CprE 3710x

Implement and Test a Curriculum Around the i281e CPU

sdmay25-31

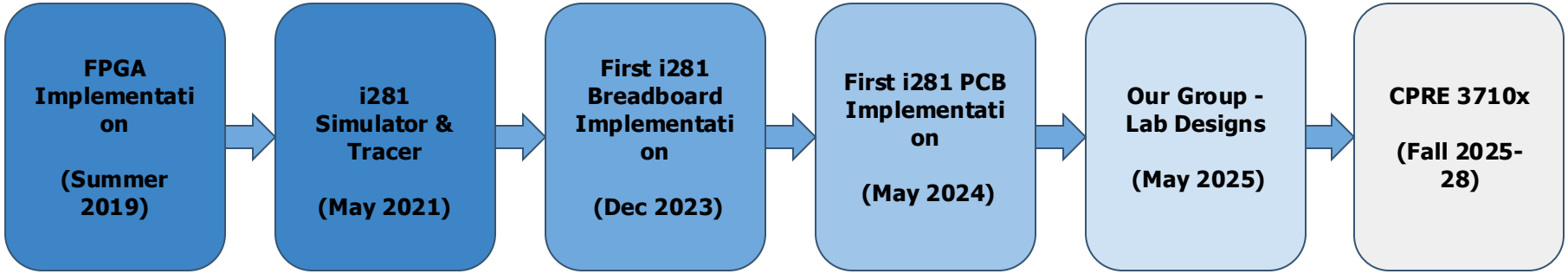
Ariana Dirksen, Gigi Harrabi, Tessa Morgan, Ethan Uhrich
Professor Alexander Stoytchev

Team Composition

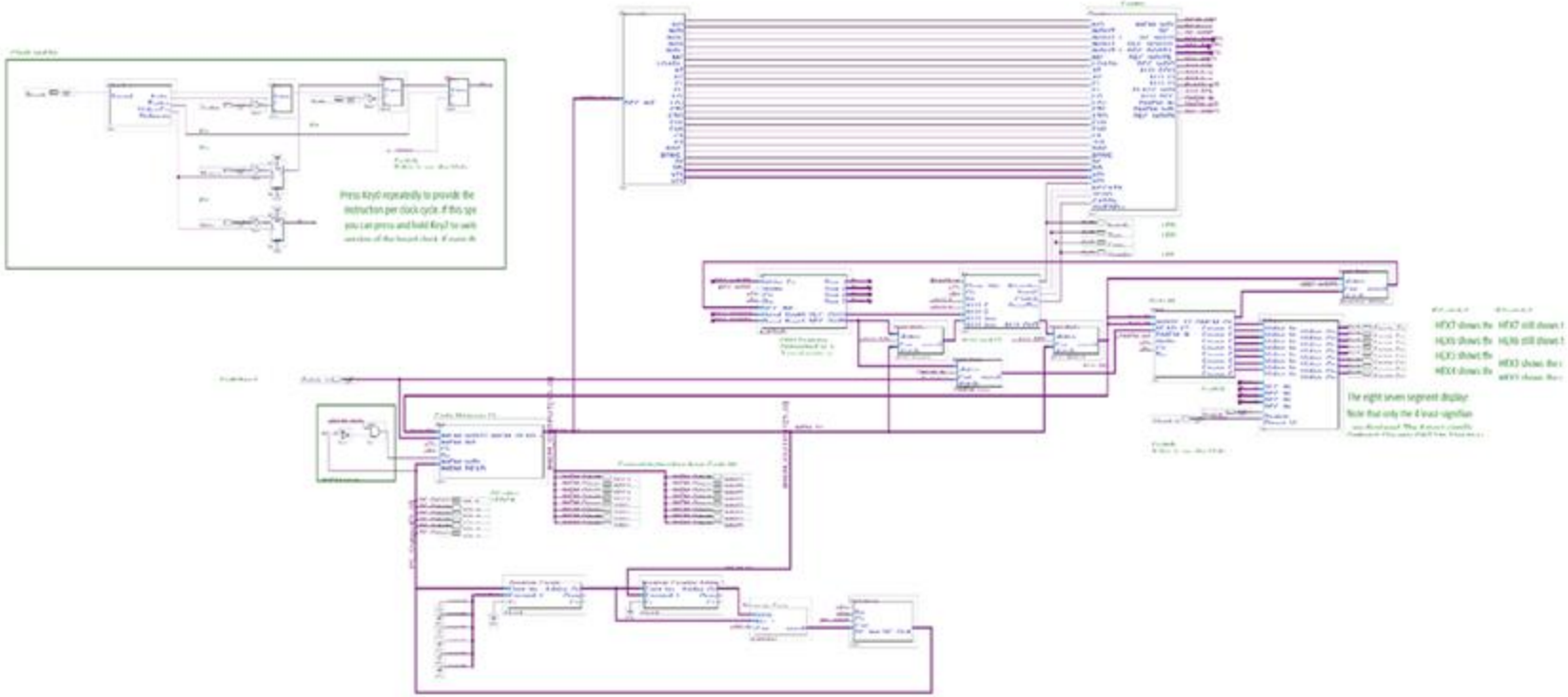
- Ethan Uhrich
  - Team Lead
  - Treasurer
  - Computer Engineer
- Ariana Dirksen
  - Editor
  - Note Taker
  - Computer Engineer
- Tessa Morgan
  - Task Manager
  - Webmaster
  - Computer Engineer
- Gigi Harrabi
  - Client Interaction Lead
  - Outreach Coordinator
  - Computer Engineer

