CSE 2021
COMPUTER ORGANIZATION

HUGH CHESSER
CSEB 1012U

W10-M

Agenda

Topics:

1. Multiple cycle implementation – review
2. State Machine
3. Control Unit implementation for Multi-cycle core

Patterson: Section 4.5

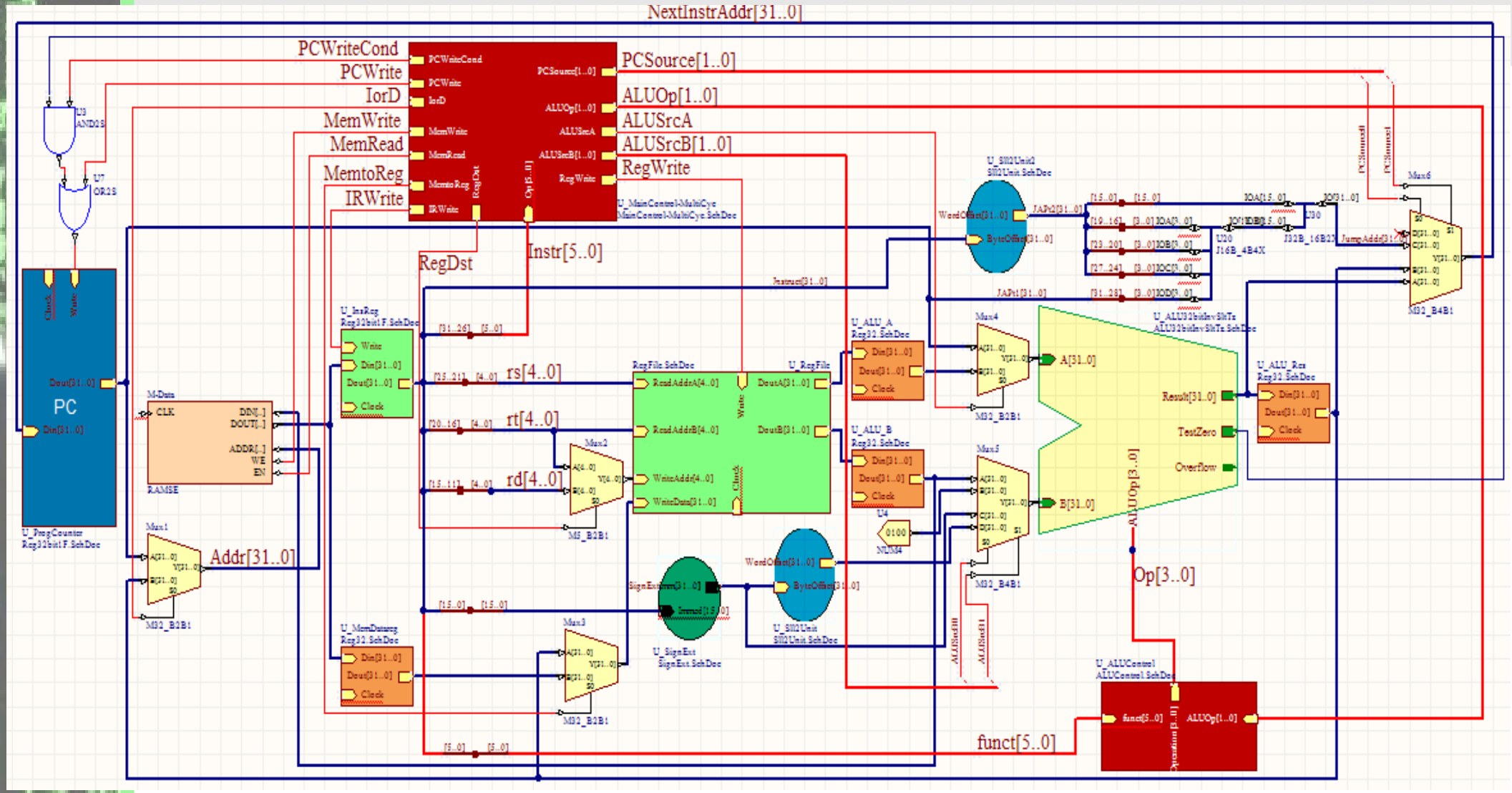
Reminder: Quiz #2 – Wednesday (November 10)

