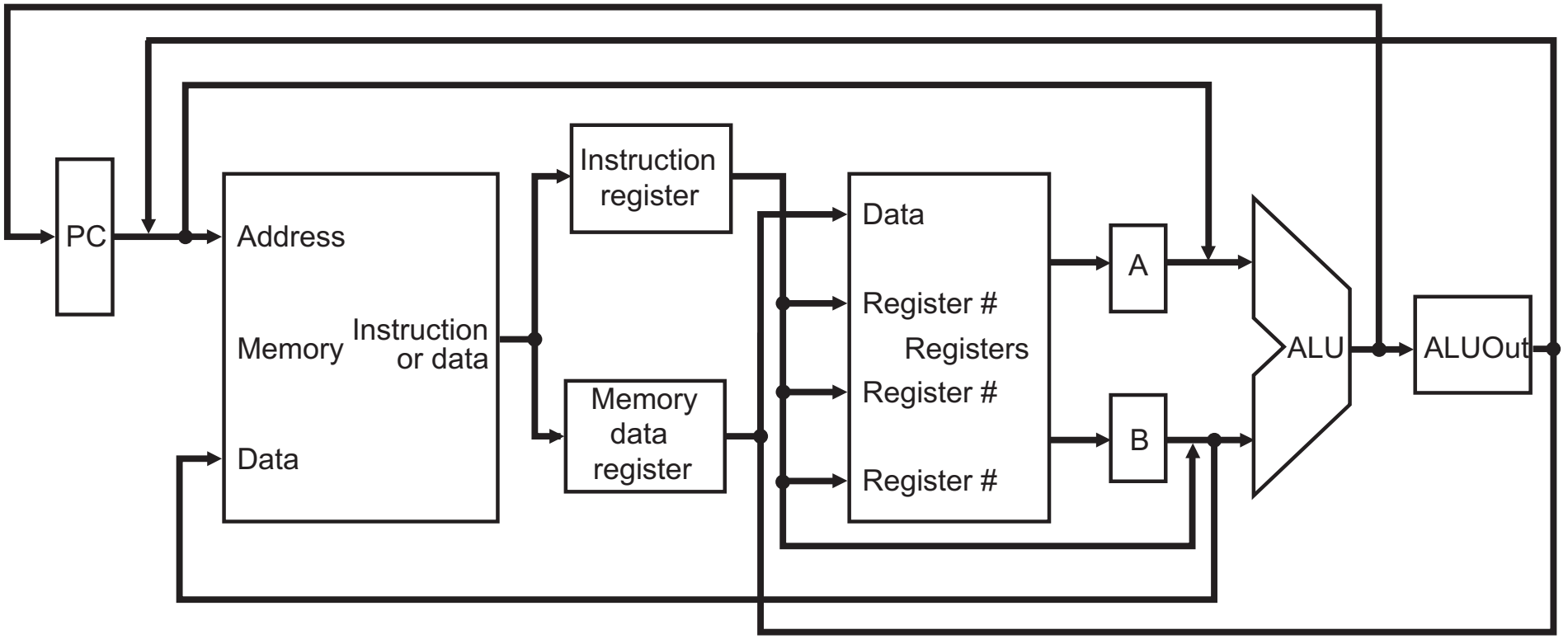


Overview of the *multi–cycle* Architecture Datapath

Multi-cycle Architecture:

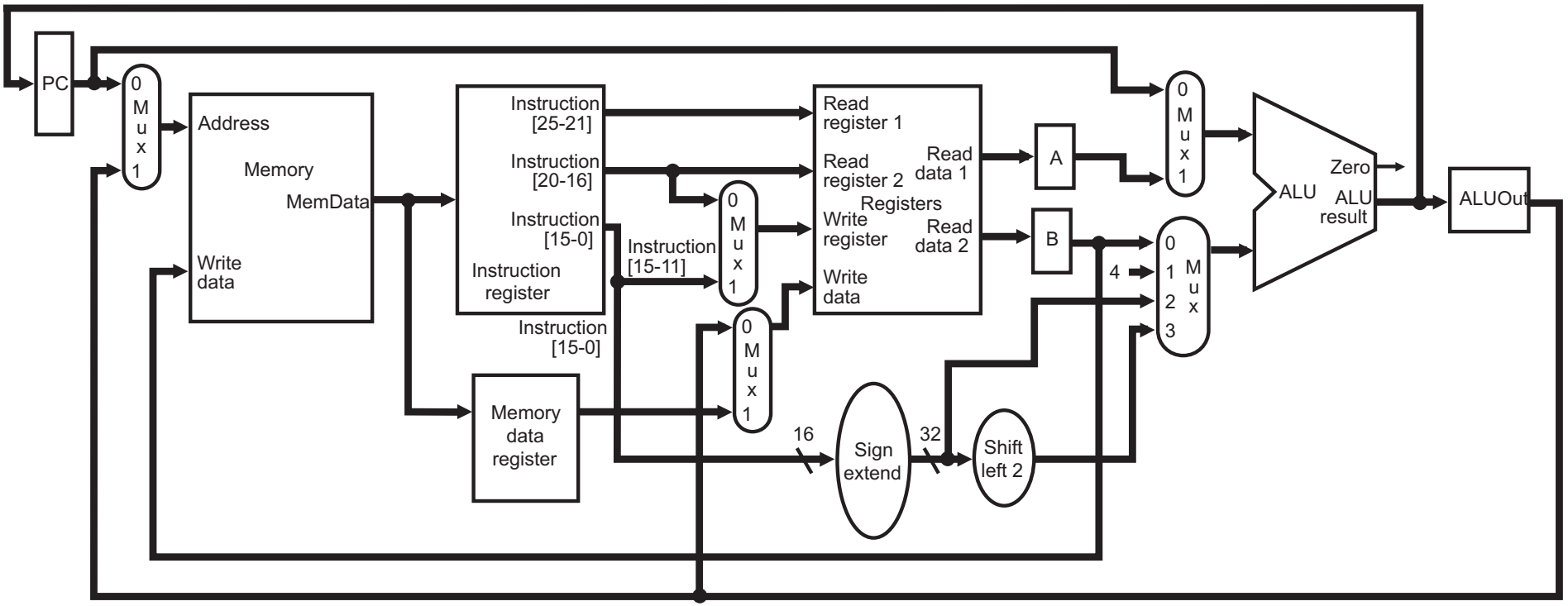
- each instruction is executed in several clock cycles
- instruction execution time (machine cycle) is variable
- instructions can store intermediate results (in intermediate registers) to be used in next stages of execution of this instruction
- the final instruction results are stored either in register file, memory or PC
- reduction of dedicated architecture elements in favor of intermediate registers

Hardware Blocks:

- common memory unit for both program and data
- register file
- single general-purpose ALU
- intermediate registers at output of each hardware block

