ECE 343L-01

Microprocessor I Lab

Dr. Mohamed Rafiquzzaman

Winter 2016

Motorola 68000 Microprocessor

March 9, 2016

Group 10

Brad Ko

Dillon Smolinski

Introduction:

The Motorola 68000 microprocessor was used for this lab project because it is very inexpensive in today’s standards. This means that as students it is a very cost-effective way to start to delve into the world of microprocessors. Its assembly language is straight-forward and easy to understand while being very efficient in being able to do instructions that uses byte, word, and long word sizes.

Along with the 68000 being inexpensive, the 6821 peripheral chip was used for a cheap, effective way to implement memory-mapped programmed I/O. The 68230 was another peripheral chip that could be used but it was cheaper to use the 6821 instead though it would be a lot easier to implement. This design prioritizes a low cost over ease of implementation. Using the slower and older 6821 allowed us to understand how to interface the 68000 with dated peripheral chip being compatible with older products. This involves the 6821 in using the E clock (which is 1/10 the 68000 clock), (valid memory address), and (valid peripheral address) to ensure that the slower 6821 correctly interfaces with the 68000. We could have used another 6821 chip by utilizing , but it was unneeded because 16 I/O pins would suffice for our purposes.

For our memory, we used two EPROMs to address even and odd memory locations using to choose even memory locations and to choose odd memory locations. These EPROMs are inexpensive as they are small and dated which are perfect for the small programs used in the lab. Because we did not use the maximum amount of memory the 68000 could handle, we could use linear addressing by using an unused address pin to distinguish between the 6821 and our EPROMs. The unused A13 pin allowed us to easily use memory-mapped locations where A13 = 0 selected the EPROMs and A13 = 1 selected the 6821.

Software:

There is no A0­ pin on the 68000 and so it must be obtained from and to separate the even and odd memory locations respectively. Thus, all the memory locations must be shifted to the right by one bit because the A0 pin does not exist and new address locations must be found.

Lab 1:

ORG 0

DC.L $00506080

DC.L $00000100

CRA EQU $2003

DDRA EQU $2001

PORTA EQU DDRA

ORG $000100

BACK BCLR.B #2,CRA ;access DDRA

MOVE.B #2,DDRA ;PORTA bit 0 = input, bit 1 = output

BSET.B #2,CRA ;access PORTA

MOVE.B PORTA,D0 ;move inputs to D0

LSL.B #1,D0 ;align input into output

MOVE.B D0,PORTA ;move aligned D0 to PORTA

BRA BACK

Memory location:

$000100 🡪 $000080

Lab 2:

ORG 0

DC.L $00506080

DC.L $00000100

CRA EQU $2003

DDRA EQU $2001

PORTA EQU DDRA

CRB EQU $2007

DDRB EQU $2005

PORTB EQU DDRB

ORG $000300

DC.B $7E ;0

DC.B $30 ;1

DC.B $6D ;2

DC.B $79 ;3

DC.B $33 ;4

DC.B $5B ;5

DC.B $5F ;6

DC.B $70 ;7

DC.B $FF ;8

DC.B $7B ;9

DC.B $00 ;-A

DC.B $00 ;-B

DC.B $00 ;-C

DC.B $00 ;-D

DC.B $00 ;-E

DC.B $00 ;-F

ORG $000100

BACK BCLR.B #2,CRA ;access DDRA

MOVE.B #$00,DDRA ;all bits are inputs

BSET.B #2,CRA ;access PORTA

BCLR.B #2,CRB ;access DDRB

MOVE.B #$FF,DDRB ;all bits are outputs

BSET.B #2,CRB ;access PORTB

LEA.L $000300,A0 ;move look-up table address to pointer

CLR.L D0 ;clear D0

MOVE.B PORTA,D0 ;move inputs to data register

ANDI.W #$00FF, D0 ;zero extend

MOVE.B (A0,D0.W),PORTB ;move byte from look-up table

BRA BACK

Memory location:

$000100 🡪 $000080

$000300 🡪 $000180

Discussion and Conclusions:

This lab was very hands-on and allowed us to work directly with what we were learning in the lecture. Everything that was connected was done so for a reason and not “just because.” This allows us to fully understand why each connection is connected to the 68000 or what external logic is used to achieve a certain output.

The first problem during the lab was when we were trying to implement the first project. When we first wire-wrapped the board, we only used one color to connected everything. We could not easily see which connections were made as there were no pin indications on the backside. To alleviate this problem, we redid the entire wire-wrapping to ensure that it would be easier to troubleshoot in the future. We added pin names and numbers on the back so it is easier to see which pin is going where. We also used different colored wires to indicate different connections and made the wire-wrapping a lot cleaner. However, even after doing this we were still unable to implement the first project and proceeded to do continuity tests.

After continuity checking the schematic three times and seeing that everything was correct, I proceeded to compare my schematic with others who have got it working. My schematics did match their schematics so it was not that I was wiring the wrong pins. We continuity tested the board three more times and eventually realized that the EPROM was programmed incorrectly. One byte in the code was overlooked which made the some of the even instructions go to the odd EPROM and vice versa. After fixing this problem, the first lab was complemented smoothly and without any further problems. The second project was completed even smoother than the first and was implemented on the first try.