Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Electrical Engineering Department

Logo, company name, sunburst chart

Description automatically generated

|  |
| --- |
| **Microprocessors Course**  **Hardware Project** |

|  |  |  |
| --- | --- | --- |
| *Authors:* |  | *Supervisor:* |
| Leen Almousa  Ahmad Alsharif | 20210268  20190528 | Dr. Heba Abdel Nabi |

*January 12, 2025*

***Abstract***

**Table of Contents**

[1 Introduction 2](#_Toc187653176)

[2 Description of work 2](#_Toc187653177)

[2.1 Hardware design 2](#_Toc187653178)

[**8086 microprocessor** 3](#_Toc187653179)

[**74LS245 transceiver** 4](#_Toc187653180)

[**Latches and decoder** 5](#_Toc187653181)

[**IO devices and 8255 PPI module** 7](#_Toc187653182)

[2.2 Software Design 8](#_Toc187653183)

[2.2.1 **Discussion** 8](#_Toc187653184)

[2.2.2 **FlowChart** 10](#_Toc187653186)

[3 Results and Discussions 12](#_Toc187653187)

[4 Conclusions 13](#_Toc187653188)

[5 References 14](#_Toc187653189)

# Introduction

This project includes design and implementation of a digital keypad interface using an 8086 microprocessor, an 8255 Programmable Peripheral Interface (PPI), and a seven-segment display. The project focuses on interfacing these components to accurately read key presses from a 4x3 matrix keypad and display the corresponding digit on the seven-segment display. The 8255 PPI plays a crucial role in managing input from the keypad and output to the display, while the 8086 provides the processing logic for scanning the keypad and generating the appropriate seven-segment codes. 74LS138 decoder is implemented to address the 8255, adhering to the specific requirements outlined in the project documentation.

# Description of work

This section provides a detailed explanation of the design and implementation of the digital lock system using an 8086 microprocessor, 8255 PPI, keypad, and seven-segment display.

## 2.1 Hardware design

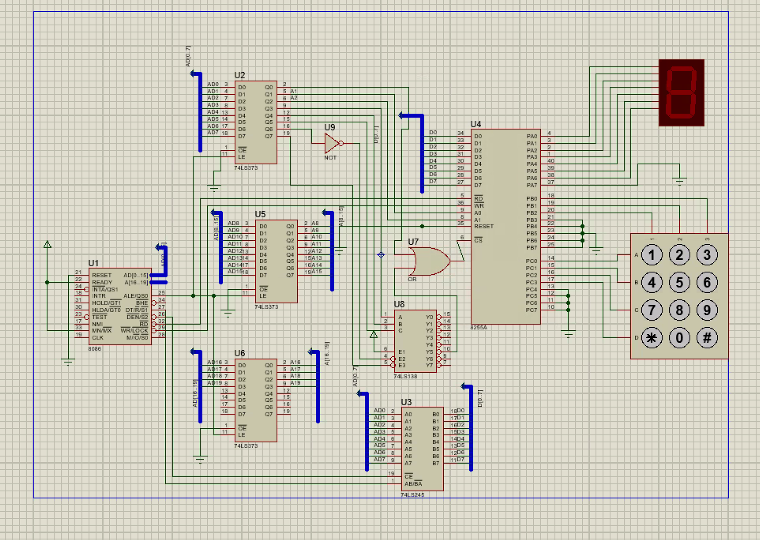


Figure :Hardware connection of the circuit.

Figure 1 illustrates the high level design simulated using Proteus. showing all hardware components connected to each other as follows:

### **8086 microprocessor**

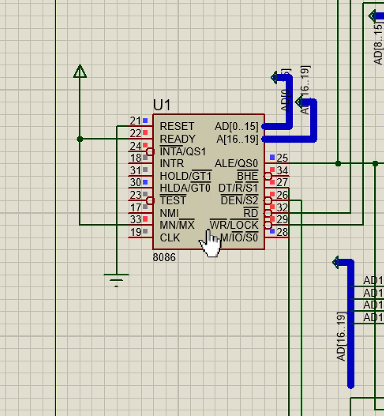


Figure :Connections of the 8086 microprocessor.

