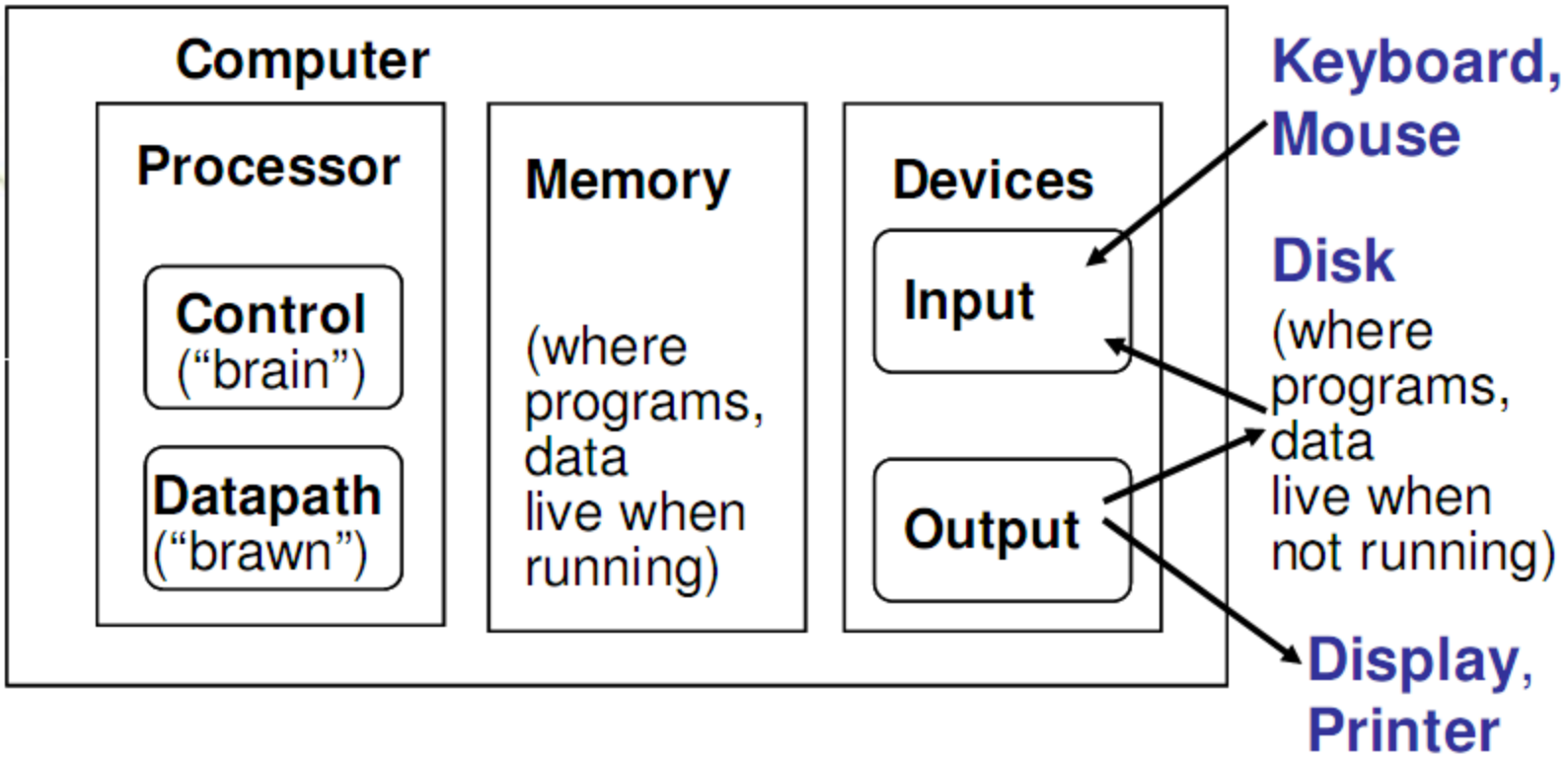
Cài đặt bộ xử lý MIPS 32 bit thu gọn

Môn học: Kiến trúc máy tính & Hợp ngữ

5 thành phần cơ bản của máy tính



Bộ vi xử lý (CPU)

