ECS 154B
Computer Architecture II
Winter 2009

Multi-Cycle MIPS
5.5

Adapted from slides by Mary Jane Irwin, Penn State

Reasons for Multi-cycle Operation

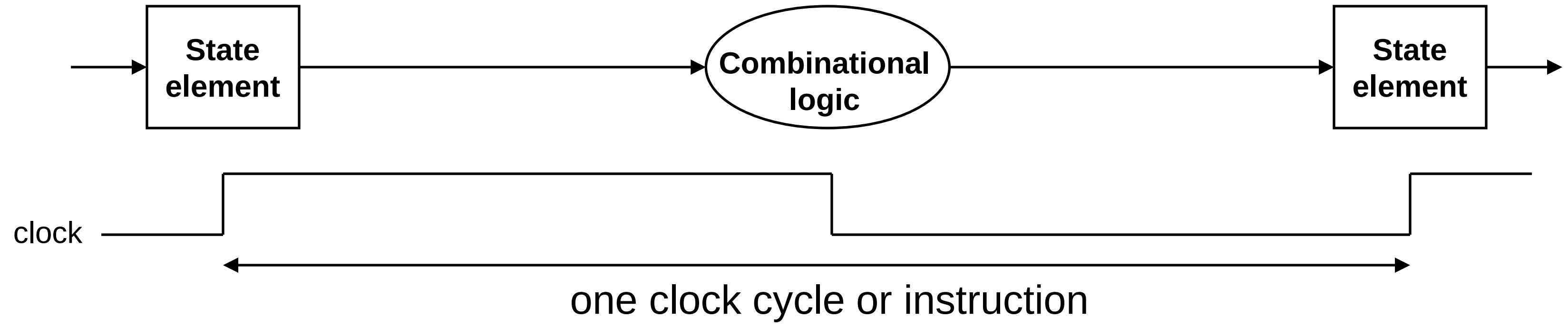
- Single cycle CPU wasteful
  - Clock period must accommodate slowest instruction
  - Multiple functional units (memory, adders)
- Multiple cycle CPU better
  - Clock period is determined by longest operation
  - Instructions can take a different number of clock cycles to complete
  - Functional units can be shared
    - Reduces chip area
    - However, only one use per clock cycle
  - Prepares you for introduction of pipelining

Multi-cycle Approach

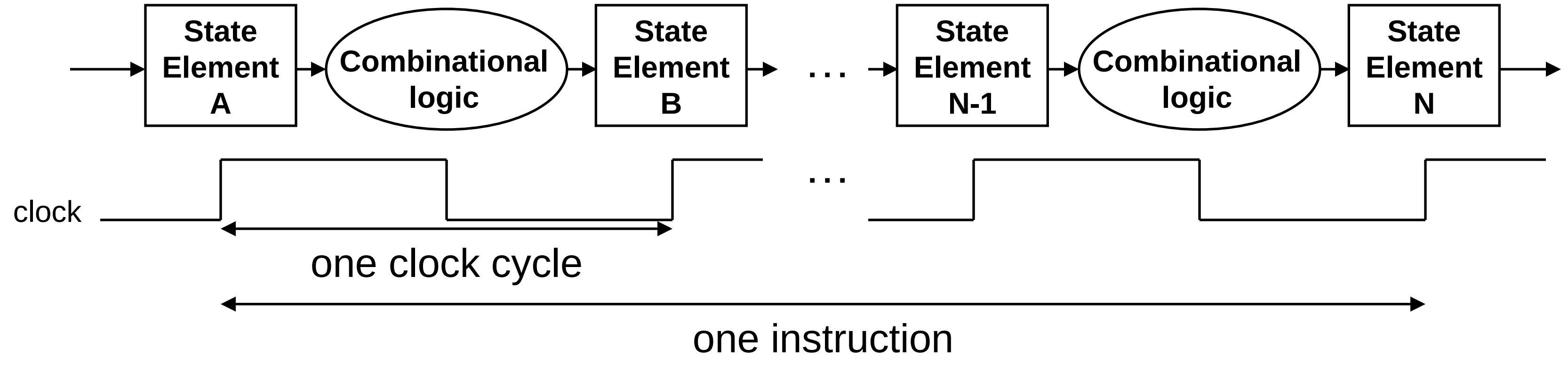
- Let instructions take more than one clock cycle
  - Break instruction into steps that occur in one cycle
    - Balance steps to minimize clock period
    - Each resource used only once per step
  - Not all instructions need every step
- Reuse resources to save chip area
  - ALU used to
    - calculate PC+4
    - memory address
    - branch destination
    - instruction operation
  - Single memory for instructions and data

