**Lab 4:**

**SRAM, LCD, Keypad and The Mortgage Calculator**

**Bruce Oshokoya**

**Jonathan Peevy**

**Natalie Morningstar**

CMPE 310: System Design & Programming

University of Maryland, Baltimore County

December 13, 2013



**Outline**

1. **Introduction -** Natalie
   1. Background
   2. Problem Description
   3. Goals and Specifications
2. **System Architecture Section -** Bruce
   1. Possible Approaches
   2. Test Plan \*Dr S. says it is an important part of the paper
3. **Hardware System Design Section** - Bruce
   1. System Clock
   2. CPU
      1. 8086
      2. Buffers
      3. Latches
      4. Decoders
      5. (?)
      6. (?)
      7. (?)
   3. Memory
      1. EEPROM
      2. SRAM
      3. CMOS
      4. (?)
   4. Input/Output
      1. Keyboard
      2. LCD Display
4. **Software System Design Section -** John
   1. Initialization
   2. Programming 8255, 8259, 8279
   3. Interrupts
   4. Functionality
   5. User Interface
5. **Experimental Results -** Natalie & John
   1. System Clock Tests
   2. CPU Tests
   3. Memory Tests
      1. EEPROM
      2. SRAM
   4. Software Testing
      1. Case Testing
      2. Edge Testing
6. **Conclusions -** John
7. **References -** Natalie
8. **Appendix -** Natalie
   1. 8086 Based System
      1. Clock Generator
      2. CPU, EEPROM and SRAM
   2. UMBC 8086 Trainer System
      1. 8086
      2. CMOS
      3. SRAM
      4. Clock Generator
      5. 82C55
      6. 8259
      7. 8254
      8. 8279
      9. 16550
      10. LCD
      11. 7-Segment Display
      12. Diodes
      13. DIP
   3. Mortgage Calculator Code

This Address/Peripheral table will probably go in report somewhere, but it is not final

|  |  |
| --- | --- |
| **Address** | **Peripheral** |
| FFFFh | PP1 Control Register |
| FFFEh | PP2 Control Register |
| FFFDh | PP1 address 1 |
| FFFCh | PP2 address 1 |
| FFFBh | PP1 address 2 |
| FFFAh | PP2 address 2 |
| FFF9h | PP3 address 3 |
| FFF8h | PP3 address 3 |
| FFF7h | PP3 Control Register |
| FFF6h | Interrupt Controller Command (8259) |
| FFF5h | PP1 address 1 |
| FFF4h | Interrupt Controller Data (8259) |
| FFF3h | PP1 address 2 |
| FFF2h | Keyboard Command (8279) |
| FFF1h | PP1 address 3 |
| FFEFh | Keyboard Data (8279) |
| FFEDh | UART address 8 |
| FFEBh | UART address 7 |
| FFE9h | UART address 6 |
| FFE7h | UART address 5 |
| FFE7h | UART address 4 |
| FFE5h | UART address 3 |
| FFE3h | UART address 2 |
| FFE1h | UART address 1 |
| FFDEh | Counter Command (8254) |
| FFDCh | Counter 2(8254) |
| FFDAh | Counter 1 (8254) |
| FFD8h | Counter 0 (8254) |
| FFD6h | LCD address 4 |
| FFD4h | LCD address 3 |
| FFD2h | LCD address 2 |
| FFD0h | LCD address 1 |
| FFCFh | 7-Segment Display 2 |
| FFCEh | 7-Segment Display 1 |
| FFCCh | Diodes |
| FFCAh | DIP Switches |

**Introduction**

The system clock and central processing unit implemented in this project is phase one of a larger more complicated system. For phase one, a system clock, comprised of a 4MHz crystal and 82C84 clock generator, and a central processing unit (CPU), comprised of a 16-bit microprocessor namely the one 8086 microprocessor, three 74LS373 latches and two 74LS245 buffer. Phase two in this project will implement everything we did in phase one with additional components. For phase two, a 74LS244 buffer, some non-volatile memory, which are two 28C256 32kx8 EEPROM’s, a 7400 NAND and a 7432 OR. It is predicted that in phase three, the system of this project will have a user interface incorporated into it. Keeping the final goal in mind is useful in understanding the level of success of phase one. For instance, all inputs and outputs of the system clock must be correct so that the 8086 microprocessor can be driven. The microprocessor multiplexed buses must be correctly demultiplexed and then the buffers must be implemented correctly so that when the system became potentially larger in phase two it met the timing requirements. All of this has to working correctly in order to successfully implement a user interface to control the microcontroller.

* **Background**

An embedded system is a constructed architecture of smaller electrical computer components designed to perform a specific function in a bigger mechanical/electrical system. Within certain embedded systems there consist microprocessors; which are a single IC that accepts coded instructions that manipulates data controlling memory and input/output. Another component that could be used in the in an embedded system are latches. Latches and buffers like microprocessors are also a single IC that has different inputs and outputs that on it that will be explained later in the report. All of these components can be connected with wires in such a way that you could have it perform a simple task.

* **Problem Description – 8086 Based System**

For phase one the problem was to construct a standalone embedded microprocessor system with a basic setup that will in the future be able to run software from memory. The CPU uses time multiplexing for address, data, and some status lines. The CPU will require it to be attached to a clock generator that provides it with clock, rest logic and ready logic. Use an oscilloscope and multimeter to demonstrate that the microprocessor operates properly. Now for phase two we are suppose to add to the microprocessor system, which is suppose to be capable of running software from non-volatile memory. Software will need to be written in a way in which its effect on the microprocessor system should be able to perform detailed timing analysis to show that the information is being read from the EEPROM by the CPU. All this needs to be displayed on an oscilloscope to verify accuracy. Phase three…

* **Problem Description for the UMBC 8086 Trainer system.**

The second assignmen in this lab was to design the schematics for the UMBC 8086 Trainer system. The exact specifications are as follows:

1. 8086 in minimum mode and its associated bus buffering/demultiplexing logic. NMI should be connected to a push-button switch with appropriate timing logic. The data bus, address bus and all control signals should be connected to headers after buffering and demultiplexing for external access to design expansion boards.
2. 256KB of CMOS flash composed of 128K x 8 28F010 CMOS Flash Memory, decoded into two banks, highest address FFFFFH.
3. 128KB of SRAM composed of 32K x 8 CY7C199 Static SRAM, decoded into two banks, lowest address 00000H.
4. 8284A Clock Generator along with crystal and reset sub circuits
5. 3 8255 chips, decoded at the following addresses (all in hex), all port connections should be pulled to headers for external access.
   1. PPI 1: Port addresses FFFF (control register), FFFD, FFFB, FFF9
   2. PPI 2: Port addresses FFFE (control register), FFFC, FFFA, FFF8
   3. PPI 3: Port addresses FFF7 (control register), FFF5, FFF3, FFF1
6. 1 8259 decoded at FFF6 (command) and FFF4 (data). IR0 connected to a push button switch with necessary circuits, IR1 connected to 8254 counter 2 output, IR2 connected to the 8279 IRQ output and IR3 connected to the 16550 INTR output. All other IR lines connected to headers for external access.
7. 1 8254 decoded at FFDE (command), FFDC, FFDA, FFD8, with all counter 1 and counter 3 pins connected to headers. Counter 2 output connected to IR1 of the 8259, gate and clock con- nected to header.
8. 1 8279 decoded at FFF2 (command) and FFF0 (data) which will be operated in the decoded mode. Connect 20 push button switches connected as four rows and five columns. Also con- nect two switches to the control and shift inputs. In your layout arrange these 22 keys as well as reset, switch connected to the IR0 input on the 8259 and NMI keys to form a 5x5 keyboard matrix layout. CLK input to the 8279 should be the PCLK signal from the 8284A.
9. 1 16550 UART decoded in the high bank at odd port address from FFEF to FFE1. The clock input to the UART should be PCLK from the 8284A and the INTR output should be con- nected to IR3 of the 8259. Make the connections for serial port using a MAX 235 line driver/ receiver and a DSUB-9 connector. Enable all receiver inputs on the MAX 235.
10. 1 20 character x 4 line LCD display with no back-light and an integrated LCD controller decoded at addresses FFD6, FFD4, FFD2 and FFD0.
11. 2 common-anode 7-seg LEDs with a decimal point segment connected to 74374 latches decoded at FFCE and FFCF, 8 LEDs connected to a 74374 latch decoded at FFCC.
12. 8 DIP switches connected to a 74244 decoded at FFCA.
13. A power terminal block to provide power to the board. A 100uF decoupling capacitor should be connected next to the power terminal block. Place a 0.1uF decoupling capacitor next to each of the major chips in your project.
14. Other misc. components which are required for the above circuits e.g. resistor packs, caps, etc. You can use discrete gate ICs, 3-to-8 decoder, 2-to-4 decoder or 16L8 (preferable) for decoding various addresses. Include the 16L8 programs if you decided to use them for decoding. Consider using the least number of chips required to perform IO and memory decoding.

The 12 schematics for this UMBC 8086 Trainer system are in the appendix

* **Goals and Specifications**

The first goal of the project would be to successfully design and create a schematic that would interface the 8086 microprocessor, the latch’s, EEPROM, AND gates, OR gates and the buffers. The next goal would be to take what was done on the schematic and translate it to a beadboard. This will require that the breadboard would be set up in an organization that would be most efficient when connecting the components on the breadboard to each other. Right after that has been done then a program needs to be written and placed on the EEPROM. Then after that is completed and all the components are connected and hooked up to a power source as well as an oscilloscope the next goal would be to have the embedded system give out the correct readings on the oscilloscope.

**The System Architecture Section**

The top level of the system that was constructed is focused on the 8086 microprocessor. The 8086 microprocessor is the brain of the system. It controls the 74LS373 latch and the 74LS245 buffer. The 8086 may be the brain but the 28C256 EEPROM’s is the memory for the 8086. These directly affected what the 8086 processor would do. In the beginning stages of this endeavor it was unknown what would be the best approach to design the embedded system. There are of total of eight different components that make up the system, the 74LS373 latch, the 74LS245, the 8086 microprocessor, a clock generator, the 28C256 32kx8 EEPROM, the 74LS244 buffer, the 7400 AND, and the 7432 OR. The goal of the project was to interface these devices in a configuration that supported correct functioning of the processor, verified by checking signals with an oscilloscope. The problem was how to organize these IC’s efficiently on a circuit board.

* **Possible Approaches**

Since the problem as described included a design for how components should be connected, the only thing left to plan was the configuration of the IC’s physically on the board that would assist us in completion of the lab successfully. One possible approach was to somehow to compact the clock generator as much as possible and put it one corner of the circuit board. This seemed logical at first but it was later determined that it would be problematic trying to understand what was going on the circuit board. For starters the clock generator would be too compact to properly function. However even if it would function correctly it would be entangled with the wires from the other part of the board. When there are too many wires on a breadboard error debugging becomes nearly impossible.

Another problem that had to be tackled was how to organize the rest of the breadboard. Since the clock generator would be located elsewhere, how would the 8086 microprocessor, the 3 latch’s and 3 buffers, logic gate IC’s and the EEPROM’s be organized that would be the most efficient when wiring them up. If it is not organized well then it is easy to lose track of the wires and make a mistake, making debugging time-costly. Originally the three different sections of IC’s were going to be lined up vertically across the circuit board so each section would be together. However it was decided that other options would be better suited to the design, due to the fact that there would be a lot of bridging as a result. When organized in a not so efficient way then there is a possibility that more long wires will be used and the more bridging that would happen. Unfortunately this problem only got worse after phase one was finished and phase two needed to be implemented it. Phase one took up so much space on the board it meant that phase two would require a lot more thought.