- Datapath
  - Registers
  - ALU
- Control unit

- Stalling: CPU = {Registers, ALU, Control unit, Internal bus}

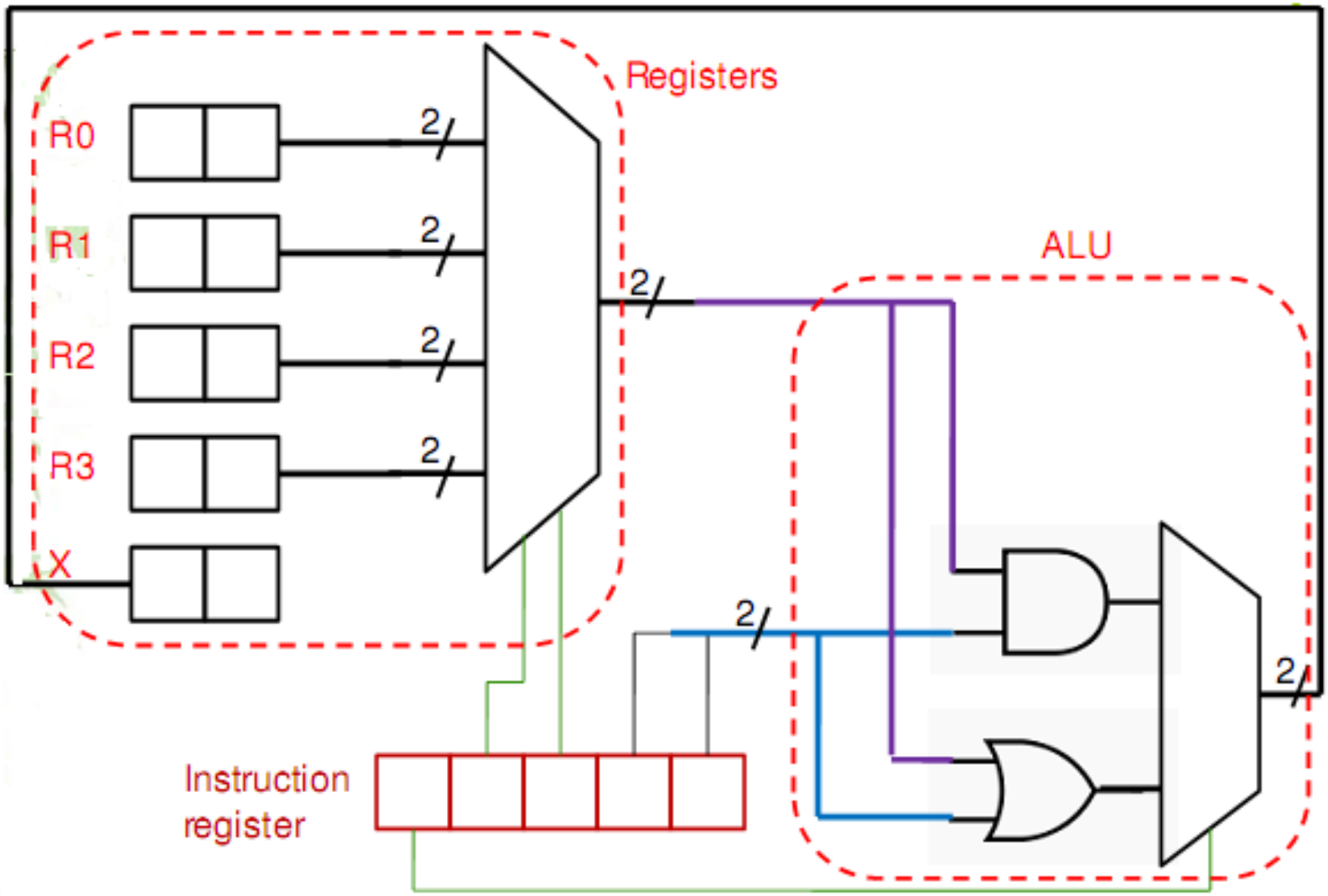
AND R, value
OR R, value

op	id	value

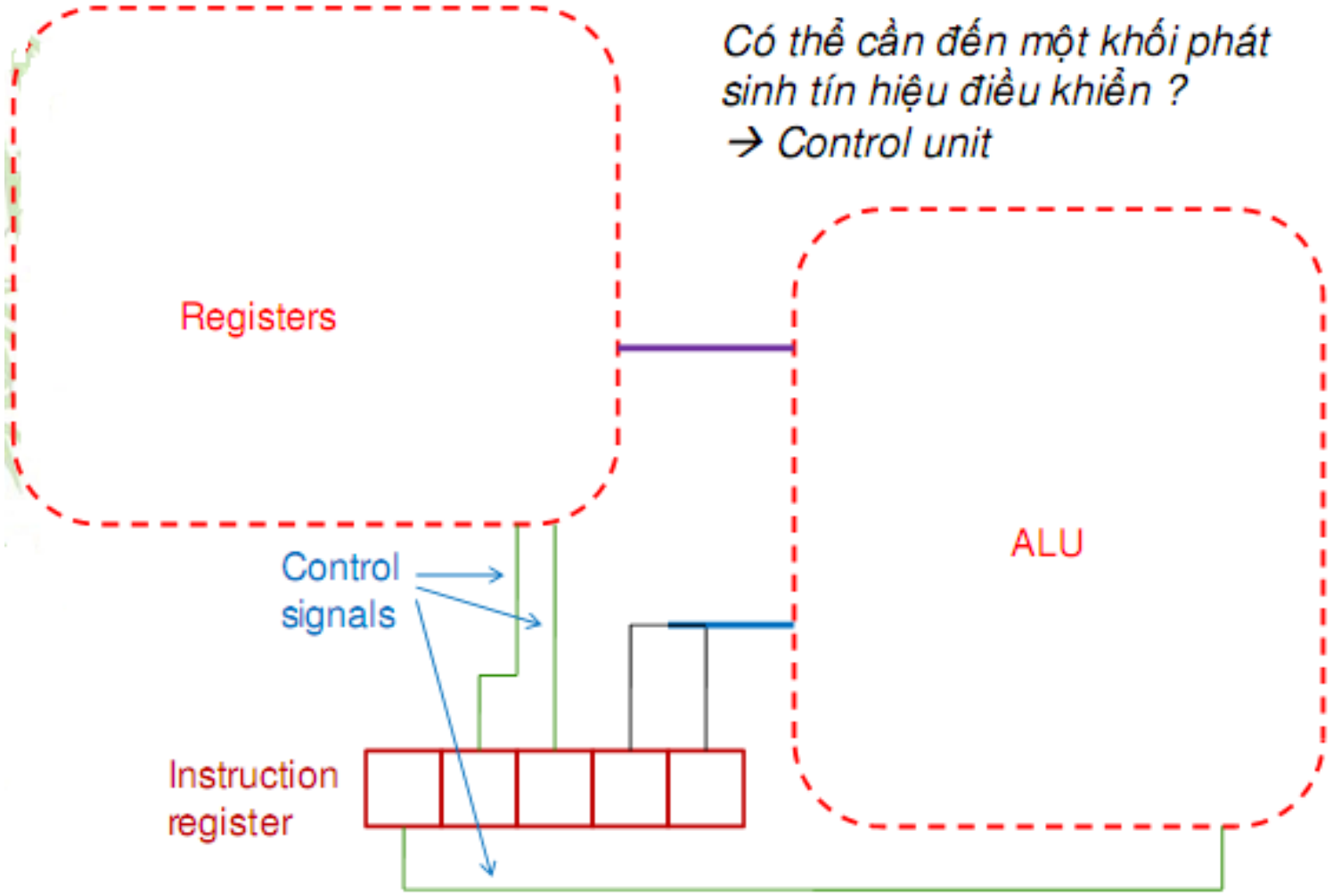
$X = R_{id}$ op value

1	0	1	1	0

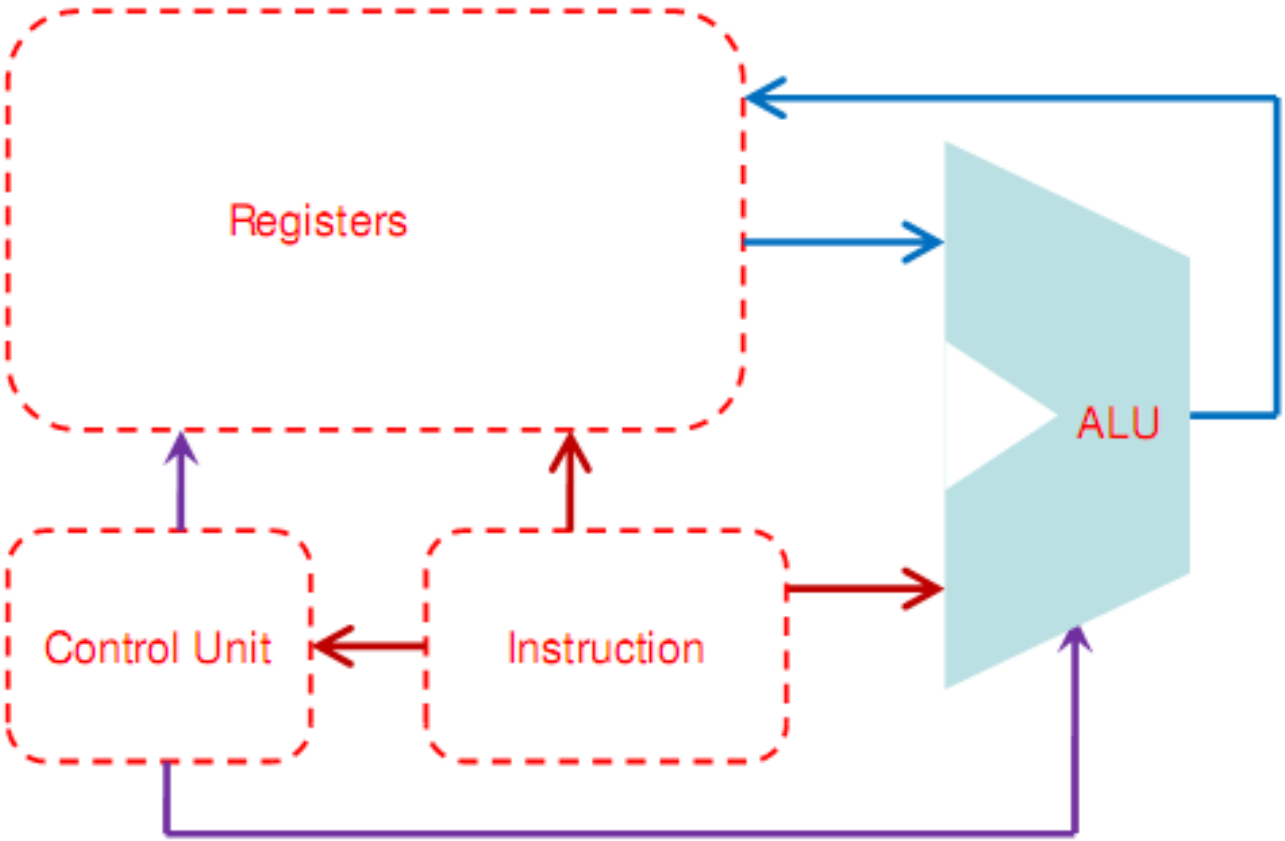
X = R1 OR 2



Control signals



Datapath & Control unit



MIPS thu gọn

- Lệnh truy xuất bộ nhớ: lw, sw
- Lệnh số học – luận lý: add, sub, and, or, slt
- Lệnh rẽ nhánh: beq, j

→ Thiết kế bộ xử lý (Datapath và Control) cho tập lệnh MIPS thu gọn này ?

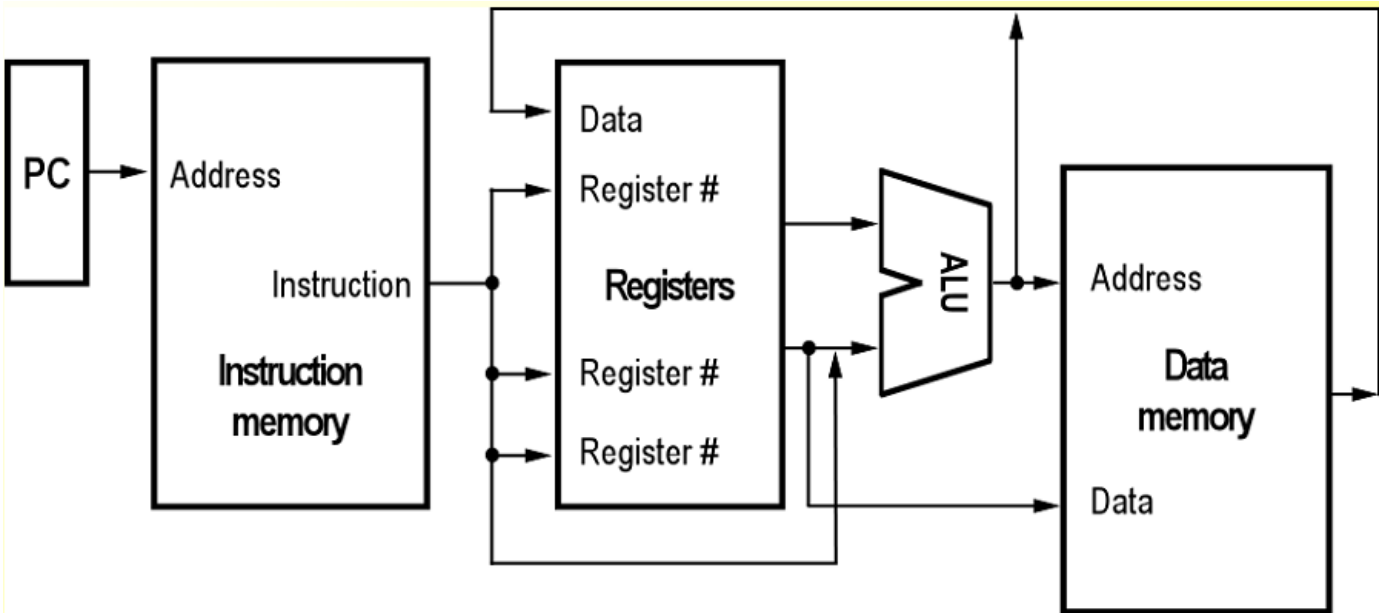
Một số lưu ý

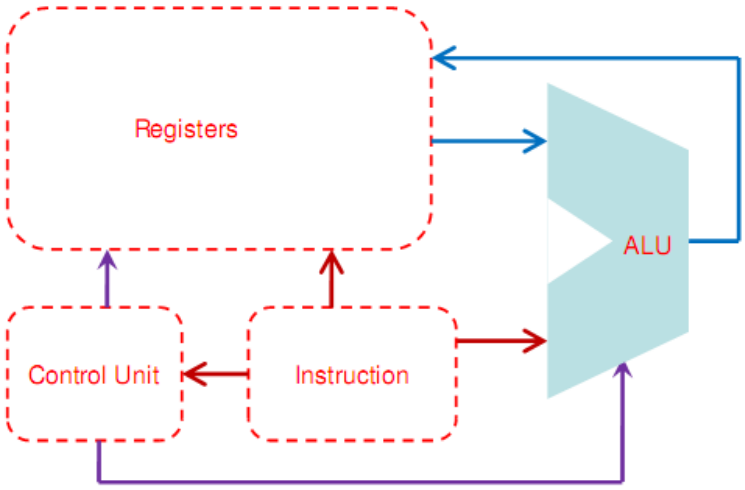
- Bất kỳ câu lệnh nào muốn thực thi cũng phải qua 2 bước đầu tiên:
  - Gửi địa chỉ lệnh chứa trong thanh ghi PC (Program counter) đến bộ nhớ lệnh để lấy nội dung câu lệnh từ bộ nhớ
  - Xác định toán hạng trong câu lệnh → Đọc các thanh ghi chứa toán hạng có địa chỉ tương ứng
- Các bước tiếp theo phụ thuộc vào từng nhóm lệnh khác nhau
- Tập lệnh MIPS thu gọn có các bước thực thi giống nhau ở khá nhiều điểm, khác biệt chủ yếu nằm ở các bước thực thi cuối của câu lệnh

