**School of Engineering**

**FAR WESTERN UNIVERSITY**

**Mahendranagar, Kanchanpur**

****

**Lab Report**

**“**Interfacing with 8086”

**“https://github.com/Dipakrajgiri/Interfacing-with-8086”**

|  |  |
| --- | --- |
| **Submitted By:**  **Dipak Raj Giri 13** | **Submitted to:**  **Er. Bipin Bhatt** |

**2080**

# ABSTRACT

This laboratory session aimed to provide hands-on experience in the field of instrumentation for computer engineering students. Under the guidance of Er. Bipin Bhatt, the practicals were conducted using Proteus software. The focus of the experiments included designing circuits for LED blinking, stepper motor control, seven-segment display interfacing, and Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC) simulations. Through this practical exposure, students gained valuable insights into the implementation of theoretical knowledge, circuit design, and programming for various instrumentation components. The use of Proteus software facilitated a virtual environment for simulating and testing these interfaces, contributing to a comprehensive understanding of instrumentation concepts.

# ACKNOWLEDGEMENT

I extend my sincere gratitude to Er. Bipin Bhatt for his guidance and support throughout the instrumentation practicals. His expertise and willingness to assist us in understanding complex concepts have been invaluable. I would also like to express my thanks to Er. Harendra Kalauni, the Head of the Department, for providing the necessary resources and fostering an environment conducive to practical learning.

I appreciate the opportunity to work on Proteus software, which enhanced our understanding of circuit design and interfacing in a virtual environment. This laboratory experience has been instrumental in bridging the gap between theoretical knowledge and practical application in the field of computer engineering.

Lastly, I would like to acknowledge the efforts of my fellow students and the university for providing a platform for hands-on learning, enabling us to develop essential skills in instrumentation.

Table of Contents

[**ABSTRACT 1**](#_Toc159575394)

[**ACKNOWLEDGEMENT 2**](#_Toc159575395)

[**1. Interfacing of Led 4**](#_Toc159575396)

[**Introduction: 4**](#_Toc159575397)

[**Circuit Diagram: 4**](#_Toc159575398)

[**Assembly Code: 4**](#_Toc159575399)

[**Result: 5**](#_Toc159575400)

[**2. Interfacing of Seven-segment display 6**](#_Toc159575401)

[**Introduction: 6**](#_Toc159575402)

[**Circuit Diagram: 6**](#_Toc159575403)

[**Assembly Code: 6**](#_Toc159575404)

[**Result: 8**](#_Toc159575405)

[**3. Interfacing of Stepper motor 9**](#_Toc159575406)

[**Introduction: 9**](#_Toc159575407)

[**Circuit Diagram: 9**](#_Toc159575408)

[**Assembly Code: 9**](#_Toc159575409)

[**Result: 12**](#_Toc159575410)

[**4. Interfacing of ADC 13**](#_Toc159575411)

[**Introduction: 13**](#_Toc159575412)

[**Circuit Diagram: 13**](#_Toc159575413)

[**Assembly Code: 13**](#_Toc159575414)

[**Result: 15**](#_Toc159575415)

[**5. Interfacing of DAC 16**](#_Toc159575416)

[**Introduction: 16**](#_Toc159575417)

[**Circuit Diagram: 16**](#_Toc159575418)

[**Assembly Code: 16**](#_Toc159575419)