Multi-cycle Approach

- Single cycle CPU

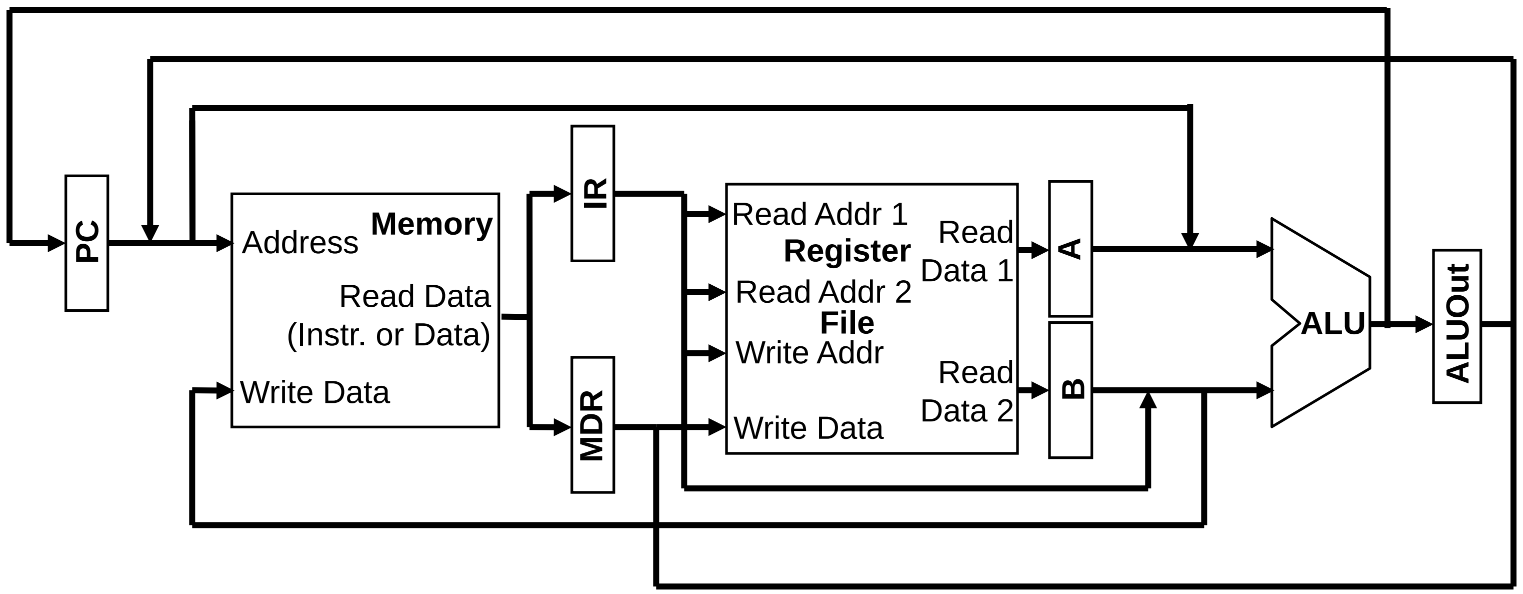


- Multi-cycle CPU
  - Requires state elements to hold intermediate values



Multi-cycle Approach

- Each cycle must
  - Store values needed in a later cycle of the current instruction in an internal register. All except IR hold data for one clock cycle.