Breaking the Instruction Execution into Clock Cycles

Execution of each instruction is broken into a series of steps

- Each step is balanced to do almost equal amount of work
- Each step takes one clock cycle
- Each step contains at the most 1 ALU operation, or 1 register file access, or 1 memory access
- Operations listed in 1 step occurs in parallel in 1 clock cycle
- Different steps occur in different clock cycles
- Different steps are:
  1. IF: Instruction fetch step
  2. ID: Instruction decode and register fetch step
  3. EX: Execution, memory address computation, or branch completion step
  4. MEM: Memory access of R-type instruction completion step
  5. WB: Write back completion step

Multicycle Implementation: Control Units added



Action of 2-bit Control Signals

Control Input	Value	Effect
ALUOp	00	ALU performs an add operation
	01	ALU performs a subtract operation
	10	The function field of Instruction defines the operation of ALU
ALUSrcB	00	The second operand of ALU comes from Register B
	01	The second operand of ALU = 4
	10	The second operand of ALU is sign extended Instruction[15-0]
	11	The second operand of ALU is sign extended, 2-bit left shifted Instruction[15-0]
PCSource	00	Output of ALU (PC + 4) is sent to PC
	01	Contents of ALUOut (branch target address = PC + 4 + 4 x offset) is sent to PC
	10	Contents of Instruction[25-0], shift left by 2, and concatenated with the MSB 4-bits of PC is sent to PC (jump instruction)

Action of 1-bit Control Signals

Control Input	Effect when Deasserted (0)	Effect when asserted (1)
IorD	PC supplies address to memory (instruction fetch)	ALUout supplies address to memory (lw/sw)
MemRead	None	Memory content specified by address is placed on "Memdata" o/p (lw/any instruction)
MemWrite	None	I/p "Write data" is stored at specified address (sw)
IRWrite	None	"MemData" o/p is written on IR (instruction fetch)
RegDst	"Write Register" specified by Instruction[20-16] (lw)	"WriteRegister" specified by Instruction[15-11] (R-type)
RegWrite	None	Data from "WriteData" i/p is written on the register specified by "WriteRegister" number
ALUSrcA	PC is the first operand in ALU (increment PC)	Register A is the first operand in ALU
MemtoReg	"WriteData" of the register file comes from ALUOut	"WriteData" of the register file comes from MDR
PCWrite	Operation at PC depends on PCWriteCond and zero output of ALU	PC is written; Source is determined by PCSource
PCWriteCond	Operation at PC depends on PCWrite	PC is written if zero o/p of ALU = 1; Source is determined by PCSource

Summary of Steps used in different Instructions

Step Name	Action for			
	R-type Instruction	Memory Reference Instruction	Branch	Jump
IF - Instruction fetch	`IR = Memory[PC];` `PC = PC + 4;`			

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

EX – R-type Execution / address comp. / Branch /Jump	`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);`

MEM - Memory Access / R-type Completion	`Reg[IR[15-11]] = ALUOut;`	`lw:`
		`MDR = Memory[ALUOut]`
		`or sw:`
		`Memory[ALUOut] = B`		

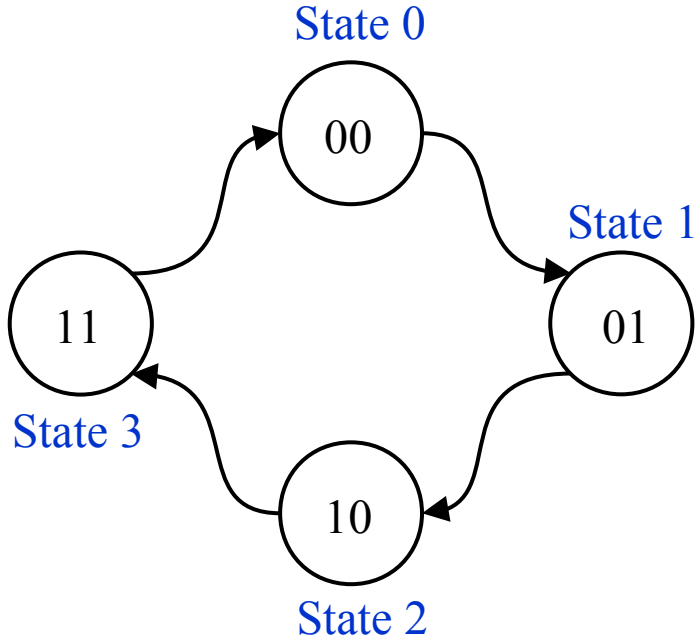
WB - Memory Read Completion		`lw:` `Reg[IR[20-16]]=MDR;`		

