8086 Microprocessor Design Project

CMPE 310
Systems Design and Programming
Sabbir Ahmed

May 6, 2017

Contents

1 Introduction

This document provides detailed instructions to develop an 8086 microprocessor board using Cadence® OrCAD® Capture software. Included are the schematics of individual IC components and their description. Details of the ICs include decoding, programming specifications, and descriptions of IC pinouts.

1.1 Purpose

As per the project description, this document is to serve as the only documentation of the operational and functional specifications of the Intel 8086. The documentation is to be thorough and concise to provide information to design a similar board.

1.2 Scope and Organization of Document

The document will elaborate on the individual building blocks of the 8086 board. The integrated circuit (IC) chips used in designing the board will be discussed, along with brief, high-level overviews of their pinouts, their various connections and their functionalities. The connections and dependencies between the different components such as memory and IO devices will be discussed in detail.

The document is organized into sections that cover the individual components and their IC pinouts, functionalities, connections and role in the 8086 board. Schematics of the different components and their circuitry are included. Code snippets, including the VHDL (VHSIC Hardware Description Language) implementations of the decoding hardware and the Assembly implementations of the data and memory addressing, are also incorporated in the document.

2 8086 Microprocessor

The 8086 microprocessor is an enhanced version of the 8085 microprocessor developed by Intel in 1978. It is a 16-bit microprocessor, with 20 address lines and 16 data lines to provide up to 1 MB of physical memory. The 8086 microprocessor described in the project will operate in its minimum mode.

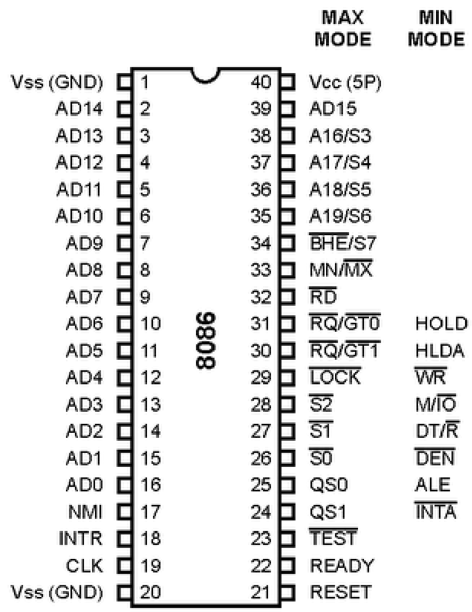

Figure 1: 8086 Microprocessor

2.1 Features

The 8086 microprocessor is known for its significant advancements since its predecessors. The most prominent features include, but are not limited to:

- 6 bytes of cache memory for faster processing
- Pipelining stages: Fetch Stage and Execute Stage
- Instruction queue
- 256 vectored interrupts
- Maximum and minimum modes of operation, suitable for multiple and single processors respectively

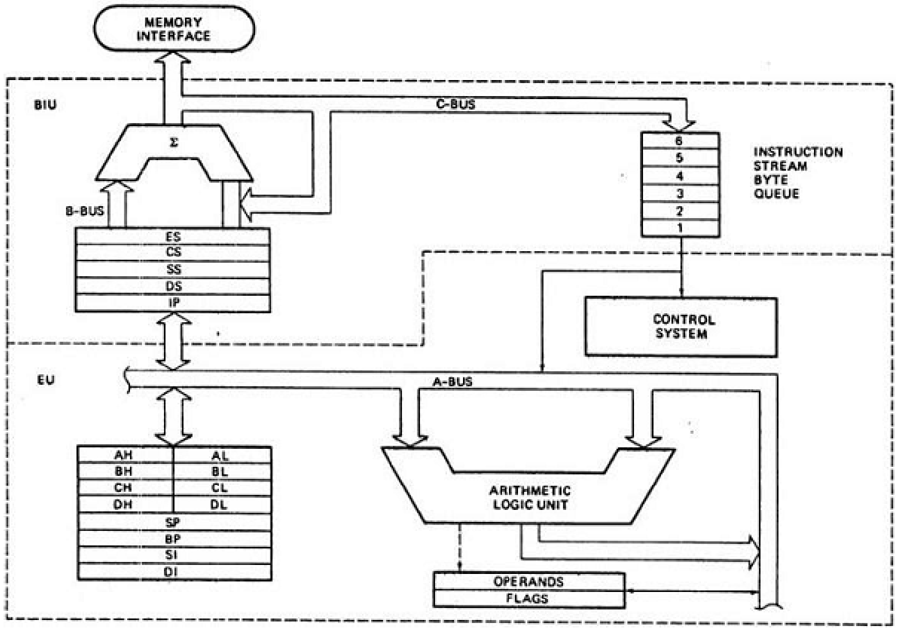

Figure 2: Architecture of 8086

2.2 Address and Data Buses

The 8086 CPU has a unidirectional address bus with 20 address lines and a bidirectional data bus with 16 data lines. [1] The address bus is used to select the desired memory or I/O device by generating a unique address which corresponds to the memory location or the location of I/O device of the system. The data bus is used to transfer data between the CPU and memory and the CPU and I/O devices.

The address bus is denoted as $A_{19} - A_0$ (20 lines) and the data bus $D_{15} - D_0$ (16 lines). The peripheral devices implemented with the 8086 in this document however consist of 8-bit data bus architectures. The data bus would therefore be multiplexed and more commonly denoted as $D_7 - D_0$ (8 lines).

2.3 Control Bus

The control bus of 8086 carries control signals which are used to specify the memory and I/O devices. [1] The bus is bidirectional and assists the CPU in synchronizing control signals to internal devices and external components. It is comprised of interrupt lines, byte enable lines, read/write signals and status lines.

2.4 Pinouts

Refer to Appendix B for the pinouts of the chip.

3 Decoding

Discrete gate integrated chips are used throughout the board for decoding and demultiplexing address, data and control lines.

3.1 Programmable Logic Device - PAL16L8

Programmable Array Logic (PAL) is a type of Programmable Logic Device (PLD) used to implement logical functions. [2] PALs comprise of an AND gate array followed by an OR gate array, as shown in Figure 3.

16L8s were programmed to be used throughout the board to implement the various functionalities required in the system.

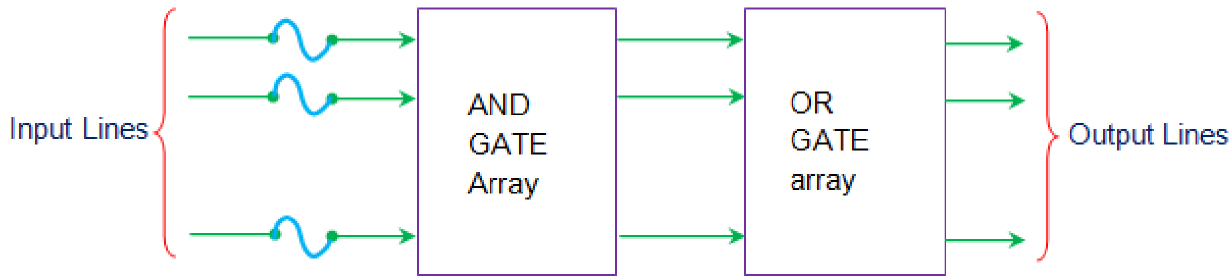

Figure 3: 8086 Microprocessor

3.2 Programming the PLD

VHSIC Hardware Description Language (VHDL) was used to program the different logical functions. The modules used for the nine (9) 16L8s used in the system are implemented in Appendix C.

3.3 Pinouts

Refer to Appendix B for the pinouts of the chip.

