chap.4.1.SingleClockCycle

2411545-2413163-2412795

November 19 2025

Bài 1:

1. Thanh ghi PC dùng để làm gì?
⇒ Chức năng: PC (Program Counter) là thanh ghi đếm chương trình. Nó dùng để lưu trữ địa chỉ của câu lệnh hiện tại đang được thực thi. Sau mỗi chu kỳ, nó thường được cập nhật để trỏ đến lệnh tiếp theo (PC + 4) hoặc địa chỉ mới nếu có lệnh rẽ nhánh/nhảy

2. Instruction memory chứa gì? Input, Output là gì?
⇒ Chứa: Chứa mã máy (machine code) của các câu lệnh trong chương trình cần thực thi.
Input: Read address (Địa chỉ đọc) - Nhận giá trị từ thanh ghi PC.
Output: Instruction [31-0] - Nội dung của câu lệnh 32-bit tại địa chỉ đó.

3. Registers là tập hợp bao nhiêu thanh ghi? Input, Output là gì? Số lượng: Trong kiến trúc MIPS chuẩn, đây là tập hợp 32 thanh ghi đa năng (32-bit).
Input:
-Read register 1 (5 bit): Địa chỉ thanh ghi nguồn 1 (Instruction [25-21]).
-Read register 2 (5 bit): Địa chỉ thanh ghi nguồn 2 (Instruction [20-16]).
-Write register (5 bit): Địa chỉ thanh ghi đích cần ghi (Instruction [20-16] hoặc [15-11] tùy vào loại lệnh).
-Write data (32 bit): Dữ liệu cần ghi vào thanh ghi.
-RegWrite: Tín hiệu điều khiển cho phép ghi.

Output:
-Read data 1 (32 bit): Giá trị đọc ra từ thanh ghi 1.
-Read data 2 (32 bit): Giá trị đọc ra từ thanh ghi 2.

4. Input và Output của ALU là gì?

Input:

-Toán hạng 1: Read data 1 từ bộ Registers.

-Toán hạng 2: Lấy từ Read data 2 (Registers) hoặc giá trị Sign-extend (Immediate) tùy thuộc vào bộ chọn MUX.

-ALU control (4 bit): Tín hiệu điều khiển phép toán cần thực hiện.

Output:

-ALU result: Kết quả tính toán (ví dụ: tổng, hiệu, kết quả AND/OR).

-Zero: Cờ báo hiệu (bằng 1 nếu kết quả ALU là 0, dùng cho lệnh rẽ nhánh beq).

5. Bộ Control nhận Input là trường nào? Output dùng để làm gì?

Input: Nhận 6 bit Opcode của lệnh, tức là trường Instruction [31-26].

Output: Các tín hiệu điều khiển (màu xanh trong hình) như: RegDst, Branch, MemRead, MemtoReg, ALUOp, MemWrite, ALUSrc, RegWrite, Jump. Các tín hiệu này dùng để điều khiển hoạt động của các MUX, ALU, và việc đọc/ghi bộ nhớ/thanh ghi .

6. Data memory chứa gì? Input, Output là gì?

Chứa: Dữ liệu của chương trình (biến, mảng, cấu trúc dữ liệu...) khi chương trình chạy.

Input:

-Address: Địa chỉ ô nhớ cần truy cập (thường đến từ kết quả tính toán của ALU).

-Write data: Dữ liệu cần ghi vào bộ nhớ (lấy từ Read data 2 của Registers).

-Các tín hiệu điều khiển MemRead (đọc) và MemWrite (ghi).

Output:

-Read data - Dữ liệu đọc được từ bộ nhớ tại địa chỉ được cung cấp.

7. Bộ chọn (MUX) có chức năng gì? Ví dụ.

Chức năng: Chọn 1 trong 2 (hoặc nhiều) luồng dữ liệu đầu vào để đưa ra đầu ra duy nhất, dựa trên giá trị của tín hiệu điều khiển (selector).

Ví dụ: MUX nằm trước đầu vào thứ 2 của ALU (được điều khiển bởi tín hiệu ALUSrc).

Nếu ALUSrc = 0: MUX chọn dữ liệu từ thanh ghi (Read data 2).

Nếu ALUSrc = 1: MUX chọn dữ liệu tức thời đã mở rộng dấu (Sign-extend).

8. Sign-extend dùng để làm gì? Ví dụ.

Chức năng: Bộ mở rộng dấu, dùng để chuyển đổi một số nguyên 16-bit (Instruction [15-0]) thành số nguyên 32-bit mà vẫn giữ nguyên giá trị và dấu của số đó.

Ví dụ: Trong lệnh addi $s1$,s2, -10 hoặc lệnh load/store (lw, sw), phần hằng số (immediate) chỉ có 16 bit. Để cộng được với thanh ghi 32-bit hoặc tính địa chỉ, bộ Sign-extend sẽ biến 16 bit này thành 32 bit trước khi đưa vào ALU .

