

# 27 March

9 to go!

Instruction Execution

# Five Execution Steps

Instruction Fetch

Instruction Decode and Register Fetch

Execution, Memory Address Computation, or Branch Completion

Memory Access or R-type instruction completion

Memory Read Completion

**INSTRUCTIONS TAKE FROM 3 - 5 CYCLES!**

A FSM looks at the *op-code* to determine how many...

# Step 1: Instruction Fetch

Use PC to get instruction and put it in the Instruction Register.

Increment the PC by 4 and put the result back in the PC.

Can be described succinctly using RTL "Register-Transfer Language"

```
IR = Memory[PC]; IR is "Instruction Register"  
PC = PC + 4;
```

*What is the advantage of updating the PC now?*

## Step 2: Instruction Decode and Register Fetch

Read registers *rs* and *rt* in case we need them

Compute the branch address in case the instruction is a branch

RTL:

```
A = Reg[IR[25-21]];
B = Reg[IR[20-16]];
ALUOut = PC + (sign-extend(IR[15-0])
<< 2);
```

We aren't setting any control lines based on the instruction type (we are busy "decoding" it in our control logic)

# Step 3 (instruction dependent)

ALU is performing one of three functions, based on instruction type

Memory Reference:

```
ALUOut = A + sign-extend(IR[15-0]);
```

R-type:

```
ALUOut = A op B;
```

Branch:

```
if (A==B) PC = ALUOut;
```

# Step 4 (R-type or memory-access)

Loads and stores access memory

$\text{MDR} = \text{Memory}[\text{ALUOut}];$  *MDR is Memory Data Register*  
or  
 $\text{Memory}[\text{ALUOut}] = \text{B};$

R-type instructions finish

$\text{Reg}[\text{IR}[15-11]] = \text{ALUOut};$

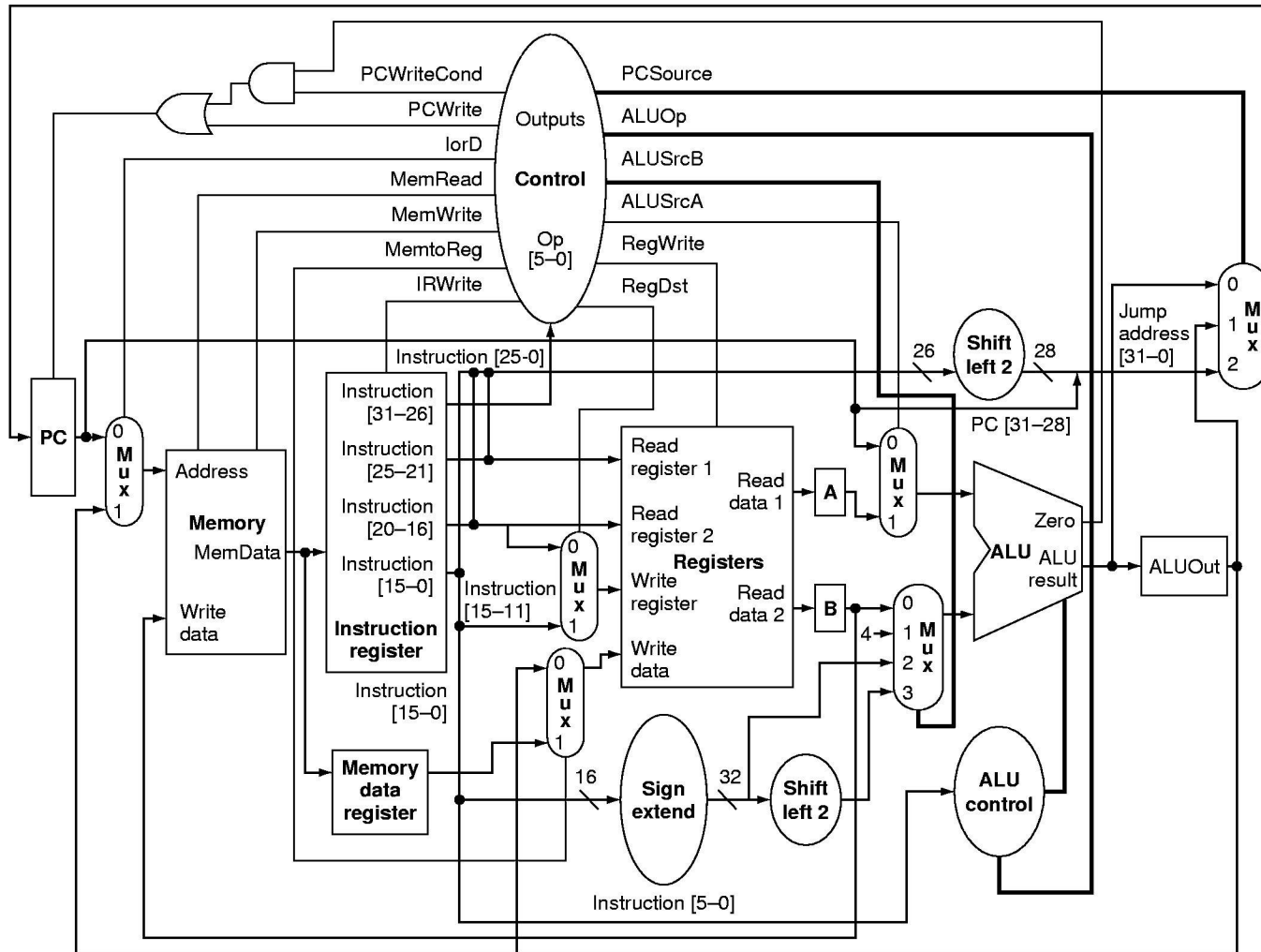
# Step 5 Memory Read Completion

```
Reg[IR[20-16]] = MDR;
```

