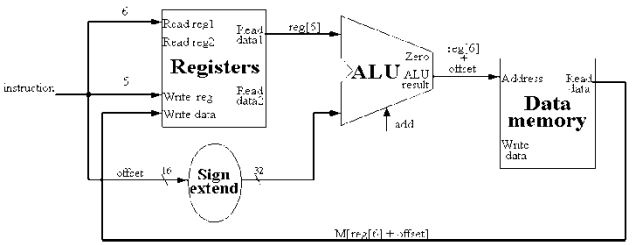
Datapath and control: 3-step derivation

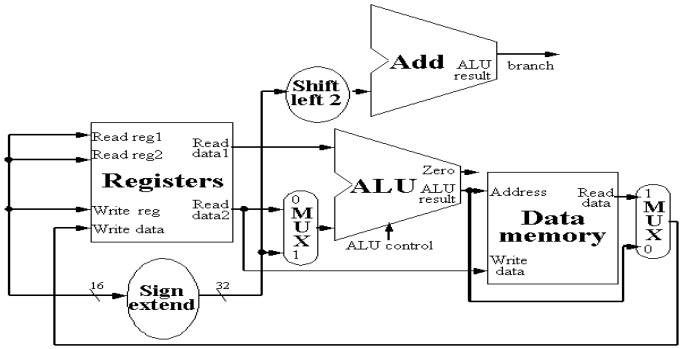
(3rd Ed: p.284-314, 4th Ed: p.300-330, 5th Ed: p.244-272)

- different datapaths for:
  - register-based instructions e.g. add, sub
  - memory-access instructions e.g. lw, sw
  - branch instructions e.g. beq, j

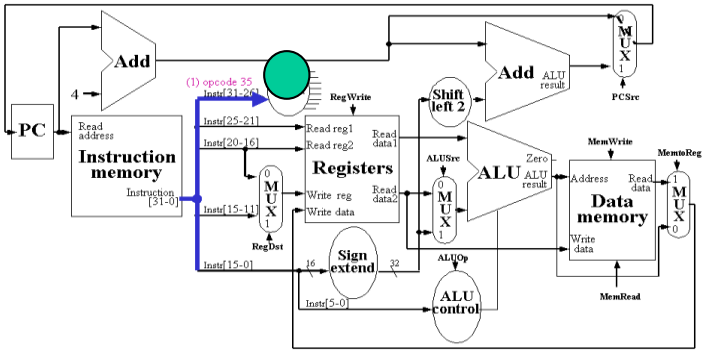


- combined datapath:

  for different instructions
  - add multiplexors



- control unit: activate relevant parts of the combined datapath for a given instruction
  - control signals for multiplexors + ALU

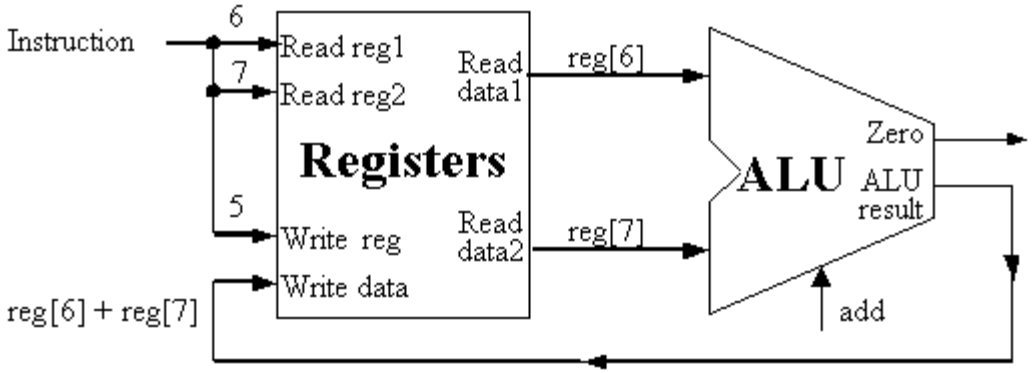


Register-based instructions

- select read/write registers from operands
- select ALU operation from opcode and function field

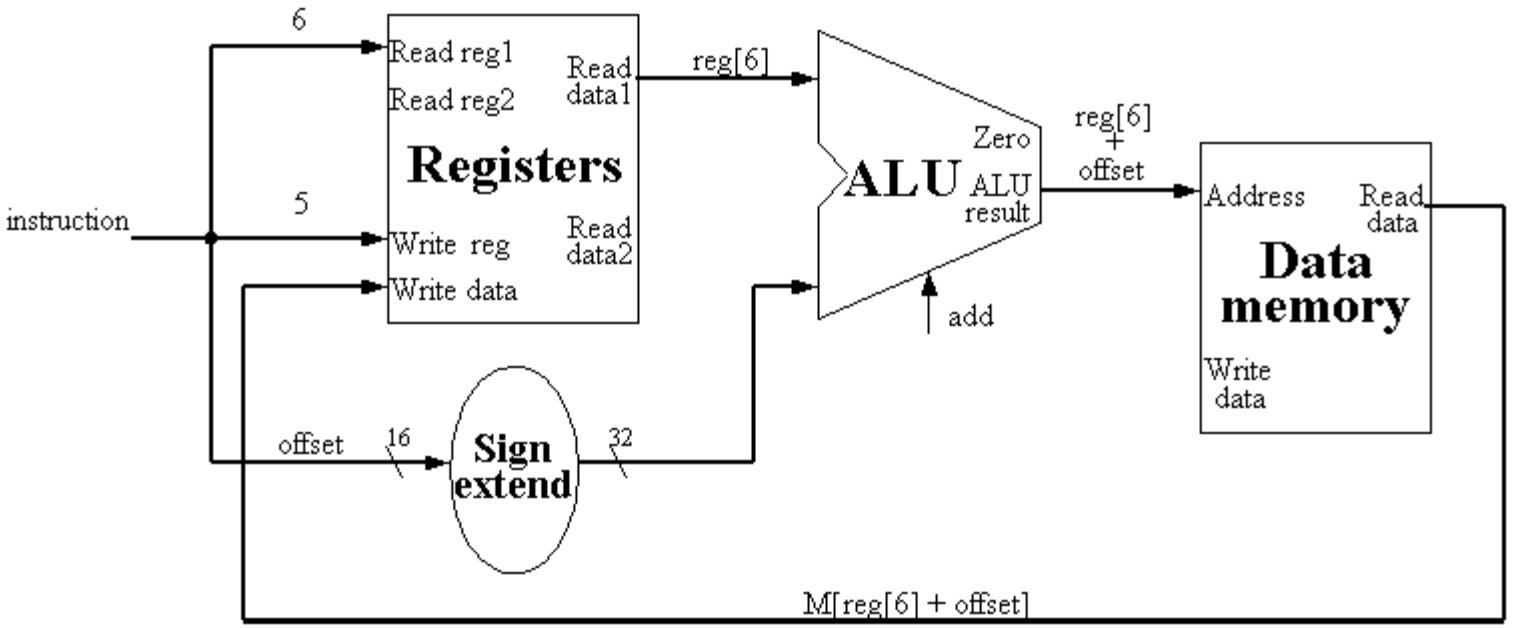
opcode	source 1	source 2	dest.	shift amt	fn code

- add $5, $6, $7 # reg[5] = reg[6] + reg[7]



Memory access instructions: load

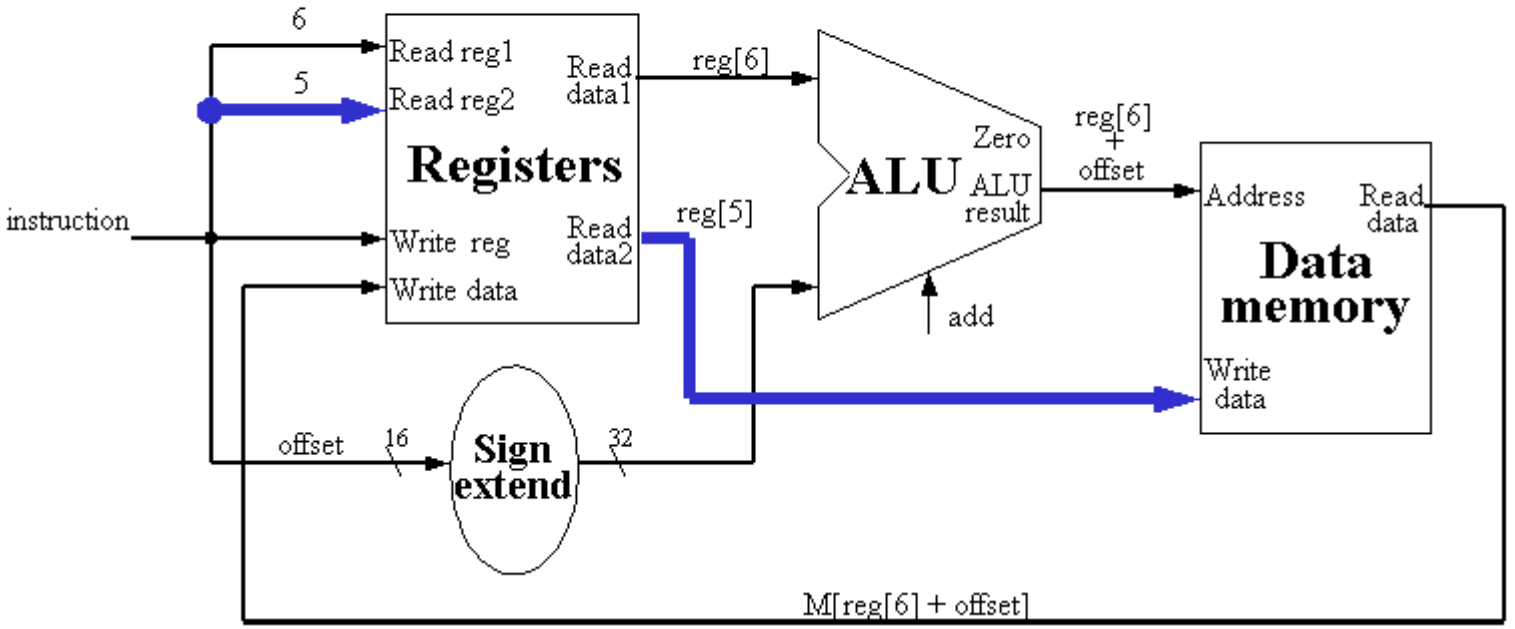
- lw $5, offset($6) # reg[5] = M[reg[6] + offset]
  assume data memory to return value in same cycle



Memory access instructions: store

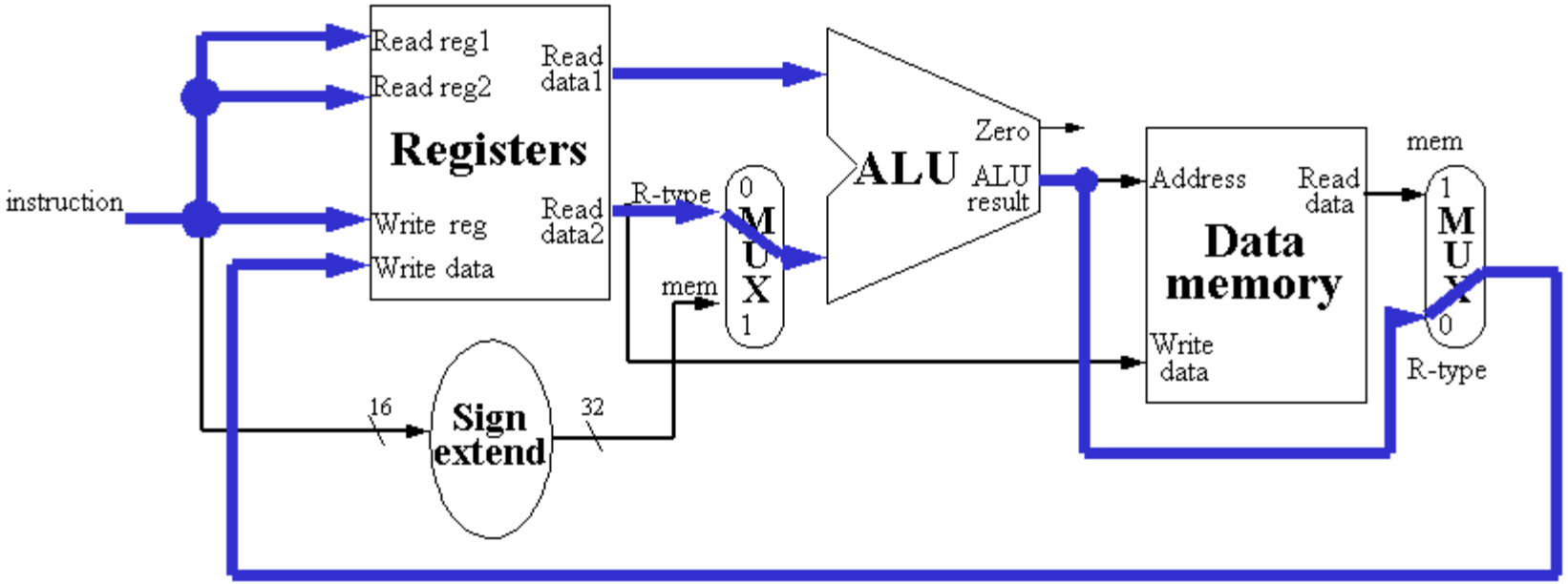
- sw $5, offset($6) # M[reg[6] + offset] = reg[5]

