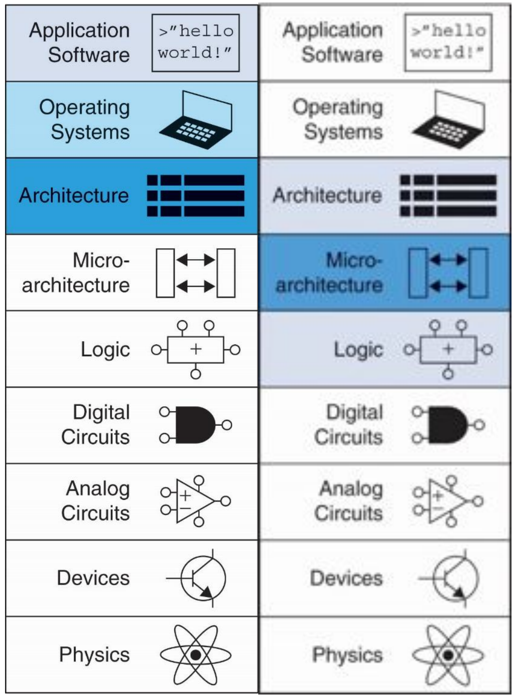
COSC175 (Systems I): Computer Organization & Design

Professor Lillian Pentecost
Fall 2024

Warm-Up November 14

- Where we were
  - Intro to microarchitecture
  - Getting to know our Single-Cycle Processor
- Where we are going
  - Continuing with example microarchitecture for single-cycle RISC-V processor
  - 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
    - A little bit different – take a look ahead of time!
  - Reading and first lab stage are **extremely** correlated! Take time to read and review!



Microarchitecture

- **Multiple implementations** for a single architecture, for example:
  - **Single-cycle:** Each instruction executes in a single cycle
  - **Multicycle:** Each instruction is broken up into series of shorter steps
  - **Pipelined:** Each instruction broken up into series of steps & multiple instructions execute at once

RISC-V Processor: first design

- Consider **subset** of RISC-V instructions:
  - **R-type ALU instructions:**
    - `add`, `sub`, `and`, `or`, `slt`
  - **Memory instructions:**
    - `lw`, `sw`
  - **Branch instructions:**
    - `beq`

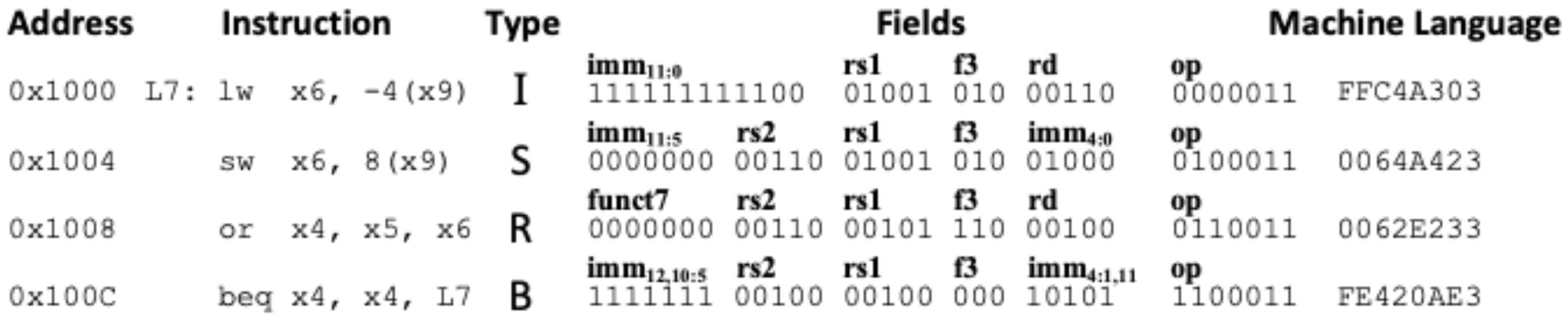
Single-Cycle RISC-V Processor

- Datapath (the components)
- Control (given current instruction, send signals to datapath)
- For each instruction:
  - At clock edge, *fetch* next instruction
  - *Decode* based on instruction type
  - *Execute* the instruction
  - Increment / *update PC*
- Choose long enough clock cycle so that this always completes before the next rising edge

Goals for Today

1. Expand datapath to support sw, beq, etc.
2. Introduce control unit to "decode" op
3. Determine the ***critical path*** of our processor, and the resulting clock cycle

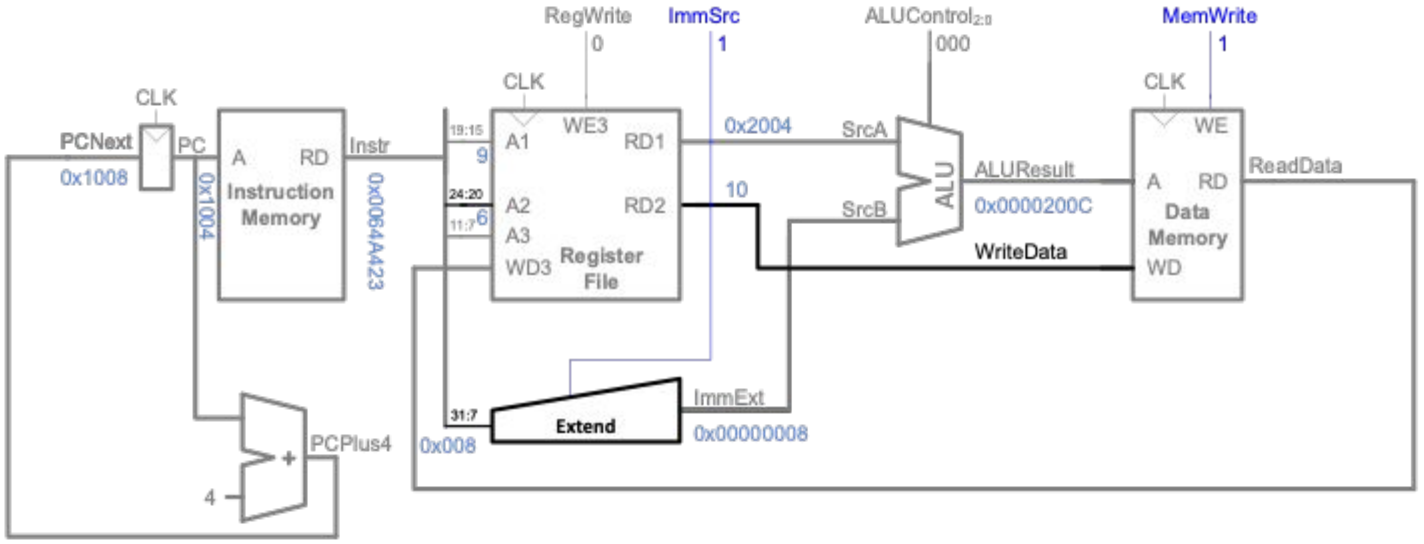
Example Program:



Address	Instruction	Type	Fields						Machine Language
0x1000	L7: lw x6, -4(x9)	I	$\text{imm}_{11:0}$ 111111111100		rs1 01001	f3 010	rd 00110	op 0000011	FFC4A303
0x1004	sw x6, 8(x9)	S	$\text{imm}_{11:5}$ 0000000	rs2 00110	rs1 01001	f3 010	$\text{imm}_{4:0}$ 01000	op 0100011	0064A423
0x1008	or x4, x5, x6	R	funct7 0000000	rs2 00110	rs1 00101	f3 110	rd 00100	op 0110011	0062E233
0x100C	beq x4, x4, L7	B	$\text{imm}_{12,10:5}$ 1111111	rs2 00100	rs1 00100	f3 000	$\text{imm}_{4:1,11}$ 10101	op 1100011	FE420AE3

Single-Cycle Datapath: `sw`

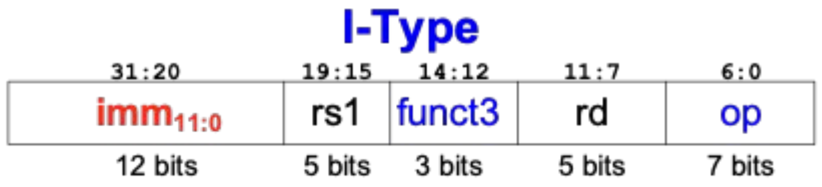
- **Immediate:** now in {instr[31:25], instr[11:7]}
- **Add control signals:** ImmSrc, MemWrite

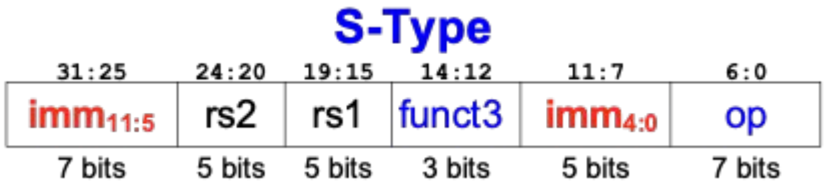


Address	Instruction	Type	Fields						Machine Language
			$imm_{11:5}$	rs2	rs1	f3	$imm_{4:0}$	op	
0x1004	`sw x6, 8(x9)`	S	0000000	00110	01001	010	01000	0100011	0064A423

Single-Cycle Datapath: Immediate

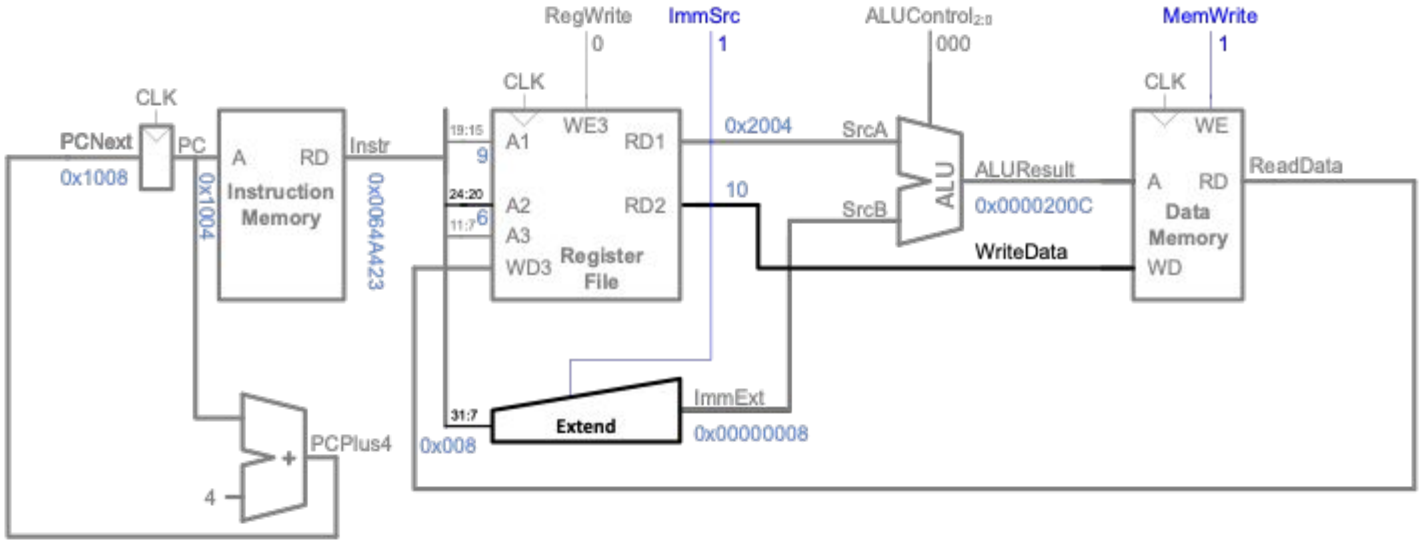
ImmSrc	ImmExt	Instruction Type
0	{{20{instr[31]}}, **instr[31:20]**}	I-Type
1	{{20{instr[31]}}, **instr[31:25], instr[11:7]**}	S-Type