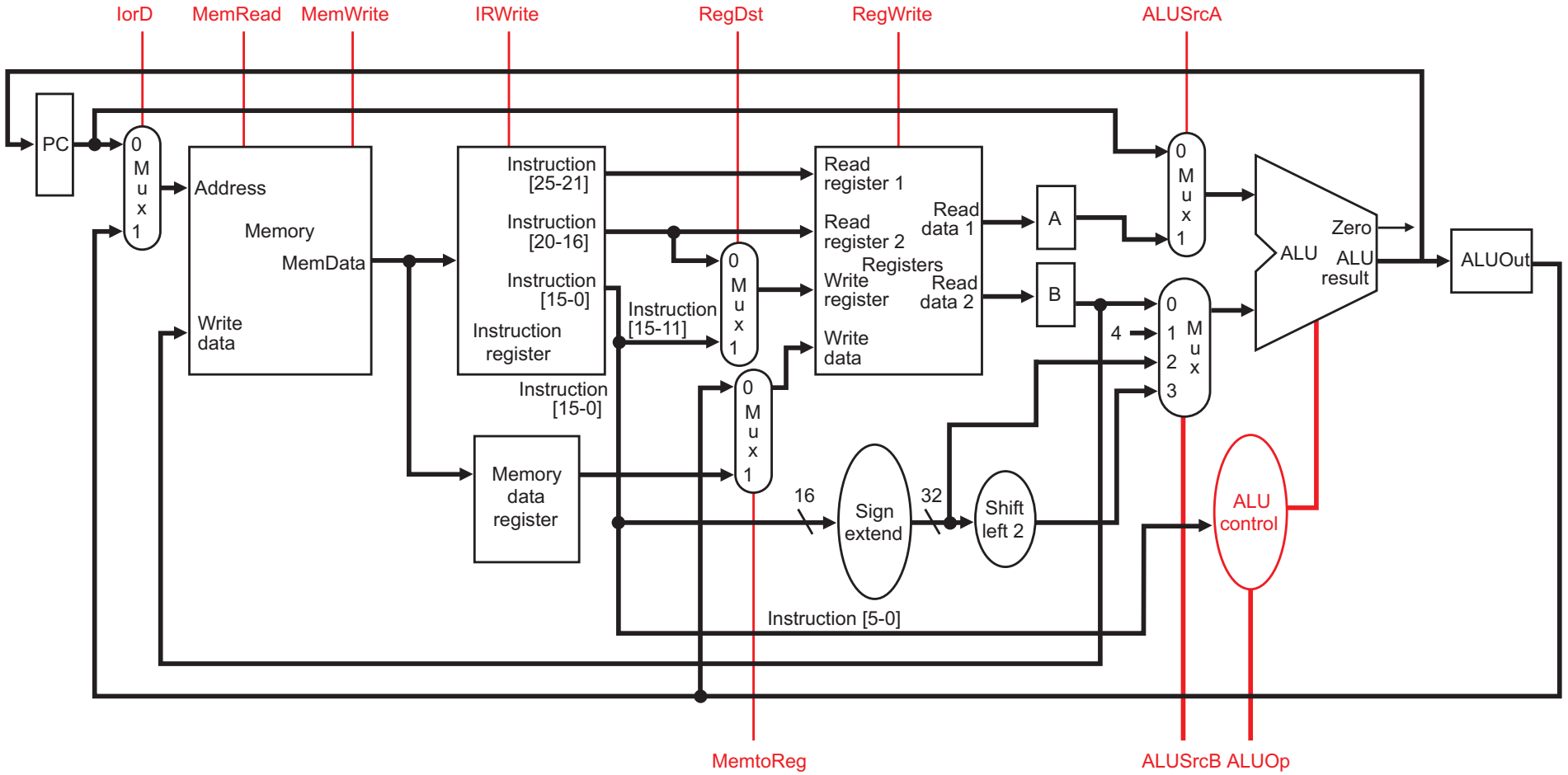
Intermediate Registers (not accessible directly)

IR — instruction register — stores instr. code during whole machine cycle
MDR — memory data register — stores the data read from the memory
A,B — store the data read from the register file
ALUOut — stores the result of ALU operation

