|  |
| --- |
|  |

***ECE 343L***

***Microprocessors I***

***Dr. Rafiquzzaman***

***By:***

***Josue Llamas***

***Beshoy Metry***

***Brian Nguyen***

***Duy Duc Pham***

***Jonathan Plata***

***Jose Ventura***

Table of Contents

***Summary 2***

***Introduction 3***

***Materials 4***

***Block Diagram 5***

***P-Spice Schematic 6***

***Memory Map 7***

***I/O Map 8***

***Software (LED) 9***

***Compiled (LED) 10***

***Software (7 Segment) 11***

***Compiled (7 Segment) 12***

***Timeline 14***

***Timeline Description 15***

***Discussion 16***

***Conclusion 19***

**Summary**

The purpose of the Microprocessor I lab is to design and build a Motorola (68000) based

microcomputer using the real mode. The lab provided an excellent understanding of programmed

I/O by using (6821) I/O to transfer data between the microprocessor and external devices (LEDs,

seven segment displays, and switches). These external devices carry out the functions provided

by the program within the microcomputer’s memory (2732), and the microcomputer will control

all transfers completely. Wire wrapping was used to construct the design and avoid noise

problems due to stray capacitances and high frequencies.

***Introduction***

The 68000 was chosen as an entry-level microprocessor for this project because of its architecture and influence on today's microprocessors. The 68K is a 16-bit processor that was designed with the next generation of 32-bit products in mind. The data registers and instruction sets both support 32-bit commands while they do not natively operate on them (most of these commands perform one 16-bit operation twice to accommodate). The 68000 was one of the most widely used chips at the time of its introduction and appeared in many notable products such as Apple Lisa and Atari ST.

The 68K is backwards compatible with 6800 peripheral chips such as the 6821 and 68230. We chose to use the 6821 to take advantage of this compatibility and to lower the cost of our overall design. The 6821 provides two 8-bit bi-directional ports in the form of memory-mapped I/O. Our 6821 chip operates at a substantially slower clock (1/10 of 68000 clock) due to its specific design for the 6800 8-bit processor. The 68000 provides 3 control lines to make up for this: E, , . The 68000 asserts  and if it sees a response on , then it begins to output a clock signal on E at 1/10th of its own main clock.

The main memory we've chosen 8 bit EPROM (2732). It provides an access time of 200ns which is fast enough to allow us to simplify our design by omitting a delay circuit on the data acknowledge line. Because the 68000 can operate at an 8-bit level, the memory must accommodate this and is subsequently broken into two distinct banks: even and odd. The 68K can access either or both chips depending on the current instruction being executed and its requirement for pulling in 8,16 or 32-bits of data. For our own consideration though, this will mean that once a program is compiled it must be broken into two sets of data, odd and even, before being programmed into the EPROMs.

***Lab materials:***

|  |  |
| --- | --- |
| ***Components*** | ***Quantity*** |
| Motorola 68000 | 1 |
| Motorola 6821 | 1 |
| 2732 EPROM | 2 |
| 555 Timer | 1 |
| 1N904 Diode | 1 |
| 4 MHz Crystal Oscillator 4 Pin | 1 |
| 74LS04 | 2 |
| 74LS32 | 1 |
| Push button Switch (2 pins) | 1 |
| Wire-wrap board | 1 |
| Wire-wrap tool for 30 gauge | 1 |
| 30g wires | 1 |
| C = 0.18uF | 1 |
| C = 6.9uF | 1 |
| C = 0.01 uF | 1 |
| R = 100k | 1 |
| R = 1M | 1 |
| IC Socket 40-pin | 1 |
| IC Socket 30-pin | 6 |
| IC Socket 24-pin | 2 |
| IC Socket 18-pin | 6 |
| IC Socket 8-pin | 1 |