Single-Cycle Datapath: `sw`

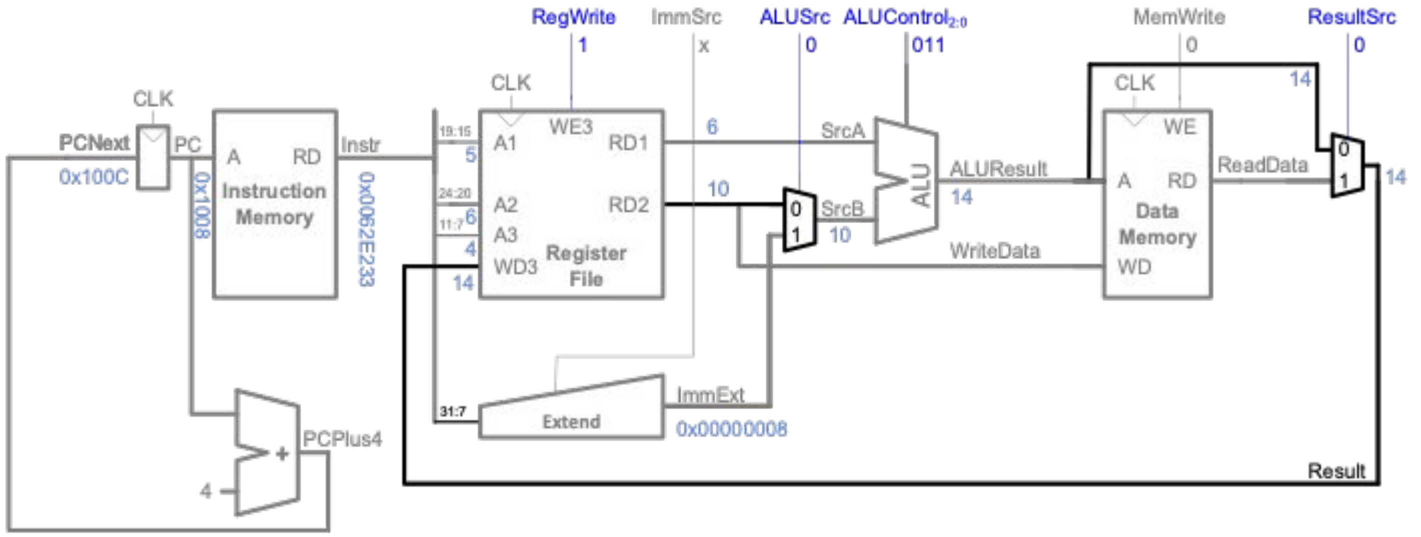
- **Immediate:** now in {instr[31:25], instr[11:7]}
- **Add control signals:** ImmSrc, MemWrite



Address	Instruction	Type	Fields						Machine Language
			$imm_{11:5}$	rs2	rs1	f3	$imm_{4:0}$	op	
0x1004	`sw x6, 8(x9)`	S	0000000	00110	01001	010	01000	0100011	0064A423

Single-Cycle Datapath: R-type

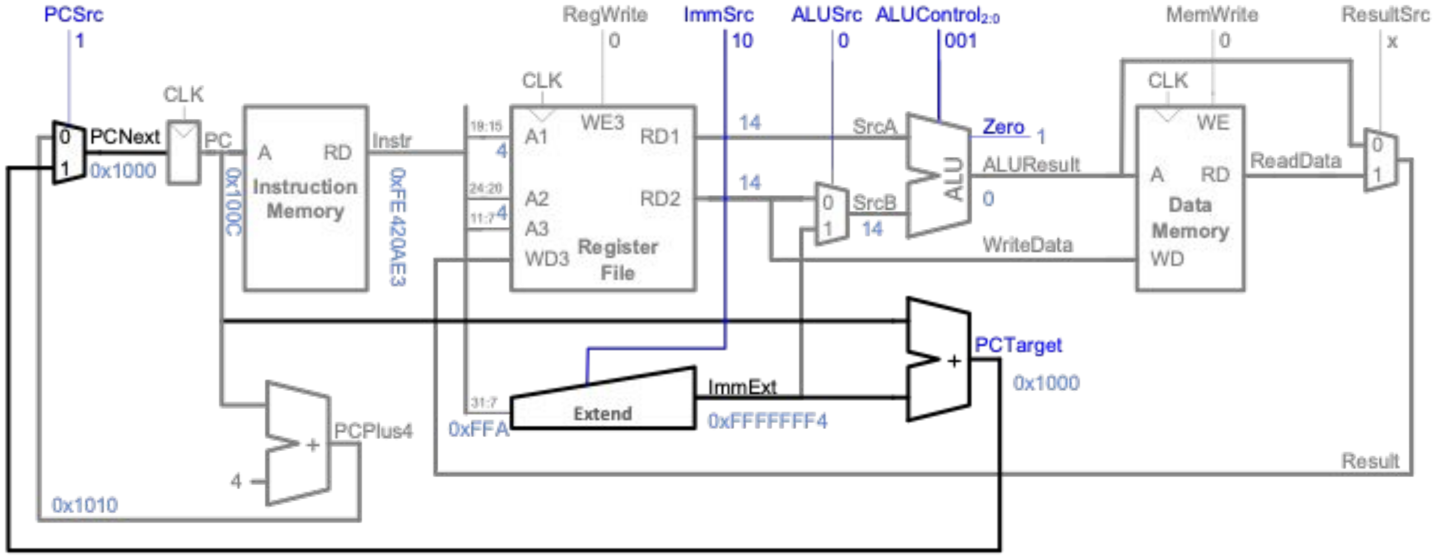
- Read from **rs1** and **rs2** (instead of **imm**)
- Write *ALUResult* to **rd**