Multipath Datapath Implementation: Control

- Recall that design of single cycle datapath was based on a combinational circuit
- Design of multicycle datapath is more complicated
  1. Instructions are executed in a series of steps
  2. Each step must occur in a sequence
  3. Control of multicycle must specify both the control signals and the next step
- The control of a multicycle datapath is based on a sequential circuit referred to as a finite state machine

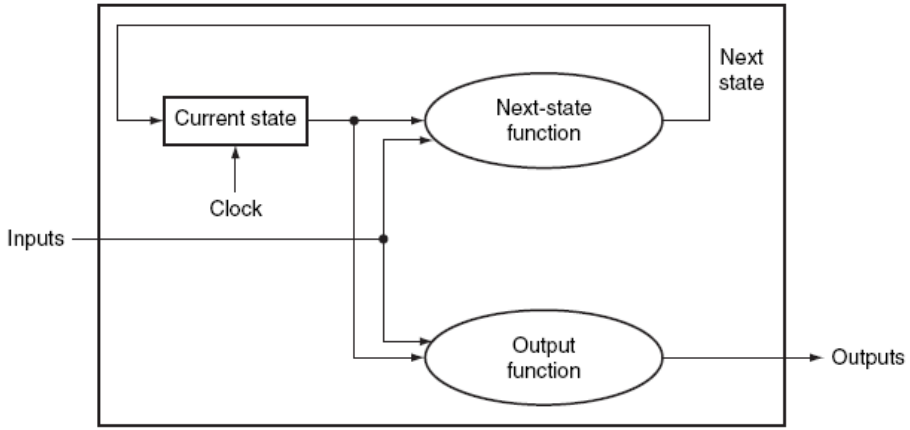
A finite state diagram for a 2-bit counter

- Each state specifies a set of outputs
- By default, unspecified outputs are assumed disabled
- The number of the arrows identify inputs

