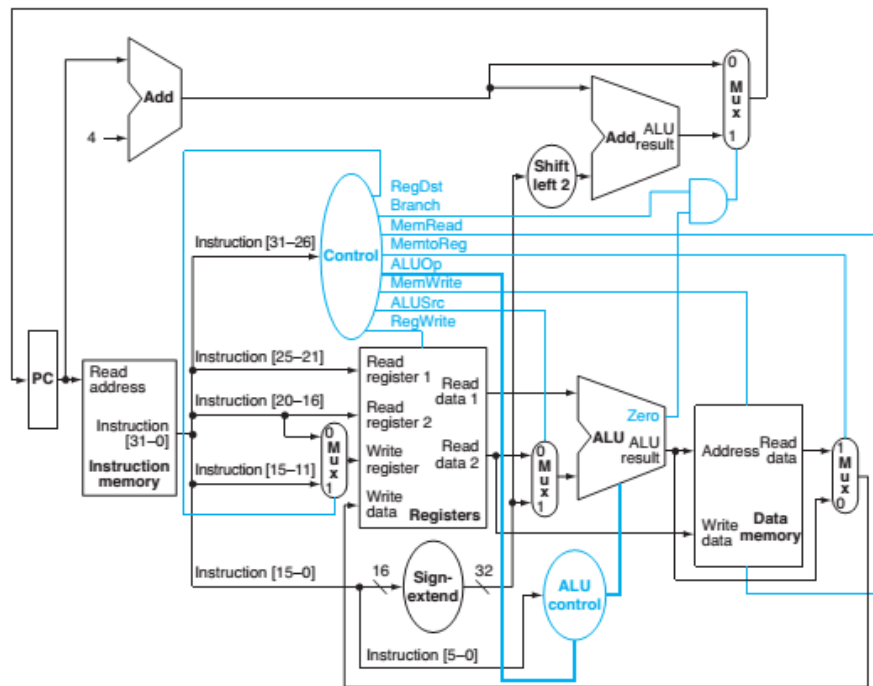


## BÀI TẬP CHƯƠNG 4 – DATAPATH



Hình 1.

### Bài 1. (4.1 – sách tham khảo)

Cho 2 lệnh như sau:

	Lệnh	Ý nghĩa
a.	add rd, rs, rt	$\text{Reg}[\text{rd}] = \text{Reg}[\text{rs}] + \text{Reg}[\text{rt}]$
b.	lw rt, offs(rs)	$\text{Reg}[\text{rt}] = \text{Mem}[\text{Reg}[\text{rs}] + \text{offs}]$

Với từng lệnh trong bảng này:

#### 1. Giá trị các tín hiệu điều khiển từ khối “Control” sẽ như thế nào?

	RegDst	RegWrite	MemRead	MemWrite	ALUOp	ALUSrc	MemToReg	Branch
a.	1	1	0	0	10	0 (Reg)	0 (ALU)	0
b.	0	1	1	0	00	1 (Imm)	1 (Mem)	0

#### 2. Các khối nào trong datapath hình 1 cần thiết, khối nào không cần thiết?

- Tất cả các khối đều cần thiết, ngoài trừ khối “Data Memory”, bộ cộng dùng cho lệnh nhảy, “shift left 2”, “sign-extend” và cổng logic AND
- Tất cả các khối đều được sử dụng, ngoài trừ bộ cộng dùng cho lệnh nhảy, “shift left 2” và cổng logic AND

#### 3. Khối nào trong datapath hình 1 có output đầu ra, nhưng output này không được sử dụng cho lệnh? Khối nào không có output?