Instruction format

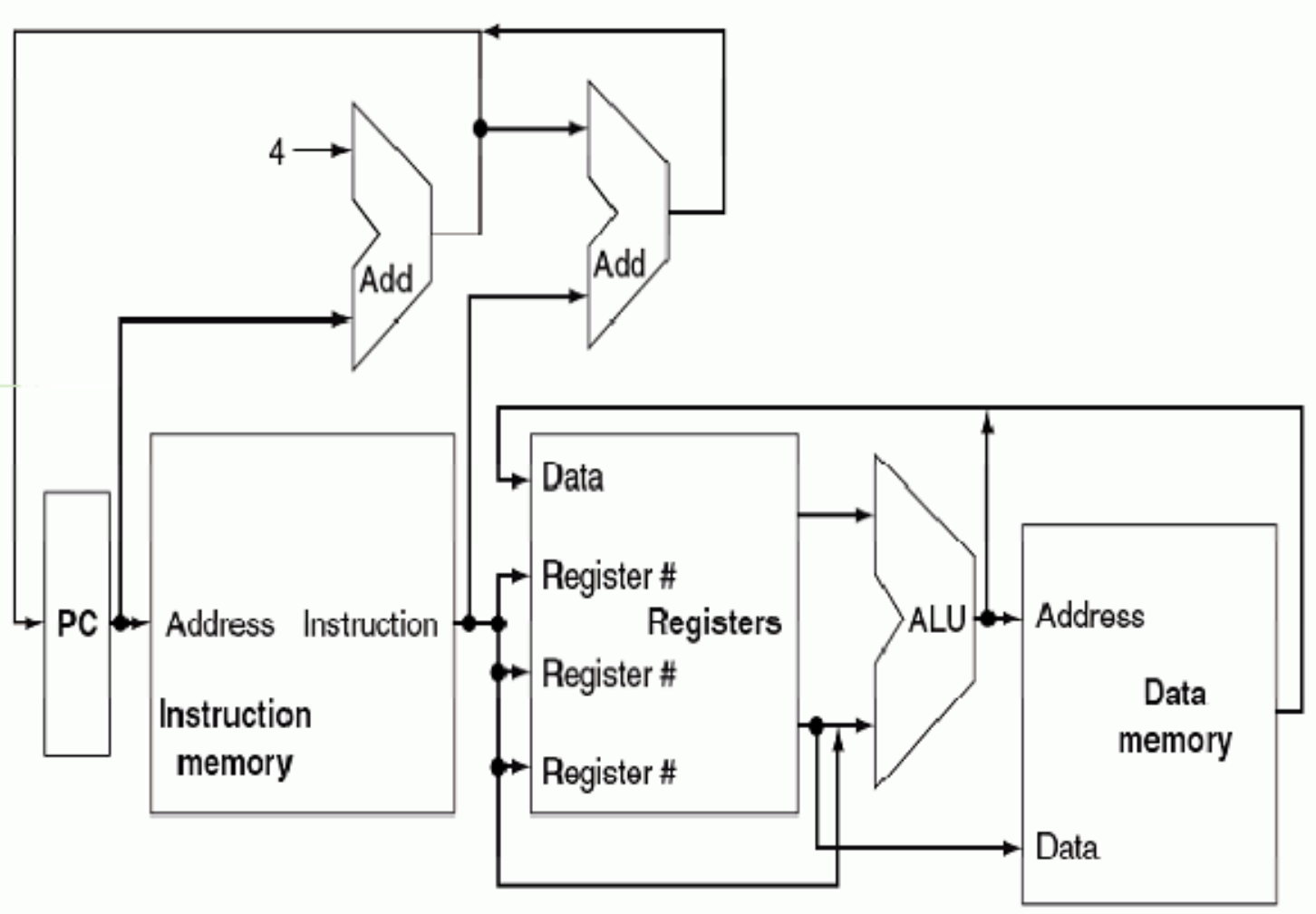
Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

Sơ đồ thực thi tổng quát

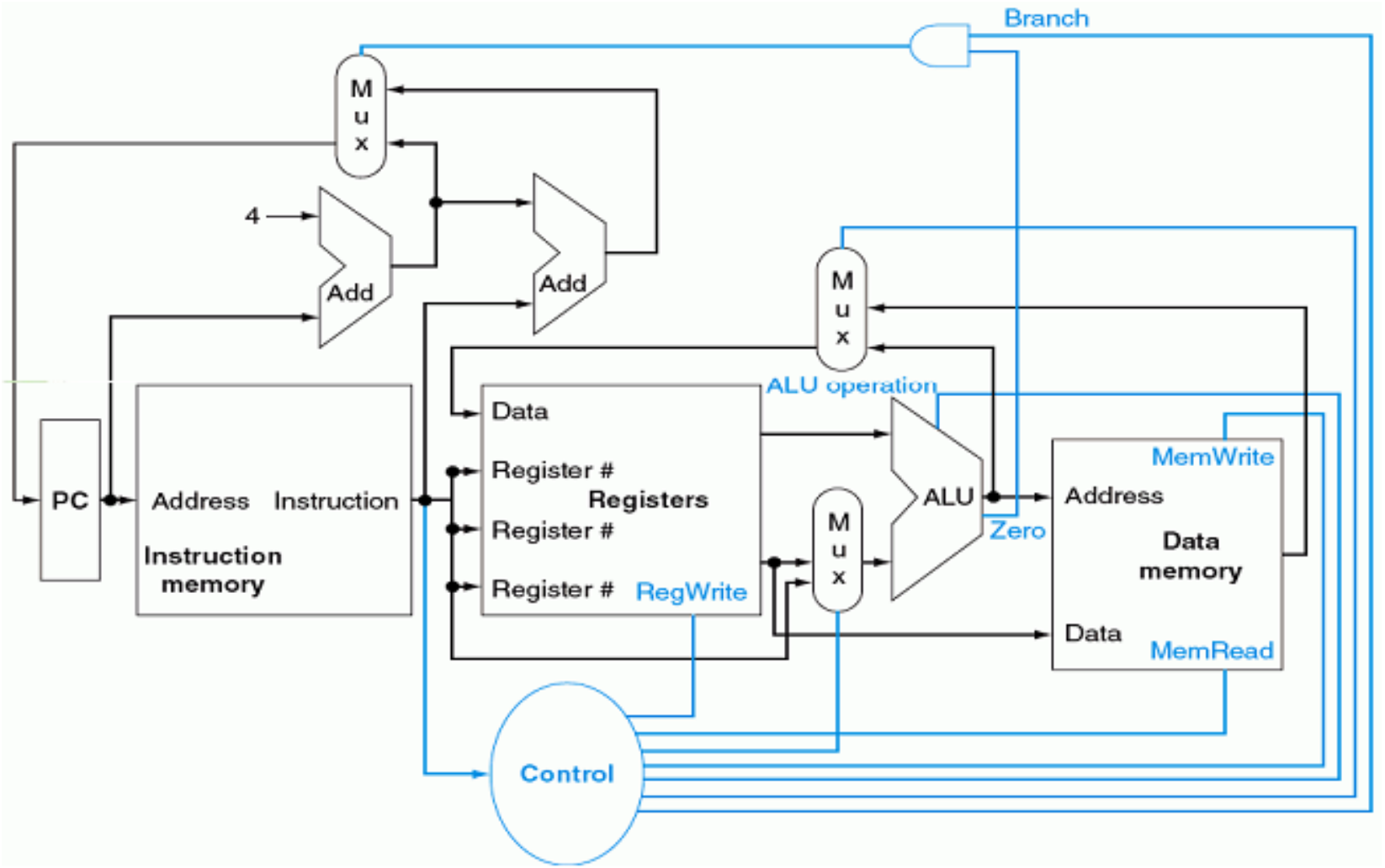




Dịch chuyển lệnh tiếp theo...



Sử dụng MUX để điều khiển

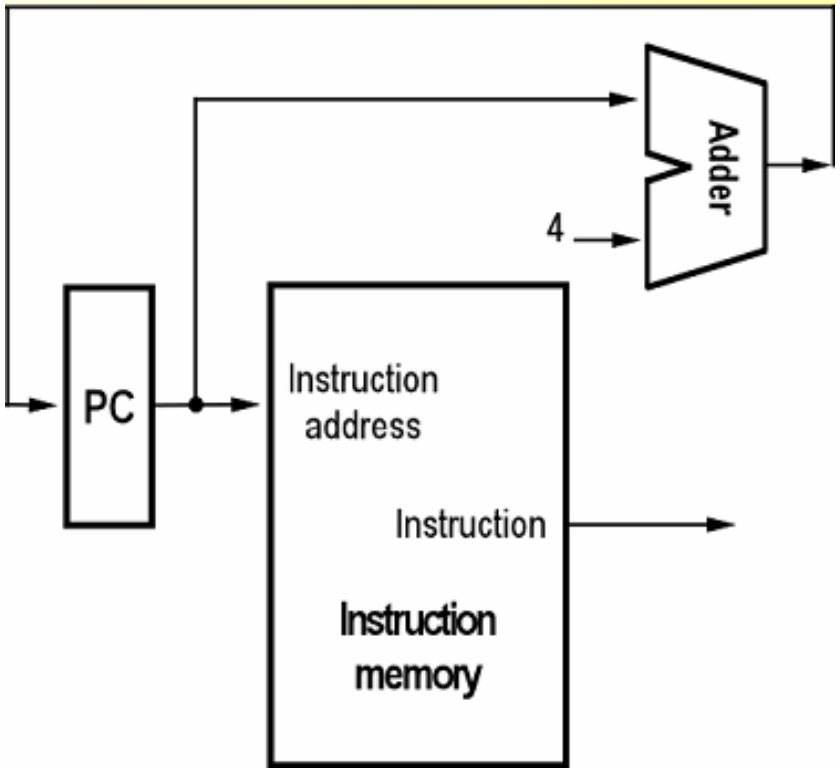


Xây dựng đường đi dữ liệu (Datapath)

