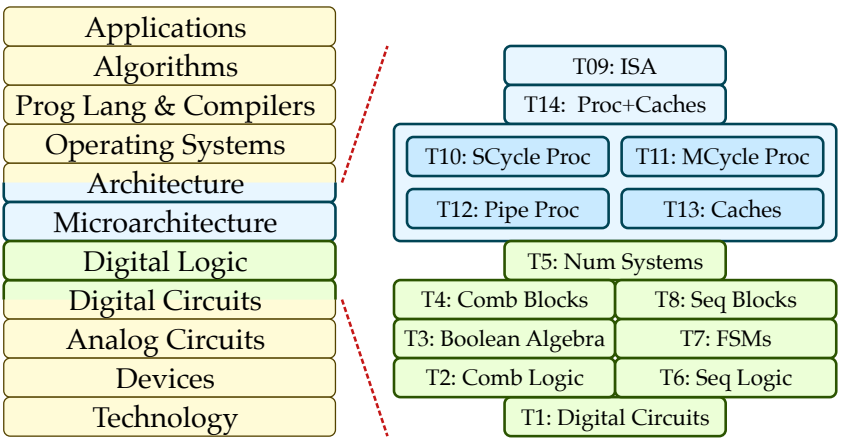
ECE 2300 Digital Logic and Computer Organization Fall 2025

Topic 10: Single-Cycle Processors

School of Electrical and Computer Engineering
Cornell University

revision: 2025-11-04-10-40

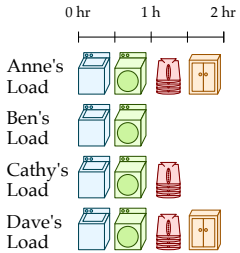


1. High-Level Idea for Single-Cycle Processors

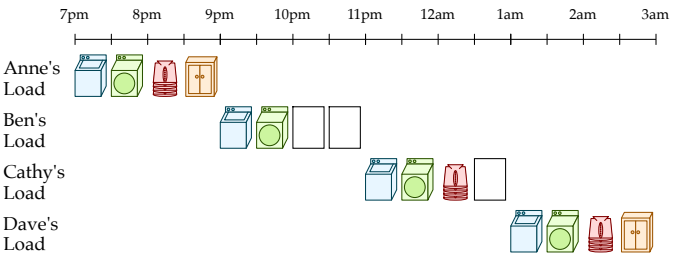
Transaction Steps

Washing (30 min)
Drying (30 min)
Folding (30 min)
Storing (30 min)

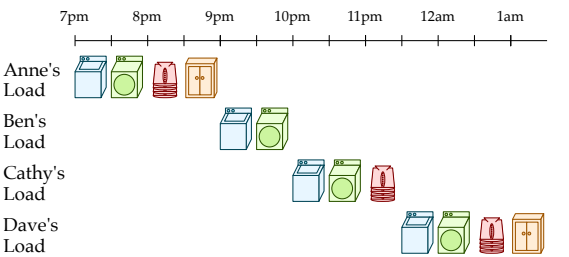
Four Types of Transactions



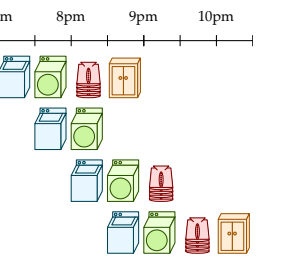
Fixed Time Slot Laundry (Single-Cycle Processors)



Variable Time Slot Laundry (Multi-Cycle Processors)



Pipelined Laundry



1.1. Transactions and Steps

- We can think of each instruction as a transaction
- Executing a transaction involves a sequence of steps

ADDI
`addi rd, rs1, imm`
R[rd] ← R[rs1] + sext(imm)
PC ← PC + 4

ADD
`add rd, rs1, rs2`
R[rd] ← R[rs1] + R[rs2]
PC ← PC + 4

MUL
`mul rd, rs1, rs2`
R[rd] ← R[rs1] × R[rs2]
PC ← PC + 4

LW
`lw rd, imm(rs1)`
R[rd] ← M[R[rs1] + sext(imm)]
PC ← PC + 4

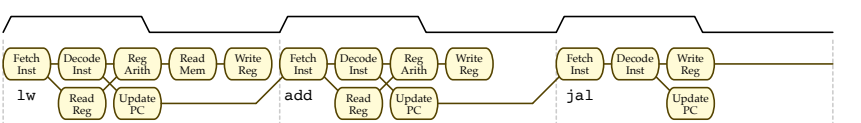
SW
`sw rs2, imm(rs1)`
M[R[rs1] + sext(imm)] ← R[rs2]
PC ← PC + 4

JAL
`jal rd, addr`
R[rd] ← PC + 4
PC ← addr

JR
`jr rs1`
PC ← R[rs1]