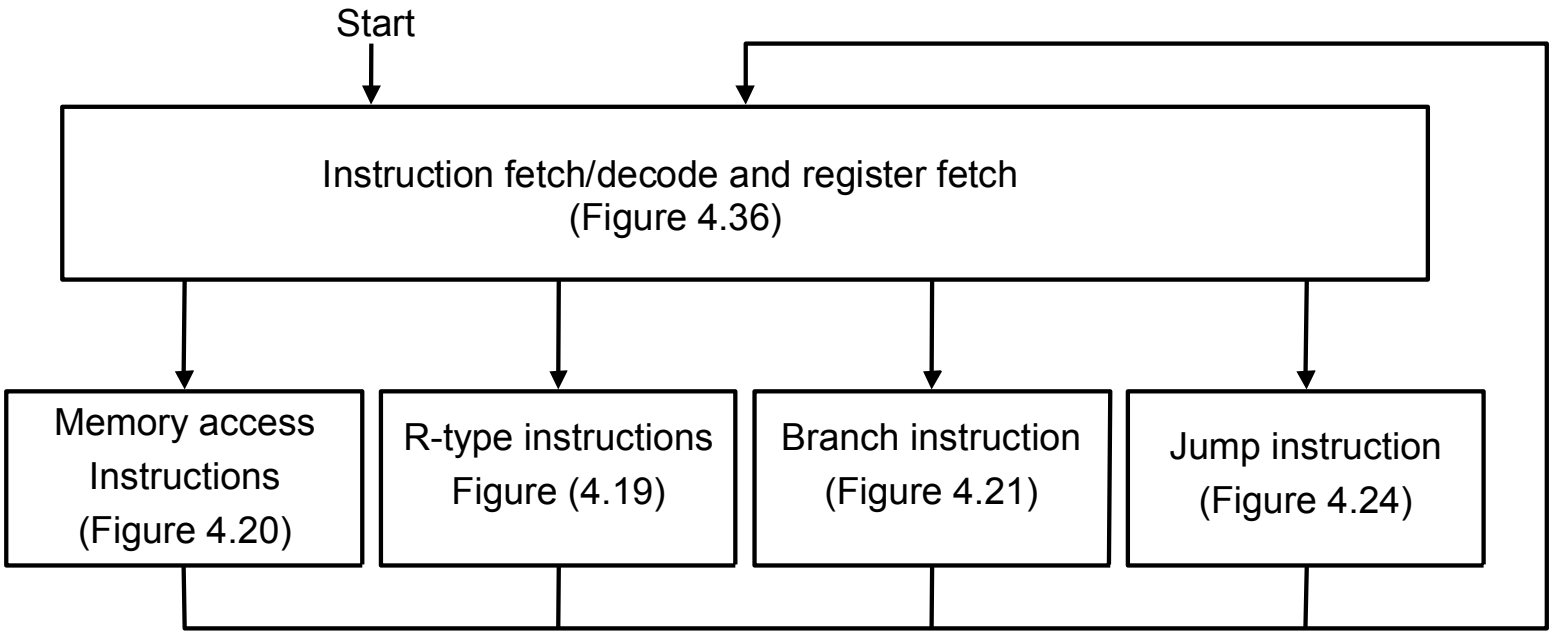


Finite State Machine?

- See Appendix C
  - A sequential logic function which has a state and inputs – the logic function determines the next state and outputs
    - Moore machine – outputs depend on just the current state
    - Mealy machine – outputs depend on current state and inputs
  - Book uses Moore machine description



Finite State Machine Control of Multicycle Datapath (1)



High-Level View

Finite State Machine Control of Multicycle Datapath (2)

