


COSC175 (Systems I): Computer Organization & Design

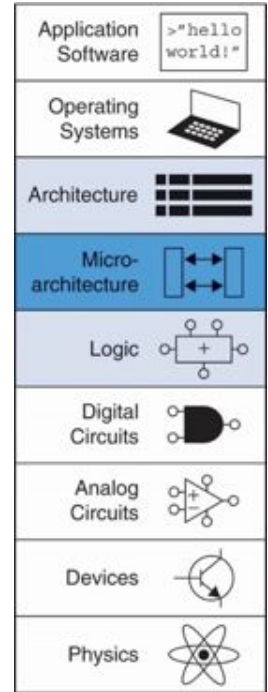


Professor Lillian Pentecost
Fall 2024



Warm-Up November 21

- Where we were
 - Alternative Microarchitectures
- Where we are going
 - Supporting more instructions, highlighting control signals
- Logistics, Reminders
 - TA help 7-9PM on Sundays, Tuesdays, Thursdays in C107
 - Use to start review of whole semester!
 - LP Office hours M 9-10:30AM, Th 2:30-4PM
 - Weekly Exercises due Friday 5PM
 - **No more weekly exercises!** Focus on your final lab, and starting to review previous assignments + exercises
 - Before next Wednesday's lab, you will need to set up a Vivado Project with a functional single-cycle processor design
 - **ATTEND FACULTY CANDIDATE TALK TODAY AT 4:30PM in A131, WITH SNACKS @ 4PM in C209**
 - *If you attend at least 3 candidate talks, then send me an email including 1 thing you learned from each talk, I'll give you 5% extra credit on any previous lab report*



Single- vs. Multicycle Processor

- **Single-cycle:**

- + simple
- cycle time limited by longest instruction (lw)
- separate memories for instruction and data
- 3 adders/ALUs

- **Multicycle:**

- + higher clock speed
- + simpler instructions run faster
- + reuse expensive hardware on multiple cycles
- sequencing overhead paid many times

Same design steps as single-cycle:

- first datapath
- then control

Goals for Today

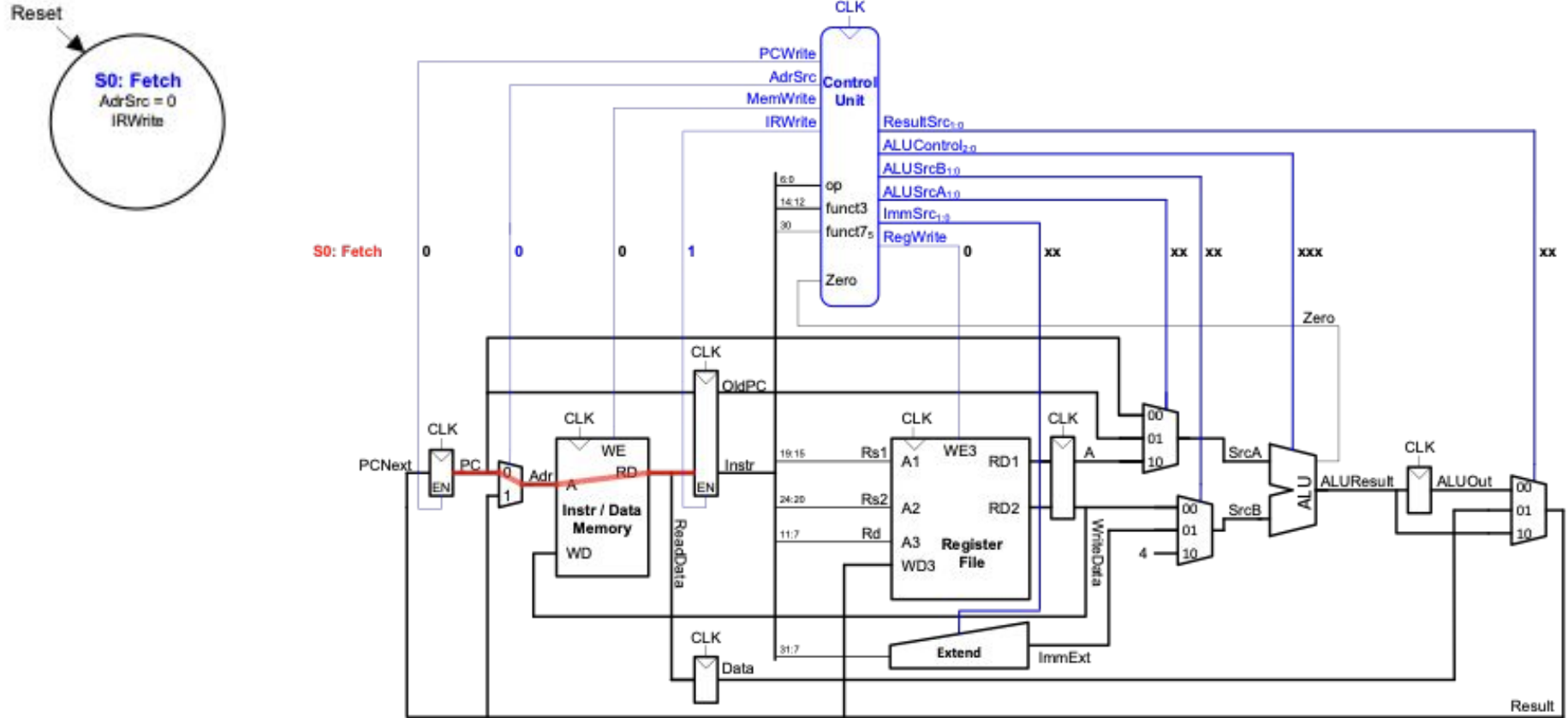
- **Continue to build out Main FSM for multicycle processor**
- **Keep in mind: what are the pros and cons of this design vs. single-cycle?**
- **How is the performance?**

Same design steps as single-cycle:

- first datapath
- then control

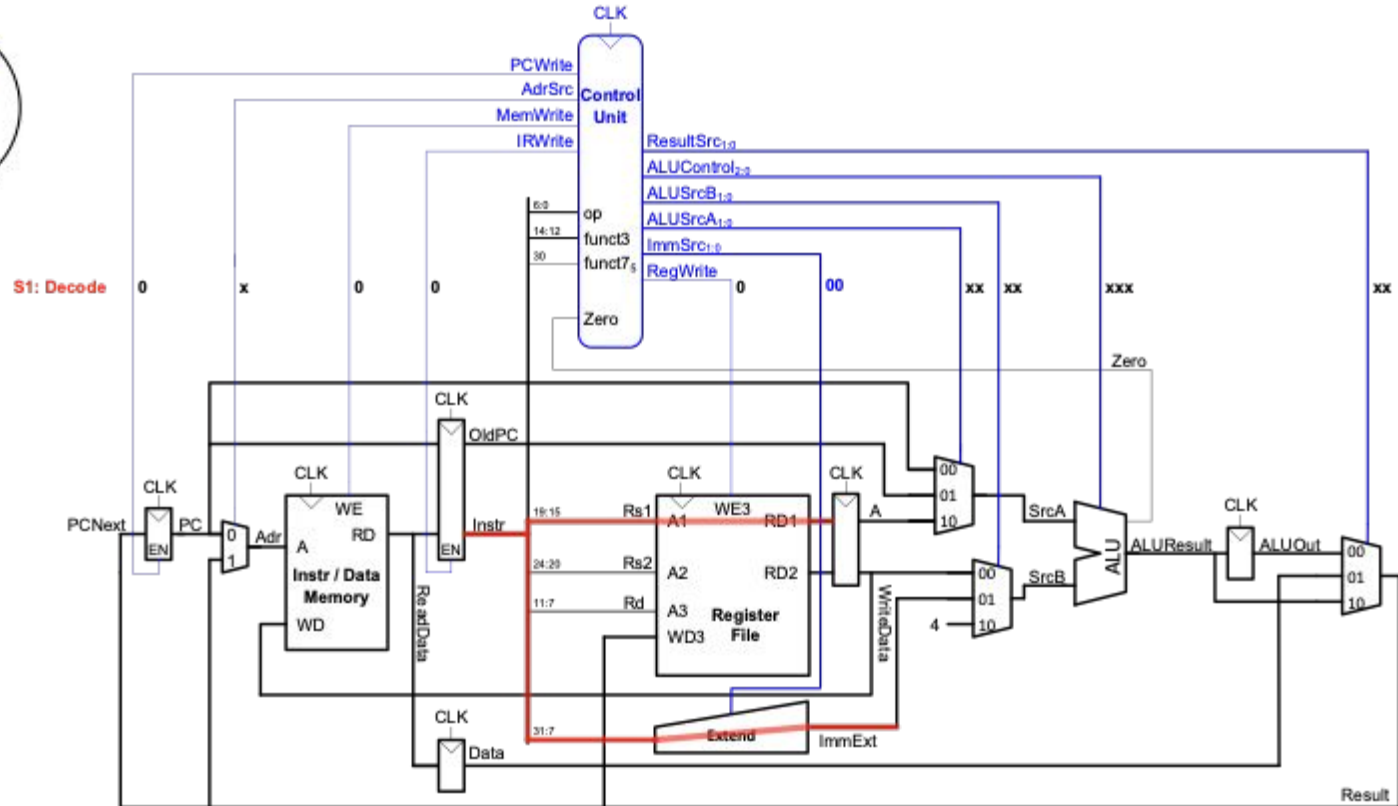
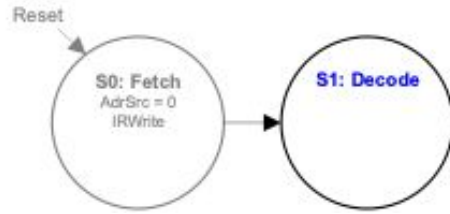
... remember the memory hierarchy, tho?

Main FSM: Fetch (S0)

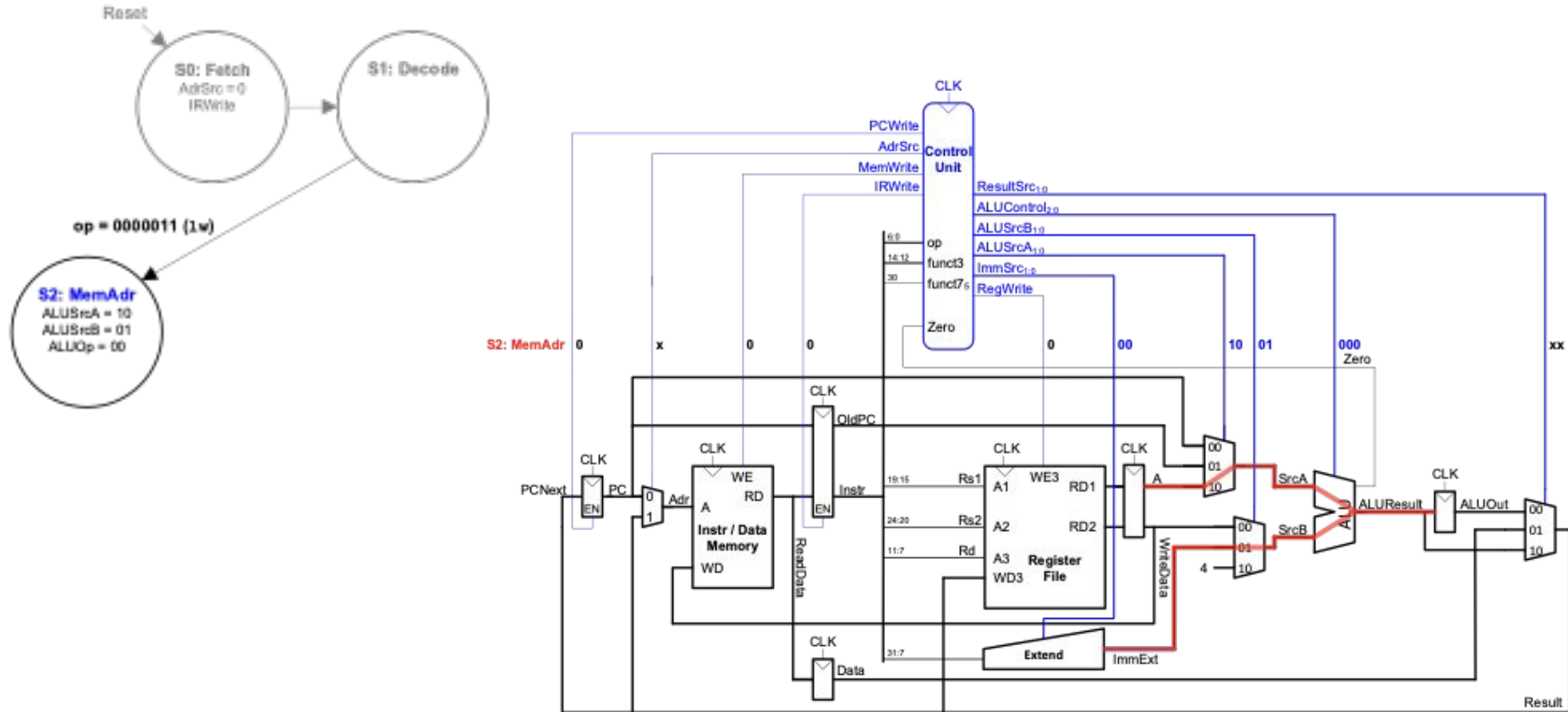


Main FSM: Decode (S1)

