ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỆN TỬ

o0o

***

BÁO CÁO

BÀI TẬP LỚN CẤU TRÚC MÁY TÍNH

LAB3:
Register File, Memory Modeling, and I/O System

GVHD:Thầy Trần Hoàng Linh

SVTT:Dương Nhật Huy – 1810162

1. Thiết kế Register và 2 Memory Models

a. Phân tích

Yêu cầu đặt ra cho thiết kế

- Data Memory với 4KB . Ghi dữ liệu ra file datamem.data
- Instruction Memory với 16KB . Đọc dữ liệu từ file instmem.data
- Registers với kích thước 32x32 bit , registor 0 luôn có giá trị 0 . Ghi dữ liệu ra file regfile.data

b) Thiết kế

i) Instruction Memory :

Input : PC 32 bit là tín hiệu địa chỉ của mỗi thanh ghi 1 byte trong bộ nhớ . Do mỗi lệnh của Risc – V bao gồm 4 byte nên cần PC + 4 để nhảy đến lệnh kế tiếp

Output: inst 32 bit là lệnh rút ra tự bộ nhớ Instruction Memory

ii) Datamemory :

Input :

- clk_i : xung clock ngõ vào
- rst_i : tín hiệu reset dữ liệu. Khi rst_i = 1 thì tất cả dữ liệu trả về giá trị 0
- addr : tín hiệu địa chỉ xác định các thanh ghi 1byte trong bộ nhớ . Do cần xây dựng bộ nhớ 4KB nên addr cần 10 bit
- wdata : 32 bit dữ liệu cần ghi vào Datamemory
- wren : 1 là ghi dữ liệu , 0 là đọc dữ liệu.

Output:

- rdata : dữ liệu đọc được từ bộ nhớ

iii) Register

Input :

- clk_i : xung clock ngõ vào
- rst_i : tín hiệu reset dữ liệu. Khi rst_i = 1 thì tất cả dữ liệu trả về giá trị 0
- rs1_addr, rs2_addr, rd_addr: địa chỉ của các thanh ghi RS1, RS2, and RD.
- rd_data: dữ liệu cần ghi vào RD.
- rd_wren: 1 nếu cần ghi dữ liệu vào Rd.

Output:

- rs1_data, rs2_data: đọc dữ liệu của thanh ghi rs1 và rs2 trong bộ nhớ

c) Kiểm tra thiết kế , kết quả và kết luận

i) Instruction Memory :

- Kiểm tra :

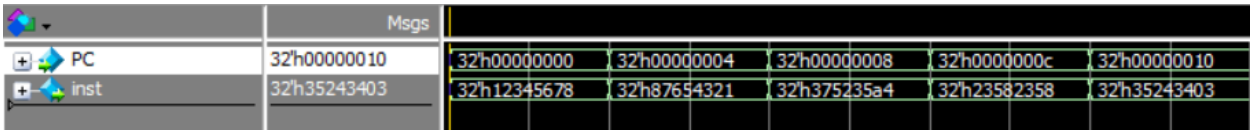
Nhập dữ liệu vào file instmem.data. Sau đó thay đổi giá trị PC ở ngõ vào . Kiểm tra xem ngõ ra có đọc chính xác

instmem.data - N

File Edit Format

```
12345678
87654321
375235a4
23582358
35243403
```

- Kết quả mô phỏng



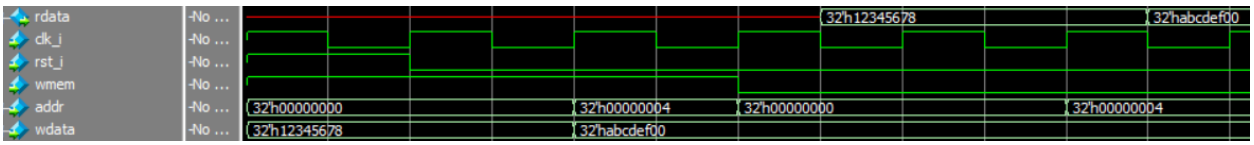
- Kết luận

Thiết kế hoạt động chính xác với yêu cầu đặt ra

ii) Data memory

- Kiểm tra

Nhập giữ liệu vào Data memory. Kiểm tra xem file datamem.data có lưu trữ dữ liệu chính xác không



- Kết quả

datamem.data

File Edit Format

```
// memory dat
// instance=/
// format=hex
78
56
34
12
00
ef
cd
ab
00
```