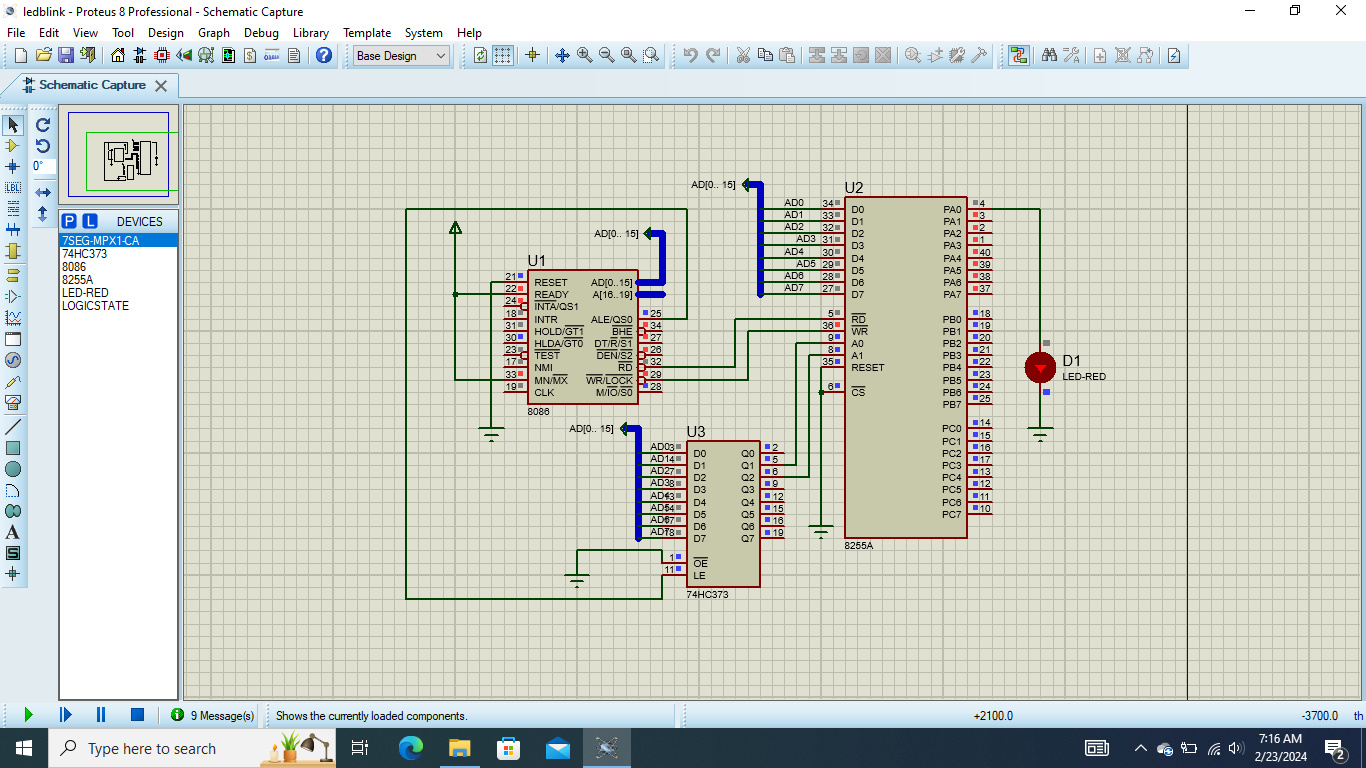
[**Result: 17**](#_Toc159575420)

# Interfacing of Led

## **Introduction:**

The provided assembly code is designed for LED interfacing on a microcontroller, utilizing Intel 8086 assembly language. The program employs port manipulation to control the state of a connected LED through PORTA. Additionally, the code configures the ports appropriately, setting PORTC and PORTA as outputs while configuring PORTB as an input. The control byte 10000000B is used to set the modes for PORTA and PORTC as output and PORTB as input.

## **Circuit Diagram:**



## **Assembly Code:**

DATA SEGMENT

PORTA EQU 00H

PORTB EQU 02H

PORTC EQU 04H

PORT\_CON EQU 06H

DATA ENDS

CODE SEGMENT

MOV AX,DATA

MOV DS, AX

ORG 0000H

START:

MOV DX, PORT\_CON

MOV AL, 10000000B

OUT DX, AL

JMP XX

XX:

MOV AL, 0000H

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

loopy1:

LOOP loopy1

MOV AL, 00FFH

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

loopy2:

LOOP loopy2

JMP XX

CODE ENDS

END

## **Result:**

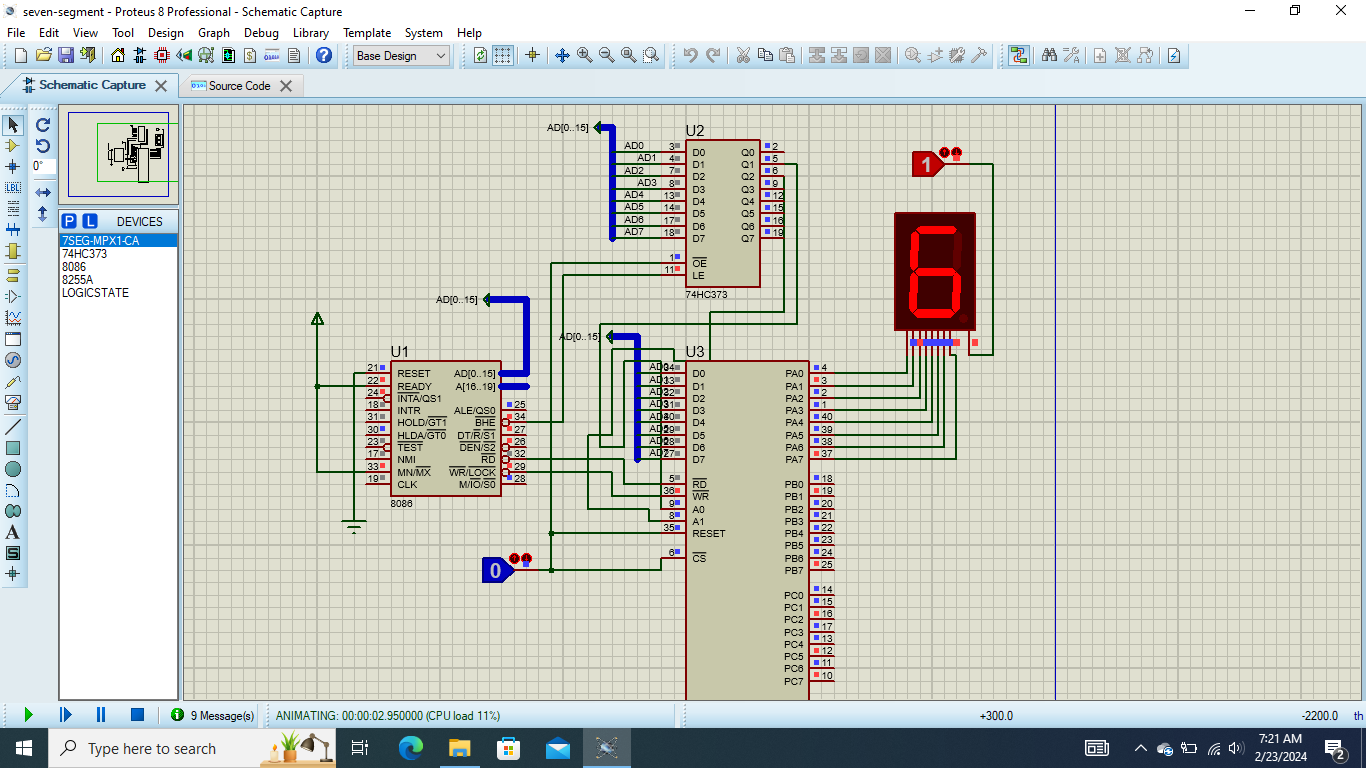
Upon successful execution of the code, the connected LED will blink alternately, transitioning between the ON and OFF states. This demonstration showcases the effective utilization of Intel 8086 assembly language for basic input-output operations and interfacing with electronic components.

# Interfacing of Seven-segment display

## **Introduction:**

The provided assembly code involves the interfacing of a seven-segment display with a microcontroller, utilizing Intel 8086 assembly language. The program configures the necessary ports and cyclically displays a sequence of binary-coded decimal (BCD) values on the seven-segment display, creating a visual pattern.

