Bài 1:

>Thanh ghi PC chứa địa chỉ của lệnh trong chương trình đang được thực thi.

>Instruction memory dùng để nhận địa chỉ lệnh được cung cấp bởi PC.

Input là địa chỉ lệnh của PC – Output là nội dung của lệnh

>Register là tập hợp của 3 thanh ghi.

Input là địa chỉ thanh ghi – Output là nội dung thanh ghi

>Input của ALU nội dung thanh ghi và hằng số 32 bit

Output là giá trị sau khi thực hiện phép toán

>Input của Control 6 bit opcode của 32 bit lệnh

Output là tín hiệu điều khiển cho Datapath

>Data memory chứa dữ liệu của máy tính.

Input là giá trị là address từ ALU, địa chỉ của thanh ghi Rt

Output là giá trị tại address trong data memory

>Bộ Mux dùng để chọn dữ liệu. Ví dụ RegDst phải chọn giữa thanh ghi Rt và Rd

>Sign-extend dùng để mở rộng dấu. Ví lệnh lw $s0, 8($a0) thì số 8 chỉ là hằng số 16 bit ta phải mở rộng ra 32 bit.

Bài 2:

>RegDst: chọn lựa thanh ghi ghi từ Rt hoặc Rd.

>RegWrite: cho phép ghi kết quả ALU result vào thanh ghi.

>MemRead: yêu cầu đọc (lw) (để đọc từ data memory thì MemRead = 1)

>MemWrite: yêu cầu ghi (sw)(= 1 khi cần ghi vào data memory)

> MemtoReg: chọn dữ liệu của BusW là kết quả từ ALU result hoặc từ Data Memory Read data

>Branch: kết hợp với cờ Zero tại ALU dùng để phép việc rẽ nhanh khi.

>Jump: dùng để cho phép việc rẽ nhánh.

>ALUSrc: chọn toán hạn nội dung của thanh ghi Rt hoặc hằng số 32 bit.

Bài 3:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RegDst | Jump | Branch | MemRead | MemToReg | ALUOp | MemWrite | ALUSrc | RegWrite |
| lw $s0, 8($a0) | 0 | 0 | 0 | 1 | 0 | ADD | 0 | 1 | 1 |
| sw $s0, 8($a0) | 0 | 0 | 0 | 0 | 1 | ADD | 1 | 1 | 1 |
| add $s0, $s1, $s2 | 1 | 0 | 0 | 0 | 0 | ADD | 0 | 0 | 1 |
| beq $t2, $t1, label | 0 | 0 | 1 | 0 | 0 | SUB | 0 | 0 | 0 |
| j label | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bài 4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Load | Store | ALU | Branch | Jump |  |
| critical path |  |  |  |  |  |  |
| hoàn thành |  |  |  |  |  |  |

Load = Instruction memory + Registers + ALU + Data memory + MUX

Store = Instruction memory + Registers + ALU + Data memory

ALU = Instruction memory + Regiters + MUX + ALU + MUX

Branch = is + R + Mux + ALU + Mux + mux

J = IS + Shift left + MUX