- connections for store



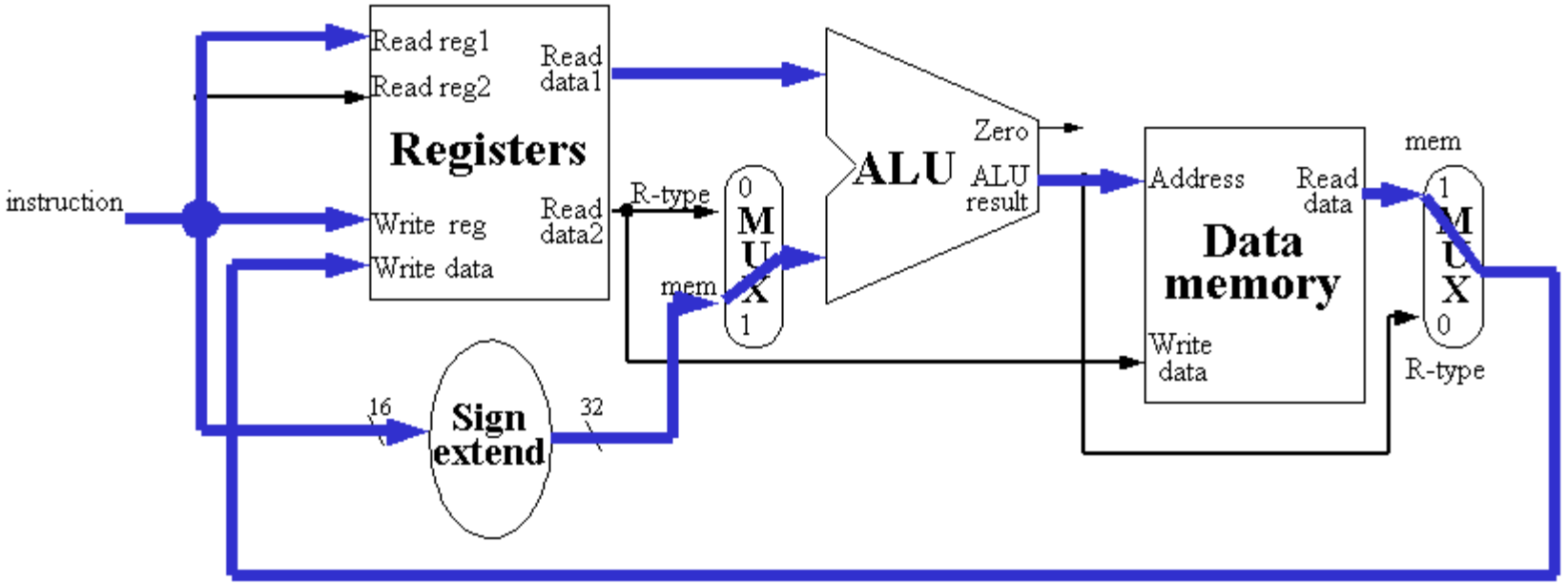
Combine datapaths for R-type and memory instructions: R-type flow

- 1 register file, 1 ALU
- ALU input mux: data from register or instruction
- register write mux: data from ALU or memory



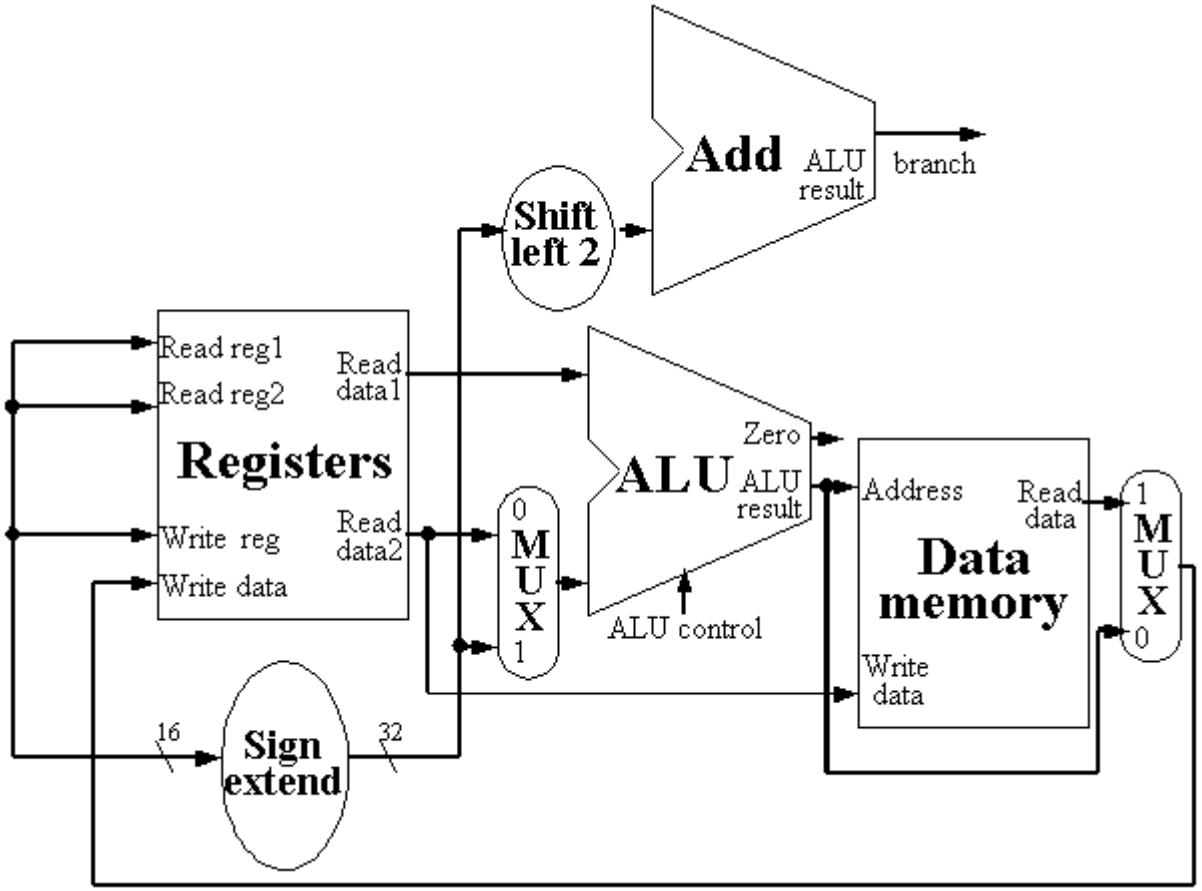
Combine datapaths for R-type and memory instructions: memory flow

- 1 register file, 1 ALU
- ALU input mux: data from register or instruction
- register write mux: data from ALU or memory

