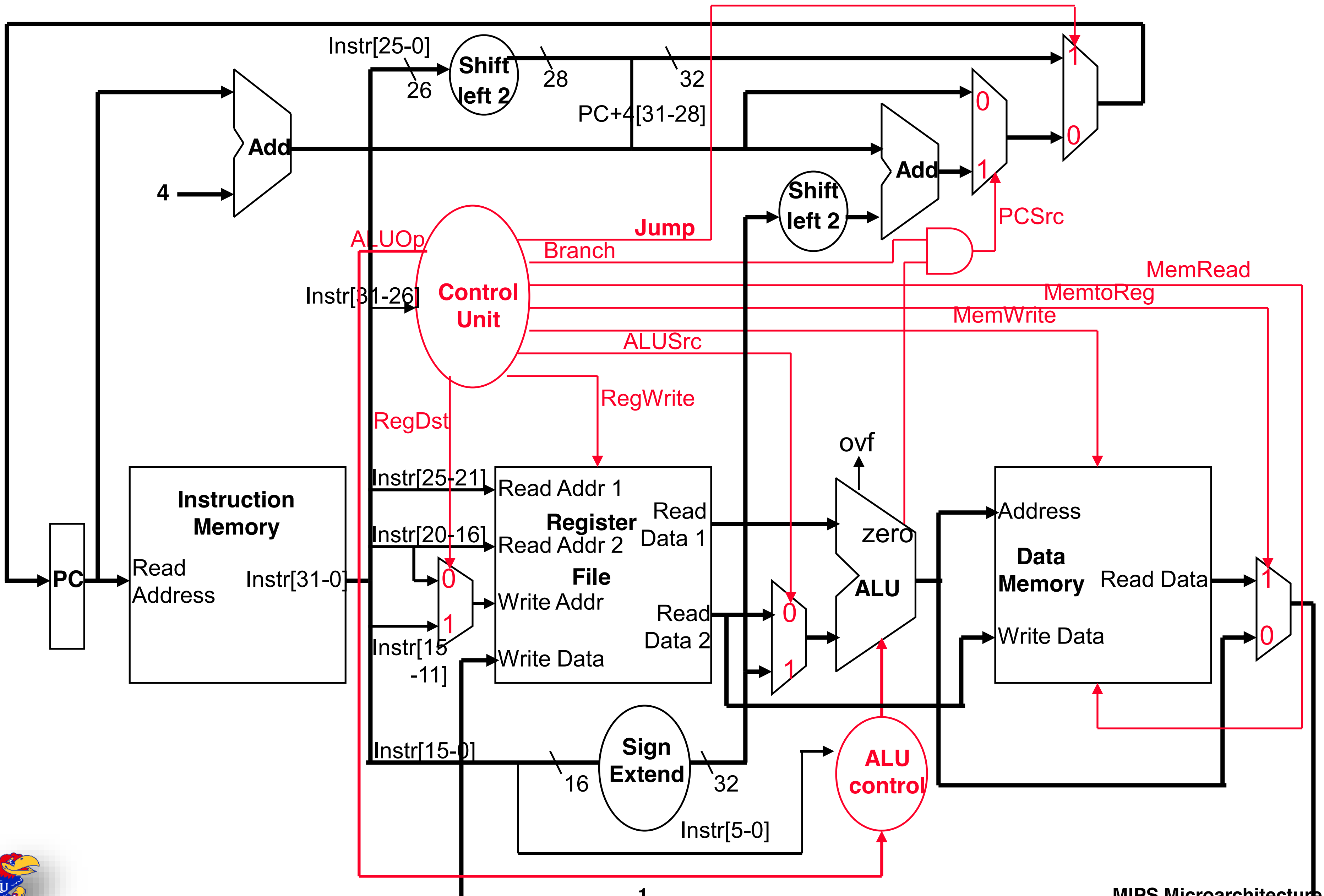
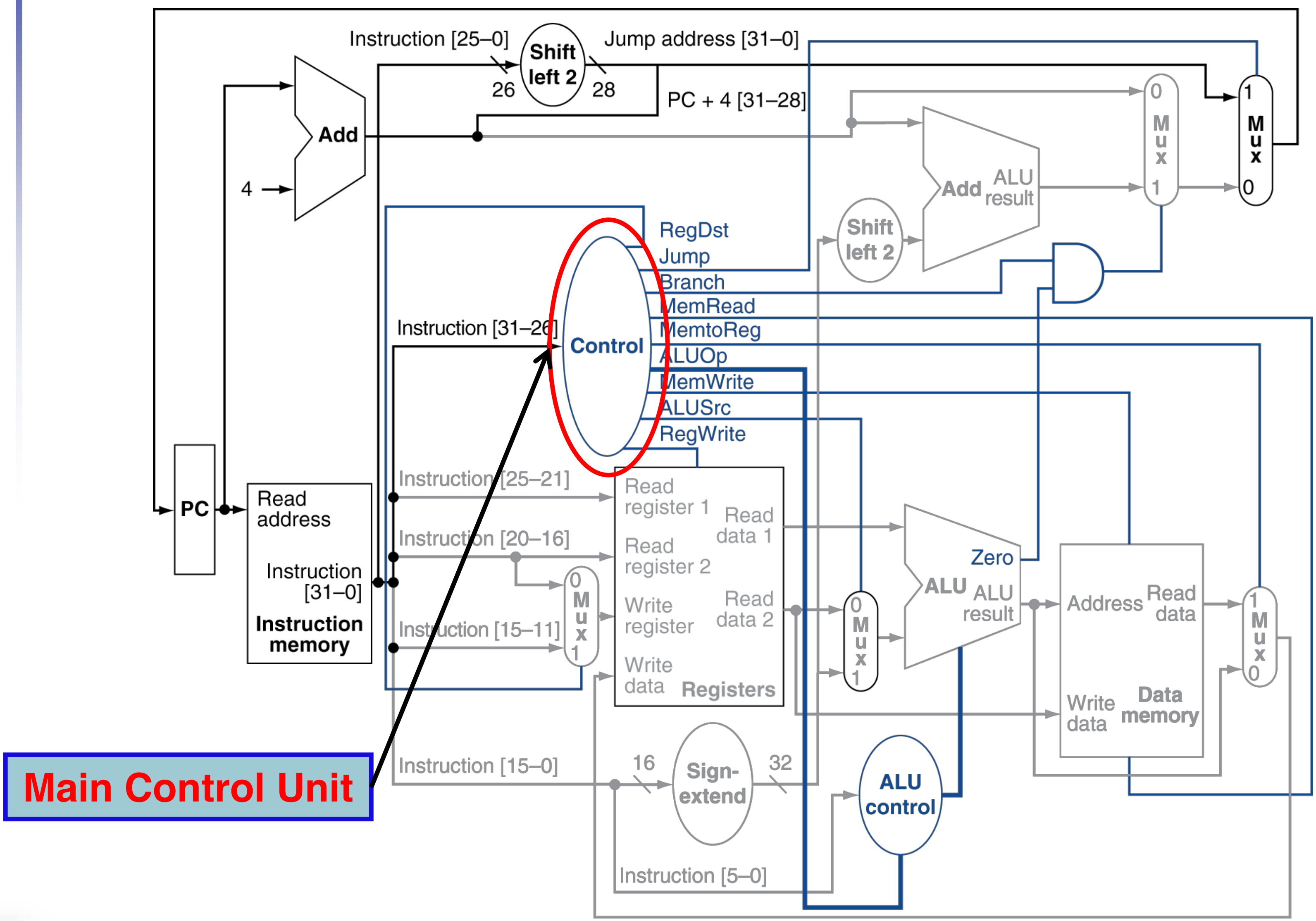
Complete Single-Cycle/Non-Pipelined Datapath