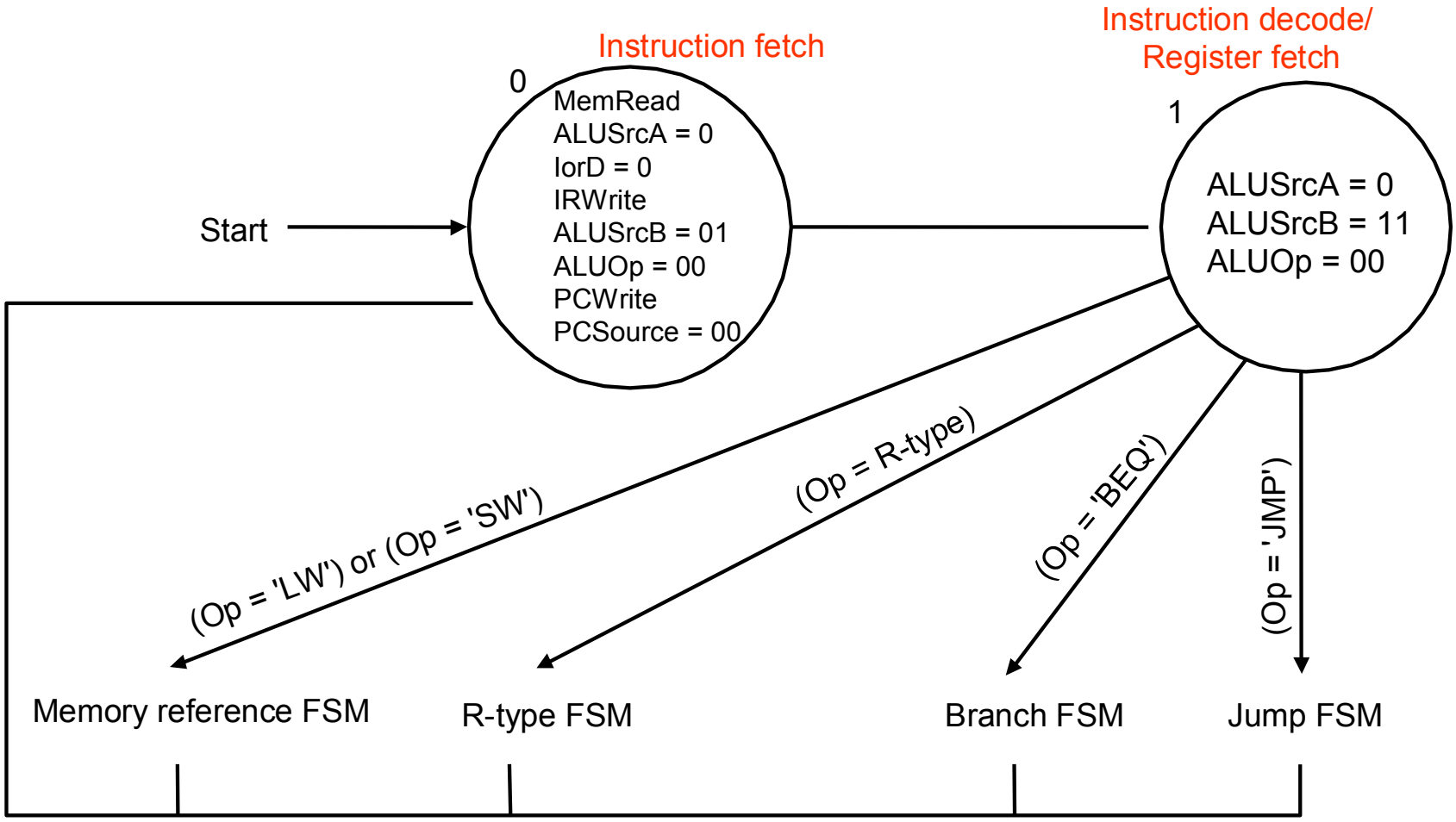
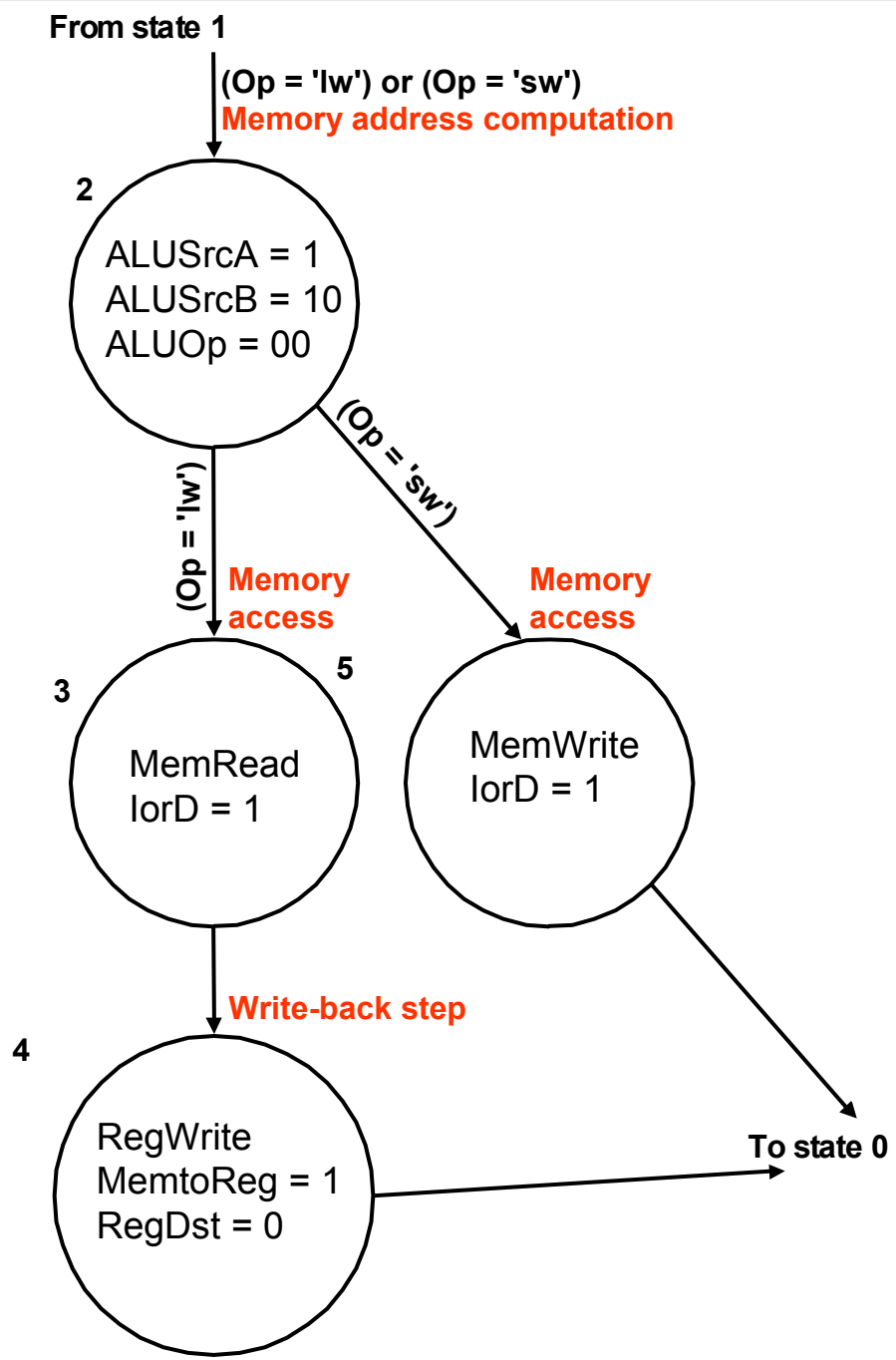


