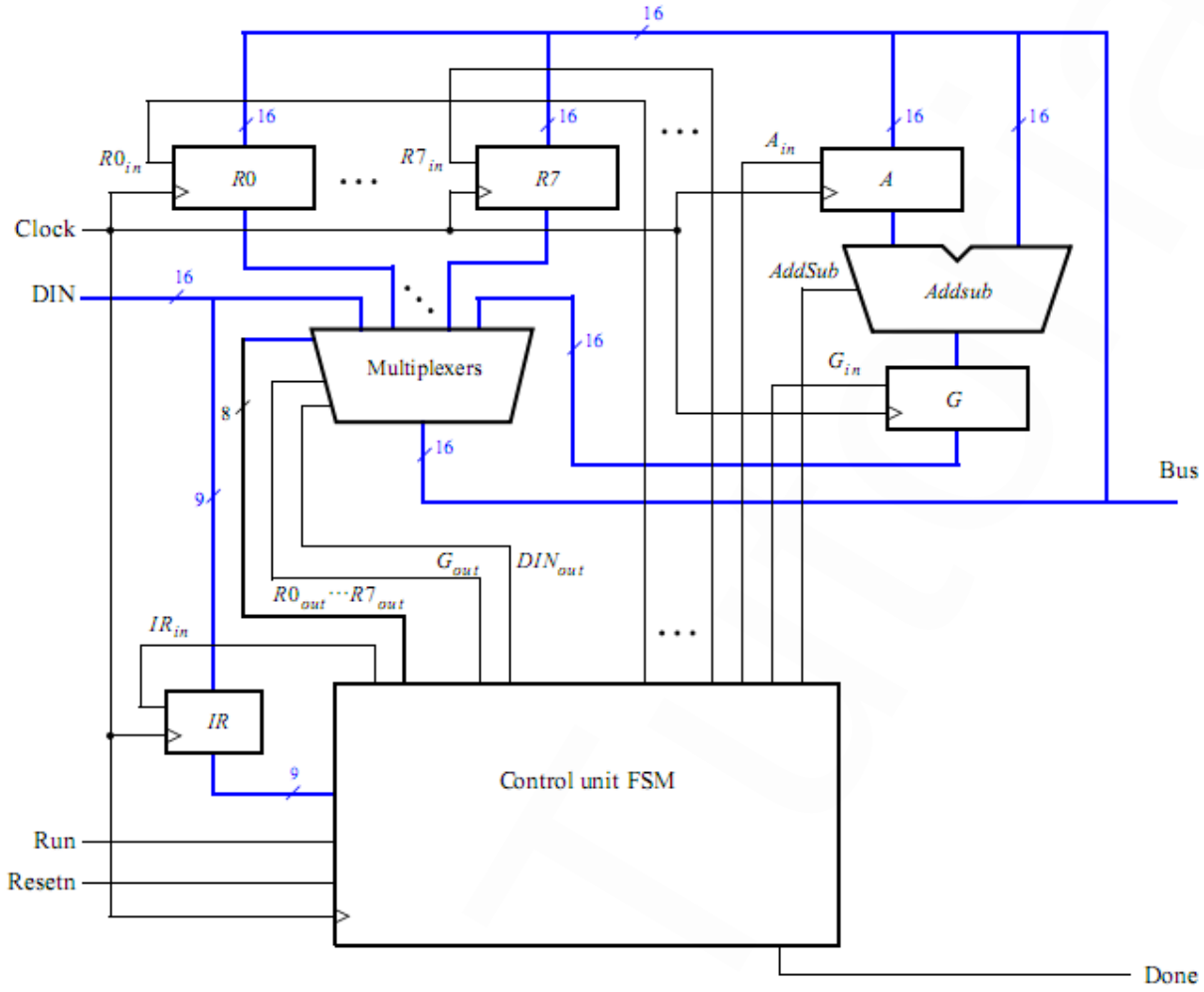
Thiết Kế CPU Đơn Giản



Một CPU bao gồm 3 thành phần cơ bản nhất:

1. Khối tính toán – ALU
2. Khối nhớ – các register
3. Khối xử lý lệnh – FSM

Ngoài ra, CPU còn rất nhiều thành phần khác, trong đó thành phần xử lý interupt là rất quan trọng nhưng vì tính chất phức tạp của nó nên không đưa vào đây.

CPU là để thực thi các lệnh từ phần mềm. Khi xử lý lệnh phần mềm thì CPU thực thi từng lệnh một, vì vậy để tiếm kiệm area ta thấy chỉ có một đường Bus (đường xanh) chạy khắp CPU làm việc chuyển data.

