Trịnh Duy Hưng MSSV: 1913652

Bài 1:

a)

Single cycle = thời gian thực thi lệnh dài nhất (load)

= 150 + 100 + 100 + 150

= 500 (ns)

Multi cycle = max (I-Mem, Regs, ALU, D-Mem, Regs)

= 150 (ns)

Pipeline cycle = max (I-Mem, Regs, ALU, D-Mem, Regs)

= 150 (ns)

b)

$CPI_{single} = 1$

$Time_{single}$ = 203 * 500 = 101 500 (ns) (Tổng cộng có 203 lệnh)

$Time_{multi}$ = 661 * 150 (661 chu kỳ)

= 99 150 (ns)

$CPI_{pipeline} = 1$

$Time_{pipeline}$ = (5 + 203 - 1) * 150

= 31 050 (ns)

c)

Speed up $_{pipeline / single}$ = 101500 / 31050 = 3,27

Speed up $_{pipeline / multi}$ = 99150 / 31050 = 3,19

d)
Single cycle = 550 (ns)

Multi cycle = max (I-Mem, Regs, ALU, D-Mem, Regs)
= 150 (ns)

Pipeline cycle = max (I-Mem, Regs, ALU, D-Mem, Regs)
= 150 (ns)

$CPI_{single} = 1$
$Time_{single} = 203 * 550 = 111\ 650$ (ns) (Tổng cộng có 203 lệnh)

$Time_{multi} = 661 * 150$
$= 99\ 150$(ns)

$CPI_{pipeline} = 1$
$Time_{pipeline} = (5 + 203 - 1) * 150$
$= 31\ 050$ (ns)

$Speed\ up_{pipeline / single} = 111650 / 31050 = 3,60$

$Speed\ up_{pipeline / multi} = 99150 / 31050 = 3,19$

Bài 3:

a)

```
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)
```

Lệnh 3 phụ thuộc lệnh 2 và 1.
Lệnh 6 phụ thuộc lệnh 5 và 4.
Lệnh 7 phụ thuộc lệnh 6.

b)
1 12345
2 12345
3 1__2345
4 12345
5 12345
6 1__2345
7 1__2345

Bộ xử lý Pipeline chia quá trình thực thi lệnh thành 5 bước, mỗi bước thực thi trong trong một chu kỳ.

1. **IF**: Lấy lệnh (khối Instruction Memory), 32bits lệnh chứa các thông tin của 1 lệnh được lấy ra từ instruction memory.
2. **ID**: Giải mã lệnh (khối Registers và Control), xác định toán tử, các tín hiệu điều khiển, nội dung các thanh ghi, giá trị immediate.
3. **EXE**: Thực thi tác vụ lệnh (khối ALU).
4. **MEM**: Truy xuất vùng nhớ (khối Data Memory) - chỉ dùng cho lệnh `load/store`.
5. **WB**: Ghi kết quả vào thanh ghi (khối Registers).

2 stall giữa lệnh 2 và lệnh 3.
2 stall giữa lệnh 5 và lệnh 6.
2 stall giữa lệnh 6 và lệnh 7.
=> cần 6 stall

c)
Nếu giải quyết data hazard bằng phương án forwarding thi ta sẽ chỉ còn 1 stall ở giữa lệnh 5 và 6.

```
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)
```

d)
Giống như câu c.

e)
Sắp xếp code lại như thứ tự sau:
Chỉ còn lại 1 stall giữa 6 và 3.

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

(4) → (5) → (1) → (2) → (6) → (3) → (7).

Bài 2:

a)

```
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
```

Dòng lệnh 4 phụ thuộc lệnh 1 và 2.
Dòng lệnh 6 phụ thuộc dòng lệnh 4.

b)

1 12345
2 12345
3 1__2345
4 12345
5 1_2345 (loop 50 lần)
6 12345

Vậy có tất cả 2 + 1 * 50 = 52 stall.

c)

Forwarding sẽ làm mất đi 2 stall đầu tiên nên sẽ còn lại 50 stall.

d)

Giống như câu c.

e)

Code trên đã tối ưu nhất nên ta giữ nguyên, không thay đổi.