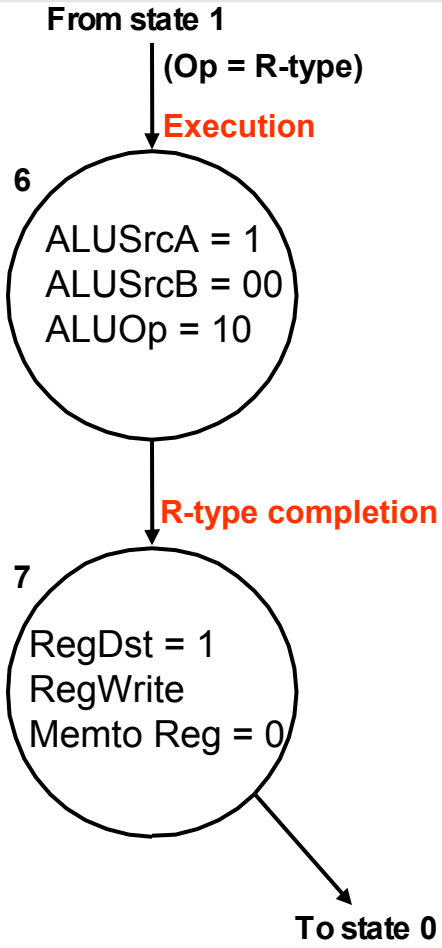
Fig. D.3.1: Steps 1 and 2: Instruction Fetch and Decode Instructions

Finite State Machine Control of Multicycle Datapath (3)

Finite State Machine for
Memory Reference Instructions

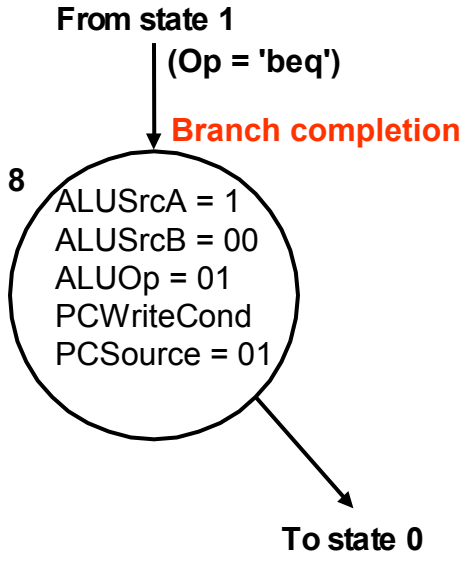


Finite State Machine Control of Multicycle Datapath (4)