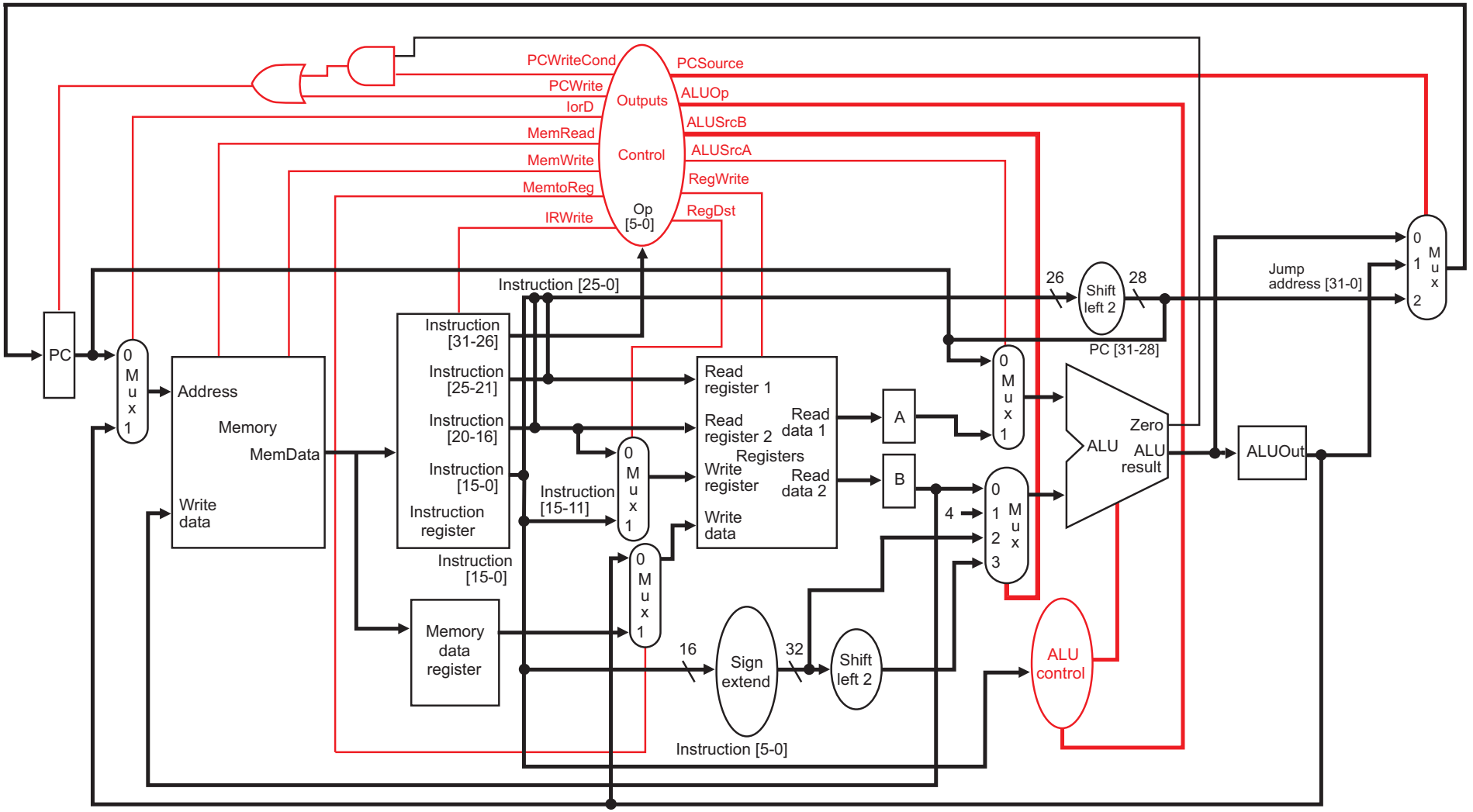


Additional multiplexers are required to make the hardware (ALU) available to all instruction operations (R-type operations, PC increment, memory address calculation)