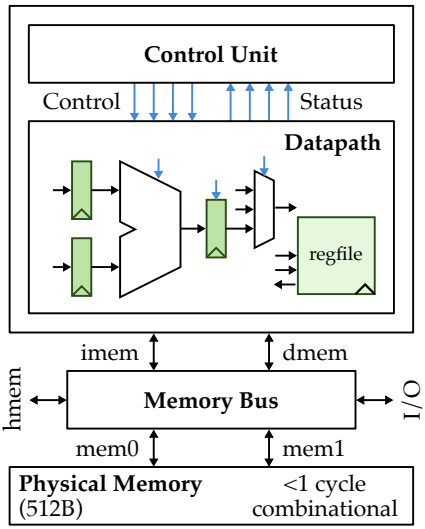
BNE
`bne rs1, rs2, addr`
if (R[rs1] != R[rs2]) PC ← addr
else PC ← PC + 4

	addi	add	mul	lw	sw	jal	jr	bne
Fetch Instruction								
Decode Instruction								
Read Registers								
Register Arithmetic								
Read Memory								
Write Memory								
Write Registers								
Update PC								



1.2. Technology and System Constraints

- Assume technology where logic is not too expensive; do not need to overly minimize the number of registers and combinational logic
- Assume multi-ported register file with a reasonable number of ports is feasible
- Assume a dual-ported combinational physical memory with 512B of capacity and wait signal
- Assume a memory bus interconnects instruction, data, host memory interfaces with physical memory and external input/output (I/O)



1.3. First-Order Performance Equation

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Avg Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

- Instructions / program depends on source code, compiler, ISA
- Avg cycles / instruction (CPI) depends on ISA, microarchitecture
- Time / cycle depends upon microarchitecture and implementation

Microarchitecture	**CPI**	**Cycle Time**
Single-Cycle Processor	1	long
Multi-Cycle Processor	>1	short
Pipelined Processor	≈1	short

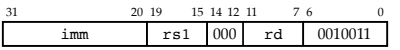
2. Single-Cycle Processor Microarchitecture

ADDI

`addi rd, rs1, imm`

R[`rd`] ← R[`rs1`] + sext(`imm`)

PC ← PC + 4

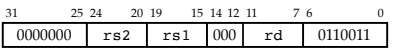


ADD

`add rd, rs1, rs2`

R[`rd`] ← R[`rs1`] + R[`rs2`]

PC ← PC + 4

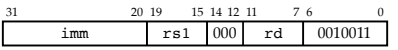


Implementing ADDI and ADD Instructions

ADDI

`addi rd, rs1, imm`

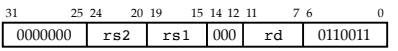
R[rd] ← R[rs1] + sext(imm)
PC ← PC + 4

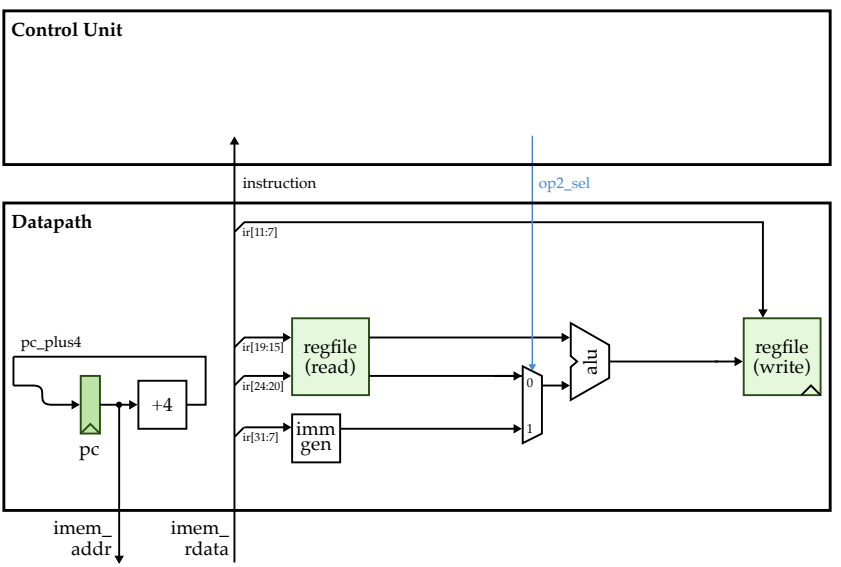


ADD

`add rd, rs1, rs2`

R[rd] ← R[rs1] + R[rs2]
PC ← PC + 4



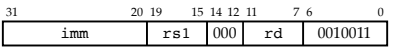


Adding MUL Instruction

ADDI

`addi rd, rs1, imm`

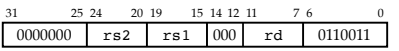
R[rd] ← R[rs1] + sext(imm)
PC ← PC + 4



ADD

`add rd, rs1, rs2`

R[rd] ← R[rs1] + R[rs2]
PC ← PC + 4



MUL

`mul rd, rs1, rs2`

R[rd] ← R[rs1] × R[rs2]
PC ← PC + 4