Address	Instruction	Type	Fields						Machine Language
			funct7	rs2	rs1	f3	rd	op	
0x1008	or x4, x5, x6	R	0000000	00110	00101	110	00100	0110011	0062E233

Single-Cycle Datapath: `beq`

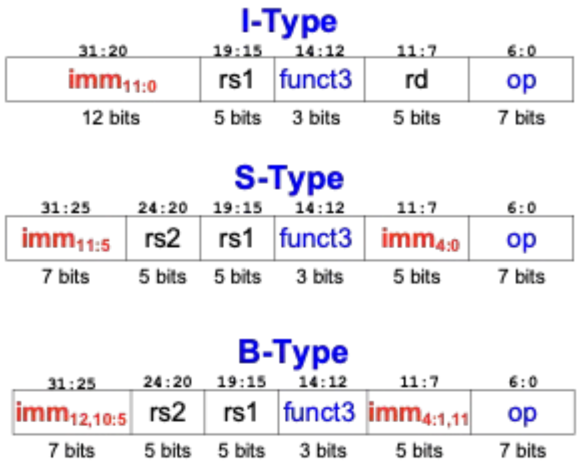
Calculate **target address:** PCTarget = PC + imm



Address	Instruction	Type	Fields						Machine Language
			$imm_{12,10:5}$	rs2	rs1	f3	$imm_{4:1,11}$	op	
0x100C	`beq x4, x4, L7`	B	1111111	00100	00100	000	10101	1100011	FE420AE3

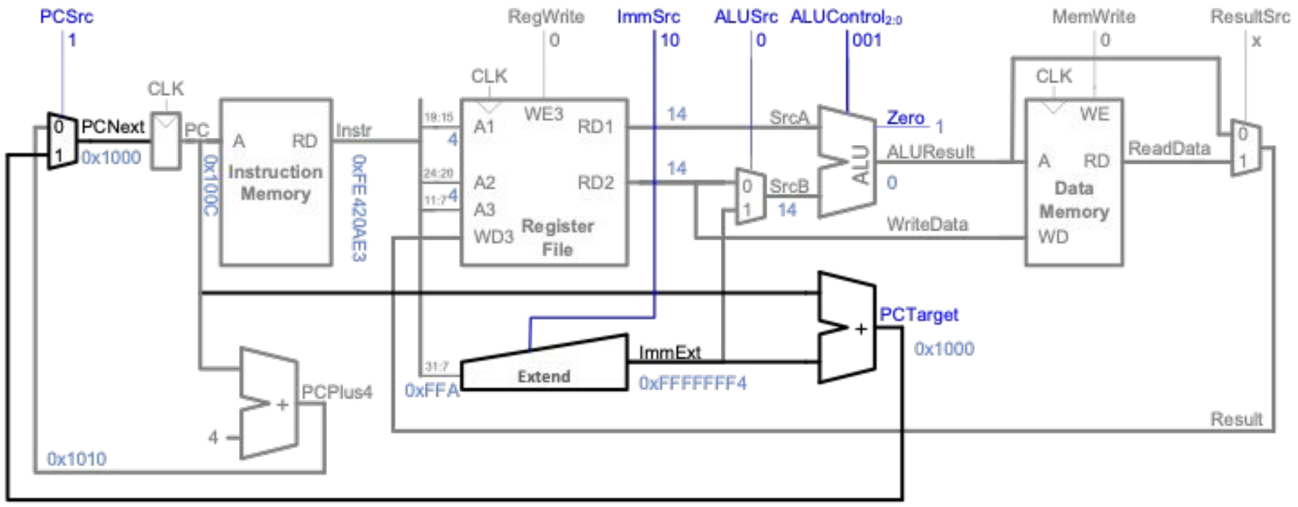
Single-Cycle Datapath: ImmExt

ImmSrc$_{1:0}$	ImmExt	Instruction Type
00	{{20{instr[31]}}, **instr[31:20]**}	I-Type
01	{{20{instr[31]}}, **instr[31:25], instr[11:7]**}	S-Type
10	{{19{instr[31]}}, **instr[31], instr[7], instr[30:25], instr[11:8], 1'b0**}	B-Type