Background

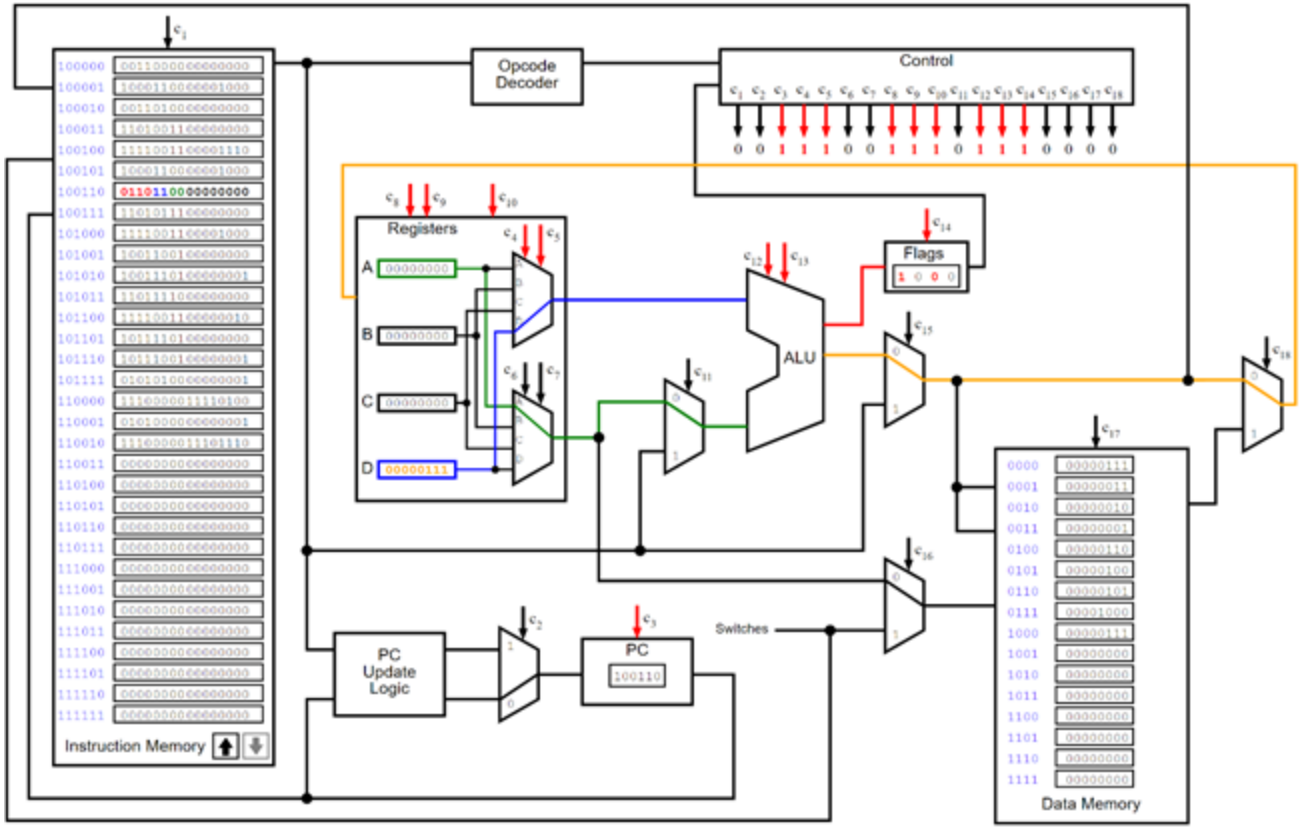
Historic Timeline

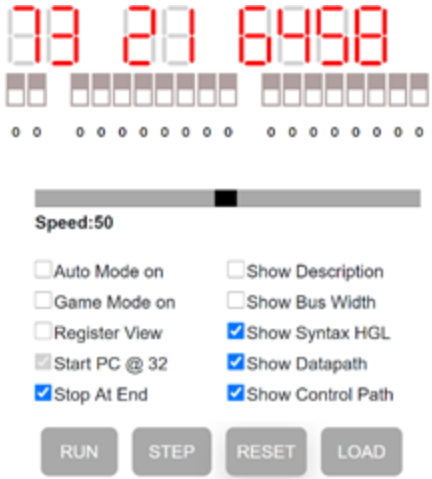


FPGA Implementation- Summer 2019

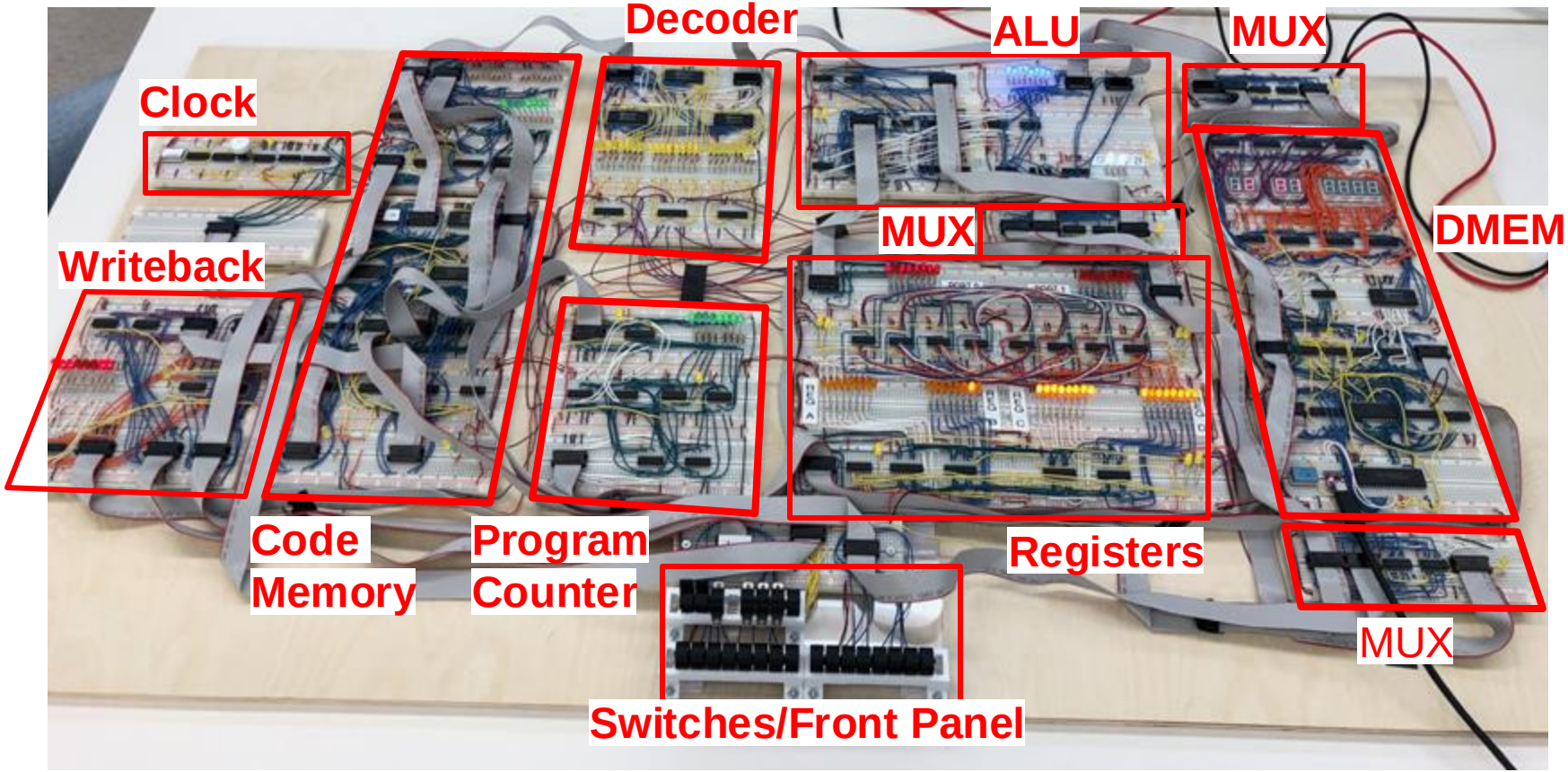


Simulator - May 2021

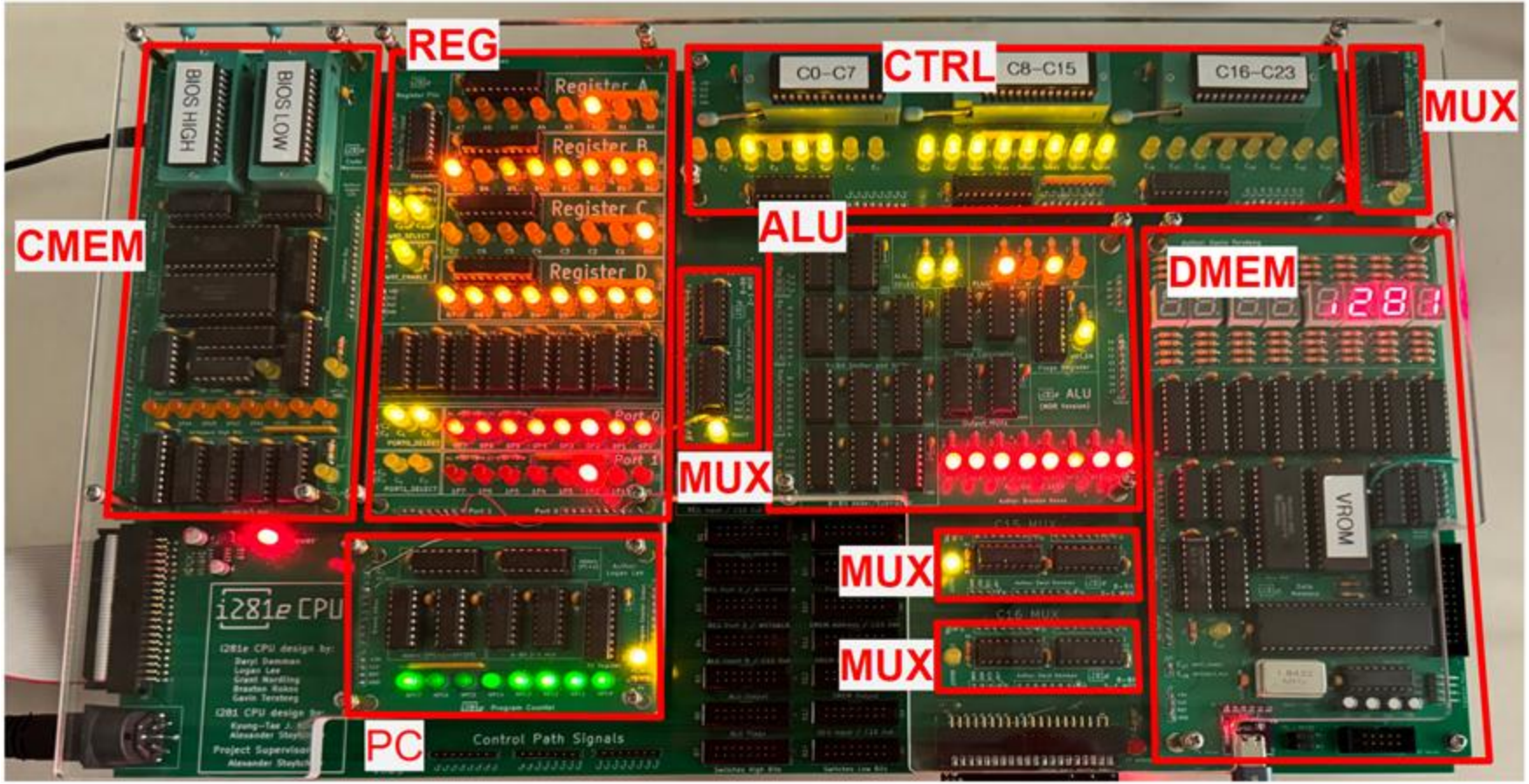




Breadboard Implementation - Dec 2023 to Feb 2024



PCB Implementation - May 2024



i281e Processor Specifications

- Clock Speed: 1Hz up to 2MHz (up to 2.5MHz overclock)
  - Processor fails around 2.75MHz
- Power Requirements: 0.8A @ 5VDC (Input 5-12VDC)
  - Fuse for overcurrent protection @ 2A
- Memory: 32 KW (64 KB) of Code RAM, 32 KB of Data RAM
  - Also includes 128 words of Code ROM for booting the system
- Compact Flash: extended memory for long-term storage
  - Acts as the “hard disk” and stores the OS and File System for DOS/281

Existing Resources

- GitLab
  - PCB Schematics
  - User Manual
- Electronics and Technology Group (ETG)
- Breadboard Implementation
- Previous Team Websites
- CprE 281 Lectures
- Professor Stoytchev

Chapter 4: i281e Programming

4.1: Structure of an Instruction

Between different instruction set architectures, there is a wide diversity in how machine code instructions are encoded. Different processors employ a wide range of techniques to balance performance, memory consumption, and technical limitations. Some, like the MIPS processor, try to make instruction formats as simple as possible to reduce decoding times. Others, like the x86 family of processors, include a wide variety of valid instructions to make the most of each execution cycle. A modern x86-64 processor can execute 981 unique instructions, ranging from one to fifteen bytes long.

The i281e does not come close to that in terms of complexity. Each instruction is exactly 16 bits long. It is fetched, executed, and terminated in exactly one clock cycle. Not every instruction ends up using all 16 bits but keeping them at that length helps keep the i281e conceptually simple. An instruction can be broken up into two parts.

- **Opcode**: The highest 8 bits, which are sent to the control table and is used to generate the control signals for that instruction. This, along with the flag inputs, determines what the processor does during a specific clock cycle. The control table is the only part of the processor to receive the opcode.
- **Operand**: The lower 8 bits, which are sent to C_{10} and C_{15} to be used in the data path of the processor. The operand has no influence on what the control signals are during the instruction execution cycle. On the hardware, the value is known as the "Immediate Value."

The opcode itself can be broken up further. The top 4 bits of the opcode determine what "group" of instruction executes. Groups are a vague category that lump similar instruction together to ease decoding logic complexity. Some instruction groups contain several similar functions, while most contain only one. The bottom 4 are used as arguments to augment the function of that instruction further.

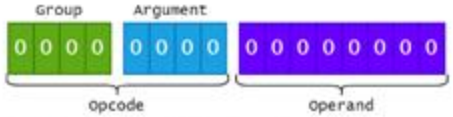


Fig 4.1.1: Basic structure of an i281e instruction

Overview and Requirements

Project Summary

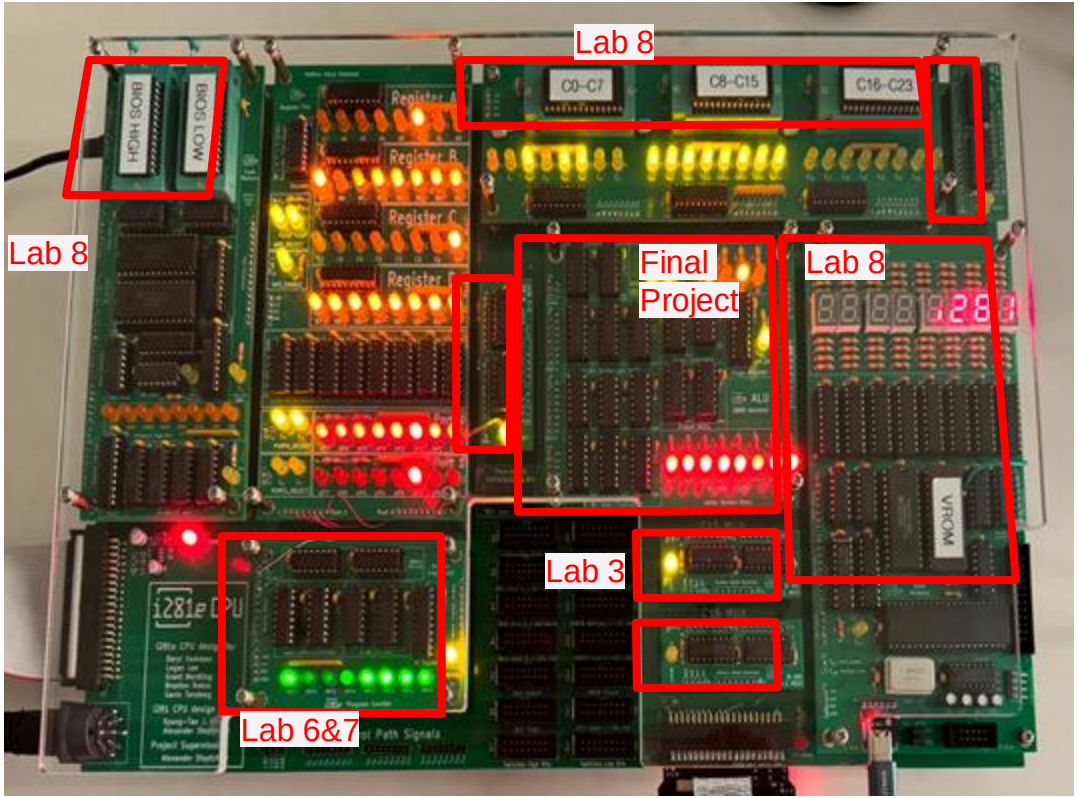
- Goal: Take these open source hardware and software designs and implement a set of curriculum and outreach activities around them
- Each activity will be tested and documented in detail
- These documents could also be used as educational materials for existing classes or to support future lectures and labs
- A subset of these materials will be used for outreach activities
- Implement another i281e CPU on PCB and document the process

Project Summary Cont.

- Design, create and test at least 10 labs/activities based around the i281e processor completable within a lab period
- To be incorporated into a new class tentatively referred to as:
  - CprE 3710x : Microprocessors and Digital Circuits

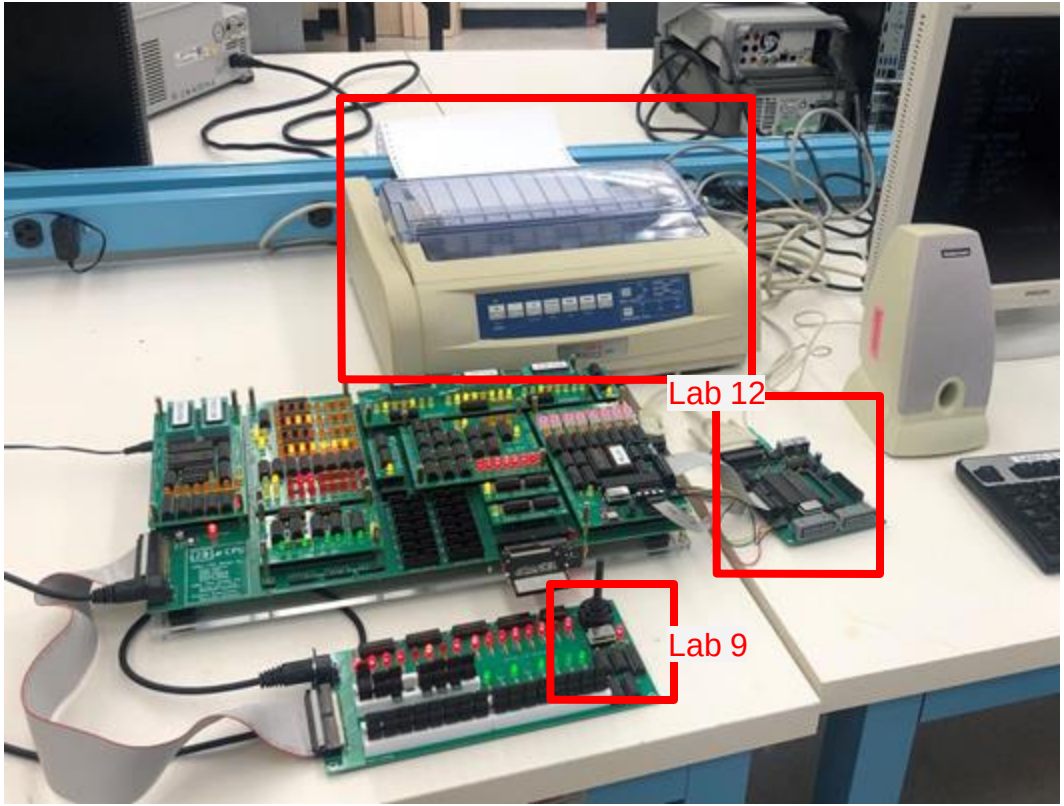
CprE 3710x

CprE 3710x Timeline by Week



1. Intro to Breadboards: 2-to-1 MUX
2. Debouncing, LEDs, Resistors
3. Standardization and Connectors: Bus MUX
4. Introduction to KiCAD
5. Mini-Project: MUX in KiCAD and order PCB
6. Program Counter Pt 1: Logic Debugging
7. Program Counter Pt 2: Hardware Debugging
8. EEPROMs: Program 7-Segment Decoder
9. Clock Circuit + Final Project Proposal
10. Assembly Level Programming
11. Video Game in Assembly
12. Peripheral Devices
13. Thanksgiving Break
14. Final Project Pt 1
15. Final Project Pt 2