Single-Cycle/Non-Pipelined Datapath (Main Control Unit)



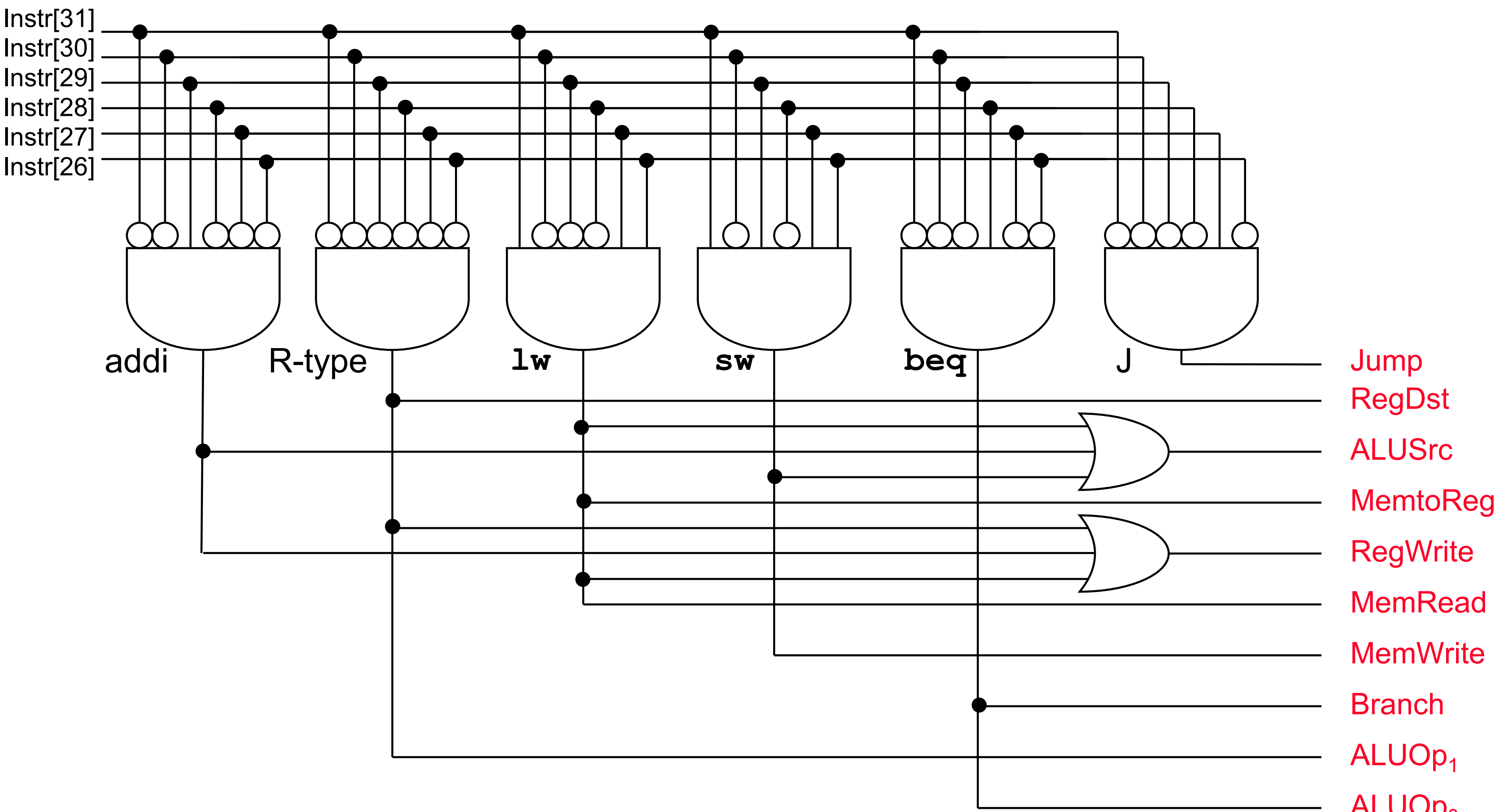
Main Control Unit

Instr. OP	RegDst	ALUSrc	MemToReg	RegWr	MemRd	MemWr	Branch	ALUOp	Jump
R-type 000000	1	0	0	1	0	0	0	10	0
lw 100011	0	1	1	1	1	0	0	00	0
sw 101011	X	1	X	0	0	1	0	00	0
beq 000100	X	0	X	0	0	0	1	01	0
j 000010	X	X	X	0	0	0	X	XX	1
addi 001000	0	1	0	1	0	0	0	00	0