## **Circuit Diagram:**



## **Assembly Code:**

DATA SEGMENT

PORTA EQU 00H

PORTB EQU 02H

PORTC EQU 04H

PORT\_CON EQU 06H

DATA ENDS

CODE SEGMENT

MOV AX,DATA

MOV DS,AX

ORG 0000H

START:

MOV DX, PORT\_CON

MOV AL, 10000000B

OUT DX, AL

MOV SI, 0

MOV DI, 0

L0:

MOV CX, 1FFFH

L1:

MOV AL, S1[SI]

MOV DX, PORTA

OUT DX, AL

LOOP L1

INC SI

CMP SI, 16

JL L0

MOV DX, PORT\_CON

MOV AL, 10000000B

OUT DX, AL

JMP START

ORG 1000H

S1 DB 11000000B

DB 11111001B

DB 10100100B

DB 10110000B

DB 10011001B

DB 10010010B

DB 10000010B

DB 11011000B

DB 10000000B

DB 10010000B

DB 10001000B

DB 10000011B

DB 11000110B

DB 10100001B

DB 10000110B

DB 10001110B

CODE ENDS

END

## **Result:**

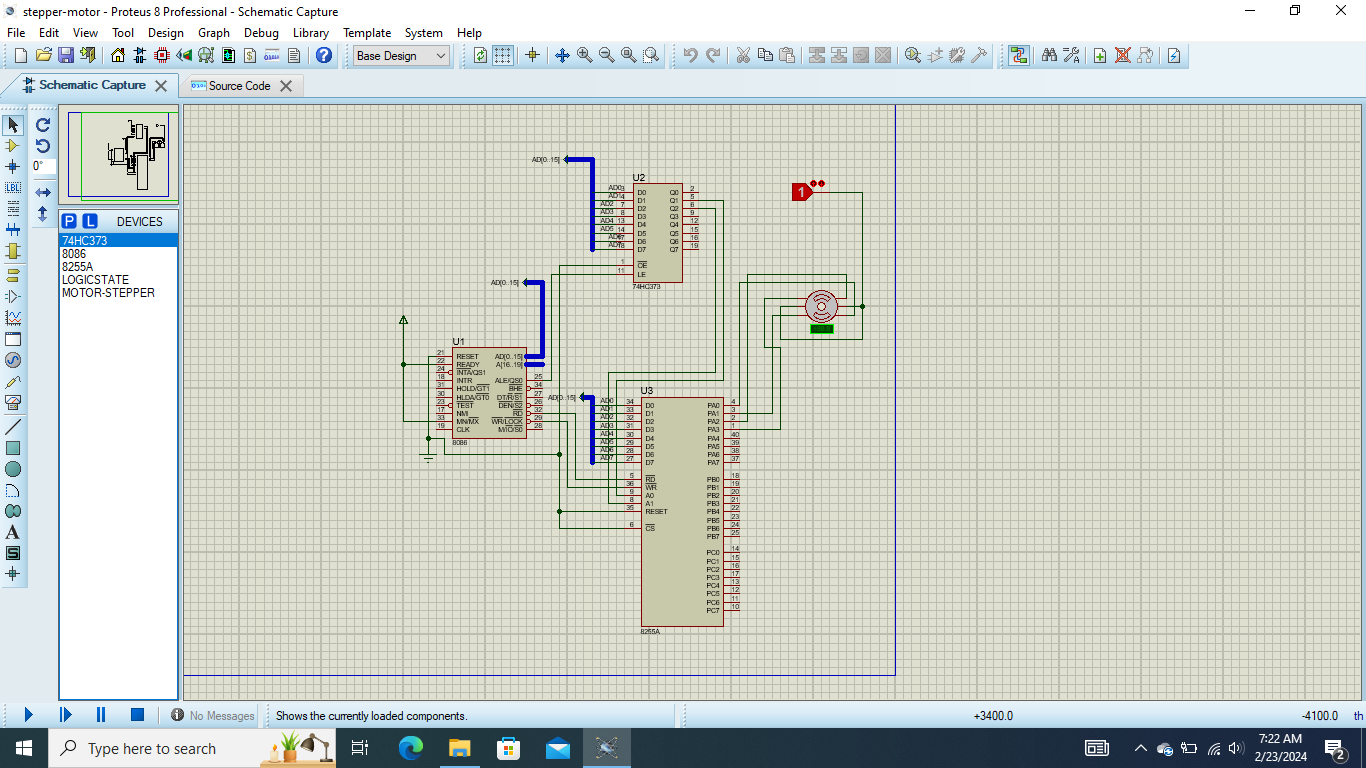
Upon successful execution of the code, the seven-segment display will cyclically show the BCD values stored in the array S1. Each BCD value will be displayed for a brief period, creating a visual pattern on the seven-segment display.