Multi–cycle Architecture Datapath with Hardware Sharing



Multi–cycle Control Signals



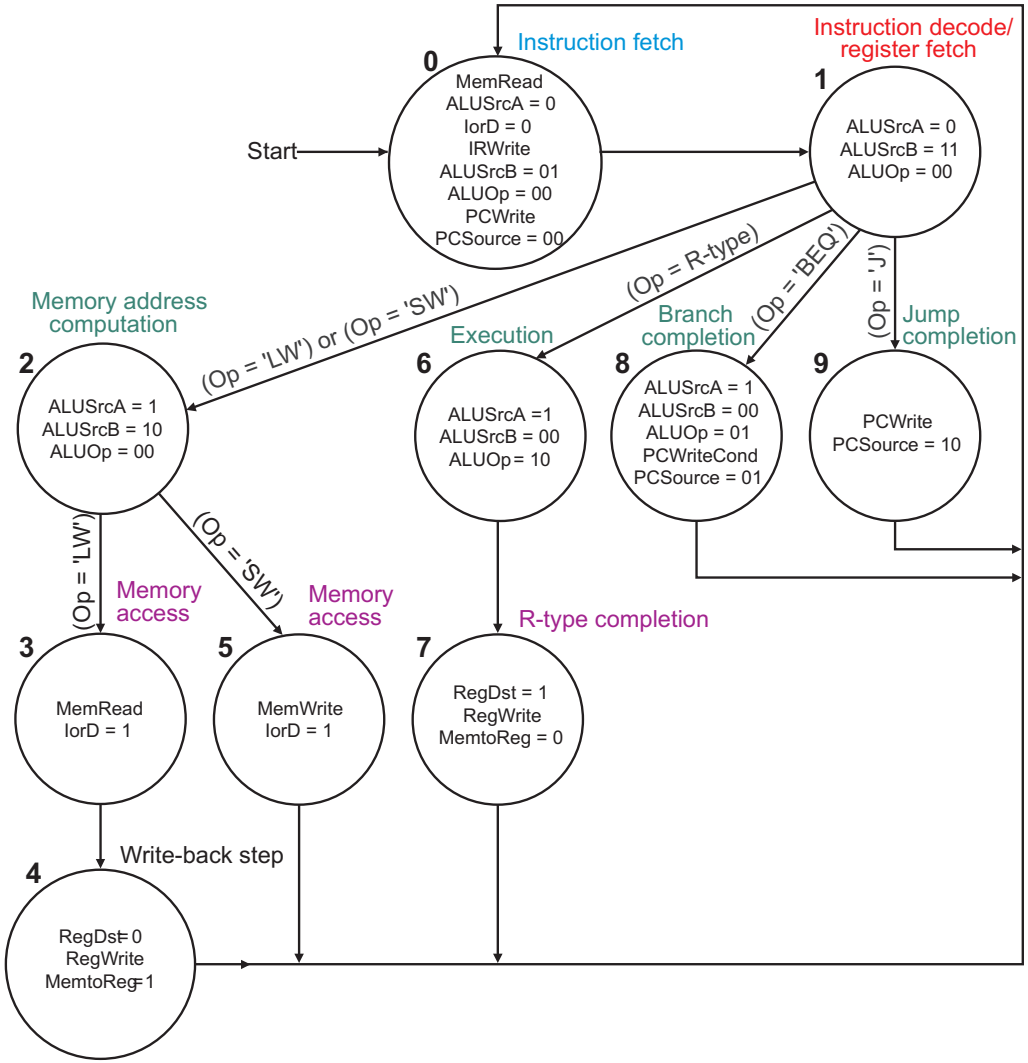
Complete *Multi–cycle* Architecture

Signal	0	1
RegDst	reg. loaded from memory	reg. modified by R-type
RegWrite	—	allow to modify the register file
ALUSrcA	PC to ALU	A to ALU
MemRead	—	allow to read the memory
MemWrite	—	allow to write the memory
MemToReg	ALUOut is to be stored in the register file	MDR is to be stored in the register file
IorD	PC addressing the memory	ALUOut addressing the memory
IRWrite	—	allow to write the IR
PCWrite	—	allow to write the PC
PCWriteCond	—	allow conditionally write the PC