4 Clock Generator - 8284A

The 8184A Clock Generator is an ancillary component to the 8086. This system clock is used to synchronize both internal and external operations using an external oscillator. The device is also used for READY and RESET synchronizations and TTL-level peripheral clock signal generation.

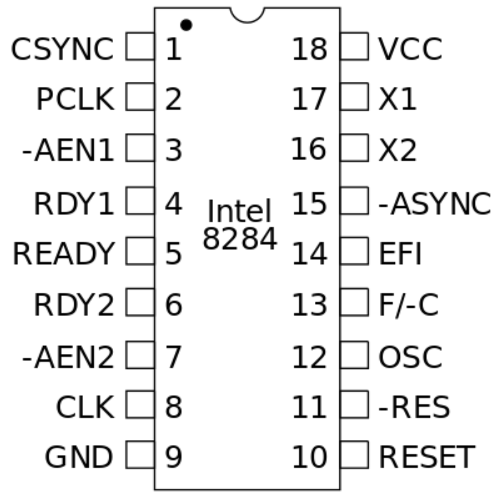

Figure 4: 8284A Clock Generator

4.1 Clock Speed

The 8086 internal clock has a frequency of 5 MHz ($\frac{1}{3}$ of CLK). The external crystal typically oscillates at 15 MHz.

```
; ----------------------------------------------------------------------------------------
; Writes "Hello, World" to the console using only system calls. Runs on 64-bit Linux only.
; To assemble and run:
;
;     nasm -felf64 hello.asm && ld hello.o && ./a.out
; ----------------------------------------------------------------------------------------

        global  _start

        section .text
_start:
        ; write(1, message, 13)
        mov     rax, 1                  ; system call 1 is write
        mov     rdi, 1                  ; file handle 1 is stdout
        mov     rsi, message            ; address of string to output
        mov     rdx, 13                 ; number of bytes
        syscall                         ; invoke operating system to do the write
```

```
        ; exit(0)
        mov     eax, 60                 ; system call 60 is exit
        xor     rdi, rdi                ; exit code 0
        syscall                         ; invoke operating system to exit
message:
        db      "Hello, World", 10      ; note the newline at the end
```

4.2 RESET Operation

Correct reset timing requires that the RESET input to the 8086 becomes a logic 1 in 4 clock cycles and remain high for at least 50 μS. The reset switch is implemented in a RC circuit with typical resistance of 100 $k\Omega$ and 10 μF.

4.3 Pinouts

Refer to Appendix B for the pinouts of the chip.

5 Memory Architecture

5.1 Static Random Access Memory - CY7C199

5.2 Interfacing Memory Banks with the Microprocessor

5.3 Addressing

5.4 CMOS Flash Memory - 28F010

5.5 Flash Memory Implementation

5.6 Addressing Flash Memory

5.7 Pinouts

Refer to Appendix B for the pinouts of the chips.

6 Programmable Keyboard/Display Interface - 8279

6.1 Description

6.2 Interfacing with a 5x5 Keyboard Matrix

6.3 Addressing

6.4 Programming the Keyboard Interface

6.5 Command Words to Program the 8279

6.6 Assembly Implementation

6.7 Pinouts

Refer to Appendix B for the pinouts of the chip.

7 Programmable Interval Timer - 8254

7.1 Description

7.2 Programming

7.3 Addressing

7.4 Assembly Implementation

7.5 Pinouts

Refer to Appendix B for the pinouts of the chip.

8 External Headers

8.1 Description

8.2 Interfacing 30-Pin Headers with the 8255

8.3 Addressing

8.4 Assembly Implementation and Programming of the 8255

8.5 Interfacing 14-Pin Headers with the 8254

8.6 Interfacing 14-Pin Headers with the 8259

8.7 Interfacing 60-Pin External Header to the Address, Data and Control Bus

9 Interrupt Controller - 8259

9.1 Description

9.2 Implementing a Master Interrupt Controller

9.3 Addressing

9.4 Assembly Implementation and Programming

10 UART

10.1 16550 UART

10.2 Addressing the 16550

10.3 Programming the 16550

10.4 Assembly Implementation

10.5 MAX-235 and D-SUB-9

10.6 Device Descriptions and Implementations

10.7 Pinouts

Refer to Appendix B for the pinouts of the chip.

11 LCD Display

11.1 Addressing

11.2 Assembly Implementation

12 LEDs and DIP Switches

12.1 Seven-Segment LEDs

12.2 Addressing

12.3 LEDs

12.4 Addressing

12.5 DIP Switches

12.6 Addressing

A Appendix A: Schematics

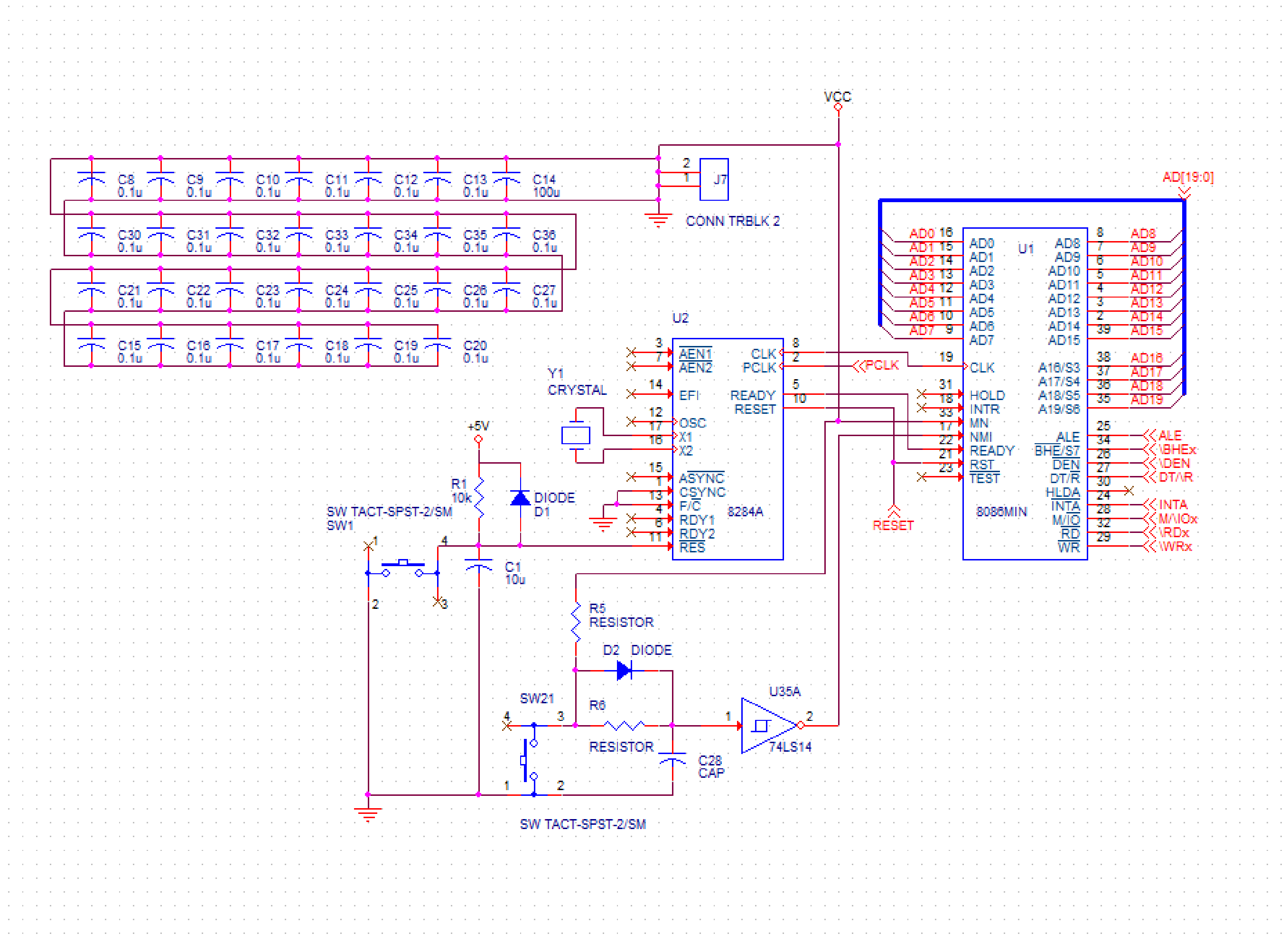

Figure 5: 8086 interfaced with the 8284A clock generator and its Reset RC Push Button Circuit, and the Power Bank of the Board