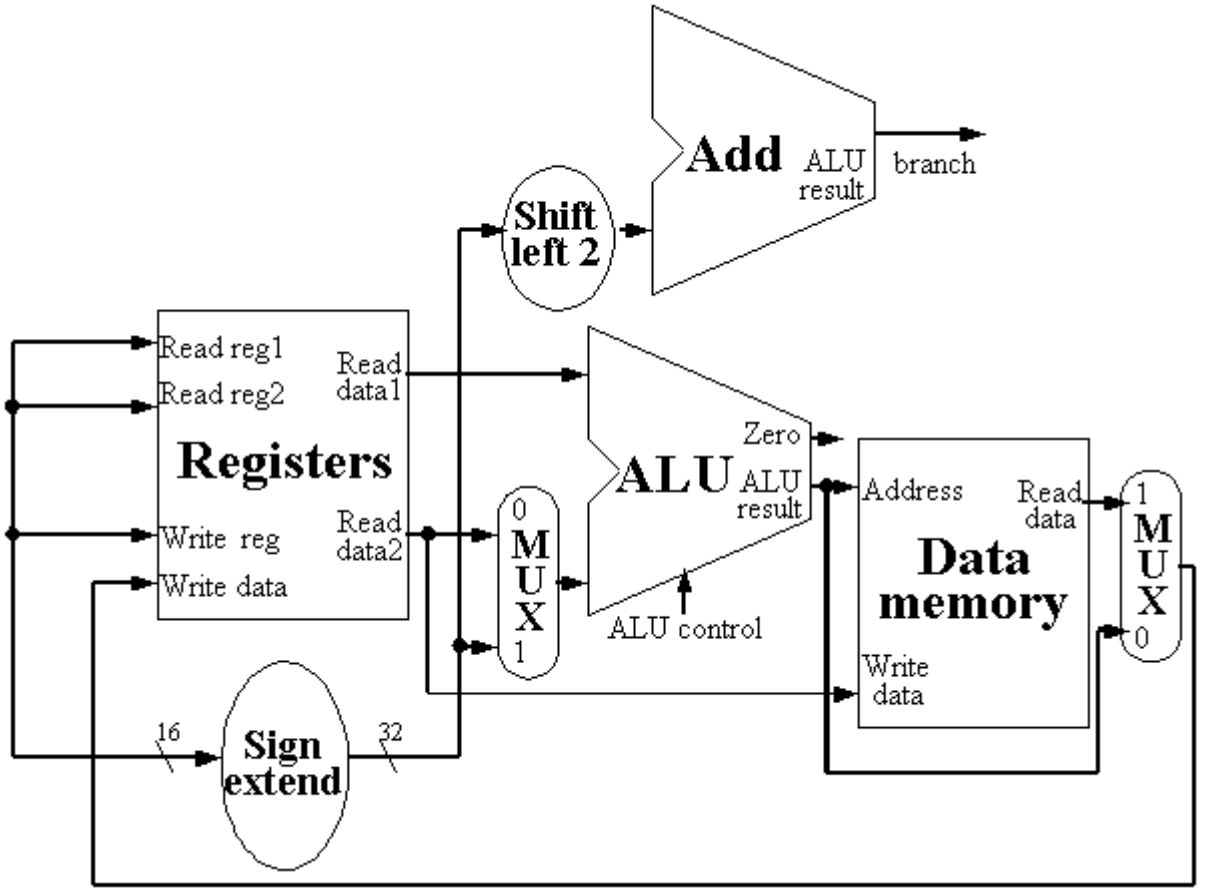


Branch instructions

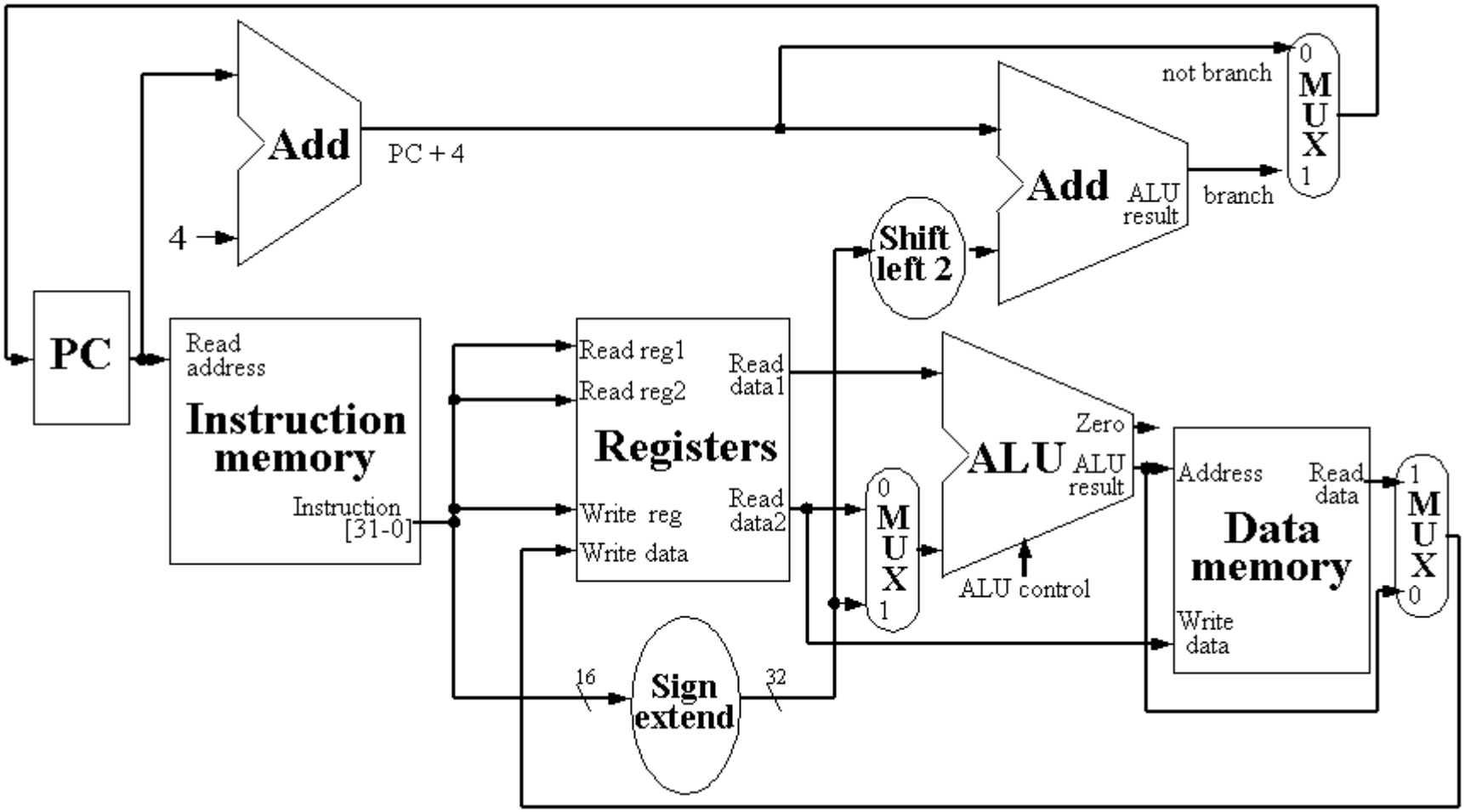
- beq $5, $6, L



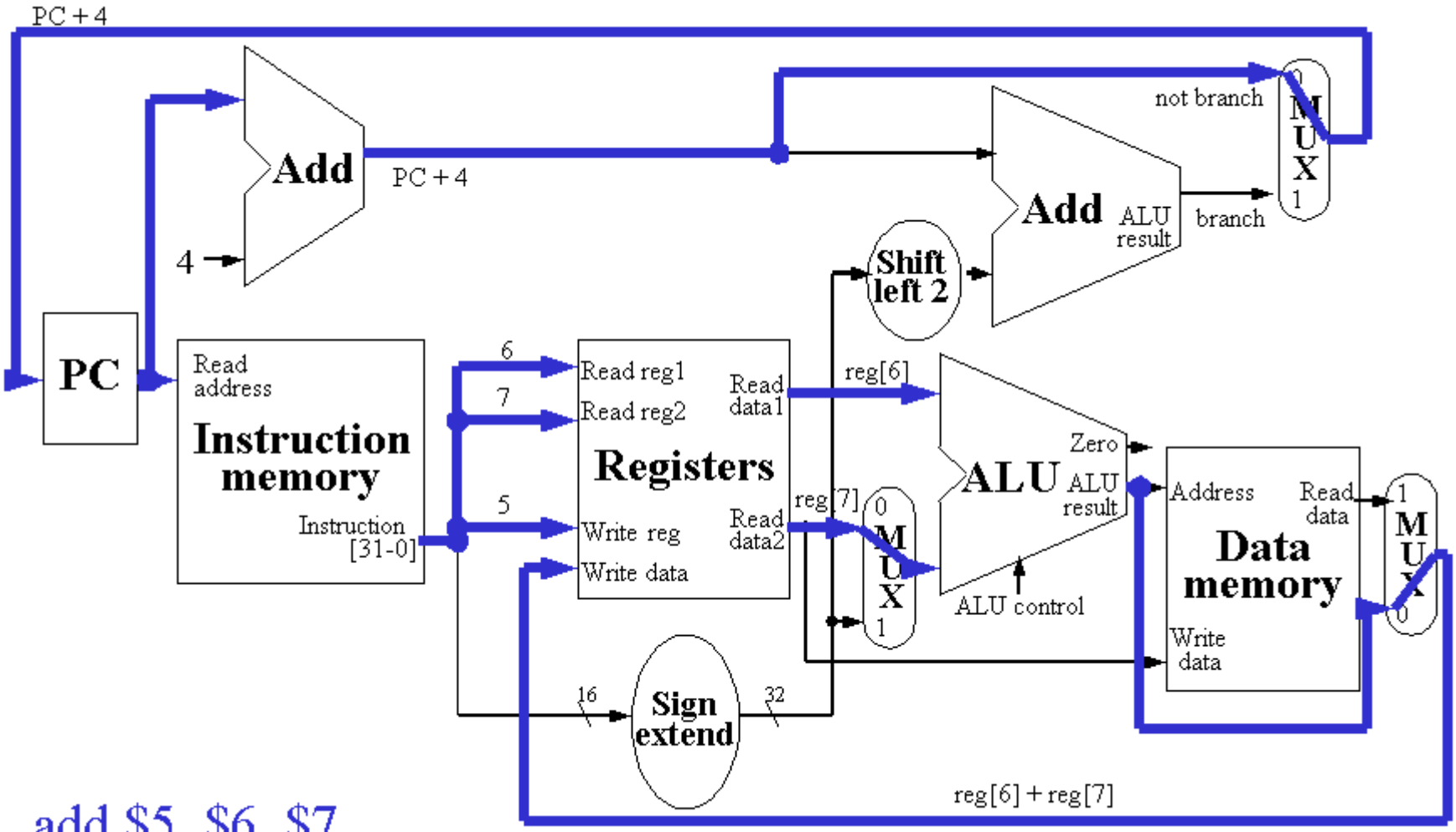
Combined datapath: without instruction memory



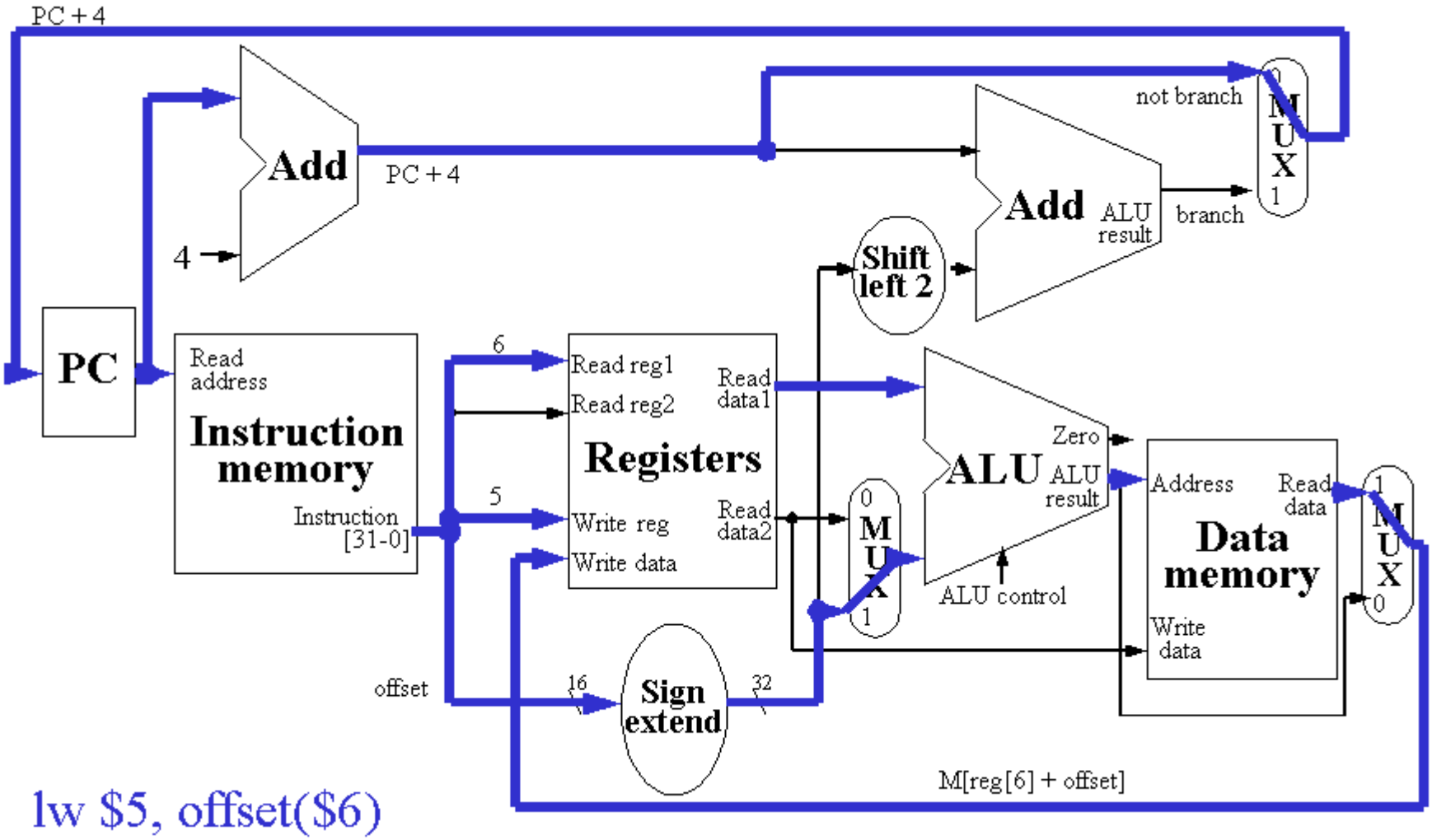
Combined datapath: with instruction memory



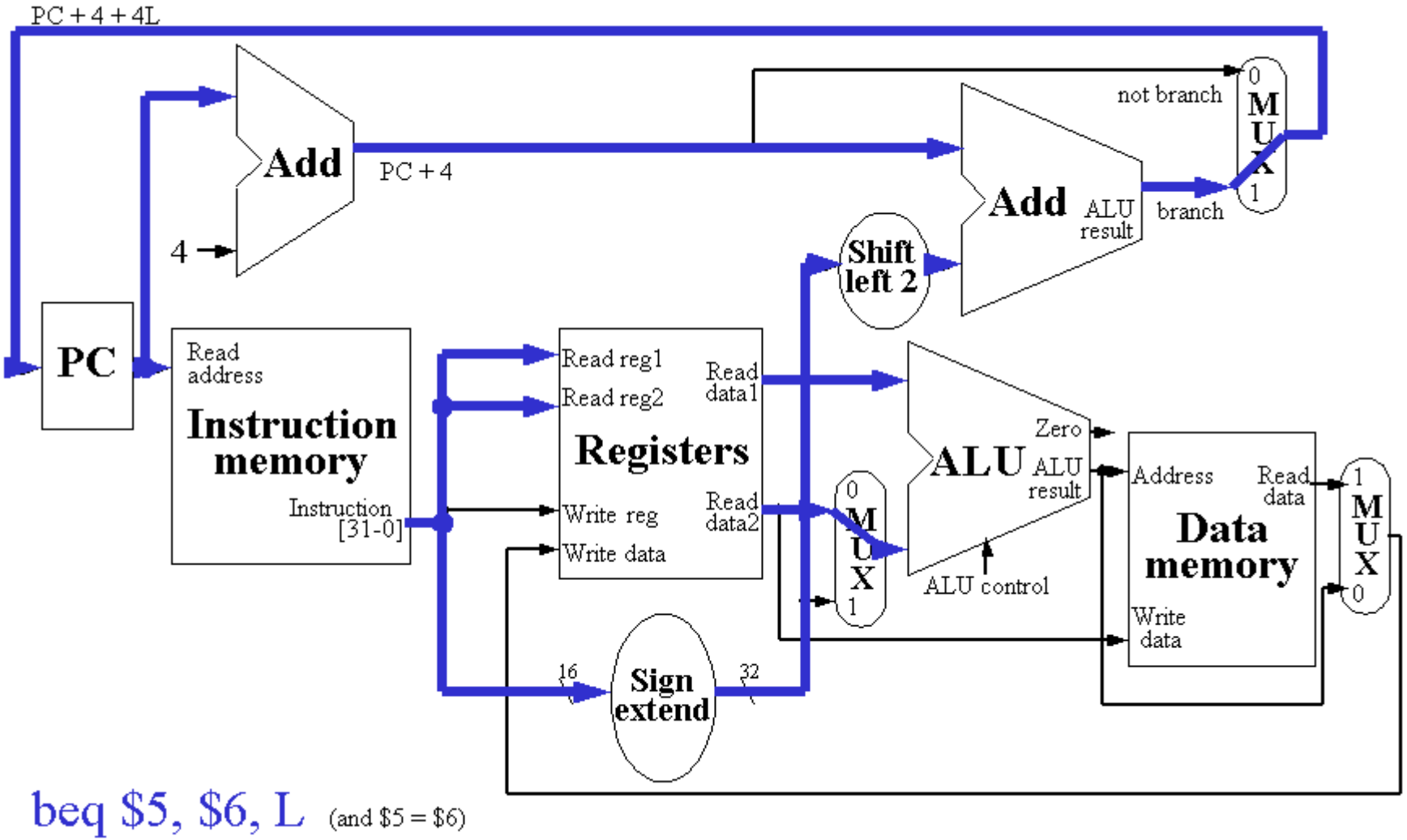
Combined datapath for R-type



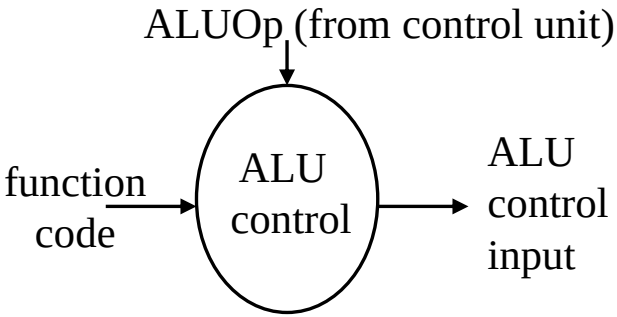
Combined datapath for load



Combined datapath for branch



beq $5, $6, L (and $5 = $6)