- Phương thức xây dựng Datapath:
  - Xác định kiến trúc của các phần tử cần thiết cho câu lệnh
  - Xây dựng dần các phân khúc cho Datapath ứng với từng công đoạn trong thực thi câu lệnh
  - Tiến đến xây dựng hoàn chỉnh Datapath cho câu lệnh

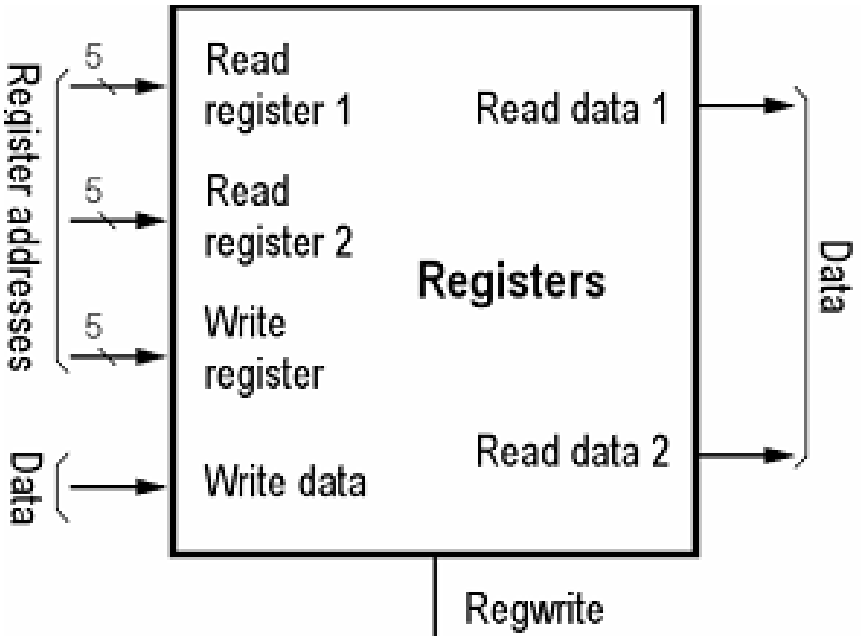
Kiến trúc các phần tử cần thiết

- Dịch chuyển lệnh:



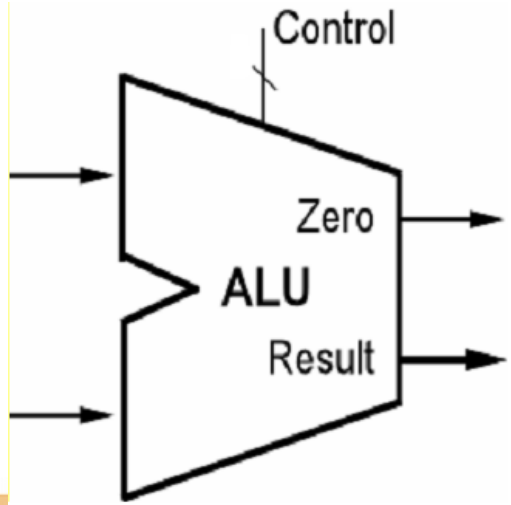
Kiến trúc các phần tử cần thiết

- Tập thanh ghi (register files)
  - 3 ngõ nhận địa chỉ thanh ghi
  - 1 ngõ ghi dữ liệu
  - 2 ngõ đọc dữ liệu (output)
  - 1 tín hiệu điều khiển ghi

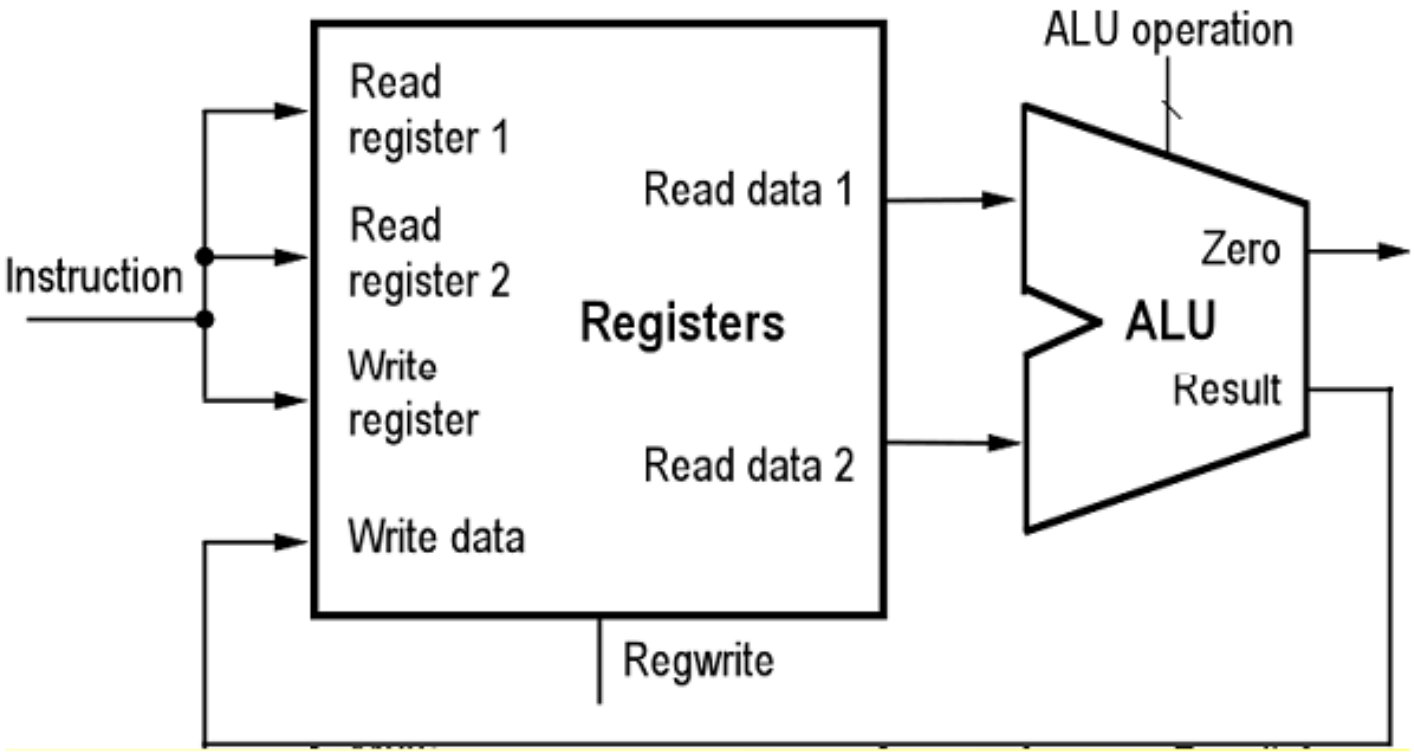


Kiến trúc các phần tử cần thiết

- Đơn vị số học – luận lý (ALU – Arithmetic Logic Unit)
  - 2 ngõ vào toán hạng (32-bit)
  - 1 ngõ ra kết quả (32 bit) và 1 bit zero (để chứa kết quả so sánh bằng)
  - 1 tín hiệu điều khiển (4 bit)



Register + ALU

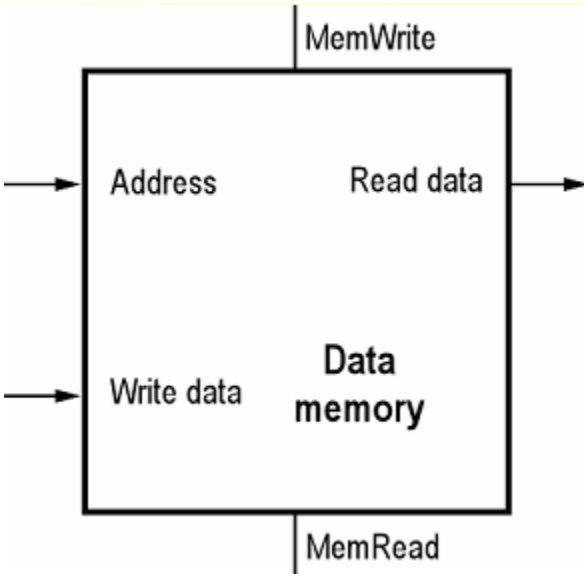


Datapath cho I,J-format (lw, sw, beq, j) ?

- Cần thêm 2 thành phần cơ bản:
  - Bộ nhớ dữ liệu (Data memory unit)
  - Bộ mở rộng dấu (Sign extended unit)

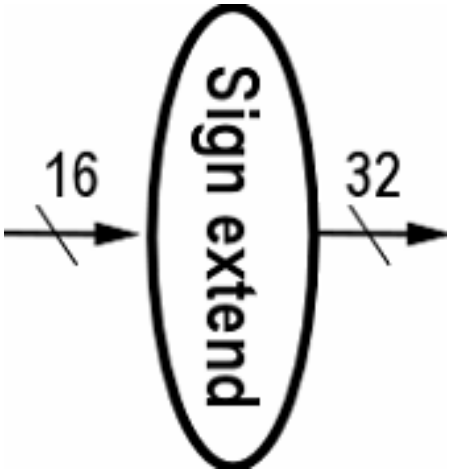
Datapath cho I,J-format (lw, sw, beq, j)

- Bộ nhớ dữ liệu (Data memory unit)
  - 1 ngõ nhận địa chỉ ô nhớ
  - 1 ngõ nhận dữ liệu cần ghi
  - 1 ngõ dữ liệu đọc (output)
  - 2 tín hiệu điều khiển đọc / ghi