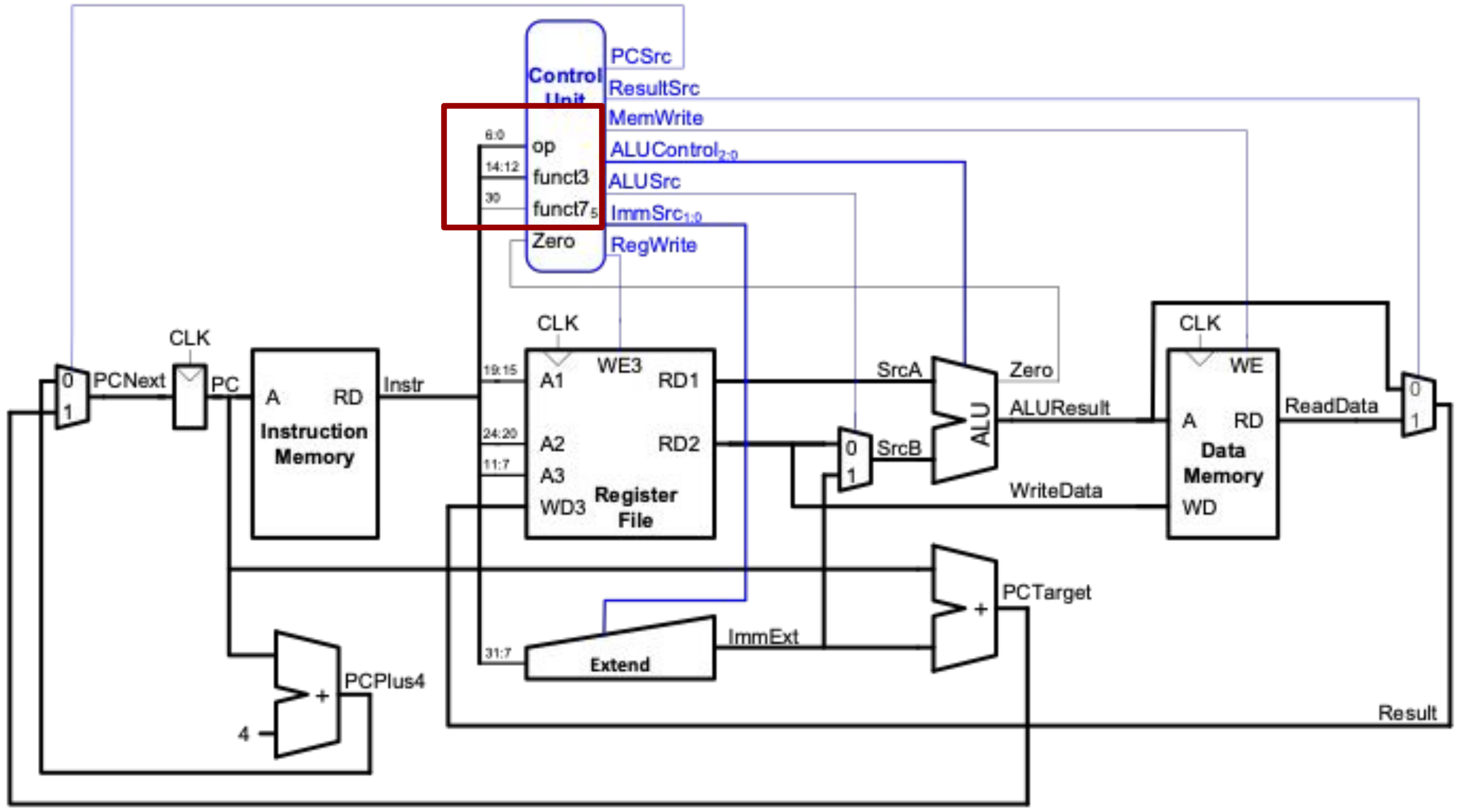
Single-Cycle RISC-V Processor

With our updated datapath, where are all these blue signals coming from? How determined?



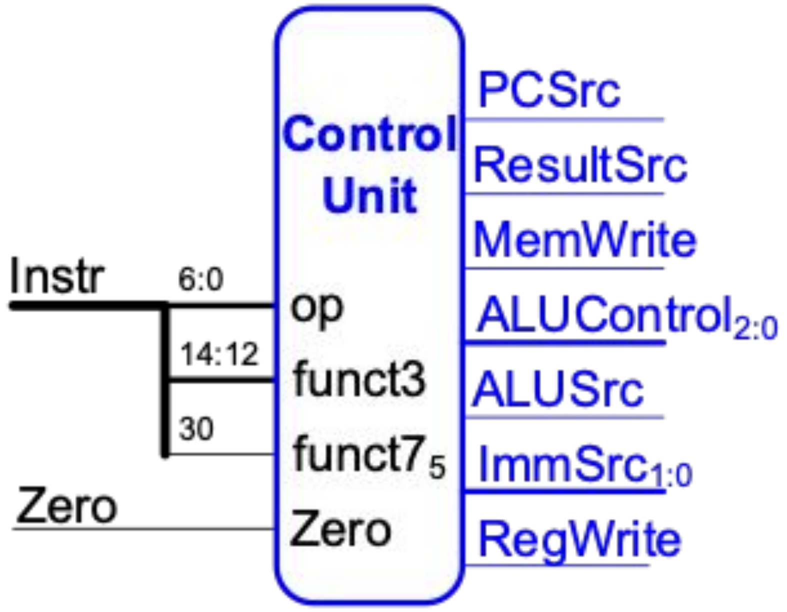
Address	Instruction	Type	Fields						Machine Language
			$imm_{12,10:5}$	rs2	rs1	f3	$imm_{4:1,11}$	op	
0x100C	beq x4, x4, L7	B	1111111	00100	00100	000	10101	1100011	FE420AE3