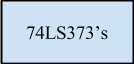
* **Selected Approach**

For phase one it was decided that in order to efficiently construct the system, two circuit boards were needed. One circuit board would contain just the clock generator, while the other would contain the rest. On the non-clock generator board the IC’s were grouped together by importance. The most important IC’s needed to be right underneath the 8086 microprocessor. It was agreed upon that the IC that was interfaced with the most was the 74LS373’s. That was placed right underneath the 8086 while the 74LS245’s was placed to the right of it. Then for phase two it was decided that the 74LS244 buffer had to be in between the two EEPROM’s due to needed wiring of the two components. As a result the three components were placed at the empty space right of the 74LS245’s. That left room for the only components left which were the 7432 OR and 7400 AND IC. So those components were placed right underneath the 74LS245’s. This arrangement gave the most room for the clock generator and minimized the amount of bridging of wires. The set up is shown in the in Figure 1 below.





**Figure 1: Implementation Approach**

* **Test Plan**

In order to make sure the finished product runs smoothly a plan will be needed for debugging the UMBC 8086 hardware and the software.

* + ***Hardware***

The hardware that will be used for the finished product is the UMBC 8086 Trainer System. However it is required that a way needs to be made in which our finished product will work in the integrated system that was talked about earlier in the report. In order for this to be possible a schematic needs to be done for both the UMBC Trainer Board and the original board. The goal of the schematic will be to first wire it up there with the logic that makes sense. After both schematics are done then the next step would be to load the program and then determine if it is a software issue or is it a hardware issue. Setting the plan to test the hardware this way also determines if anything was overlooked in building the schematic. Then if it is determined that it is a hardware issue we will go back and change the schematic. It’s from the schematic that we can determine how to program the code correctly.

* + ***Software***

In order for the software to work correctly it has been split into 5 different versions. Splitting into 5 different versions will allow for incremental testing as well for efficient debugging.

1. **Lab4v1.asm**

Labv1.asm is the first version of the code that will be tested. It has only one function, it test displaying on the LCD. The code displays a welcome message that reads:

Welcome to CMPE 310

Fall 13

Trainer Board

8086 - Project 4

Testing the program to see if it can display to the LCD screen is important in the development of the product because it tests the boundaries of the screen that will be displayed. A restriction was given that the screen can only show 20 characters per line and a total of four lines can be shown at a time. This display is the only verification that our code works correctly. In conclusion having knowledge of the medium that will be worked with is very important.

1. **Lab4v2.asm- (Includes IC)**

This is the version is which the IC’s that will be used in the project will actually be programmed.

1. **Lab4v3.asm- (Includes interrupts)**

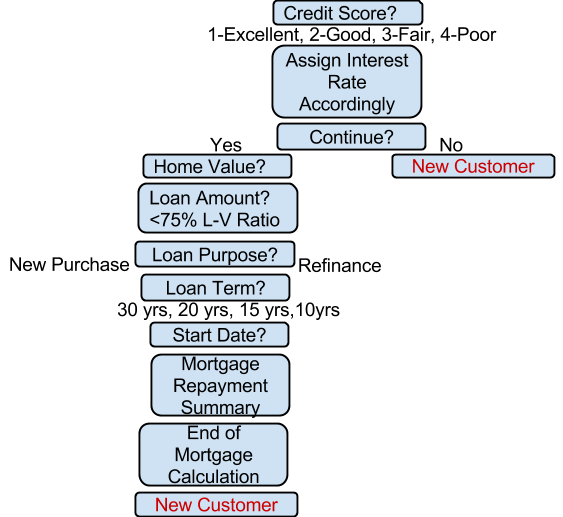
In this version of the code all the interrupts will be programmed so the 8086 will know when to stop what it is doing and do a requested function.

1. **Lab4v4.asm- (Includes functionality)**

This is the version of the code in which we will actually be programming the Mortgage calculator. The program may or may not be set up to print to the LCD however it will be programmed to do the correct calculations.

1. **Lab4v5.asm- (Includes user interface)**

This is the final version of the program and this part definitely includes printing to the LCD in the right format restrictions. This part also includes a lot of polishing of the code to make it run faster et cetera.



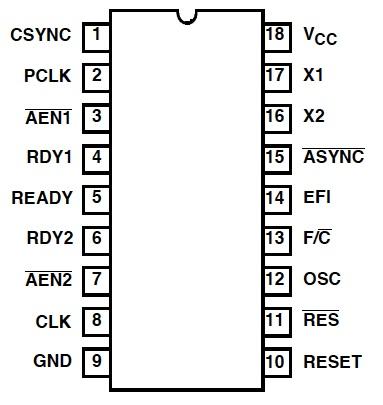
**The Hardware System Design Section**

There are eight types of IC’s that this project will utilize: the 8284 clock generator, the 8086 microprocessor, the 74LS373 octal transparent latch, the 74LS245 bidirectional buffer, the 74LS244 unidirectional buffer, the 7432 OR gate, the 7400 NAND gate, and the 82C256 EEPROM. The main new component in this lab is the EEPROM, as interfacing this device is the main objective of the lab. The other new components (NAND gate, OR gate, and unidirectional buffer) are used to generate control signals for the EEPROM functionality.

* **System Clock**

The system clock of the CPU is generated by the 8284 clock generator and 4 MHz crystal. The RC circuit and switch are used to safely reset the CPU. In general, a system clock is an essential component in computing. It is best described as the heartbeat that allows all devices to be synchronous. Without a clock, it would be impossible to create synchronous signals.