CprE 3710x Timeline by Week



1. Intro to Breadboards: 2-to-1 MUX
2. Debouncing, LEDs, Resistors
3. Standardization and Connectors: Bus MUX
4. Introduction to KiCAD
5. Mini-Project: MUX in KiCAD and order PCB
6. Program Counter Pt 1: Logic Debugging
7. Program Counter Pt 2: Hardware Debugging
8. EEPROMs: Program 7-Segment Decoder
9. Clock Circuit + Final Project Proposal
10. Assembly Level Programming
11. Video Game in Assembly
12. Peripheral Devices
13. Thanksgiving Break
14. Final Project Pt 1
15. Final Project Pt 2

Considerations

Users

Primary

- Prof. Stoytchev (Client)
  - Curriculum
  - Past i281 CPU Work
- Undergraduate Students
  - Previous knowledge
  - Time constraints
  - Lab room constraints

Secondary

- Teaching Assistants
- Outreach Coordinators (WiSE)
- Middle and High School students

Hardware

Chips:

- Need to be available with plenty of stock for the labs
- Some chips used in i281e processor no longer in production (EEPROM)
- Need to find replacements that are pin compatible to processor

Wiring:

- The lab room for the class might not allow for cutting wires
- Wire kits have limited lengths and colors
- Some hardware, like the power supply may short, causing delays in testing

Image shown is a representation only. Exact specifications should be obtained from the product data sheet.

W27C512-45Z

DigiKey Part Number	W27C512-45Z-ND
Manufacturer	Winbond Electronics
Manufacturer Product Number	W27C512-45Z
Description	IC EEPROM 512KBIT PARALLEL 28DIP
Customer Reference	
Detailed Description	EEPROM Memory IC 512Kbit Parallel 45 ns 28-PDIP
Datasheet	Datasheet
EDA/CAD Models	W27C512-45Z Models

Software

AUTODESK® TINKERCAD®

KiCad

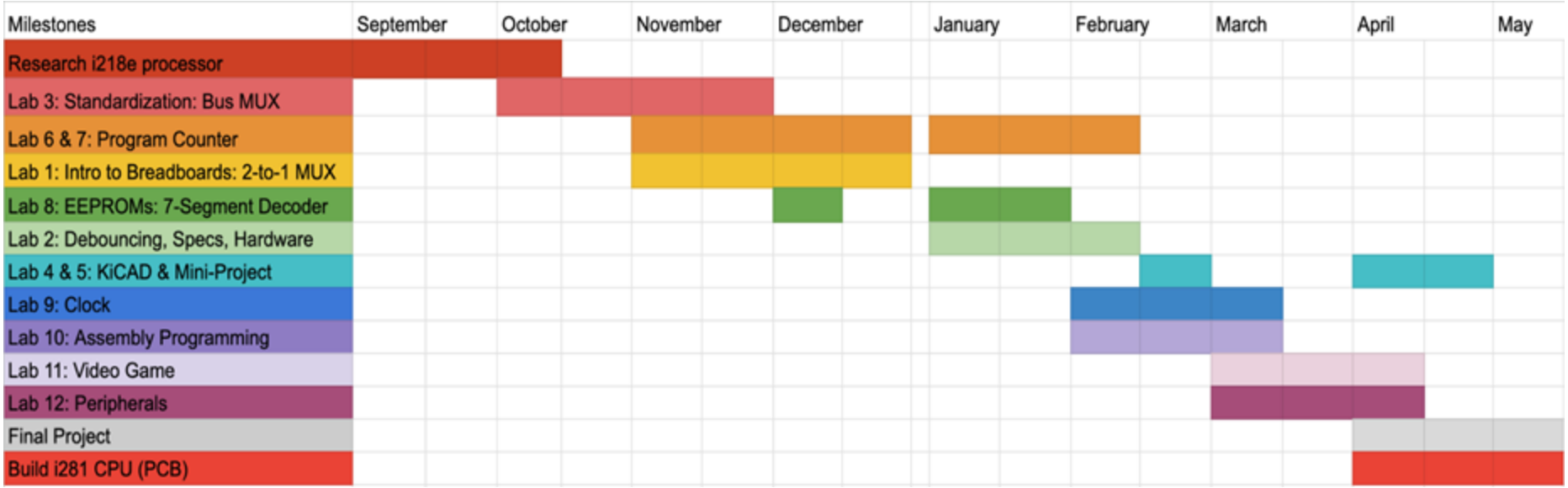
XGecu®

Ordering Parts

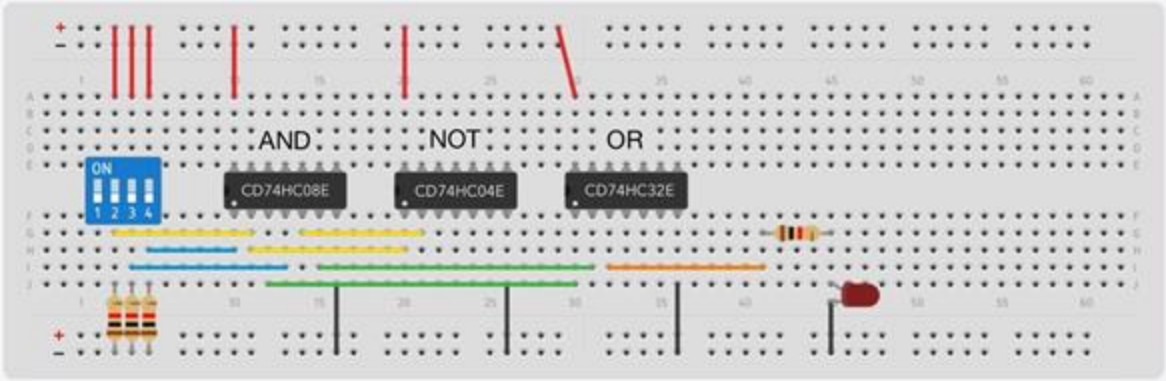
- Parts for the labs were ordered through ETG
- For microchip specifications, master BOM from i281e group was used for simplicity and compatibility between our modules and the processor
- Used websites like Digikey to find required information for the ETG to get these parts ordered
- Each order took about a week to a week and a half to arrive

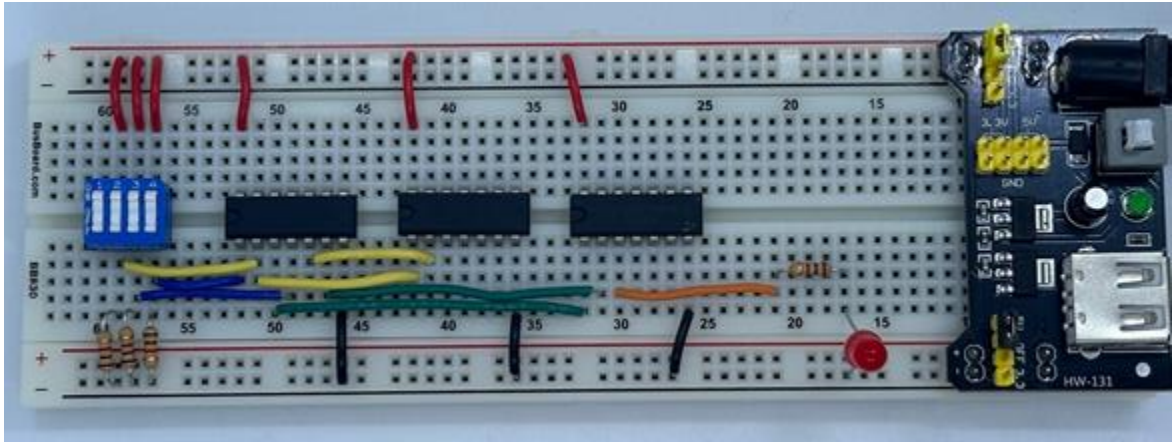
Progress and Future Plans

Gantt Chart

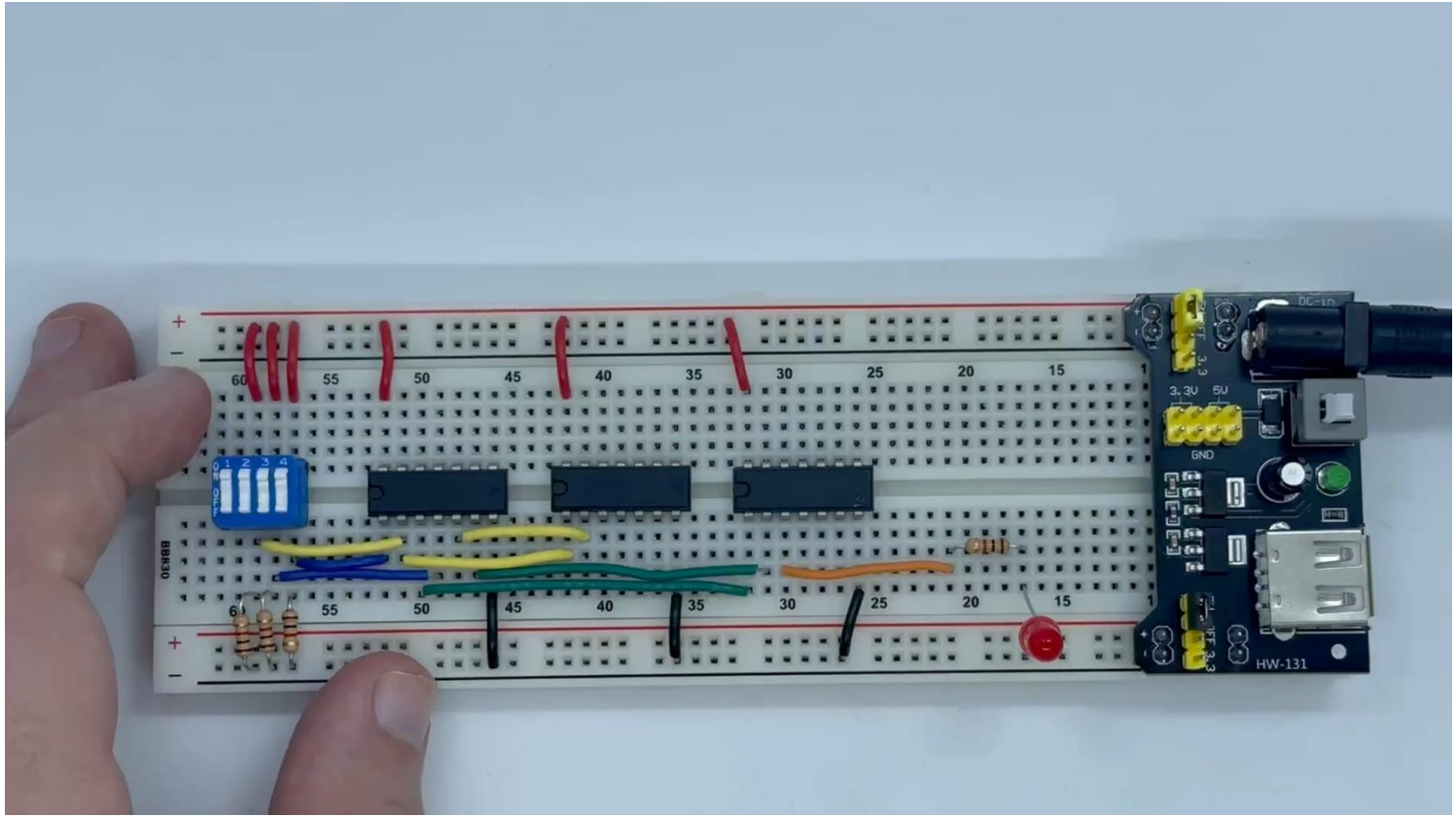

Milestones
September
October
November
December
January
February
March
April
May
Research i218e processor
Lab 3: Standardization: Bus MUX
Lab 6 & 7: Program Counter
Lab 1: Intro to Breadboards: 2-to-1 MUX
Lab 8: EEPROMs: 7-Segment Decoder
Lab 2: Debouncing, Specs, Hardware
Lab 4 & 5: KiCAD & Mini-Project
Lab 9: Clock
Lab 10: Assembly Programming
Lab 11: Video Game
Lab 12: Peripherals
Final Project
Build i281 CPU (PCB)