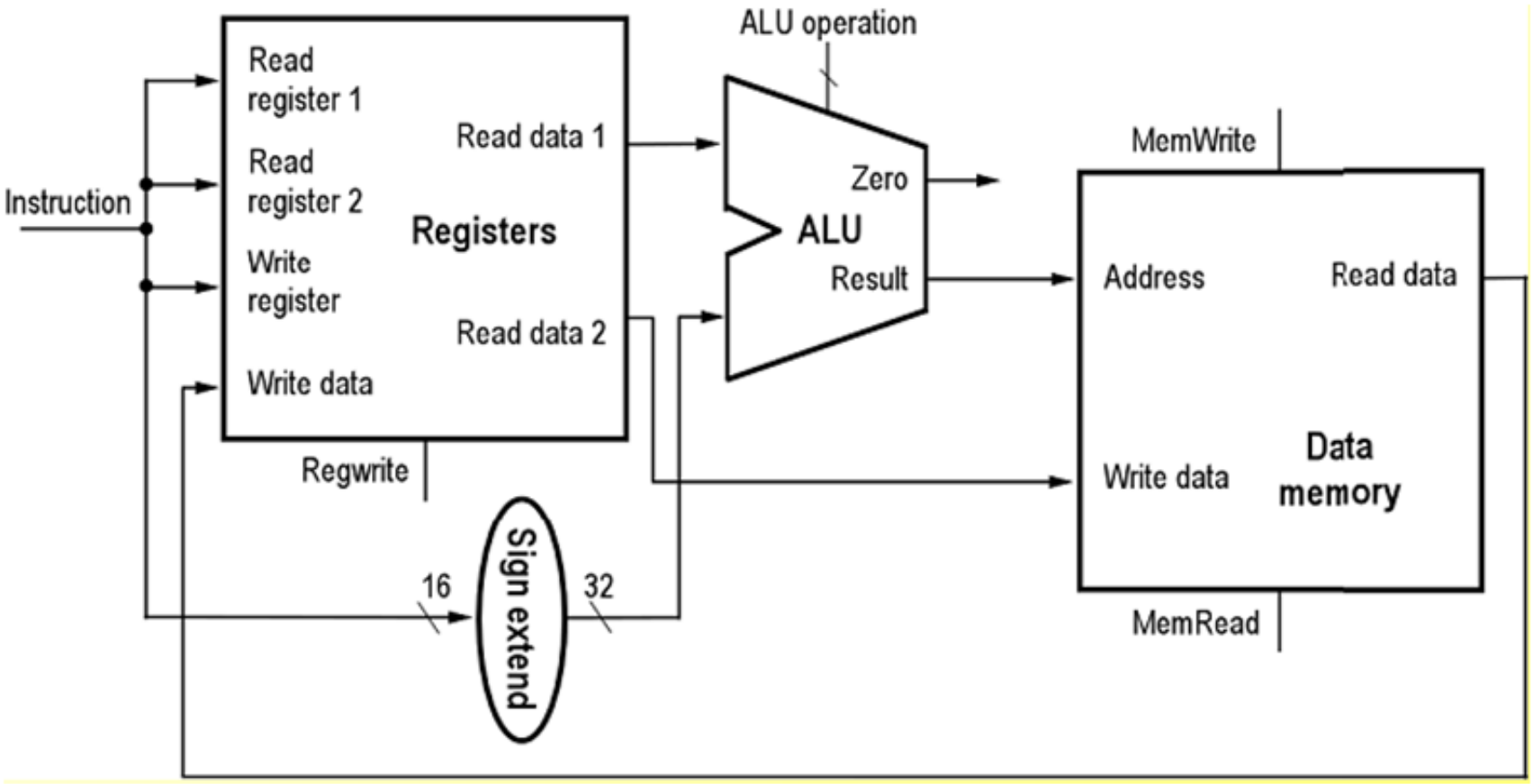
Datapath cho I-format (lw, sw, beq)

- Bộ mở rộng dấu (Sign extended unit)
  - 1 ngõ nhập dữ liệu 16-bit
  - 1 ngõ ra dữ liệu 32-bit

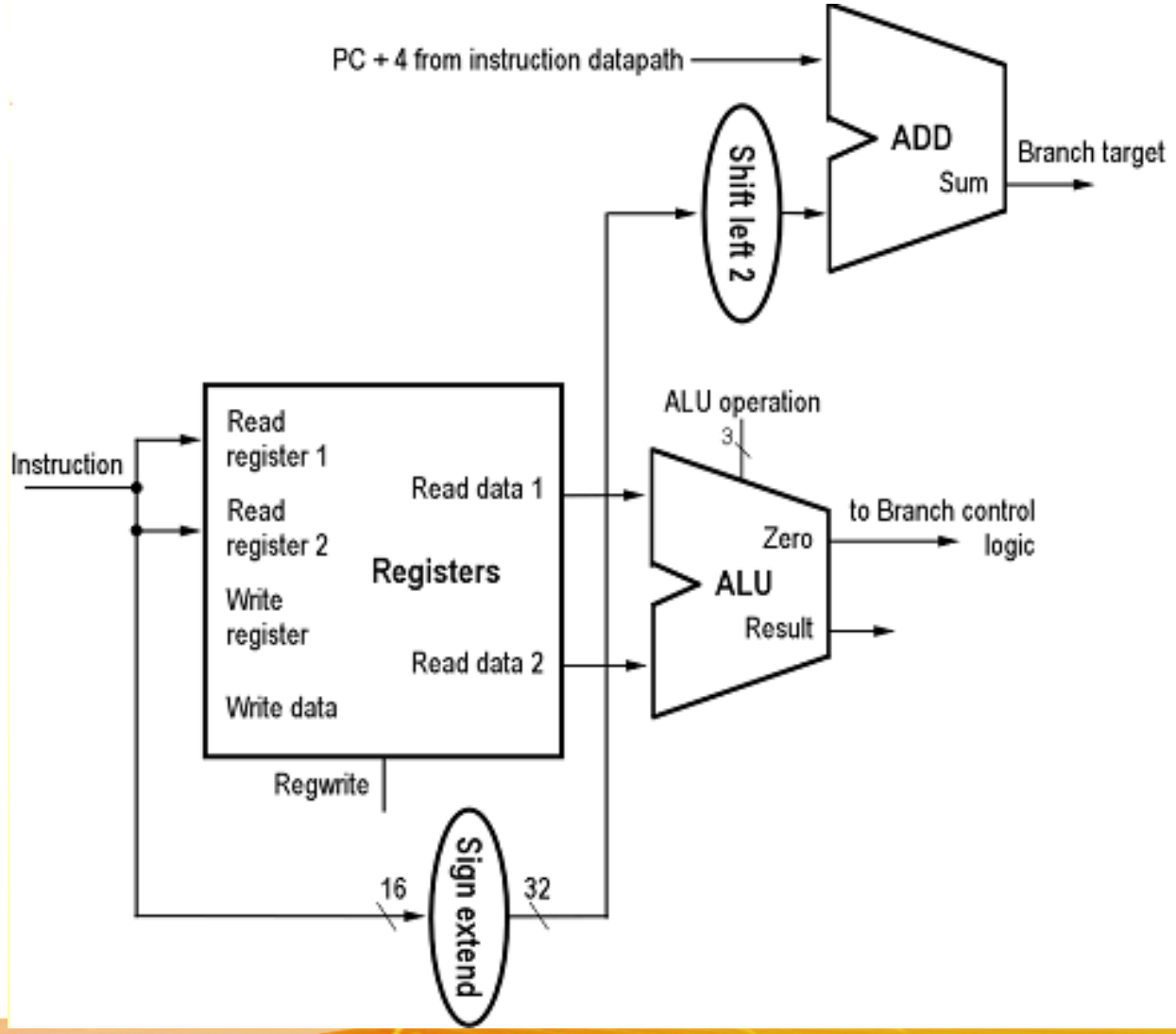


- lw $s1, 4($s0) → 4: 16 bit → 04: 32 bit (sign-extended)
- beq $s0, $s1, target_label → target-label: 16 bit → target-label: 32 bit (sign-extended)

Datapath cho lệnh bộ nhớ (lw,sw)



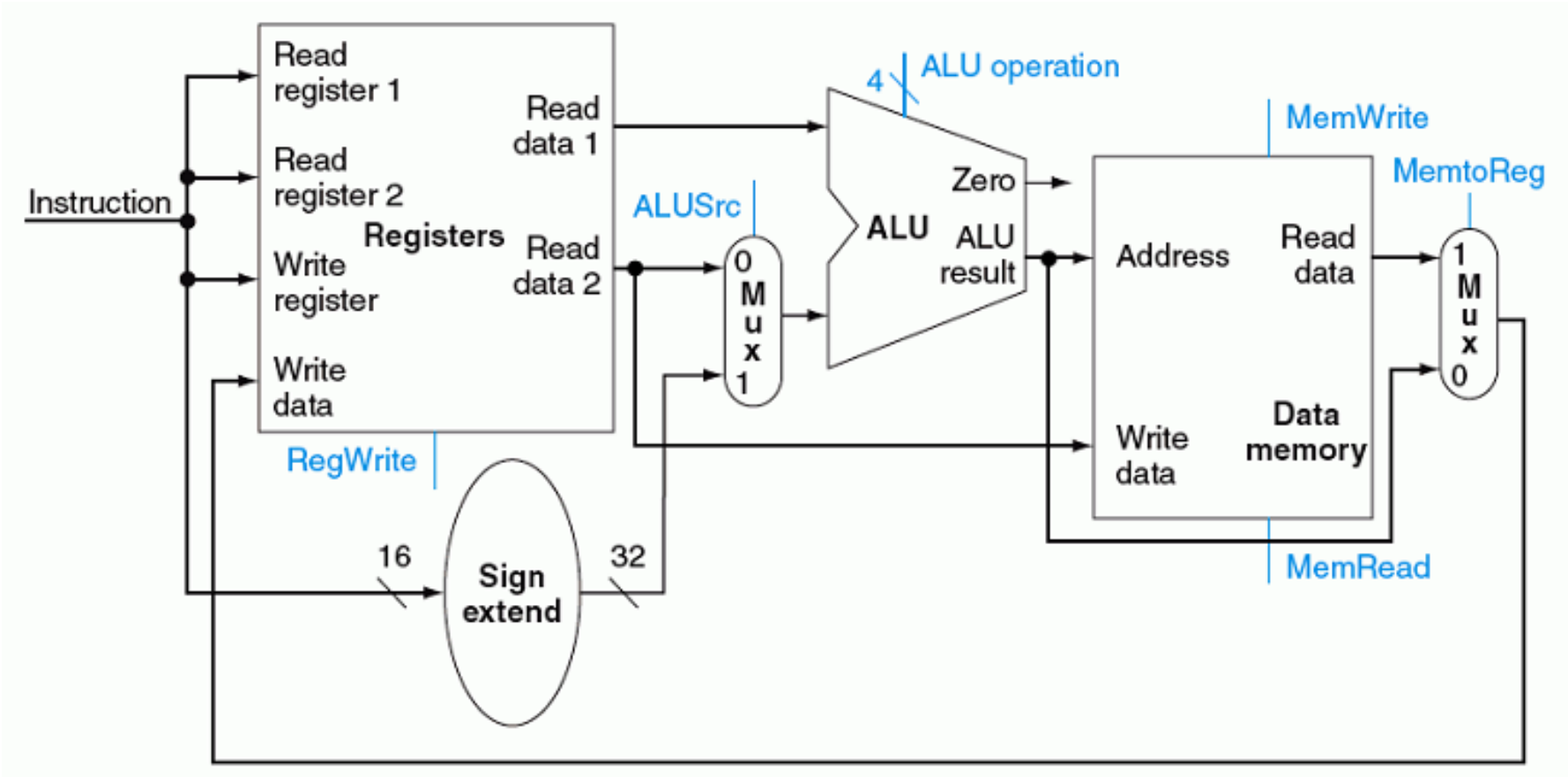
Datapath cho lệnh rẽ nhánh (beq,j)