# Interfacing of Stepper motor

## **Introduction:**

The provided assembly code demonstrates the interfacing of a stepper motor with a microcontroller, utilizing Intel 8086 assembly language. The program configures the necessary ports and controls the stepper motor to rotate at specific angles, creating a rotational sequence. The motor is stepped at intervals of 45 degrees, showcasing a smooth motion in a full 360-degree rotation.

## **Circuit Diagram:**



## **Assembly Code:**

DATA SEGMENT

PORTA EQU 00H

PORTB EQU 02H

PORTC EQU 04H

PORT\_CON EQU 06H

DATA ENDS

CODE SEGMENT

MOV AX,DATA

MOV DS, AX

ORG 0000H

START:

MOV DX, PORT\_CON

MOV AL, 10000000B

OUT DX, AL

JMP XX

XX:

MOV AL, 03H ; At 0 degree

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay0: LOOP Delay0

MOV AL, 0BH ; At 45 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay1: LOOP Delay1

MOV AL, 0AH ; At 90 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay2: LOOP Delay2

MOV AL, 0EH ; At 135 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay3: LOOP Delay3

MOV AL, 0CH ; At 180 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay4: LOOP Delay4

MOV AL, 0DH ; At 225 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay5: LOOP Delay5

MOV AL, 05H ; At 270 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay6: LOOP Delay6

MOV AL, 07H ; At 315 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay7: LOOP Delay7

MOV AL, 03H ; At 360 degrees

MOV DX, PORTA

OUT DX, AL

MOV CX, 0DF36H; Delay

Delay8: LOOP Delay8

JMP XX

JMP START

CODE ENDS

END

## **Result:**

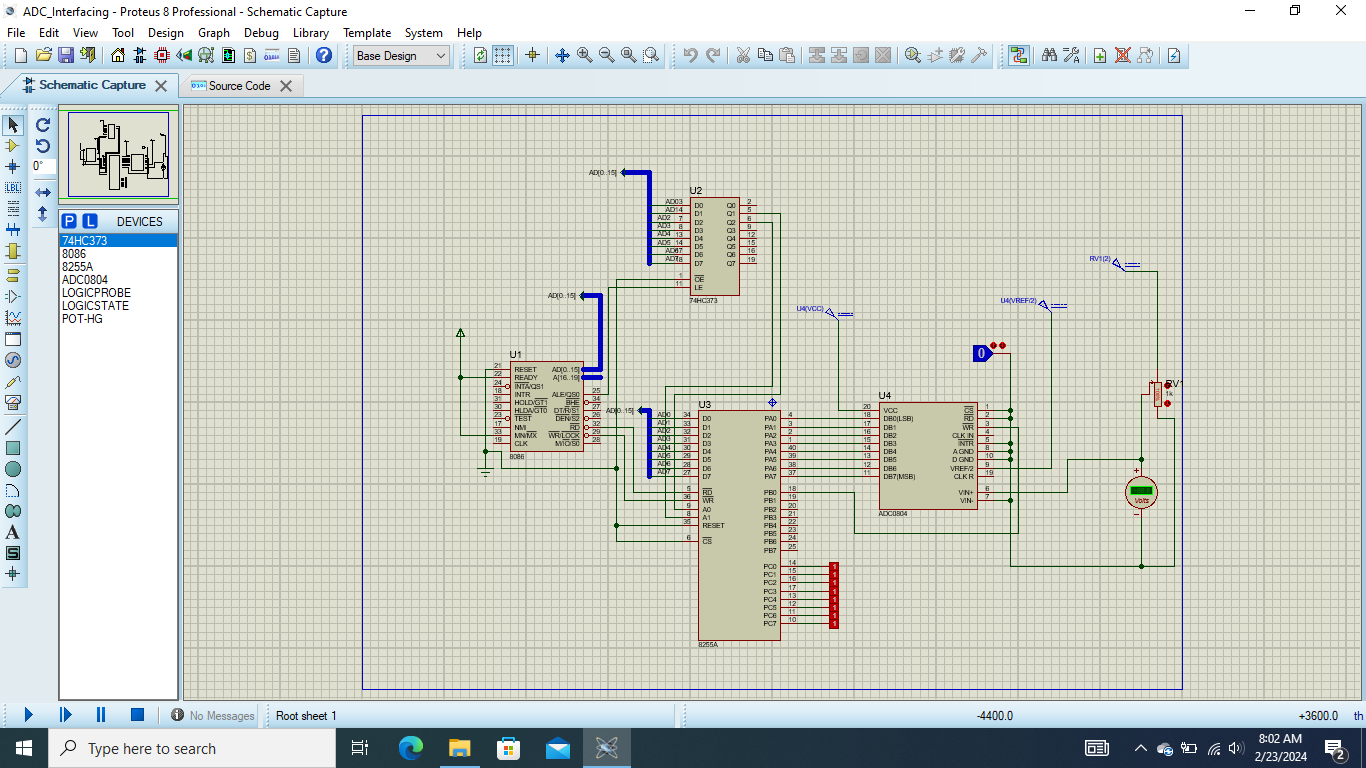
Upon successful execution of the code, the stepper motor will rotate in a sequence, stepping at intervals of 45 degrees and completing a full 360-degree rotation. This program showcases the control of a stepper motor through an Intel 8086 microcontroller, demonstrating its capability to perform precise and controlled rotational movements.