IR – Instruction Register　　**MDR** – Memory Data Register

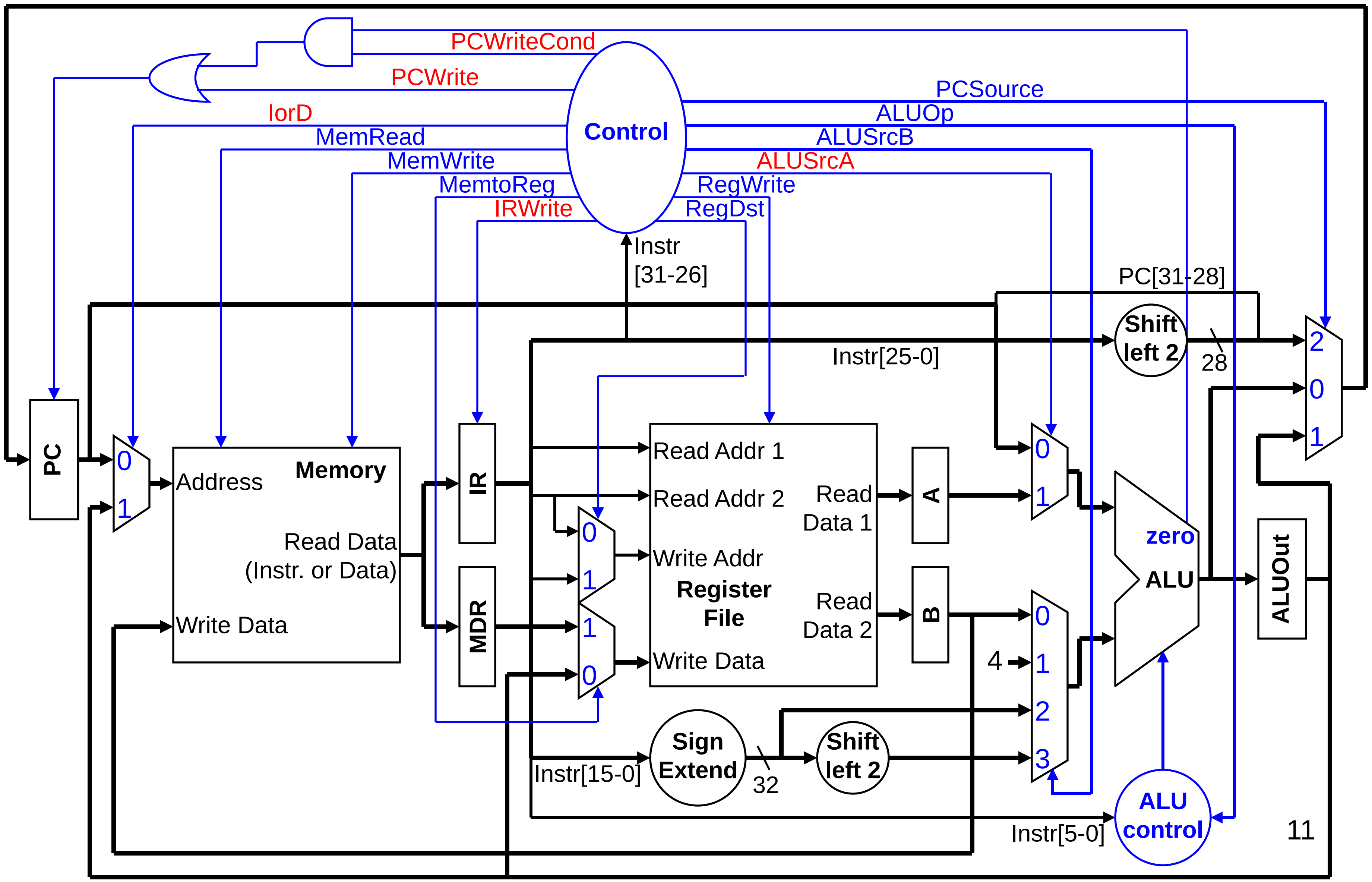
A, **B** – Register File data　　**ALUOut** – ALU Result Register

  - Store values needed by subsequent instructions in the register file or memory

Multi-cycle Control

- New control signals needed
  - PCWriteCond is set during a beq instruction
    - Formerly called Branch signal
  - PCWrite is set to write PC
    - Unconditional write signal needed during Fetch cycle
  - IorD controls what address is used for the memory
    - PC holds address for fetch cycle
    - ALUOut holds address for memory access instructions
  - IRWrite controls when the IR is written
  - ALUSrcA control one input to ALU
    - rs register for most operations
    - PC for branch instructions
    - Old ALUSrc renamed ALUSrcB and expanded