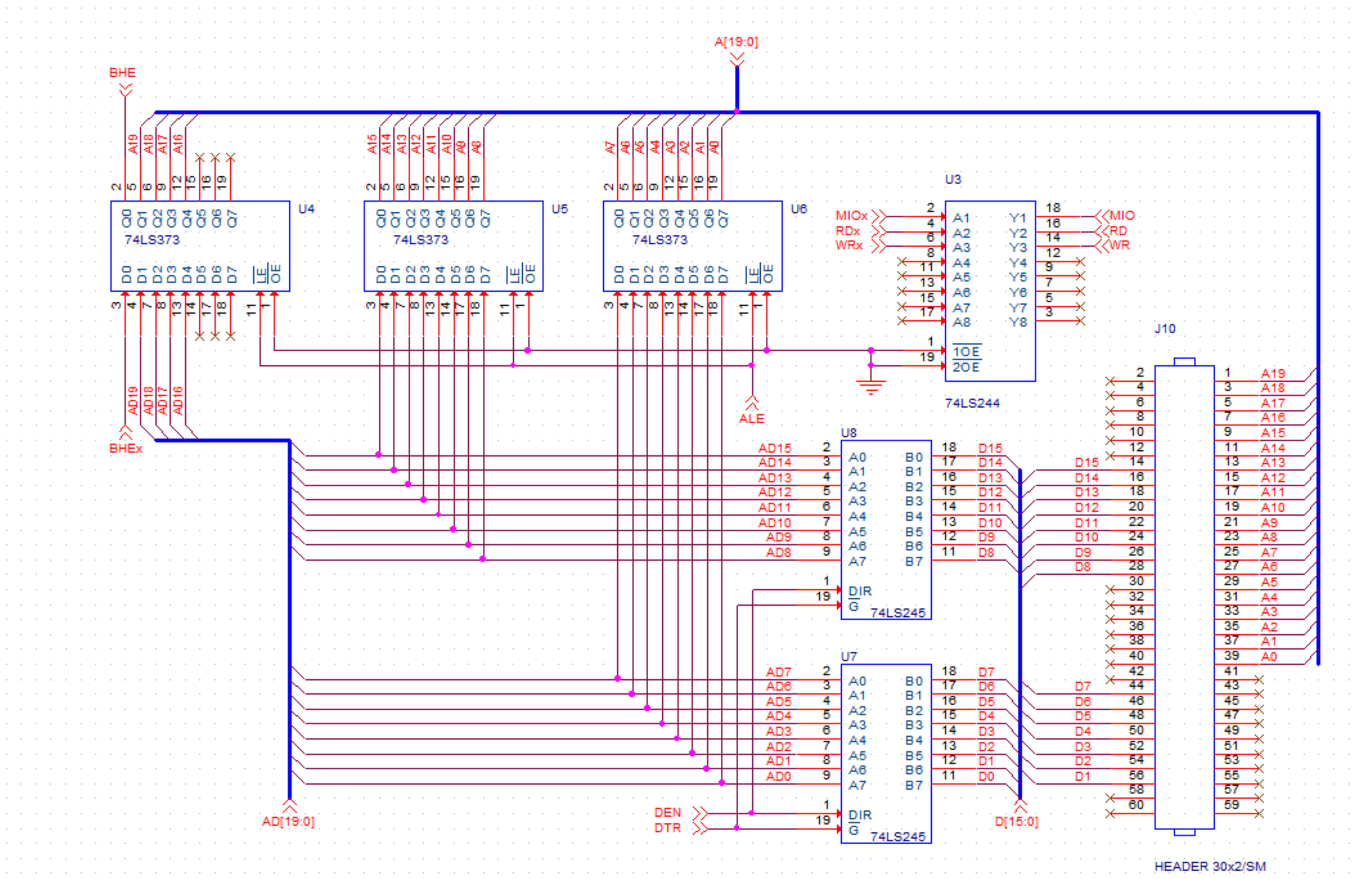

Figure 6: 8086 Demultiplexed with Address and Data Buses Pulled into Headers

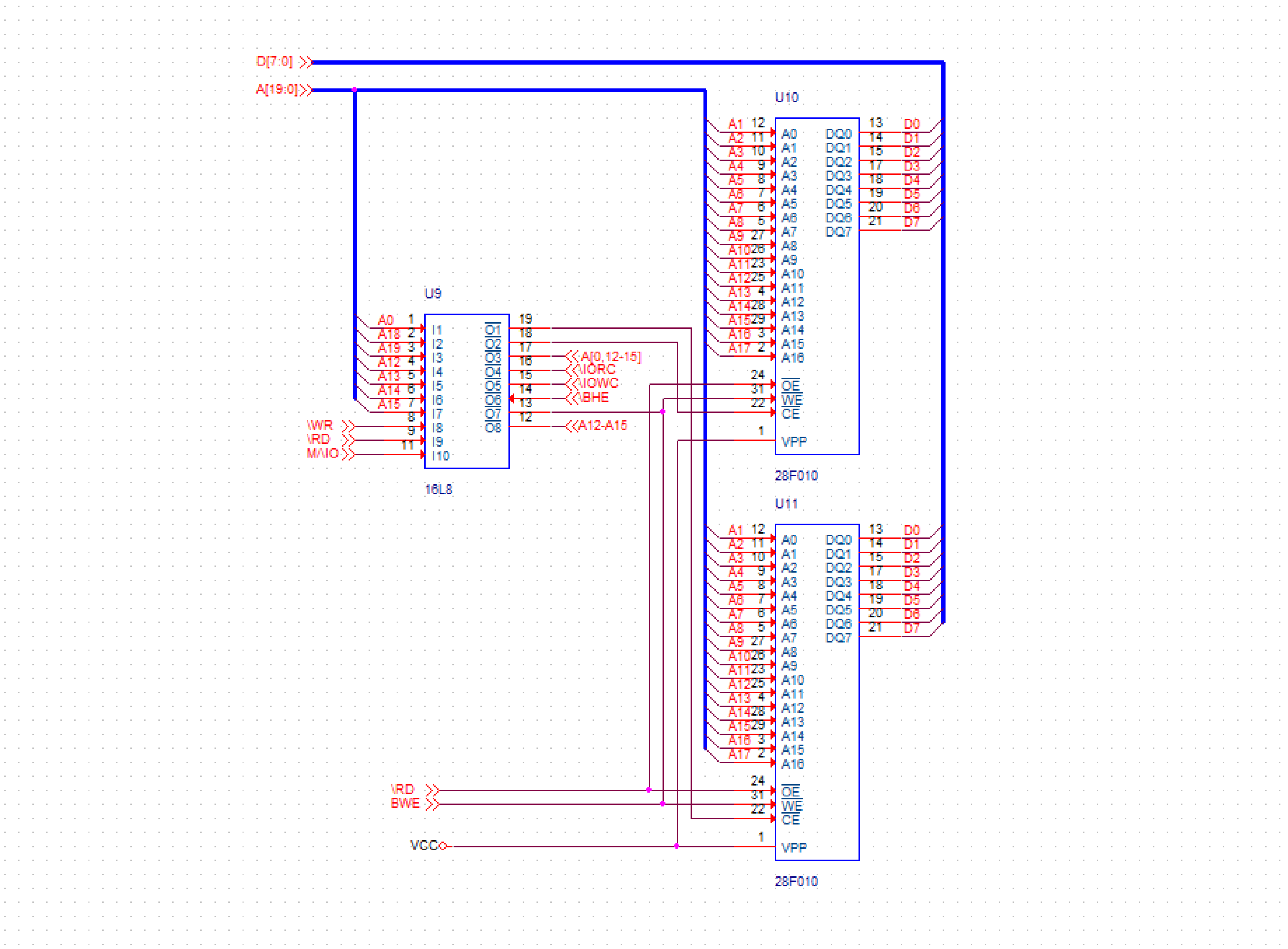

Figure 7: 256 kB of CMOS Flash Memory and 128 kB Static SRAM