ALU control

- from opcode and fn code, determine ALU control

Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	**add**	**010**
SW	00	store word	XXXXXX	**add**	**010**
Branch equal	01	branch equal	XXXXXX	subtract	110
R-type	10	add	100000	**add**	**010**
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type	10	OR	100101	or	001
R-type	10	set-on-less-than	101010	set-on-less-than	111

from opcode
specify instruction type

constant

lecture on ALU

ALU control: optimisation

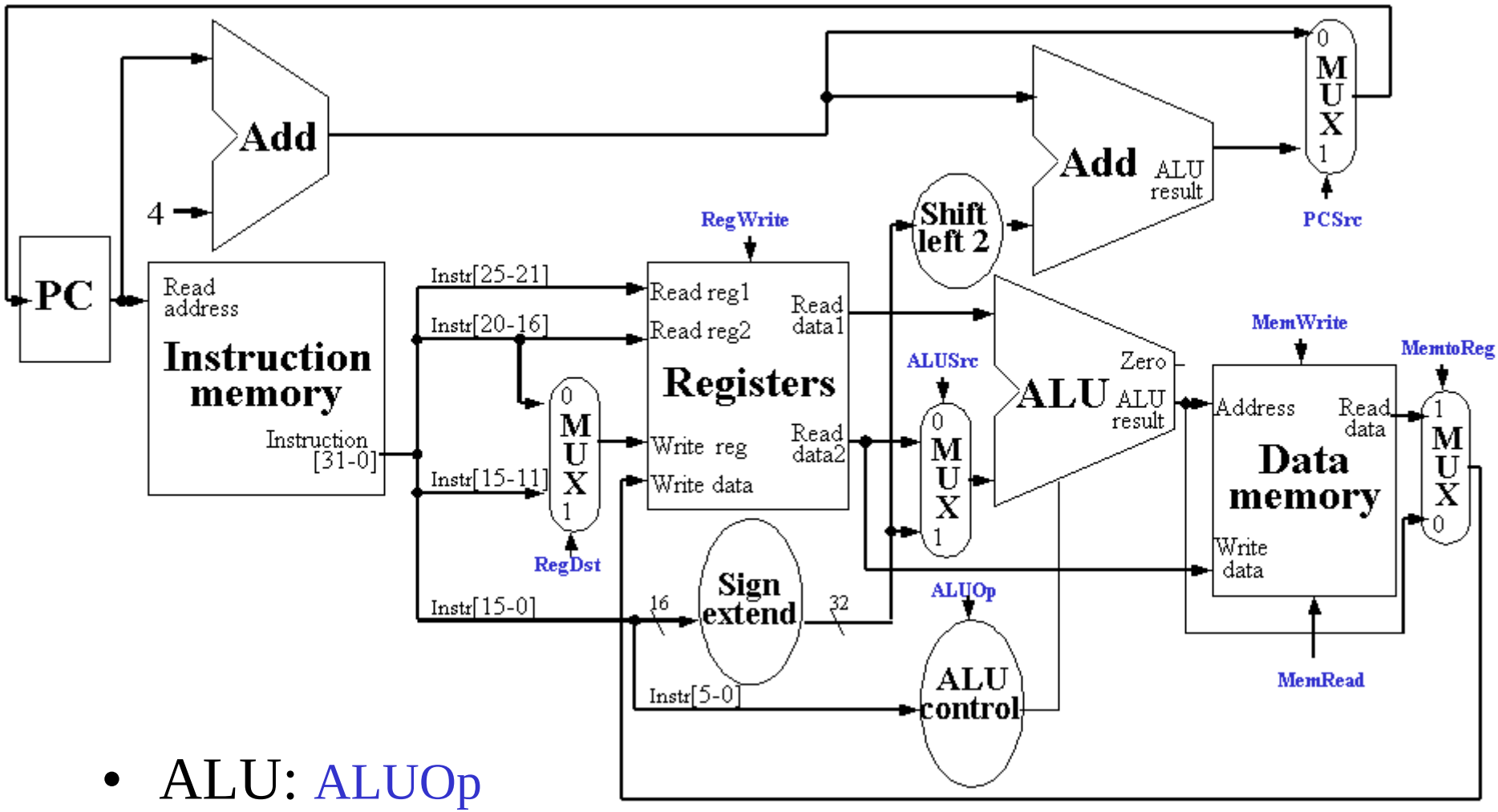
insert don't cares (Xs) since 11 never arises

ALUOp		Function code						ALU control
ALUOp1	ALUOp2	F5	F4	F3	F2	F1	F0	input
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	**X**	X	X	0	0	0	0	010
1	**X**	X	X	0	0	**1**	0	110
1	**X**	X	X	0	**1**	0	0	000
1	**X**	X	X	0	**1**	0	**1**	001
1	**X**	X	X	**1**	0	**1**	0	111

- K-map for rightmost bit: F0 or F3
- combine ALUOp: ALUOp1 and (F0 or F3)

F0,F1 \ F2,F3	00	01	11	10
00	0	X	X	0
01	0	1	X	X
11	X	X	X	X
10	X	X	X	1

Identify control signals

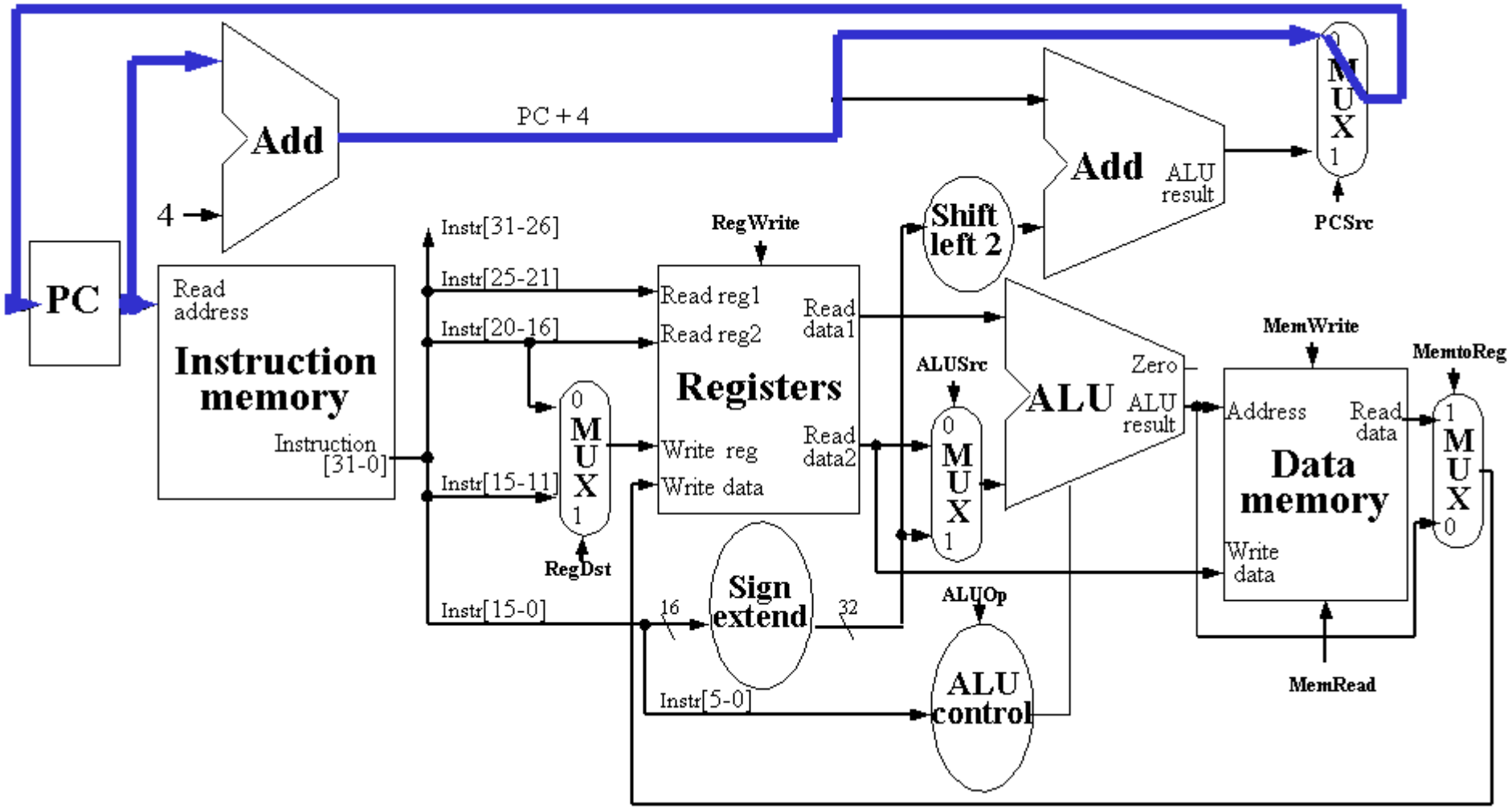


- ALU: ALUOp
- control 4 mux: RegDst, ALUSrc, MemtoReg, PCSrc
- storage: Register Write, Memory Read/Write

Control signal summary

Signal name	Effect when deasserted	Effect when asserted
MemRead	None (R, sw, beq)	Data memory contents at the read address are put on read data output. (lw)
MemWrite	None (R, lw, beq)	Data memory contents at address given by write addtess is replaced by value on write data input. (sw)
ALUSrc	The second ALU operand comes form the second register file output. (R, beq)	The second ALU operand is the sign-extend lower 16-bits of the instruction. (lw, sw)
RegDst	The register destination number for the Write register comes from the rt field. (lw)	The register destination number for the Write register comes from the rd field. (R)
RegWrite	None (sw, beq)	The register on the Write register input is written into with the value on the write data input. (R, lw)
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4	The PC is replaced by the output of the adder that computes the branch target.
MemtoReg	The value fed to the register write data input comes from the ALU. (R)	The value fed to the register write data input comes from the data memory. (lw)
Branch	0 (R, lw, sw)	1 (beq)

