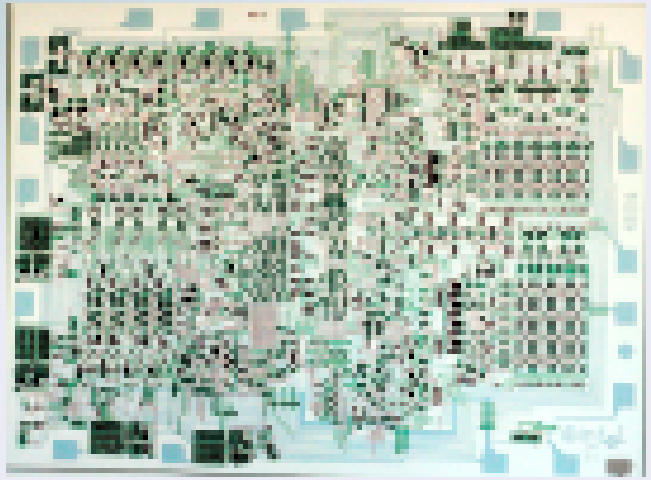


Lecture 4b: Introduction to Computers

CSCI11: **Computer Architecture and Organization**

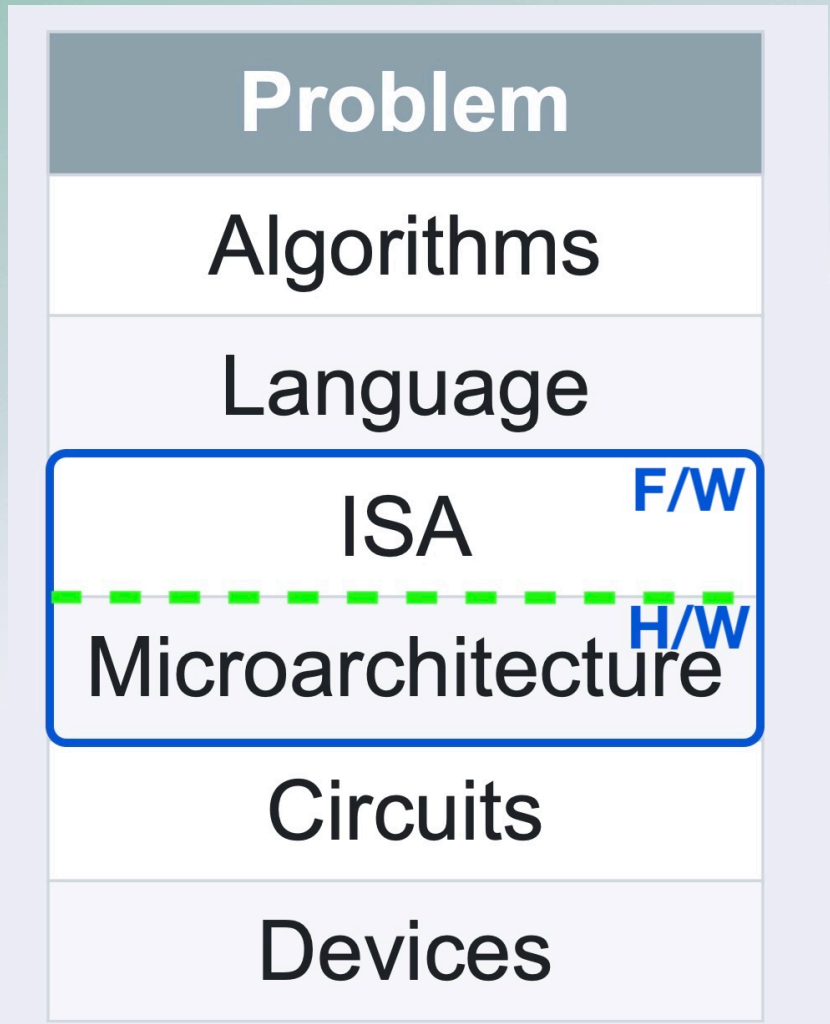
Chapter 4.1-4.3 in Patt and Patel: Introduction to Computing Systems...