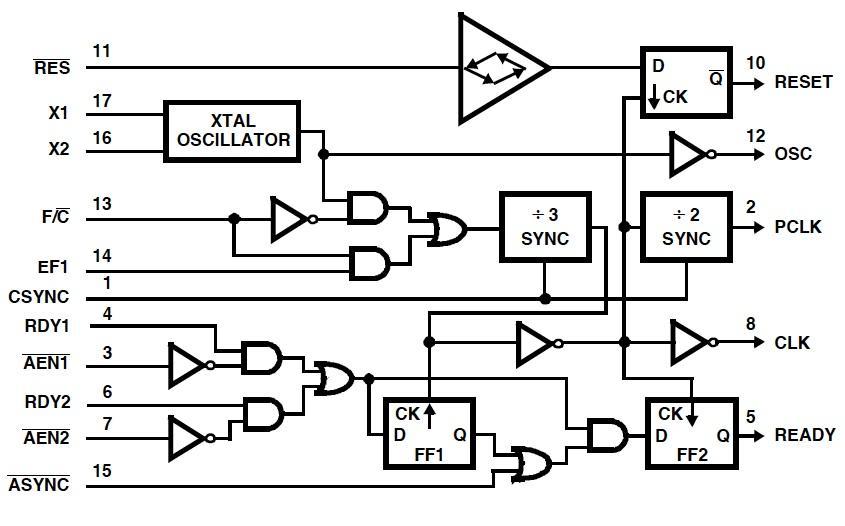
The clock generator's primary functions are clock outputs, reset synchronization, ready synchronization, oscillator output and clock synchronization. Each of these primary functions have designated pins that can be seen in Figure 2 below.



**Figure 2: 82C84 Clock Generator**

**Pin Configuration**

The clock outputs for this IC are CLK and PCLK. The CLK output drives the the CPU and in this case, the 8086 CLK input. The duty cycle of the CLK output is 33% and the duty cycle of the PCLK is 16.6%. The PCLK is a peripheral clock signal used to drive peripheral devices in the system. The clock rate at which the CPU requires to be operated will be explored further in the Experimental Results section. Proper timing constraints must be met in order to reset the CPU. This is done with the RESET synchronization, which is partially implemented externally with an RC circuit. In order for the CPU to be reset, RESET must be held for at least 50 microseconds. READY synchronization is used to select between RDY1 and RD2 or both. These ready signals control insertion and the wait states of the CPU. The oscillator or OSC as seen as pin 12 in Figure 2 is predominantly used so that other systems can derive a control, namely the buffered crystal. There are two standard clock synchronization methods on the 8284. The two options available are defined through the frequency/crystal select as seen on Figure 2 on pin 13. In this system, the frequency/crystal select is grounded so that clock synchronization can be enabled by the crystal and not an external frequency input (EFI). The schematics for the completed system clock can be seen in Appendix A: System Clock. Both functionalities of the RESET synchronization and clock synchronization can be seen in Figure 3. The top half of Figure 3 details the clock synchronization and the bottom half details the RESET synchronization.



**Figure 3: 82C84 Clock Generator Functional Diagram**

The results of the physical demonstration validate all the above primary functions of the system clock. Given that the system clock is functioning properly, it will drive the CPU, which will then drive all other primary functions of the system.

* **Central Processing Unit**

The main components of this systems central processing unit are Intel’s 16-bit 8086 microprocessor, 74LS373 latches and 74LS245 buffers. Each component is dependent on the previous functioning correctly. The 8086 and latches/buffers form the complete CPU unit and are unaltered from the successful previous lab.

* + **Intel 8086 Processor**

The Intel 8086 is at the core of the system. Whereas other components make possible the functioning of the 8086, or other functions, the 8086 in itself is the brain: able to communicate with components via I/O, as well as process data, and control the system as a whole.

The 8086 has two modes of operation, each having a slightly different pin-out scheme. In MIN mode, the processor is allowed full control of the I/O bus. This is the more simple of the two operation modes, but is only for use with a single processor. MAX mode enables the ability to have multiple processors sharing the same bus. This is achieved through the use of a HOLD-type interrupt system. The processor requiring use of the bus sends a lock signal to the coprocessors (all running in MAX mode), which causes the coprocessors to effectively detach themselves from the bus, giving bus use to the original processor until it has finished its use of the bus. For the purposes of this lab, MIN mode is sufficient, due to requiring only one processing unit.

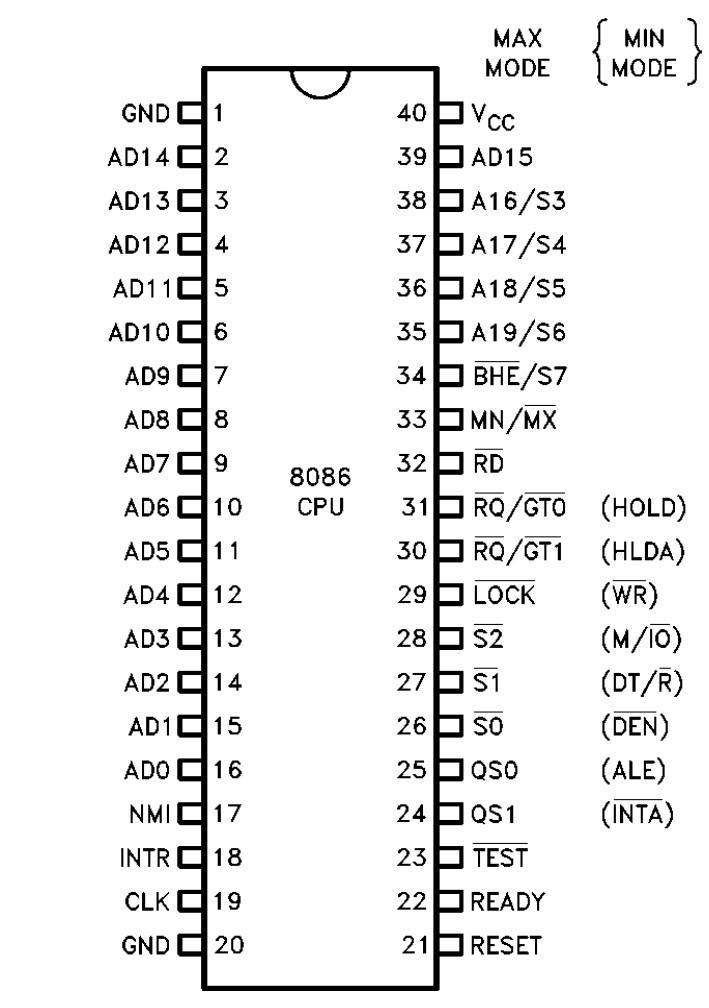
In MIN mode operation, the 8086’s pins can be grouped into three interfaces: the Memory/IO interface, the Interrupt interface, and the DMA interface. Each interface suits a purpose, with IO being responsible for the communication of the processor with other components. The interrupt interface handles hardware and software interrupts to the processor, while the DMA interface handles processing holding, in which the processor “detaches” from the bus, giving control to other devices.