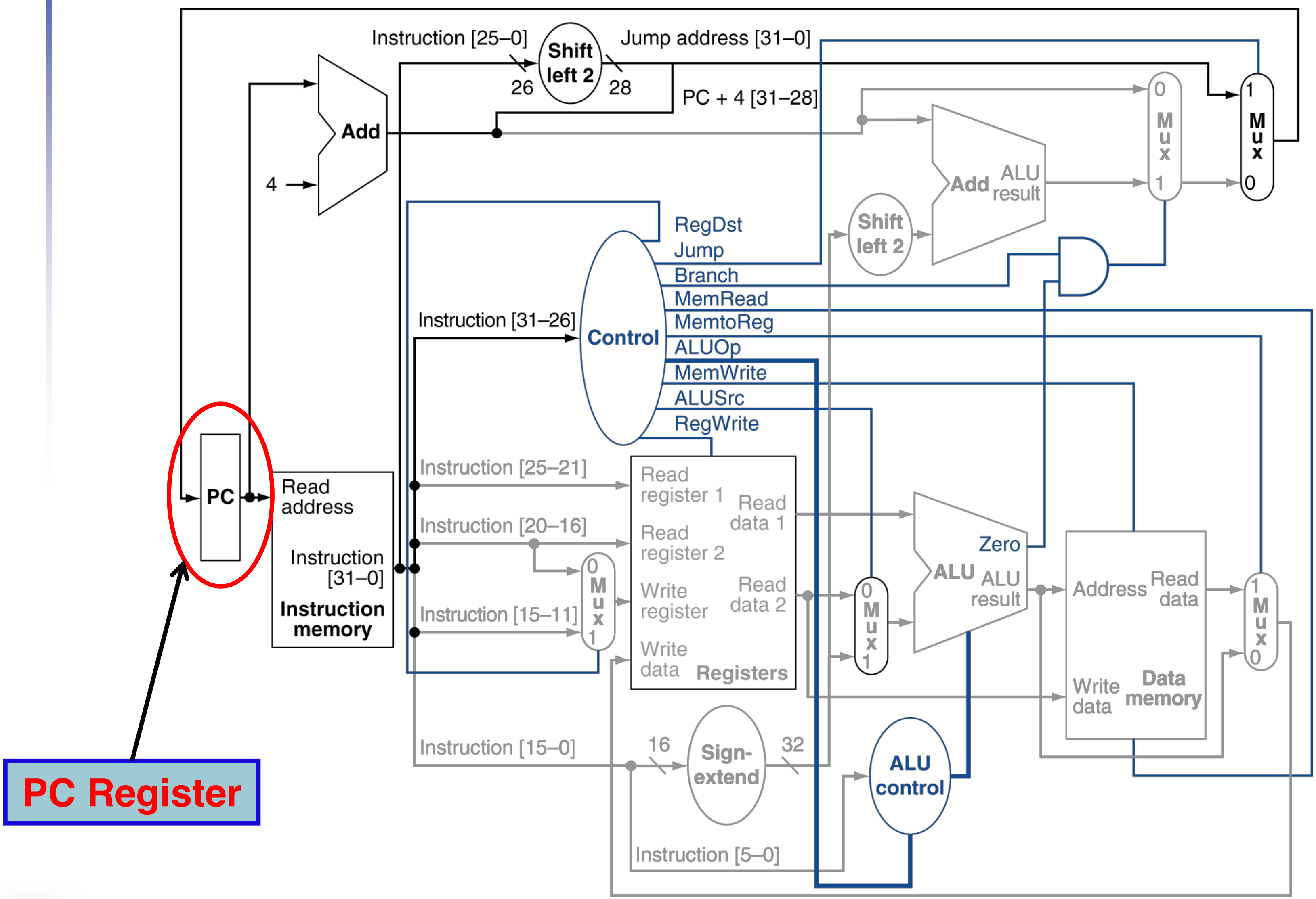
- Setting of the MemRd signal (for R-type, sw, beq, j) depends on the memory design (could have to be 0 or could be a X (don't care))