Multi-cycle Control and Datapath



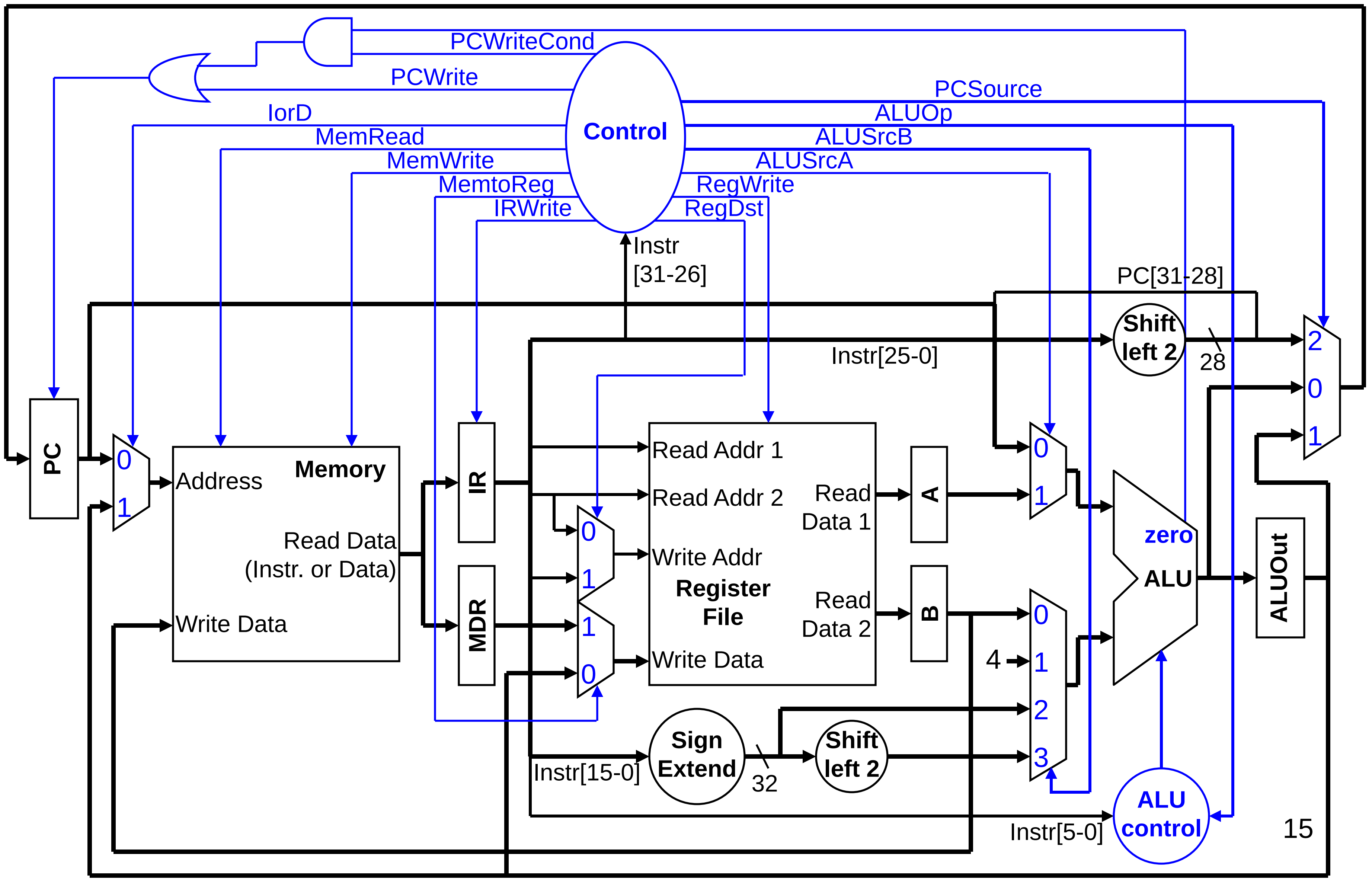
Multi-cycle Steps

- Instruction Fetch
- Decode and Register Fetch
- Execution
- Memory Access
- Write Register File

(1) Instruction Fetch Cycle

- Read instruction from memory
  - IR = M[PC]
- Increment PC using ALU
  - PC = PC + 4
- Control signals must
  - Select memory address source
  - Enable memory reading
  - Enable PC and IR write
  - Select PC source
  - Select ALU input as PC and constant 4
  - Select ALU operation (addition)