Figure 2 illustrates the setup of the 8086 microprocessor in minimum mode, with the RESET pin grounded and the READY and MN/MX pins connected to 5V. The Address/Data Bus is interfaced with a 74LS245 transceiver and three latches (U2, U5, and U6). The DT and DEN’ pins control the transceiver's data signals, where DEN’ enables the transceiver and DT determines the data direction. The ALE pin connects to the latches to enable them. Finally, the WR and RD pins are connected to the 8255 PPI, enabling it to perform the desired operations.

### **74LS245 transceiver**

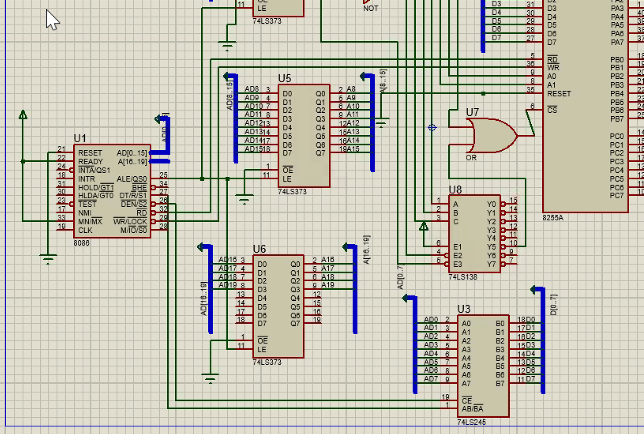


Figure :The transceiver’s connections.

Figure 3 shows the transceiver interfaced with both the 8086 microprocessor and the 8255 PPI, acting as a buffer for the 8086’s input data and preventing data conflicts during transmission.

### **Latches and decoder**

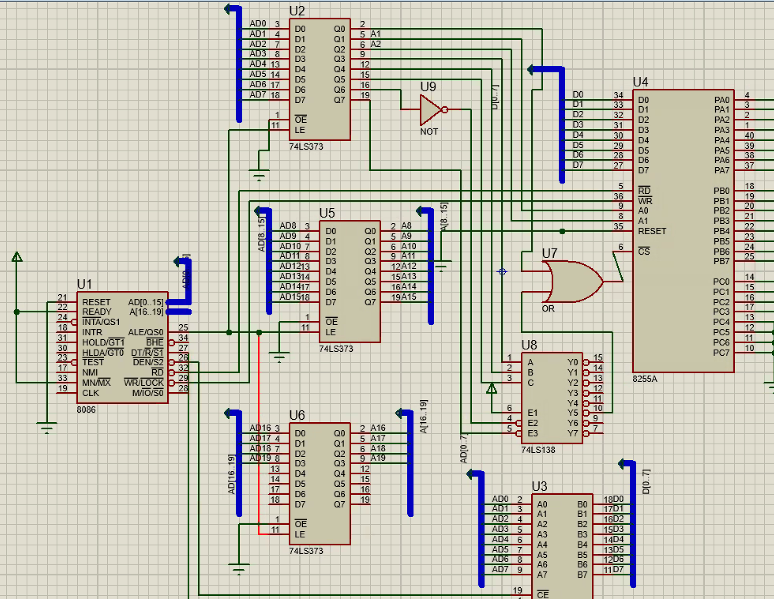


Figure : The connections of the latches.

Figure 4 shows three latches interfaced to both the 8086 microprocessor and 8255 PPI, preventing data loss due to the speed difference between the IO device and the microprocessor.

The first latch referred to as U2 in the figure interfaces the first 8 AD (AD0-AD7) bits from the microprocessor to a decoder.

The second latch referred to as U2 in the figure interfaces the second 8 AD (AD8-AD15) bits from the microprocessor with the 8255 PPI’s data bus.

The third latch referred to as U6 in the diagram interfaces the rest of the 8086’s AD bus.

**Decoding circuit:**

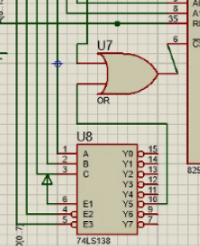
****

Figure : The decoding circuit’s connections and design.