Single-Cycle RISC-V Processor

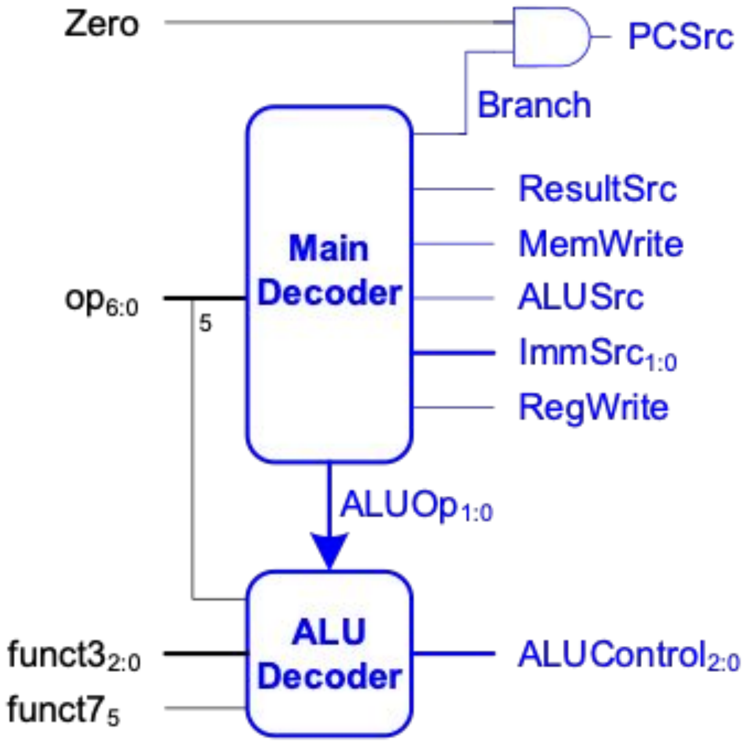


Single-Cycle Control

High-Level View

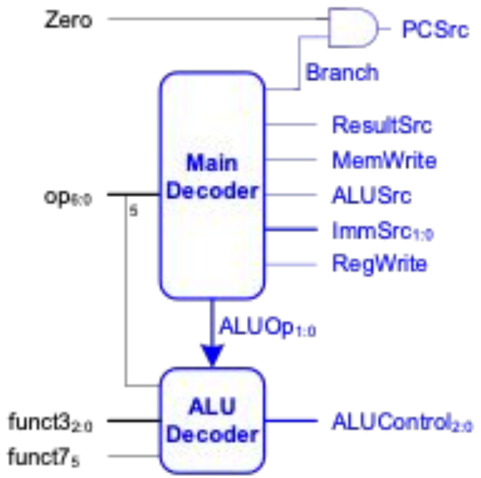


Low-Level View



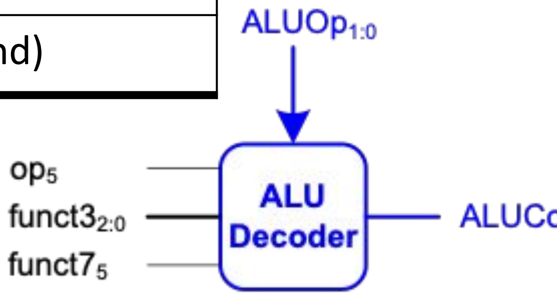
Single-Cycle Control: Main Decoder

op	Instr.	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
3	`lw`	1	00	1	0	1	0	00
35	`sw`	0	01	1	1	X	0	00
51	R-type	1	XX	0	0	0	0	10
99	`beq`	0	10	0	0	X	1	01



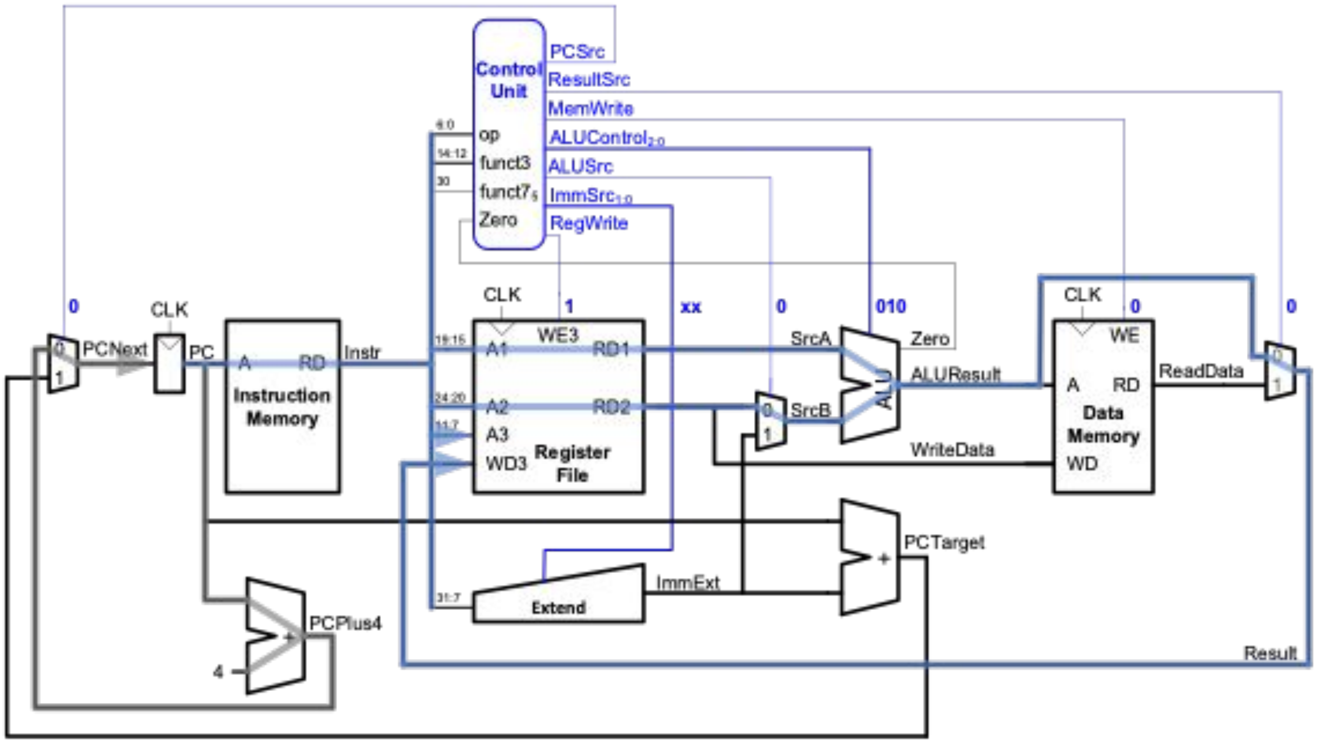
Single-Cycle Control: ALU Decoder

ALUOp	funct3	op_5 , $funct7_5$	Instruction	$ALUControl_{2:0}$
00	x	x	`lw, sw`	000 (add)
01	x	x	`beq`	001 (subtract)
10	000	00, 01, 10	`add`	000 (add)
	000	11	`sub`	001 (subtract)
	010	x	`slt`	101 (set less than)
	110	x	`or`	011 (or)
	111	x	`Slt`	010 (and)



Example: `and`

op	Instruct	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
51	**R-type**	1	XX	0	0	0	0	10