Main Control Unit

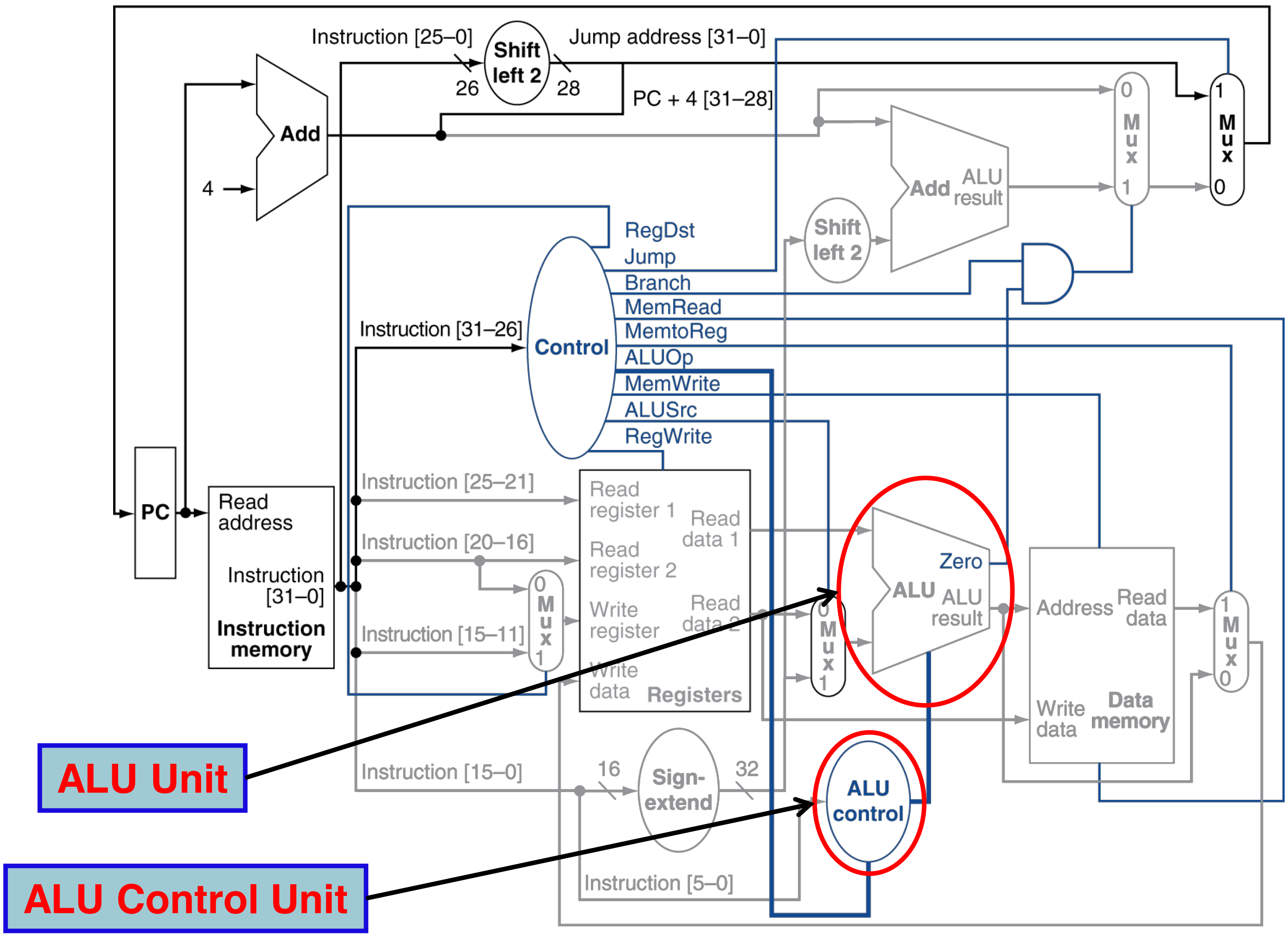
- From the truth table can design the Main Control logic



Single-Cycle/Non-Pipelined Datapath (PC Register)



Single-Cycle/Non-Pipelined Datapath (ALU Unit & ALU Control Unit)



ALU Unit & ALU Control Unit

- Assume 2-bit ALUOp derived from opcode
  - Combinational logic derives ALU control

opcode	rs	rt	rd	shamt	funct
31:26	25:21	20:16	15:11	10:6	5:0

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw ≡ 100011	00	load word	XXXXXX	add	0010
sw ≡ 101011	00	store word	XXXXXX	add	0010
beq ≡ 000100	01	branch equal	XXXXXX	subtract	0110
R-type ≡ 000000	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