- Kết luận

Thiết kế hoạt động chính xác với yêu cầu

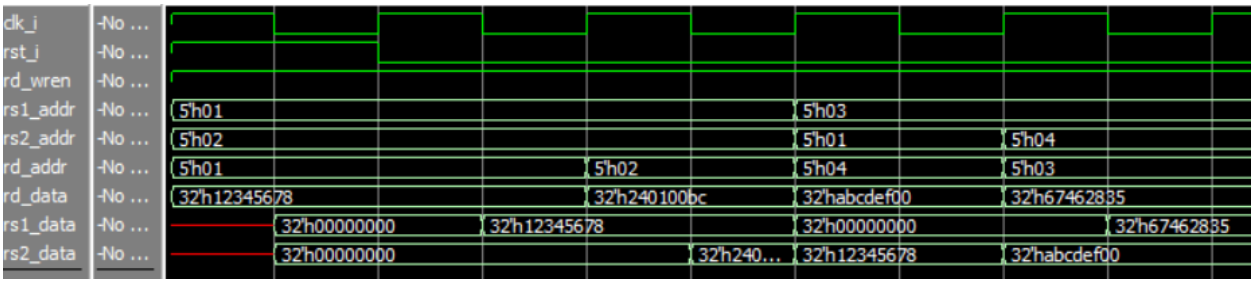
iii) Register file

- Kiểm tra

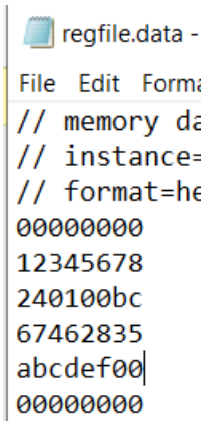
Nhập giữ liệu vào Registor File. Kiểm tra :

+ Rs1_data , Rs2_data nhận dữ liệu chính xác không

+ File regfile.data có lưu trữ dữ liệu chính xác không



- Kết quả



- Kết luận

Thiết kế hoạt chính xác với yêu cầu

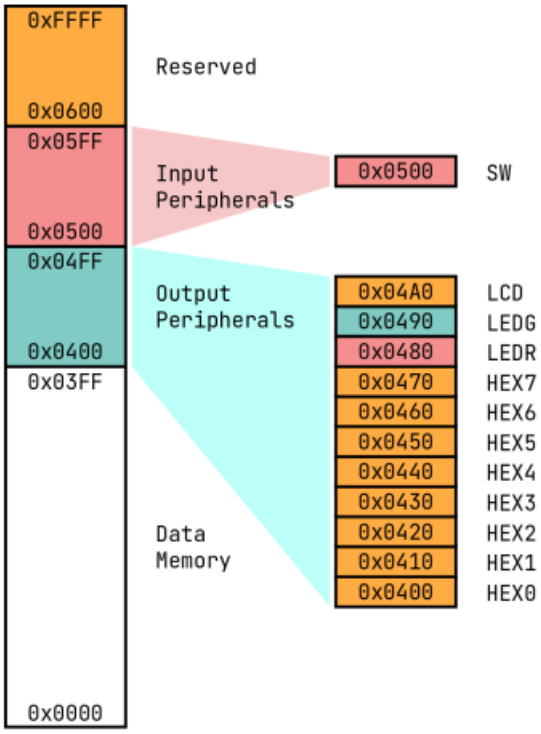
2 . Thiết kế load store – unit

a. Phân tích

Trong thực tế, một bộ xử lý giao tiếp với các thiết bị ngoại vi để xuất dữ liệu hoặc đọc dữ liệu. Điều này có thể là hoàn thành bằng cách thiết kế Hệ thống I / O. Một số thiết bị ngoại vi tiêu chuẩn là đèn LED, màn hình LCD hoặc công tắc, v.v.

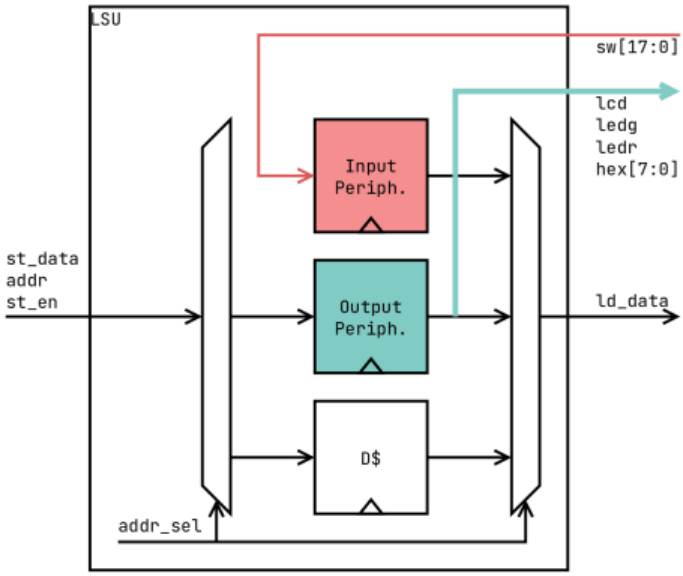
trên thực tế, thiết bị ngoại vi được xem như là “bộ nhớ”. Ví dụ: khi thanh ghi 32 bit được gán cho 32 đèn LED, việc ghi