Processor Performance

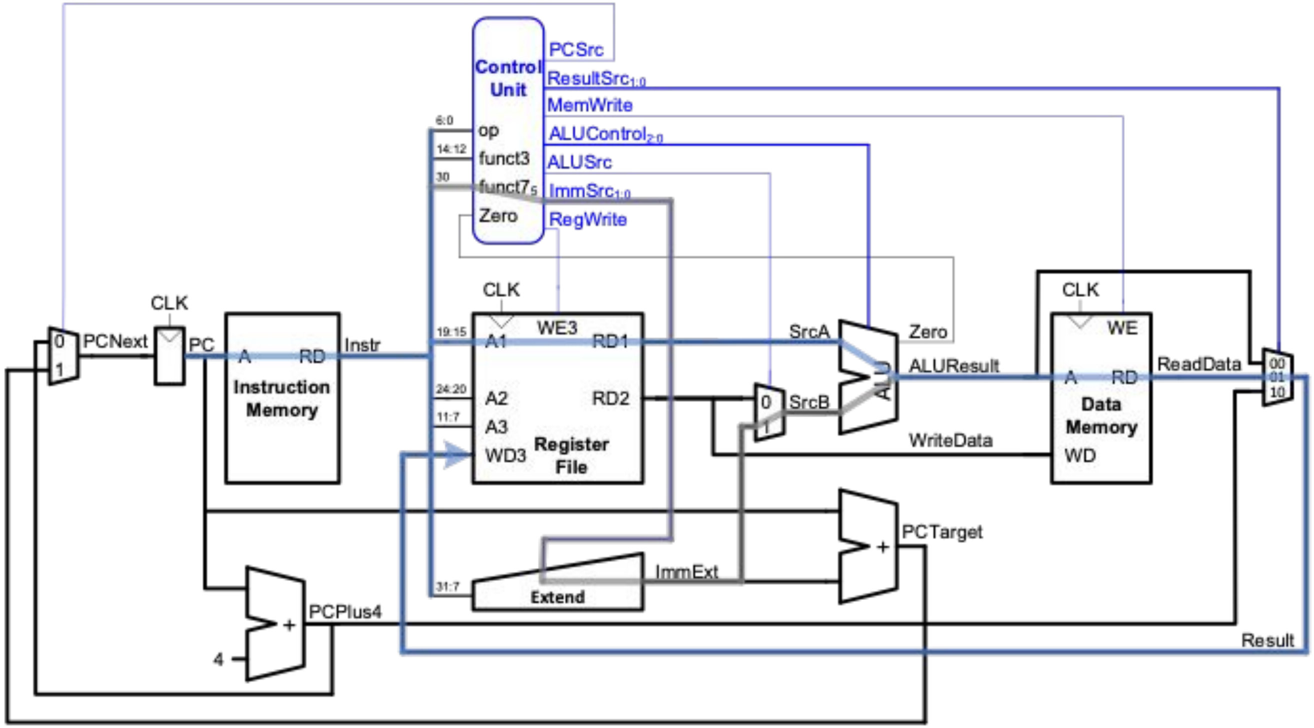
Okay, but *how long does this take*?

Program Execution Time

= (#instructions)(cycles/instruction)(seconds/cycle)

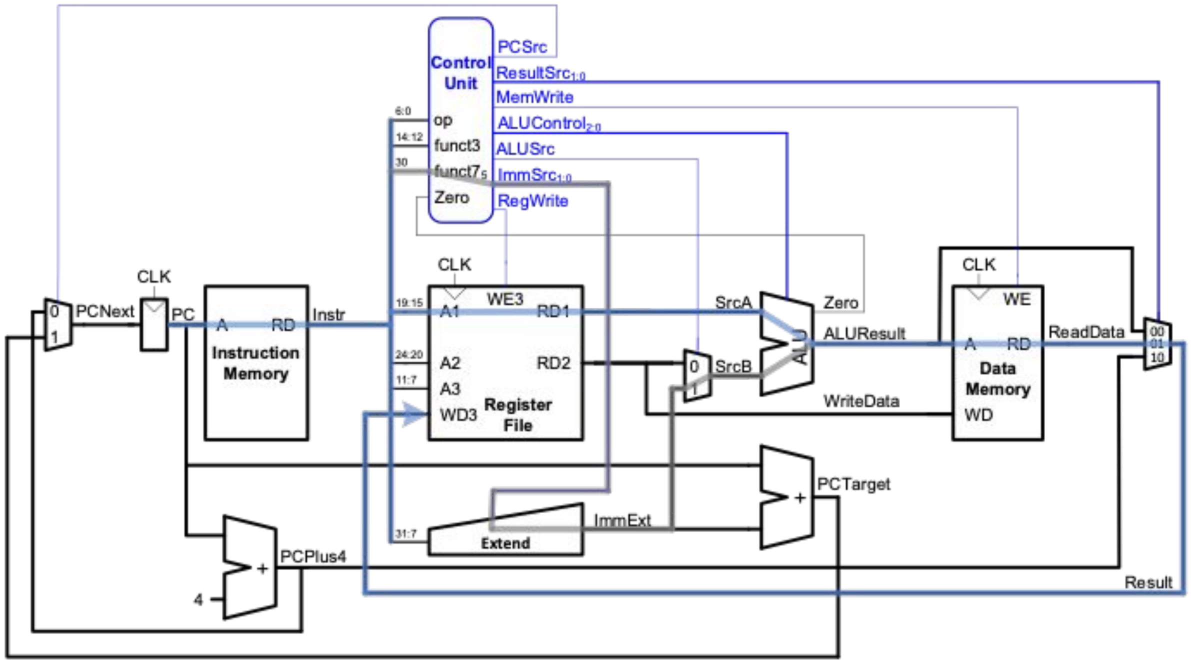
= # instructions x CPI x T_C

Single-Cycle Processor Performance



T_C limited by critical path (`lw`)

Single-Cycle Processor Performance



T_c limited by critical path (lw)

$T_{c\_single} = t_{pcq\_PC} + t_{\mathrm{mem}} + \max[t_{RFread}, t_{dec} + t_{ext} + t_{\mathrm{mux}}] + t_{\mathrm{ALU}} + t_{\mathrm{mem}} + t_{\mathrm{mux}} + t_{RF\mathrm{setup}}$