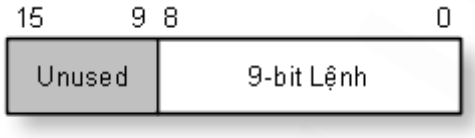
Đây là CPU 16-bit Bus, và 9-bit lệnh với 3-bit mã lệnh. Ta thấy từ DIN 16-bit vào có 1 nhánh 9-bit rẽ vào register IR, đó là lệnh từ ngoài đưa vào cho CPU. Cấu trúc của 9-bit như sau:

Ta có tất cả 8 registers (R0 đến R7) nên ta cần 3-bit để chỉ ra register nào cần dùng. Các bit [2:0] chỉ ra register nguồn, các bit [5:3] chỉ ra register đích, còn lại các bit [8:6] là mã lệnh. Với 3-bit mã lệnh thì ta có tổng cộng 8 lệnh, tuy nhiên để đơn giản ở đây ta chỉ làm việc với 4 lệnh cơ bản nhất:

Mã lệnh	Lệnh	Chức năng
000	**mv** *Rx,Ry*	Rx ← [Ry]
001	**mvi** *Rx,#D*	Rx ← DIN
010	**add** *Rx,Ry*	Rx ← [Rx] + [Ry]
011	**sub** *Rx,Ry*	Rx ← [Rx] – [Ry]

Phân biệt: Rx là chỉ đến register nào trong 8 registers, [Rx] là chỉ đến giá trị chứa trong register đó

Để thống nhất ta chọn 9-bit lệnh là 9-bit thấp từ 16-bit Bus:



Ta có 8 registers từ R0 đến R7 để lưu giá trị trong quá trình làm việc.

Register A và G giúp cho quá trình tính toán của ALU.

Register IR để lưu lệnh cho FSM hoạt động.

FSM là khối chính điều khiển hoạt động của CPU, FSM điều khiển các tín hiệu:

- R0in đến R7in: tín hiệu enable cho register tương ứng lấy giá trị từ Bus vào.
- R0out đến R7out: tín hiệu điều khiển khối mux để nối ngõ ra của register tương ứng đến Bus.
- Ain: tín hiệu enable cho register A lấy giá trị từ Bus vào.
- Gin: tín hiệu enable cho register G lấy kết quả tính toán từ ALU vào.
- Gout: tín hiệu điều khiển khối mux để nối ngõ ra của register G đến Bus.
- DINout: tín hiệu điều khiển khối mux đẻ nối DIN vào Bus.
- IRin: tín hiệu enable cho register IR lấy lệnh từ DIN vào.
- AddSub: tín hiệu điều khiển khối ALU thực hiện phép cộng hoặc phép trừ.
- Done: tín hiệu báo cho bên ngoài biết CPU đã thực thi xong một lệnh.

FSM chịu sự điều khiển từ 2 tín hiệu:

- Run: khi tín hiệu này ở mức cao thì FSM mới hoạt động. Nếu ở mức thấp thì FSM tạm dừng.
- Resetn: khi tín hiệu này ở mức thấp thì FSM reset.

Hướng dẫn:

Đầu tiên ta phân tích spec ở trên thành 4 thành phần:

1. Các thanh ghi R0 đến R7, thanh ghi lệnh IR.
2. Bộ multiplexer.
3. Khối tính toán gồm ALU và 2 thanh ghi phụ trợ là A và G.
4. Khối điều khiển FSM.

Trong đó FSM là khó nhất vì vậy để sau cùng. Ta viết code để miêu tả 3 thành phần ở trên trước, sau đó nối chúng lại với nhau cho thành một cái khung hoàn chỉnh chỉ thiếu mỗi FSM mà thôi. Mô phỏng cách mà FSM hành xử để kiểm tra khung đã hoạt động tốt hay chưa.

Viết FSM độc lập với khung ở trên. Khung đã được test rồi vì vậy cứ gắn FSM vào khung và test FSM.

Cách mà FSM hành xử:

	T1	**T2**	**T3**
mv **000**	RYout, RXin, Done		
mvi **001**	DINout, RXin, Done		
add **010**	RXout, Ain	RYout, Gin	Gout, RXin, Done
sub **011**	RXout, Ain	RYout, Gin, AddSub	Gout, RXin, Done

T1, T2 và T3 là tượng trưng cho số chu kỳ clock

Ta thấy với lệnh **mv** và **mvi** thì CPU chỉ mất 1 clock, còn lệnh tính toán mất 3 clock để hoàn thành.

Ví dụ với lệnh **add** thì FSM sẽ hành xử như sau:

1. Sau khi nhận lệnh **add**, FSM sẽ bật RXout và Ain ngay trong chu kỳ đầu (T1).
2. Hạ RXout và Ain, bật RYout và Gin trong chu kỳ thứ hai (T2).
3. Hạ RYout và Gin, bật Gout và RXin và Done trong chu kỳ thứ ba (T3).
4. Hạ Gout, RXin và Done, bật IRin để lấy lệnh tiếp theo. Đây gọi là chu kỳ nghỉ.

Khi viết FSM thì nên vẽ mô hình ra, đánh số thứ tự hết cho các trạng thái, rồi mới dựa vào mô hình đấy để viết. Như vậy thì mình sẽ không nhầm lẫn và dễ quản lý, hơn nữa nếu có sai sót thì sửa trên mô hình vẫn dễ hơn là sửa trực tiếp trong code.