

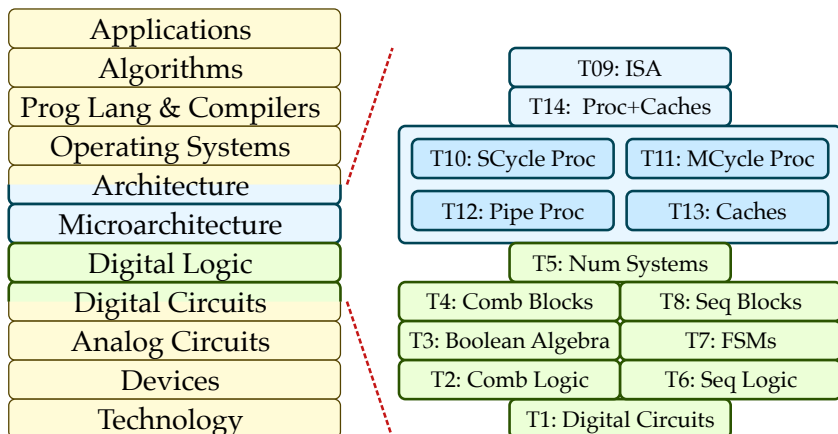
ECE 2300 Digital Logic and Computer Organization Fall 2025

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	3
1.1.	Transactions and Steps	4
1.2.	Technology and System Constraints	5
1.3.	First-Order Performance Equation	6
2	Single-Cycle Processor Microarchitecture	7
3	Analyzing Performance	16



Copyright © 2025 Christopher Batten. All rights reserved. This handout was prepared by Prof. Christopher Batten at Cornell University for ECE 2300 / ENGRD 2300 Digital Logic and Computer Organization. Download and use of this handout is permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial or non-commercial means requires written permission.

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

0 hr 1 h 2 hr Transaction Latency

Anne's Load



2.0 hr

Anne requires all four steps

Ben's Load



1.0 hr

Ben is messy, leaves unfolded clothes in his laundry basket

Cathy's Load



1.5 hr

Cathy does not have a bureau, leaves folded clothes in basket

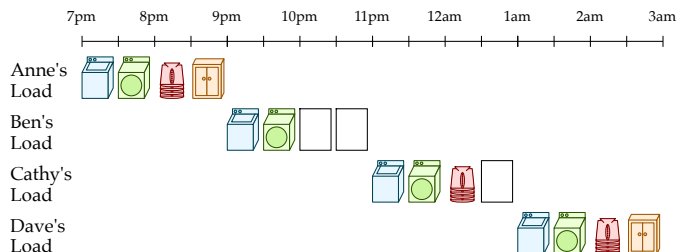
Dave's Load



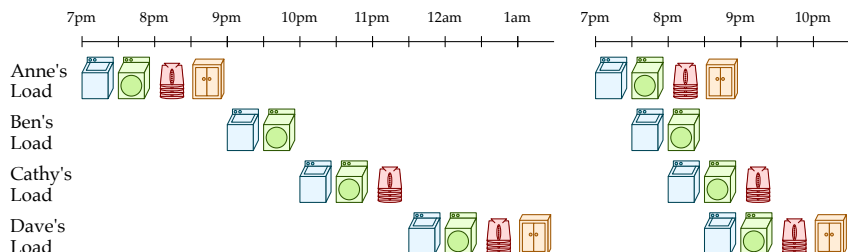
2.0 hr

Dave requires all four steps

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
 $R[rd] \leftarrow R[rs1] + \text{sext}(imm)$ 
 $PC \leftarrow PC + 4$ 
```

ADD

```
add rd, rs1, rs2
 $R[rd] \leftarrow R[rs1] + R[rs2]$ 
 $PC \leftarrow PC + 4$ 
```

MUL

```
mul rd, rs1, rs2
 $R[rd] \leftarrow R[rs1] \times R[rs2]$ 
 $PC \leftarrow PC + 4$ 
```

LW

```
lw rd, imm(rs1)
 $R[rd] \leftarrow M[R[rs1] + \text{sext}(imm)]$ 
 $PC \leftarrow PC + 4$ 
```

SW

```
sw rs2, imm(rs1)
 $M[R[rs1] + \text{sext}(imm)] \leftarrow R[rs2]$ 
 $PC \leftarrow PC + 4$ 
```

JAL

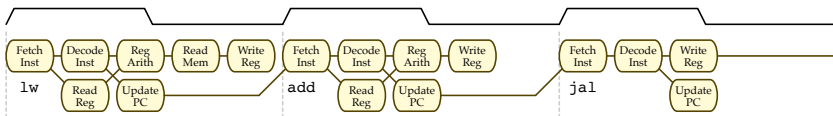
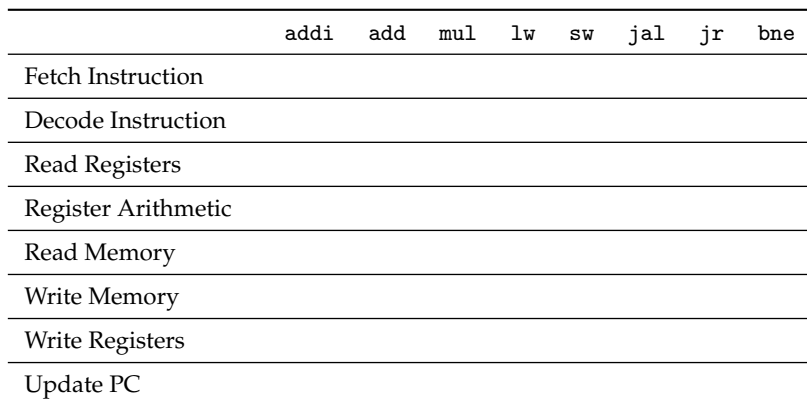
```
jal rd, addr
 $R[rd] \leftarrow PC + 4$ 
 $PC \leftarrow \text{addr}$ 
```

JR