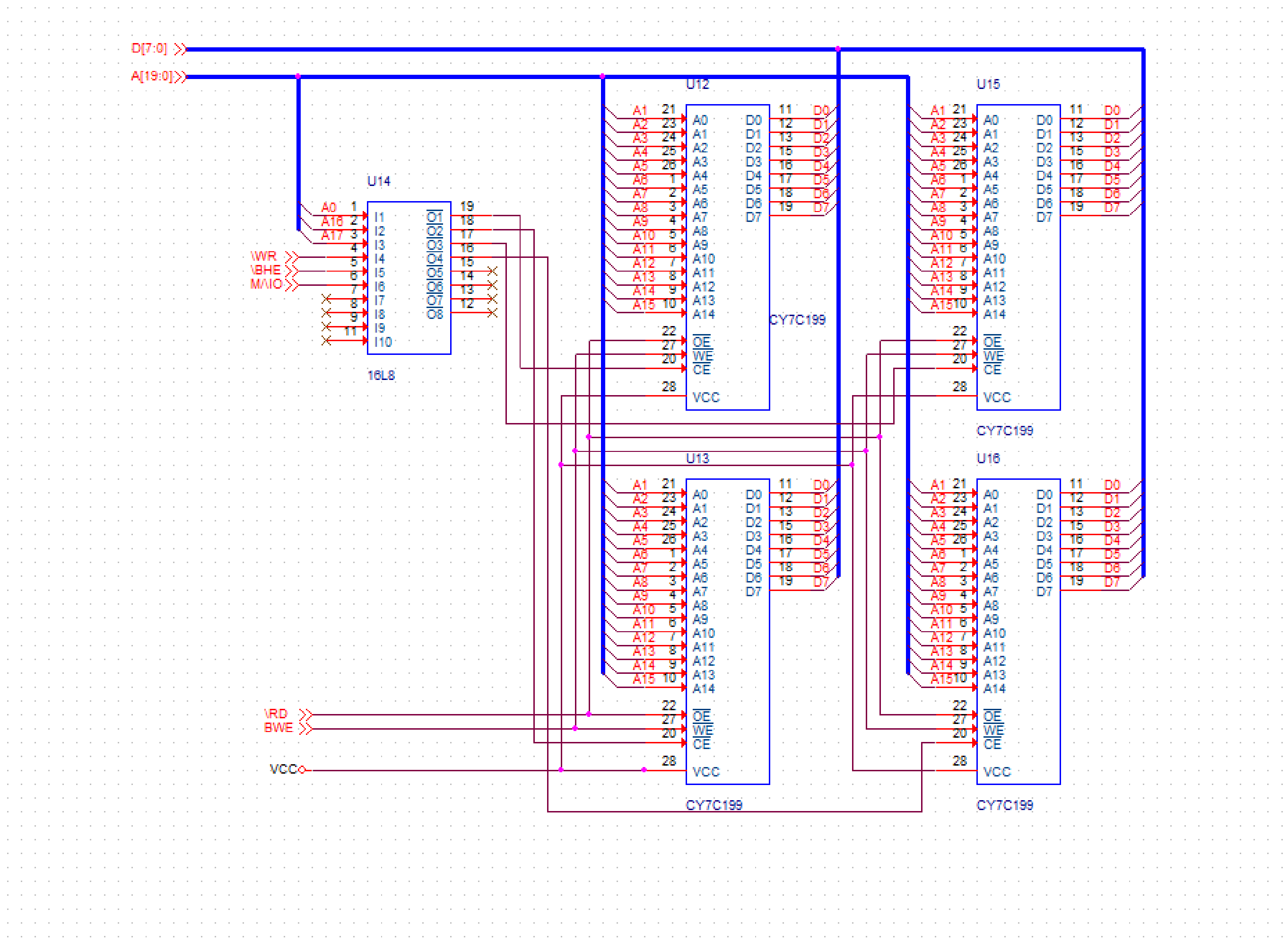

Figure 8: 128 kB Static SRAM

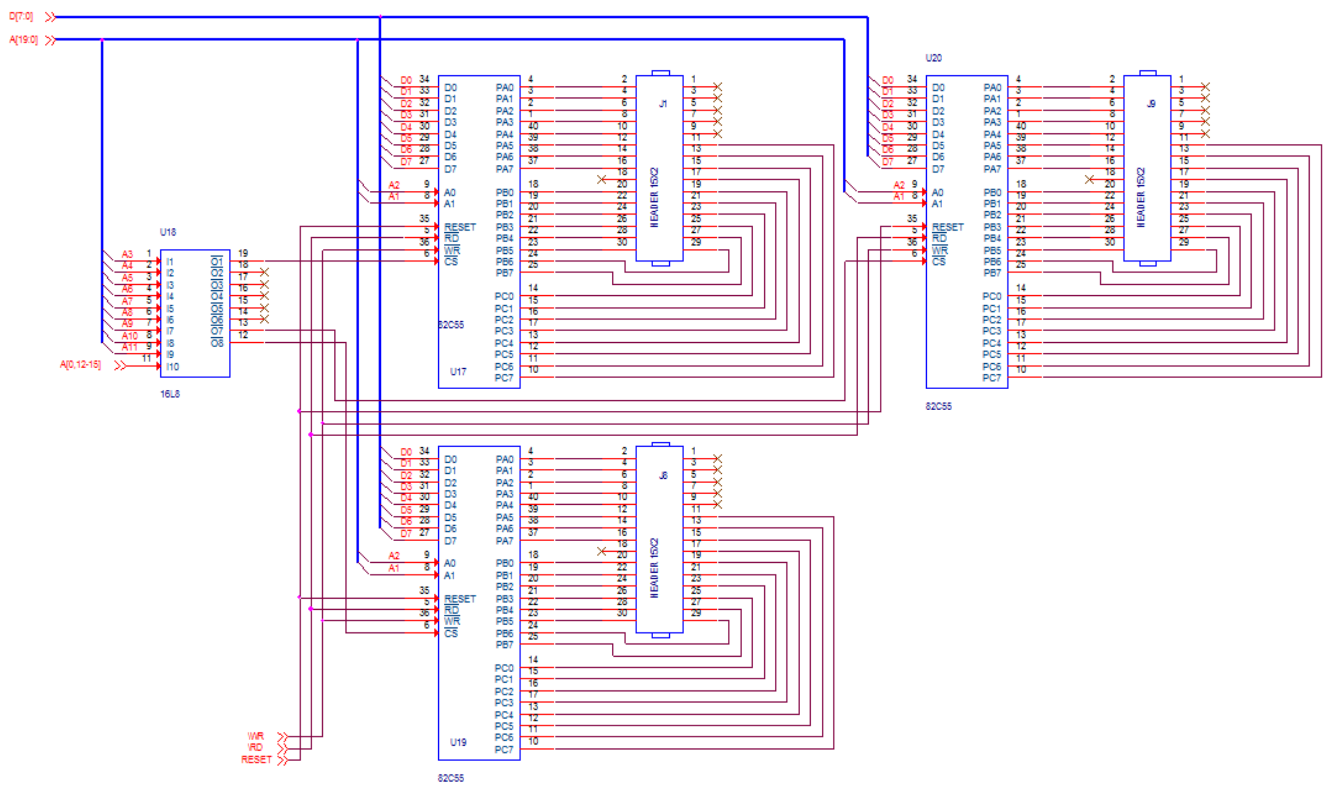

Figure 9: Programmable Peripheral Interface Chips with Port Connections Pulled into Headers