Next Few Weeks

- Today Simple CPU - a digital design-based processor
- Today - Expansion of the Traffic Signal
- Today - Introduction to Von Neumann and LC-3
- Monday - 2/23 Test Review/Mock Test
- Wednesday - 2/25 Test 1
- Monday - 3/2 Begin LC3 Assembly Language

What Do We Know?

- Data representation (binary, unsigned, 2's complement,...)
- Transistors (p-type, n-type, CMOS)
- Gates (complementary logic)
- Combinational logic circuits (memory (latches, flip-flops, ...)
- Sequential logic circuits (finite state machines)
- Simple "processors" (a programmable traffic sign)



MicroArchitecture/ISA Terms

- **Program Counter (PC)** - tracks the instructions to be executed
- **Opcode** - machine instruction to be executed
- **Operand** - data to which the *Opcode* will execute with

Opcode and Operand will be expressed in both:

- Assembly Language (English mnemonic)
- Machine Language (Hex or Binary)

LC-3 Example

ADD

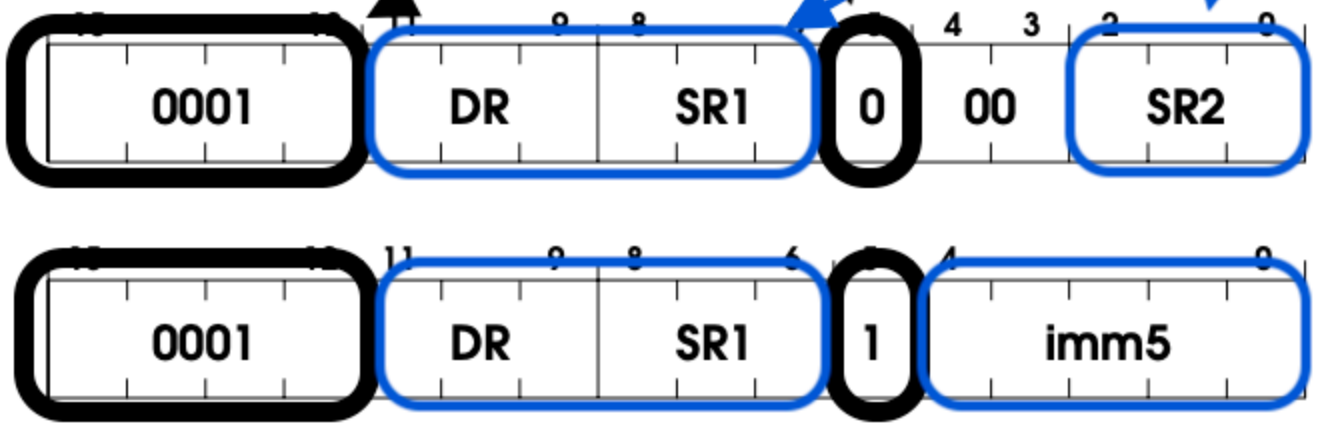
Addition

Assembler Formats

```
ADD   DR, SR1, SR2
ADD   DR, SR1, imm5
```

Opcode Operand

Encodings

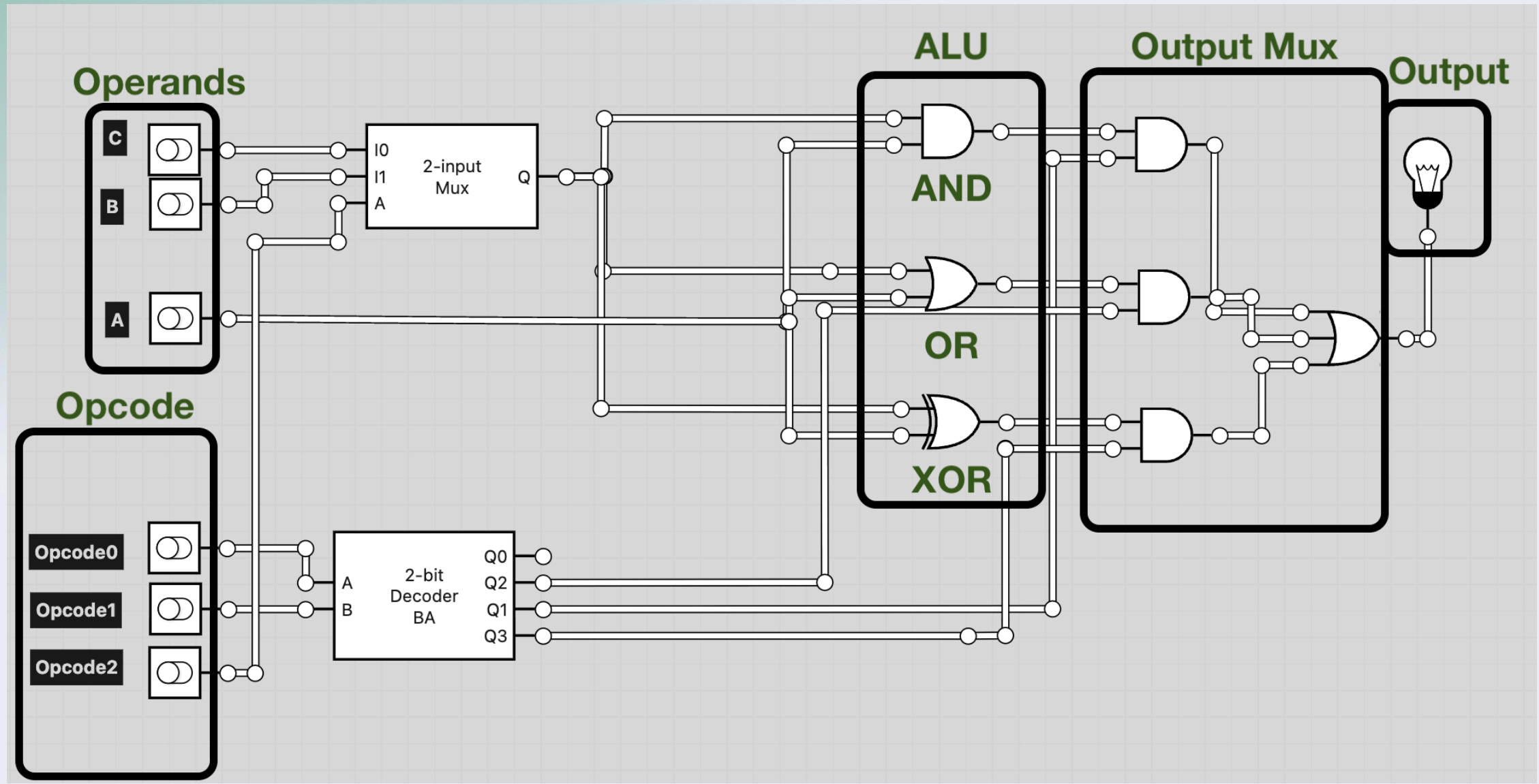


Discussion of a Simple CPU

- An introduction of a simple CPU
- Uses known digital design elements
- Basis of CPU is the same as the ones soon to be introduced(universality)
- Stepping stone to LC-3 Introduction