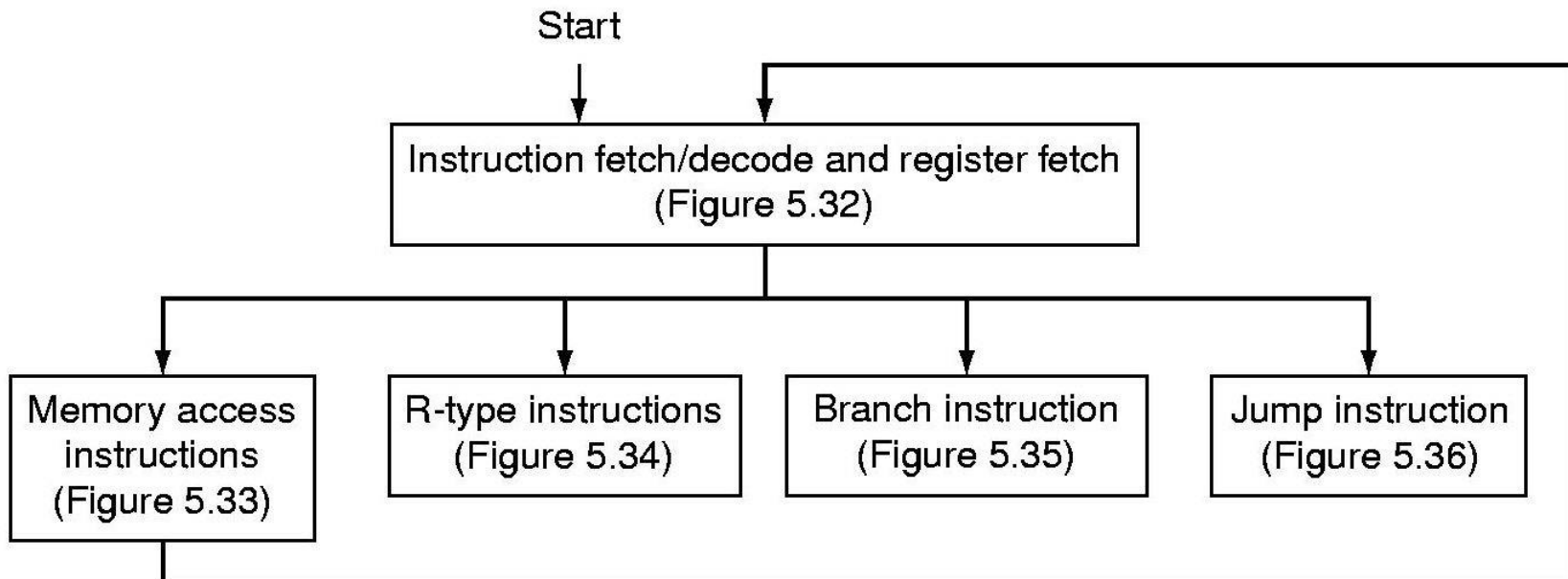
# Summary:

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR = Memory[PC] PC = PC + 4			
Instruction decode/register fetch	A = Reg [IR[25-21]] B = Reg [IR[20-16]] ALUOut = PC + (sign-extend (IR[15-0]) << 2)			
Execution, address computation, branch/jump completion	ALUOut = A op B	ALUOut = A + sign-extend (IR[15-0])	if (A ==B) then PC = ALUOut	PC = PC [31-28]    (IR[25-0]<<2)
Memory access or R-type completion	Reg [IR[15-11]] = ALUOut	Load: MDR = Memory[ALUOut] or Store: Memory [ALUOut] = B		
Memory read completion		Load: Reg[IR[20-16]] = MDR		

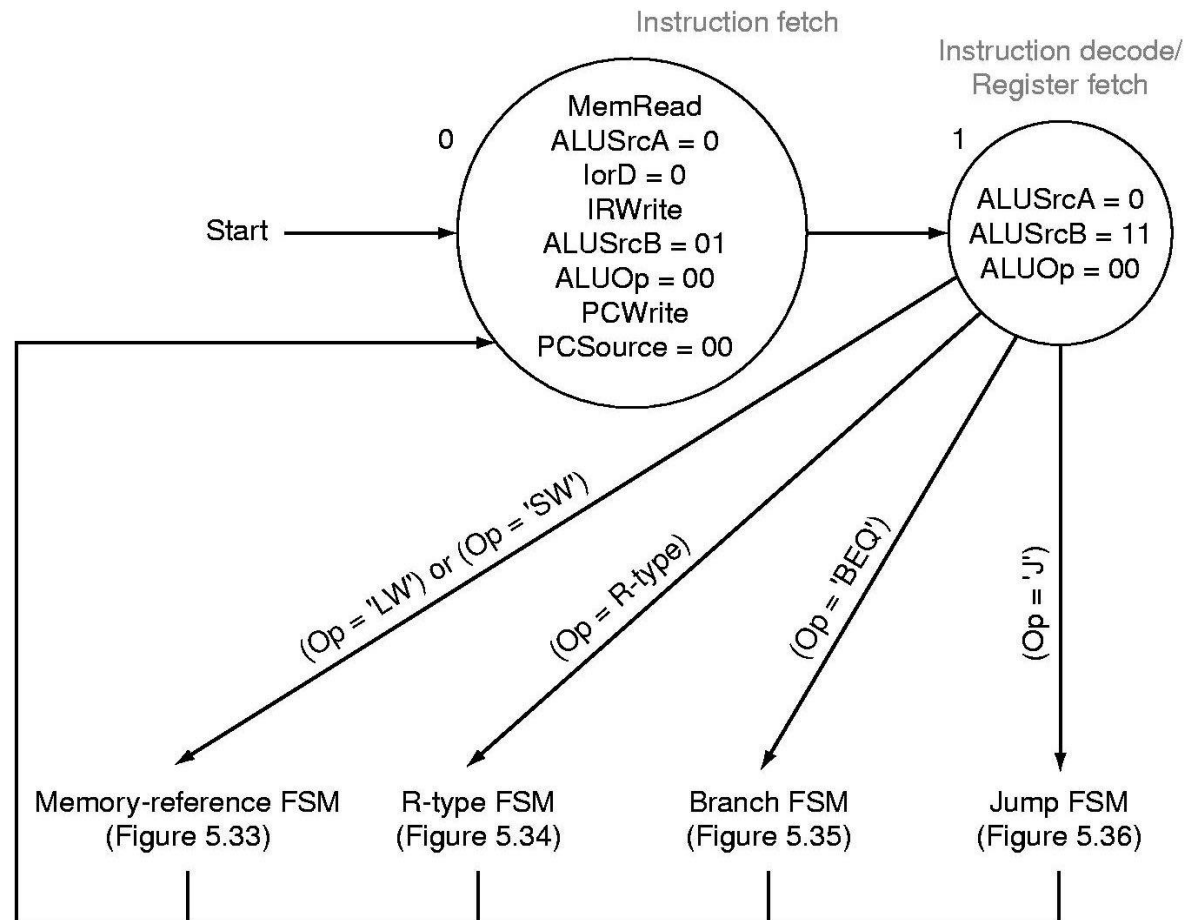
# Data Path with Control (5.28)