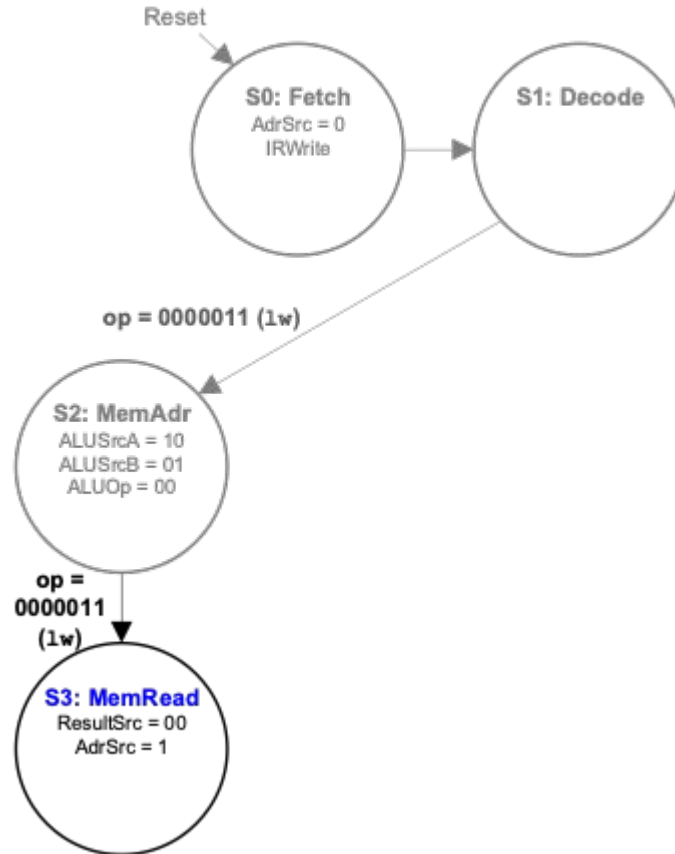
ImmSrc is determined by the Instruction Decoder



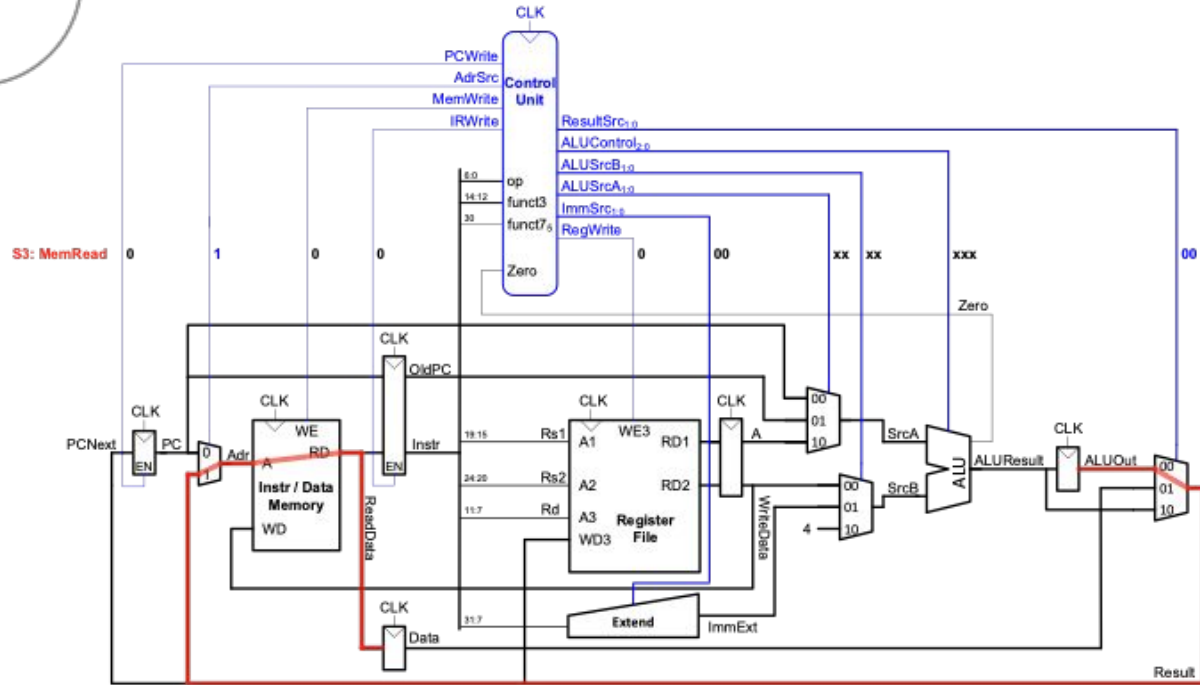
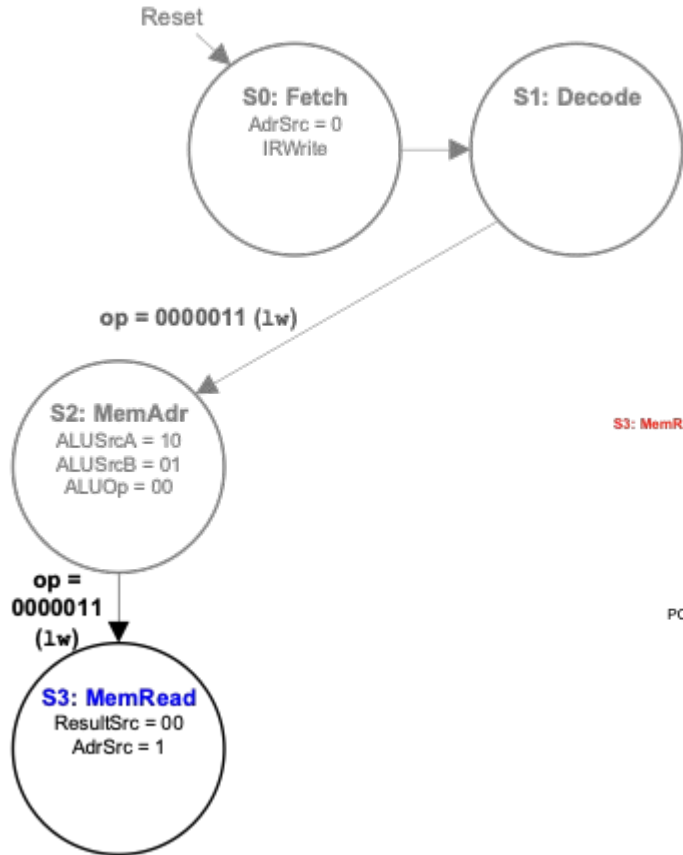
Main FSM: Generate Address (S2)