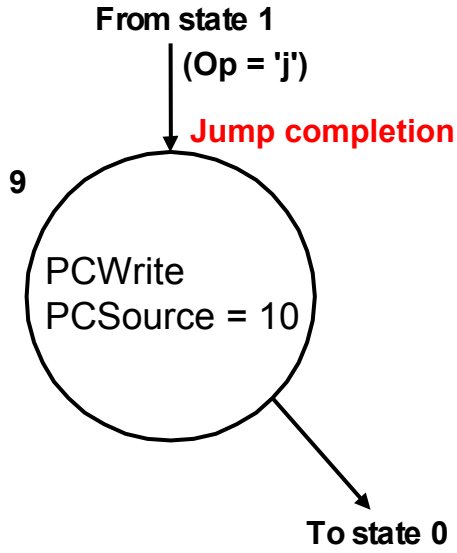


Finite State Machines for R-type Instructions

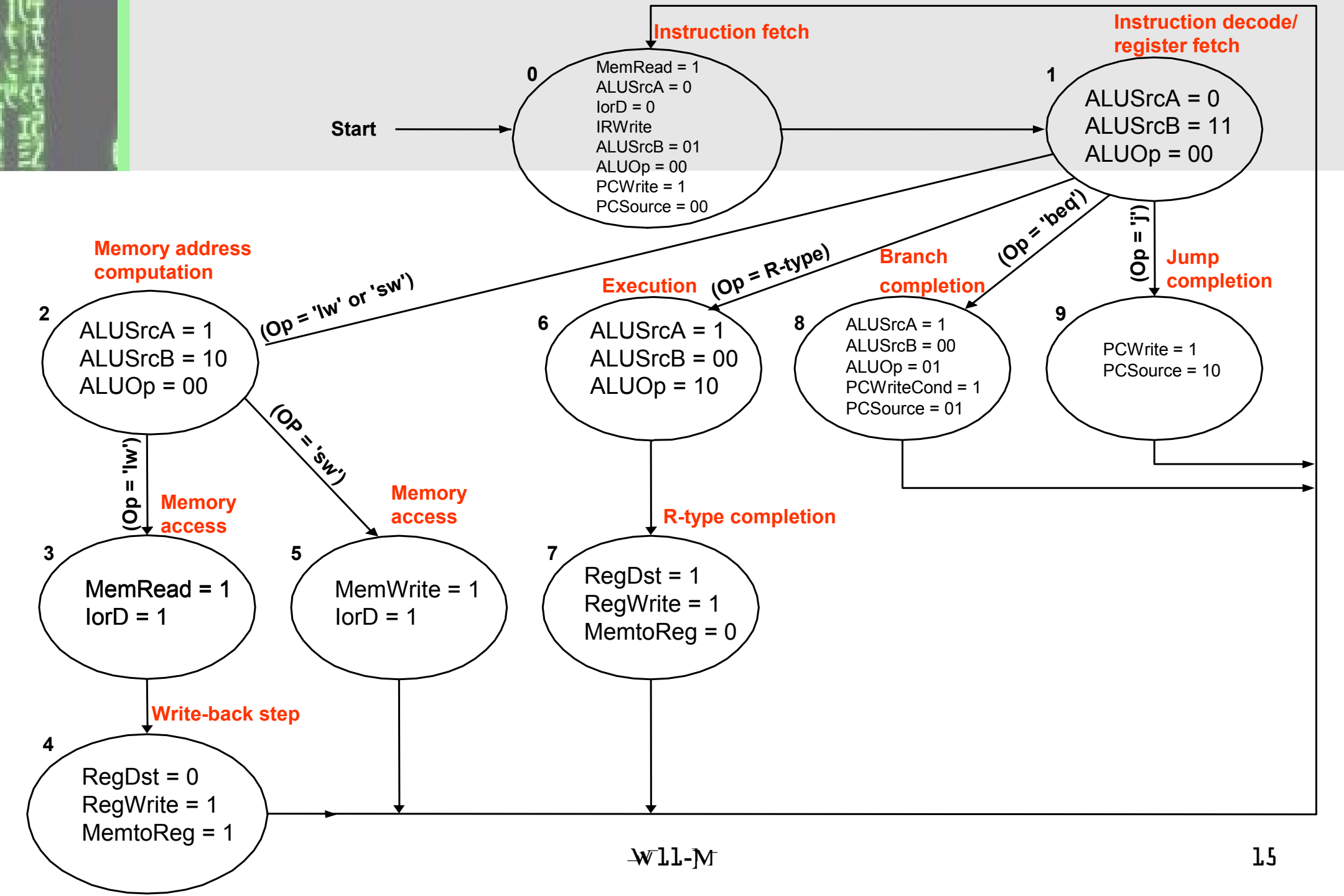
Finite State Machine Control of Multicycle Datapath (5)