Lab 1: Bitwise MUX

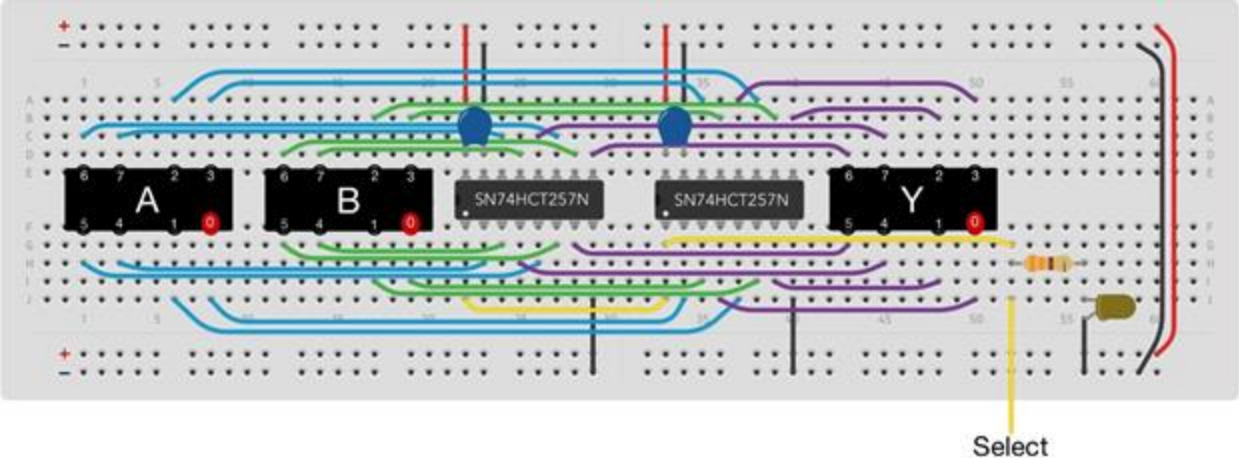


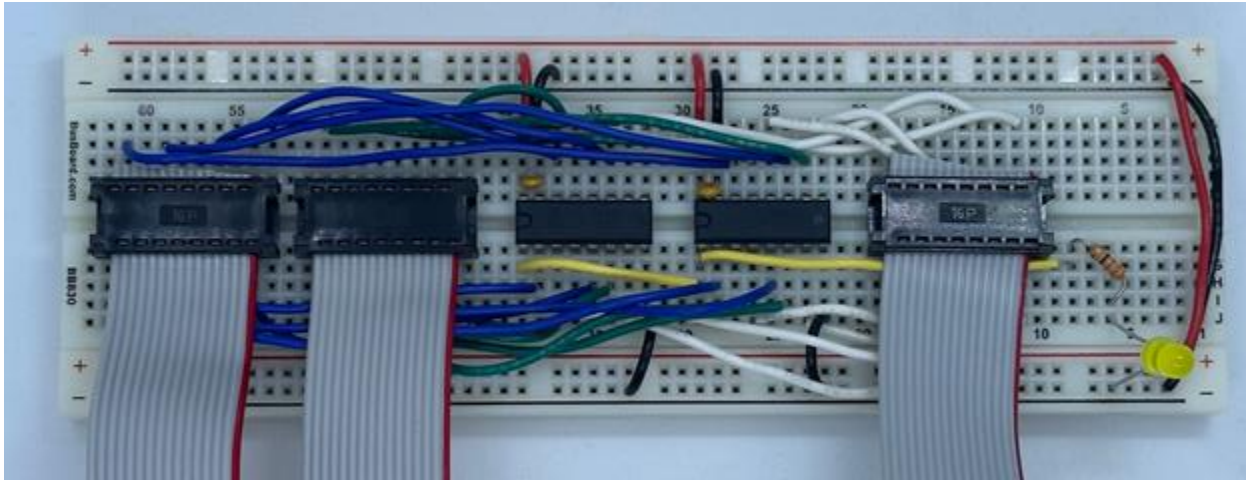


Lab 1: Demo

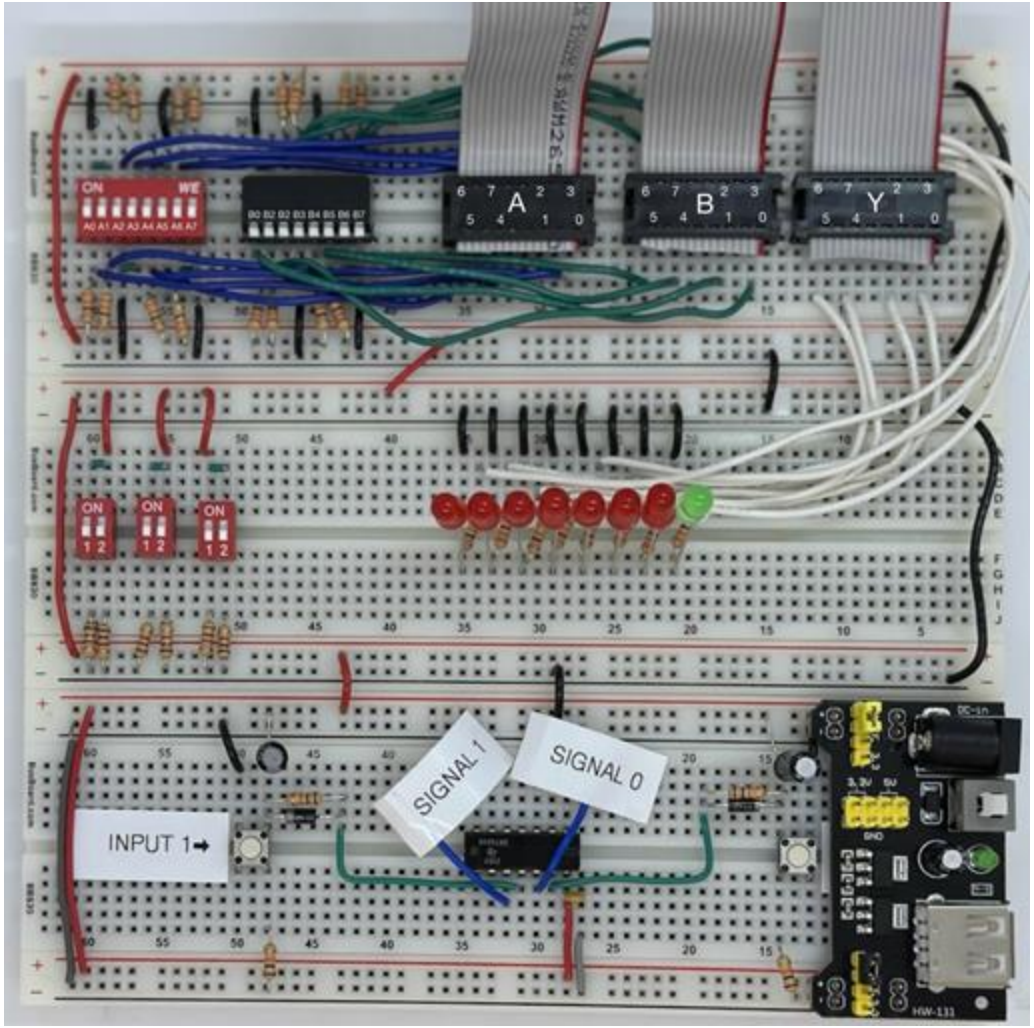


Lab 3: Bus MUX

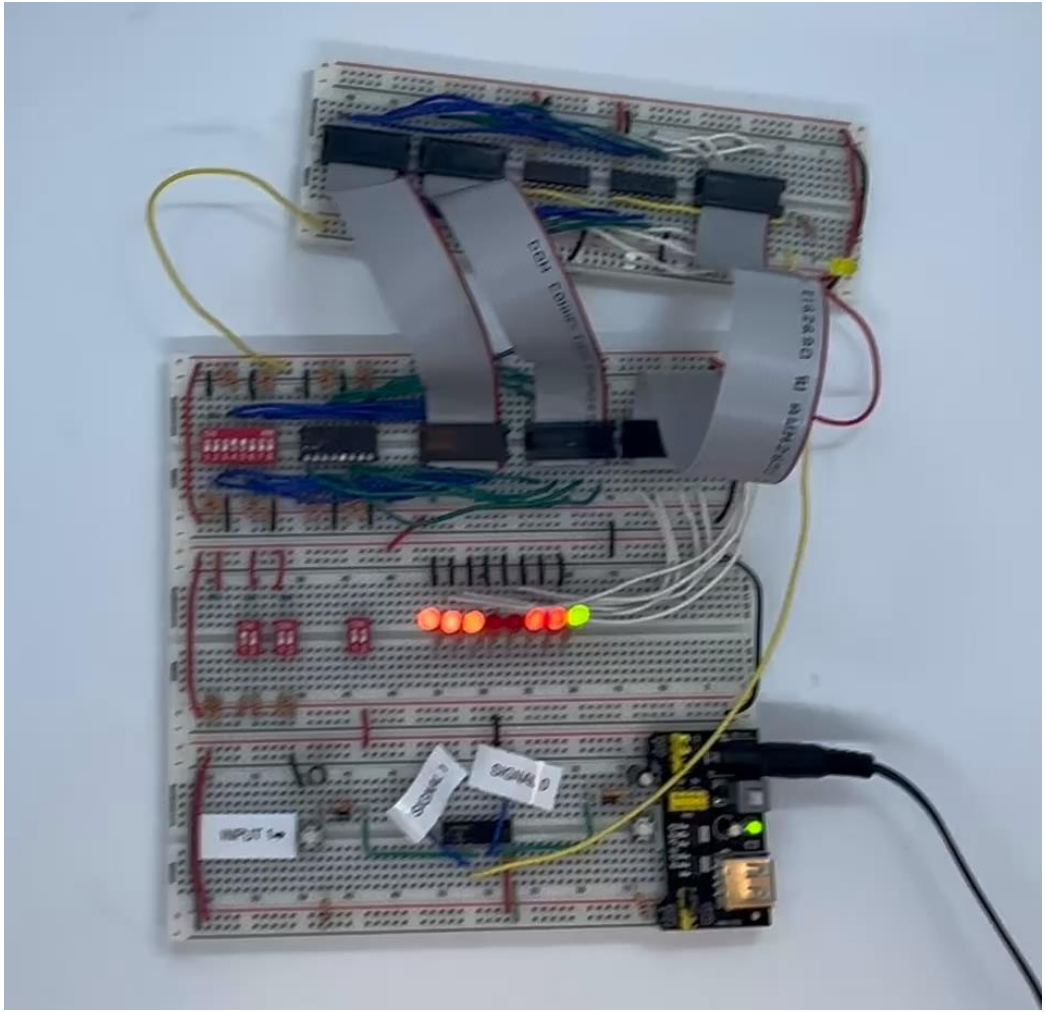




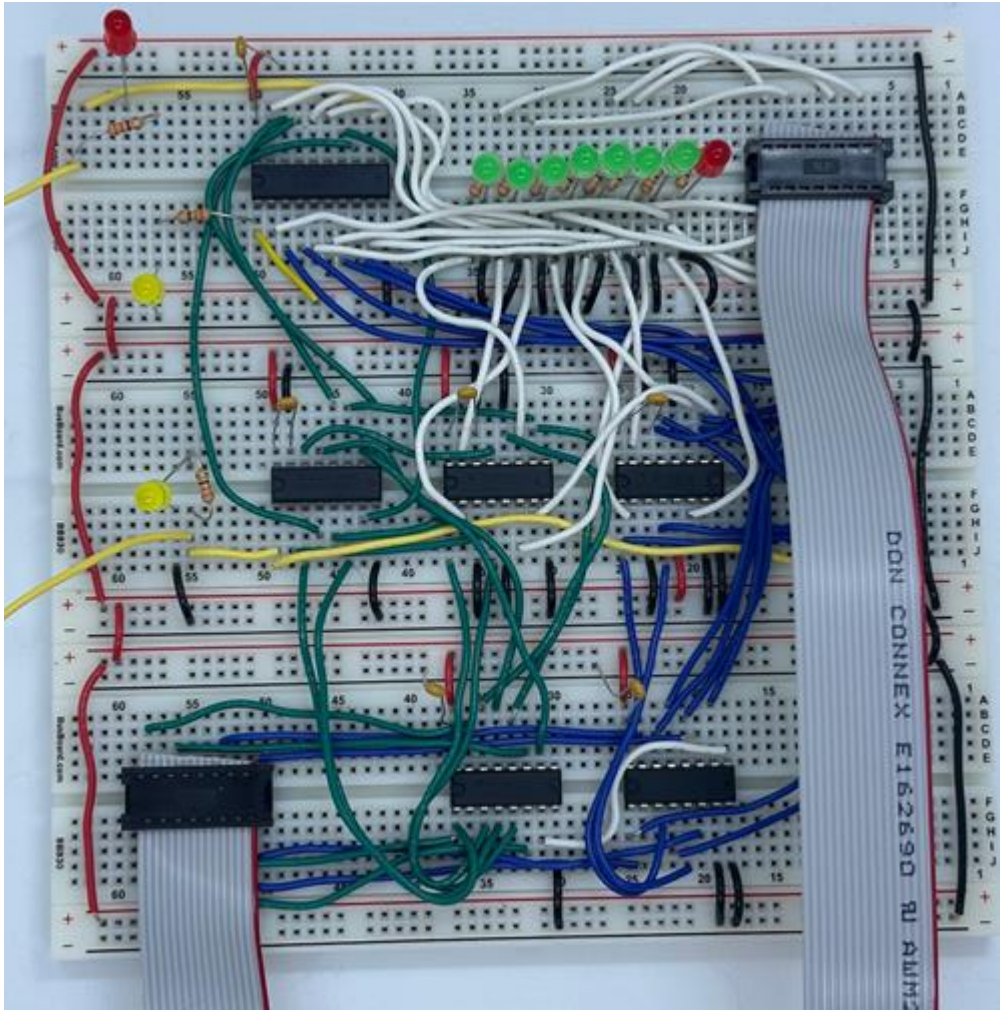
Tester Circuit



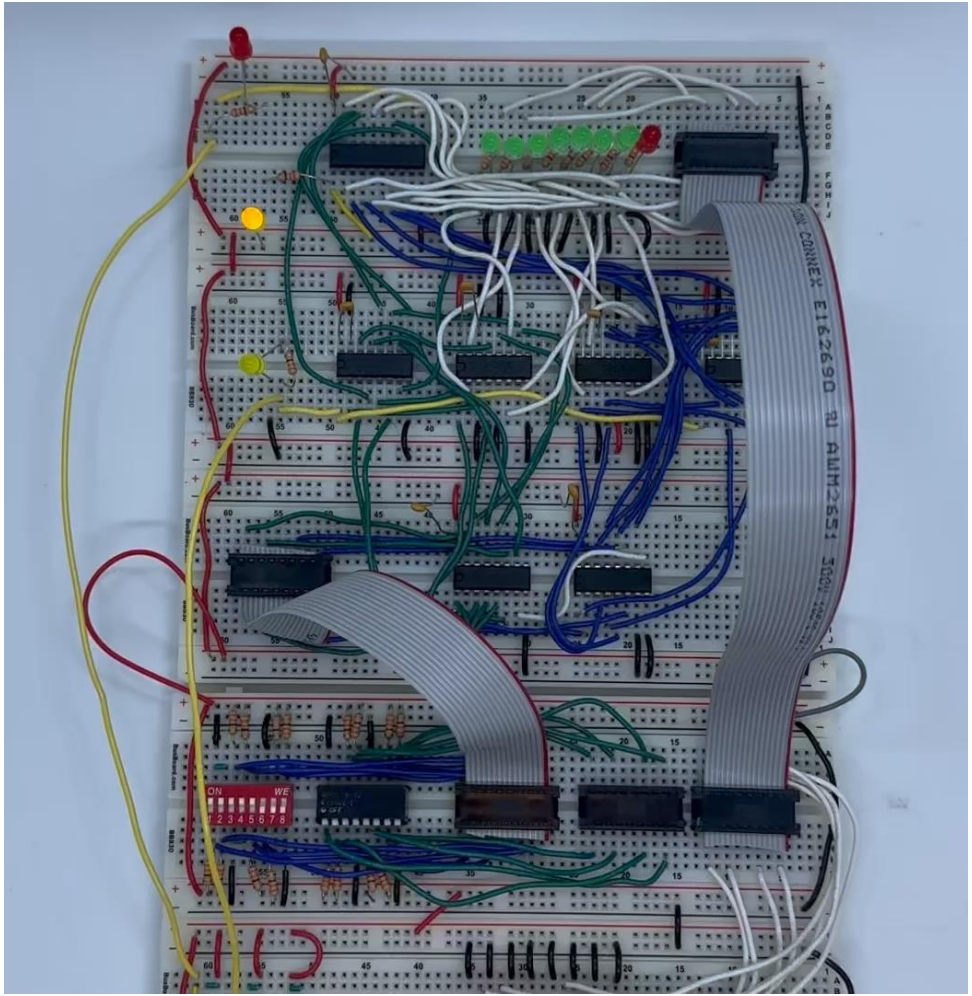
Bus MUX Demo