***Block Diagram***

***Part I***

Input Switches

PortA

PortB

Motorola Based Microcomputer

***Part II***

PortA

PortB

Motorola Based Microcomputer

Input Switches

***Microcomputer Block Diagram***

2732 Even

68000

D8-D15

555 Timer based

Reset Circuit

A1-A12

6821

Odd

2732 Odd

D0-D7

D0-D7

***Schematic***

***Schematic***

******

***Memory Map***

2732 Even

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A23 | A22 | A21 | A20 | A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | All 0's to 1's | | | | | | | | | | | | 0 |

$000 000, $000 002, $000 004, $000 006, …, $001 FFE

2732 Odd

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A23 | A22 | A21 | A20 | A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | All 0's to 1's | | | | | | | | | | | | 1 |

$000 001, $000 003, $000 005, $000 007, …, $001 FFF

***I/O Map***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A23 | A22 | A21 | A20 | A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | Register Selected |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Port A |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | DDRA |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | CRA |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Port B |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | DDRB |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | CRB |

**Port A: $002001**

**DDRA: $002001**

**Port B: $002005**

**DDRB: $002005**

**CRA: $002003**

**CRB: $002007**

***Software***

***LED***

*The Code for the Led:*

PORTA EQU $002001

DDRA EQUPORTA

CRA EQU $002003

ORG $0

DC.L $00010000 ; A7'

DC.L $00000100 ; PC

ORG $100

BACK BCLR.B #2, CRA ; Set bit for DDR access

MOVE.B #2, DDRA ; Set bit 1 as input

BSET.B #2, CRA ; Clear bit for Port access

MOVE.B PORTA, D0 ; Pull PortA into D0

LSL.B #1, D0 ; Left-shift to move switch bit to LED bit

MOVE.B D0, PORTA ; Move results to PortA

BRA BACK ; Loop indefinitely

FINISH JMP FINISH

***Compiled Code***

***LED***

***even/odd split:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Even** |  | **Odd** |  |
| $000 | 00 | $000 | 01 |
| $001 | 00 | $001 | 00 |
| $002 | 00 | $002 | 00 |
| $003 | 01 | $003 | 00 |
| $080 | 08 | $080 | B8 |
| $081 | 00 | $081 | 02 |
| $082 | 20 | $082 | 03 |
| $083 | 11 | $083 | FC |
| $084 | 00 | $084 | 02 |
| $085 | 20 | $085 | 01 |
| $086 | 08 | $086 | F8 |
| $087 | 00 | $087 | 02 |
| $088 | 20 | $088 | 03 |
| $089 | 10 | $089 | 38 |
| $08A | 20 | $08A | 01 |
| $08B | E3 | $08B | 08 |
| $08C | 11 | $08C | C0 |
| $08D | 20 | $08D | 01 |
| $08E | 60 | $08E | E2 |
| $08F | 4E | $08F | F8 |
| $090 | 01 | $090 | 1E |

***Software***

***Seven Seg***

*The code for the seven segment display:*

PORTA EQU $002001

DDRA EQU PORTA

CRA EQU $002003

PORTB EQU $002005

DDRB EQU PORTB

CRB EQU $002007

ORG $0

DC.L $00010000 ; A7'

DC.L $00000100 ; PC

ORG $200

DC.B $40; 0

DC.B $CF; 1

DC.B $24 ;2

DC.B $30 ;3

DC.B $19 ;4

DC.B $12 ;5

DC.B $02 ;6

DC.B $78 ;7

DC.B $00 ;8

DC.B $18 ;9

ORG $100

BACK BCLR.B #2,CRA ; clear bit 2 for DDRA access

MOVE.B #0,DDRA ; make PortA as an input

BSET.B #2,CRA ; set bit for Port access

BCLR.B #2,CRB ; clear bit 2 for DDRB access

MOVE.B #$FF,DDRB ; configure port B as output

BSET.B #2,CRB ; set bit for Port access

CLR.W D0 ; clear the content of D0

CLR.W D1; clear the content of D1

LEA.L $200,A0 ; initialize pointer A0

MOVE.B PORTA,D0 ; Pull PortA into D0

MOVE.B (A0,D0.W),D1 ; add content of Do and content of $200 , move that content to