Cho thời gian cần để hoàn thành của từng khối trong hình 1 như sau (khối nào không có trong bảng xem như thời gian cần để hoàn thành bằng 0):

	I-Mem	Add	Mux	ALU	Regs	D-Mem
a.	400ps	100ps	30ps	120ps	200ps	350ps
b.	500ps	150ps	100ps	180ps	220	1000ps

	Các khối có output, nhưng không sử dụng	Các khối không có output
a.	bộ cộng dùng cho lệnh nhảy, shift left 2, sign-extend, ngõ Zero của ALU	Data Memory
b.	bộ cộng dùng cho lệnh nhảy, shift left 2, ngõ Zero của ALU, ngõ Read Data2 của khối Registers	Không (Tất cả các khối đều có output)

**4. Tính thời gian cần để hoàn thành lớn nhất của lệnh “and” trong kiến trúc MIPS và cho biết “critical path” của lệnh?**

*Chú ý: “Critical path” của một lệnh là đường đi có thời gian trễ lớn nhất trong số các đường có thể khi lệnh thực thi.*

a. - Theo trường hợp 1:

Độ trễ lớn nhất:  $400 + 200 + 30 + 120 + 30 = 780\text{ps}$

Critical path: I-Mem, Mux, Regs, Mux, ALU, Mux

- Theo trường hợp 2:

Độ trễ lớn nhất:  $400 + 30 + 200 + 30 + 120 + 30 + 200 = 1010\text{ps}$

Critical path: I-Mem, Mux, Regs, Mux, ALU, Mux, Regs

b. Trường hợp 2:

Độ trễ lớn nhất:  $500 + 100 + 220 + 100 + 180 + 100 + 220 = 1420\text{ps}$

Critical path: I-Mem, Mux, Regs, Mux, ALU, Mux, Regs

**5. Tính thời gian cần để hoàn thành lớn nhất của lệnh “lw” trong kiến trúc MIPS và cho biết “critical path” của lệnh?**

a. Trường hợp 2:

Độ trễ lớn nhất:  $400 + 30 + 200 + 120 + 350 + 30 + 200 = 1330\text{ps}$

Critical path: I-Mem, Mux, Regs, ALU, D-Mem, Mux, Regs

b. Trường hợp 2:

Độ trễ lớn nhất:  $500 + 100 + 220 + 180 + 1000 + 100 + 220 = 2320\text{ps}$

Critical path: I-Mem, Mux, Regs, ALU, D-Mem, Mux, Regs

**6. Tính thời gian cần để hoàn thành lớn nhất của lệnh “beq” trong kiến trúc MIPS và cho biết “critical path” của lệnh?**

a. Độ trễ lớn nhất:  $400 + 200 + 30 + 120 + 30 = 780\text{ps}$

Critical path: I-Mem, Regs, Mux, ALU, Mux

b. Độ trễ lớn nhất:  $500 + 220 + 100 + 180 + 100 = 1100\text{ps}$

Critical path: I-Mem, Regs, Mux, ALU, Mux

**Bài 2. (4.2 – sách tham khảo)**

Giả sử tập lệnh có thêm hai lệnh mới như sau:

	Lệnh	Ý nghĩa
a.	add3 rd, rs, rt, rx	$\text{Reg}[\text{rd}] = \text{Reg}[\text{rs}] + \text{Reg}[\text{rt}] + \text{Reg}[\text{rx}]$
b.	sll rd, rt, shift	$\text{Reg}[\text{rd}] = \text{Reg}[\text{rt}] \ll \text{shift}$ (dịch trái shift bits)

Với từng lệnh trên:

**1. Khối nào đang có trong hình 1 có thể sử dụng cho các lệnh này?**

a. Lệnh này cần sử dụng các khối: instruction memory, Registers (cả 2 cổng đọc và cổng ghi), ALU

b. Lệnh này cần sử dụng các khối: instruction memory, Registers (nhưng chỉ 1 cổng đọc và cổng ghi), đường truyền số tức thời tới ALU