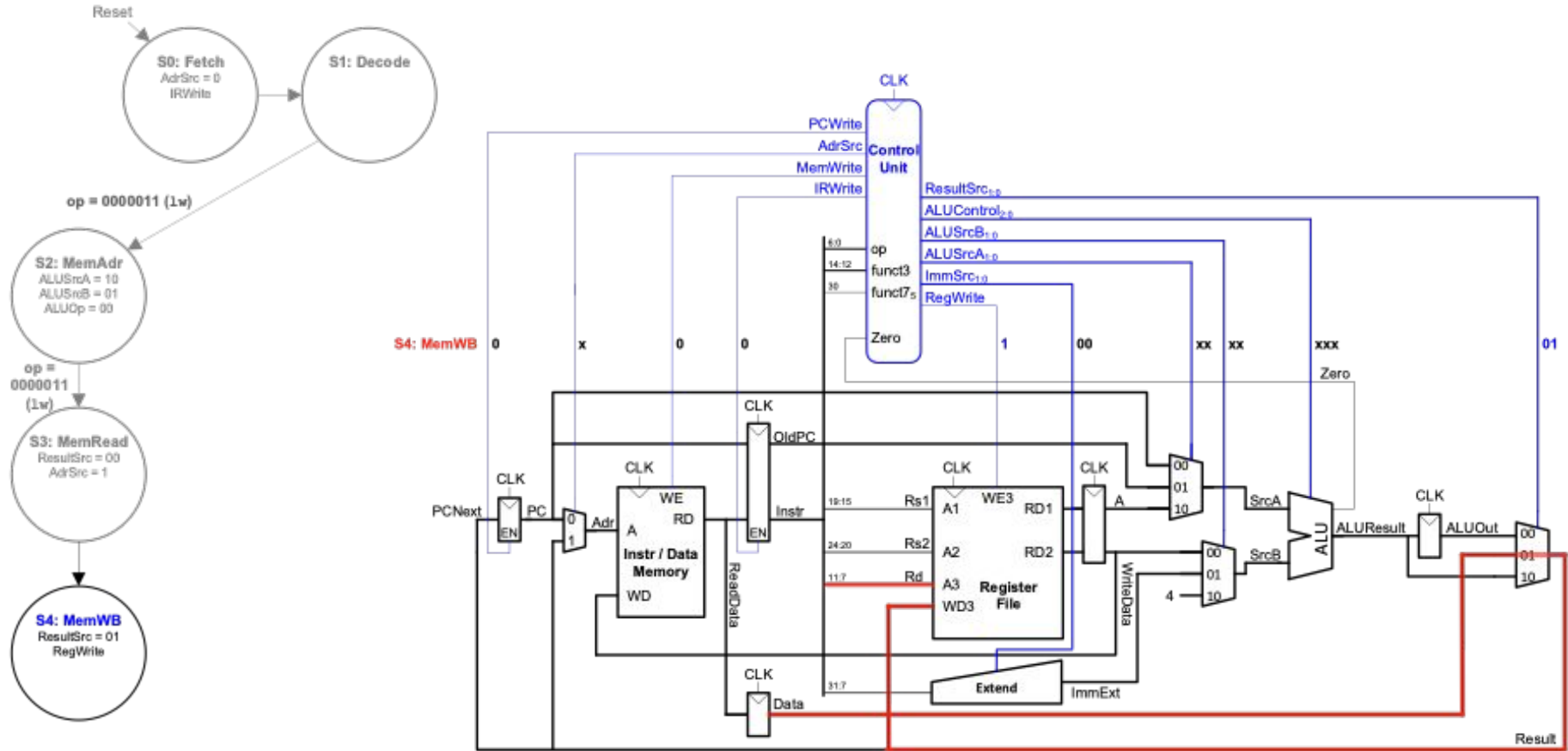
Main FSM: Read Memory (S3)



Main FSM: Read Memory (S3)

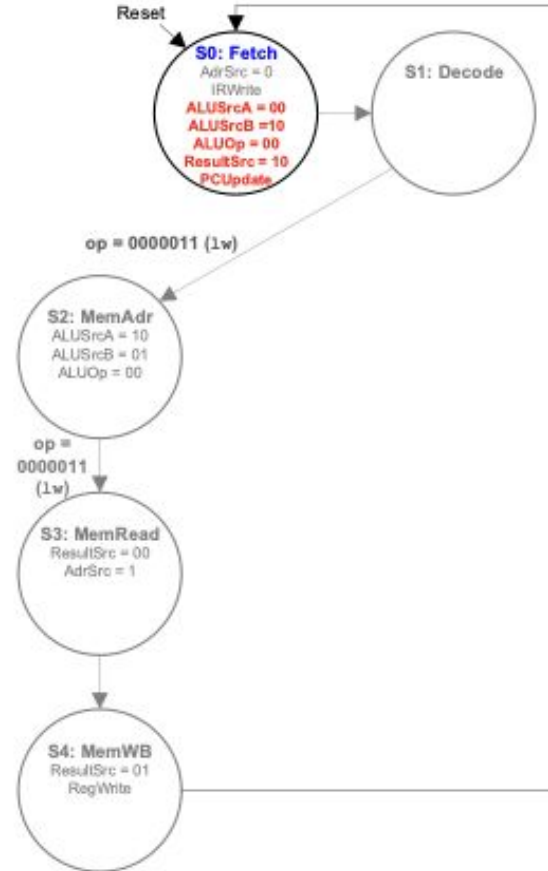


Main FSM: Write Back to RF (S4)