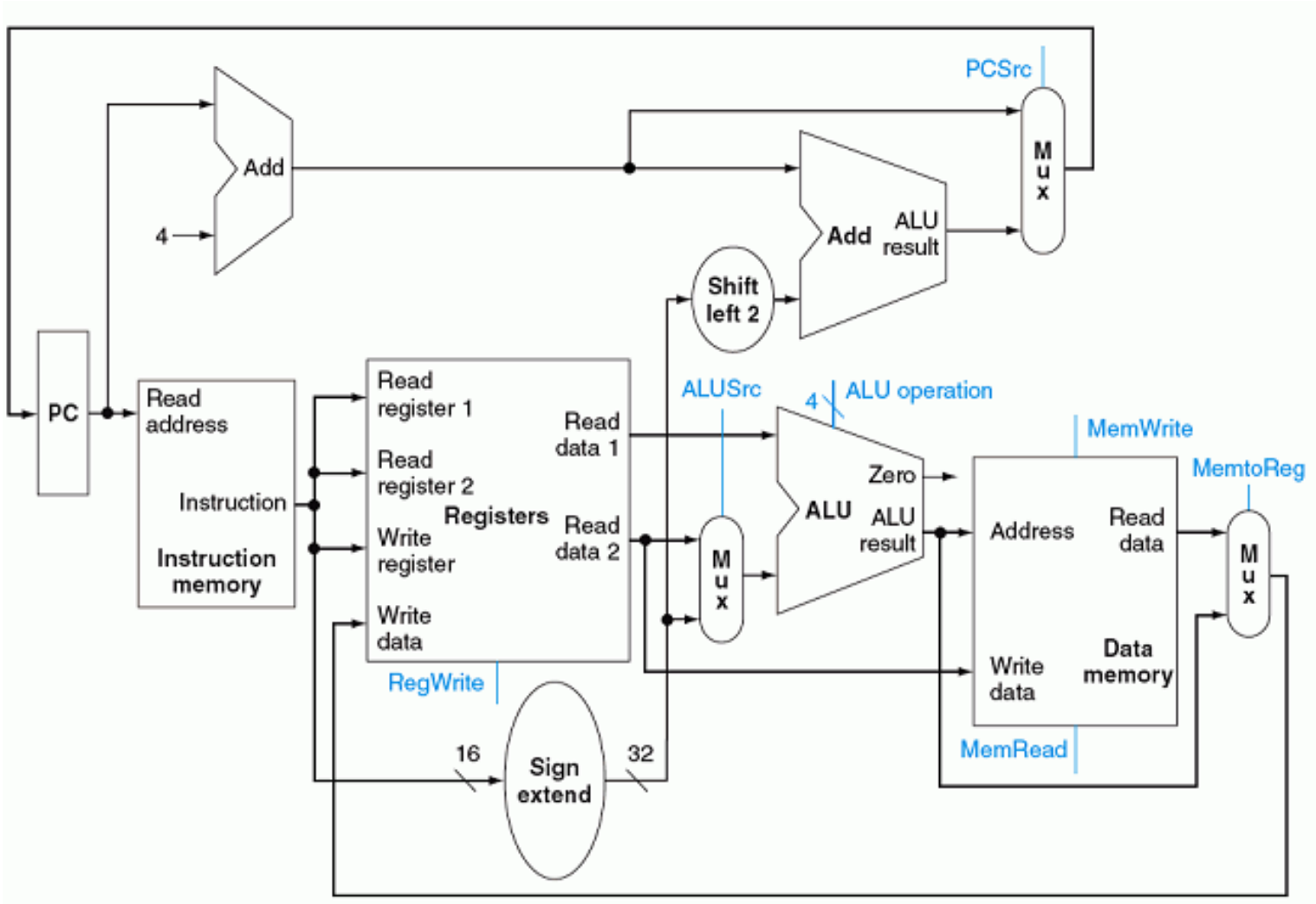
Datapath cho R-format ?

- Làm sao xây dựng Datapath cho R-format "xài chung" Datapath của I và J-format?
- Cần những bộ MUX đóng vai trò data selector để chia sẻ và lựa chọn những phần tử kiến trúc giữa những nhóm lệnh khác nhau
- Lưu ý: Hiện tại chúng ta chỉ xét CPU theo kiến trúc đơn chu kỳ (single cycle) – Mọi câu lệnh chỉ thực thi trong 1 chu kỳ clock

Datapath cho R-format



Datapath cho I,J,R-format

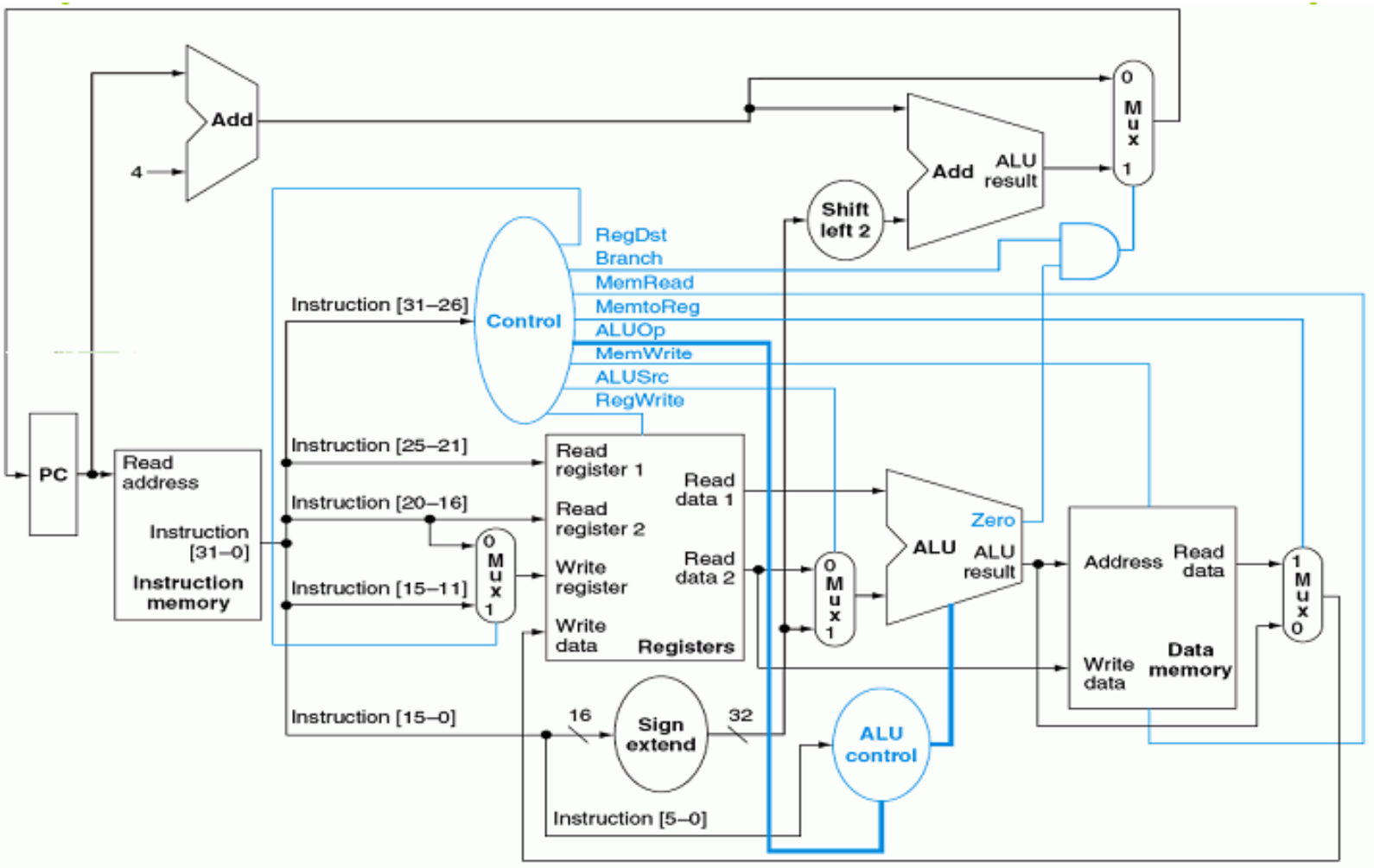


Tín hiệu điều khiển

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
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.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

Control unit ?