```
jr rs1
 $PC \leftarrow R[rs1]$ 
```

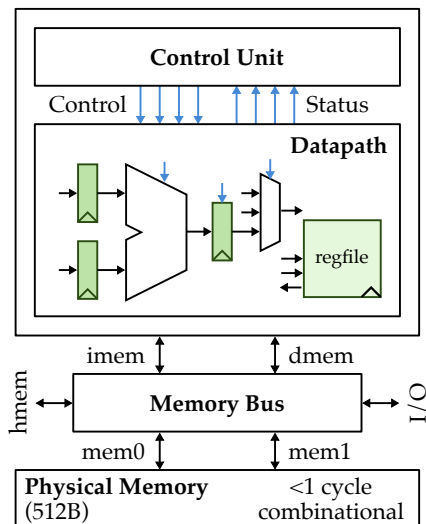
BNE

```
bne rs1, rs2, addr
if ( $R[rs1] \neq R[rs2]$ )  $PC \leftarrow \text{addr}$ 
else  $PC \leftarrow PC + 4$ 
```



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
```

$$R[rd] \leftarrow R[rs1] + \text{sext}(imm)$$

$$PC \leftarrow PC + 4$$

31	20	19	15	14	12	11	7	6	0
imm						rs1	000	rd	0010011



ADD

```
add rd, rs1, rs2
```

$$R[rd] \leftarrow R[rs1] + R[rs2]$$

$$PC \leftarrow PC + 4$$

31	25	24	20	19	15	14	12	11	7	6	0		
0000000				rs2		rs1		000		rd		0110011	



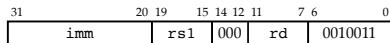
Implementing ADDI and ADD Instructions

ADDI

addi rd, rs1, imm

$R[rd] \leftarrow R[rs1] + \text{sext}(imm)$

$PC \leftarrow PC + 4$

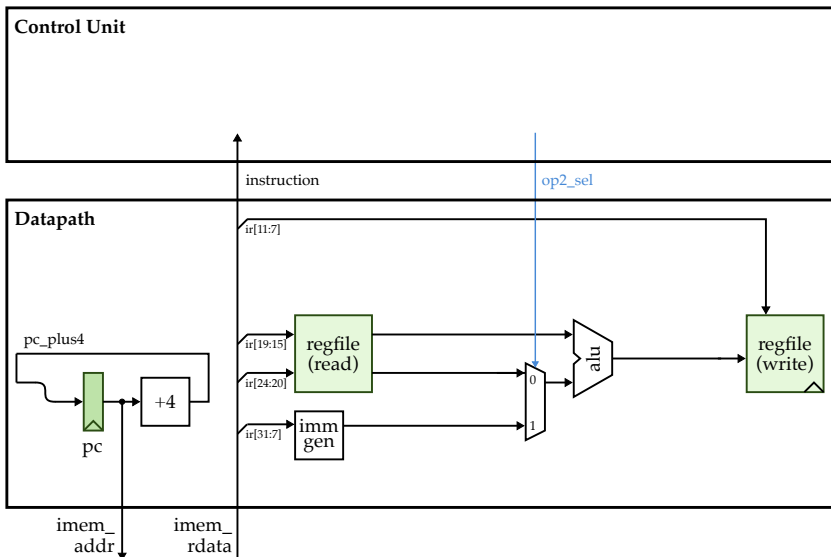
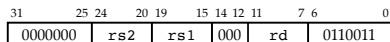


ADD

add rd, rs1, rs2

$R[rd] \leftarrow R[rs1] + R[rs2]$

$PC \leftarrow PC + 4$



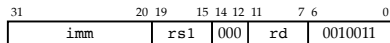
Adding MUL Instruction

ADDI

`addi rd, rs1, imm`

$R[rd] \leftarrow R[rs1] + \text{sext}(imm)$

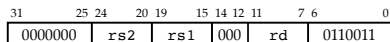
$PC \leftarrow PC + 4$

**ADD**

`add rd, rs1, rs2`

$R[rd] \leftarrow R[rs1] + R[rs2]$

$PC \leftarrow PC + 4$

**MUL**

`mul rd, rs1, rs2`

$R[rd] \leftarrow R[rs1] \times R[rs2]$

$PC \leftarrow PC + 4$

