

BÀI TẬP CHƯƠNG 4

4.1. Instruction: And Rd, Rs, Rt.

4.1.1. Lập bảng các giá trị cuối cùng tương ứng như sau:

RegWrite	MemRead	ALUSrc	MemWrite	ALUOp	RegRsrc	Branch
1	0	0(Reg)	0	10	1(ALU)	0

* Giải thích từng lệnh:

a) RegWrite = 1, vì lệnh này có ghi vào thanh ghi:

b) MemRead = 0, do bộ nhớ không cần được để lấy dữ liệu.

c) ALUSrc = 0, số 0 ở đây quyết định là lấy đầu vào từ thanh ghi.

d) MemWrite = 0, do không cho phép ghi vào thanh ghi vì đầu ra của ALU không phải là địa chỉ để lưu, được.

e) ALUOp = 10, do là kiểu phép toán R-Type.

f) RegRsrc = 1, vì 1 là lấy dữ liệu từ ALU, do kết quả của phép AND là do ALU đưa ra nên lấy dữ liệu cuối cùng từ ALU.

g) Branch = 0: Do không phải là lệnh có rẽ nhánh.



4.1.2. Khối Registers và khối ALU là hữu ích nhất cho lệnh này.

4.1.3. Bỏ đầu ra mà không được sử dụng: Branch Add mà dù đó rình lệnh nhảy nhưng lại không dùng đến.

Không có đầu ra đó là Data Memory, vì đầu ra của ALU là chuẩn sau khi AND giữa các bit để thành ghi mà không phải là address.

4.4

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps

4.4.1 * Trong trường hợp này, chỉ có một thao tác duy nhất là truy xuất lệnh từ bộ nhớ. Thời gian thực hiện chính là thời gian của I-Mem.

* Để lưu: Thời gian chu kỳ (T) = 200ps

4.4.2 Lệnh nhảy không điều kiện cần phải có lệnh:

1) I-Mem	200
2) Sign-Extend	15
3) Shift-Left-2	10
4) Add	70
5) Mux:	20
	<hr/> 315 ps

4.4.3 Lệnh nhảy có điều kiện.



P - Mem.

200

Các lệnh cần sử dụng Shift left 2

Regs

90

jump

Mux

20

Beq, Bne,

ALU

90

SU, SAL

Mx

20

420 ps

4.4.4 Thiết kế này được sử dụng cho lệnh xử nhân liên quan đến PC như beq, bne

4.7

1010 1100 0110 0010' 0000 0000 0001 0100

sw. 82, 20(83)

4.7.1

Khối Sign - Extend: Đưa vào bit dấu của lệnh, khối này sẽ mở rộng dấu để tạo ra một số nguyên có độ dài như phải là 32 bits

Shift left 2: Dịch trái 2 bit của mỗi số nguyên dương với nhân số đó với 4.

4.7.3

Đi chỉ PC mới sau khi thực hiện lệnh xử lý theo lệnh đó là gì. Trong trường hợp lệnh sw. Đi chỉ PC là PC + 4

4.7.4

Cho mỗi rux giá trị của nó sẽ là 1, bởi vì lệnh jump sẽ nhảy đến địa chỉ mới khác với $PC + 4$ và hoặc là $PC + 4 + offset \times 4$

Mệnh lệnh rux khác đều là 0 do cấu hình sẵn của khối control

4.7.5

ALU: Thực hiện phép toán số học và logic.

Add 4: Điều khiển lấy giá trị của address cộng thêm 4 để có thể đưa đến lệnh tiếp theo.

Add cân lại: Thực hiện phép cộng giữa $offset \times 4$ với $PC + 4$ dùng cho các lệnh nhảy. (bex, bne)

4.7.6 Registers: Có các đầu vào là Read Register 1, Read Register 2, Write register, Write data

Read Registers: Dùng để đọc các giá trị của thanh ghi.

Write register: Đưa vào một thanh ghi chứa giá trị đích.

Write data: Đưa giá trị để đưa vào thanh ghi.