To ensure the 8255 PPI operates only at the desired address (68h), a decoding circuit was implemented and connected to the CS' pin of the 8255. This circuit uses an OR gate between A0 and Y5 of the 74LS138 decoder. The desired address, 01101000 (68h), considers the first three bits (A0 for even/odd banking and A1, A2 for the 8255 address lines), so the decoding circuit starts at bit 4 . I will start the decoding circuit from bit 4, setting its inputs to 101. To ensure the chip outputs zero when bits 4-6 are 101 since it is an active low decoder. Additionally, I configure A6 and A7 to work with the enable pins, depending on the chip is designed as some enables are active high and others are active low.

### **IO devices and 8255 PPI module**

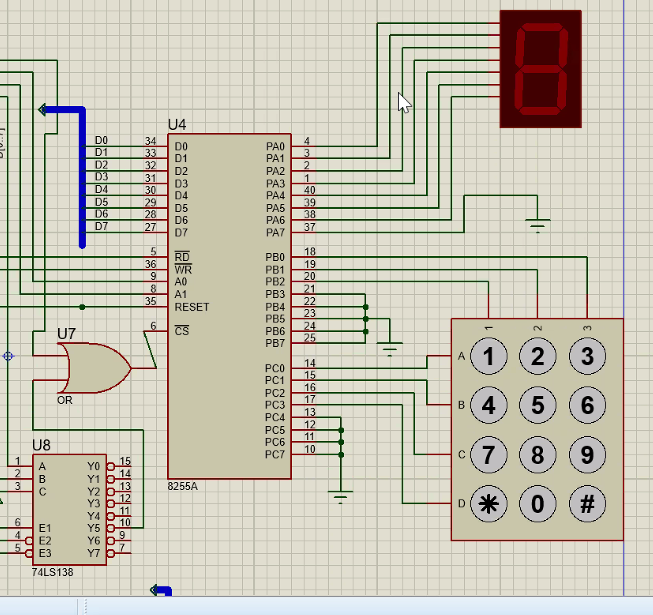


Figure : The connections of the I/O devices along with the 8255.

Figure 6 illustrates the seven-segment display interfaced to port A bus of the 8255 PPI, acting as an output for the system to display the typed digit on the keypad.

The keypad is enterface to both port B and port C of the 8255 PPI, where port B is programed as an output port providing signals on each column of the keypad which helps port C, programed as an input, to collect the signal from each row of the keypad and help the microprocessor identify which key is pressed. Some pins from both port B and C are connected to GND as they’re not needed in this implementation.

The communication between the microprocessor and the PPI is controlled by the pins RD and WR to decide whether a read or write operation is to happen; the pins A0 and A1 are connected to the first latch bus to control the port selection.

## software design

### 2.2.1 **Discussion**

This 8086 assembly code implements a simple keypad scanner interfaced with a seven-segment display using an 8255 PPI. The code continuously scans a 4x3 matrix keypad and displays the pressed key on the seven-segment display. The following steps describe the code:

### **Initialization**

The code initializes the 8255 PPI by sending the control word 88h to the control port. This configures Port A as output (for the seven-segment display), Port B as output (for column scanning), and the upper nibble of Port C as input (for row reading).It also set all of the ports in mode 0.



Figure ; Code for setting the 8255.

1. **Keypad Scanning**

The code implements a column scanning routine. It iterates through the three columns of the keypad, one at a time. For each column:

It sets the corresponding bit in Port B high to activate that column.

It reads the upper nibble of Port C to check for a keypress in that column.

If a key is pressed (a bit in Port C is high), it jumps to the appropriate column subroutine (column1, column2, or column3) to determine the specific row where another subroutine (and therefore the key).

If no key is pressed in the current column, it moves to the next column.

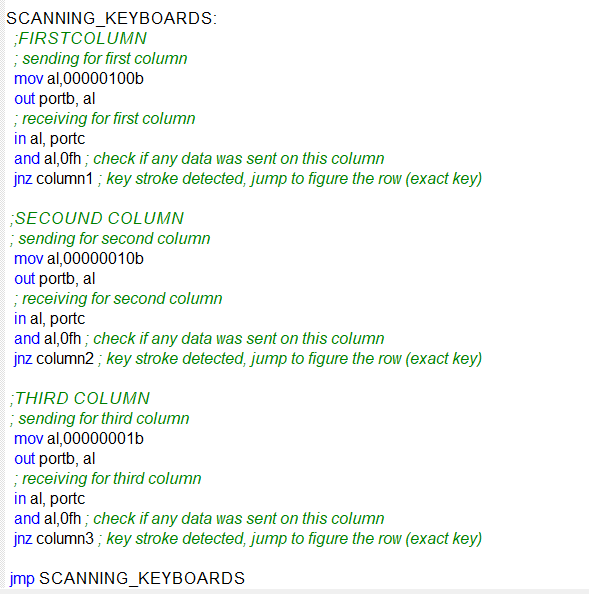


Figure :Code for deciding which column the key was pressed is in.