The IO interface consists of all the address/data bus (AD) lines connected to the processor, as well as the data control lines. The bus lines are initially multiplexed together to save pin room on the IC, and must be demultiplexed by the latches and buffers in order to enable use of the system. Address lines specify the address of a device to interact with, while data lines transfer information between these devices and the processor. Data control lines give more information on the data operations, such as status/read-or-write/etc.

The interrupt interface controls the operation of the processor. It includes the hardware and software interrupt pins (INTR and NMI, respectively), as well as the hard reset (RESET) coming from a push-button switch. These are used to synchronize operation of the processor with operation of other components. When a device requires processing, a hardware interrupt is deployed to the processor, causing a cease in communication on the bus until the processor has completed the task. In other situations, such as the use of “listener” software, a software interrupt may be of use. Once a software interrupt has been initiated using NMI, the interrupt by design cannot be reset until the entire system has been reset.

Finally the DMA interface deals with the operation of the processor with other components attached. Using the HOLD and HLDA pins (hold and hold acknowledge), the processor is able to logically detach from the bus and process information independently while other devices use the bus. This is a major feature in bus design and allows for efficient use of the buses in the system.

Below in Figure 4 is the pin diagram of Intel 8086 16-bit microprocessor. Later in the Experimental Results section certain signals are tested with the oscilloscope.



**Figure 4: Intel 8086 Microprocessor**

**Pin Configuration**

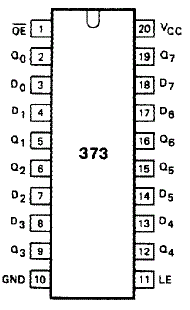
* + **Latches**

The 74LS374 that was used was manufactured using advanced Low Power Schottky technology. It is an Octal D-type Flip-Flop containing separate D-type inputs for each flip-flop. The 74LS373 has 3 different state outputs to be used in coordination with a bus. Since it is a latch IC it uses flip-flops which changes asynchronously whenever the latch enable is high. However when it is low the data is latched. Data only appears on the bus system when Output enable is set to low, but when set to high the bus output is in a high impedance state.

These are all of its features listed out:

1. Eight Latches in a Single Package
2. 3-State Outputs for Bus Interfacing
3. Hysteresis on Latch Enable
4. Edge-Triggered D-Type Inputs
5. Buffered Positive Edge-Triggered Clock
6. Hysteresis on Clock Input to Improve Noise Margin
7. Input Clamp Diodes Limit High Speed Termination Effects

Below in Figure 5 is a pin diagram of the 74L373 Latch. Later in the Experimental Results section the outputs of the Latch are tested to verify functionality.



**Figure 5: 74LS373 Latch**

**Pin Configuration**

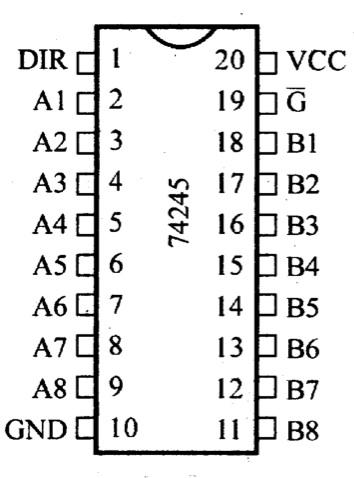
* + **Buffers**

The 74LS245 octal bus transceivers are set up in way in which it allows asynchronous two-way traffic between data buses. It allows data transmission from the A bus to the B bus or from the B bus to the A bus, depending on the direction-control input. Implemented in the IC there is a control-function which minimizes external timing requirements. Its output-enable input makes it easy to disable the IC when the buses need to be isolated.

These are its features listed out:

1. 3-State Outputs Drive Bus Lines Directly
2. PNP Inputs Reduce DC Loading on Bus Lines
3. Hysteresis at Bus Inputs Improves Noise Margins
4. Typical Propagation Delay Times Port to Port, 8 ns

Below in Figure 6 the 74S245 Buffer can be seen. Later in the Experimental Results section various pins are tested for signal output.



**Figure 6: 74LS245 Buffer**

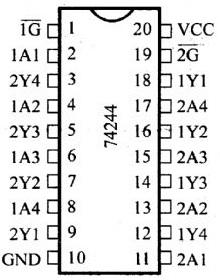
**Pin Configuration**

The 74LS244, octal buffer is designed specifically to improve both the performance and density of three-state memory address drivers, clock drivers, and bus-oriented receivers and transmitters. It is made in a way in which the designer has a choice of combinations dealing with inverting and noninverting outputs, symmetrical, active-low output-control inputs, and complementary output-control inputs. It features a high fan-out, improved fan-in, and 400-mV noise margin.

These are its features listed out:

1. Hysteresis at Inputs to Improve Noise Margins
2. 3-State Outputs Drive Bus Lines or Buffer Memory Address Registers
3. Input Clamp Diodes Limit High-Speed Termination Effects

Below in Figure 6a the 74S244 Buffer can be seen.



**Figure 6a: 74LS244 Buffer**

**Pin Configuration**

* **Memory**

Words....