- ALU cần tín hiệu điều khiển hoạt động từ ALU Control



ALU Control Unit

- Các tín hiệu điều khiển ALU (4 bit):

ALU control Input	Function
0000	and
0001	or
0010	add
0110	sub
0111	slt
1100	nor

ALU Control Unit

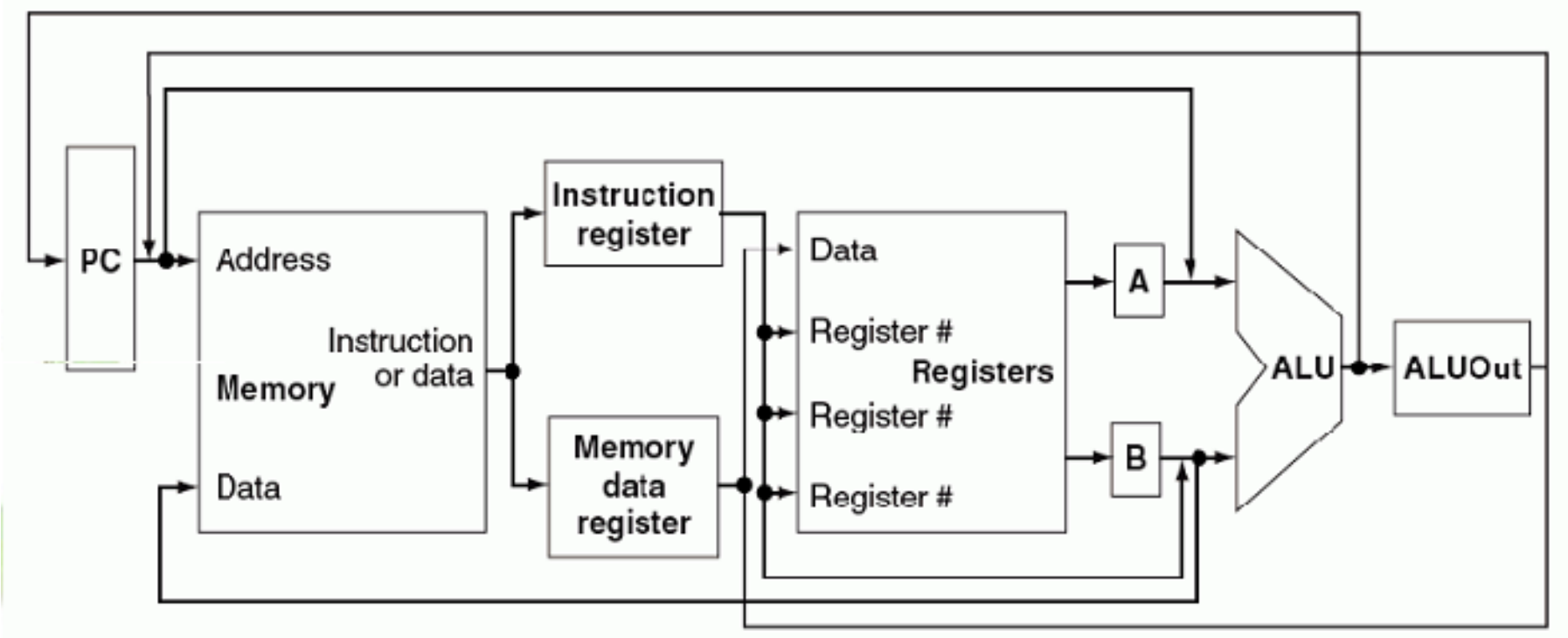
Instruction (Control Unit → ALU Control)			ALU control input (to ALU)
Operation	ALU Opcode	Function	
lw	00	xx xx xx	0010 (add)
sw	00	xx xx xx	0010 (add)
beq	01	xx xx xx	0110 (subtract)
add (R-type)	10	10 00 00	0010 (add)
subtract (R-type)	10	10 00 10	0110 (subtract)
and (R-type)	10	10 01 00	0000 (and)
or (R-type)	10	10 01 01	0001 (or)
slt (R-type)	10	10 10 10	0111 (slt)

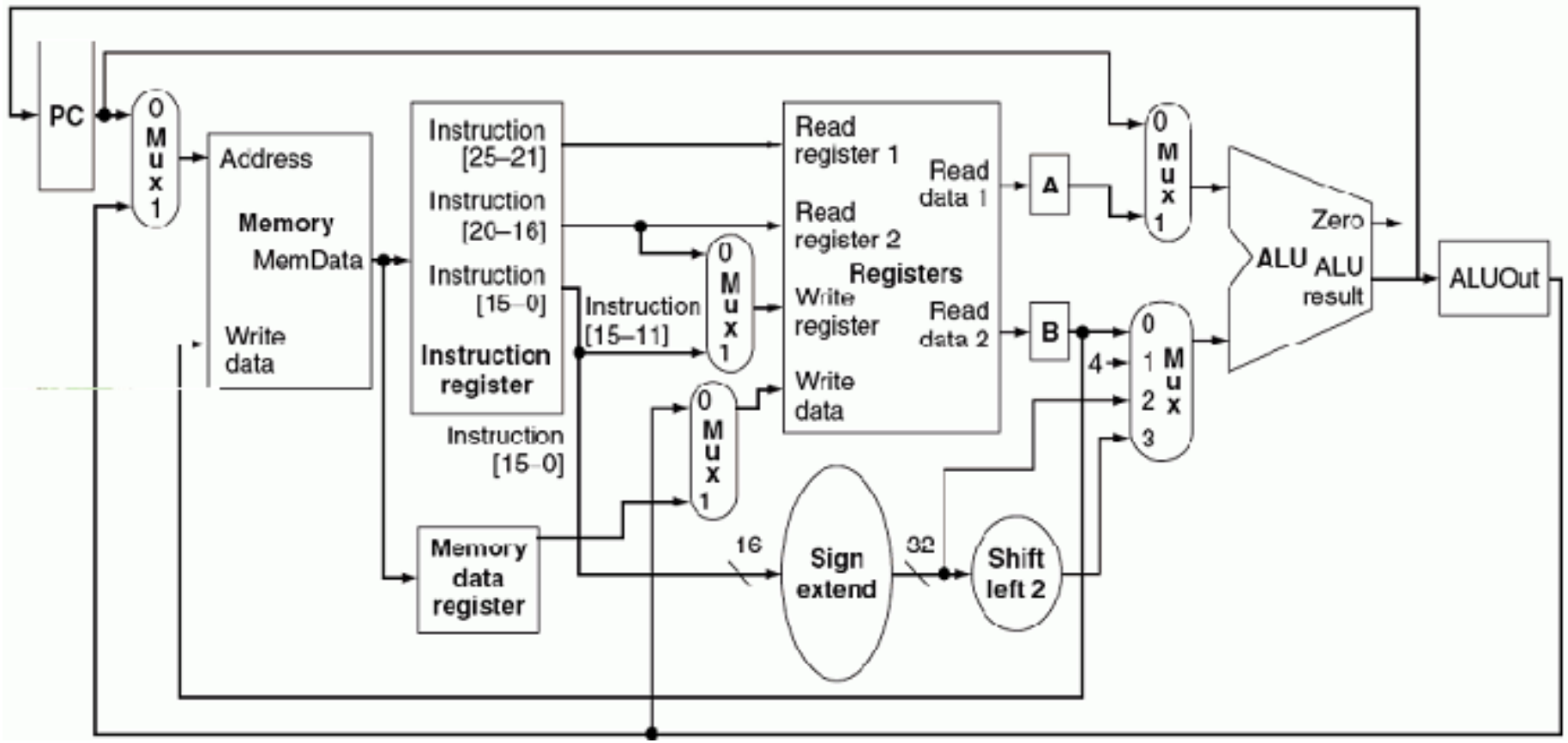
Bảng chân trị 4-bit ALU Control

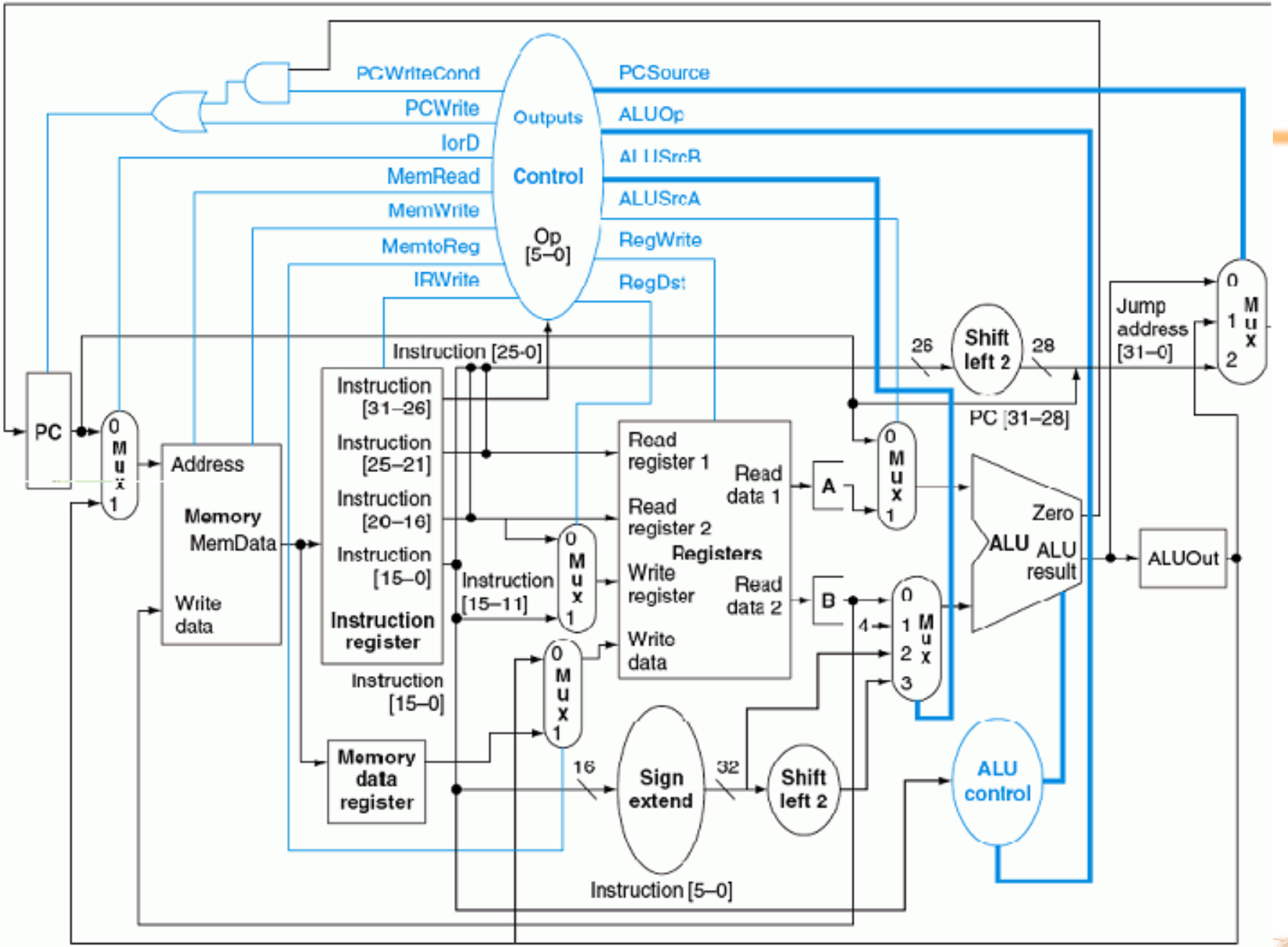
ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