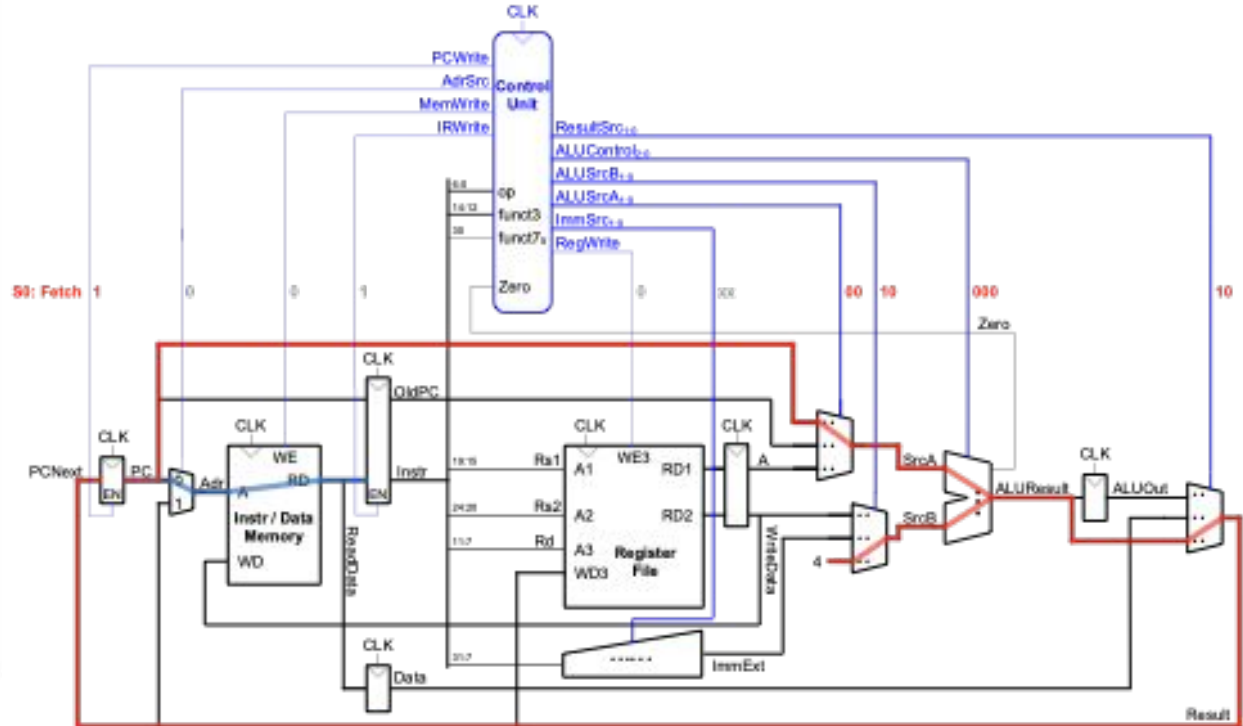
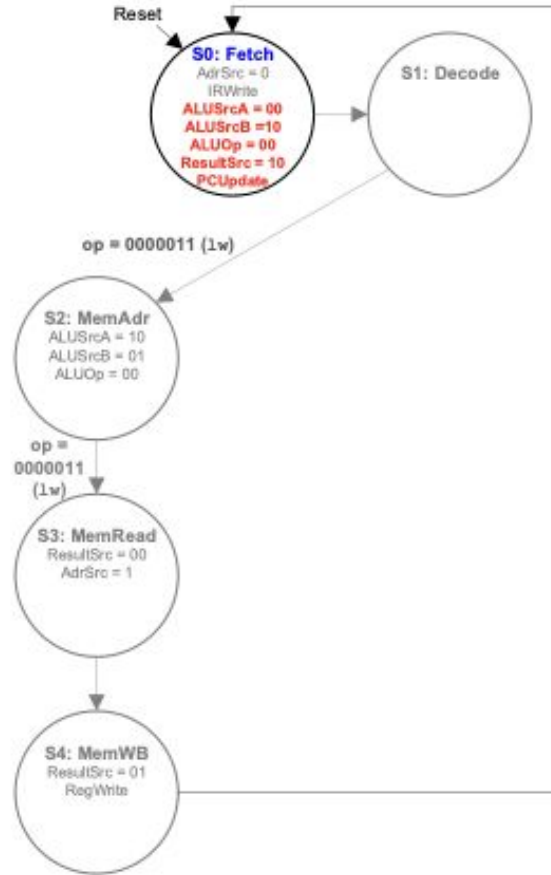
Main FSM: Fetch Revisited (S0)

*Opportunity to
combine steps:*
Calculate **PC+4**
during Fetch
stage (ALU isn't
being used)

