ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

MÔN HỌC: KIẾN TRÚC MÁY TÍNH (THỰC HÀNH) (CO2008)

Bài tập/Thực hành 7

CHƯƠNG 4 KIẾN TRÚC MIPS: PIPELINE

LỚP THỰC HÀNH L03 – HỌC KỲ 212

Giảng viên hướng dẫn: Vũ Trọng Thiên

Sinh viên thực hiện	Mã số sinh viên
Phạm Duy Quang	2011899

Thành phố Hồ Chí Minh, tháng 05 năm 2022

Bài tập và Thực hành

Bài 1: Xác định clock cycle

Cho thời gian delay của các khối như Bảng 1

Bảng 1: Delay của các khối phần cứng

Phần cứng	Delay (ns)
Instruction memory	150
Register	100
ALU	100
Data memory	150
Các bộ phần cứng khác	0

Xét đoạn chương trình như sau:

```
1         addi $t1, $zero, 100
2         addi $t2, $zero, 0
3  loop:
4         beq  $t1, $t2, exit
5         addi $t1, $t1, -1
6         addi $t2, $t2, 1
7         j loop
```

(a) Xác định clock cycle của hệ thống single clock, multi clock và pipeline clock.

Clock cycle :

- Single clock: Mỗi lệnh tương đương với 1 clock cycle. Clock clycle chính là số lệnh thực thi.

$$Clock\ cycle = 2 + 50 \times 4 + 1 = 203$$

- Multi – clock: lệnh addi thực thi 4 clock cycle, lệnh beq thực thi 3 clock cycle, lệnh j thực thi 2 clock cycle.

$$Clock\ cycle = 4 \times 2 + (3 + 4 \times 2 + 2) + 50 \times 3 = 661$$

- Pipeline clock: Mỗi lệnh thực thi 5 chu kỳ còn các lệnh mỗi chu kỳ hoàn thành xong 1 lệnh.

$$Clock\ cycle = 203 + 4 = 207$$

(b) Xác định thời gian thực thi của chương trình trên khi chạy với hệ thống single cycle, multi cycle và pipeline cycle (không xét stall).

Thời gian thực thi: $T = Clock\ cycle\ \times Clock\ cycle\ time$

- Single cycle:

$$Clock\ cycle\ time = 150 + 100 + 100 + 150 = 500\ ns$$

$$T_{single} = 203 \times 500 = 101500\ ns$$

- Multi – cycle :

$Clock\ cycle\ time = 150\ ns$ (thời gian thực thi dài nhất của các khối phần cứng)

$$T_{multi} = 661 \times 150 = 99150\ ns$$

- Pipeline cycle:

$Clock\ cycle\ time = 150\ ns$ (thời gian thực thi dài nhất của các khối phần cứng).

$$T_{pipeline} = 207 \times 150 = 31050\ ns$$

(c) Tính speed up của hệ thống pipeline với hệ thống multi cycle và với single cycle.

Speed up pipeline và multi cycle:

$$Speed\ up = \frac{T_{multi}}{T_{pipeline}} = \frac{99150}{31050} = 3.193$$

Speed up pipeline và single cycle:

$$Speed\ up = \frac{T_{single}}{T_{pipeline}} = \frac{101500}{31050} = 3.2689$$

(d) Khi delay ALU thay đổi từ 100 –> 150. Tính lại kết quả câu a, b, c.

Câu a: Clock cycle không thay đổi.

Câu b: Thời gian thực thi của hệ thống single cycle thay đổi, multi cycle và pipeline cycle không thay đổi.

$$Clock\ cycle\ time = 150 + 100 + 150 + 150 = 550\ ns$$

$$T_{single} = 203 \times 500 = 111650\ ns$$

Câu c: Speed up của pipeline và single cycle thay đổi:

$$Speed\ up = \frac{T_{single}}{T_{pipeline}} = \frac{111650}{31050} = 3.6$$

Bài 2: Xử lý Hazard.

Dùng lại đoạn code của **Bài 1**:

(a) Xác định sự phụ thuộc dữ liệu trong đoạn chương trình trên.

Lệnh beq thứ ba, tức beq($t2), beq($t1) chưa kịp lấy dữ liệu từ WB của 2 lệnh addi ở trên về dữ liệu của 2 thanh ghi $t1 và $t2 ở dòng 1 và 2.