**2. Khối mới nào cần được thêm vào?**

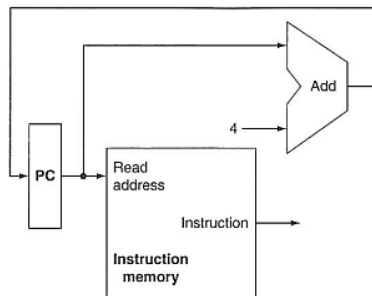
- Các khối mới cần được thêm vào: Thêm một cổng đọc vào khối Registers và thêm một ALU để tính tổng Rx với Rs + Rt (hoặc sửa ALU đang có thành ALU với 3 input)
  - Các khối mới cần được thêm vào: Đưa thêm tính năng dịch vào ALU hiện tại
- 3. Tín hiệu mới nào cần được thêm vào từ khối “Control” để hỗ trợ?**
- Các tín hiệu điều khiển mới cần thêm vào:
    - Thêm một tín hiệu điều khiển để điều khiển ALU mới (trong trường hợp ALU câu 2.2.a chọn thêm 1 ALU mới)
    - Thay đổi lại khối “ALU Control” để điều khiển ALU 3 đầu vào (trong trường hợp ALU câu 2.2.a chọn sửa lại ALU 2 đầu vào thành 3 đầu vào)
  - Các tín hiệu điều khiển mới cần thêm vào:
    - Thay đổi lại khối “ALU Control” để điều khiển ALU có thêm tính năng sll

**Bài 3. (4.6 – sách tham khảo)**

Giả sử các khối trong datapath (hình 1) có độ trễ như sau:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-left-2
a.	400ps	100ps	30ps	120ps	200ps	350ps	20ps	2ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	90ps	20ps

- 1. Giả sử việc duy nhất được thực hiện trong processor chỉ là nạp lệnh liên tục (như hình bên dưới), chu kỳ xung clock cần cho thiết kế là bao nhiêu?**



- 400ps
  - 500ps
- 2. Giả sử processor chỉ thực hiện duy nhất mỗi lệnh nhảy (như beq nhưng không cần điều kiện bằng), chu kỳ xung clock cần cho thiết kế là bao nhiêu?**
- Critical path cho lệnh này: instruction memory, sign-extend, shift-left-2, bộ cộng (để tính địa chỉ mới) và Mux.
- $400 + 20 + 2 + 100 + 30 = 552\text{ps}$
  - $500 + 90 + 20 + 150 + 100 = 860\text{ps}$
- 3. Như câu 2, nhưng lệnh nhảy trong trường hợp này có xét đến điều kiện bằng (như beq), chu kỳ xung clock cần cho thiết kế là bao nhiêu?**

Cho khối chức năng sau:

a.	Add 4 (bộ cộng dùng để cộng PC với 4)
b.	Data Memory

Ngoài đường dẫn tính địa chỉ mới cho lệnh nhảy (instruction memory, sign-extend, shift-left-2, bộ cộng, và Mux), còn một đường dẫn khác qua: instruction memory, Mux, Registers, Mux, ALU, Mux để tính điều kiện bằng (Các đường dẫn khác không đáng kể so với hai đường chính này)

Độ trễ của đường dẫn tính điều kiện bằng:

- $400 + 30 + 200 + 30 + 120 + 30 = 810\text{ps}$

b.  $500 + 100 + 220 + 100 + 180 + 100 = 1200\text{ps}$

Vì đường này có độ trễ dài hơn đường tính địa chỉ mới, nên chu kỳ xung clock cần cho thiết kế:

a. 810ps

b. 1200ps

**4. Dạng lệnh nào cần các khối chức năng trên**

a. Tất cả các lệnh, ngoài trừ các lệnh nhảy thuộc nhóm “not PC-relative” (jal, jalr, j, jr)

b. Các lệnh liên quan đến ‘load’ và ‘store’

**5. Dạng lệnh nào mà các khối chức năng trên nằm trong critical path?**

a. Không lệnh nào (Vì khối “Instruction memory” luôn có độ trễ cao hơn “Add 4” và tất cả các lệnh (bao gồm cả NOP) đều cần phải qua Instruction memory cho việc đọc lệnh).

b. ‘load’ và ‘store’

**Bài 4. (4.7 – Sách tham khảo)**

Cho độ trễ của các khối trong datapath như sau:

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-extend	Shift-left-2
a.	400ps	100ps	30ps	120ps	200ps	350ps	20ps	0ps
b.	500ps	150ps	100ps	180ps	220ps	1000ps	90ps	20ps