;D1

MOVE.B D1,PORTB ; OUTPUT TO PORTB

BRA BACK ; Loop indefinitely

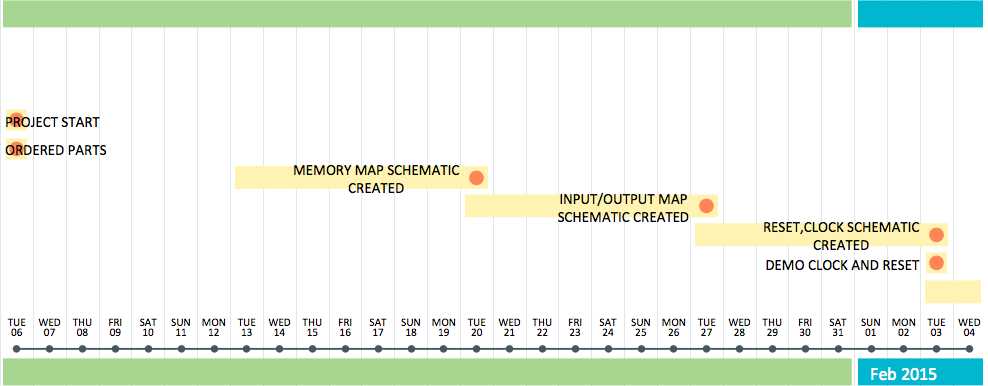
***Compiled Code***

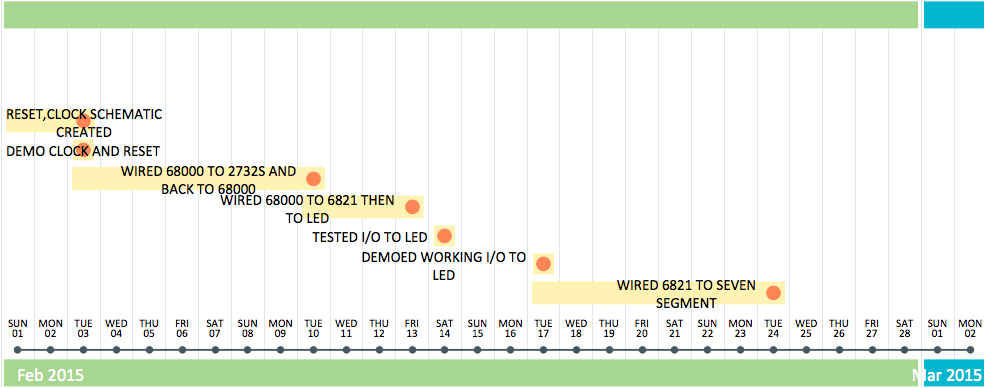
***Seven Segment***

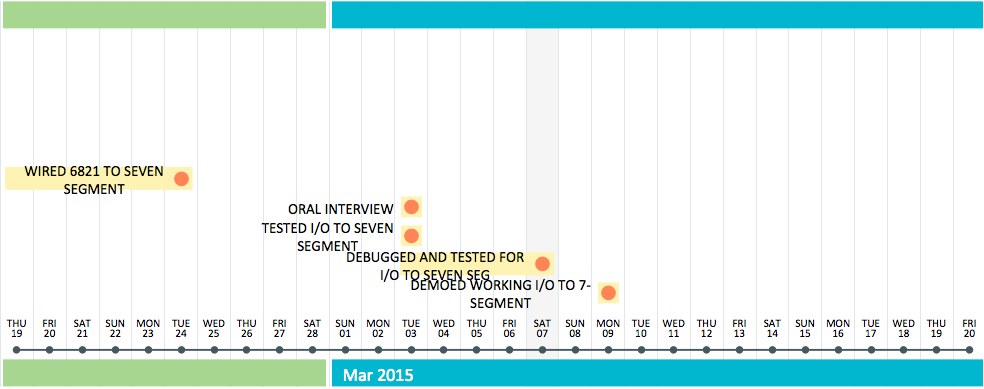
***even/odd split:***

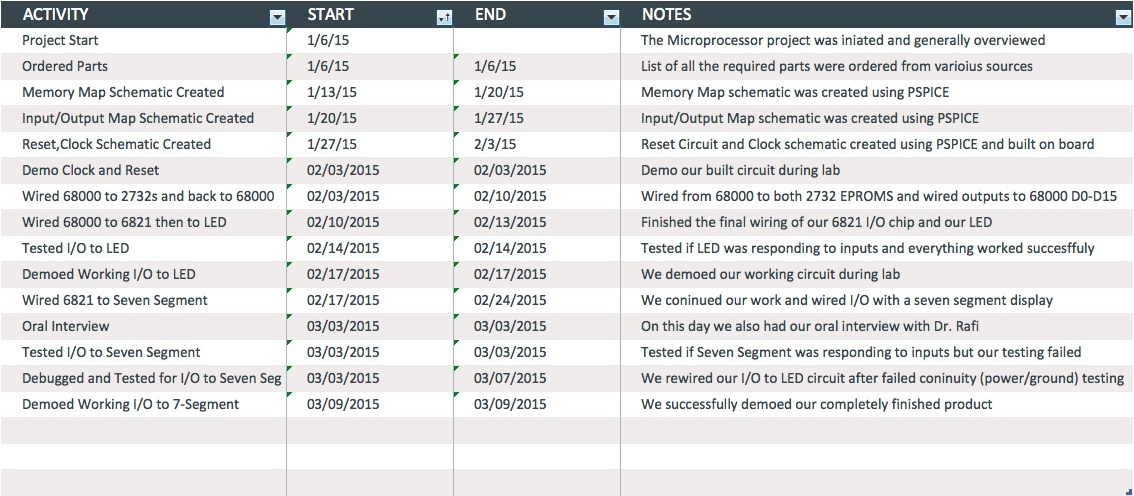
|  |  |  |  |
| --- | --- | --- | --- |
| **Even** |  | **Odd** |  |
| $000 | 00 | $000 | 01 |
| $001 | 00 | $001 | 00 |
| $002 | 00 | $002 | 00 |
| $003 | 01 | $003 | 00 |
| $080 | 08 | $080 | B8 |
| $081 | 00 | $081 | 02 |
| $082 | 20 | $082 | 03 |
| $083 | 11 | $083 | FC |
| $084 | 00 | $084 | 00 |
| $085 | 20 | $085 | 01 |
| $086 | 08 | $086 | F8 |
| $087 | 00 | $087 | 02 |
| $088 | 20 | $088 | 03 |
| $089 | 08 | $089 | B8 |
| $08A | 00 | $08A | 02 |
| $08B | 20 | $08B | 07 |
| $08C | 11 | $08C | FC |
| $08D | 00 | $08D | FF |
| $08E | 20 | $08E | 05 |
| $08F | 08 | $08F | F8 |
| $090 | 00 | $090 | 02 |
| $091 | 20 | $091 | 07 |
| $092 | 41 | $092 | F8 |
| $093 | 02 | $093 | 00 |
| $094 | 42 | $094 | 40 |
| $095 | 10 | $095 | 38 |
| $096 | 20 | $096 | 01 |
| $097 | 12 | $097 | 30 |
| $098 | 00 | $098 | 00 |
| $09A | 11 | $09A | C1 |
| $09B | 20 | $09B | 05 |
| $09C | 60 | $09C | C8 |
| $100 | 40 | $100 | CF |
| $101 | 24 | $101 | 30 |
| $102 | 19 | $102 | 12 |
| $103 | 02 | $103 | 78 |
| $104 | 01 | $104 | 18 |