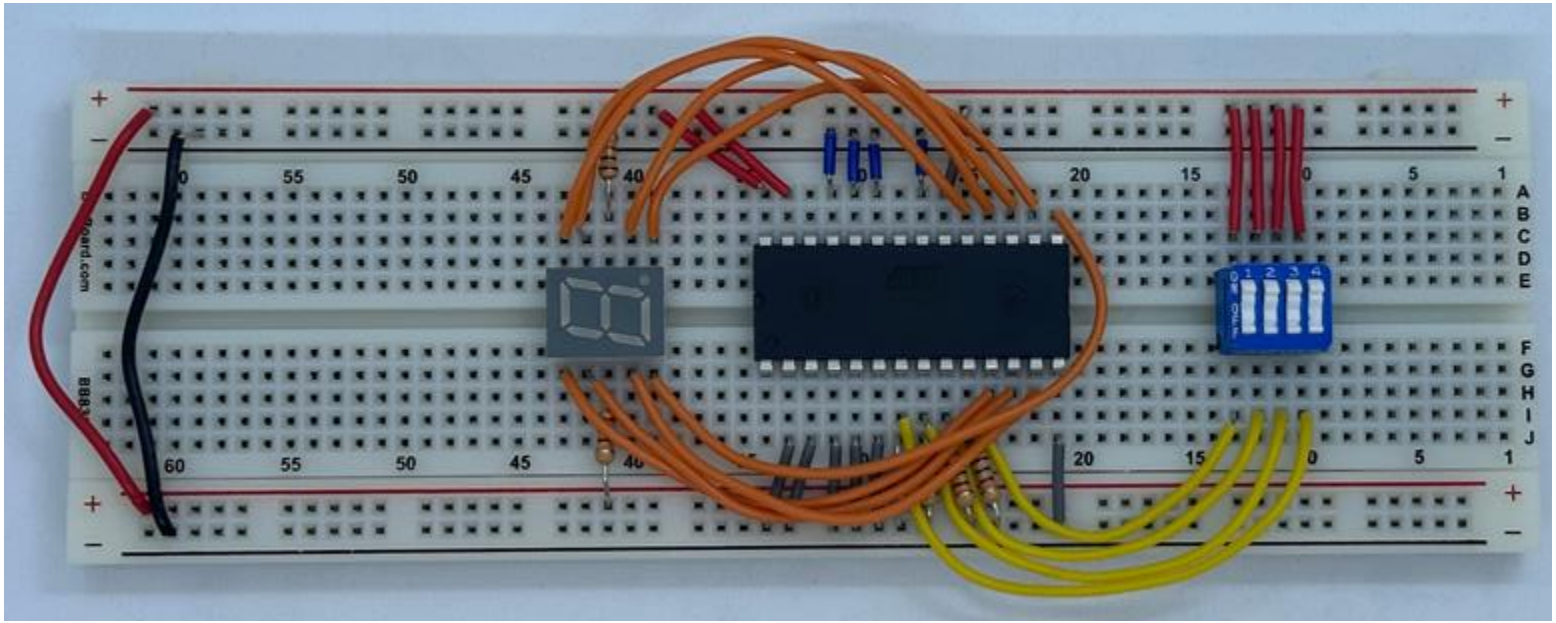
Program Counter



Program Counter Demo



7-Segment Decoder with EEPROM



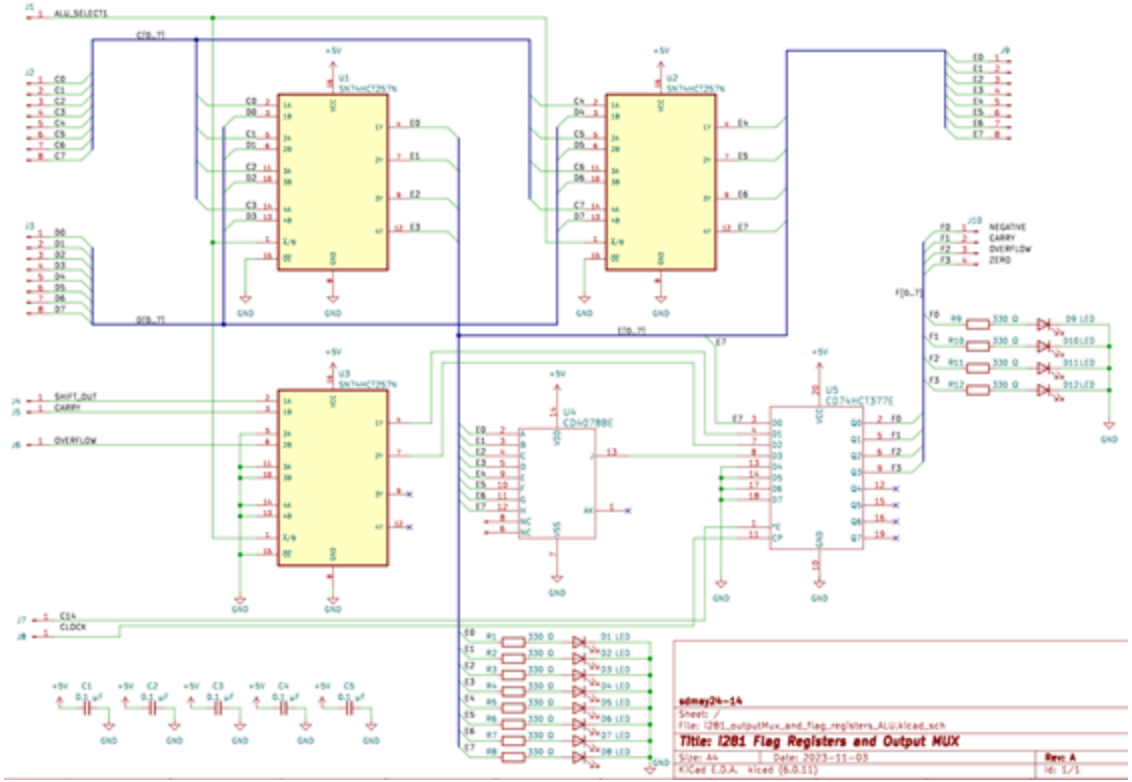
Assembly Level Programming

- Implement a video game in assembly
- Rock, Paper, Scissors
- Inputs with push buttons
- Score and timing visualized by 7-segment display
- Modify program to be appropriate difficulty