**1. Chu kỳ xung clock là bao nhiêu nếu datapath chỉ hỗ trợ các lệnh thuộc nhóm logic và số học (như add, and, ...)?**

Nếu datapath chỉ hỗ trợ các lệnh nhóm logic và số học thì chu kỳ xung clock sẽ bằng thời gian của lệnh của công đoạn dài nhất. Và các lệnh thuộc nhóm này có thời gian thực thi như nhau nên chúng ta có thể chọn lệnh “add” để tính chu kỳ:

Critical path: I-Mem, Mux, Regs, Mux, ALU, Mux, Regs

a. Tổng thời gian là:  $400\text{ps} + 30\text{ps} + 200\text{ps} + 30\text{ps} + 120\text{ps} + 30\text{ps} + 200\text{ps} = 1010\text{ps}$

Chu kỳ xung clock trong thiết kế này là 1010ps

b. Tổng thời gian là:  $500\text{ps} + 100\text{ps} + 220\text{ps} + 100\text{ps} + 180\text{ps} + 100\text{ps} + 220\text{ps} = 1420\text{ps}$

Chu kỳ xung clock trong thiết kế này là 1420ps

**2. Chu kỳ xung clock là bao nhiêu nếu datapath chỉ hỗ trợ lệnh lw?**

Nếu datapath chỉ hỗ trợ lệnh lw thì chu kỳ xung clock chính là thời gian thực thi lệnh lw

Critical path: I-Mem, Mux, Regs, Mux, ALU, D-Mem, Mux, Regs

a. Tổng thời gian là:  $400 + 30 + 200 + 30 + 120 + 350 + 30 + 200 = 1360\text{ps}$

Chu kỳ xung clock = 1360ps

b. Tổng thời gian là:  $500 + 100 + 220 + 100 + 180 + 1000 + 30 + 220 = 2420\text{ps}$

Chu kỳ xung clock = 2420ps

**3. Chu kỳ xung clock là bao nhiêu nếu datapath hỗ trợ các lệnh: add, beq, lw, sw?**

Giả sử tỉ lệ các lệnh được thực hiện trong một đoạn lệnh như sau (Processor không pipeline):

	add	addi	not	beq	lw	sw
a.	30%	15%	5%	20%	20%	10%
b.	25%	5%	5%	15%	35%	15%

Tương tự câu 2, chu kỳ xung clock của thiết kế là critical path của lệnh dài nhất trong tổng số các lệnh.

lw có critical path dài nhất trong số các lệnh trên.

**4. Bao nhiêu phần trăm chu kỳ xung clock có sử dụng khối “Data memory”?**

“Data memory” bị truy xuất chỉ với lw và sw

a.  $20\% + 10\% = 30\%$

b.  $35\% + 15\% = 50\%$

**5. Bao nhiêu phần chu kỳ xung clock có sử dụng khối “Sign-extend”?**

Thật sự khối “Sign-extend” đều có tính toán ra một kết quả nào đó trong mỗi chu kỳ, nhưng output của nó chỉ được cần cho các lệnh addi, beq, lw và sw; và bỏ qua với các lệnh còn lại. Vì vậy:

a.  $15\% + 20\% + 20\% + 10\% = 65\%$

b.  $5\% + 15\% + 35\% + 15\% = 70\%$

**Bài 5. (4.9 – Sách tham khảo)**

	Lệnh
a.	lw \$1, 40(\$6)
b.	label: beq \$1, \$2, label

**1. Mã máy của hai lệnh trên là gì?**

	Binary	Hexadecimal
a.	10001100110000010000000000101000	8CC10028
b.	00010000001000101111111111111111	1022FFFF

**2. Chỉ số cung cấp cho input “Read register 1”, “Read register 2” của khối “Registers” là gì? Các thanh ghi này có thật sự được đọc và được sử dụng không? (Xem datapath hình 1)**