Sau khi chạy loop thì tiếp tục lệnh beq($t2) bị hazard khi lấy dữ liệu của thanh ghi $t2 để thực hiện bước tiếp theo trên bộ ALU.

(b) Giải quyết data hazard bằng chèn stall (giải quyết bằng phần mềm), khi thực thi đoạn code trên với hệ thống pipeline thì cần chèn vào bao nhiêu stall (khựng lại) ?

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18
addi $t1,$zero,100	IF	ID	EX	MEM	WB													
addi $t2,$zero,0		IF	ID	EX	MEM	WB												
loop:																		
beq $t1,$t2,exit			IF	ID	EX	MEM	WB											
				IF	ID	EX	MEM	WB										
			Stall	Stall	IF	ID	EX	MEM	WB									
addi $t1,$t1,-1						IF	ID	EX	MEM	WB								
addi $t2,$t2,1							IF	ID	EX	MEM	WB							
j loop								IF	ID	EX	MEM	WB						
loop:																		
beq $t1,$t2,exit									IF	ID	EX	MEM	WB					
									Stall	IF	ID	EX	MEM	WB				
addi $t1,$t1,-1											IF	ID	EX	MEM	WB			
addi $t2,$t2,1												IF	ID	EX	MEM	WB		
j loop													IF	ID	EX	MEM	WB	

Dựa vào hình thì khi dùng pipeline ta cần chèn thêm 2 stall để giải quyết data hazard.

```
        addi $t1, $zero,100
        addi $t2, $zero, 0
        NOP
loop:
        NOP
        beq  $t1, $t2, exit
        addi $t1, $t1, -1
        addi $t2, $t2, 1
        j loop
```

(c) Dùng cơ chế forward để giải quyết hazard (giải quyết bằng phần cứng), khi đó có bao nhiêu stall? Vẽ hình minh họa.

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17
addi $t1,$zero,100	IF	ID	EX	MEM	WB												
addi $t2,$zero,0		IF	ID	EX	MEM	WB											
loop:																	
beq $t1,$t2,exit			IF	ID	EX	MEM	WB										
			Stall	IF	ID	EX	MEM	WB									
addi $t1,$t1,-1					IF	ID	EX	MEM	WB								
addi $t2,$t2,1						IF	ID	EX	MEM	WB							
j loop							IF	ID	EX	MEM	WB						
loop:																	
beq $t1,$t2,exit								IF	ID	EX	MEM	WB					
addi $t1,$t1,-1									IF	ID	EX	MEM	WB				
addi $t2,$t2,1										IF	ID	EX	MEM	WB			
j loop											IF	ID	EX	MEM	WB		

Khi dùng cơ chế forwading để giải quyết hazard ta thấy rằng ta cần chèn thêm 1 stall như trên.

```
        addi $t1, $zero,100
        addi $t2, $zero,0
        NOP
loop:
        beq  $t1, $t2, exit
        addi $t1, $t1,-1
        addi $t2, $t2,1
        j loop
```

Khi đó có 1 stall.

(d) Dùng cơ chế forward, stall để giải quyết hazard (control và data), khi đó có bao nhiêu stall?

- Data hazard ta giải quyết giống câu c. Control Hazard ta phải cần 2 stall để biết lệnh nào sẽ được thực thi tiếp theo.

- Cần $1 + 50 \times 2 = 101$ stall.

(e) Ngoài 2 cơ chế ở trên, ta có thể giảm stall bằng cách sắp xếp lại thứ tự code (giải quyết bằng trình biên dịch compiler). Hãy sắp xếp lại code sao cho ít stall nhất.

Việc sắp xếp lại lệnh không làm giảm số stall cần chèn vào ban đầu.

Bài 3: Xử lý Hazard (lệnh load)

Cho đoạn code sau:

```
1  addi $t1, $zero, 100
2  addi $t2, $zero, 100
3  add  $t3, $t1,   $t2
4  lw   $t4, 0($a0)
5  lw   $t5, 4($a0)
6  and  $t6, $t4, $t5
7  sw   $t6, 8($a0)
```

Trả lời câu hỏi trong **Bài 2**:

(a) Xác định sự phụ thuộc dữ liệu trong đoạn chương trình trên.