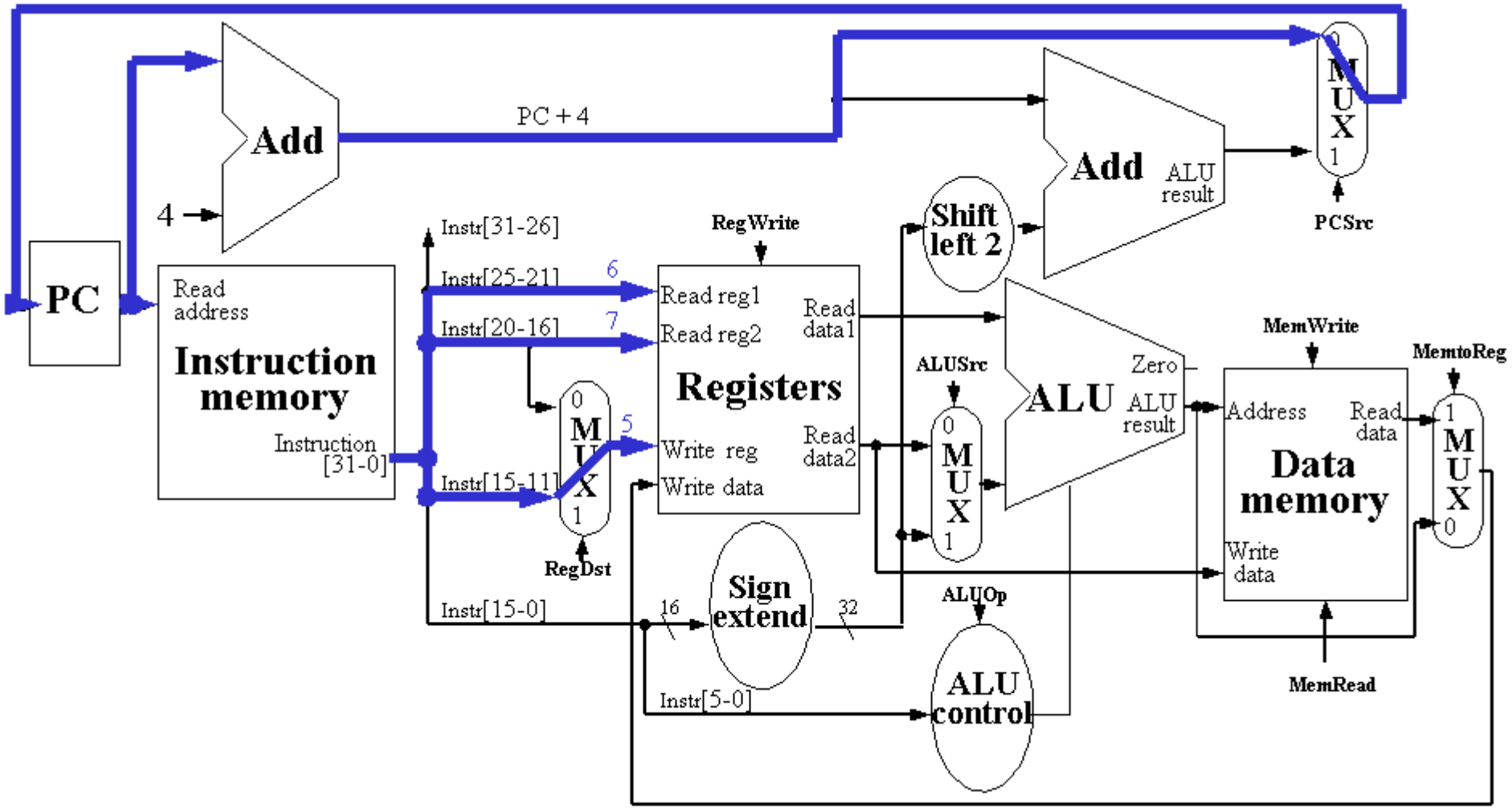
R-type instruction (1)



active connections for `add $5,$6,$7`

- note the PC loop and Register loop

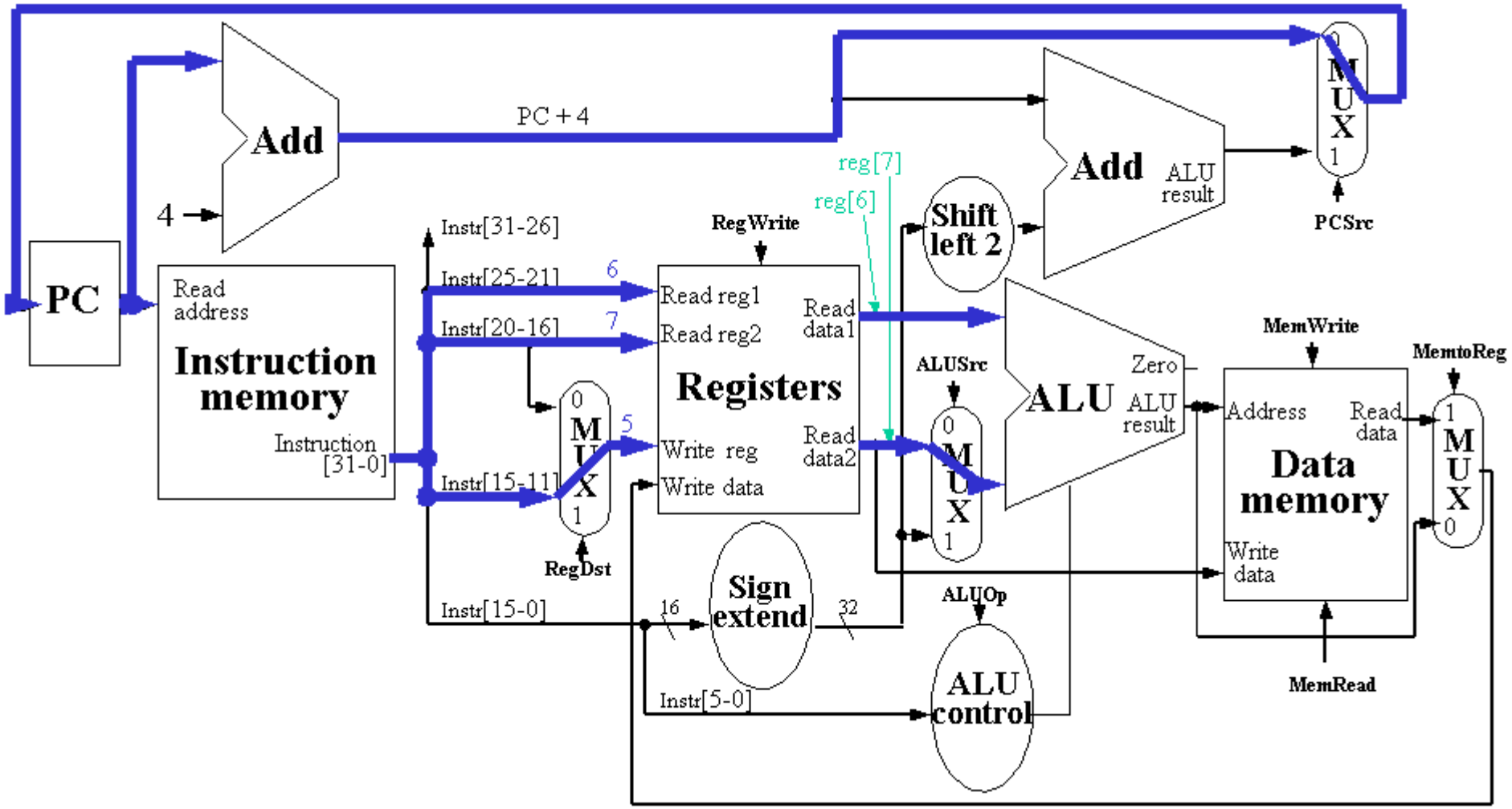
R-type instruction (2)



active connections for `add $5,$6,$7`

- note the PC loop and Register loop

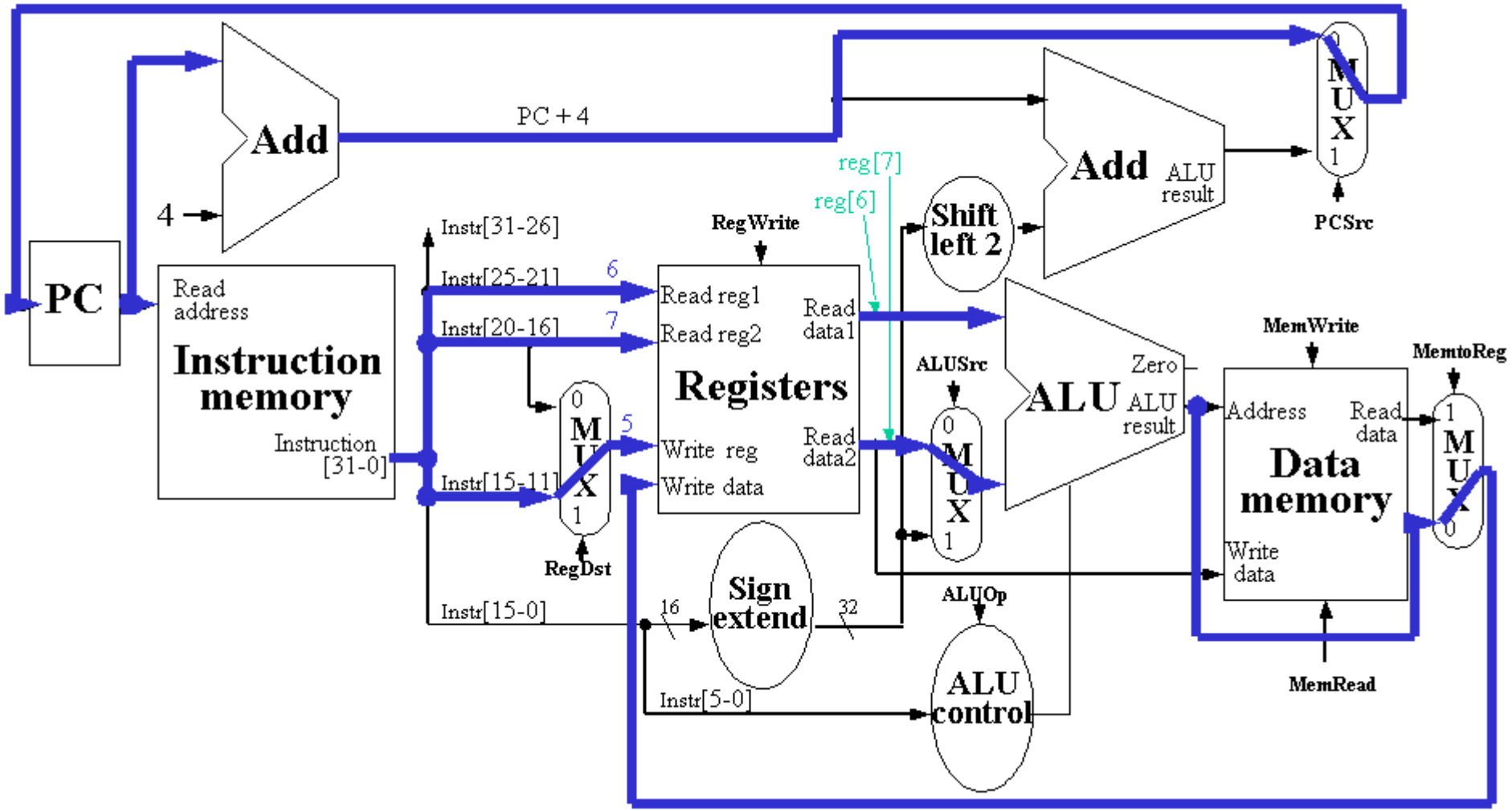
R-type instruction (3)



active connections for `add $5,$6,$7`

- note the PC loop and Register loop

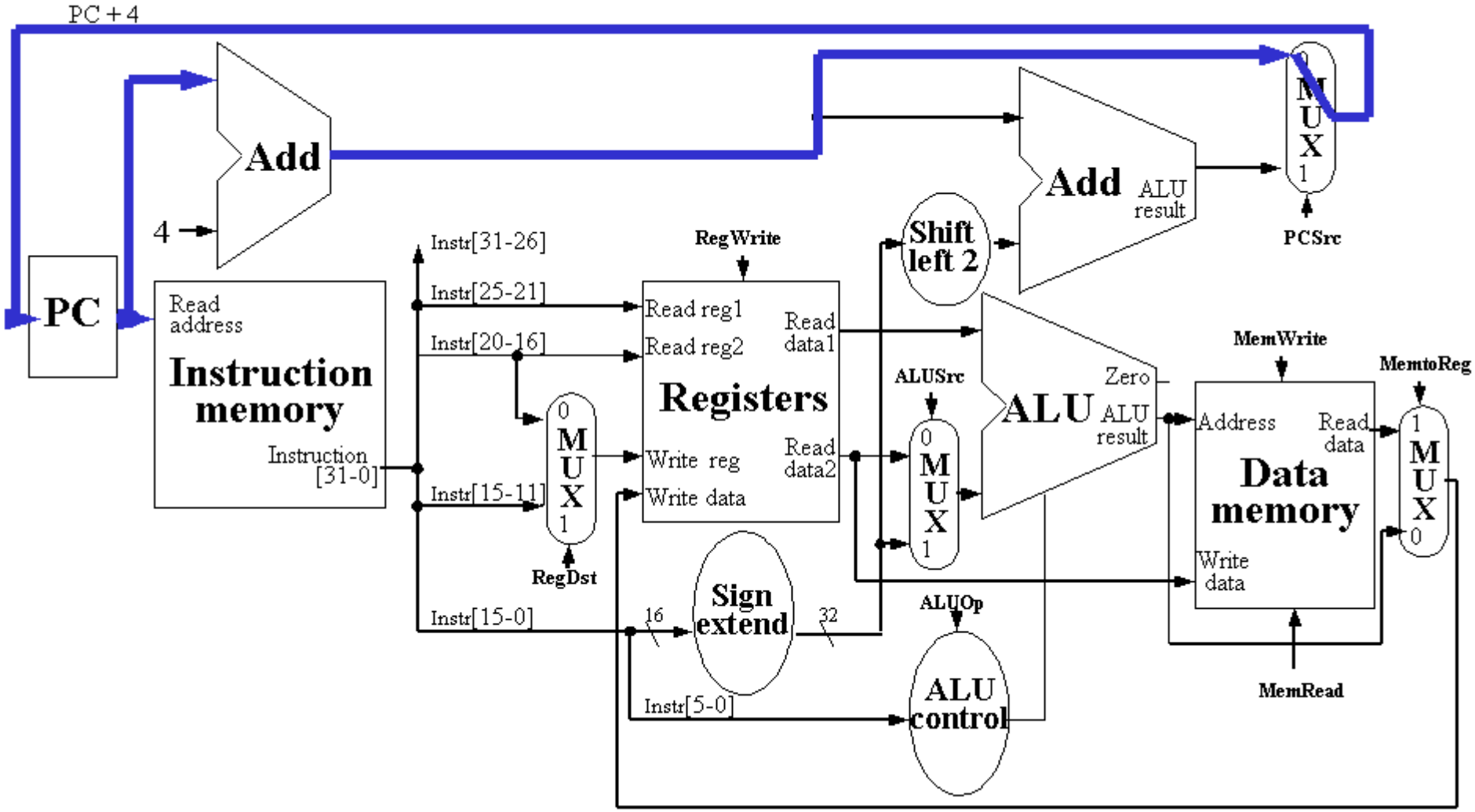
R-type instruction (4)