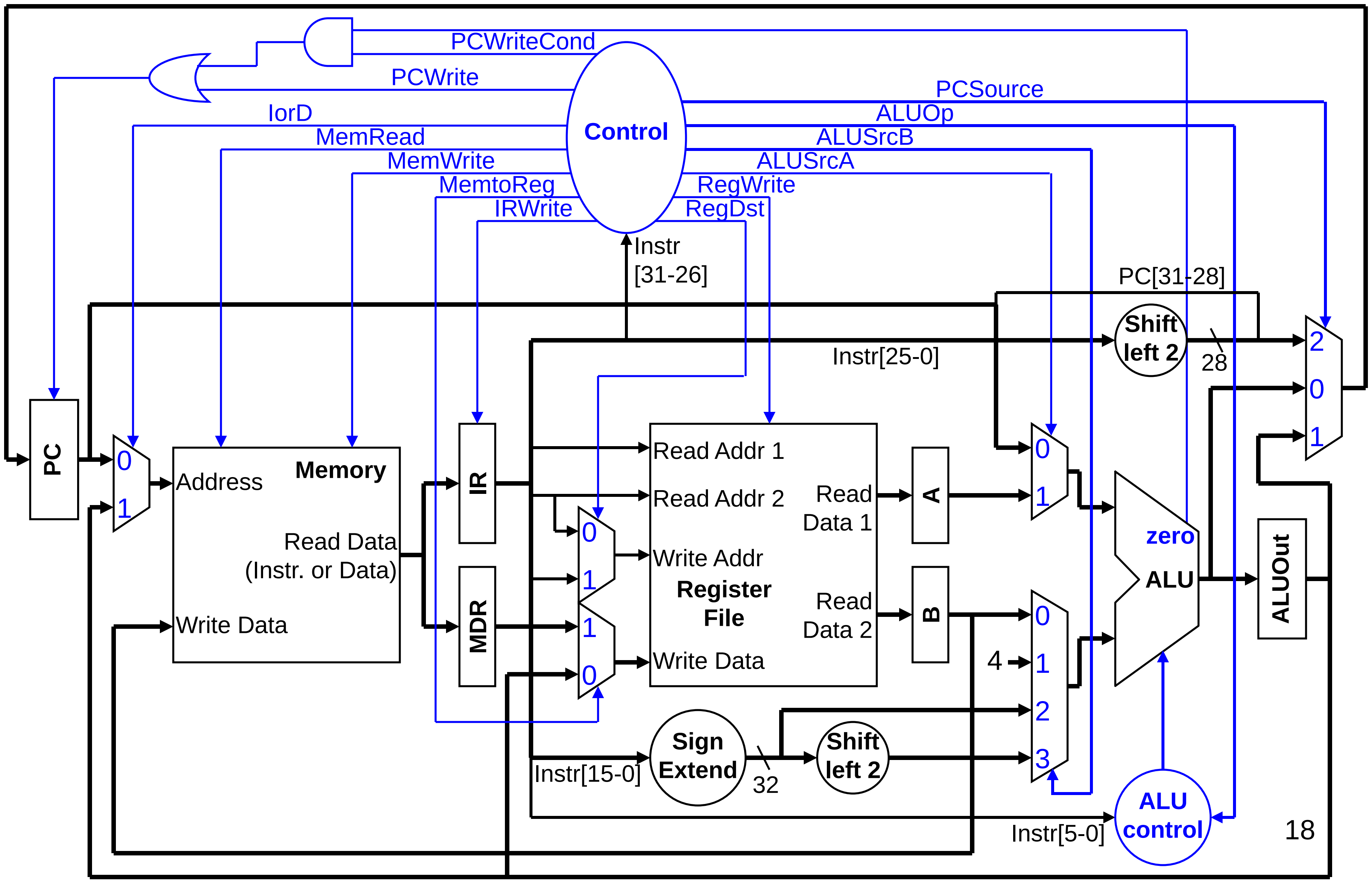
Instruction Fetch



(2) Decode and Register Fetch Cycle

- Read register values
  - A = R[rs], B = R[rt]
- Compute branch destination
  - ALUOut = PC + sign extended immediate value
- Prepare for next step based on instruction
- Control signals must
  - Select ALU inputs as PC and immediate value
  - Select ALU operation (addition)