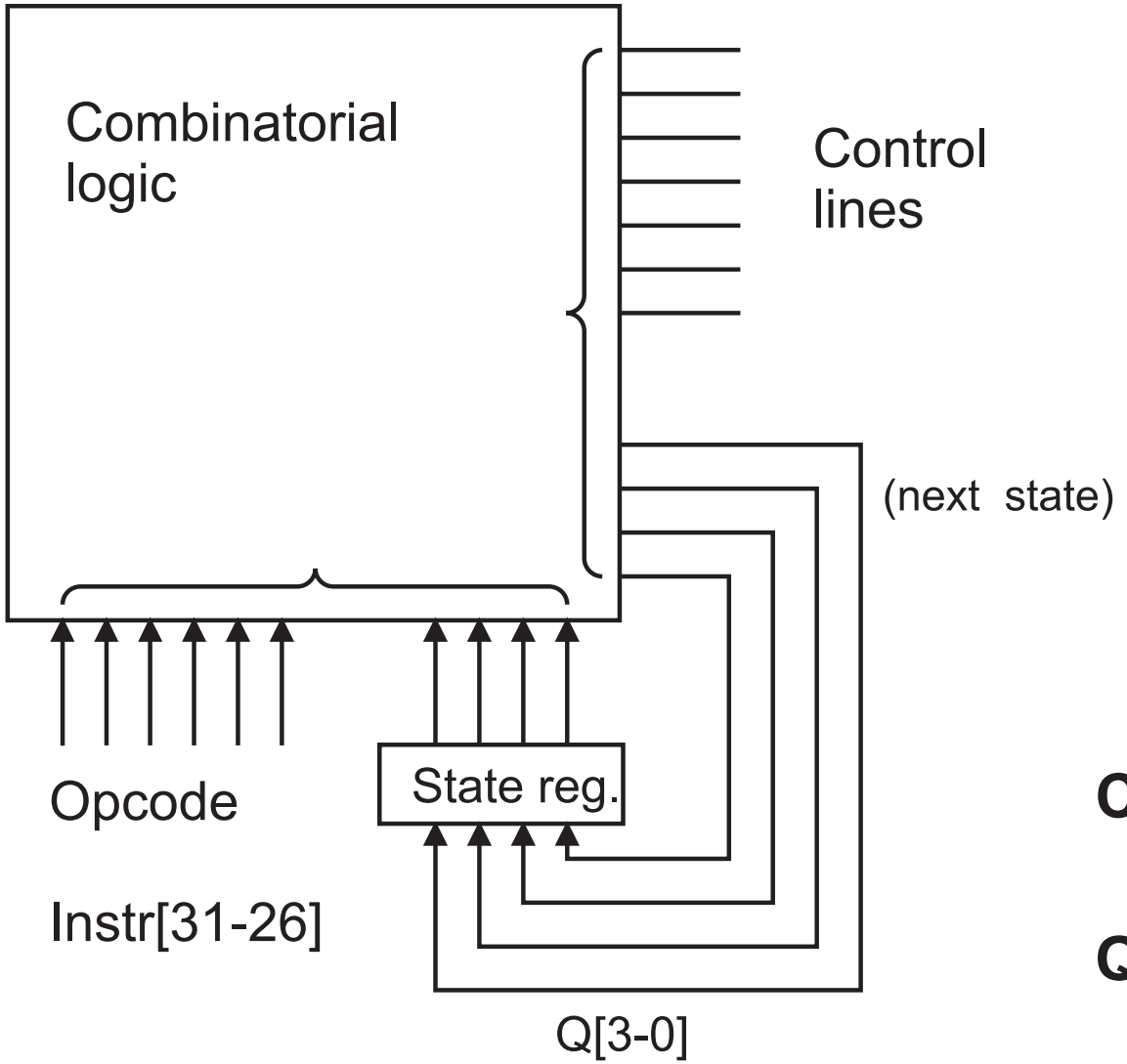
1-bit Control Signals

Signal	Value	Operation
ALUOp	00 01 10	add (load-store) subtract (branch) depending on function field (R-type)
ALUSrcB	00 01 10 11	B (R-type) to ALU 4 to ALU instruction[15-0] (load-store) to ALU instruction[15-0]¡¡2 (branch) to ALU
PCSource	00 01 10	ALU (PC+4) to PC ALUOut (branch) to PC PC[31-26]+IR[25-0]¡¡2 (jump) to PC