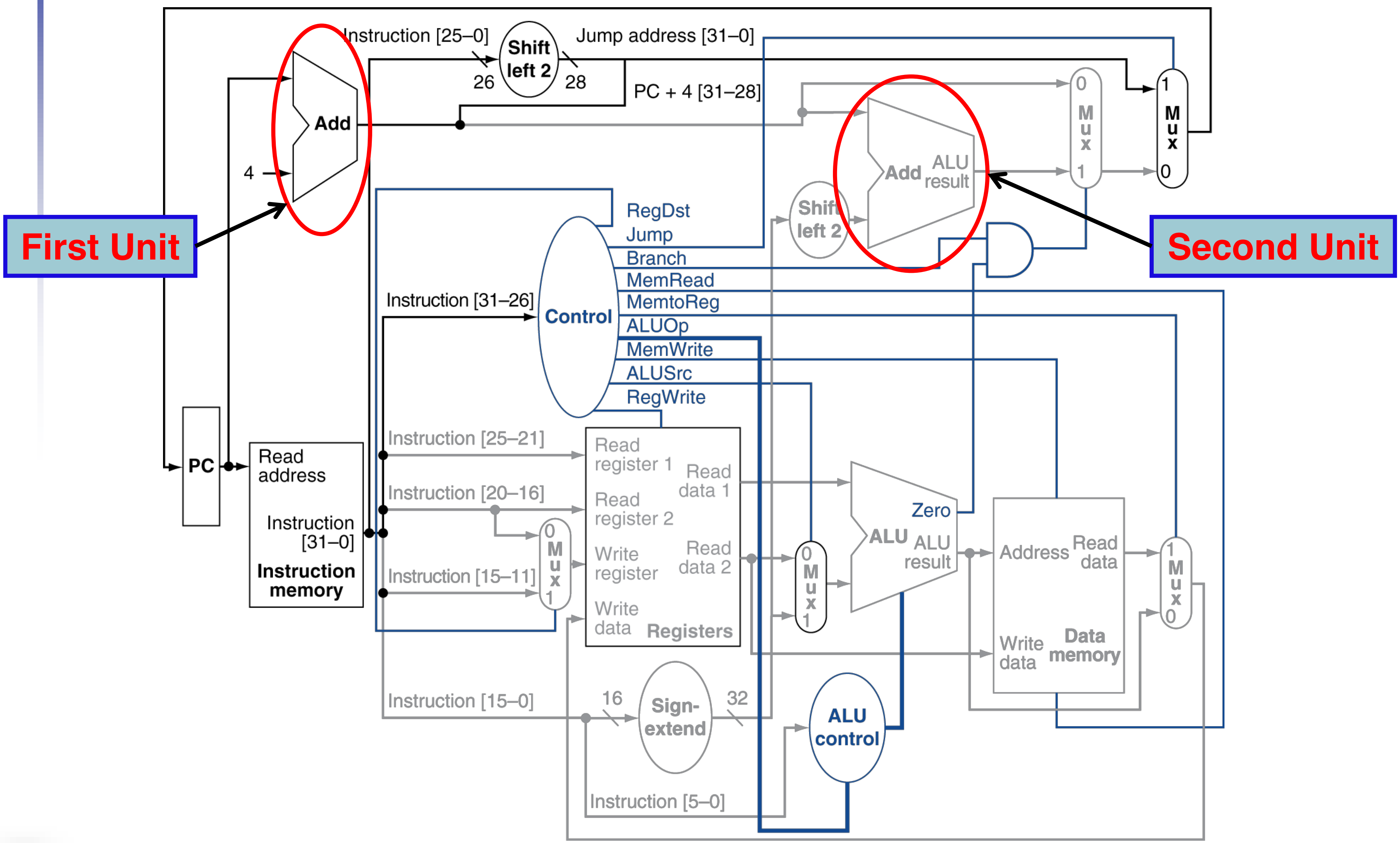
```
module MIPSALU (ALUctl, A, B, ALUOut, Zero);
   input [3:0] ALUctl;
   input [31:0] A,B;
   output reg [31:0] ALUOut;
   output Zero;
   assign Zero = (ALUOut==0); //Zero is true if ALUOut is 0
   always @(ALUctl, A, B) begin //reevaluate if these change
      case (ALUctl)
         0: ALUOut <= A & B;
         1: ALUOut <= A | B;
         2: ALUOut <= A + B;
         6: ALUOut <= A - B;
         7: ALUOut <= A < B ? 1 : 0;
         12: ALUOut <= ~(A | B); // result is nor
         default: ALUOut <= 0;
      endcase
    end
endmodule
```

FIGURE C.5.15 A Verilog behavioral definition of a MIPS ALU.

