ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO LAB 7

LỚP: L06

NHÓM: 8

Họ và tên	Mã số sinh viên
Lê Võ Đăng Khoa	2211606
Võ Xuân Hạ	2210916
Nguyễn Thị Hiền Hạnh	2210920
Thái Trí Thịnh	2213308

Thành phố Hồ Chí Minh, tháng 12, năm 2023

Mục lục

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 thời gian một clock của hệ thống single clock, multi clock và pipeline clock.

(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).

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

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

(a) Thời gian một clock:

- Single clock: Clock_time = Instruction memory + Register + ALU + Data memory = 500 ns
- Multi clock: Clock_time = Instruction memory hoặc Data memory = 150 ns
- Pipeline clock: Clock_time = Instruction memory hoặc Data memory = 150 ns

(b) Chương trình có:

IC_ALU = 102 lệnh

IC_Branch = 51 lệnh

IC_Jump = 50 lệnh

IC = IC_ALU + IC_Branch + IC_Jump = 102 + 51 + 50 = 203

- Single cycle: CPU_time = Clock_time*IC = 500*203 = 101500ns
- Multi cycle:

Lệnh	Số chu kì
ALU	4
Branch	3
Jump	2

- CPU_time = Clock_time*(4*IC_{ALU} + 3*IC_{Branch} + 2*IC_{Jump}) = 150*(4*102 + 3*51 + 2*50) = 99150 ns

- Pipeline cycle: CPU_time = (Số chu kì thực hiện lệnh đầu tiên + IC - 1)*Clock_time = (5 + 203 - 1)*150 = 31050 ns

(c)

- Speed up của Pipeline so với Single cycle: T_{single}/ $T_{pipeline}$ =101500 / 31050 = 3.269
- Speed up của Pipeline so với Multi cyclce: T_{multi}/ $T_{pipeline}$ =99150 / 31050 = 3.193

(d) Thời gian một clock:

- Single clock: Clock_time = Instruction memory + Register + ALU + Data memory = 550 ns
- Multi clock: Clock_time = Instruction memory hoặc Data memory = 150 ns
- Pipeline clock: Clock_time = Instruction memory hoặc Data memory = 150 ns

Tính thời gian thực thi:

- Single cycle: CPU_time = Clock_time*IC = 550*203 = 111650 ns
- Multi cycle:

Lệnh	Số chu kì
ALU	4
Branch	3
Jump	2

- CPU_time = Clock_time*(4*IC_ALU + 3*IC_Branch + 2*IC_Jump) = 150*(4*102 + 3*51 + 2*50) = 99150 ns
- Pipeline cycle: CPU_time = (Số chu kì thực hiện lệnh đầu tiên + IC - 1)*Clock_time = (5 + 203 - 1)*150 = 31050 ns

Tính speedup:

- Speed up của Pipeline so với Single cycle: T_single/ T_pipeline =111650 / 31050 = 3.596
- Speed up của Pipeline so với Multi cyclce: T_multi/ T_pipeline =99150 / 31050 = 3.193

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.

(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) ?

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

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

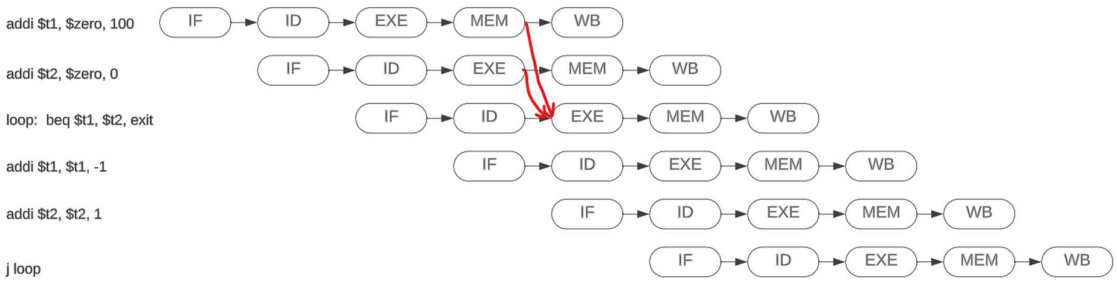
(e) Ngoài 2 cơ chế ở trên, ta có thể giảm stall bằng cách sắp xếp lại thứ tự code (giải quyết bằng trình biên dịch compiler). Hãy sắp xếp lại code sao cho ít stall nhất.

```
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) Lệnh 3 yêu cầu dữ liệu được ghi vào $t1 và $t2 của lệnh 1, 2 và lệnh 4, 5

(b) Chèn 2 stall giữa (2) và (3) và 1 stall sau lệnh 6 (do sau J là beq), vì vòng lặp chạy 50 lần nên cần tổng cộng 50 + 2 = 52 stalls

(c) Không cần chèn stall



(d) - Không có so sánh sớm:

3 stall giữa (3) và (4) ; 1 stall sau (6)

Tổng: 4*50 + 3 = 203 stalls

- Có so sánh sớm:

1 stall giữa (2) và (3); 1 stall giữa (3) và (4); 1 stall sau (6)

Tổng : 1 + 2*50 + 1 = 102 stalls

(e) Không thể sắp xếp để tối ưu hơn được nữa

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.

(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) ?

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

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

(a) Lệnh 3 yêu cầu dữ liệu được ghi vào $t1 và $t2 từ lệnh 1, 2

Lệnh 6 yêu cầu dữ liệu được ghi vào $t4 và $t5 từ lệnh 4, 5

Lệnh 7 yêu cầu dữ liệu được ghi vào $t6 từ lệnh 6

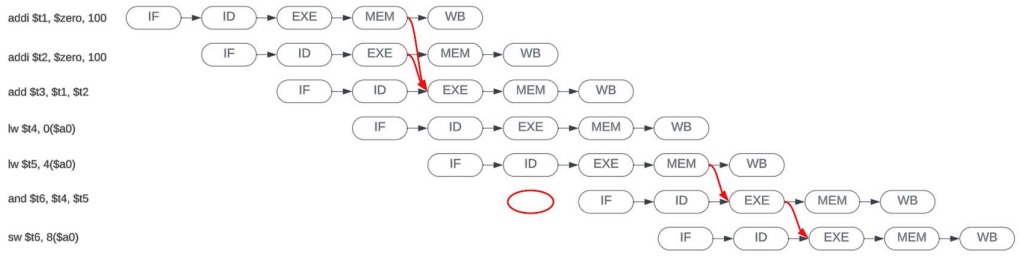
(b)

Giữa (2) và (3) : 2 stalls

Giữa (5) và (6) : 2 stalls

Giữa (6) và (7) : 2 stalls

(c) Có 1 stall giữa (5) và (6)



(d) Vì không có lệnh branch nên không đề cập đến

(e) Sắp xếp:

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

- TH1: Chỉ cần thêm 1 stall ở giữa (5) và (6) nếu không có forwarding
- TH2: Không cần chèn stall nếu có forwarding