	Read register 1	Thật sự được đọc và được sử dụng?	Read register 2	Thật sự được đọc và được sử dụng?
a.	6 (00110 <sub>(2)</sub> )	Được đọc, được sử dụng	1(00001 <sub>(2)</sub> )	Được đọc, nhưng <b>không</b> được sử dụng
b.	1(00001 <sub>(2)</sub> )	Được đọc, được sử dụng	2(00010 <sub>(2)</sub> )	Được đọc, được sử dụng

**3. Chỉ số cung cấp cho input “Write register” của khối “Registers” là gì? Thanh ghi này có thật sự được ghi vào không? (Xem datapath hình 1)**

	Write register 1	Thanh ghi thật sự được ghi không?
a.	1 (00001 <sub>(2)</sub> )	Được
b.	Hoặc là 2 (00010 <sub>(2)</sub> ) hoặc là 31 (11111 <sub>(2)</sub> ) (không biết vì tín hiệu RegDst là ‘x’ trong trường hợp này)	Không

**Bài 6.** Cho một kiến trúc máy tính MIPS với datapath và tín hiệu điều khiển như hình.

Đối với lệnh: *addi Rt, Rs, Imm*

**a) Lệnh addi chạy được với datapath như trên không? Những khối nào sẽ cần sử dụng cho lệnh trên, khối nào không cần sử dụng?**

Được.

Tất cả các block đều được sử dụng ngoài trừ Data memory, shift left 2, Add 2

**b) Cho biết giá trị của các tín hiệu điều khiển?**

RegDst	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
0	0	0	0	(Phụ thuộc vào thiết kế của khối)	0	1	1

				ALU Control			
--	--	--	--	----------------	--	--	--

c) Những khối nào có cho dữ liệu output nhưng dữ liệu này không sử dụng? Những khối nào không cho output?

- Những block có cho dữ liệu output nhưng dữ liệu này không sử dụng: **Bộ cộng thứ 2** (bộ cộng mà có một input qua khối shift\_left\_2 trước khi vào bộ cộng)
- Những block không cho output: Data memory

d) Giả sử có lệnh mới như sau “addi Rt, Rs, Rx, Imm” (ý nghĩa  $Rt = Rs + Rx + Imm$ ) thì phải thay đổi hay thêm vào hình trên những block nào?

- Việc cộng thực hiện trên 3 toán hạng nên: hoặc sử dụng thêm 1 ALU hoặc chỉnh sửa lại ALU đang có bằng cách đưa thêm một input thứ 3 vào
- Trường opcode (6 bits), 3 thanh ghi (mỗi thanh ghi 5 bits) □ số bits trống còn lại trong format lệnh trên là 11 bits.

➔ Vậy trường Imm có thể sử dụng bao nhiêu bits tùy vào thiết kế, nhưng không quá 11 bits này. Gọi n là số bits cho trường Imm

➔ Khối Sign-extend hiện tại là mở rộng có dấu từ số tức thời 16 bits thành 32 bits; vì vậy hoặc sử dụng thêm một khối Sign-extend với input là n bits hoặc chỉnh sửa khối Sign-extend sao cho có thể nhận cả input 16 bits và n bits

## Bài 7.

Một bộ xử lý MIPS 32 bits có datapath như hình và thực thi đoạn chương trình assembly như sau: (Biết khi bắt đầu thanh ghi  $\$t0 = 0x00000064$  và  $\$t1 = 0x100010FC$ )

or  $\$t9, \$zero, \$t0$

add  $\$s0, \$zero, \$t1$

sw  $\$t9, 12(\$s0)$

a. Giá trị output của khối “Instruction memory” là bao nhiêu khi bộ xử lý trên thực thi ở câu lệnh thứ 3?