ISA of Simple CPU

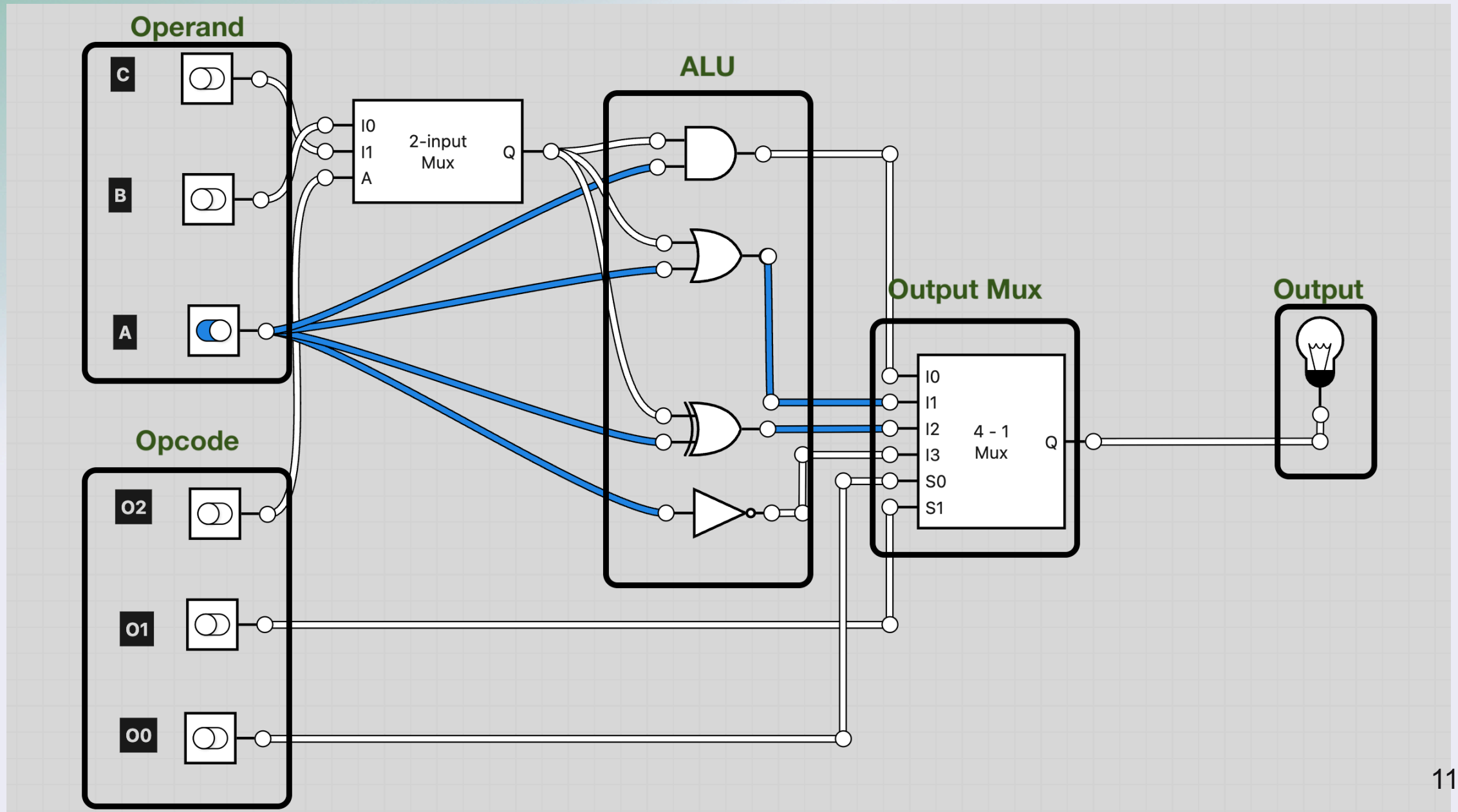
Opcode	Operand	Assembly Language
000		NOP - no operation
001	AXC	AND A, C
010	AXC	OR A, C
011	AXC	XOR A, C
101	ABX	AND A, B
110	ABX	OR A, B
111	ABX	XOR A, B


Operands
C
B
A
I0
I1
A
2-input
Mux
Q
ALU
AND
OR
XOR
Output Mux
Output
Opcode
Opcode0
Opcode1
Opcode2
A
B
2-bit
Decoder
BA
Q0
Q2
Q1
Q3

Demo of Simple CPU version 1

ISA of Simple CPU version 2

Opcode	Operand	Assembly Language
000	A	NOT A
001	ABX	AND A, B
010	ABX	OR A, B
011	ABX	XOR A, B
101	AXC	AND A, C
110	AXC	OR A, C
111	AXC	XOR A, C


Operand
C
B
A
I0
I1
A
2-input
Mux
Q
ALU
Output Mux
I0
I1
I2
I3
S0
S1
4 - 1
Mux
Q
Output
Opcode
O2
O1
O0