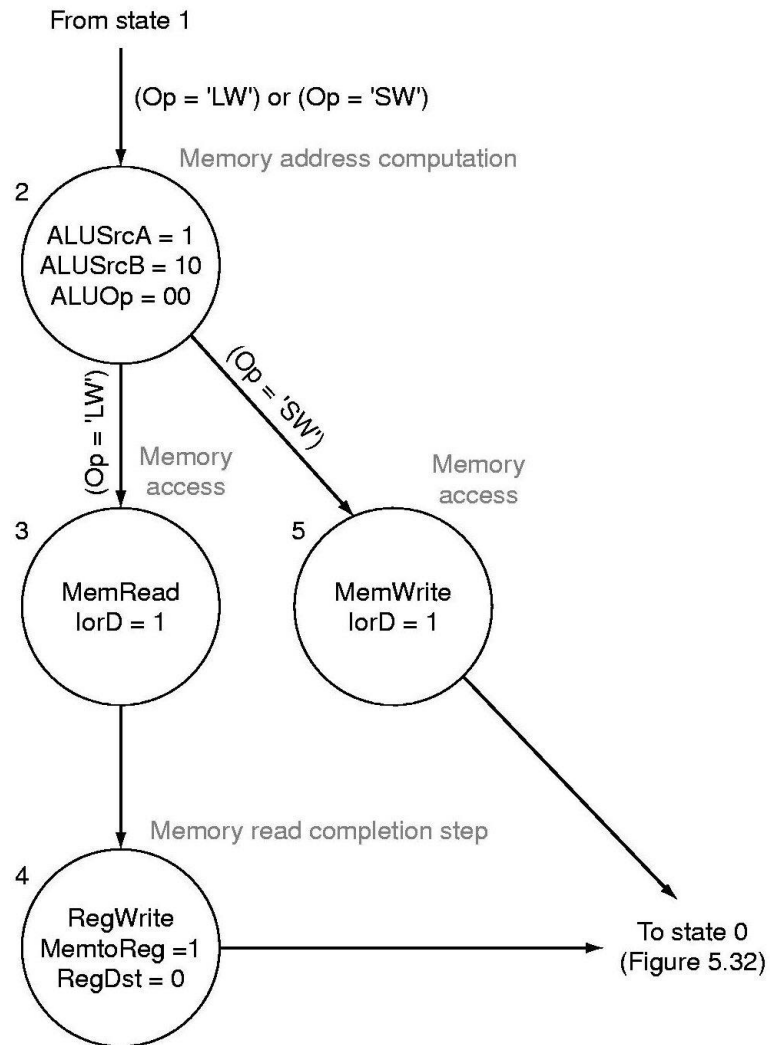
# FSM Control (5.31)



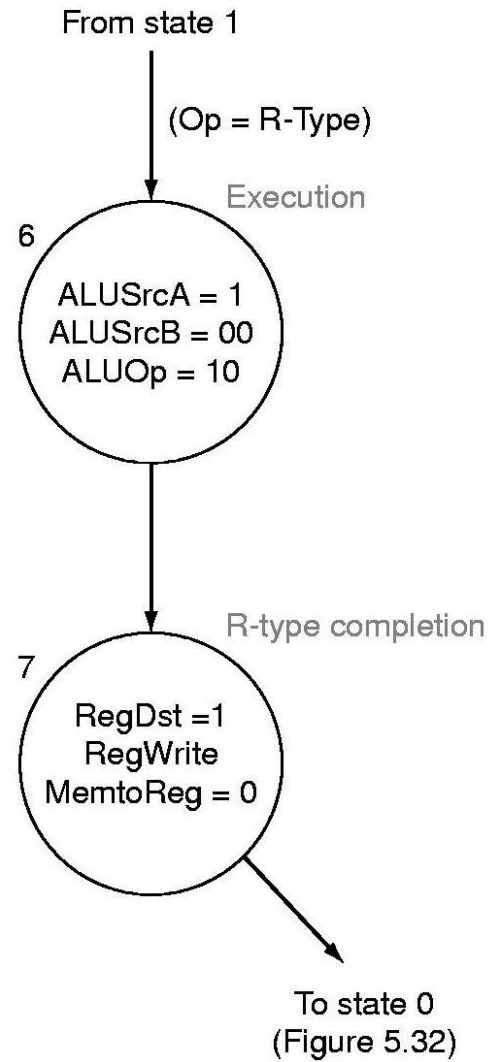
# IFetch and Decode (5.32)