CPU đa chu kỳ (multiple-cycle)

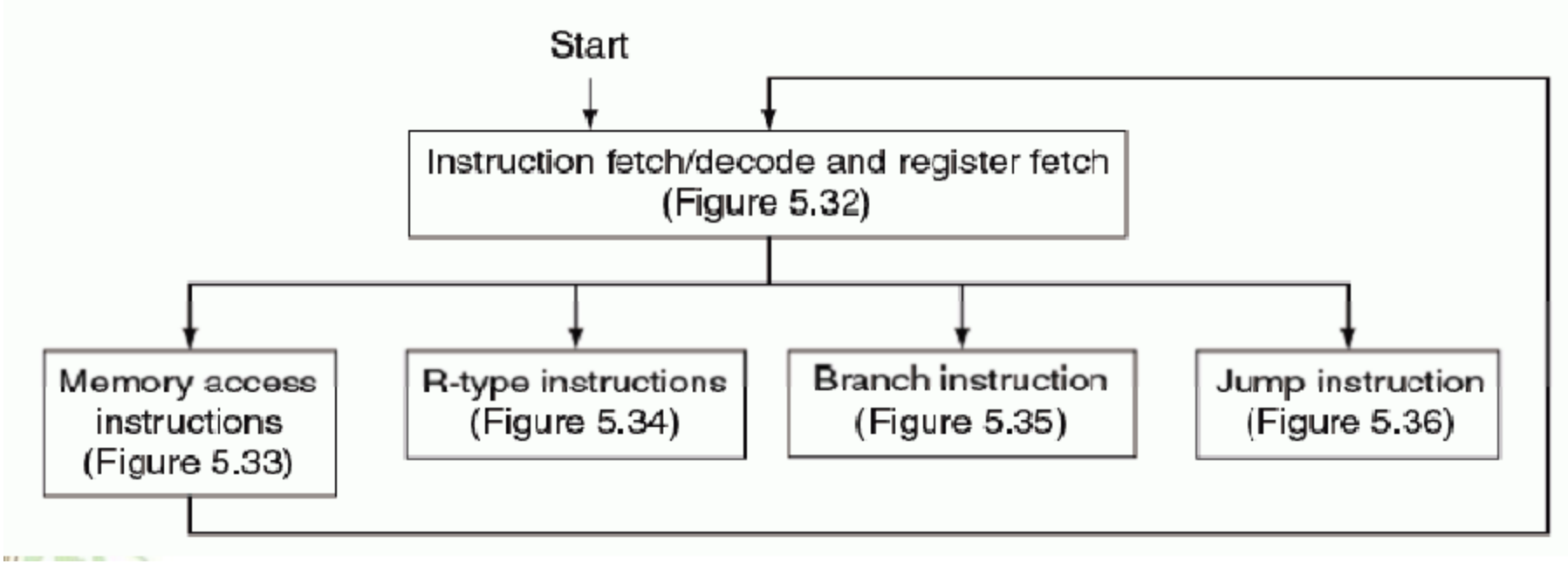
- Trong thực tế không sử dụng CPU single-cycle vì các lý do:
  - Thời gian thực hiện các câu lệnh luôn khác nhau → Phải chọn chu kỳ hoạt động của CPU bằng với chu kỳ thực thu câu lệnh dài nhất !
  - Khả năng trùng lắp các phần tử chức năng cao
- Ở CPU đa chu kỳ (multiple-cycle), quá trình thực thi 1 câu lệnh diễn ra thành nhiều chu kỳ clock
- Một số khác biệt so với single-cycle:
  - Tinh chỉnh thời gian thực thi từng câu lệnh theo giản đồ trạng thái
  - Có thể sử dụng 1 bộ nhớ chung cho cả câu lệnh lẫn dữ liệu
  - Thêm vào 1 số thanh ghi để chứa dữ liệu/kết quả trung gian

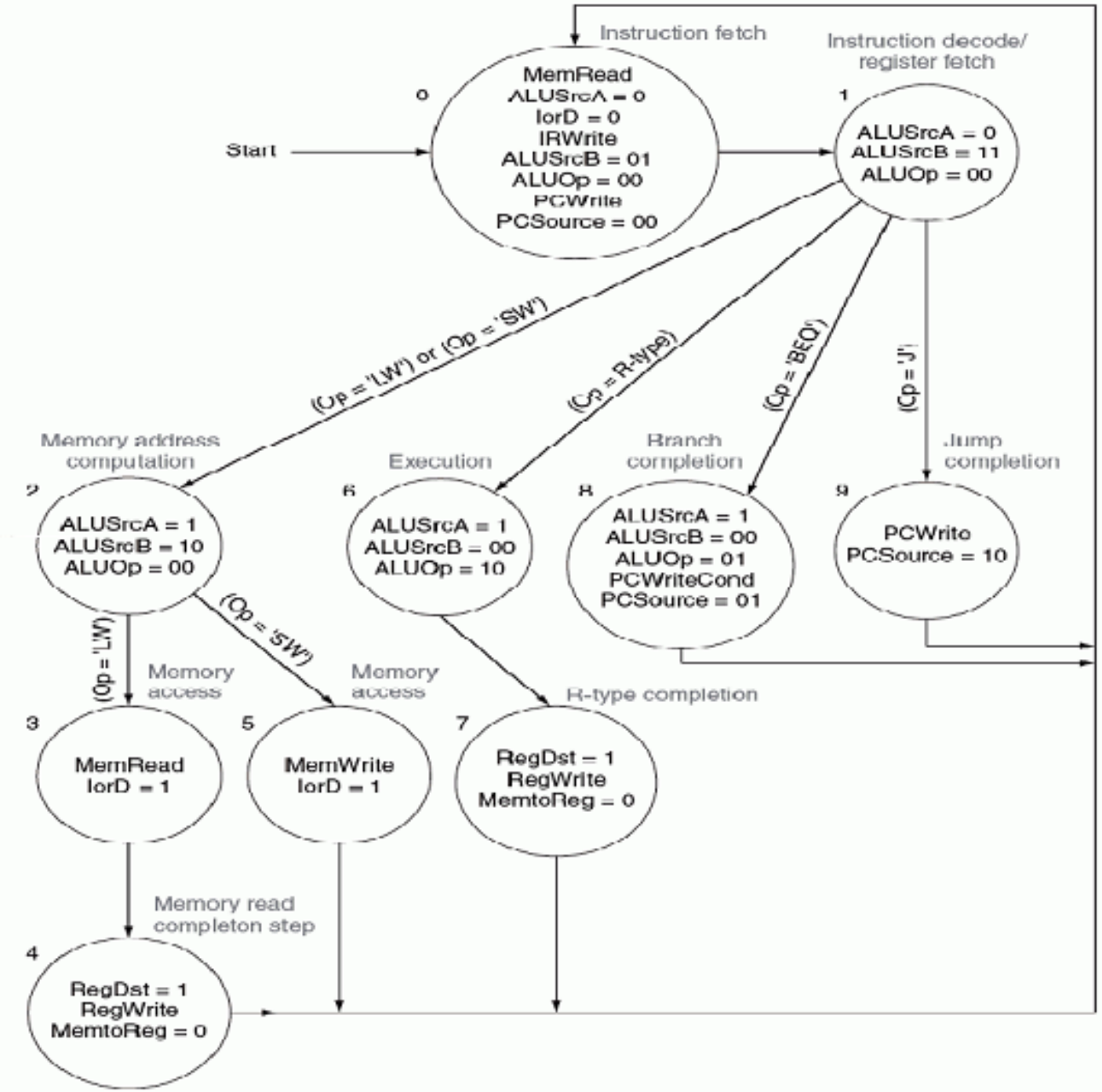

PC
Address
Instruction or data
Memory
Data
Instruction register
Memory data register
Data
Register #
Registers
Register #
Register #
A
B
ALU
ALUOut


PC
0
M
u
x
1
Address
Memory
MemData
Write
data
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Memory
data
register
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
16
Sign
extend
32
Shift
left 2
ALU
Zero
ALU
result
ALUOut


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op
[5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
Instruction [25-0]
26
Shift
left 2
28
PC [31–28]
Jump
address
[31–0]
M
u
x
0
1
2
PC
Address
Memory
MemData
Write
data
Instruction
[31–26]
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
Zero
ALU
ALU
result
ALUOut
Memory
data
register
16
Sign
extend
32
Shift
left 2
ALU
control
Instruction [5–0]

Quy trình thực thi lệnh




Instruction fetch
Instruction decode/
register fetch
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Start
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Memory address
computation
Execution
Branch
completion
Jump
completion
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
9
PCWrite
PCSource = 10
(Op = 'LW')
(Op = 'SW')
Memory
access
Memory
access
R-type completion
3
MemRead
IorD = 1
5
MemWrite
IorD = 1
7
RegDst = 1
RegWrite
MemtoReg = 0
Memory read
completon step
4
RegDst = 1
RegWrite
MemtoReg = 0

Homework

- Sách Petterson & Hennessy: Đọc chương 5