# Interfacing of ADC

## **Introduction:**

The provided assembly code illustrates the interfacing of an Analog-to-Digital Converter (ADC) with a microcontroller, utilizing Intel 8086 assembly language. The program configures the necessary ports, reads an analog input from PORTA, converts it to a digital value using the ADC, and displays the result on PORTC. The ADC reading is updated continuously in a loop, providing a simple example of ADC interfacing.

## **Circuit Diagram:**



## **Assembly Code:**

DATA SEGMENT

PORTA EQU 00H

PORTB EQU 02H

PORTC EQU 04H

PORT\_CON EQU 06H

DATA ENDS

CODE SEGMENT

MOV AX,DATA

MOV DS, AX

ORG 0000H

START:

MOV DX, PORT\_CON

MOV AL, 10010000B

OUT DX, AL

MOV AL, 00H

XX:

MOV DX, PORTA

IN AL, DX

MOV DX, PORTC

OUT DX, AL

MOV DX, PORTB

MOV AL, 00000000B

OUT DX, AL

MOV CX, 0FFH

D1: LOOP D1

MOV DX, PORTB

MOV AL, 00000001B

OUT DX, AL

MOV CX, 0FFH

D2: LOOP D2

JMP XX

CODE ENDS

END

## **Result:**

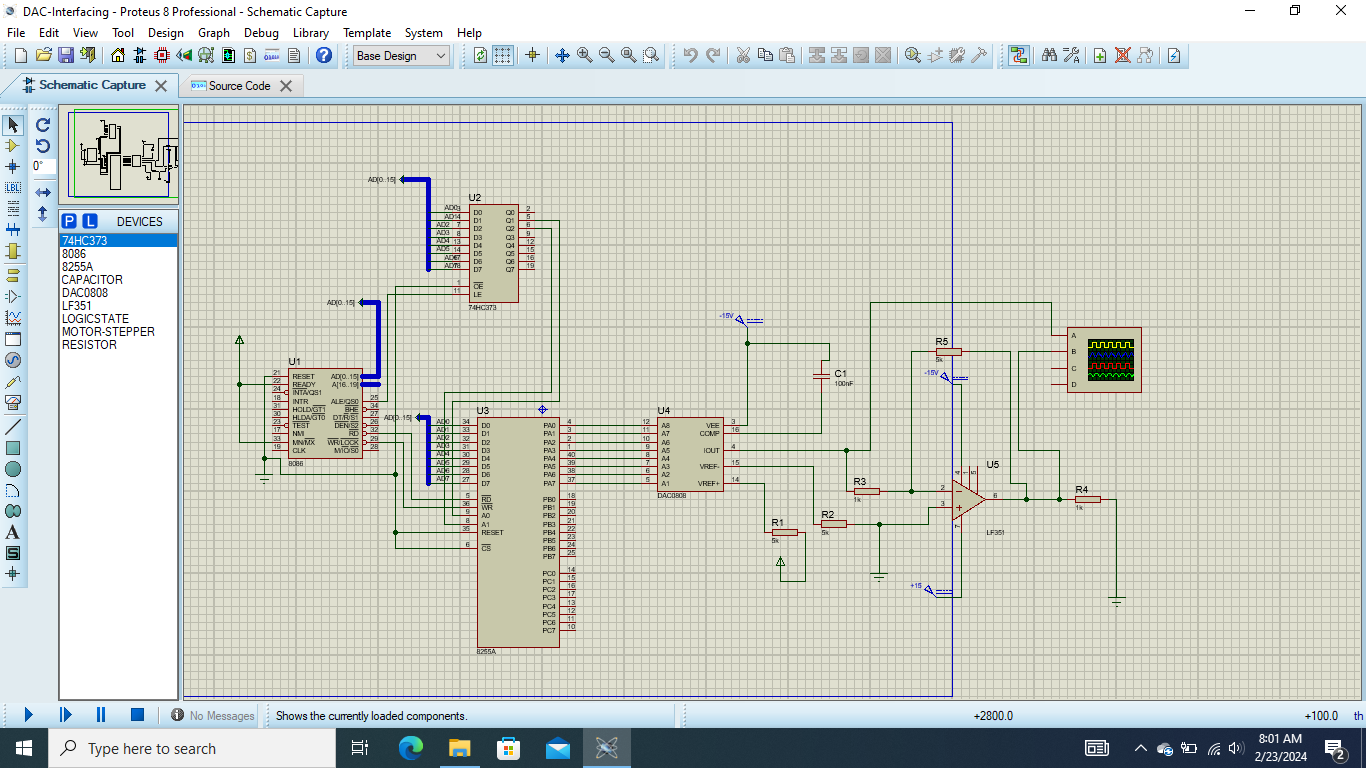
Upon successful execution of the code, the program continuously reads the analog input from PORTA, converts it to a digital value using the ADC, and displays the result on PORTC. The code is structured in a loop, updating the ADC reading and displaying it on PORTC at regular intervals. This simple example demonstrates the basic functionality of interfacing an ADC with a microcontroller.