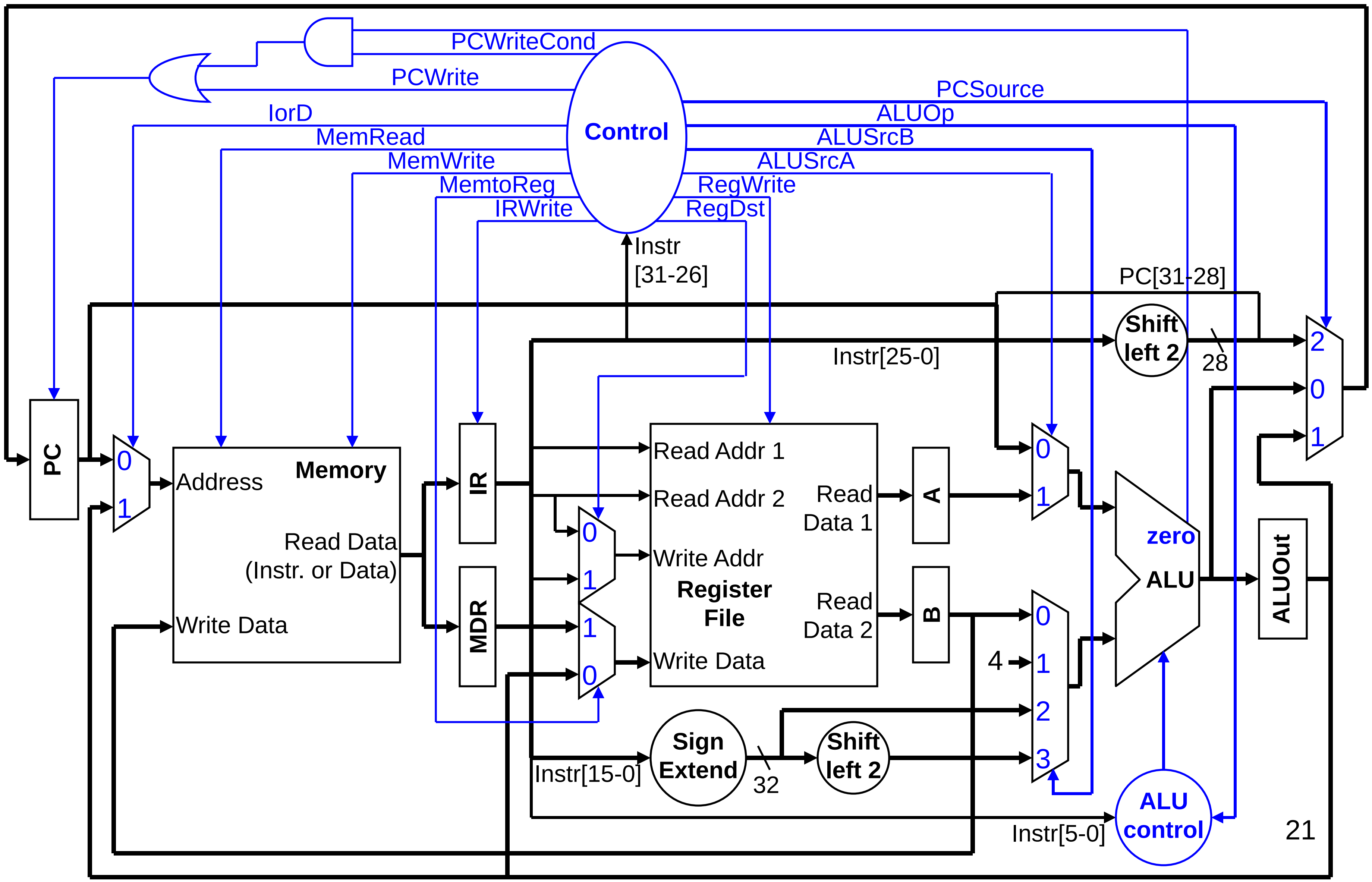
Decode and Register Fetch



(3) Execution Cycle

- Functionality varies with instructions
  - Memory reference
    - Compute address
    - ALUOut = A + sign extended immediate
  - R-type
    - Compute operation
    - ALUOut = A op B
  - Branch
    - Store new PC if needed
    - PC = ALUOut
    - ALUOut contains branch destination from previous cycle
- Control signals will depend on instruction type
  - Mem/R-type: Select ALU input and operation
  - Branch: Select PC source and set PC write control signal if needed