Demo of Simple CPU version 2

One More FSM/Traffic Light

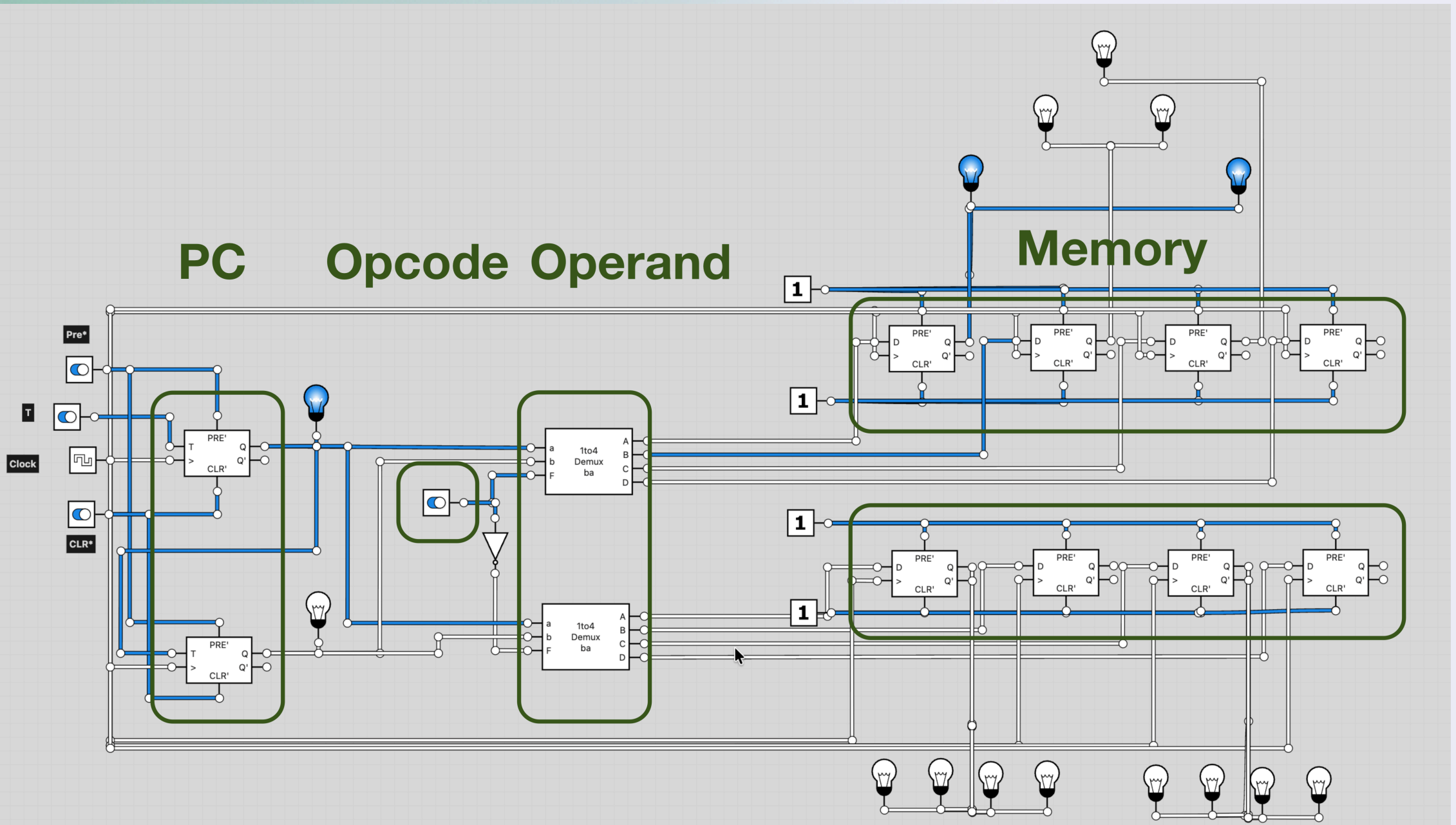
Is there a better way to do it?

1. Begin thinking in sequences, like a computer
2. Create a 2-bit counter with T Flip Flops, as a program counter
3. Use Edge-Triggered D Flip Flops as program memory
4. Use a demux for LED selection
5. Add a switch to provide two instructions, 0 and 1

Logic.ly Demonstration of Dual Traffic Signal

1. Arrow to move traffic in a specific direction
2. Dash to stop traffic from entering

Traffic Signal is a Two Instruction Computer


PC
Opcode
Operand
Memory
Pre*
T
Clock
CLR*
PRE'
CLR'
1to4
Demux
ba
1

What's next?

- Apply all this to traditional computing
- Von Neumann Model - theoretical model of computer
- LC-3 - instructional computer, your new BFF

Today: Review Von Neumann Machine and the LC-3

Courtesy of ETH Zurich

2025 Course