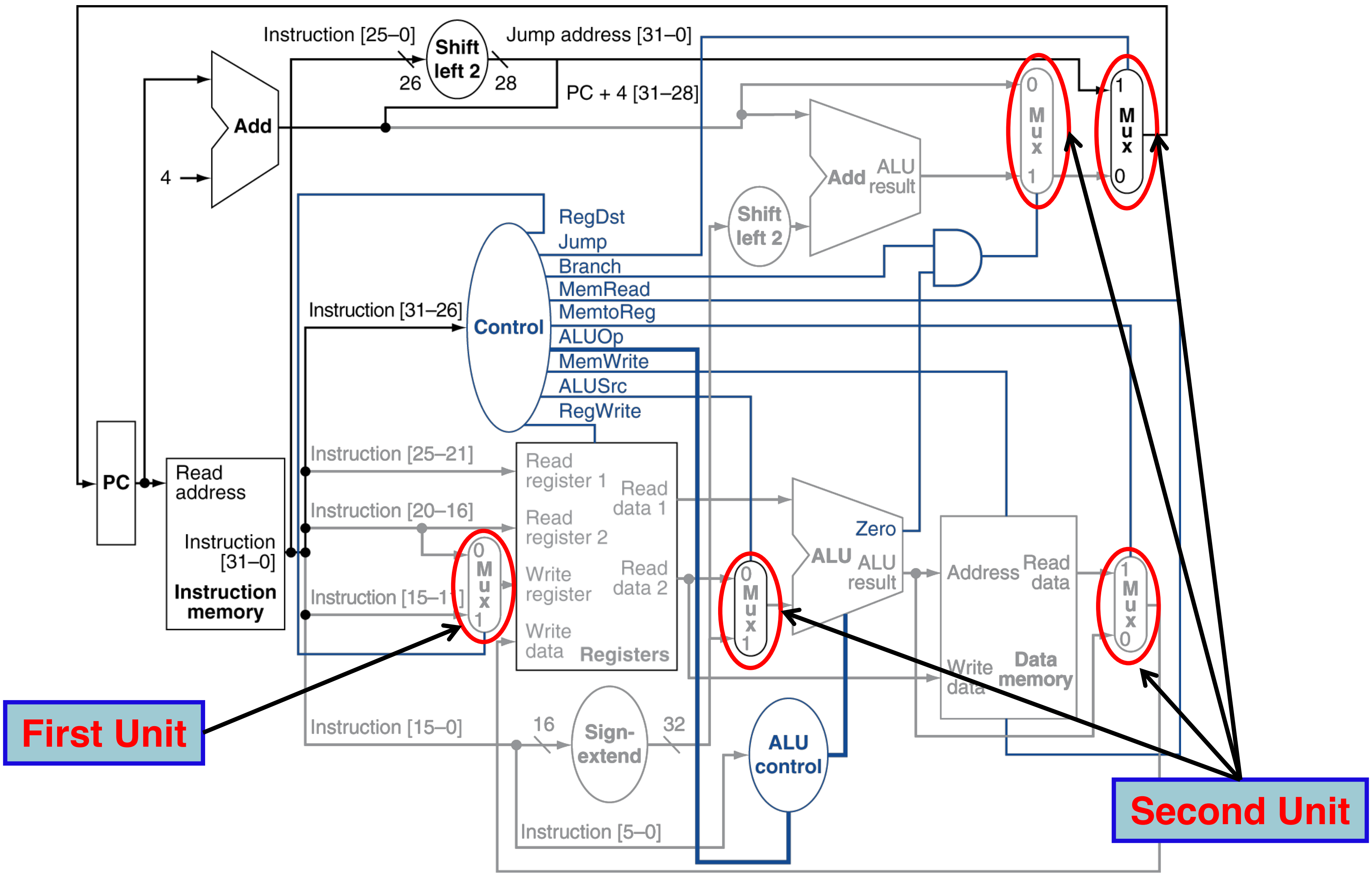
```
module ALUControl (ALUOp, FuncCode, ALUCtl);
   input [1:0] ALUOp;
   input [5:0] FuncCode;
   output [3:0] reg ALUCtl;
   always case (FuncCode)
      32: ALUCtl <=2; // add
      34: ALUCtl <=6; //subtract
      36: ALUCtl <=0; // and
      37: ALUCtl <=1; // or
      42: ALUCtl <=7; // slt
      default: ALUCtl <=15; // should not happen
   endcase
endmodule
```

FIGURE C.5.16 The MIPS ALU control: a simple piece of combinational control logic.