Giá trị output của khối “Instruction memory” khi bộ xử lý trên thực thi ở câu lệnh thứ 3 là mã máy của lệnh “sw  $\$t9, 12(\$s0)$ ”:  $0xAE19000C = 10101110000110010000000000001100_{(2)}$

b. Khi bộ xử lý trên thực thi ở câu lệnh thứ 3, điền các giá trị cho các thanh ghi, tín hiệu điều khiển và các ngõ input/output của datapath theo yêu cầu của bảng sau:

Ngõ vào		Điều khiển		Kết quả	
Thanh ghi	Giá trị	Tín hiệu	Giá trị	Ngõ	Giá trị
Instruction[25-21]	16/10000 <sub>2</sub>	RegDst	X	ALUResult (của ALU)	0x10001108
Instruction [20-16]	25/11001 <sub>2</sub>	RegWrite	0	WriteData (của khối Registers)	X
Instruction [15-11]	0/00000 <sub>2</sub>	ALUSrc	1	WriteData (Của khối Data Memory)	0x64
ReadData1	0x100010FC	Branch	0		
ReadData2	0x00000064	MemtoReg	X		
		MemWrite	1		
		MemRead	0		

### Bài 8.

Một bộ xử lý MIPS 32 bits (có datapath và control như hình) thực thi đoạn chương trình assembly như sau:

*addi \$t0, \$t1, 8*

*lw \$s0, 4(\$t0)*

*sw \$t0, 4(\$t0)*

Biết khi bắt đầu thanh ghi PC = 0x400000;  $\$t1 = 0x10010000$ ;  $\$s0 = 0x00000001$ ; word nhớ tại địa chỉ 0x1001000c đang có nội dung (hay giá trị) bằng 0x0000ffff.

Khi bộ xử lý trên thực thi ở câu lệnh thứ hai, điền các giá trị (tín hiệu, input và output) cho từng khối vào bảng sau:

Tên khối	Ngõ	Giá trị ( <b>Sinh viên điền vào cột này</b> )
<b>Instruction Memory</b>	Read address	0x400004
	Instruction[31-0]	0x8d100004
<b>Registers</b>	Read register 1	01000 <sub>(2)</sub>
	Read register 2	10000 <sub>(2)</sub>
	Write register	10000 <sub>(2)</sub>
	Write data	0x0000ffff
	Read data 1	0x10010008
	Read data 2	0x00000001
<b>ALU</b>	Input thứ nhất của ALU	0x10010008
	Input thứ hai của ALU	4 <sub>(10)</sub>
	ALU result	0x1001000c
	Zero	0
<b>Data Memory</b>	Address	<b>0x1001000c</b>
	Write data	0x00000001
	Read data	0x0000ffff
<b>Control</b>	Instruction [31-26]	<b>100011</b> <sub>(2)</sub>
	RegDst	<b>0</b>
	Branch	<b>0</b>
	MemRead	<b>1</b>
	MemtoReg	<b>1</b>
	ALUOp ( <b>Chỉ cần cho biết ALU thực hiện phép toán gì</b> )	+
	MemWrite	<b>0</b>
	ALUSrc	<b>1</b>
	RegWrite	<b>1</b>

### Bài 9.

Cho một bộ xử lý MIPS 32 bits (có datapath và control như hình).

**Biết** PC = 0x400000;  $\$t1 = 0x00008000$ ;  $\$t3 = 0x00000015$ ; Word nhớ tại địa chỉ 0x00008008 có nội dung/giá trị bằng 0x00000015

Nếu đoạn chương trình sau được thực thi:

*addi \$s0, \$t1, 4*

*lw \$t2, 4(\$s0)*

*beq \$t3, \$t2, ABC*

*add \$t2, \$t3, \$t4*

*ABC: sub \$t3, \$t4, \$t5*

Khi bộ xử lý trên thực thi ở câu lệnh thứ ba, hỏi:

- a. Với khối “Instruction Memory” các ngõ “Read address” và “Instruction[31-0]” bằng bao nhiêu?  
“Read address” có giá trị là giá trị của PC khi tới câu lệnh thứ 3, mà PC ở lệnh thứ 1 khi bắt đầu chạy bằng 0x400000, vậy “Read address” lúc này bằng  $0x400000 + 0x8 = 0x400008$   
Giá trị ở ngõ ra instruction[31-0] là mã máy của lệnh thứ 3, tức mã máy của “beq \$t3, \$t2, 1”: 0x116a0001 = 0001 0001 0110 1010 0000 0000 0000 0001<sub>(2)</sub>
- b. Với khối “Registers”, các ngõ “Read register 1”, “Read register 2”, “Write register”, “Write data”, “Read data 1” và “Read data 2”, “RegWrite” bằng bao nhiêu?

