Lab 6

Câu 1:

* Thanh ghi PC (Program Counter): Dùng để lưu trữ địa chỉ của lệnh tiếp theo sẽ được thực thi trong bộ nhớ lệnh.
* Instruction memory: Chứa các lệnh máy để thực thi. Input là địa chỉ của lệnh cần đọc, output là lệnh tại địa chỉ đó.
* Registers: Là tập hợp của 32 thanh ghi trong bộ vi xử lý MIPS. Input là địa chỉ của thanh ghi và giá trị cần ghi, output là giá trị tại địa chỉ thanh ghi.
* ALU (Arithmetic Logic Unit): Input là hai toán hạng và một toán tử, output là kết quả của phép toán.
* Bộ Control: Nhận input là các trường của lệnh máy, output là các tín hiệu điều khiển cho các bộ phận khác của CPU.
* Data memory: Chứa dữ liệu. Input là địa chỉ và giá trị cần ghi (nếu là lệnh ghi), output là giá trị tại địa chỉ (nếu là lệnh đọc).
* Bộ chọn (MUX): Dùng để chọn một trong nhiều tín hiệu vào. Ví dụ, nếu có hai tín hiệu A và B, MUX sẽ chọn A hoặc B dựa trên tín hiệu điều khiển.
* Sign-extend: Dùng để mở rộng số bit của một số, giữ nguyên dấu của số. Ví dụ, số -1 biểu diễn bằng 8 bit là 11111111, khi sign-extend lên 16 bit sẽ thành 1111111111111111.

Câu 2:

* **RegDst**: Điều khiển thanh ghi đích trong bộ đăng ký. Nếu RegDst=1, lệnh sẽ ghi vào thanh ghi Rd; nếu RegDst=0, lệnh sẽ ghi vào thanh ghi Rt.
* **RegWrite**: Nếu RegWrite=1, kết quả của ALU hoặc dữ liệu đọc từ bộ nhớ sẽ được ghi vào thanh ghi đích.
* **MemRead**: Nếu MemRead=1, CPU sẽ thực hiện đọc dữ liệu từ bộ nhớ.
* **MemWrite**: Nếu MemWrite=1, CPU sẽ thực hiện ghi dữ liệu vào bộ nhớ.
* **MemtoReg**: Điều khiển nguồn dữ liệu ghi vào thanh ghi. Nếu MemtoReg=1, dữ liệu đọc từ bộ nhớ sẽ được ghi vào thanh ghi; nếu MemtoReg=0, kết quả của ALU sẽ được ghi vào thanh ghi.
* **Branch**: Nếu Branch=1 và kết quả của ALU là 0, PC sẽ nhảy đến địa chỉ mới.
* **Jump**: Nếu Jump=1, PC sẽ nhảy đến địa chỉ mới không phụ thuộc vào kết quả của ALU.
* **ALUSrc**: Điều khiển toán hạng thứ hai của ALU. Nếu ALUSrc=1, toán hạng thứ hai là số hằng; nếu ALUSrc=0, toán hạng thứ hai là giá trị từ thanh ghi.

Câu 3:

1. lw $s0, 8($a0):
   * RegDst = 0
   * RegWrite = 1
   * MemRead = 1
   * MemWrite = 0
   * MemtoReg = 1
   * Branch = 0
   * Jump = 0
   * ALUSrc = 1
2. sw $s0, 8($a0):
   * RegDst = X (không quan trọng)
   * RegWrite = 0
   * MemRead = 0
   * MemWrite = 1
   * MemtoReg = X (không quan trọng)
   * Branch = 0
   * Jump = 0
   * ALUSrc = 1
3. add $s0, $s1, $s2:
   * RegDst = 1
   * RegWrite = 1
   * MemRead = 0
   * MemWrite = 0
   * MemtoReg = 0
   * Branch = 0
   * Jump = 0
   * ALUSrc = 0
4. beq $t2, $t1, label:
   * RegDst = X (không quan trọng)
   * RegWrite = 0
   * MemRead = 0
   * MemWrite = 0
   * MemtoReg = X (không quan trọng)
   * Branch = 1
   * Jump = 0
   * ALUSrc = 0
5. j label:
   * RegDst = X (không quan trọng)
   * RegWrite = 0
   * MemRead = 0
   * MemWrite = 0
   * MemtoReg = X (không quan trọng)
   * Branch = 0
   * Jump = 1
   * ALUSrc = X (không quan trọng)

Câu 4:

(a) Dưới đây là đường đi critical path (đường đi có độ trễ lâu nhất) và thời gian hoàn thành của các kiểu lệnh:

* **Load**: Instruction memory (200ns) -> Registers (150ns) -> ALU (100ns) -> Data memory (200ns) -> Mux (10ns) -> Registers (150ns). Thời gian hoàn thành: 810ns.
* **Store**: Instruction memory (200ns) -> Registers (150ns) -> ALU (100ns) -> Data memory (200ns). Thời gian hoàn thành: 650ns.
* **ALU**: Instruction memory (200ns) -> Registers (150ns) -> ALU (100ns) -> Mux (10ns) -> Registers (150ns). Thời gian hoàn thành: 610ns.
* **Branch**: Instruction memory (200ns) -> Registers (150ns) -> ALU (100ns) -> Add (10ns). Thời gian hoàn thành: 460ns.
* **Jump**: Instruction memory (200ns) -> Shift left (10ns) -> Add (10ns). Thời gian hoàn thành: 220ns.

(b) Thời gian chu kỳ của hệ thống sẽ bằng thời gian hoàn thành lâu nhất của tất cả các lệnh, tức là 810ns (tương ứng với lệnh Load). Điều này đảm bảo rằng trong một chu kỳ, bất kỳ lệnh nào cũng sẽ được thực thi xong.