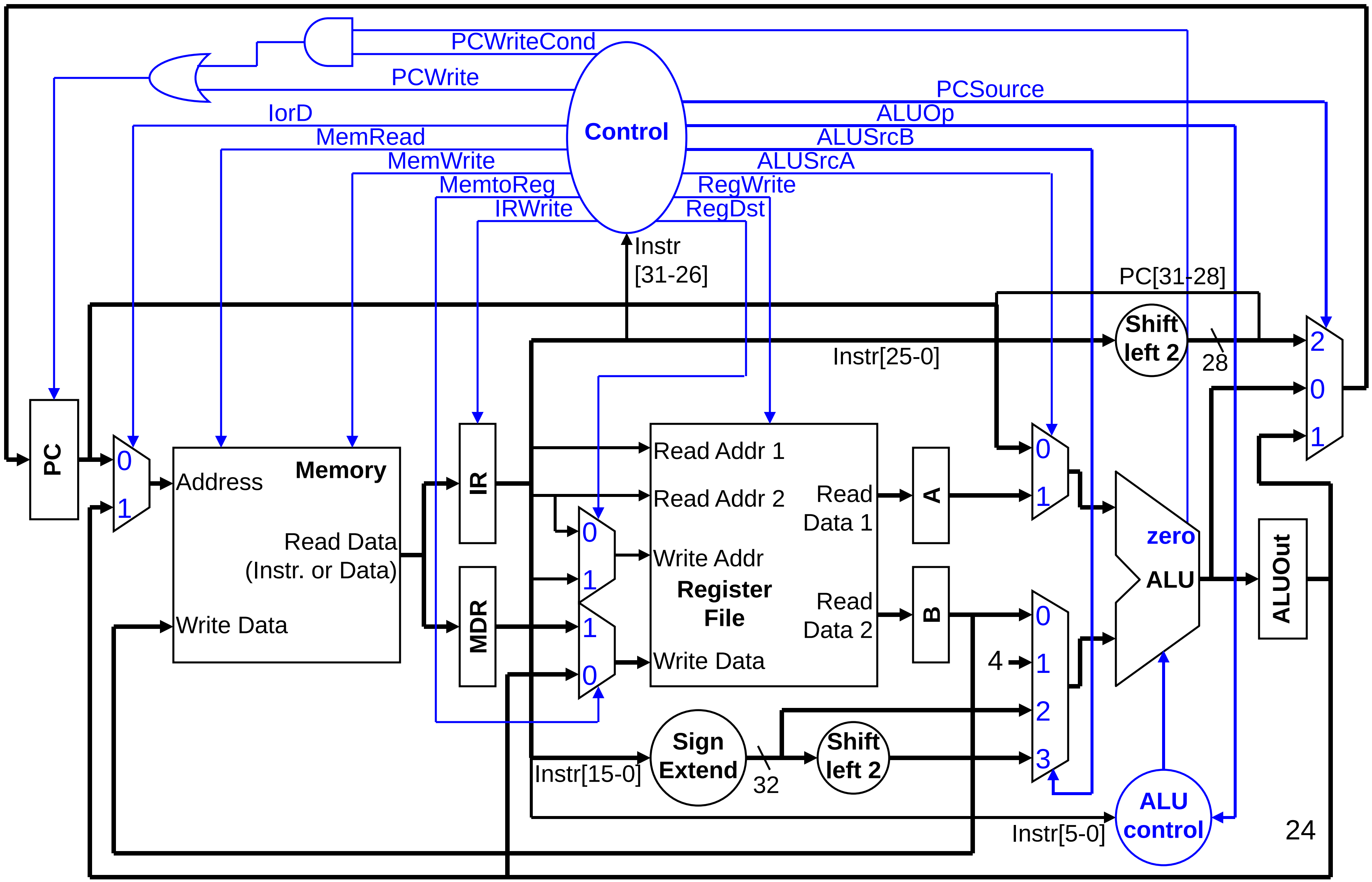
Execute Branch



(4) Memory Access Cycle

- Functionality varies with instructions
  - Memory reference
    - Read memory (lw) or write memory (sw)
    - MDR = M[ALUOut] or M[ALUOut] = B
  - R-type
    - Write result to register file
    - R[rd] = ALUOut
- Control signals will depend on instruction type
  - Memory reference
    - Enable memory read or write
    - Select memory address
  - R-type
    - Select register file write address and data
    - Enable register file write