* + **28C256 32K X 8 EEPROM**

The 28C256 32K X 8 EEPROM is a high performance electrically erasable and programmable read only memory. Its 256K of memory is organized as 32,768 words by 8 bits. The 28C256 is accessed like a Static RAM for the read or write cycle without the need for external components. The device a 64 byte page register to allow writing of up to 64 bytes simultaneously. During a write cycle the addresses and 1 to 64 bytes of data are internally latched , freeing the address and the data bus for other operations. Following the initiation of a write cycle, the device will automatically write the latched data using an internal control time. The end of a write cycle can be detected by DATA Polling of I/07. Once the end of a write cycle has been detected a new access for a read or write can begin.

These are its features listed out:

1. Fast read Access Time - 150 ns
2. Fast Write Cycle times
3. Low Power dissipation
4. Hardware and Software Data Protection
5. DATA polling for END of Write Detection
6. Single 5V+- 10% Supply
7. Automatic Page Write Options

Below in Figure 6b the 28C256 32K X 8 EEPROM can be seen.



**Figure 6b: 28C256 32K X 8 EEPROM**

**Pin Configuration**

* + **SRAM**

Words...

**The Software System Design**

Words...

* Initialization

Words..

* Programming 8255, 8259 and 8279

Words..

* Interrupts

Words..

* Functionality

Words..

* User Interface

Words...

**Experimental Results**

Prior to building the system, the datasheets for the IC’s were reviewed, schematic were designed and expected values were predicted. Following that procedure ensured that testing upon demonstration would be successful. Testing was divided into three main efforts, one the system clock, CPU and EEPROM. The expected results of the CPU and EEPROM were dependent completely on the system clock function properly.

* **System Clock**

The system clock was constructed from the 82C84 clock generator, crystal and simple RC circuit. In order to ensure that the system clock was function properly a set tests were implemented. Knowing that the crystal would output a clock rate of 4MHz, it was expected that the CLK output would be 1.33MHz and that PCLK would output 0.666MHz (of 666kHz). In Figures 7 and 8 both expected results are met.

|  |  |
| --- | --- |
| **Figure 7: 82C84 Pin 8 CLK** | **Figure 8: 82C84 Pin 2 PCLK** |

To confirm that the oscillator, OSC, was functioning properly, it was expected that it should look like a square wave with the same frequency of the 4MHz crystal. Samples were taken from both the crystal and OSC. Figure 9 to the left shows a clean signal sine wave with an output of 3.99MHz. The expected reading was 4MHz, but knowing that crystals are very sensitive, it was not a surprise that it was not exact. Figure 10 below is the sample OSC reading. Output rate requirement was met at 4MHz, but the square wave is not very clean. However, it is a square wave and met within the expected results.

|  |  |
| --- | --- |
| **Figure 9: 4MHz Crystal** | **Figure 10: 82C84 Pin 12 OSC** |

The last two signals needed for complete verification were the RESET and READY signals. It was expected that when the one, the reset switch would switch the state of the machine and two that the READY would hold a constant wait state. In Figure 11 the RESET pulse was sample to show a change in states and in Figure 12 the constant wait state is captured.

|  |  |
| --- | --- |
| **Figure 11: 8284 Pin 10 RESET** | **Figure 12: 8284 Pin 4 READY** |

Having validated CLK, PCLK, OSC, RESET, READY and crystal, further testing of the CPU and EEPROM can be diagnosed.

* **Central Processing Unit**

Figure 13 and Figure 14 shows the Address Latch Enable and Q0 of 8086 respectively. Looking at the figure below it shows that the 8086 is resetting properly. The Address Latch enable is taking in 5V and giving out 250 MHz. The peaks in the ALE signal indicate enabling of the latches used to demultiplex the buses. Q0 is taking in 2V giving out a frequency 83.33Mhz, currently showing a signal of not much interest. Once the 8086 is interfaced with memory, these lines will show much more interesting results. The 8086 is viewed during reset at this time resets correctly, with values returning to original state. If a device had been interfaced with the 8086, more evident changes would be available.

|  |  |
| --- | --- |
| **Figure 13: 8086 ALE** | **Figure 14: 373 Q0** |

In Figures 15-17 the signal outputs of the 74LS245 buffer are tested.

|  |  |  |
| --- | --- | --- |
| **Figure 15: 245 B0** | **Figure 16: 245 DIR** | **Figure 17: 245 AD0** |

* **EEPROM Unit**

In order to check the functionality of the EEPROM several tests were recorded. It is important to recall that the 8086 has a 16-bit data bus and therefore can use what is called a write stobe to select between memory banks. The low and high banks of memory are selected using combinational logic of the Write pin (WR), Bus High Enable (BHE) and Bus Low Enable (BLE). The Low Bank is selected using BLE, in this case A0, and the High Bank is selected using BHE. For the purposes of testing, results were only compared to BLE or A0, not BHE.

|  |  |  |
| --- | --- | --- |
| **Ch1: ALE, Ch2: A0** | **CH1: ALE, Ch2: CE** | **Ch1: ALE, Ch2: RD** |

Address Latch Enable (ALE) tells the processor there is a transmission of memory address information happening. Below describes the logic required for data transmission between memory and CPU.

BHE A0

0 0 Word on

0 1 Upper byte on

1 0 Lower byte on

1 1 Upper byte on

Whenever the Read Signal (RD) is a logic zero, data is taken from the memory. Below are three samples taken with RD and D1.

|  |  |  |
| --- | --- | --- |
| **Ch1: RD, Ch2: D1 sample 1** | **Ch1: RD, Ch2: D1 sample 2** | **Ch1: RD, Ch2: D1 sample 3** |

Below is a sample that shows A0 (BLE) and A1. Here we can see that when A0 is a logic zero, A1 switches to a logic 0. Another sample is taken here to show that both the Write Signals for Low and High Banks are zero which is correct because we are not writing any data.

|  |  |
| --- | --- |
| **Ch1: A1, Ch2: A0** | **Ch1: WRL, Ch2: WRH** |

The results taken from phase two show that 8086 processor is receiving write signals generated from the infinite loop hard coded in the High and Low banks of memory. The data is matches the code that was hard coded so it stands to reason that this system is working.