1. **Row Decoding**

Inside each column subroutine, the code checks which row the keypress occurred in by comparing the value read from Port C with the expected bit patterns for each value.

Based on the row, it sends the appropriate seven-segment display code to Port A to display the corresponding digit. Note that the code for 0 is handled under column2 after checking for 8. The provided codes represent the segments to be lit up for each digit.

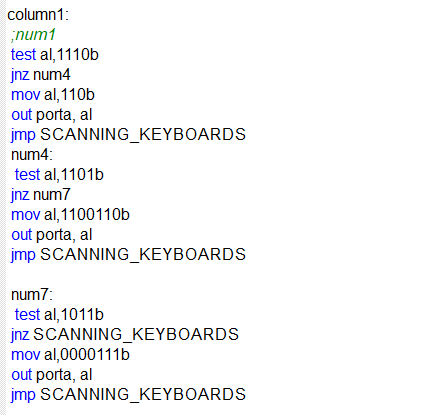


Figure : Code for deciding which row the key pressed is in.

1. **Continuous Scanning**

After displaying a digit, the code jumps back to SCANNING\_KEYBOARDS to continuously scan the keypad. This creates a loop that constantly monitors for keypresses.

### 2.2.2 **FlowChart**

The following figure describes the flow of the program:

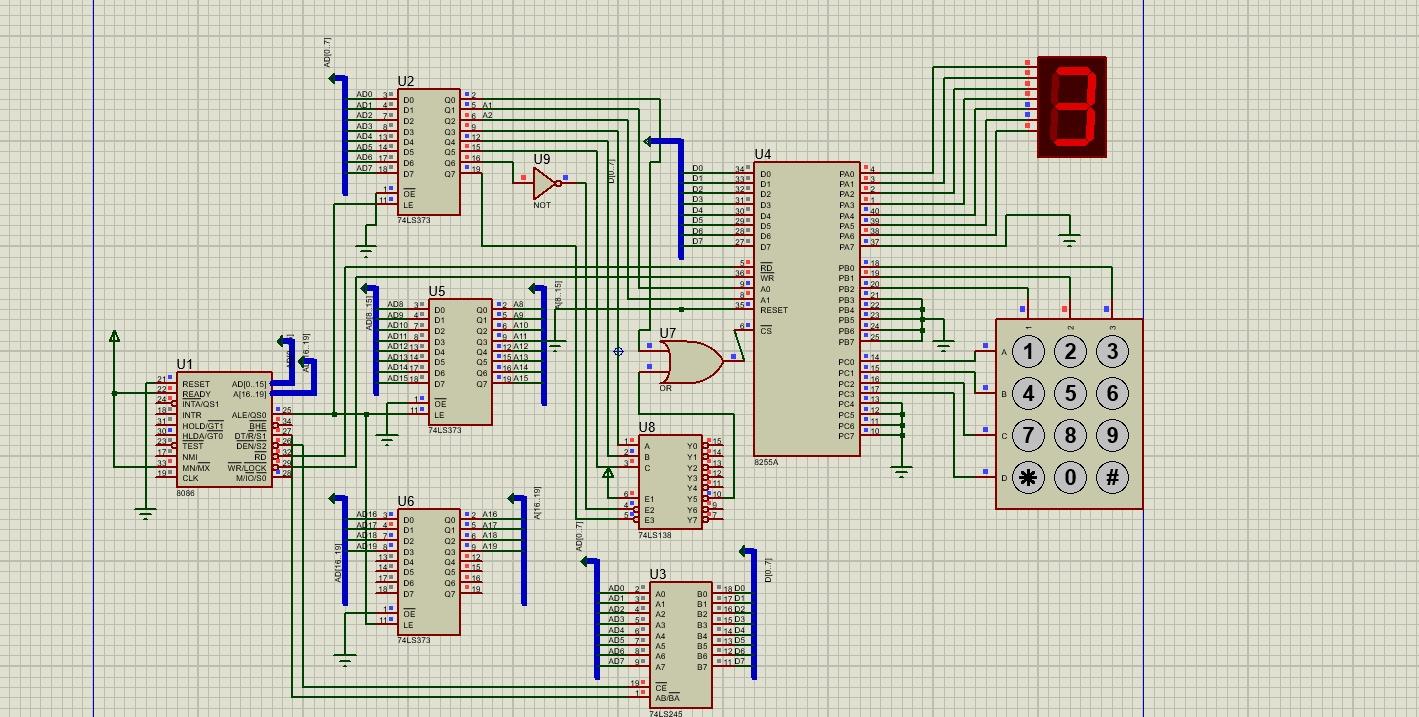