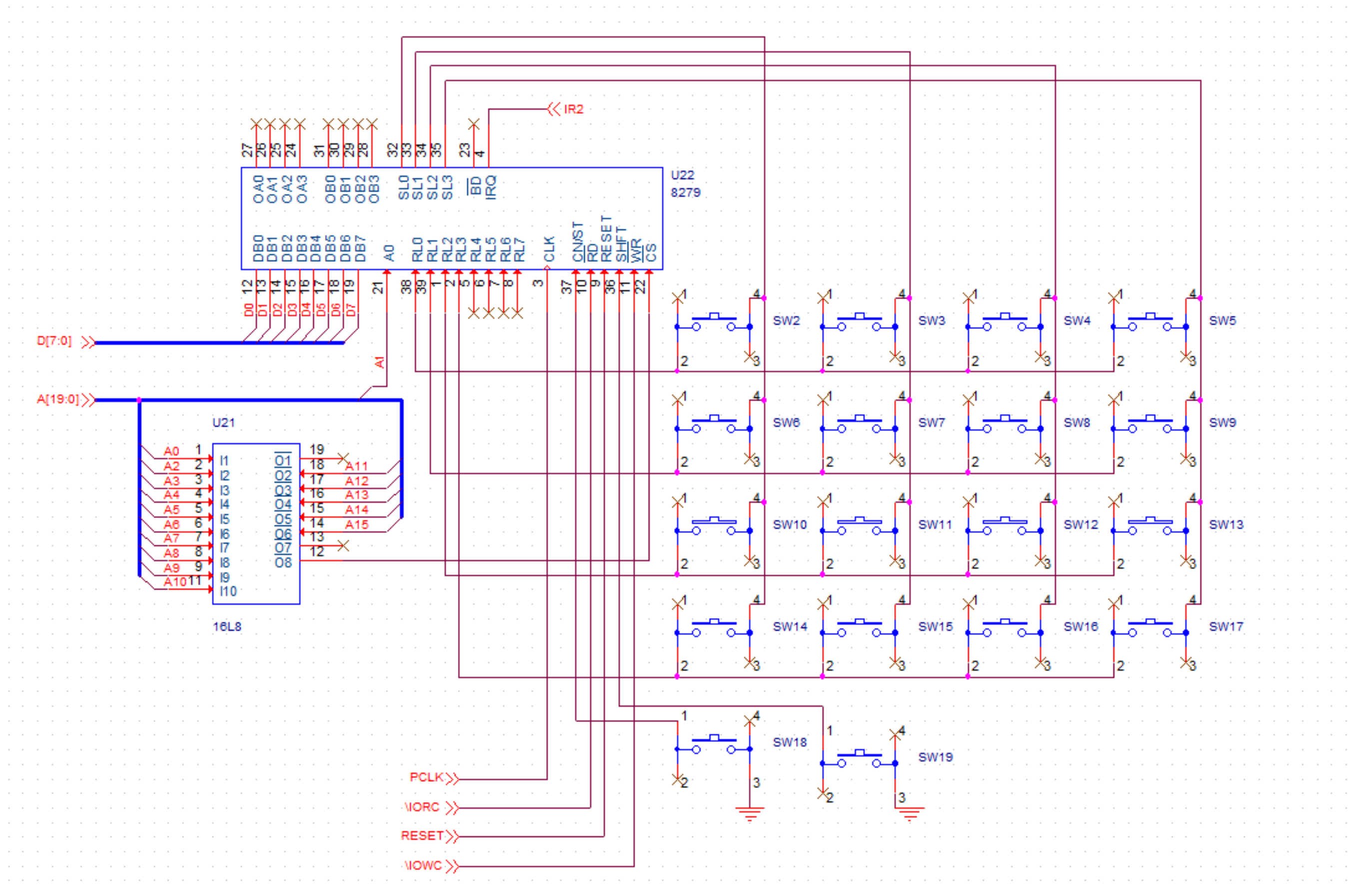

Figure 10: 5×4 Keyboard Matrix

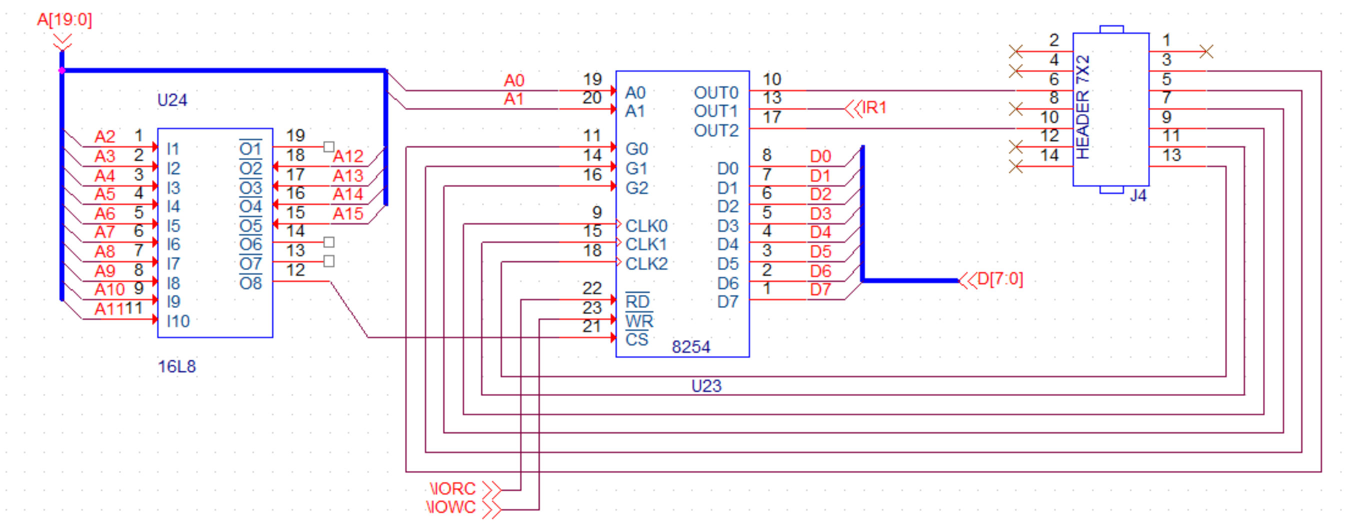

Figure 11: Programmable Interval Timer

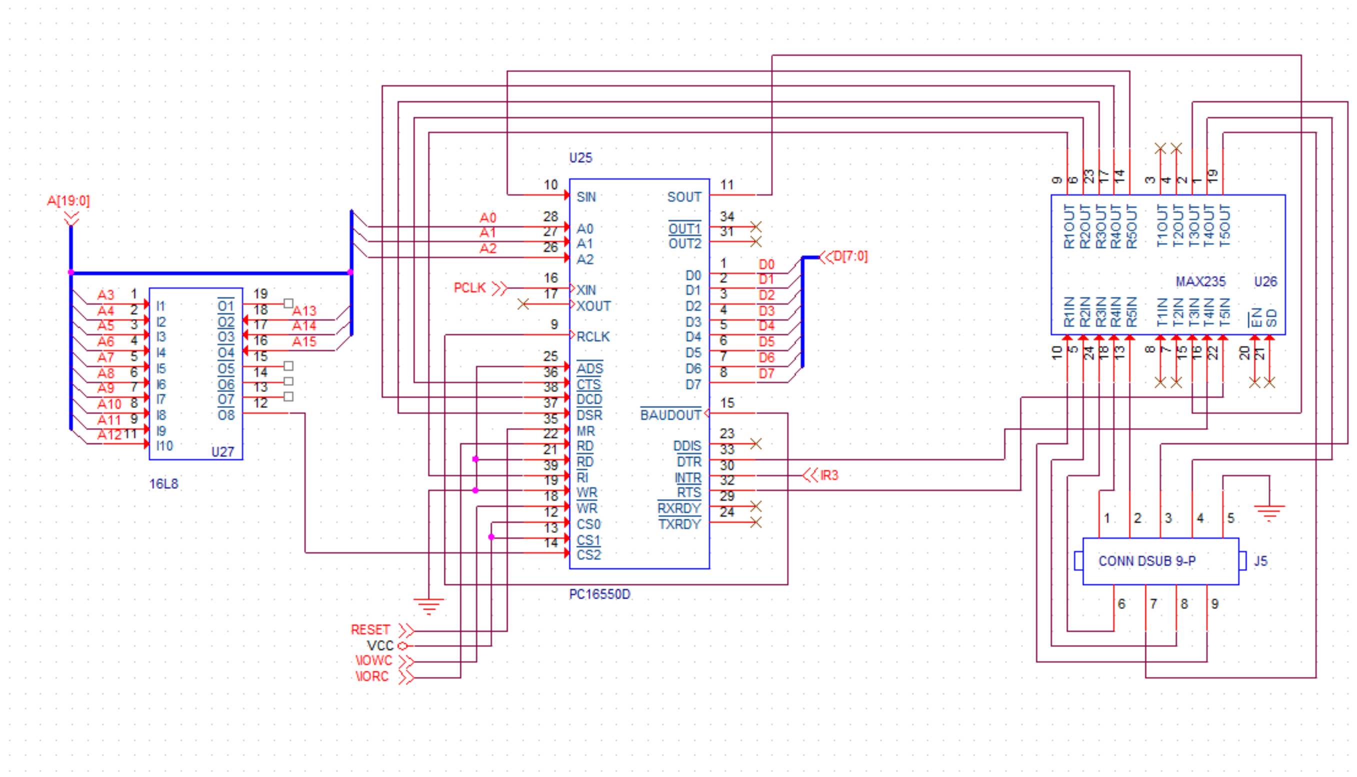

Figure 12: UART Connected for Serial Port Using a Line Driver/Receiver and a DSUB-9 connector