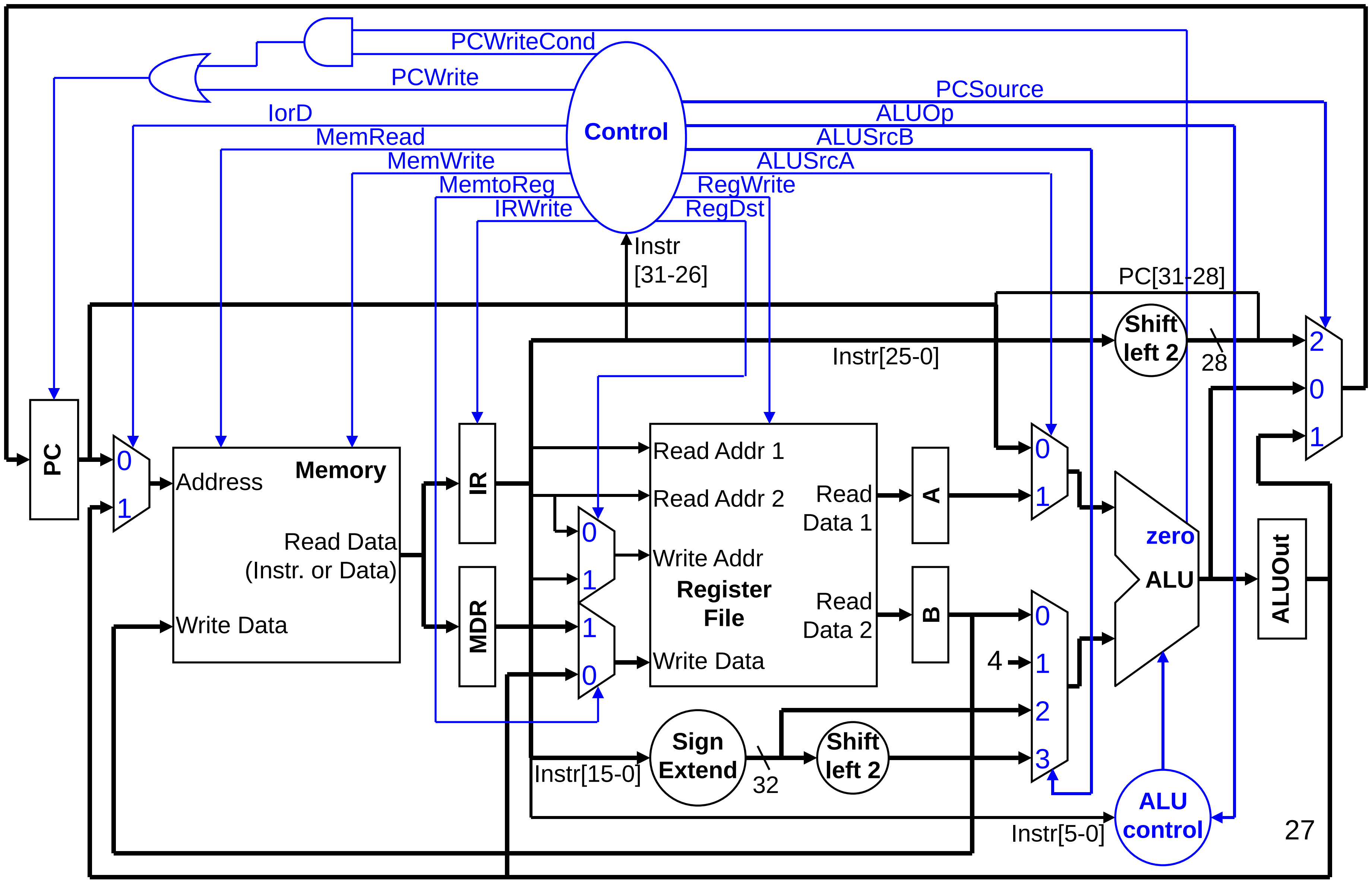
R-Type “Memory Access”



(5) Write Register File Cycle

- Only used by load instructions
- Write memory value to register
  - Reg[rt] = MDR
- Control signals must
  - Enable register file write
  - Select register file write address and data

lw Write Registers


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Control
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
Instr
[31-26]
PC[31-28]
Shift
left 2
28
Instr[25-0]
PC
0
1
Address
Memory
Read Data
(Instr. or Data)
Write Data
IR
MDR
Read Addr 1
Read Addr 2
Write Addr
Register
File
Write Data
Read
Data 1
Read
Data 2
A
B
zero
ALU
ALUOut
4
2
3
Sign
Extend
32
Shift
left 2
Instr[15-0]
Instr[5-0]
ALU
control