Cần Thiết kế Load – Store Unit thỏa mãn



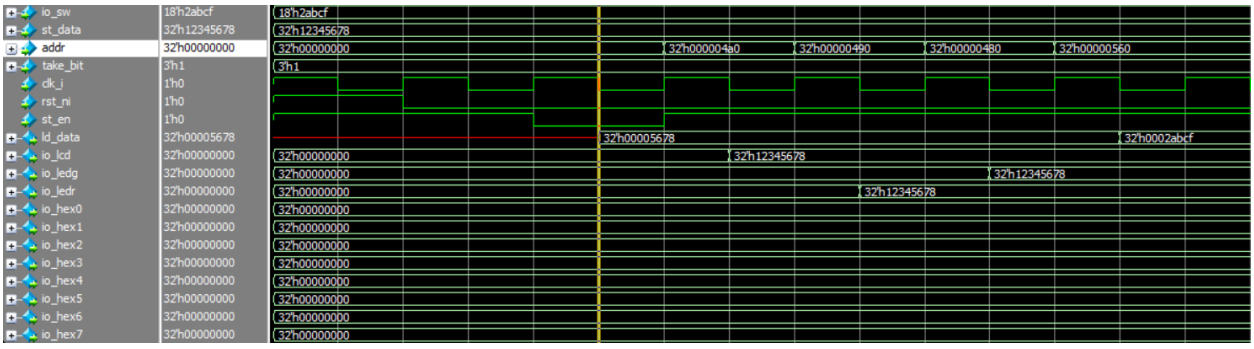
b) Thiết kế

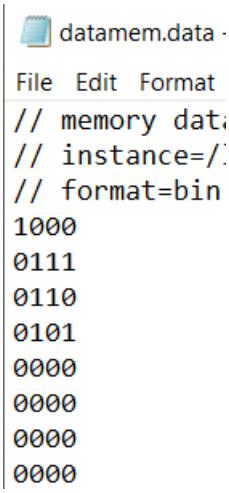
i. Sơ đồ khối :



ii. Mô tả chi tiết

- Inputs:
 + clk_i, rst_ni: xung clock và tín hiệu reset dữ liệu.
 + take_bit: quyết định thực hiện lệnh nào trong các lệnh lw, lh, lb, lbu, lbhu, sw, sb, sh
 + addr: 32 bit địa chỉ đọc hoặc ghi dữ liệu
 + st_data: 32 bit dữ liệu cần lưu giữ trong LSU
 + st_en: 1 nếu ghi dữ liệu, 0 đọc dữ liệu.
 + io_sw: 18-bit data từ 18 switches trên Kit DE2
 - Output:
 + ld_data: 32 bit load data.
 + io_lcd: 32 bit data to drive LCD.
 + io_ledg: 32-bit data to drive green LEDs.
 + io_ledr: 32-bit data to drive red LEDs.
 + io_hex0 .. io_hex7: 8 32-bit data to drive 8 7-segment LEDs.
 - Giải thuật :
+ addr :
 Nếu addr = 5XXH và st_en = 1 thì ld_data = io_sw
 Nếu addr = 4XXH và st_en = 1 thì thực hiện các thao tác ghi dữ liệu ra LCD và LED
 Nếu addr < 400H thì thực hiện các thao tác với Datamem. Nếu st_en = 0 thì ta dùng các lệnh load data. Nếu st_en = 0 thì ta dùng các lệnh store data.
+ take_bit tín hiệu xác định các lệnh load data và store data
 000: lb,sb; 001:lh,sh; 010:lw,sw; 100 lbu; 101 lhu
c) Kiểm tra thiết kế
Thay đổi các giá trị ngõ vào clk_i, rst_ni, take_bit, addr, st_data,st_en, io_sw. Kiểm tra giá trị ngõ ra.
Kiểm tra file datamem
d) Kết quả mô phỏng





d) Kết luận

Thiết kế hoạt động đúng với bài test đã đặt ra

3. Câu hỏi

i.What are the diferences between a packed array and an unpacked array?

Packed array được đóng gói đề cập đến các kích thước được khai báo sau kiểu và trước tên mã định danh dữ liệu. Unpacked array đề cập đến các kích thước được khai báo sau tên mã định danh dữ liệu.

ii. How many address bits are required for a 16KB memory?

16kb = 16 x 1024 (bit) = 2^14 = > cần 14 địa chỉ bit

How many address bits are required for a 4KB memory?

4kb = 4 x 1024 (bit) = 2^12= > cần 12 địa chỉ bit

What do those keywords mean? $writememh, $writememb, $readmemh, $readmemb. Which one should

$writememh, $writemem Ghi dữ liệu từ chương trình sang một tệp nào đó định dạng mã hex hoặc nhị phân

$writememh, $writemem Đọc dữ liệu từ chương trình sang một tệp nào đó định dạng mã hex hoặc nhị phân