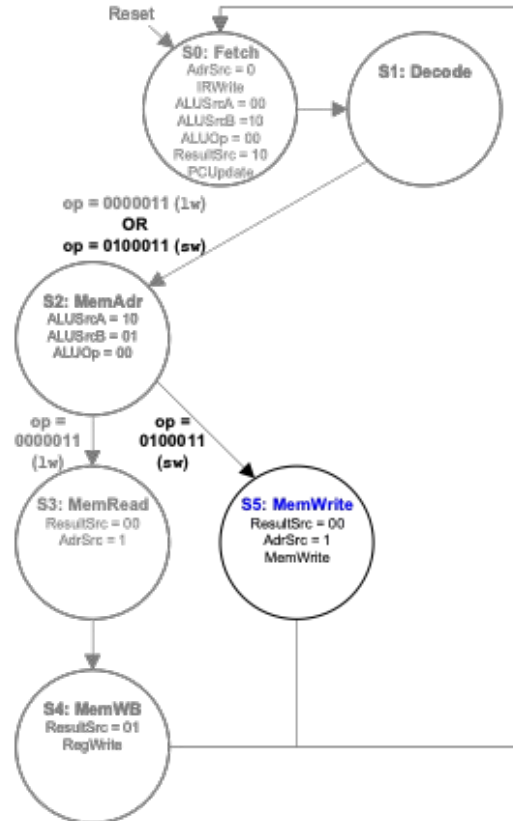


Main FSM: Fetch (PC+4) Datapath

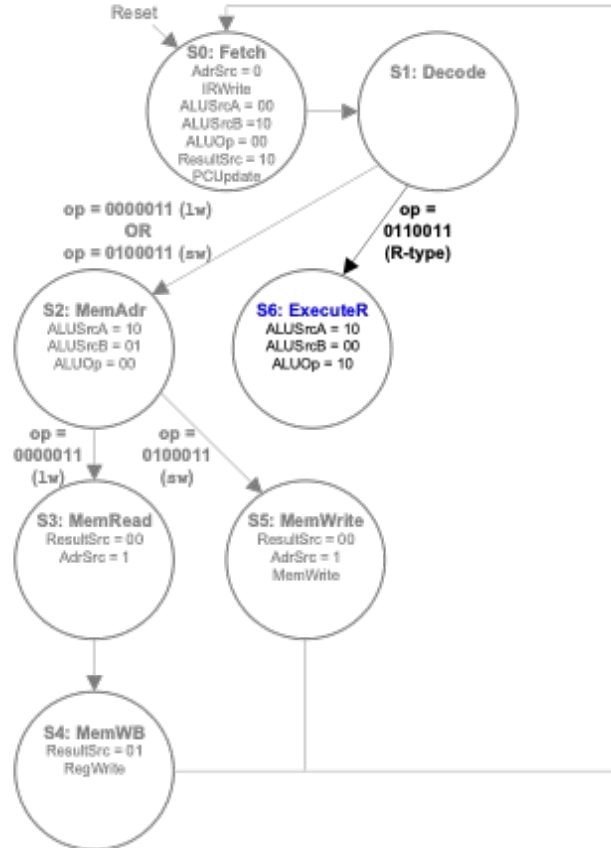
**Fetch Instruction and
Increment PC**



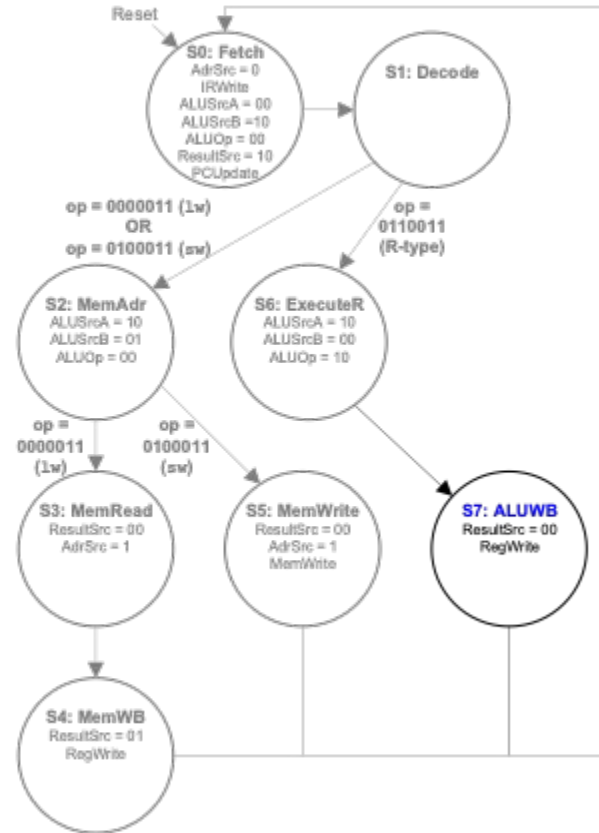
Main FSM: add support for S_W



Main FSM: Add support for R-Type: Execution



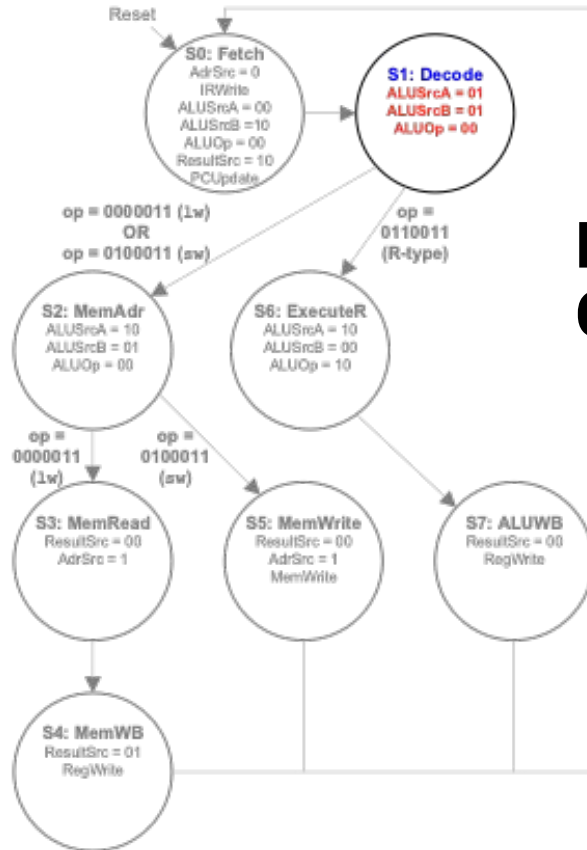
Main FSM: Add support for R-Type: ALU Write Back



Main FSM: add support for beq

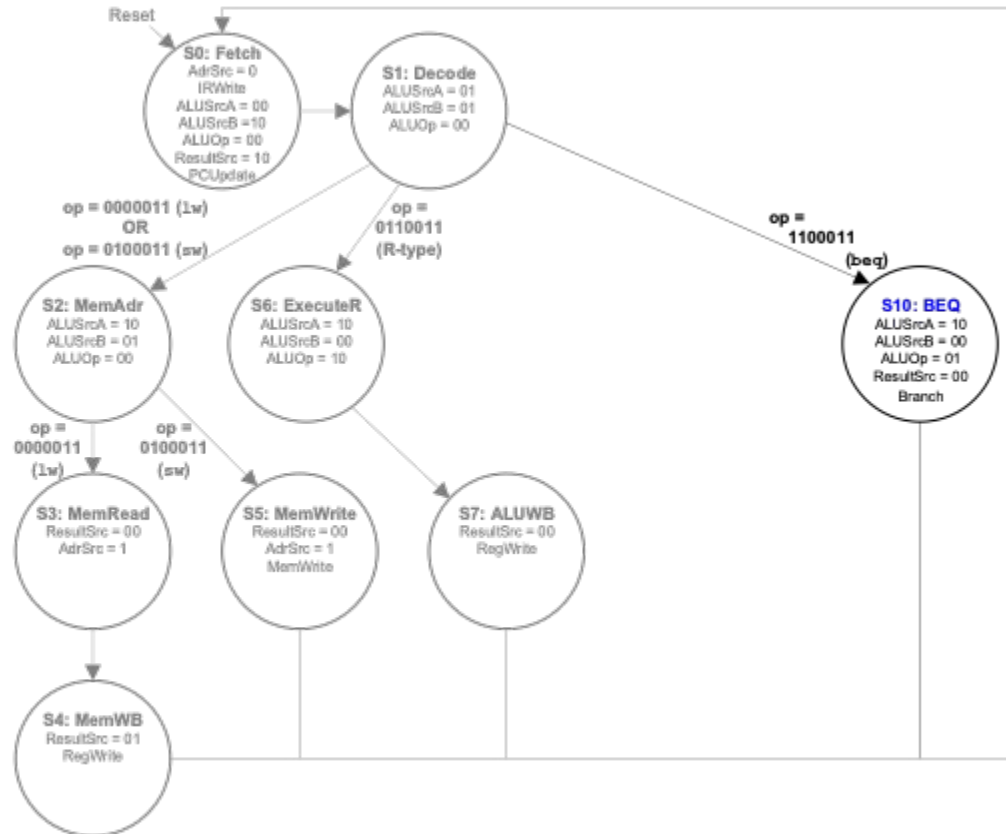
- **Need to calculate:**
 - Branch Target Address
 - **rs1 - rs2** (to see if equal)
- **ALU** isn't being used in Decode stage
 - Use it to calculate Target Address ($PC + imm$)
- Don't add more states if HW is underutilized during an existing state!