* **SRAM Testing**

Words...

* Code Testing
  + Case Testing

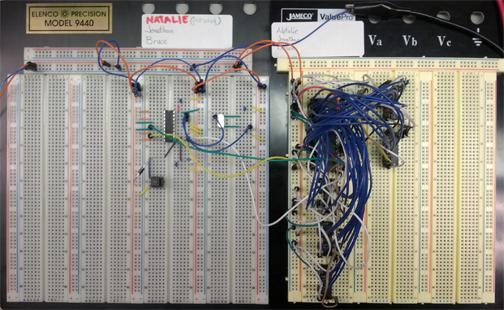
Words...

* + Edge Testing

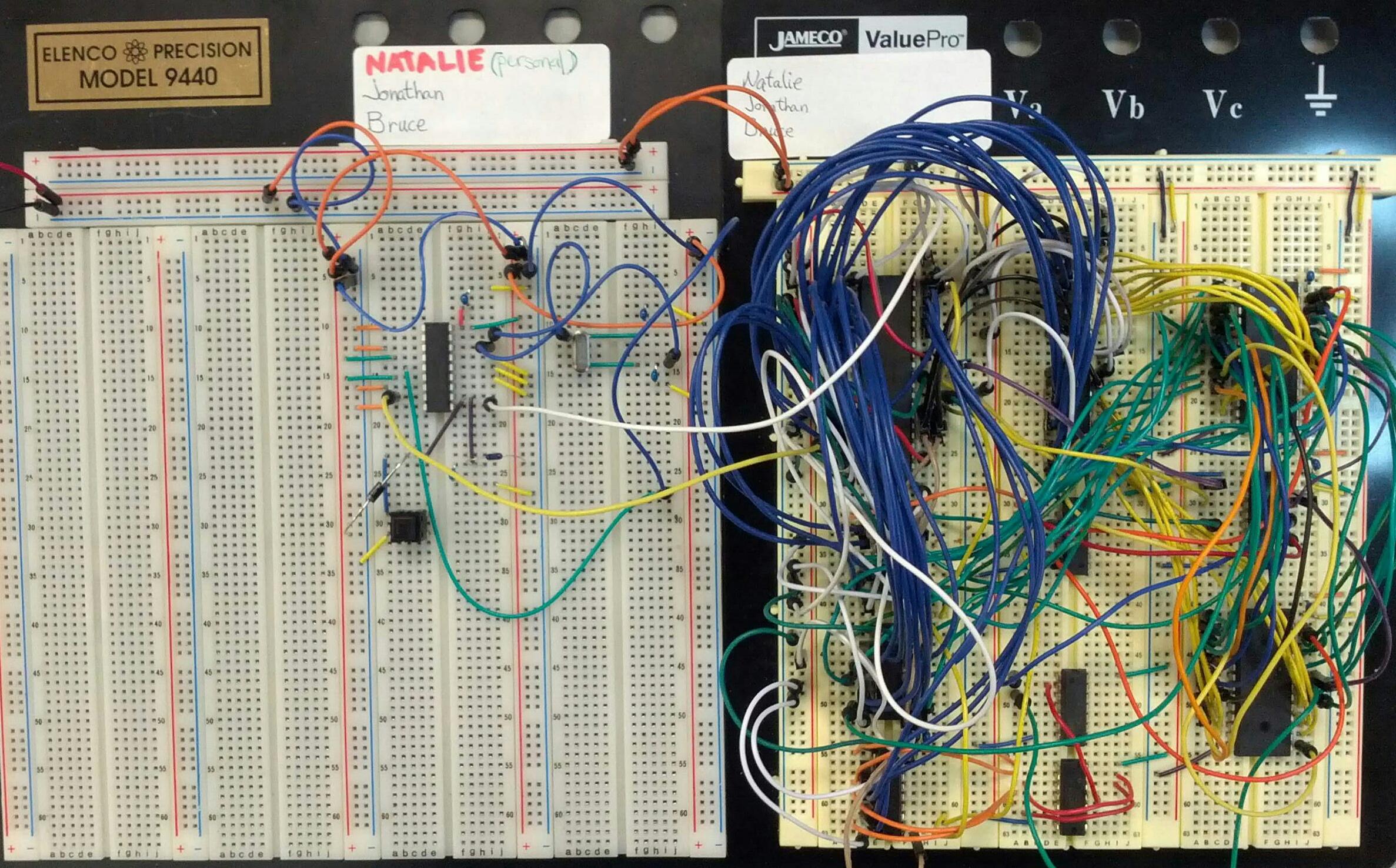
Words...

**Conclusions**

In order to complete this lab it was important to develop an understanding of how memory is interfaced with the microprocessor, as well as how the memory is virtualized within the processor, in order to program correctly. Seeing the various signals trigger and switch as the program executed was interesting and helped solidify concepts featured in the lab description. Careful planning was a huge component in this team’s success. Schematics were designed early, which allowed time for errors to be fixed. Also defining a best guess of expected signal outputs helped to make implementation day less stressful. Each time we checked our signals for our expected outcomes we were always surprised to find that our careful planning paid off, once output was properly formatted on the oscilloscope. We completed the project with minor debugging and were overall very pleased with our results, see Figure 18 for our initial build, and final system implementation. We feel confident moving on to the next phase of this system.



**Figure 18: Phase 1 Clock Generator and CPU Implementation**



**Phase 2 Clock Generator, CPU and EEPROM Implementation**

**References**

Brey, Barry. The Intel Microprocessors: Architecture, Programming, and Interfacing. Upper

Saddle River: Prentice Hall PTR, 2009. Print.

Mano, Morris and Michael Ciletti. *Digital Design, 4th Edition*. India: Dorling Kindersley Pvt.

Ltd., Pearson Education South Asia, 2007. Print.