Với khối “Registers”

Read register 1 nhận giá trị từ Instruction[25-21] = 01 011<sub>(2)</sub> = 11<sub>(10)</sub> (chỉ số thanh ghi \$t3)

Read register 2 nhận giá trị từ Instruction[20-16] = 0 1010<sub>(2)</sub> = 10<sub>(10)</sub> (chỉ số thanh ghi \$t2)

Vì khối “Registers” lúc này không thực hiện chức năng ghi nên giá trị đầu vào của Write register lúc này có thể là từ Instruction[20-16] hoặc từ Instruction[15-11] tùy vào tín hiệu điều khiển RegDst điều khiển MUX bằng 0 hay 1. Dù giá trị đưa vào là gì thì ngõ Write register cũng không được sử dụng với lệnh beq nên:

Write register = X

Tương tự, Write data = X

Read data 1 = nội dung thanh ghi có chỉ số 11, tức nội dung thanh ghi \$t3 = 0x15

Read data 2 = nội dung thanh ghi có chỉ số 10, tức nội dung thanh ghi \$t2 = 0x15

(Sau khi lệnh thứ nhất thực thi: \$s0 = 0x8004. Khi lệnh thứ hai thực hiện, lệnh này lấy nội dung của word nhớ tại địa chỉ  $4 + \$s0 = 0x8008$  đưa vào thanh ghi \$t2; mà đề cho nội dung word nhớ tại 0x8008 đang bằng 0x15, vậy sau khi lệnh thứ 2 thực hiện, thanh ghi \$t2 = 0x15)

RegWrite = 0

- c. Với khối “ALU”, input thứ 1, input thứ hai, “ALU result” và “zero” bằng bao nhiêu?

Input thứ nhất của ALU = Read data 1 = 0x15

Input thứ hai của ALU = Read data 2 = 0x15

ALU result của ALU = 0 (ALU thực hiện phép trừ hai input)

Zero = 1

- d. Với khối “Data Memory”, “Address”, “Write data”, “Read data”, “MemWrite”, “MemRead” bằng bao nhiêu?

Address của khối “Data memory” = ALU result = 0 (nhưng không sử dụng)

Write data của khối “Data memory” = Read data 2 của khối “Registers” = 0x15 (nhưng không sử dụng)

Read data = X

MemWrite = 0

MemRead = 0

- e. Các tín hiệu điều khiển của 3 MUX: RegDst, ALUSrc và MemToReg bằng bao nhiêu?

RegDst = X

ALUSrc = 0

MemToReg = X

- f. Đầu vào và đầu ra của khối “Sign-extend” bằng bao nhiêu?

Đầu vào của “Sign-extend” = Instruction [15-0] = 0000 0000 0000 0001<sub>(2)</sub>

Đầu ra của “Sign-extend” = 0000 0000 0000 0000 0000 0000 0000 0001<sub>(2)</sub>



**g. Đầu vào và đầu ra của khối “Shift left 2” bằng bao nhiêu?**

Đầu vào của “Shift left 2” = 0000 0000 0000 0000 0000 0000 0000 0001<sub>(2)</sub>

Đầu ra của “Shift left 2” = 0000 0000 0000 0000 0000 0000 0000 0100<sub>(2)</sub>

**h. Cổng “AND” trong trường hợp này có kết quả bằng bao nhiêu?**

Cổng “AND” nhận input thứ nhất là tín hiệu “Branch” từ khối “Control”; do lệnh đang thực hiện là lệnh beq nên Branch có giá trị 1. Input thứ hai của cổng “AND” là Zero từ khối ALU; do ALU thực hiện phép trừ có kết quả đang là 0 nên zero cũng đang bằng 1

AND có hai input đều là 1 nên đầu ra bằng 1.