Main FSM: Update decode to support **beq**



**Read Registers and
Calculate Target Address (PC+imm)**

Main FSM: support (rs1 - rs2) for beq



Main FSM: Datapath including beq

Compare registers

Send Target PC (ALUOut) to PCNext

S10: BEQ

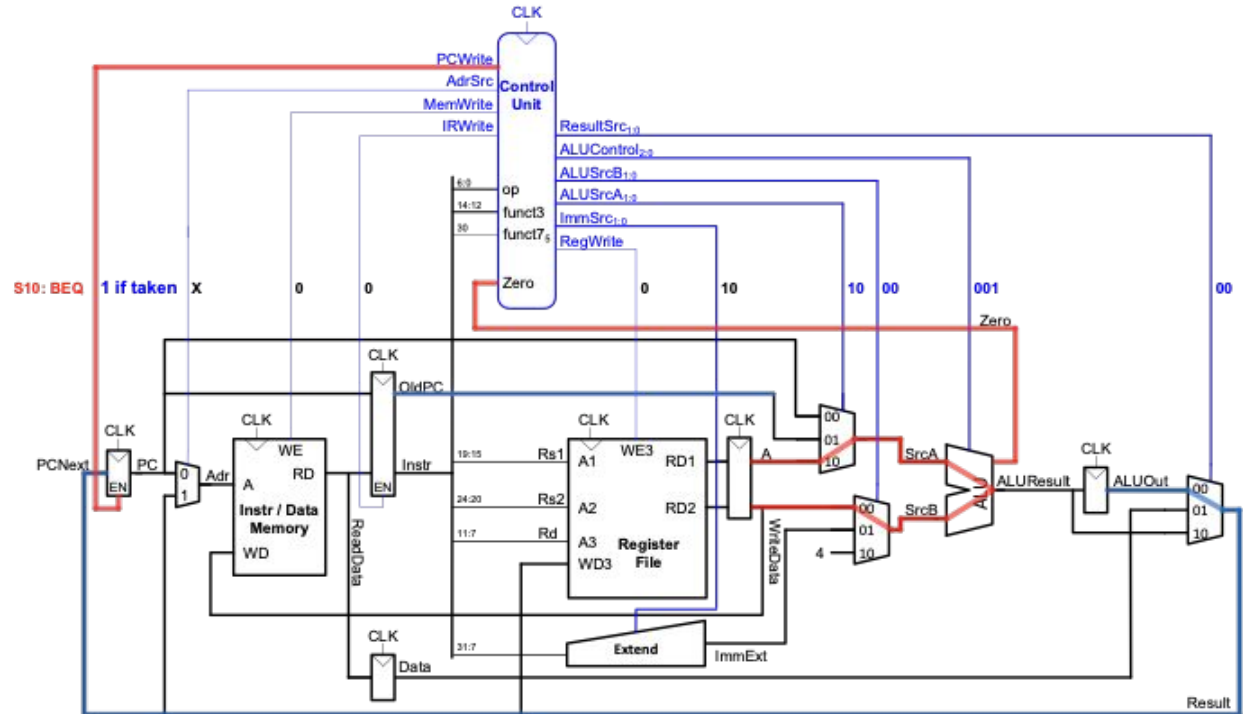
ALUSrcA = 10

ALUSrcB = 00

ALUOp = 01

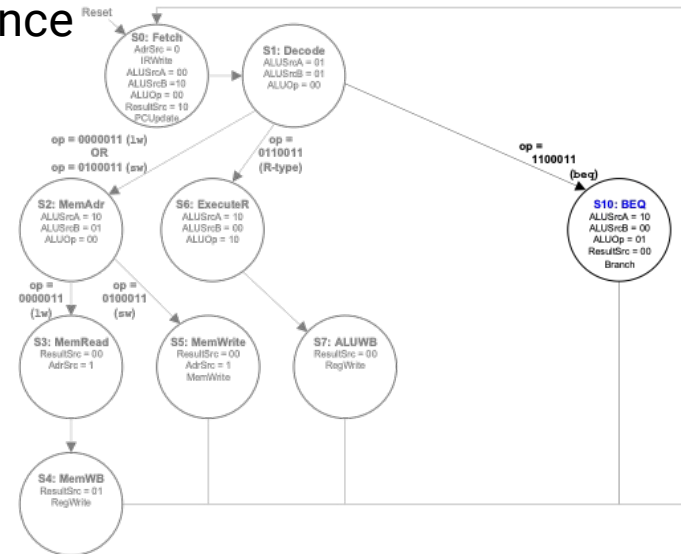
ResultSrc = 00

Branch



Multicycle Processor Performance

- Instructions take different number of cycles:
 - 3 cycles: beq
 - 4 cycles: R-type, addi, sw , jal
 - 5 cycles: lw
- Revisit our expression for program performance
 - What information do we need?

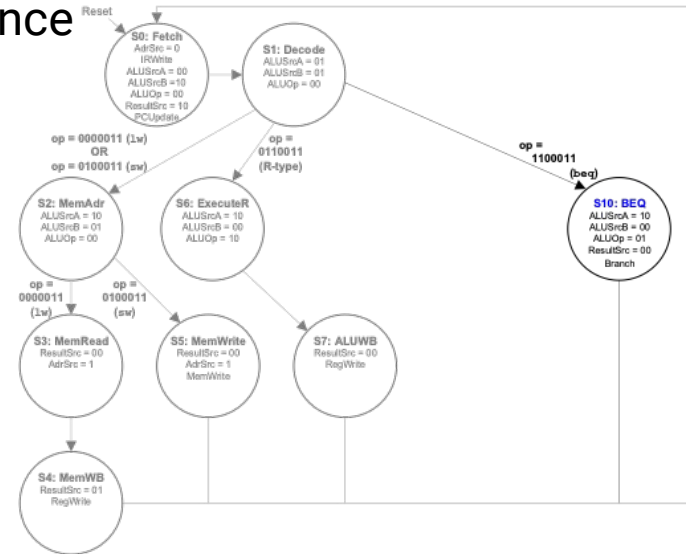


Multicycle Processor Performance

- Instructions take different number of cycles:
 - 3 cycles: beq
 - 4 cycles: R-type, addi, sw , jal
 - 5 cycles: lw
- Revisit our expression for program performance
 - What information do we need?

CPI (cycles per instruction)

Clock Cycle (seconds)