active connections for `add $5,$6,$7`

- note the PC loop and Register loop

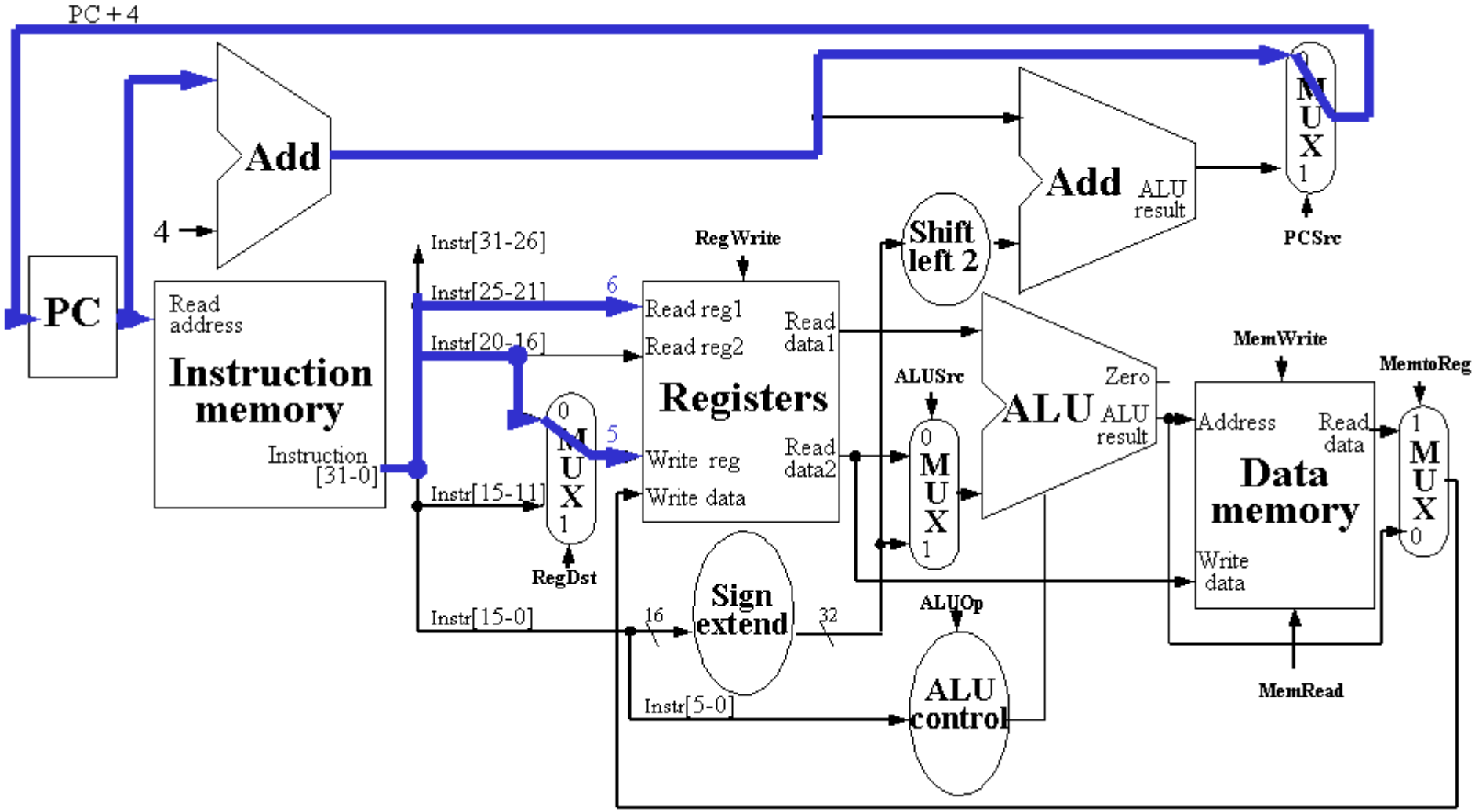
Load instruction (1)



active connections for `lw $5,offset($6)`

- note the PC loop and Register loop

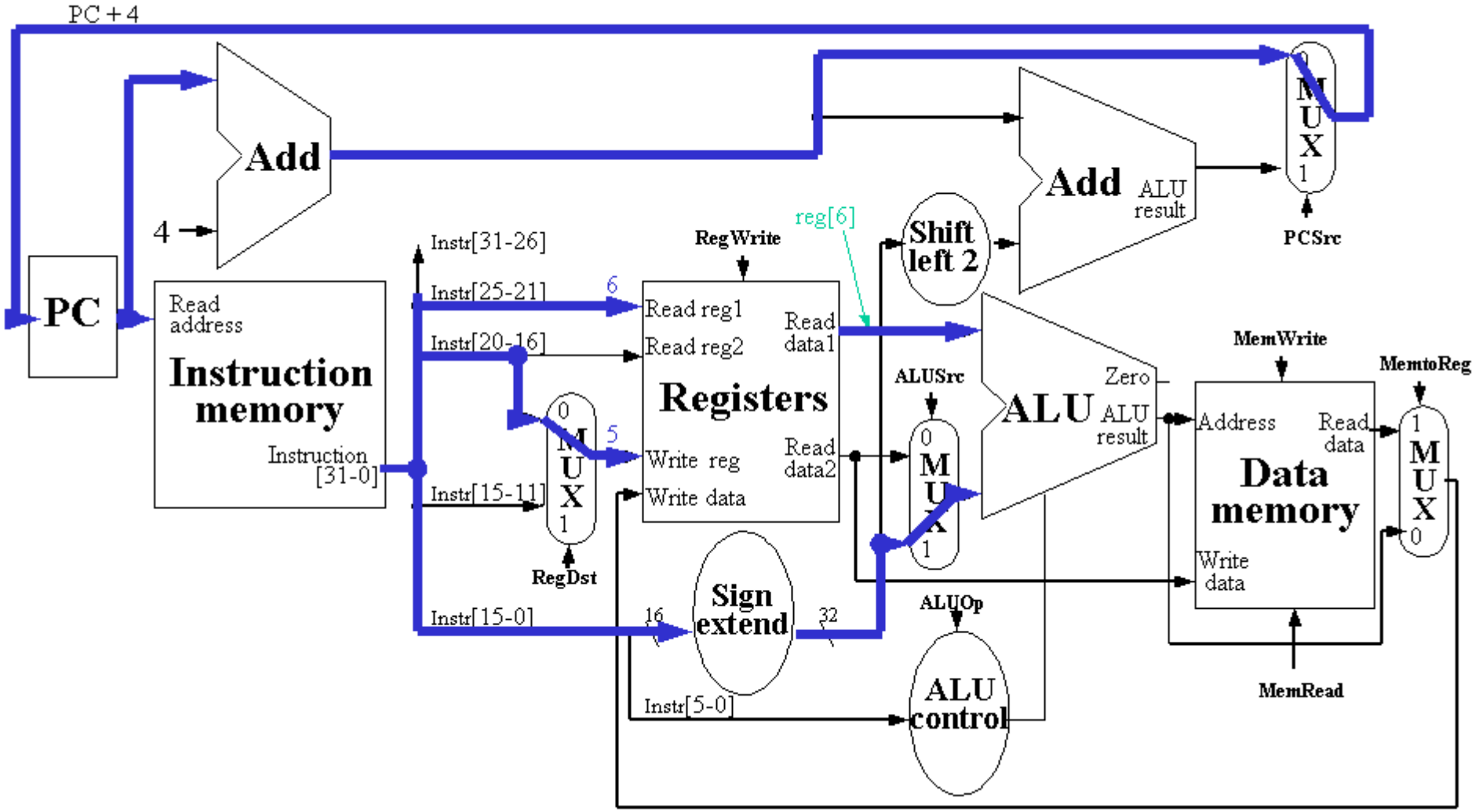
Load instruction (2)



active connections for `lw $5,offset($6)`

- note the PC loop and Register loop

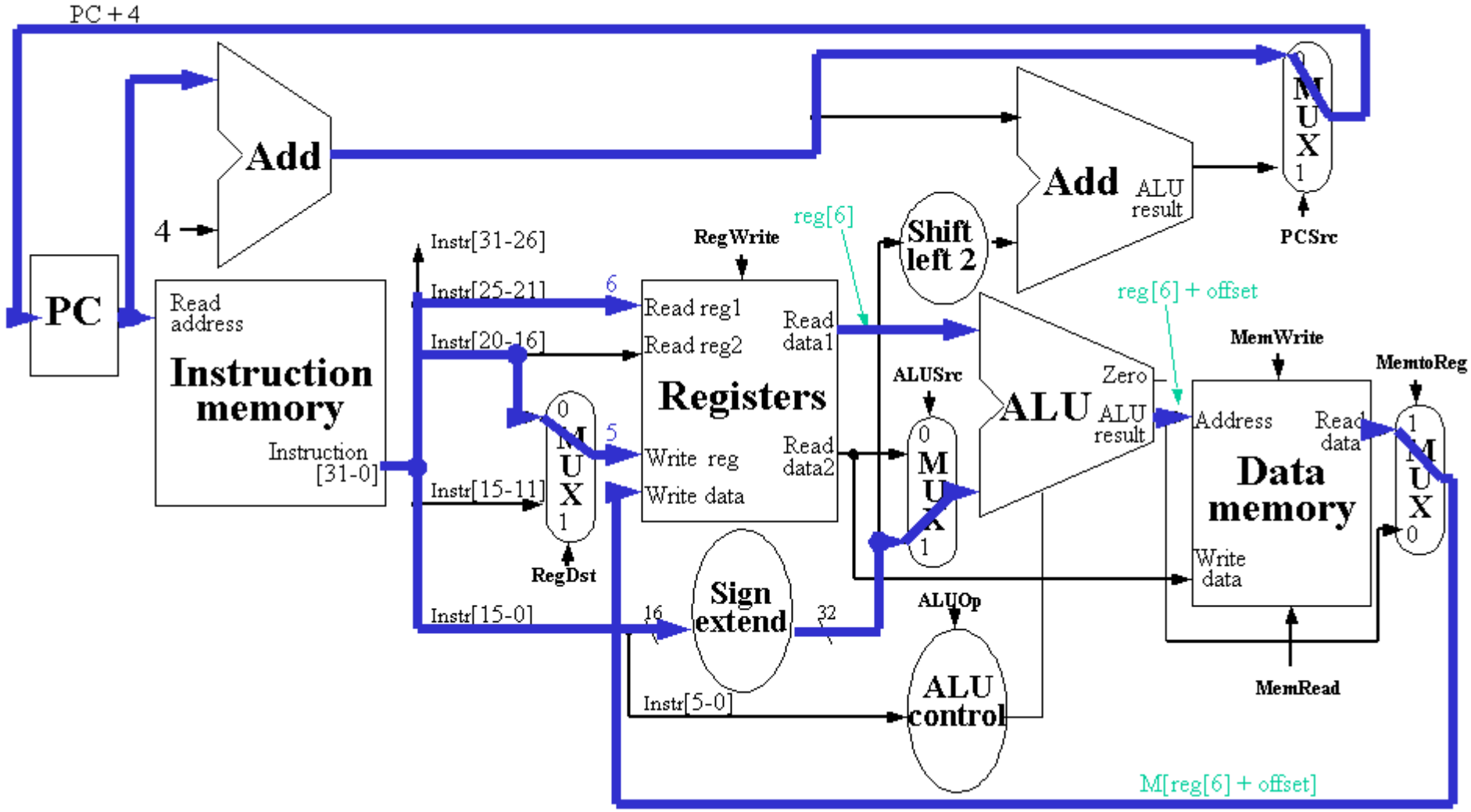
Load instruction (3)