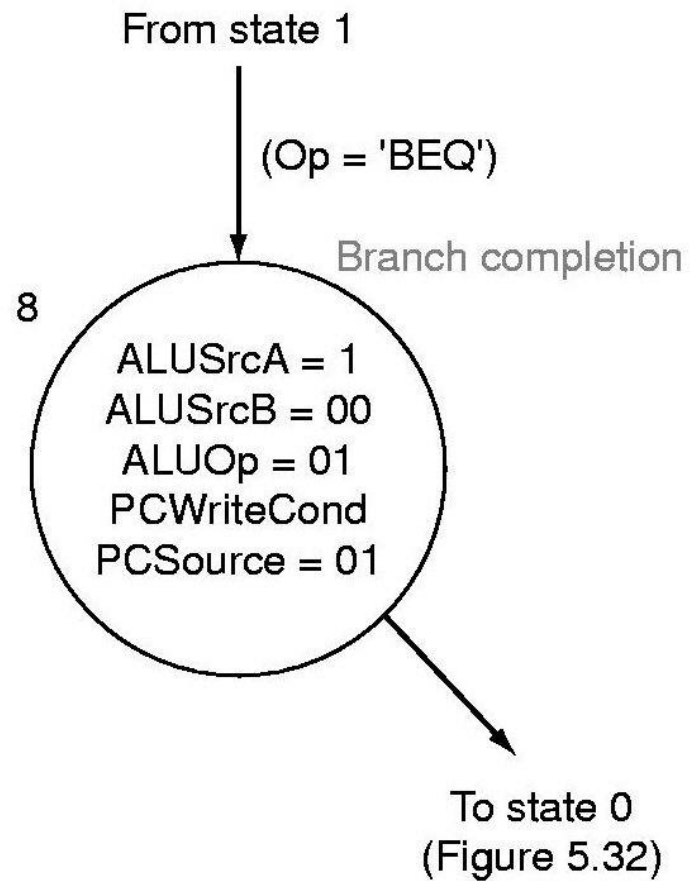
# Memory Ref (5.33)



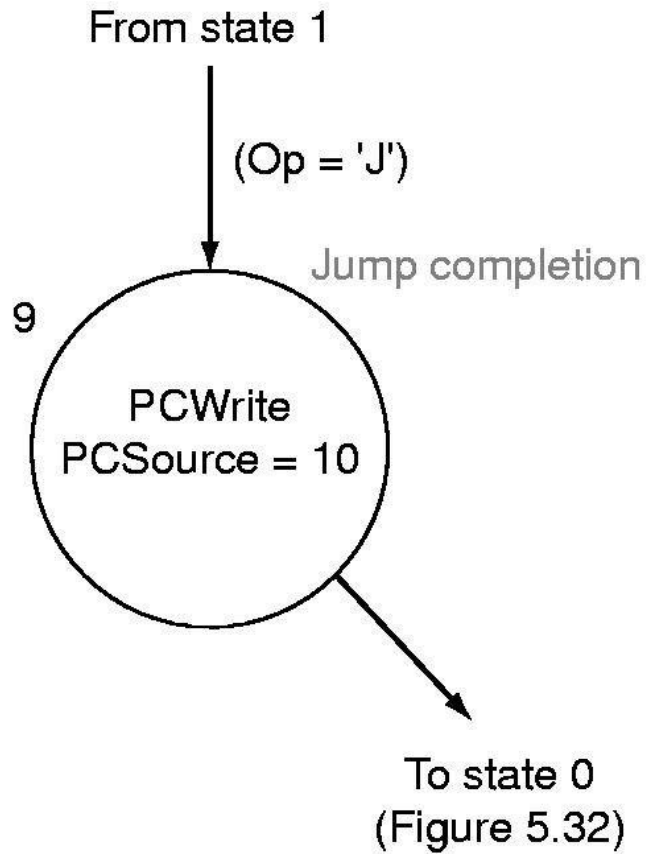
# R-type (5.34)