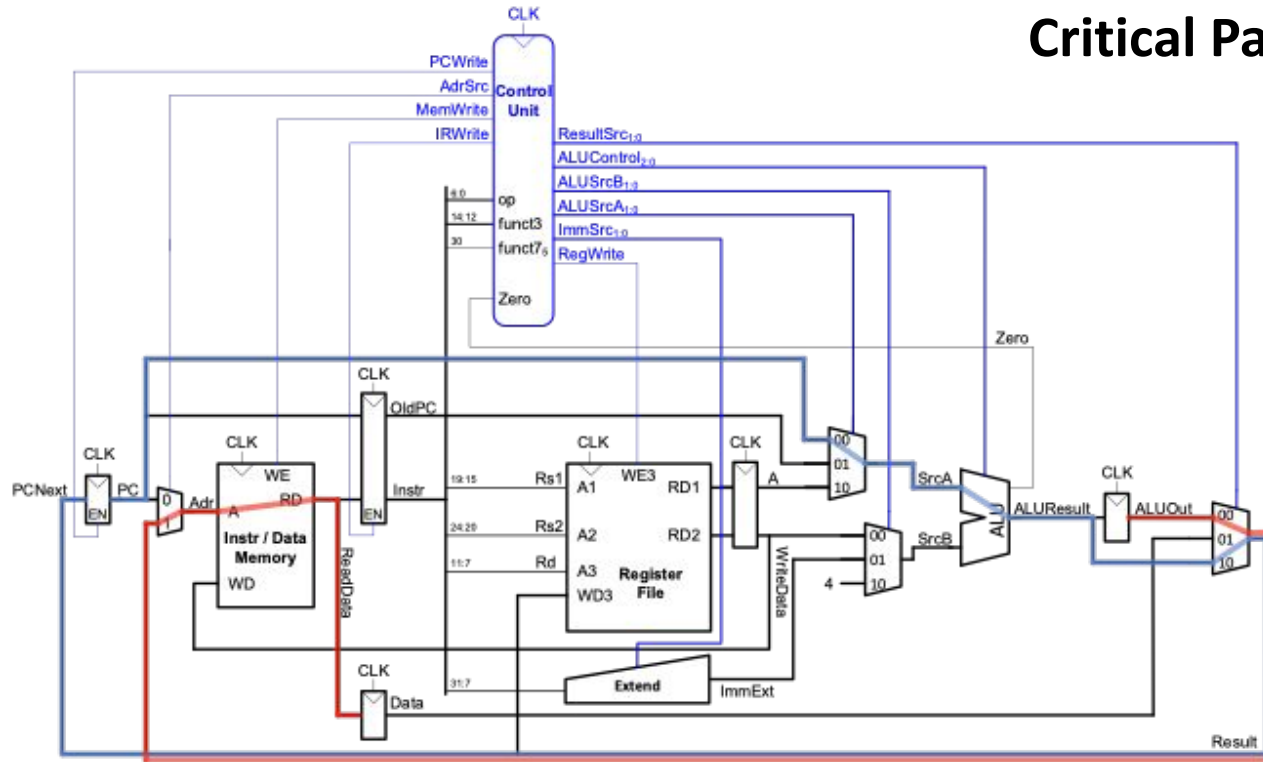
Multicycle Processor Performance: CPI

- Instructions take different number of cycles:
 - 3 cycles: beq
 - 4 cycles: R-type, addi, sw , jal
 - 5 cycles: lw
- CPI is weighted average, depends on frequency of different types of instructions
- SPECINT2000 benchmark (a “standard” set of programs):
 - **25%** loads
 - **10%** stores
 - **13%** branches
 - **52%** R-type

$$\text{Average CPI} = (0.13)(3) + (0.52 + 0.10)(4) + (0.25)(5) = 4.12$$

Multicycle Critical Path: cycle length

Critical Path: **Read Memory (S3)**



$$T_{c_multi} = t_{pcq} + t_{dec} + 2t_{mux} + \max(t_{ALU}, t_{mem}) + t_{setup}$$

Multicycle Performance Example

Element	Parameter	Delay (ps)
Register clock-to-Q	t_{pcq_PC}	40
Register setup	t_{setup}	50
Multiplexer	t_{mux}	30
AND-OR gate	t_{AND-OR}	20
ALU	t_{ALU}	120
Decoder (Control Unit)	t_{dec}	25
Extend unit	t_{dec}	35
Memory read	t_{mem}	200
Register file read	t_{RFread}	100
Register file setup	$t_{RFsetup}$	60

$$\begin{aligned}T_{c_multi} &= t_{pcq} + t_{dec} + 2t_{mux} + \max(t_{ALU}, t_{mem}) + t_{setup} \\&= (40 + 25 + 2*30 + 200 + 50) \text{ ps} = \mathbf{375 \text{ ps}}\end{aligned}$$

Multicycle Performance Example

For a program with **100 billion** instructions executing on a **multicycle** RISC-V processor

– **CPI** = 4.12 cycles/instruction

– **Clock cycle time:** $T_{c_multi} = 375 \text{ ps}$

$$\begin{aligned}\text{Execution Time} &= (\# \text{ instructions}) \times \text{CPI} \times T_c \\ &= (100 \times 10^9)(4.12)(375 \times 10^{-12}) \\ &= \mathbf{155 \text{ seconds}}\end{aligned}$$

This is *slower* than the single-cycle processor (75 sec.)

Check-In: Multicycle Performance Analysis

For each of the programs on the worksheet, compute how many ***cycles*** it takes to run this program with our multicycle processor design.

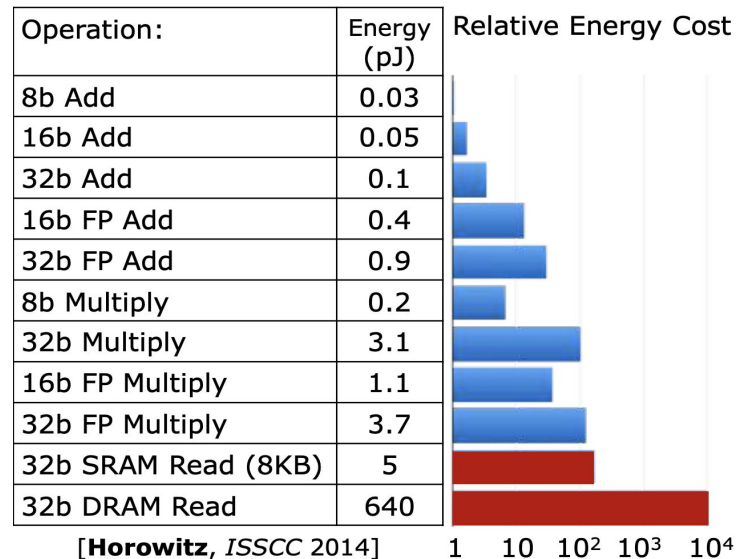
Then, you will brainstorm and compare to our single-cycle design.

Remember to assign roles and take notes to hand in!

Computer design question to worry about over break

- We have been using an ideal (super unrealistic) memory array
- How can we maintain program performance if memory access actually takes at least **10-100x** longer than our ALU?

	Capacity	Read Latency (approx)
RegFile	< 1KB	10-100ps
SRAM	1-64MB	1-10ns
DRAM	4-64GB	~100ns (maybe more)
Flash, SSD, hard drive	256-1024GB	~100us



Wrap-Up November 21



- Coming up next!
 - More and more elements and optimizations for modern systems, setting up for COSC275!
 - Keeping both the programmer AND microarchitectural perspective in mind
- Logistics, Reminders
 - TA help 7-9PM on Sundays, Tuesdays, Thursdays in C107
 - LP Office hours M 9-10:30AM, Th 2:30-4PM
 - Weekly Exercises due Friday 5PM (last ones!!)
 - Have a wonderful break!!!!
 - **ATTEND FACULTY CANDIDATE TALK TODAY AT 4:30PM in A131, WITH SNACKS @ 4PM in C209**
 - *If you attend at least 3 candidate talks, then send me an email including 1 thing you learned from each talk, I'll give you 5% extra credit on any previous lab report*
 - **POLL FOR ADVANCED TOPIC!! Available on Moodle!!**
- FEEDBACK
 - <https://forms.gle/5Aafcm3iJthX78jx6>