	0x-0	0x-1	0x-2	0x-3	0x-4	0x-5	0x-6	0x-7	0x-8	0x-9	0x-A	0x-B	0x-C	0x-D	0x-E	0x-F
0x0-	BANK A+*	---	---	---	BANK B+*	---	---	---	BANK C+*	---	---	---	BANK D+*	---	---	---
0x1-	INPUTC [*]	INPUTCF [A+*]	INPUTD [*]	INPUTDF [A+*]	CACHE A	INPUTCF [B+*]	WRITE [B+*],A	INPUTDF [B+*]	---	INPUTCF [C+*]	WRITE [C+*],A	INPUTDF [C+*]	---	INPUTCF [D+*]	WRITE [D+*],A	INPUTDF [D+*]
0x2-	MOV A,A NOOP	MOV A,B	MOV A,C	MOV A,D	MOV B,A	MOV B,B NOOP	MOV B,C	MOV B,D	MOV C,A	MOV C,B	MOV C,C NOOP	MOV C,D	MOV D,A	MOV D,B	MOV D,C	MOV D,D NOOP
0x3-	LOADI A,*	---	---	---	LOADI B,*	---	---	---	LOADI C,*	---	---	---	LOADI D,*	---	---	---
0x4-	ADD A,A SHIFTL A	ADD A,B	ADD A,C	ADD A,D	ADD B,A	ADD B,B SHIFTL B	ADD B,C	ADD B,D	ADD C,A	ADD C,B	ADD C,C SHIFTL C	ADD C,D	ADD D,A	ADD D,B	ADD D,C	ADD D,D SHIFTL D
0x5-	ADDI A,*	---	---	---	ADDI B,*	---	---	---	ADDI C,*	---	---	---	ADDI D,*	---	---	---
0x6-	SUB A,A	SUB A,B	SUB A,C	SUB A,D	SUB B,A	SUB B,B	SUB B,C	SUB B,D	SUB C,A	SUB C,B	SUB C,C	SUB C,D	SUB D,A	SUB D,B	SUB D,C	SUB D,D
0x7-	SUBI A,*	---	---	---	SUBI B,*	---	---	---	SUBI C,*	---	---	---	SUBI D,*	---	---	---
0x8-	LOAD A,[*]	---	---	---	LOAD B,[*]	---	---	---	LOAD C,[*]	---	---	---	LOAD D,[*]	---	---	---
0x9-	LOADF A, [A+*]	LOADF A, [B+*]	LOADF A, [C+*]	LOADF A, [D+*]	LOADF B, [A+*]	LOADF B, [B+*]	LOADF B, [C+*]	LOADF B, [D+*]	LOADF C, [A+*]	LOADF C, [B+*]	LOADF C, [C+*]	LOADF C, [D+*]	LOADF D, [A+*]	LOADF D, [B+*]	LOADF D, [C+*]	LOADF D, [D+*]
0xA-	STORE [*],A	---	---	---	STORE [*],B	---	---	---	STORE [*],C	---	---	---	STORE [*],D	---	---	---
0xB-	STOREF [A+*],A	STOREF [B+*],A	STOREF [C+*],A	STOREF [D+*],A	STOREF [A+*],B	STOREF [B+*],B	STOREF [C+*],B	STOREF [D+*],B	STOREF [A+*],C	STOREF [B+*],C	STOREF [C+*],C	STOREF [D+*],C	STOREF [A+*],D	STOREF [B+*],D	STOREF [C+*],D	STOREF [D+*],D
0xC-	NORI A,*	SHIFTR A	---	---	NORI B,*	SHIFTR B	---	---	NORI C,*	SHIFTR C	---	---	NORI D,*	SHIFTR D	---	---
0xD-	CMP A,A	CMP A,B	CMP A,C	CMP A,D	CMP B,A	CMP B,B	CMP B,C	CMP B,D	CMP C,A	CMP C,B	CMP C,C	CMP C,D	CMP D,A	CMP D,B	CMP D,C	CMP D,D
0xE-	NOR A,A	NOR A,B	NOR A,C	NOR A,D	NOR B,A	NOR B,B	NOR B,C	NOR B,D	NOR C,A	NOR C,B	NOR C,C	NOR C,D	NOR D,A	NOR D,B	NOR D,C	NOR D,D
0xF-	BRC * BRAE *	BRNC * BRB *	BRO *	BRNO *	BRN *	BRNN * BRP *	BRZ * BRE *	BRNZ * BRNE *	BRA *	BRBE *	BRG *	BRGE *	BRL *	BRLE *	JUMPR C+*	JUMP *

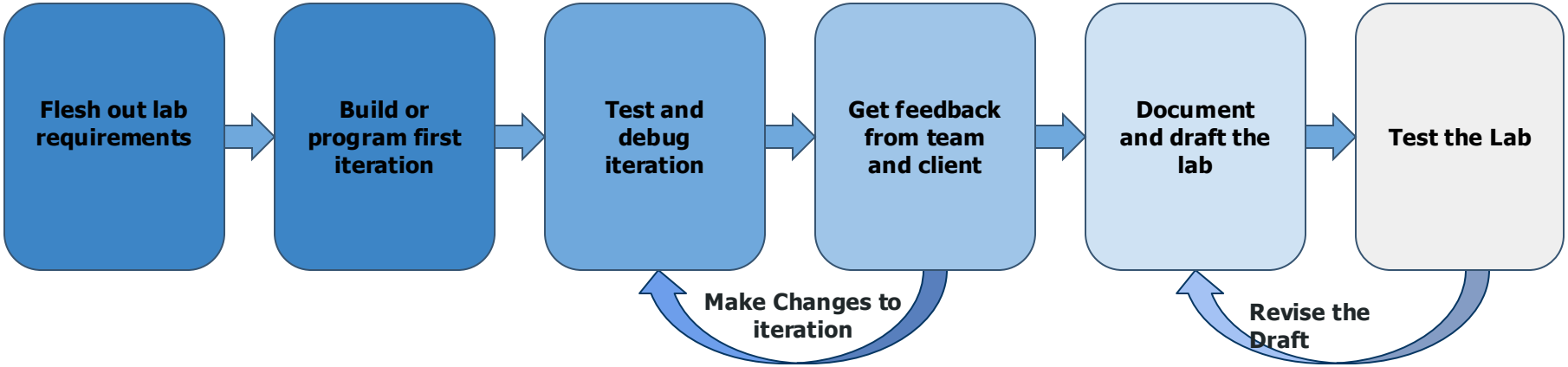
Final Project

- Write a Device Driver
- Implement a Register File
- Implement the ALU w/ modifications
- Design a peripheral device



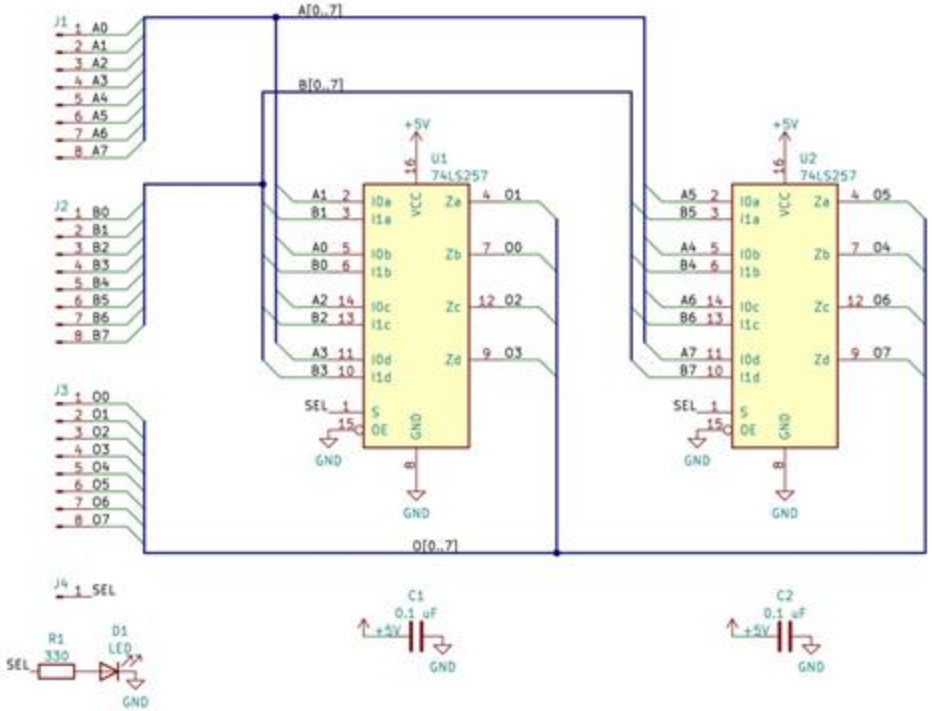
Design Process

Design Iterations



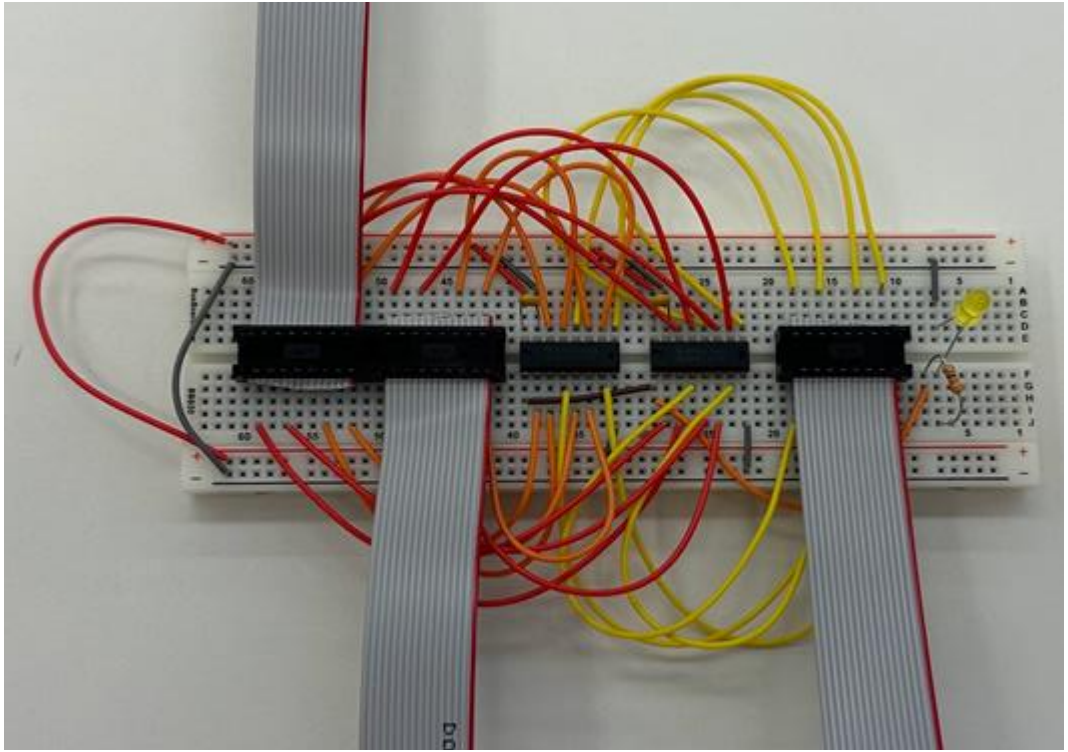
Flesh Out Lab Requirements

- What do we want to accomplish with this lab?
  - Teach students how to implement a hardware Bus
  - Familiarize students with CPU component
- What do the students need to learn?
  - What is a Bus?
  - What is Standardization?
- How can we accomplish this?
  - Have the students make a component they've already learned about, but as a Bus



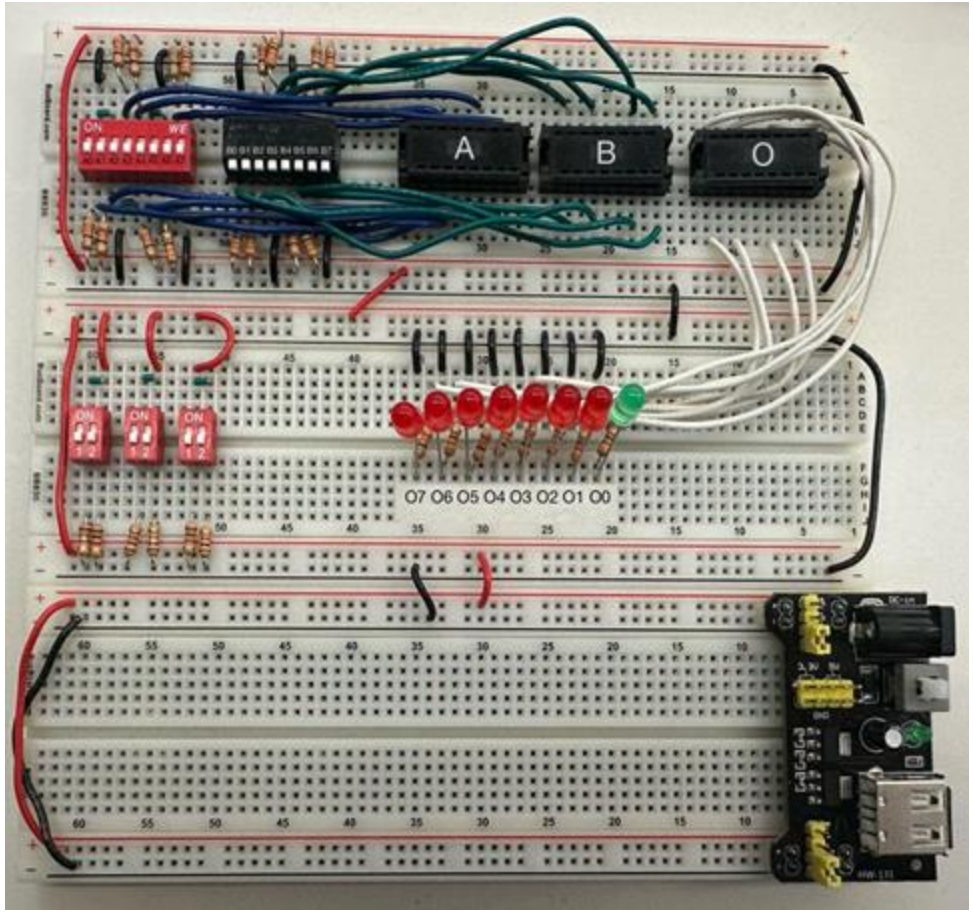
Build Our First Iteration

- Using wires with predetermined length we implemented the logic
- Based our design off of the 2-to-1 8-bit Bus Multiplexers from the i281 breadboard implementation



