Trịnh Duy Hưng MSSV: 191 3652

**Bài 1:**

1. Thanh ghi PC (Program Counter) dùng để chứa địa chỉ của lệnh chuẩn bị được thực thi. Sau khi bộ nhớ lệnh thực hiện việc truy xuất lệnh tại địa chỉ được cung cấp bởi thanh ghi PC thì lệnh được đưa ra ngoài, đồng thời PC tăng lên 4 để chứa địa chỉ lệnh chuẩn bị doc975 thực thi tiếp theo.
2. Instruction memory (Bộ nhớ lệnh) chứa toàn bộ các lệnh của (các) chương trình đang được thực thi. Input là địa chỉ và output là lệnh.
3. Registers là tập hợp 32 thanh ghi đa dụng của MIPS.
4. Input của ALU là giá trị các toán hạng tham gia vào phép toán đều có kích thước 32 bit. Output là kết quả của phép toán trên hi toán hạng, bên cạnh đó còn ngõ ra Zero, nếu kết quả là 0 thì giá trị sẽ là 1, ngược lại là 0.

1. Bộ Control nhận input là trường opcode. Output là tín hiệu điều khiển được dùng để điều khiển các khối chức năng.
2. Data memory (Bộ nhớ dữ liệu) là nơi lưu trữ dự liệu của các chương trình đang thực thi. Khác với bộ nhớ lệnh, bộ nhớ dữ liệu ngoài truy xuất đọc bộ nhớ còn được truy xuất để đọc dữ liệu (ghi bộ nhớ). Input là địa chỉ (32 bit) và dữ liệu vào (trường hợp thực hiện lệnh lưu trữ). Output là dữ liệu ra (32 bit).
3. Bộ chọn MUX có chức năng tùy chọn giữa hai tín hiệu đầu vào.

vd: trong tập lệnh R, bộ mux dùng để lựa chọn giữa việc ghi nội dung từ thanh ghi nào, tùy thuộc vào 0 hay 1.

1. Sign-extend đùng để mở rộng dấu đối với các lệnh cần mở rộng dấu khi kích thước dữ liệu truyền vào nhỏ hơn kích thước thanh ghi.

**Bài 2:**

1. RegDst xác định thanh ghi đích (rt hoặc rd).
2. RegWrite cho biết có viết lên thanh ghi hay không.
3. MemRead xác định bộ nhớ thực hiện việc đọc.
4. MemWrite xác định bộ nhớ thực hiện việc ghi.
5. MemtoReg xác định giá trị được ghi đến từ đâu (ALU hoặc bộ nhớ).
6. Branch kết hợp với Boolean kiểm tra đề3 cho phép tải địa chỉ nhánh mục tiêu vào PC.
7. Jump cho phép tải địa chỉ nhảy vào PC.
8. ALUSrc chọn toán hạng nguồn thứ hai cho ALU (rt hoặc sign-extend).

**Bài 3:**

1 lw $s0, 8($a0) # load $s0 from memory at address $t2 + 8

2 sw $s0, 8($a0) # store $s0 to memory at address $a0 + 8

3 add $s0, $s1, $s2 # add s0 = s1 + s2

4 beq $t2, $t1, label # branch on equal, if $t2 == $t1 branch to label

5 j label # jump to

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RegDst | RegWrite | MemRead | MemWrite | MemtoReg | Branch | Jump | ALUSrc |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | x | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | x | 0 | 0 | 0 | x | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x |

**Bài 4:**

1. Load instruction:

PC -> Instruction Memory -> Registers -> MUX -> ALU ->Data Memory -> MUX

Cycle = 200 + 150 + 10 + 100 + 200 + 10 = 670 ns.

1. Store instruction:

PC -> Instruction Memory -> Registers -> Data Memory -> MUX

Cycle = 200 + 150 + 200 + 10 = 560 ns.

1. Branch:

PC -> Instruction Memory -> Registers -> ALU -> MUX -> MUX

Cycle = 200 + 150 + 100 + 10 + 10 = 470 ns.

1. Jump:

PC -> Instruction Memory -> MUX

Cycle = 200 + 10 = 210 ns.

Thời gian của hệ thống trên là 670 ns đề đảm bảo một lệnh bất kỳ được hoàn thành.