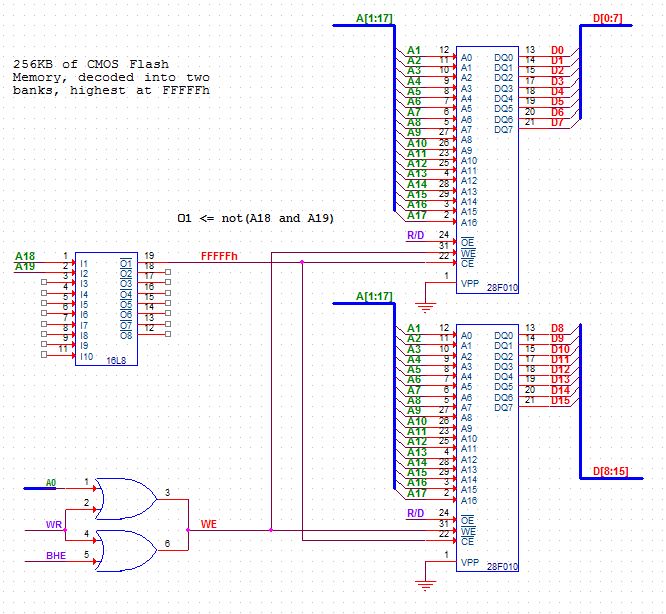
**Appendix A-1: 8086 Based System - Clock Generator**

****

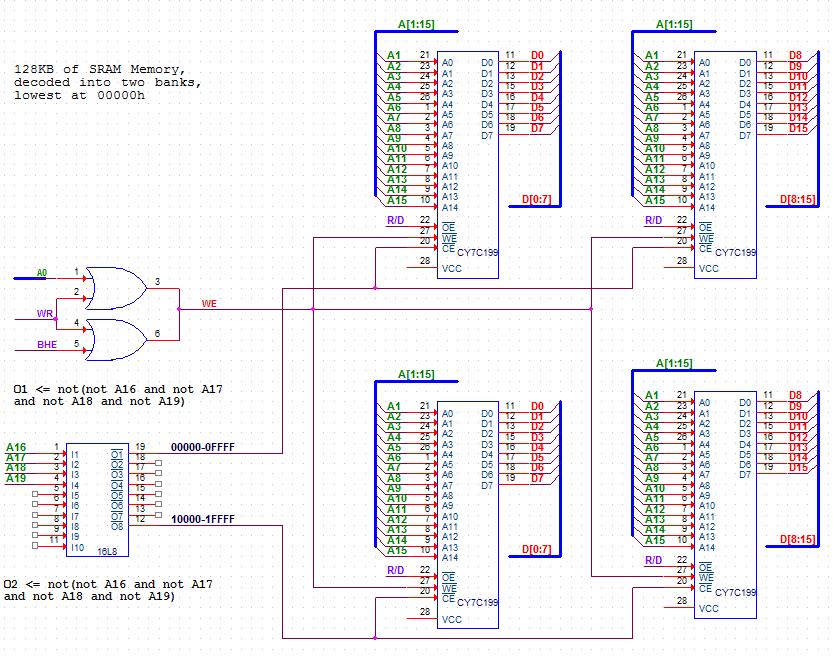
**Appendix A-2: 8086 Based System - CPU, EEPROM and SRAM**

**Appendix B-1: UMBC 8086 Trainer System - 8086**

**Appendix B-2: UMBC 8086 Trainer System – CMOS**

****

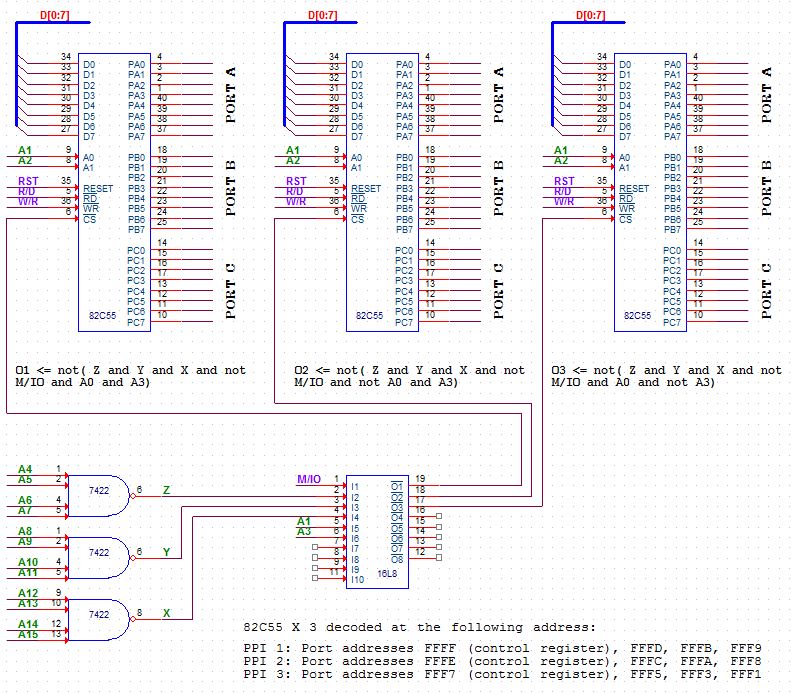
**Appendix B-3: UMBC 8086 Trainer System – SRAM**

****

**Appendix B-4: UMBC 8086 Trainer System - Clock Generator**

****

**Appendix B-5: UMBC 8086 Trainer System - 8255**

****

**Appendix B-6: UMBC 8086 Trainer System - 8259**

**Appendix B-7: UMBC 8086 Trainer System - 8254**

**Appendix B-8: UMBC 8086 Trainer System - 8279**

**Appendix B-9: UMBC 8086 Trainer System - 16550**

**Appendix B-10: UMBC 8086 Trainer System - LCD**

**Appendix B-11: UMBC 8086 Trainer System - 7-Segment Display & Diodes**

**Appendix B-12: UMBC 8086 Trainer System - DIP**

**Appendix C: Mortgage Calculator Code**