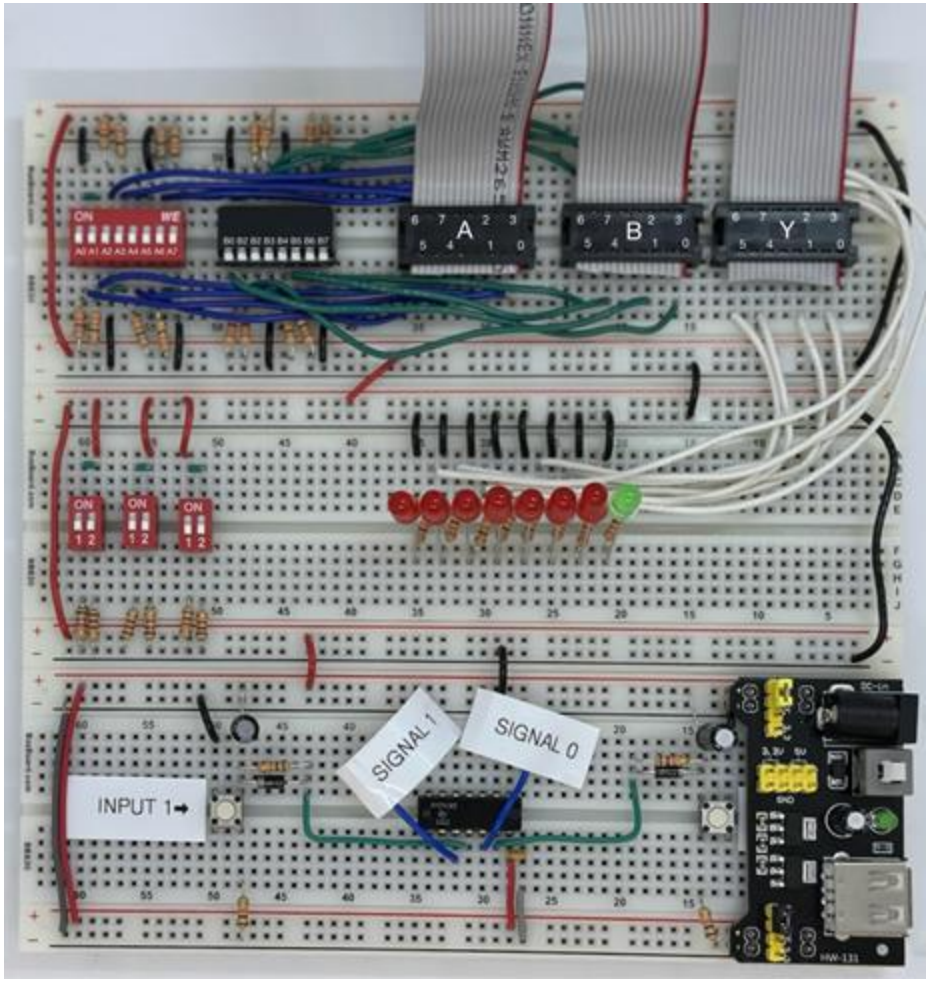
Test and Debug Circuit

- Needed to test the circuit to verify that the circuit works as expected
- Created a testing circuit



Test and Debug Circuit

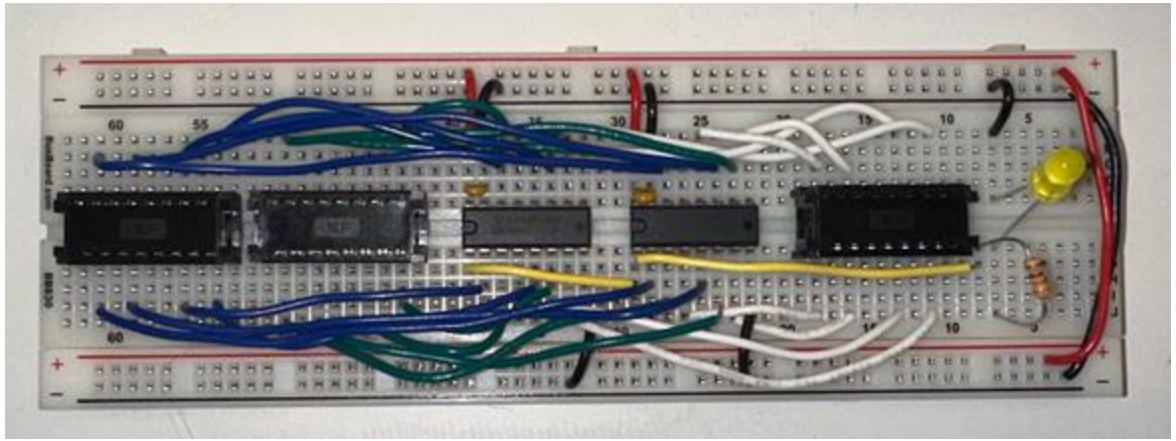
- Through testing our tester circuit we found that the tester needed debouncing added to account for implementations that use clocks



Get Feedback from Team and Client

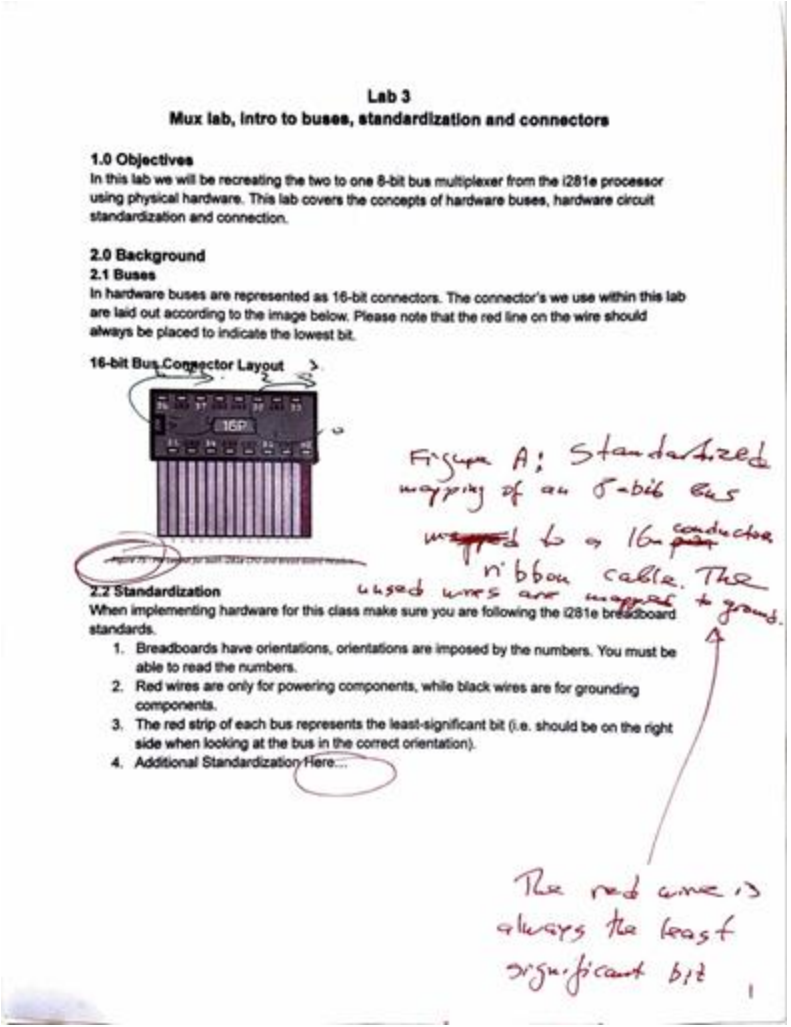
Used Cut Wires to Standardize Colors:

- Input A as blue
- Input B as green
- Output as white
- Select as yellow
- Power as red
- Ground as black



Document and Draft the Lab

- Documented circuit design and began creating instructions
- Added relevant background information and pre-lab
- Finalize testing procedure
- Get feedback from client
- Repeat



Lab 3

Mux lab, intro to buses, standardization and connectors

1.0 Objectives

In this lab we will be recreating the two to one 8-bit bus multiplexer from the i281e processor using physical hardware. This lab covers the concepts of hardware buses, hardware circuit standardization and connection.

2.0 Background

2.1 Buses

In hardware buses are represented as 16-bit connectors. The connector's we use within this lab are laid out according to the image below. Please note that the red line on the wire should always be placed to indicate the lowest bit.

16-bit Bus Connector Layout

Figure A: Standardized mapping of an 8-bit bus mapped to a 16 conductor ribbon cable. The unused wires are mapped to ground.

2.2 Standardization

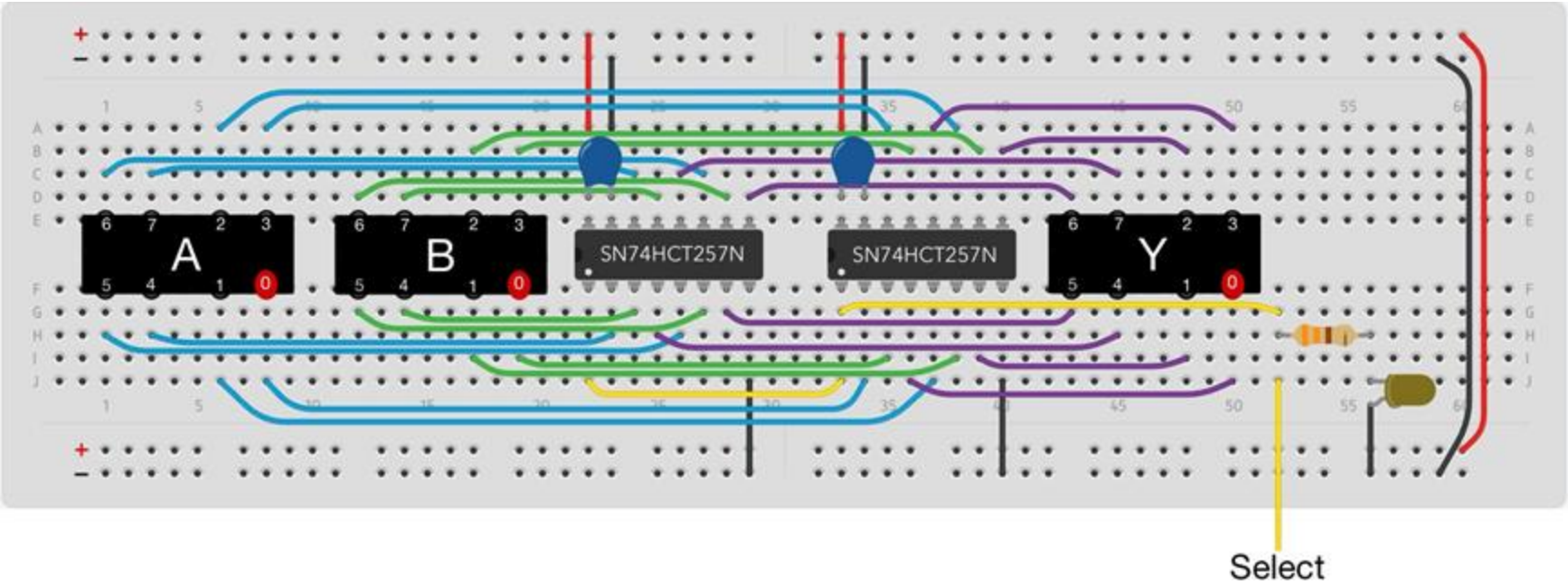
When implementing hardware for this class make sure you are following the i281e breadboard standards.

1. Breadboards have orientations, orientations are imposed by the numbers. You must be able to read the numbers.
2. Red wires are only for powering components, while black wires are for grounding components.
3. The red strip of each bus represents the least-significant bit (i.e. should be on the right side when looking at the bus in the correct orientation).
4. Additional Standardization Here...

The red wire is always the least significant bit

Test the Lab

- Have some volunteers follow along the lab
- See how long it takes them
- Get feedback



CprE 3710x Lab 3
Electrical and Computer Engineering
Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

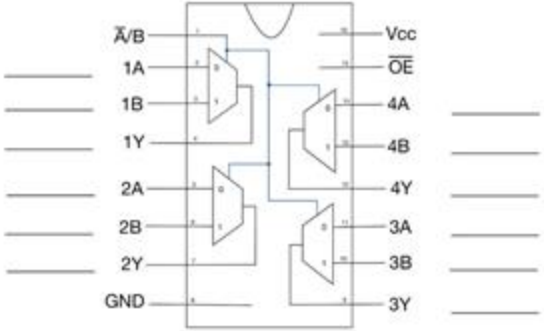
Name: _____ Student ID: _____

Lab Section: _____ Date: _____

Prelab

1. What is the operating voltage for the SN74HCT257N Pin?

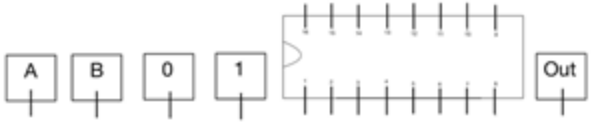
2. Re-label the SN74HCT257N diagram below to be zero-based.