Bài 2:

Tín hiệu	**Chức năng**
$RegDst$	Điều khiển MUX chọn thanh ghi đích ($WriteRegister$). $0 \rightarrow$ Chọn bit [20-16] (rt). $1 \rightarrow$ Chọn bit [15-11] (rd).
$RegWrite$	Cho phép ghi vào Register File. Nếu bằng 1, dữ liệu sẽ được ghi vào thanh ghi được chỉ định.
$MemRead$	Cho phép đọc dữ liệu từ Data Memory.
$MemWrite$	Cho phép ghi dữ liệu vào Data Memory.
$MemtoReg$	Điều khiển MUX chọn dữ liệu ghi về thanh ghi (Write Back). $0 \rightarrow$ Lấy từ kết quả ALU. $1 \rightarrow$ Lấy từ bộ nhớ (Data Memory).
$Branch$	Kết hợp với cờ $Zero$ của ALU để quyết định có thực hiện rẽ nhánh hay không (cập nhật PC mới).
$Jump$	Điều khiển MUX để nạp địa chỉ nhảy không điều kiện vào PC.
$ALUSrc$	Điều khiển MUX chọn toán hạng thứ 2 cho ALU. $0 \rightarrow$ Lấy từ thanh ghi ($ReadData2$). $1 \rightarrow$ Lấy từ giá trị tức thời ($SignExtend$).

Bài 3:

Bảng dưới đây trình bày trạng thái các tín hiệu điều khiển cho các lệnh: `lw`, `sw`, `add`, `beq`, và `j`.
Ghi chú:

- **X**: (Don't Care) Giá trị tín hiệu không ảnh hưởng đến hoạt động của lệnh.
- **ALUOp**:
  - 00: Phép cộng (dùng tính địa chỉ cho lw/sw).
  - 01: Phép trừ (dùng so sánh cho beq).

Lệnh	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite
`lw`	0	1	1	1	1	0
`sw`	X	1	X	0	0	1
`add`	1	0	0	1	0	0
`beq`	X	0	X	0	0	0
`j`	X	X	X	0	0	0

  - 10: Phụ thuộc vào trường *funct* (dùng cho lệnh R-type như add).

Bài 4:

Dựa trên bảng thông số độ trễ (Delay) đã cho:

- Instruction Memory (IM): 200ns
- Data Memory (DM): 200ns
- Registers (Regs): 150ns
- ALU: 100ns
- Mux / Add / Shift / Sign-extend: 10ns

(a) Thời gian hoàn thành của các kiểu lệnh

1. **Lệnh Load (lw):** Là lệnh có đường đi dài nhất (Critical Path).

$$\begin{aligned} T_{lw} &= T_{IM} + \max(T_{Regs}, T_{SignExt} + T_{Mux}) + T_{ALU} + T_{DM} + T_{Mux} \\ &= 200 + 150 + 100 + 200 + 10 \\ &= \mathbf{660} \text{ ns} \end{aligned}$$

   Lưu ý: Tại đầu vào ALU, đường từ Register (150ns) chậm hơn đường SignExtend (20ns) nên ta lấy 150ns.

2. **Lệnh Store (sw):**

$$\begin{aligned} T_{sw} &= T_{IM} + T_{Regs} + T_{ALU} + T_{DM} \\ &= 200 + 150 + 100 + 200 \\ &= \mathbf{650} \text{ ns} \end{aligned}$$

3. **Lệnh R-type (add, sub...):**

$$\begin{aligned} T_{R-type} &= T_{IM} + (T_{Regs} + T_{Mux\_ALUSrc}) + T_{ALU} + T_{Mux\_WB} \\ &= 200 + (150 + 10) + 100 + 10 \\ &= \mathbf{470}\ \text{ns} \end{aligned}$$

Lưu ý: Với R-type, dữ liệu đi qua Mux ALUSrc (chọn Rt), nên ta cộng thêm delay của Mux vào đường Register.

4. **Lệnh Branch (beq):**

$$\begin{aligned} T_{beq} &= T_{IM} + (T_{Regs} + T_{Mux}) + T_{ALU} + T_{Mux\_PC} \\ &= 200 + 160 + 100 + 10 \\ &= \mathbf{470}\ \text{ns} \end{aligned}$$

5. **Lệnh Jump (j):**

$$\begin{aligned} T_{jump} &= T_{IM} + T_{Shift} + T_{Mux} \\ &= 200 + 10 + 10 \\ &= \mathbf{220}\ \text{ns} \end{aligned}$$

(b) Thời gian Cycle của hệ thống

Trong kiến trúc Single Clock Cycle, chu kỳ máy phải đủ lớn để thực hiện lệnh tốn nhiều thời gian nhất.

- Lệnh dài nhất: **Load (lw)** với 660ns.
- **Vậy thời gian chu kỳ hệ thống (Cycle Time) = 660ns.**