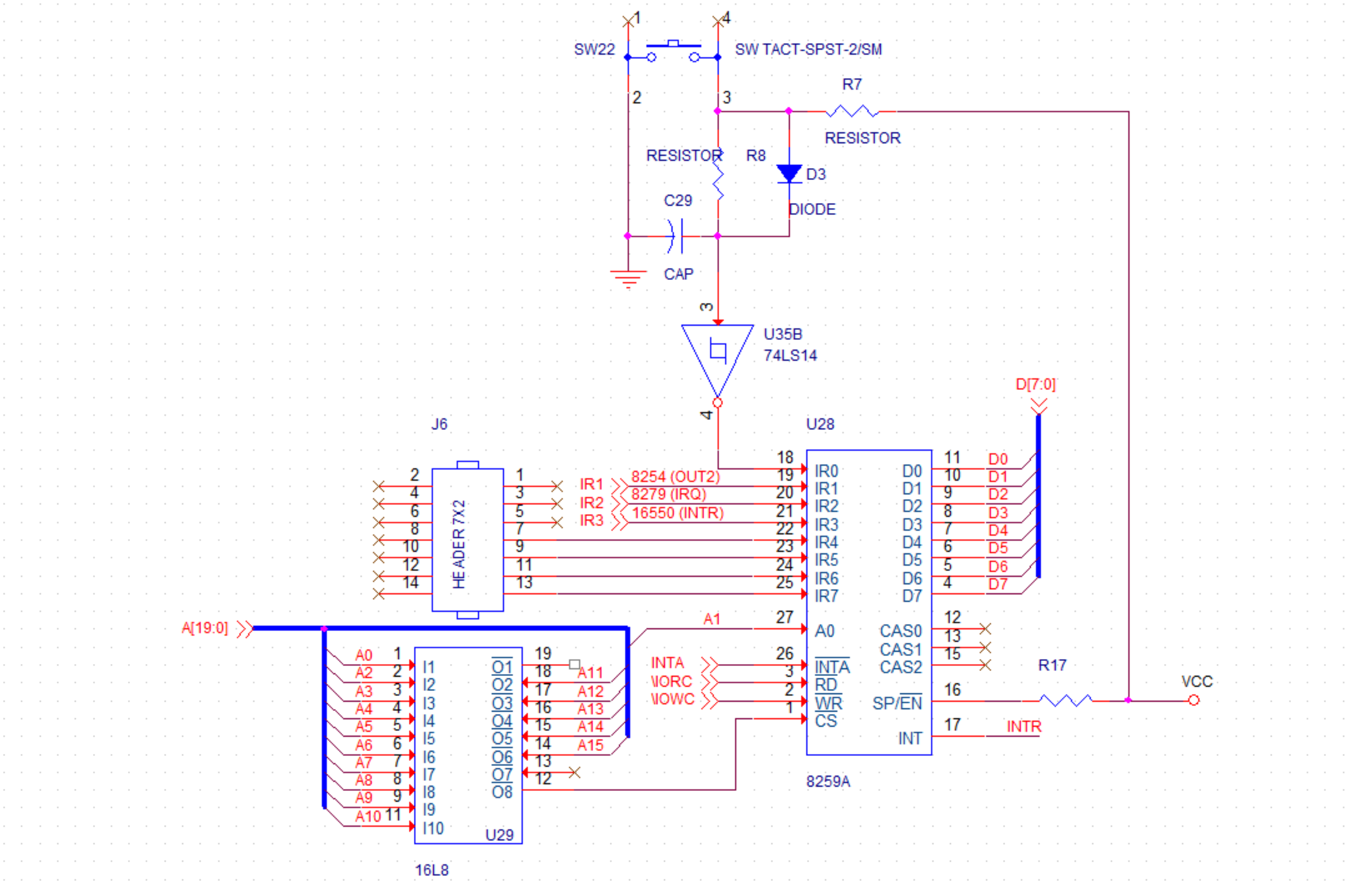

Figure 13: Programmable Interrupt Controller with Headers fro External Access

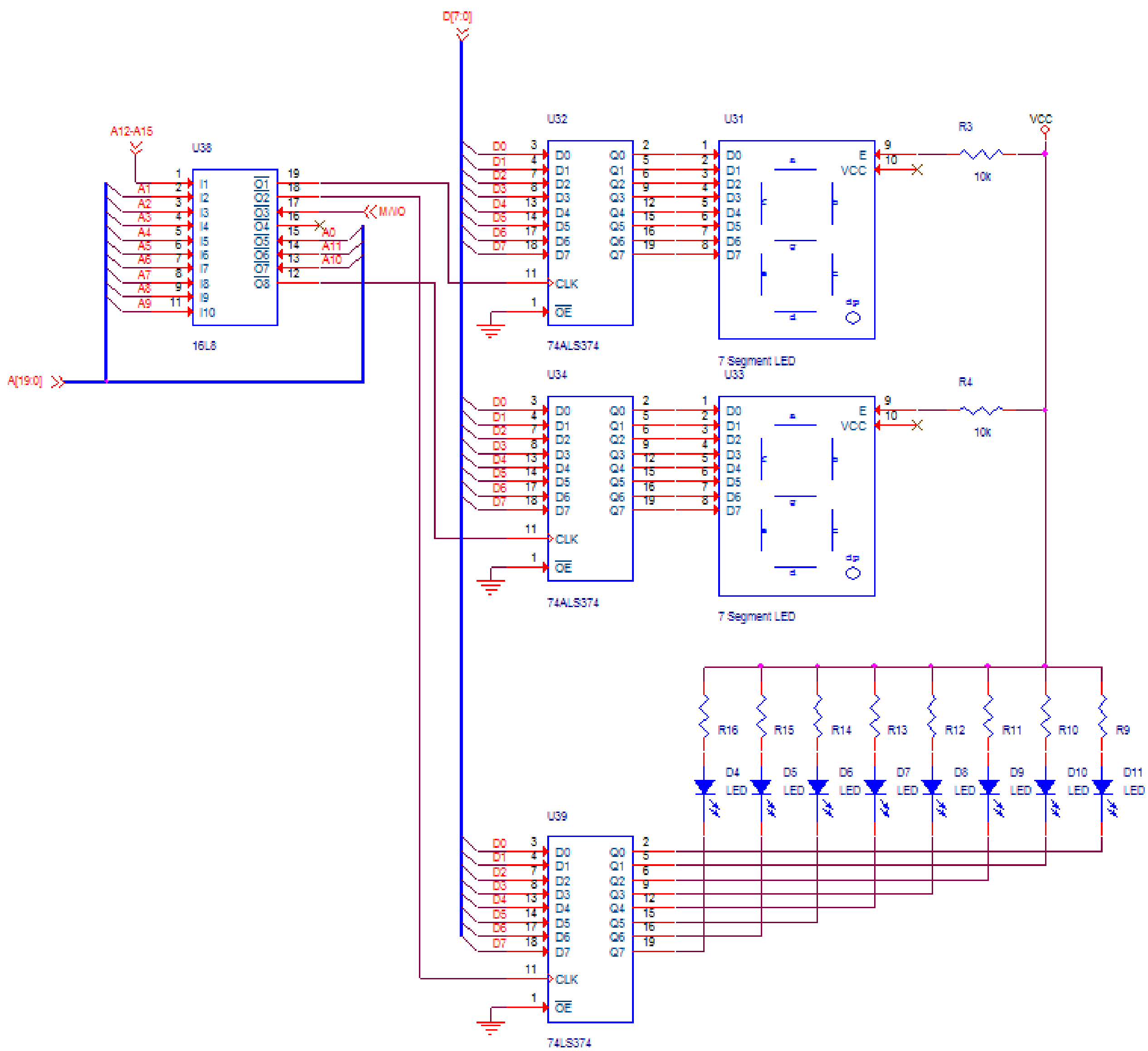

Figure 14: Common-Anode 7-Segment LEDs with 8 LEDs

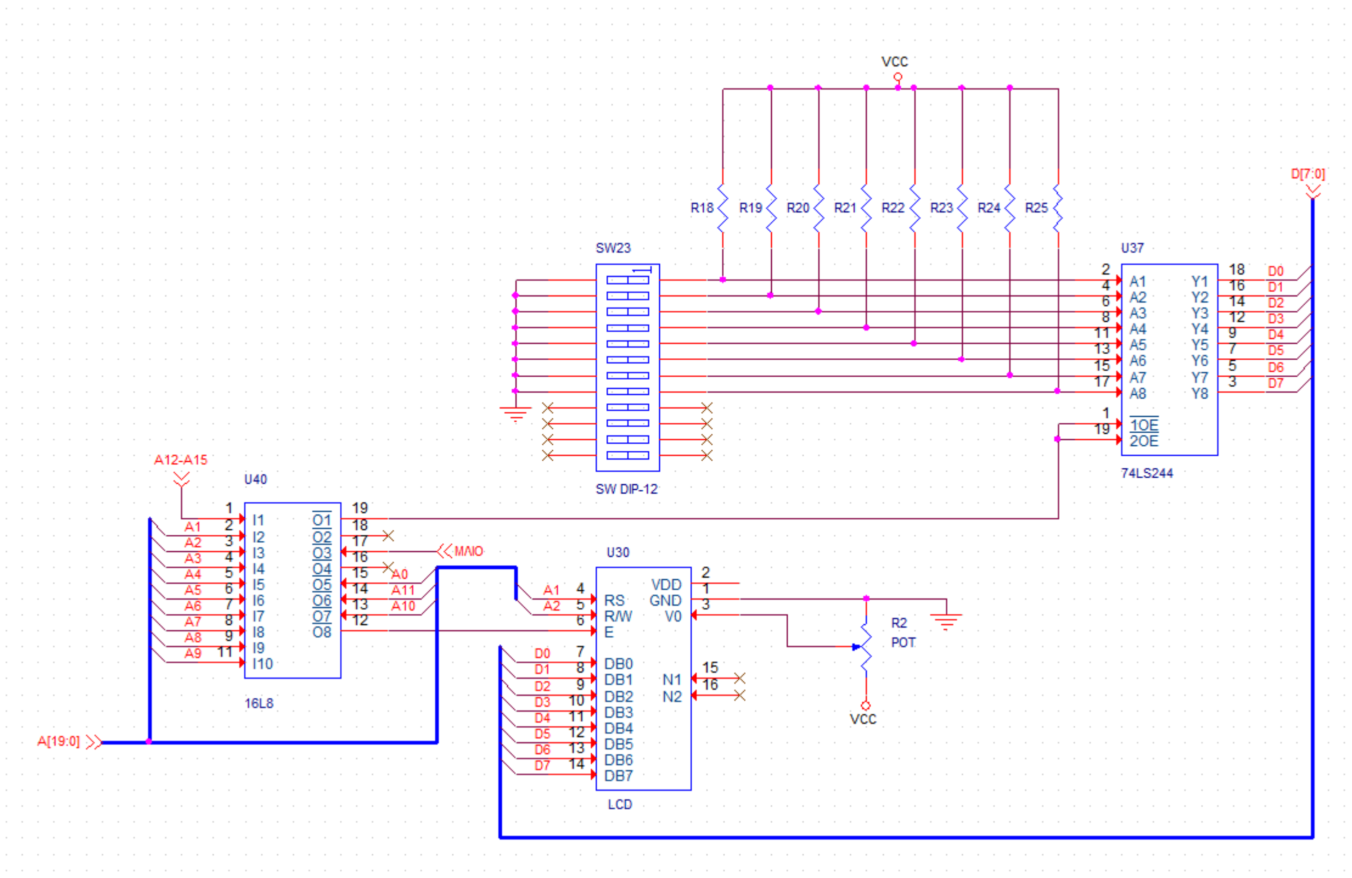

Figure 15: 20 character × 4 line LCD Display with an Integrated LCD Controller

B Appendix B: Pinouts

B.1 8086 Chip

- $M/\overline{IO}$: (Memory/ I/O) indicates if the address is a memory or I/O address
- $\overline{INTA}$: (Interrupt Acknowledgment) generated in response to $INTR$ to put the interrupt vector on the data bus
- ALE: (Address Latch Enable) when 1, address data bus contains a memory or I/O address
- $\overline{DEN}$: (Data Bus Enable) activates external data bus buffers

