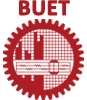
**BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY**



**Department of Electrical and Electronic Engineering**

**Course No. :** EEE 416

**Course Title:** Microprocessor and Interfacing Laboratory

**Input Output Interface to 8086 Microprocessor**

**Stepper Motor Control and FND Display**

**Name:** Mir Sayeed Mohammad

**ID:** 1606003

**Level:** 4

**Term:** 1

**Section:** A

**Submission Deadline:** 03 - 06 -2021

**Problem 1**

Modify the circuit diagram shown in lab to connect 3 units of 7 segment displays with the 8255. The pins a-g should all be connected in parallel to one port, while the common anode pin should be connected to separate pins of another port. Write a program that would display the last 3 digits of your roll number in the 7 segment displays. Attach the code, Proteus VSM Design file and a doc lab report with steps and screenshots.

**Solution:**

**Step 1: Blank Project Creation on Proteus**

After running Proteus 8.6 in administrator mode, ***New Project*** > and setting the project name and directory and hitting ***Next***

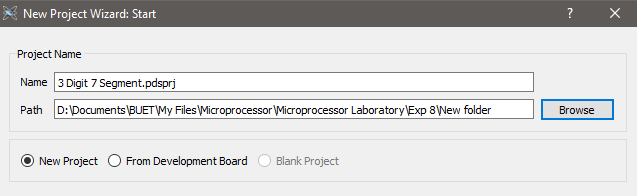


Fig: Creating new project

For ease of printing circuit schematic, setting schematic size to A4 and proceeding

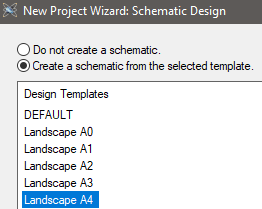


Fig: Schematic capture sheet size setup

No PCB layouts are required, so the selection has to be done accordingly

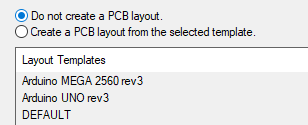


Fig: PCB layout selection pop-up

On the next screen, we have to select a firmware project in order to work with 8086 processor.

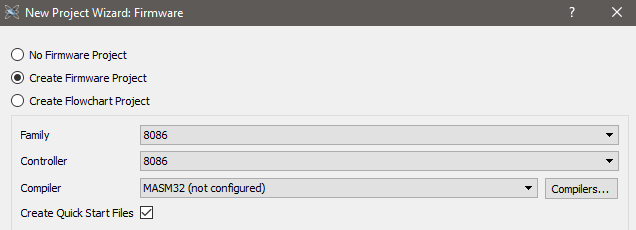


Fig: Firmware selection

After setting the parameters, we have to download the MASM32 assembly compiler by clicking on the ***Compilers*** button.

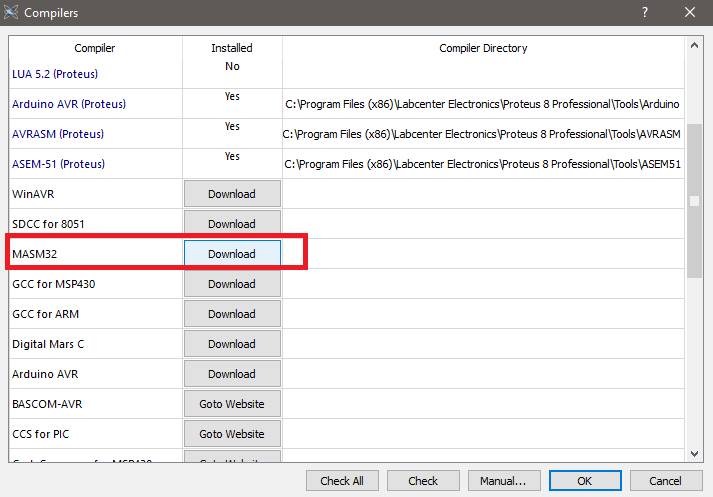


Fig: Downloading Microsoft ASM compiler

After download finishes, installation prompt for the compiler pops up on screen.

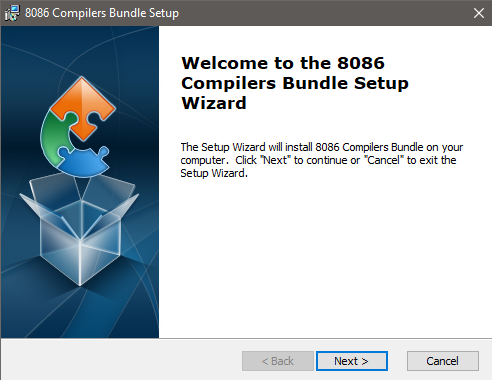


Fig: Compiler installation wizard

After installation finishes, we have to click ***OK*** and ***OK*** again to exit the compiler setup window.