A diagram of a flowchart

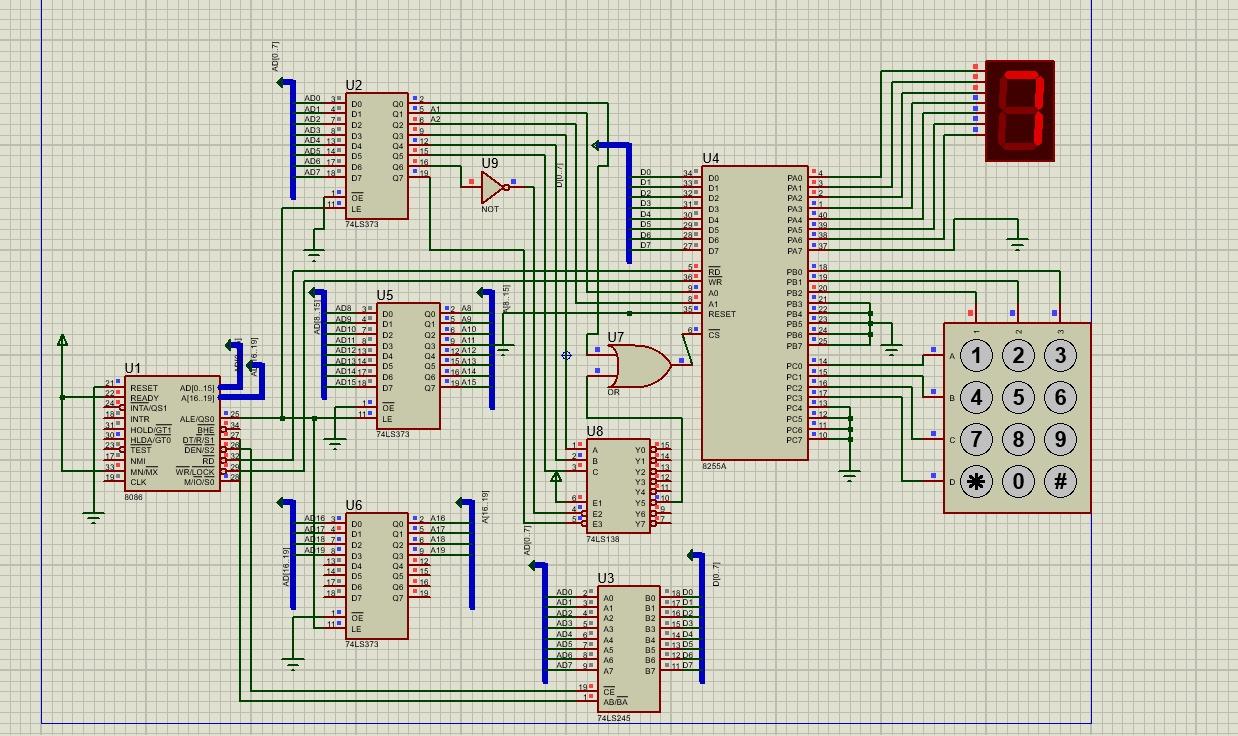
Description automatically generated

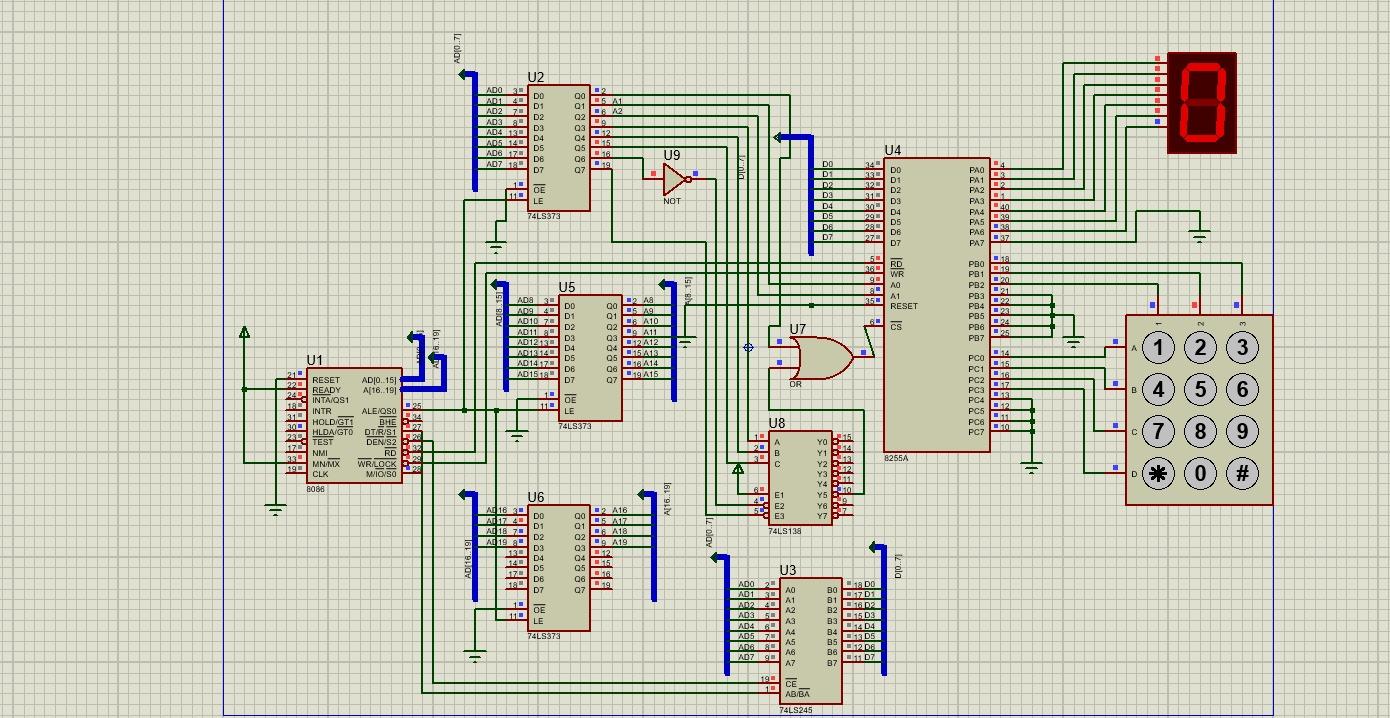
Figure :Flow chart of the program.

# Results and Discussions

In order to test the functionality of the implementation, Protouse has been used for simulating both the hardware and software implementation.The following figures show some of test cases, As seen, the project is fully functional as it served the purpose it was designed for: displaying the number pressed on the keypad onto the 7-segment display.







# Conclusions

This project successfully implemented a simplified keypad input system using an 8086 microprocessor, an 8255 PPI, and a seven-segment display. By utilizing the 8255 in mode 0 and implementing a scanning routine, we were able to detect key presses on a 4x3 keypad matrix. The decoded key value was then displayed on the seven-segment display, demonstrating the core functionality of interfacing input and output devices with the 8086. This project demonstrated core concepts in microprocessors including memory addressing, I/O port manipulation, and assembly language programming for embedded systems.

# References

* + **Rafiquzzaman, M.** (2005). *Microprocessor Theory and Applications with 8085 and 8086*. Wiley.
  + **Bhurchandi, K., & Ray, A. K.** (2006). *Advanced Microprocessors and Peripherals: Architecture, Programming and Interfacing*. Tata McGraw-Hill Education.
  + **Hall, D. V.** (2005). *Microprocessors and Interfacing: Programming and Hardware*. McGraw-Hill Education.
  + **Mazidi, M. A., Mazidi, J. G., & Causey, D.** (2011). *The 8086 Microprocessor: Programming and Interfacing the PC*. Pearson Education.