2-bit Control Signals



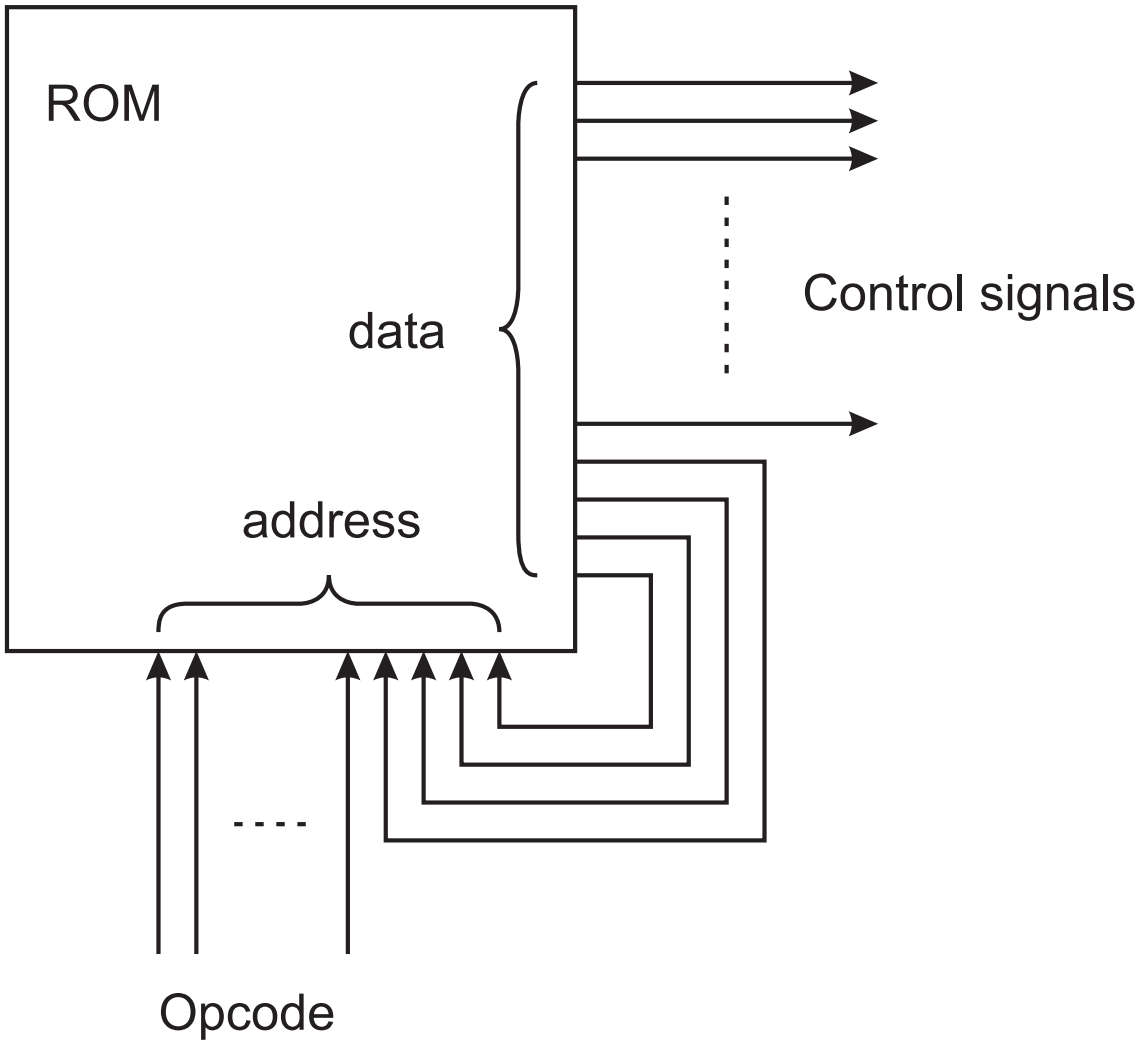
state Diagram of *Multi–cycle* Control Block



Control = F(Q(t), Opcode)

Q(t+1) = F(Q(t), Opcode)

State Machine of the *Multi–cycle* Control Block



ROM-based Hardware Implementation of *multi–cycle* Control Block