Notes:

* What is a multicycle processor?
  + Processor that breaks an instruction into multiple steps and uses multiple cycles to execute.
* Why use a multicycle processor  
  (trade-offs over single cycle)
  + Combined instruction memory and data memory
    - Pros: Realistic, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
    - Cons: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  + Single ALU
    - Pros: Fewer complex parts, cheaper, reduced chip size
    - Cons: Needs non-architectural state registers and a finite state machine to share ALU
  + Faster clock
    - Pro: Clock is determined by slowest step not slowest instruction
* 5 Step Process for Handling Instructions
  + Fetch and PC increment
  + Decode and register readout
  + ALU Operation
    - \_\_\_\_\_\_\_\_\_\_\_\_\_\_
  + Memory Access/Register Write
    - \_\_\_\_\_\_\_\_\_\_\_\_\_\_
  + Register Write
    - \_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Datapath
  + Control Signals dictate the datapath
  + Different datapath for each of the following types of instructions
    - * \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
      * \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
      * \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Multicycle Performance
  + Performance is measured by Cycles Per Instruction (CPI)
  + CPI can be translated into execution time
  + Cycles per instruction
    - Load word - \_\_\_
    - Store word - \_\_\_
    - R type - \_\_\_
    - Branch - \_\_\_
    - Jump - \_\_\_
  + SpecInt2000 Benchmark
    - 25% loads
    - 10% stores
    - 11% branches
    - 2% jumps
    - 52% R-type instructions
    - Average CPI 4.12

# Assembly Description Address Machine

Main: addi $2, $0, 5 # initialize $2 = 5 0 20020005

addi $3, $0, 12 # initialize $3 = 12 4 2003000c

addi $7, $3, -9 # initialize $7 = 3 8 2067fff7

or $4, $7, $2 # $4 <= 3 or 5 = 7 c 00e22025

and $5, $3, $4 # $5 <= 12 and 7 = 4 10 00642824

add $5, $5, $4 # $5 = 4 + 7 = 11 14 00a42820

beq $5, $7, end # not taken 18 10a7000a

slt $4, $3, $4 # $4 = 12 < 7 = 0 1c 0064202a

beq $4, $0, Mid #taken 20 10800001

addi $5, $0, 0 # doesn’t happen 24 20050000

Mid: slt $4, $7, $2 # $4 = 3 < 5 = 1 28 00e2202a

add $7, $4, $5 # $7 = 1 + 11 = 12 2c 00853820

sub $7, $7, $2 # $7 = 12 - 5 = 7 30 00e23822

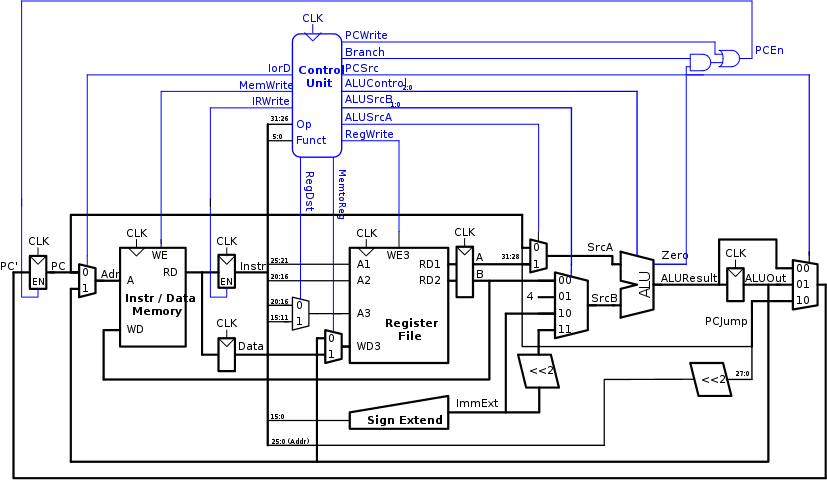
sw $7, 68($3) # [80] = 7 34 ac670044

lw $2, 80($0) # $2 = [80] = 7 38 8c020050

j end # should be taken 3c 08000011

addi $2, $0, 1 # shouldn’t happen 40 20020001

end: sw $2, 84($0) # write addr 84 = 7 44 ac020054

Figure : Datapath and Controller

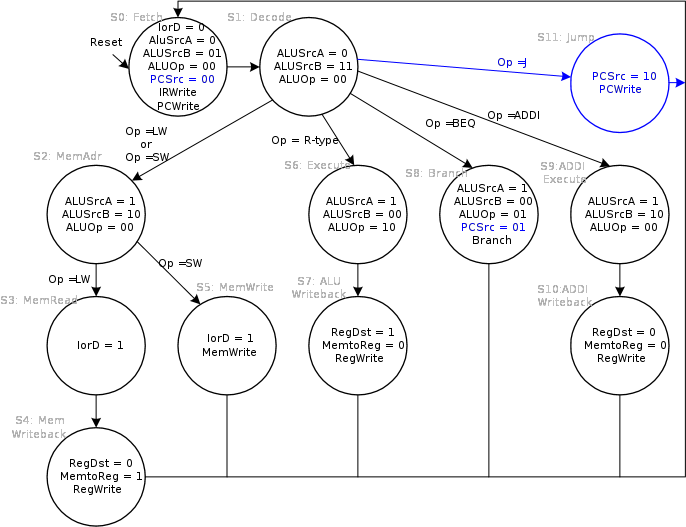


Figure : Controller Finite State Machine