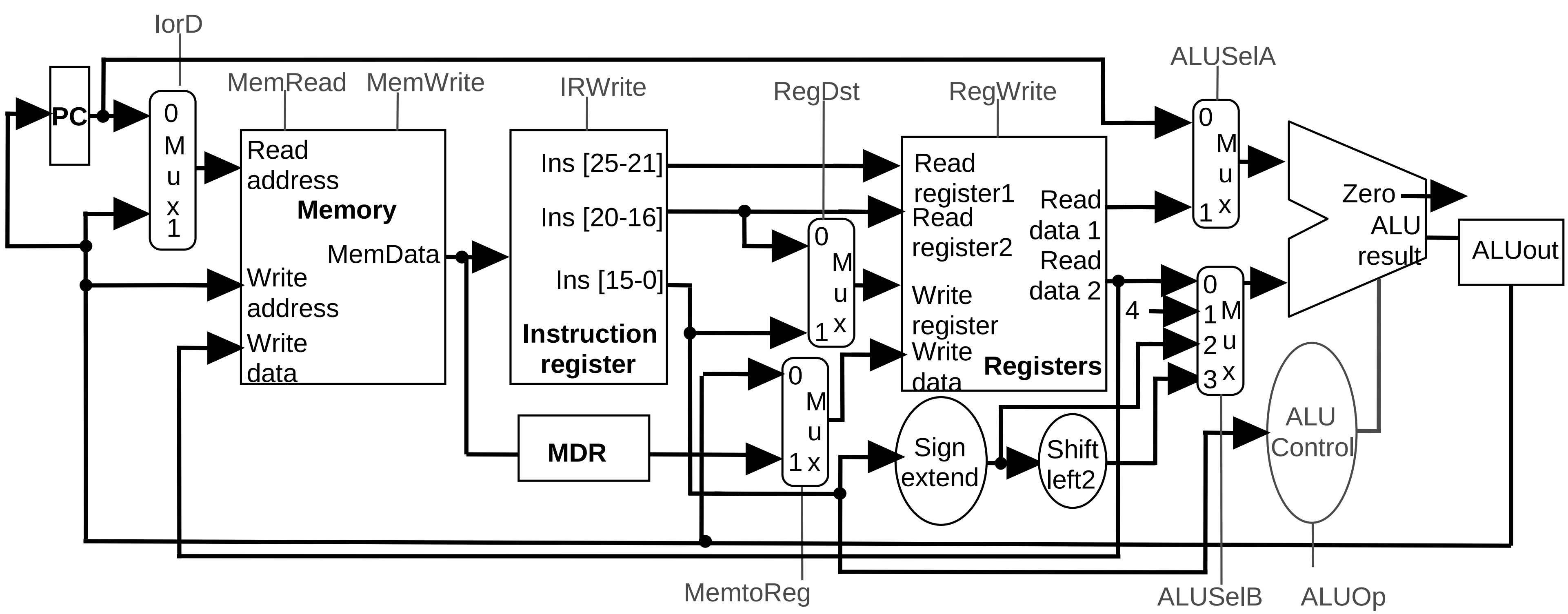
Memory[ALUout] = B

R-type Competion: Reg[IR[15-11]] = ALUout



Write-back

Register[IR[20-16]] = MDR



Instruction Cycles

Step Name	Action for R-type	Action for Mem Ref	Branch
Instruction fetch	IR=Memory[PC] PC=PC+4		
Instruction decode/ register fetch	A=Reg[IR[25-21]] B=Reg[IR[20-16]] ALUout=PC+(xtnd(IR[15-0])<<2)		
Execution, addr comp or branch completion	ALUout=A op B	ALUout=A+ xtnd(IR[15-0]])	if (A==B) then PC=ALUout
Memory access or R-type completion	Reg[IR[15-11]= ALUout	MDR=ALUout or Mem[ALUout]=B	
Write back		Reg[IR[20-16]]= MDR	