# Interfacing of DAC

## **Introduction:**

The provided assembly code demonstrates the interfacing of a Digital-to-Analog Converter (DAC) with a microcontroller, utilizing Intel 8086 assembly language. The program configures the necessary ports and continuously outputs a binary pattern to PORTA, simulating the generation of an analog voltage through the DAC.

## **Circuit Diagram:**



## **Assembly Code:**

CODE SEGMENT

PORTA EQU 00H ; Address of Port A = 00H

Config EQU 06H ; Address of Config. Word = 06H

ORG 100H

MOV DX, Config

MOV AL, 10000000B; Port C=output, port A=output in mode 0, PORT B=output in mode 0

OUT DX, AL

START:

MOV AL, 00000000B; All pins of port A will be (0)

MOV DX, PORTA

OUT DX, AL

;--------------------

MOV CX, 0FFH ; Small Delay

loopy1:

LOOP loopy1

;--------------------

MOV AL, 11111111B; All pins of port A will be (1)

MOV DX, PORTA

OUT DX, AL

;--------------------

MOV CX, 0FFH ; Small Delay again

loopy2:

LOOP loopy2

;--------------------

JMP START ; Loop forever

CODE ENDS

END

## **Result:**

Upon successful execution of the code, the program continuously generates a binary pattern on PORTA, simulating the output of a Digital-to-Analog Converter. This pattern represents a changing analog voltage, demonstrating the basic functionality of interfacing a DAC with a microcontroller.