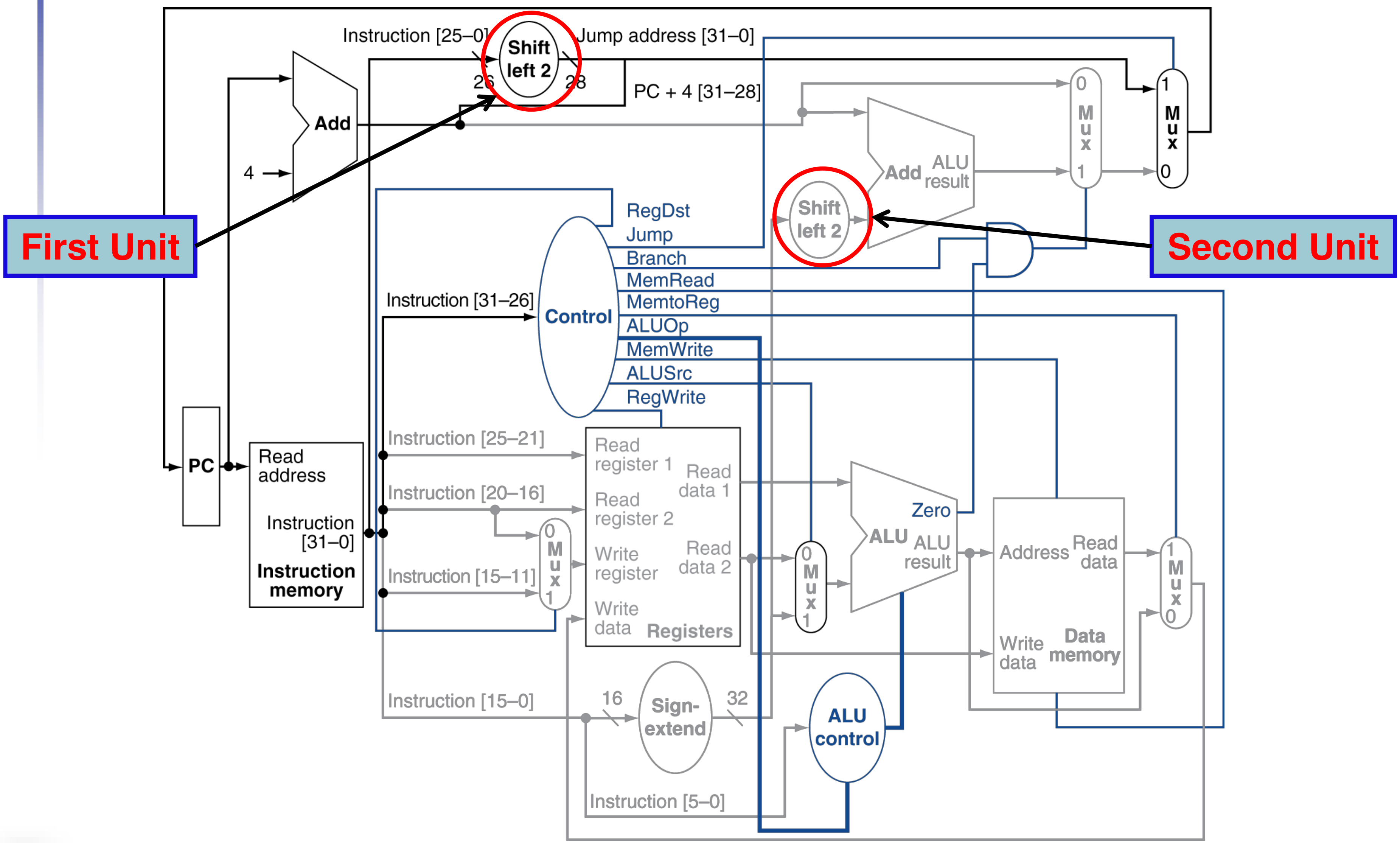
Single-Cycle/Non-Pipelined Datapath (Adder Units)



Single-Cycle/Non-Pipelined Datapath (Multiplexer Units)



Single-Cycle/Non-Pipelined Datapath (Shift-Left Units)



Single-Cycle/Non-Pipelined Datapath (Sign-Extend Unit)