CprE 3710x Lab 3
Electrical and Computer Engineering
Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

3. On the diagram below, connect inputs A and B to any MUX on the SN74HCT257N chip. Connect the output of the MUX to the box labeled "Out" and use the '0' or '1' to select A as the output.



Lab

4.1 Verify that you placed your chips, connectors, power and ground wires correctly. Show your progress on the breadboard implementation to the TA before you proceed.

TA Initials: _____

4.3 Verify that you connected the inputs to the SN74HCT257N(MUX) chips correctly. Show your progress on the breadboard implementation to the TA before you proceed.

TA Initials: _____

4.5 Verify that you connected the outputs from the SN74HCT257N(MUX) chips to the output connector correctly. Also, verify that you connected the select wire to the SN74HCT257N(MUX) chips and the LED. Make sure that the cathode of the LED is connected to ground. Show your completed breadboard implementation to the TA before you proceed.

TA Initials: _____

CprE 3710x Lab 3

Electrical and Computer Engineering

Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

1.0 Objectives

In this lab, you will recreate the two-to-one 8-bit bus multiplexer from the i281e processor on a breadboard. This lab also covers the concepts of hardware buses, hardware circuit standardization, and connectors.

2.0 Parts List

Quantity	Items	
1	White 830-point Breadboard	
Set of	Breadboard Wire Spools	Pre Cut Wire Kit
1	Wire Cutters Electronic Grade	
1	Wire Strippers Electronic Grade	
2	Quad 2-1 MUX (SN74HCT257N) Chips	
2	0.1 uF Ceramic Capacitor	
1	5mm Yellow LED	
1	330 Ω THT Resistor	
3	Connectors, 16 position ribbon cable, DIP header connector IDC through hole	
3	Connectors, 16 position rectangular receptacle connector IDC	
3	8 to 12 inches Ribbon Cable	

CprE 3710x Lab 3

Electrical and Computer Engineering

Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

3.0 Background

3.1 8-bit Buses

Figure 3 shows an 8-bit bus implemented with a 16-conductor ribbon cable. Because we have twice as many wires than we need,half of which are not used. Figure 1 shows the mapping of the wires to the 16-position connector. The unused wires are labeled with G. Figure 2 shows the pins of the connector designed to plug into the breadboard. Please note that the red wire indicates the least significant bit.

Figure 1: Standardized mapping of an 8-bit bus to a 16 conductor ribbon cable.

Figure 2: Side view of the connector pins.

Figure 3: An 8-bit bus with connectors on both sides.

Bus Multiplexer and Intro to Standardization and Connectors

To implement the 8-bit 2-to-1 bus multiplexer, you will use two SN74HCT257N chips. As shown in Figure 4 each of them contains four 2-to-1 multiplexers that share the same select line. To complete the entire circuit, we need to combine two of the chips and a select line to choose between the two input buses A = (A_7 … A_0) and B = (B_7 … B_0).
The output bus is called Y = (Y_7 … Y_0).
Please refer to the datasheet for additional information about the SN74HCT257N chip.

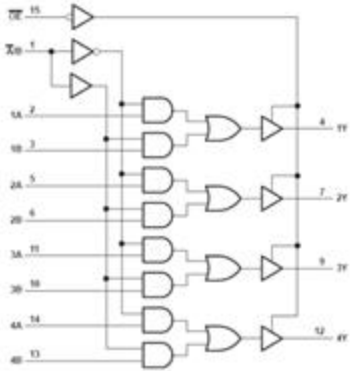


Figure 6: Pin layout for the Quad 2-to-1 MUX Chip (SN74HCT257N).

3.2 Standardization

When implementing circuits for this class, ensure you follow the following breadboard standards:

1. Each Breadboard has an orientation imposed by the numbers. You must be able to read the numbers. (i.e., they are not upside down).
2. Red wires are only for powering components, while black wires are for grounding.
3. The red wire on each ribbon cable represents the least significant bit (i.e., it should be on the right side when looking at the ribbon cable in the correct orientation).
4. When visualizing a 8-bit binary number with LEDs, the least significant bit should always be on the right side when you look at the breadboard from the correct orientation.

CprE 3710x Lab 3
Electrical and Computer Engineering
Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

3.3 Implementation

A 8-bit 2-to-1 bus MUX is typically drawn as shown in Figure 3. This expands into Figure 4 to show the individual 2-to-1 multiplexers, which have the same select input.

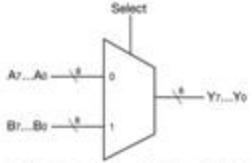


Figure 4: Graphical symbol for a 2-to-1 bus MUX (8-bits wide).

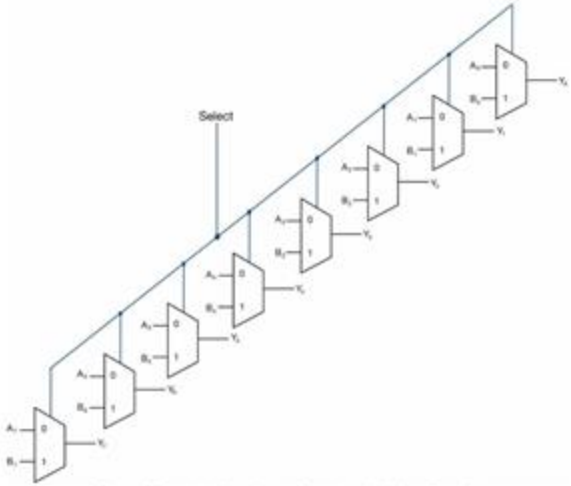


Figure 5: Expanded wiring diagram for the bus MUX from Figure 3.

Bus Multiplexer and Intro to Standardization and Connectors

4.0 Activity

Please complete the pre-lab before starting with the lab. Specifically, you need to understand which pins go where and the orientation of the chips.

4.1 Place the Bus Connectors and the Chips

- Place the chips as shown in Figure 6.
- Connect them to power (pin 16) and ground (pin 8).
- Connect $\overline{OE}$ (pin 15) to ground.
- Place the three bus connectors as shown in Figure 7.

Before continuing to the next step, have your circuit checked by a TA.

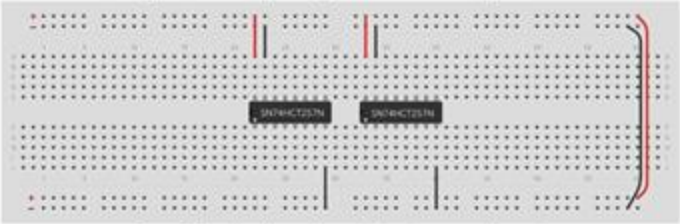


Figure 7: Place the chips and connect them to power and ground.
Also connect their output enable pin to ground (pin 15, $\overline{OE} = 0$).

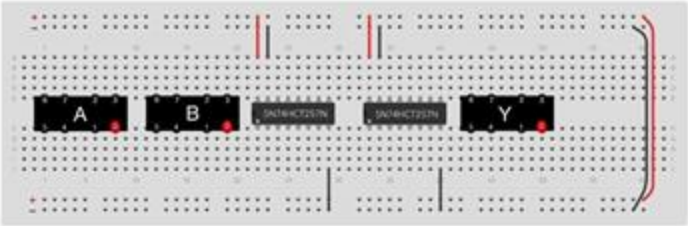


Figure 8: Add the input and output bus connectors. The numbers indicate the breadboard locations that correspond to the data lines.

CprE 3710x Lab 3
Electrical and Computer Engineering
Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

4.4 Connect the Output pins

Start by connecting the four most significant bits (left chip) to the output:

- Top: Connect pins 9 and 12 on the chip to bits Y6 and Y7, respectively.
- Bottom: Connect pins 4 and 7 on the chip to bits Y4 and Y5, respectively.

Next, connect the least significant bits (right chip) to the output

- Top: Connect pins 9 and 12 on the chip to bits Y2 and Y3, respectively.
- Bottom: Connect pins 4 and 7 on the chip to bits Y0 and Y1, respectively.

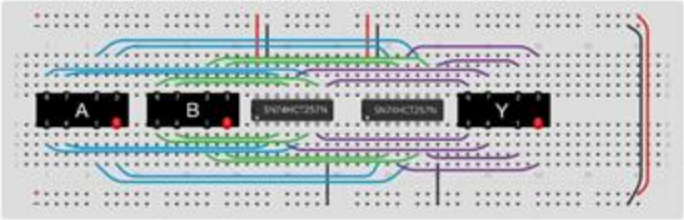


Figure 11: Connect the wires to the output bus (purple wires).

4.5 Finish the Circuit

- Place an LED on the right side of the circuit and ground it.
- Connect the anode of the LED to a 330 Ω resistor. This pin should also be connected to the select input, which may be grounded for now but will connect to the test circuit later.
- Connect the other side of the resistor to both chips at pin 1 (yellow wires in Figure 11).
- Place two 0.1 μF capacitors connecting pins 15 and 16 of each chip.

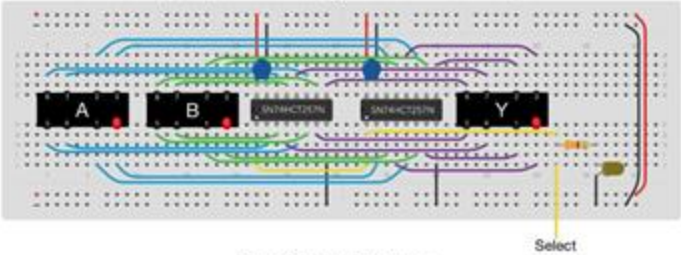


Figure 12: Add the final touches.

4.6 Make Three Ribbon Cables with Connectors

- Next, line up the ribbon with the connector so each connector pin is in contact with one bit of the ribbon.
- Once the ribbon is in place, firmly press down with a flat object until it clicks into place.

4.7 Connect the Ribbon Cables to the MUX

- Figure 14 shows the completed circuit. At this point it uses 3 connectors as placeholders. They are not connected to anything at this point.
- Start with the breadboard upright (the numbers are readable facing you) and replace the connectors with a connector that already has a ribbon cable attached to it. The ribbon should go out the bottom of the breadboard with the red wire on the right side.
- Repeat these steps with the other two ribbons and connectors on your MUX.

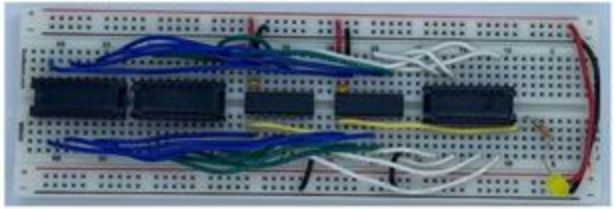


Figure 14: The finished bus MUX with place-holder connectors.

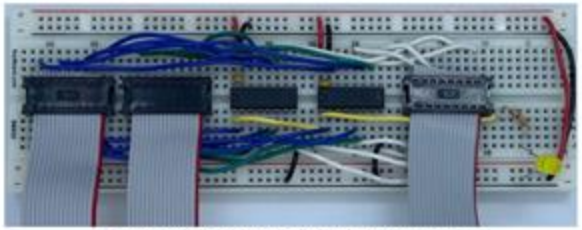


Figure 15: The finished bus MUX with ribbon cables attached.

CprE 3710x Lab 3
Electrical and Computer Engineering
Iowa State University

Bus Multiplexer and Intro to Standardization and Connectors

5.1 Connect the Ribbon Cables to the Tester Circuit

- Place your Bus MUX circuit above the tester circuit. Input A is the leftmost connector on the MUX and tester.
- Repeat this for bus B and the output bus Y.
- Next, connect the select line. To do this, use the select line that was grounded in step 4.5 to a switch on the tester circuit.

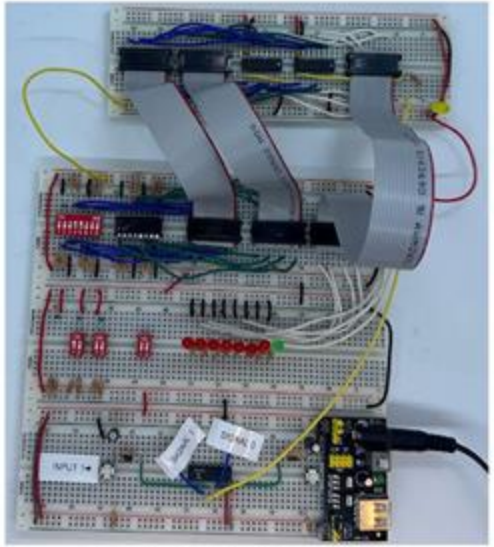


Figure 16: Bus MUX circuit connected to tester circuit.

Conclusion

- 10+ lab activities for a new CprE 3710x class
- Have two labs finalized and progress on three others
- Our labs consist of a mix of digital logic and hardware
- There is a timeline worked out for finishing all the labs
- Also plan to build another PCB implementation

Questions?