**i. Ngõ “ALU Result” của bộ “Add” (mà có một đầu vào là kết quả của “Shift left 2”) có giá trị bao nhiêu?**

Ngõ “ALU Result” của bộ “Add” (mà có một đầu vào là kết quả của “Shift left 2”) =

$$400008_{(16)} + 4_{(10)} + 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100_{(2)}$$

$$= 400008_{(16)} + 4_{(16)} + 4_{(16)}$$

$$= 400010_{(16)}$$

**j. Thanh ghi PC cuối cùng có giá trị bao nhiêu?**

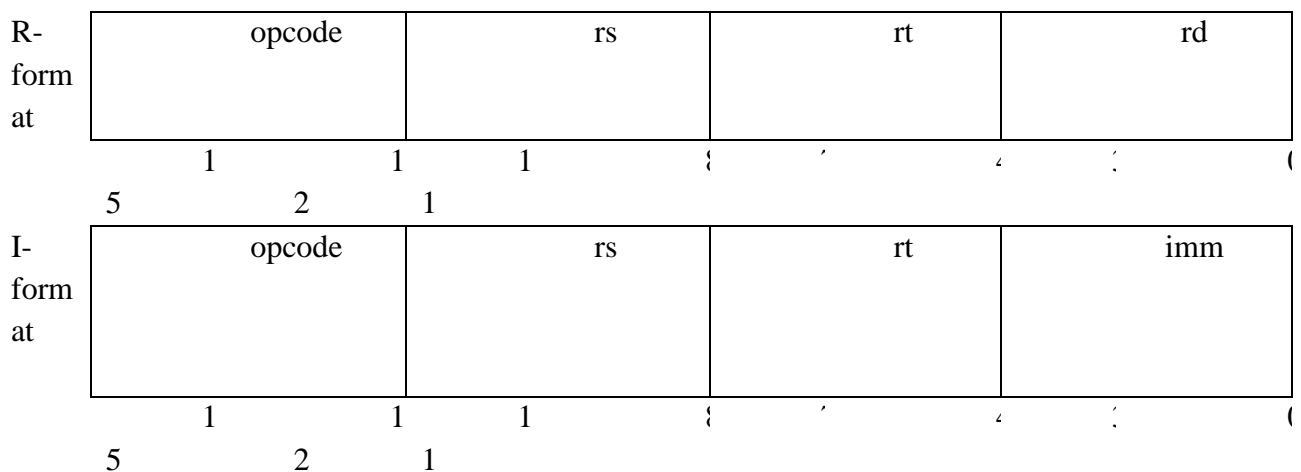
PC cuối cùng trong trường hợp này sẽ bằng 0x400010 (Vì cổng AND có kết quả bằng 1 → điều khiển MUX (sau bộ cộng Add) → PC nhận giá trị là ALU result của bộ cộng Add; nên PC = 0x400010)

**Bài 10.**

Cho một processor 16 bit có 4 lệnh như sau

Lệnh	Chức năng	Định
add rd, rs,	$R[rd] = R[rs] + R[rt]$	R-
addi rt, rs,	$R[rt] = R[rs] +$	I-
lw rt,	$R[rt] = M[R[rs] +$	I-
bne rs, rt,	$if(R[rd] \neq R[rs])\ PC = PC +$	I-

**Lưu ý:** rd, rs, rt: thanh ghi                      imm: số tức thời  
R[x]: giá trị thanh ghi x                      M[y]: nội dung của từ nhớ tại địa chỉ y  
SignExt(imm): mở rộng có dấu số tức thời imm từ 4 bit thành 16 bit



Với các khối cho sẵn như hình 3, vẽ thêm các đường cần thiết để hoàn chỉnh datapath cho processor có tập lệnh trên (Tại mỗi dấu “?” phải điền vào giá trị tương ứng)

- Có thể dùng thêm bộ MUX, bộ cộng, bộ dịch trái/phải và các loại cổng logic nếu cần
- ALU chỉ nhận input đầu vào là số 16 bit

Đầu ra Control sẽ nối tới ngõ điều khiển của MUX 1, MUX 2, MUX 3, và các ngõ RegWrite, ALUOperation, MemWrite, MenRead.