C Appendix C: Code Implementations

C.1 U9

```
-------------------------------------------------------------------------------
--
-- Module: DECODER_U9
-- Architecture used to decode address and control lines to the CMOS
-- flash memory
--
-------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U9 is
    port (
        A0, A12, A13, A14, A15, A18, A19, WR, RD, MIO, BHE: in STD_LOGIC;
        O1, O2, A12_A15, A0xA12_A15, IORC, IOWC: out STD_LOGIC
    );
end DECODER_U9;

architecture V1 of DECODER_U9 is
begin
    O1 <= A19 or A18 or A15 or A14 or A13 or A12 or not(A0);
    O2 <= A19 or A18 or A15 or A14 or A13 or A12 or A0;
    A12_A15 <= A15 or A14 or A13 or A12;
    A0xA12_A15 <= A15 or A14 or A13 or A12 or A0;
    IORC <= RD or MIO;
    IOWC <= WR or MIO;
end V1;
```

C.2 U14

```
-------------------------------------------------------------------------------
--
```

```
-- Module: DECODER_U14
-- Architecture used to decode address and control lines to the SRAM
--
-------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U14 is
    port (
        A0, A16, A17, WR, BHE, MIO: in STD_LOGIC;
        O1, O2, O3, O4: out STD_LOGIC
    );
end DECODER_U14;

architecture V1 of DECODER_U14 is
begin
    O1 <= not(A17) or not(A16) or A0 or not(WR) or not(BHE) or not(MIO);
    O2 <= not(A17) or A16 or A0 or not(WR) or not(BHE) or not(MIO);
    O3 <= A17 or not(A16) or A0 or not(WR) or not(BHE) or not(MIO);
    O4 <= A17 or A16 or A0 or not(WR) or not(BHE) or not(MIO);
end V1;
```

C.3 U18

```
-------------------------------------------------------------------------------
--
-- Module: DECODER_U18
-- Architecture used to decode address and control lines to the PPI chips
--
-------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U18 is
    port (
        A12_A15, A3, A4, A5, A6, A7, A8, A9, A10, A11, MIO: in STD_LOGIC;
        PPI1, PPI2, PPI3: out STD_LOGIC
    );
end DECODER_U18;

architecture V1 of DECODER_U18 is
begin
    A12_A15 <= A15 or A14 or A13 or A12;
    PPI1 <= A12_A15 or
            A11 or A10 or A9 or A8 or
```

```
            A7 or A6 or A5 or A4 or
            not(A3) or not(A0) or not(MIO);
    PPI2 <= A12_A15 or
            A11 or A10 or A9 or A8 or
            A7 or A6 or A5 or A4 or
            not(A3) or A0 or not(MIO);
    PPI3 <= A12_A15 or
            A11 or A10 or A9 or A8 or
            A7 or A6 or A5 or A4 or
            A3 or not(A0) or not(MIO);
end V1;
```

C.4 U21

```
------------------------------------------------------------------------------
--
-- Module: DECODER_U21
-- Architecture used to decode address and control lines to the keyboard
--
------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U21 is
    port (
        A0, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14,
            A15: in STD_LOGIC;
        O8: out STD_LOGIC
    );
end DECODER_U21;

architecture V1 of DECODER_U21 is
begin
    O8 <= A15 or A14 or A13 or A12 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or not(A5) or not(A4) or
          A3 or A2 or A0;
end V1;
```

C.5 U24

```
------------------------------------------------------------------------------
--
-- Module: DECODER_U24
-- Architecture used to decode address and control lines to the
-- programmable interval counter
--
```

```
--------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U24 is
    port (
        A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14,
            A15: in STD_LOGIC;
        O8: out STD_LOGIC
    );
end DECODER_U24;

architecture V1 of DECODER_U24 is
begin
    O8 <= not(A15) or not(A14) or not(A13) or not(A12) or
          not(A11) or not(A10) or not(A9) or not(A8) or
          not(A7) or not(A6) or not(A5) or A4 or
          not(A3) or A2;
end V1;
```

C.6 U27

```
--------------------------------------------------------------------------------
--
-- Module: DECODER_U27
-- Architecture used to decode address and control lines to the UART
--
--------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U27 is
    port (
        A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14,
            A15: in STD_LOGIC;
        O8: out STD_LOGIC
    );
end DECODER_U27;

architecture V1 of DECODER_U27 is
begin
    O8 <= A15 or A14 or A13 or A12 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or not(A5) or A4 or
          A3;
```

```
end V1;
```

C.7 U29

```
--------------------------------------------------------------------------------
--
-- Module: DECODER_U29
-- Architecture used to decode address and control lines to the interrupt
-- controller
--
--------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U29 is
    port (
        A0, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14,
            A15: in STD_LOGIC;
        O8: out STD_LOGIC
    );
end DECODER_U29;

architecture V1 of DECODER_U29 is
begin
    O8 <= not(A15) or not(A14) or not(A13) or not(A12) or
          not(A11) or not(A10) or not(A9) or not(A8) or
          not(A7) or not(A6) or not(A5) or not(A4) or
          A3 or not(A2);
end V1;
```

C.8 U38

```
--------------------------------------------------------------------------------
--
-- Module: DECODER_U38
-- Architecture used to decode address and control lines to the LEDs
--
--------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U38 is
    port (
        A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12_A15,
            MIO: in STD_LOGIC;
```

```
        O1, O2, O8: out STD_LOGIC
    );
end DECODER_U38;

architecture V1 of DECODER_U38 is
begin
    A12_A15 <= A15 or A14 or A13 or A12;
    O1 <= A12_A15 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or A5 or A4 or
          not(A3) or not(A2) or not(A1) or A0;
    O2 <= A12_A15 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or A5 or A4 or
          not(A3) or not(A2) or A1 or A0;
    O8 <= A12_A15 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or A5 or A4 or
          not(A3) or not(A2) or not(A1) or not(A0);
end V1;
```

C.9 U40

```
------------------------------------------------------------------------------
--
-- Module: DECODER_U40
-- Architecture used to decode address and control lines to the LCD and DIP
-- switches
--
------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;

entity DECODER_U40 is
    port (
        A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12_A15,
            MIO: in STD_LOGIC;
        O1, O8: out STD_LOGIC
    );
end DECODER_U40;

architecture V1 of DECODER_U40 is
begin
    A12_A15 <= A15 or A14 or A13 or A12;
    O1 <= A12_A15 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or A5 or A4 or
```

```
          not(A3) or A2 or not(A1) or A0;
    O8 <= A12_A15 or
          A11 or A10 or A9 or A8 or
          not(A7) or not(A6) or A5 or not(A4) or
          A3 or A0;
end V1;
```

References

[1] DBHJDS http://gradestack.com/Microprocessors-and/Architecture-of-8086-and/Address-Bus-Data-Bus-/19317-3912-38171-study-wtw

[2] Programmable Array Logic https://www.electrical4u.com/programmable-array-logic/