Single-Cycle Processor Performance

- **Single-cycle critical path:**

  $T_{c\_single} = t_{pcq\_PC} + t_{\mathrm{mem}} + \max[t_{RFread}, t_{dec} + t_{ext} + t_{\mathrm{mux}}] + t_{\mathrm{ALU}} + t_{\mathrm{mem}} + t_{\mathrm{mux}} + t_{RF\mathrm{setup}}$

- **Typically, limiting paths are:**
  - **memory, ALU, register file**
  - *So*, $T_{c\_single} = t_{pcq\_PC} + t_{\mathrm{mem}} + t_{RF\mathrm{read}} + t_{\mathrm{ALU}} + t_{\mathrm{mem}} + t_{\mathrm{mux}} + t_{RF\mathrm{setup}}$

    $= t_{pcq\_PC} + 2t_{\mathrm{mem}} + t_{RF\mathrm{read}} + t_{\mathrm{ALU}} + t_{\mathrm{mux}} + t_{RF\mathrm{setup}}$

Check-In from last time: what is my clock cycle?

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

$$T_{c\_single} = t_{pcq\_PC} + 2t_{\mathrm{mem}} + t_{RF\mathrm{read}} + t_{\mathrm{ALU}} + t_{\mathrm{mux}} + t_{RF\mathrm{setup}}$$

$$= (40 + 2*200 + 100 + 120 + 30 + 60)\ \mathrm{ps} = \mathbf{750\ ps}$$

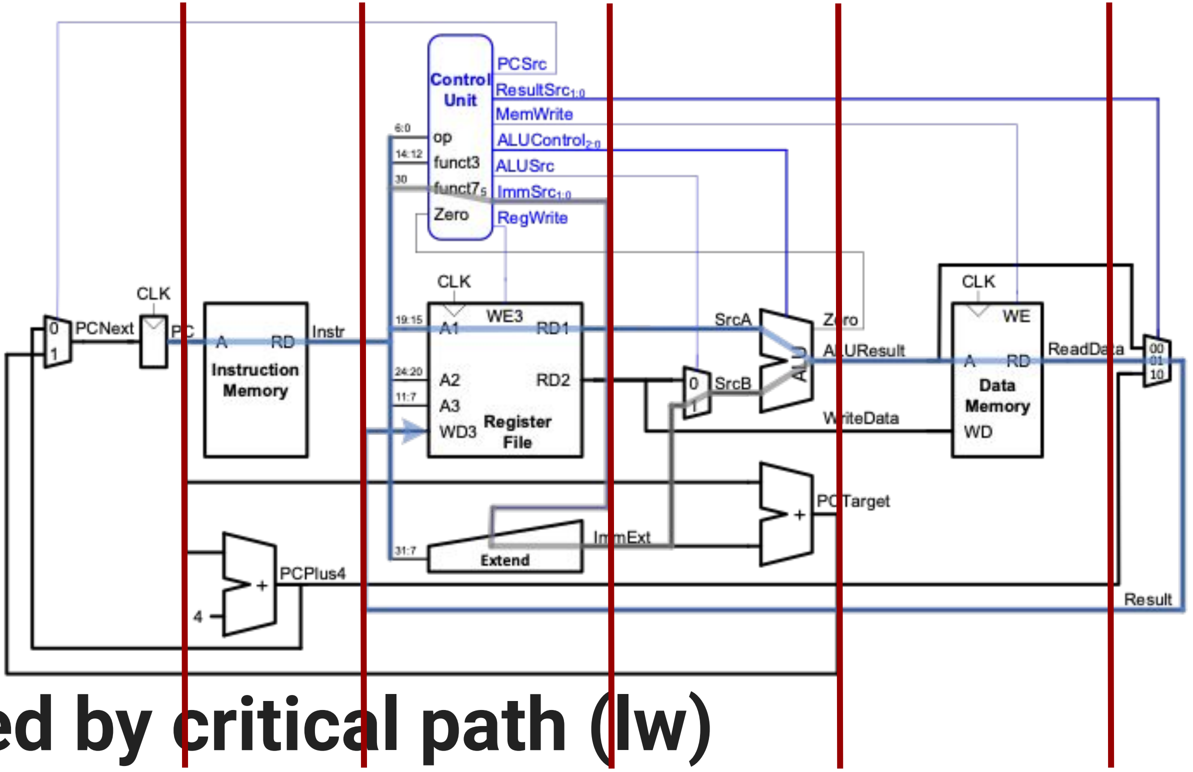
Today’s Check-In: Calculate Program Execution

With a partner, on a notecard:

1. For this single-cycle processor design, how long does it take to execute a program with 100 billion instructions?

Show your work, and jot down any additional questions you have about our single-cycle design so far.

Preview for next time: smaller steps



T_c limited by critical path (lw)

$T_{c\_single} = t_{pcq\_PC} + t_{\text{mem}} + \max[t_{RFread}, t_{dec} + t_{ext} + t_{\text{mux}}] + t_{\text{ALU}} + t_{\text{mem}} + t_{\text{mux}} + t_{RF\text{setup}}$

Wrap-Up November 14

SCAN ME

- Coming up next!
  - Exploring & testing your first processor design!
  - Keeping both the programmer AND microarchitectural perspective in mind
    - *As you complete the weekly exercises, think about the way each high-level operation will be broken down into instructions, then interact with your datapath!*
- 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
  - Pay attention to deliverables, stay in sync with material AND begin your review of first half of the semester
- FEEDBACK
  - https://forms.gle/5Aafcm3iJthX78jx6

Today’s Check-In: Calculate Program Execution

With a partner, on a notecard:

1. For this single-cycle processor design, how long does it take to execute a program with 100 billion instructions?

Execution Time = # instructions x CPI x T_C

$= (100 \times 10^{9})(1)(750 \times 10^{-12}\,\mathrm{s})$

= **75 seconds**