***Timeline***







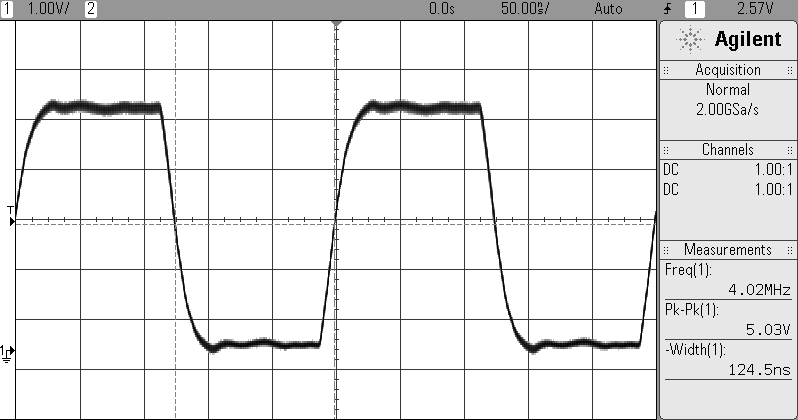
***Timeline Description***

***Discussion***

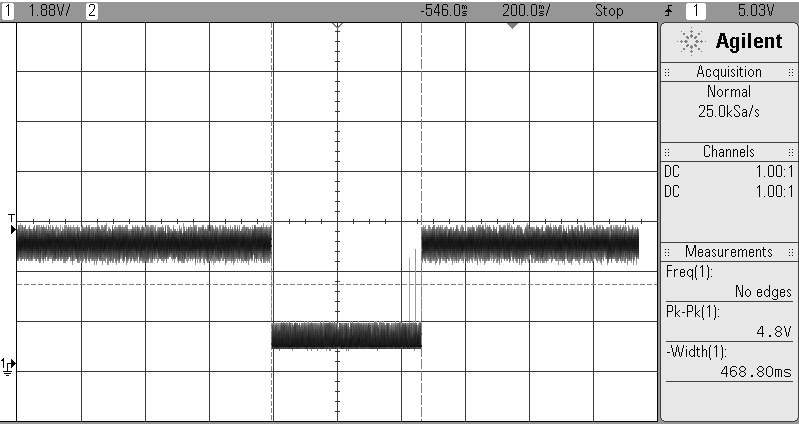
Unfortunately throughout this entire project we ran into various troubleshooting issues. The first began with the reset circuitry. The problem was that we had a lot of noise being picked up by the oscilloscope when we pushed down the reset button. We were not sure what exactly was causing this issue. At first we believed that it had to do with the extra length of wires, but we then realized that our wires were not wire wrapped properly. This mistake was due to our inexperience with wire wrapping circuits. We rewired the reset circuitry and were able to accurately display a clock frequency of 4 MHz and a reset pulse greater than 100 milliseconds.

The images captured from the oscilloscope are displayed below:

***Clock:***

******

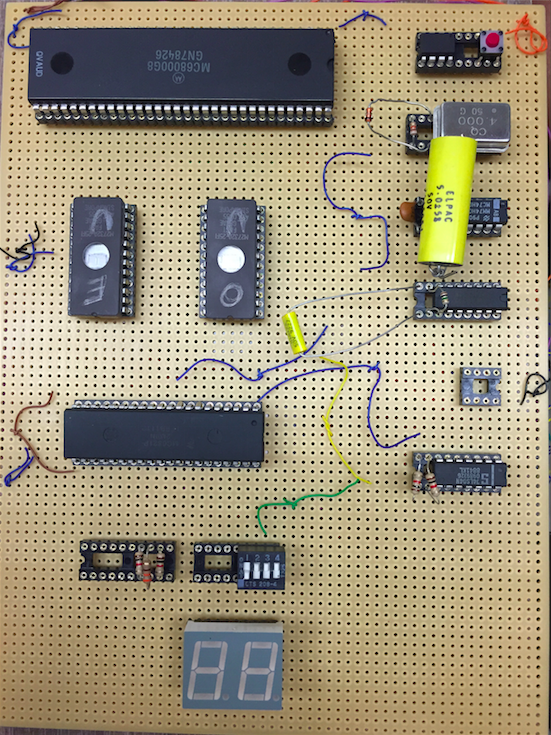
***Reset Pulse:***

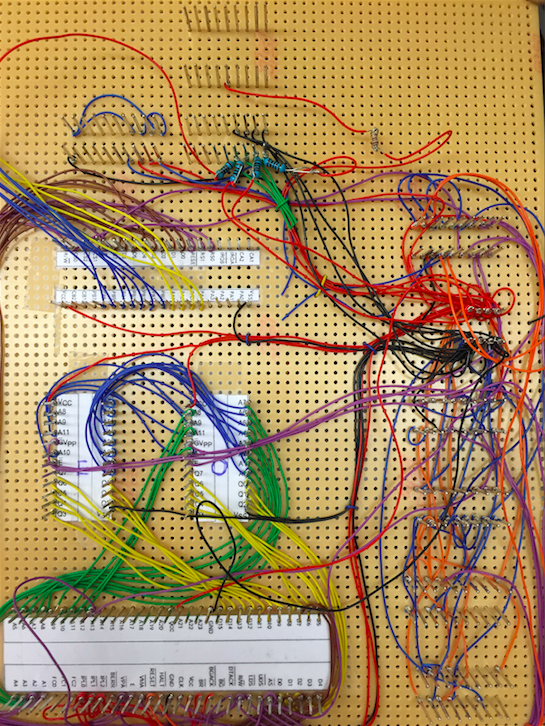


Another issue that we later encountered when we interfaced the input switches with the Seven Segment Display was that the Seven Segment was not responding accordingly to our switch values. While observing this malfunction we noticed that our 555 timer was becoming extremely hot. We knew this was not normal so we expected there to be some issue with our wires not being connected properly. With this in mind we followed by testing our circuit for continuity. Surprising, all our pins passed their individual continuity tests. We did not know where to go from here, so we proceeded by changing our 555-timer chip with a new one. This helped because our 555 timer was no longer heating up. Regardless of this, our Seven Segment was still not responding to our switch values. In desperation we modified the code we had coded into the eproms. Unfortunately this did not work either and ultimately due to the lack of understanding at the time we did know how to further progress through our debugging. In the end we decided that we would more diligently wire the seven segment and switch onto a breadboard rather than have it wire-wrapped with the rest of our microcomputer. Once we finished rewiring the circuitry and tested our input and output interfacing with the Seven Segment the Seven Segment responded accordingly. This means that there was something had not been initially wire-wrapped correctly. We proceeded with our testing while the seven segment and switch were wired onto the bread board. After successfully testing the I/O functionality of our microcomputer we went back and wired the seven segment and switch onto the wire wrap board

***Conclusion***

In the conclusion, after much troubleshooting and testing we were able to successfully make our project work. We believe that our microcomputer was able to work successfully because of our ambition to be successful in this class and learn from all the resources we were provided. Ultimately this was a very enriching lab and we hope to continue expanding our knowledge of microprocessors by continuing our studies outside of lecture and through possible internship opportunities.

Below are images of all our hard work and the work done by each member in the group.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Duy Pham Duc** | **Jonathan Plata** | **Jose Ventura** | **Brian Nguyen** | **Josue LLamas** | **Beshoy**  **Metry** |
| Wire wrap | Wire wrap | Wire wrap | Wire wrap | Wire wrap | Wire Wrap a little |
| Driving out to buy extra parts required  EPROM programming | Building Lab 1 and Lab 2 Circuit implementation | Debugging and  Continuity Testing | Building Lab 1 and Lab 2 Circuit  Implementation | Debugging and  Continuity Testing | Eprom Programming and Debugging |
| Written final report –  Memory Map, I/O Map, | Written final report –  Problem Definition | Written final report –  Introduction,  Pictures | Written final report –  Summary,  Block Diagram | Written final report –  Discussion & Conclusion | Written final report –  Software  Eprom (Even and ODD) |