- Lệnh add($t1), add($t2) chưa kịp nhận giá trị hai thanh ghi $t1, $t2 từ hai câu lệnh addi ở dòng 1 và dòng 2.

- Lệnh and($t4), and($t5) chưa kịp nhận giá trị hai thanh ghi $t4, $t5 từ hai câu lệnh lw ở dòng 4 và dòng 5.

- Lệnh sw($t6) chưa kịp nhận giá trị từ thanh ghi $t6 của câu lệnh and ở dòng 6.

(b) Giải quyết data hazard bằng chèn stall (giải quyết bằng phần mềm), khi thực thi đoạn code trên với hệ thống pipeline thì cần chèn vào bao nhiêu stall (khựng lại) ?

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17
addi $t1,$zero, 100	IF	ID	EX	MEM	WB												
addi $t2,$zero, 100		IF	ID	EX	MEM	WB											
add $t3,$t1,$t2			IF	ID	EX	MEM	WB										
				IF	ID	EX	MEM	WB									
			Stall	Stall	IF	ID	EX	MEM	WB								
lw $t4, 0($a0)						IF	ID	EX	MEM	WB							
lw $t5, 4($a0)							IF	ID	EX	MEM	WB						
and $t6,$t4,$t5								IF	ID	EX	MEM	WB					
									IF	ID	EX	MEM	WB				
								Stall	Stall	IF	ID	EX	MEM	WB			
sw $t6, 8($a0)											IF	ID	EX	MEM	WB		
												IF	ID	EX	MEM	WB	
											Stall	Stall	IF	ID	EX	MEM	WB

Dựa vào hình thì khi dùng pipeline ta cần chèn thêm 6 stall để giải quyết data hazard.

addi $t1, $zero, 100

addi $t2, $zero, 100

NOP (Stall 1)

NOP (Stall 2)

add $t3, $t1, $t2

lw $t4, 0($a0)

lw $t5, 4($a0)

NOP (Stall 3)

NOP (Stall 4)

and $t6, $t4, $t5

NOP (Stall 5)

NOP (Stall 6)

sw $t6, 8($a0)

(c) Dùng cơ chế forward để giải quyết hazard (giải quyết bằng phần cứng), khi đó có bao nhiêu stall? Vẽ hình minh họa.

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14
addi $t1,$zero, 100	IF	ID	EX	MEM	WB									
addi $t2,$zero, 100		IF	ID	EX	MEM	WB								
add $t3,$t1,$t2			IF	ID	EX	MEM	WB							
lw $t4, 0($a0)				IF	ID	EX	MEM	WB						
lw $t5, 4($a0)					IF	ID	EX	MEM	WB					
and $t6,$t4,$t5						IF	ID	EX	MEM	WB				
						Stall	IF	ID	EX	MEM	WB			
sw $t6, 8($a0)								IF	ID	EX	MEM	WB		
								Stall	IF	ID	EX	MEM	WB	

Khi dùng cơ chế forwading để giải quyết hazard ta thấy rằng ta cần chèn thêm 2 stall như trên.

```
addi $t1, $zero, 100
addi $t2, $zero, 100
add  $t3, $t1, $t2
lw   $t4, 0($a0)
lw   $t5, 4($a0)
NOP (Stall 1)
and  $t6, $t4, $t5
NOP (Stall 2)
sw   $t6, 8($a0)
```

(d) Dùng cơ chế forward, stall để giải quyết hazard (control và data), khi đó có bao nhiêu stall?

- Data hazard giống ở câu c.
- Vì ở đây không có lệnh branch nào cả cho nên control hazard không làm thay đổi số stall cần giải quyết là 2 stall.

(e) Ngoài 2 cơ chế ở trên, ta có thể giảm stall bằng cách sắp xếp lại thứ tự code (giải quyết bằng trình biên dịch compiler). Hãy sắp xếp lại code sao cho ít stall nhất.

Có thể rút gọn về 1 stall là số stall ít nhất của đoạn code trên:

```
lw   $t4, 0($a0)
lw   $t5, 4($a0)
addi $t1, $zero,100
addi $t2, $zero,100
and  $t6, $t4, $t5
NOP (Stall)
add  $t3, $t1, $t2
sw   $t6, 8($a0)
```