Finite State Machine for Branch Instruction

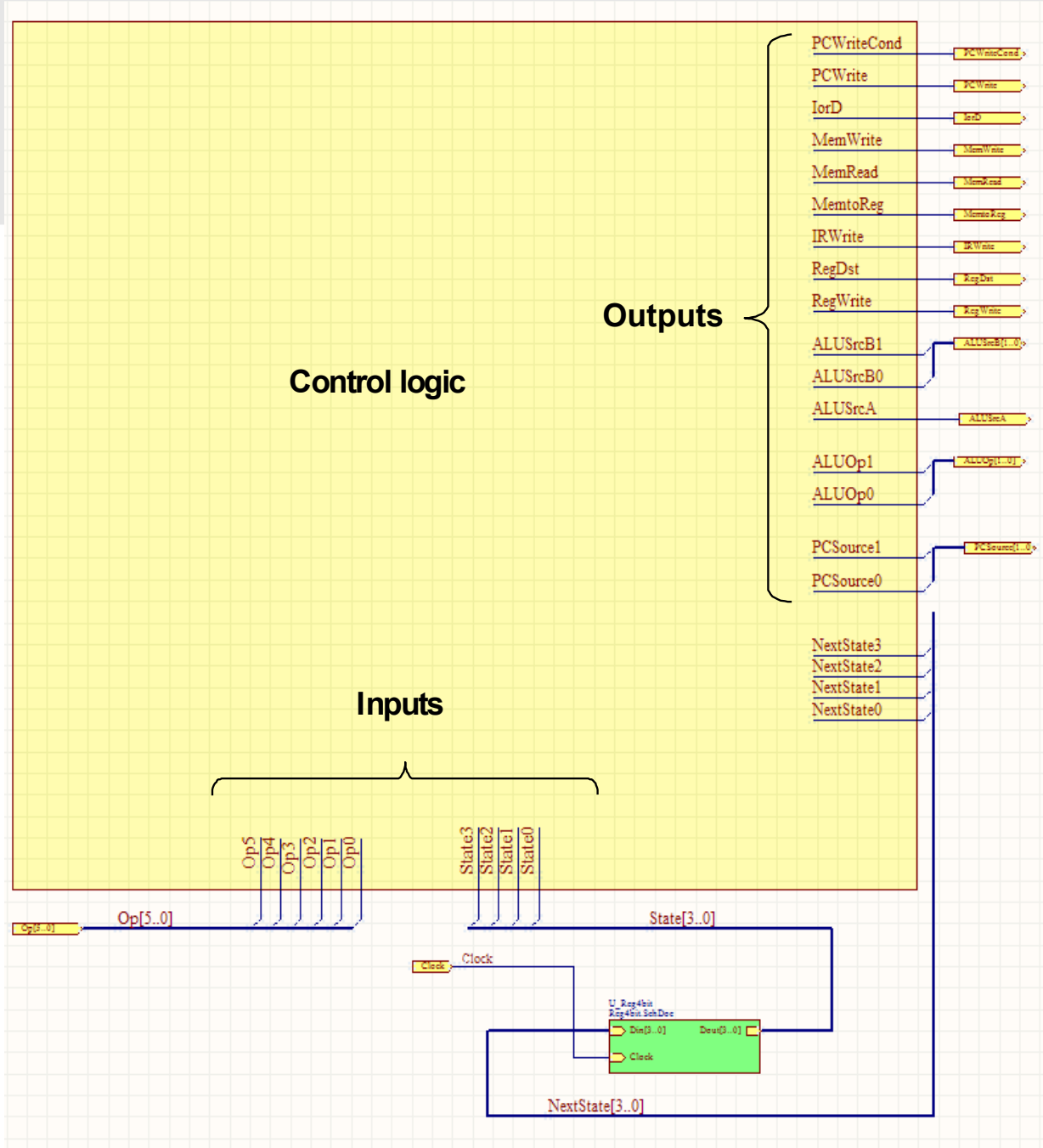


Finite State Machine for Jump Instruction


Start
0
MemRead = 1
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite = 1
PCSource = 00
Instruction fetch
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
Instruction decode/
register fetch
(Op = 'lw' or 'sw')
(Op = R-type)
(Op = 'beq')
(Op = 'j')
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
Memory address
computation
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10
Execution
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond = 1
PCSource = 01
Branch
completion
9
PCWrite = 1
PCSource = 10
Jump
completion
(Op = 'lw')
(OP = 'sw')
3
MemRead = 1
IorD = 1
Memory
access
5
MemWrite = 1
IorD = 1
Memory
access
7
RegDst = 1
RegWrite = 1
MemtoReg = 0
R-type completion
Write-back step
4
RegDst = 0
RegWrite = 1
MemtoReg = 1

Finite State Machine

Control of Multicycle Datapath (5)



Control Logic – Truth Table

Note that control outputs depend only on current state (Op column is blank for all output rows)

Next state depends on current state and inputs (opcode from instruction)

Output	Current states	Op
PCWrite	state0 + state9	
PCWriteCond	state8	
IorD	state3 + state5	
MemRead	state0 + state3	
MemWrite	state5	
IRWrite	state0	
MemtoReg	state4	
PCSource1	state9	
PCSource0	state8	
ALUOp1	state6	
ALUOp0	state8	
ALUSrcB1	state1 +state2	
ALUSrcB0	state0 + state1	
ALUSrcA	state2 + state6 + state8	
RegWrite	state4 + state7	
RegDst	state7	
NextState0	state4 + state5 + state7 + state8 + state9	
NextState1	state0	
NextState2	state1	(Op = 'lw') + (Op = 'sw')
NextState3	state2	(Op = 'lw')
NextState4	state3	
NextState5	state2	(Op = 'sw')
NextState6	state1	(Op = 'R-type')
NextState7	state6	
NextState8	state1	(Op = 'beq')
NextState9	state1	(Op = 'jmp')