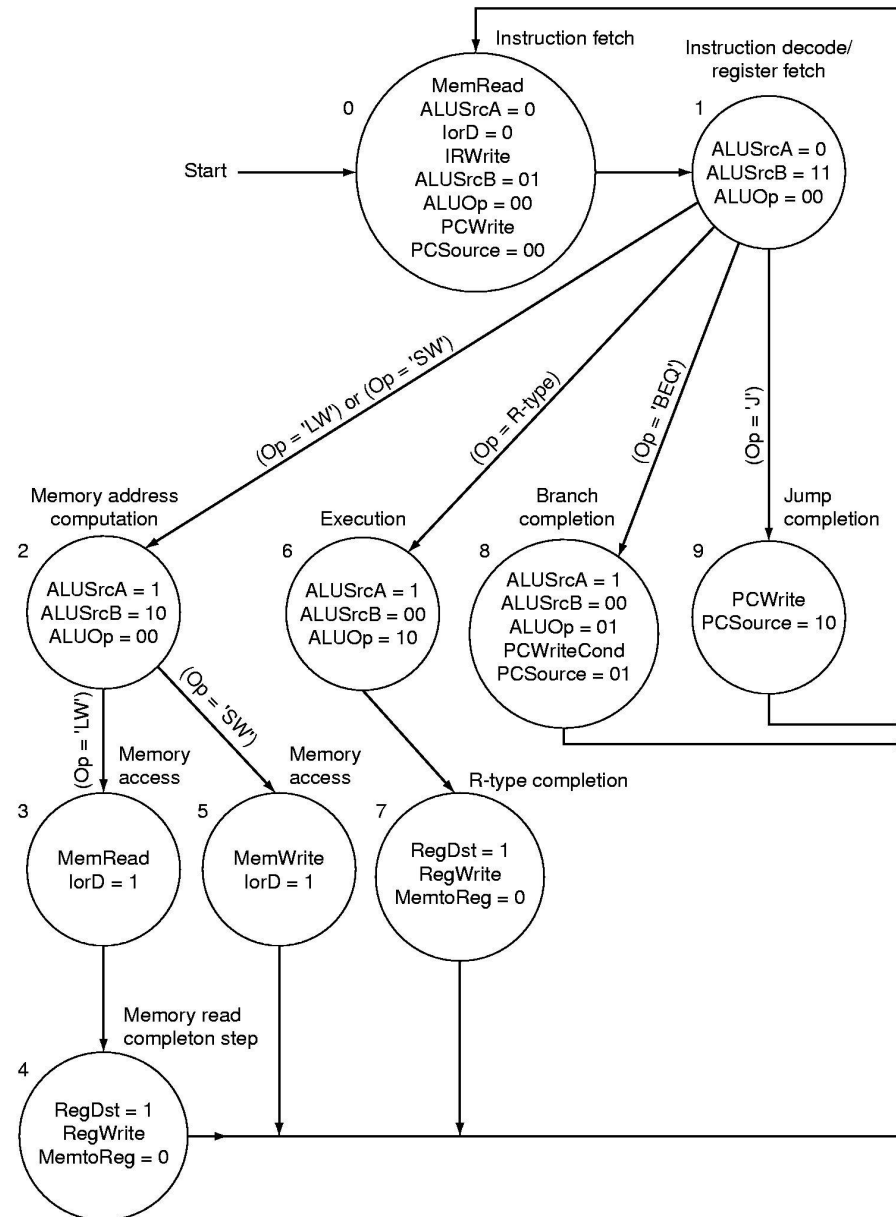
# Branch (5.35)



# Jump (5.36)



# Full State Diagram



# FSM