active connections for `lw $5,offset($6)`

- note the PC loop and Register loop

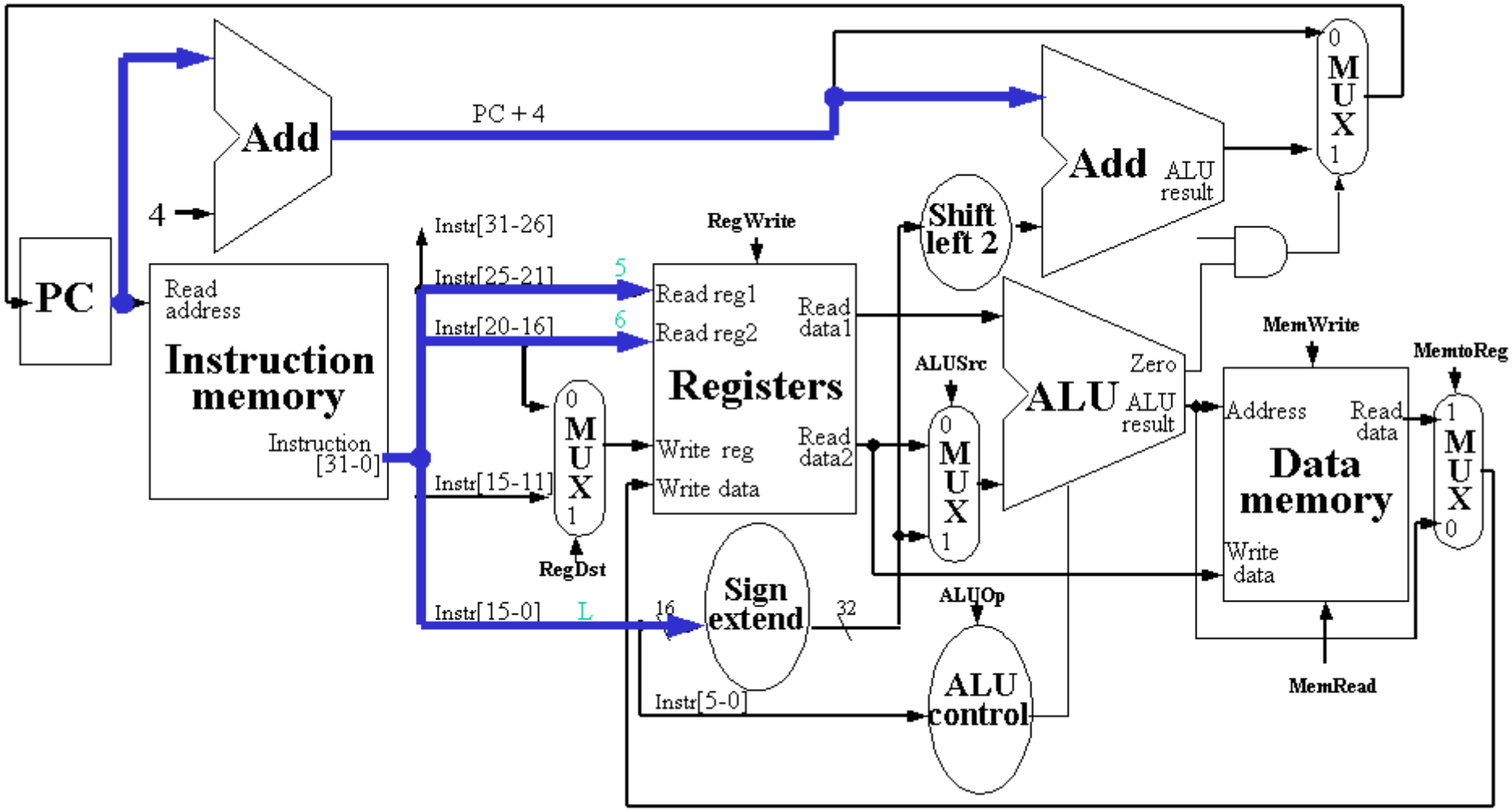
Load instruction (4) and (5)



active connections for `lw $5,offset($6)`

- note the PC loop and Register loop

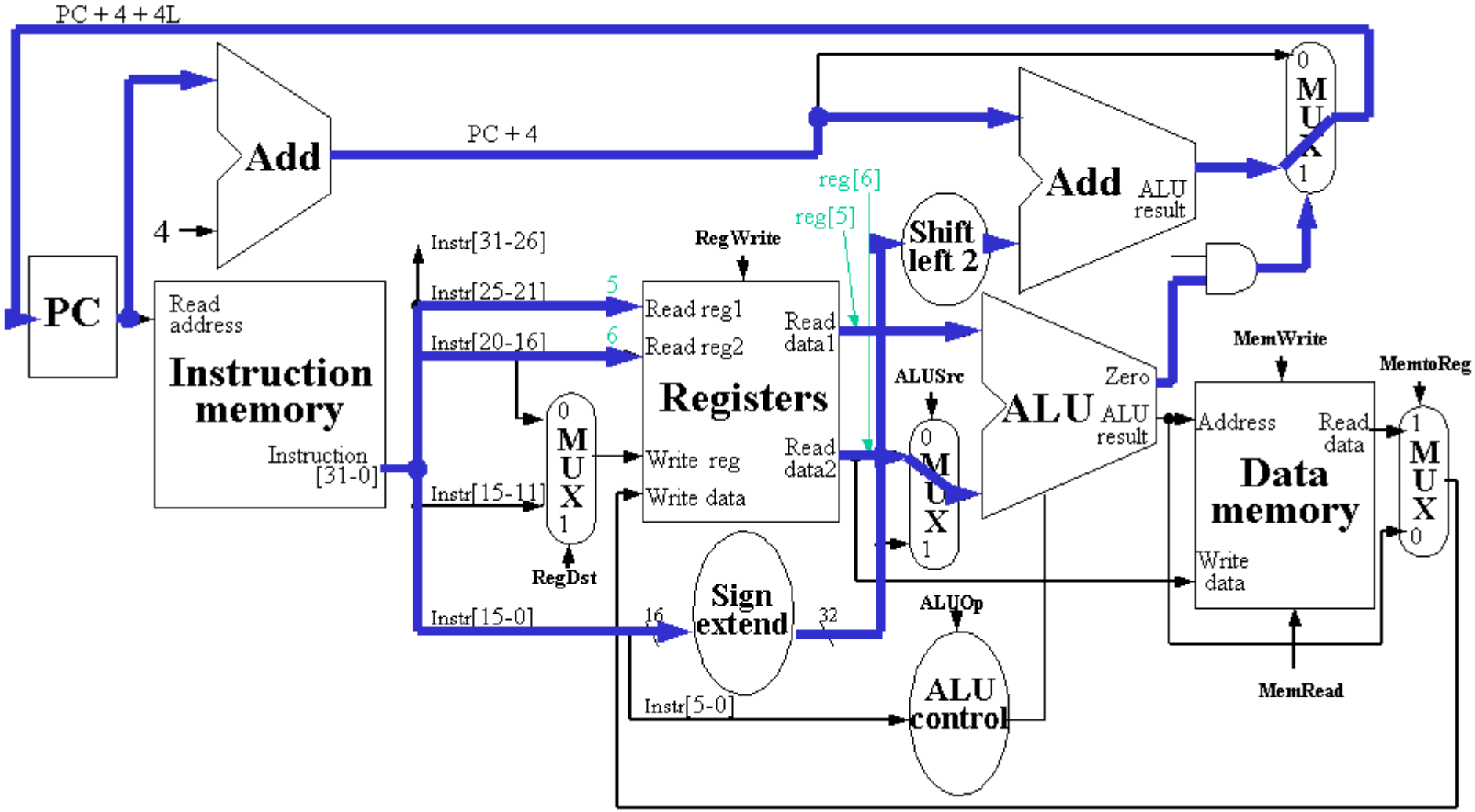
Branch instruction (1)



active connections for `beq $5,$6,L`

- exercise: extend datapath to support j-type instructions

Branch instruction (2)



active connections for `beq $5,$6,L` (and $5 = $6)

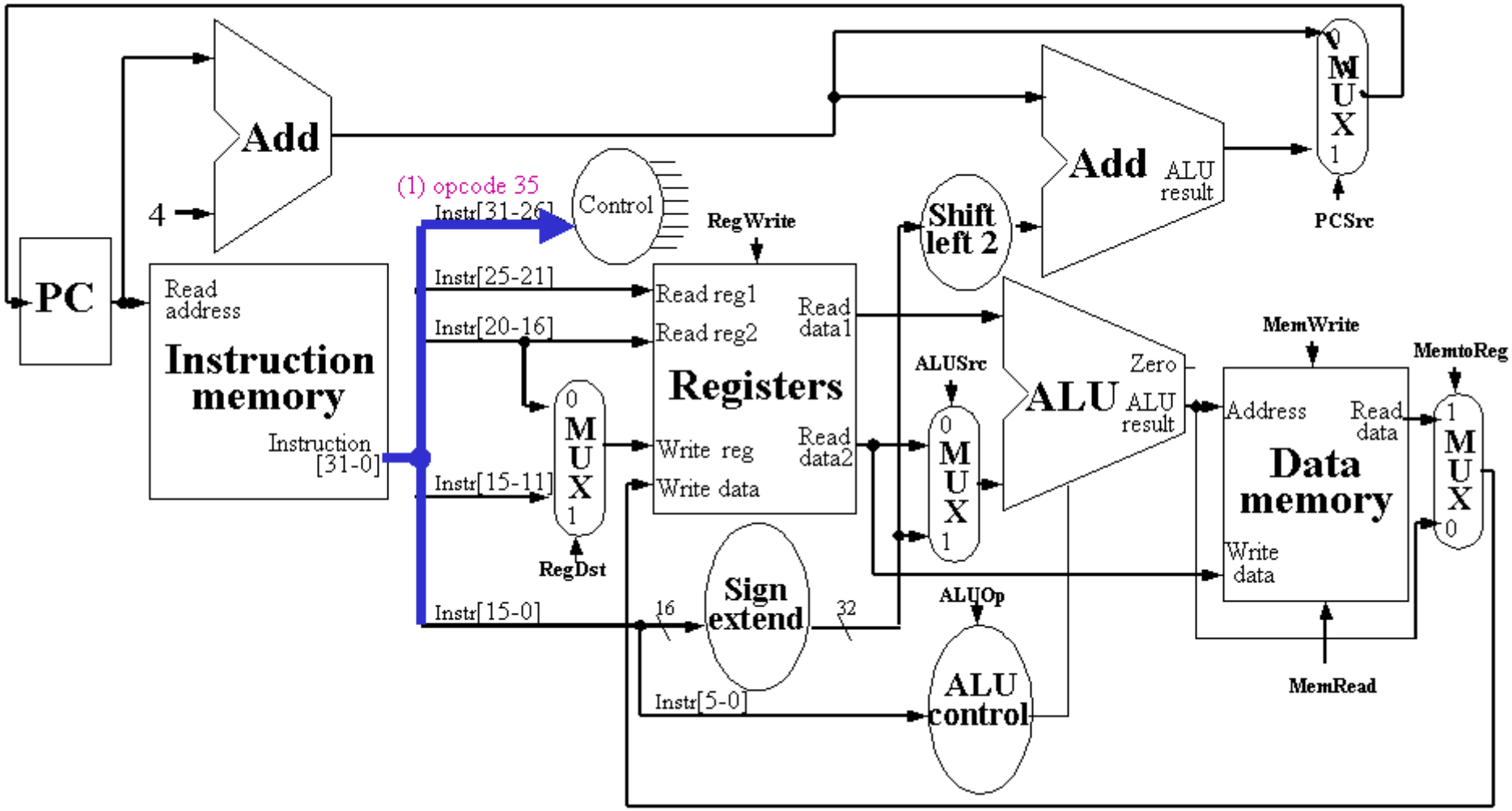
- exercise: extend datapath to support j-type instructions

Control unit for single-cycle datapath

- combinational circuit: no registers
- 6-bit opcode input
  e.g. lw 100011 (35_{ten})
- 9-bit output, control mux, ALU, read/write op
- e.g. lw:

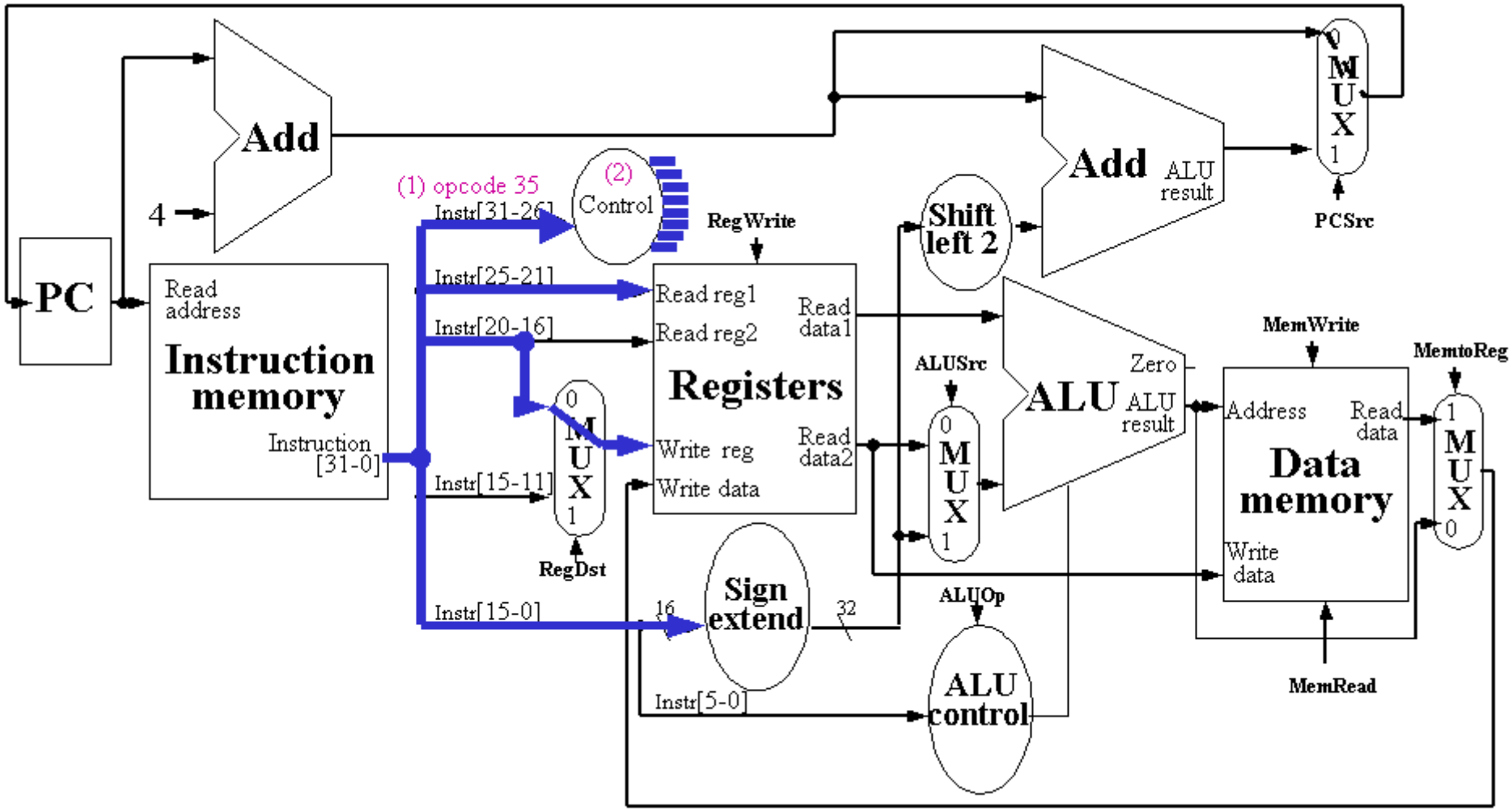
mux	memory/register	ALU
RegDst = 0	MemRead = 1	ALUOp = 0
Branch = 0	MemWrite = 0	
MemtoReg = 1	RegWrite = 1	
ALUSrc = 1		

Load instruction (1)



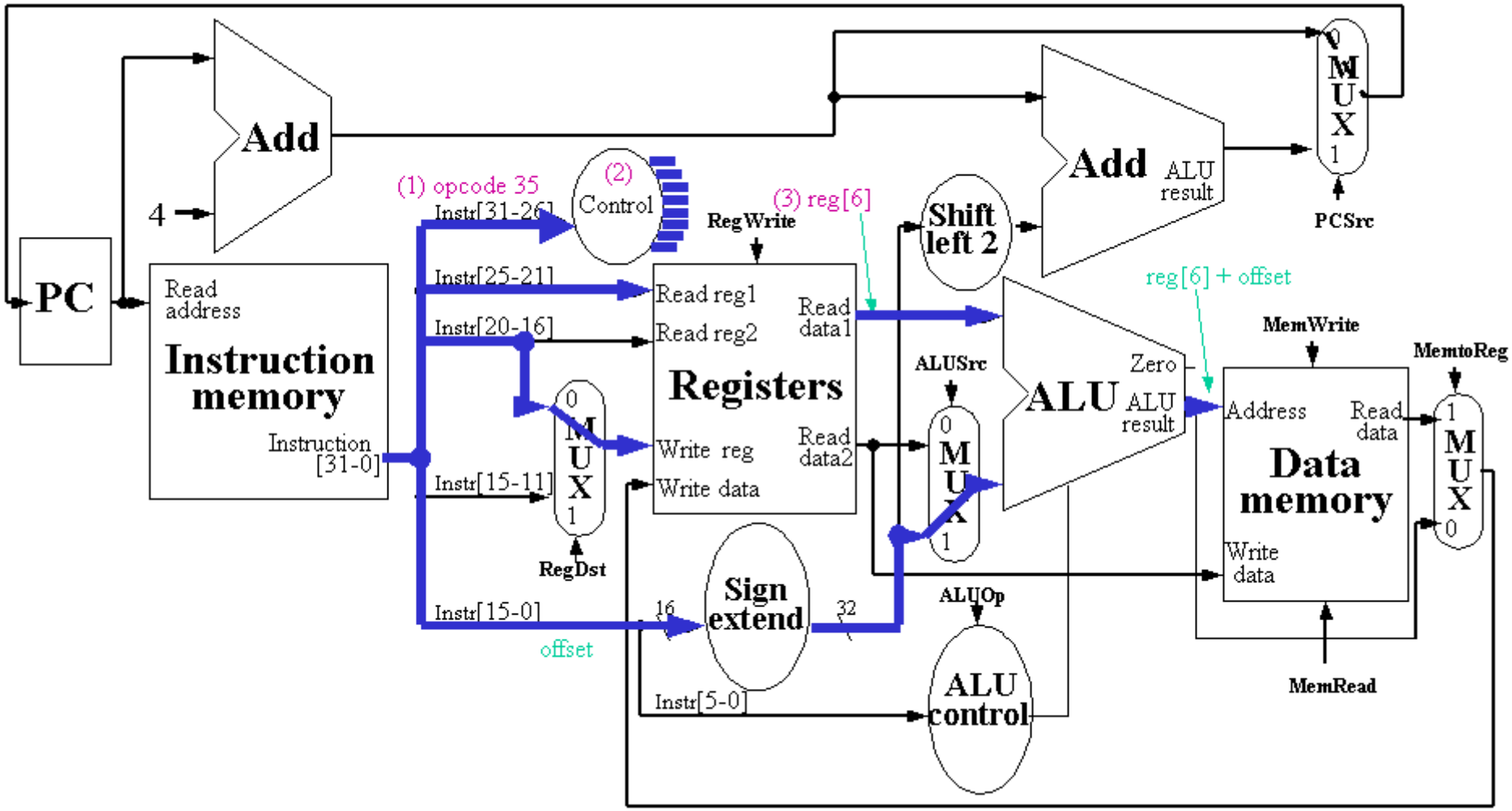
- `lw $5,offset($6)`

Load instruction (2)



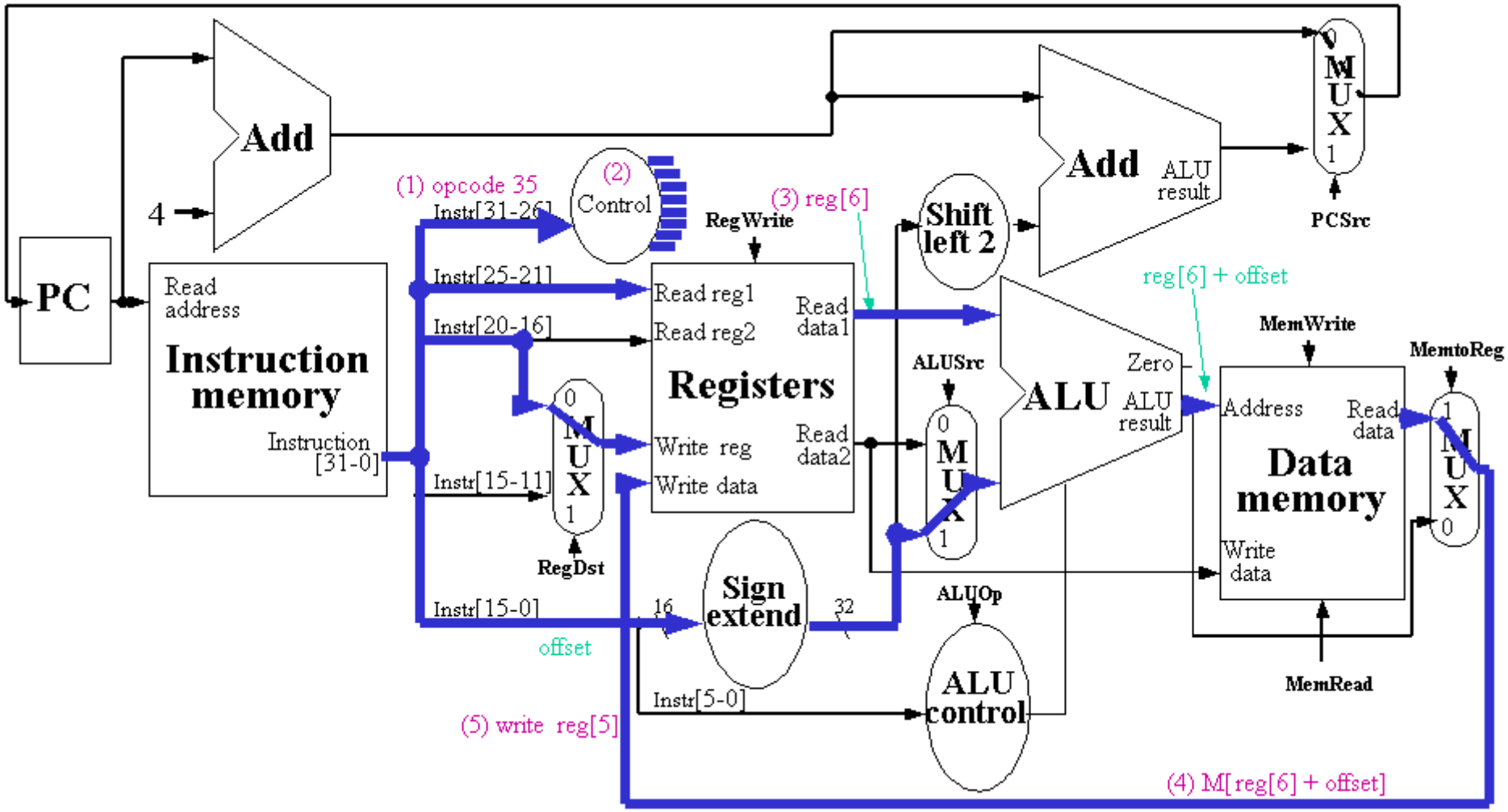
- `lw $5,offset($6)`

Load instruction (3)



- `lw $5,offset($6)`

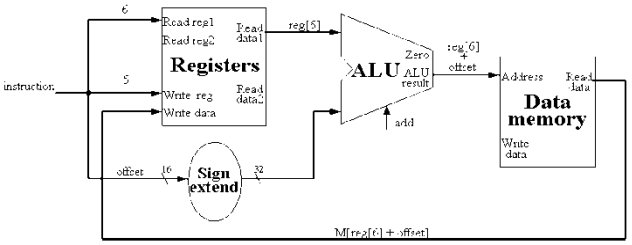
Load instruction (4) and (5)



- `lw $5,offset($6)`

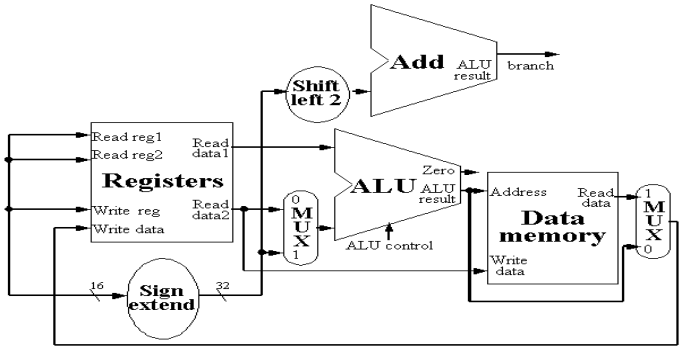
Summary

- different datapaths for:
  - register-based instructions e.g. add, sub
  - memory-access instructions e.g. lw, sw
  - branch instructions e.g. beq, j



- combined datapath:

  for different instructions
  - add multiplexors



- control unit: activate relevant parts of the combined datapath for a given instruction
  - control signals for multiplexors + ALU