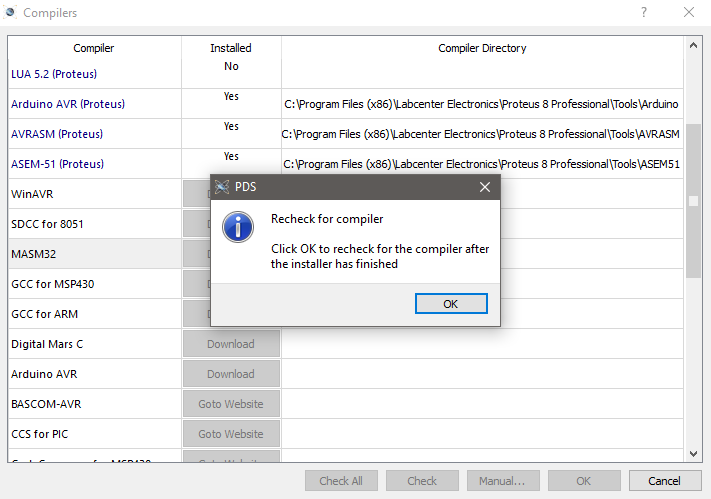


Fig: Successful installation of assembler

After finishing compiler setup, we proceed to the next screen and click on ***Finish***. A blank project has been created, ***Schematic Capture*** and ***Source Code*** tabs show up on the window.

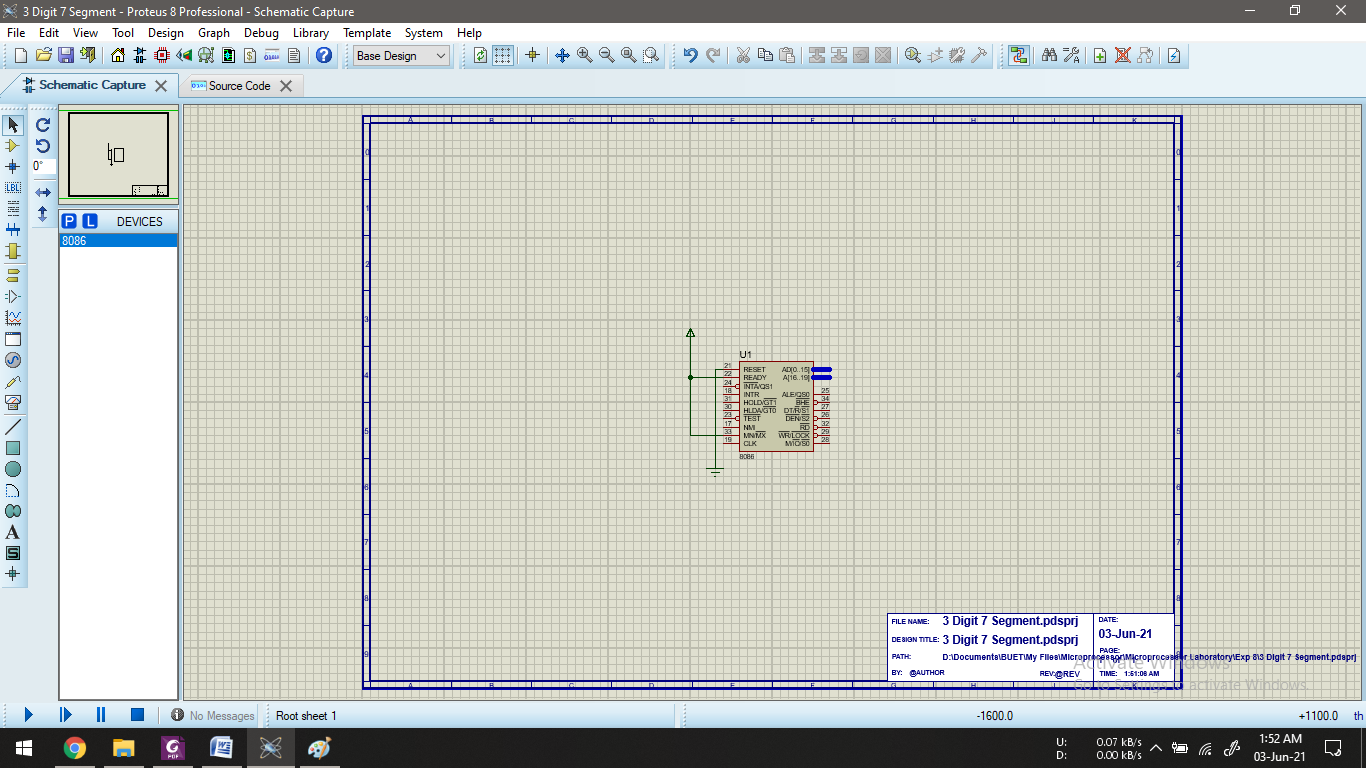


Fig: Schematic capture screen

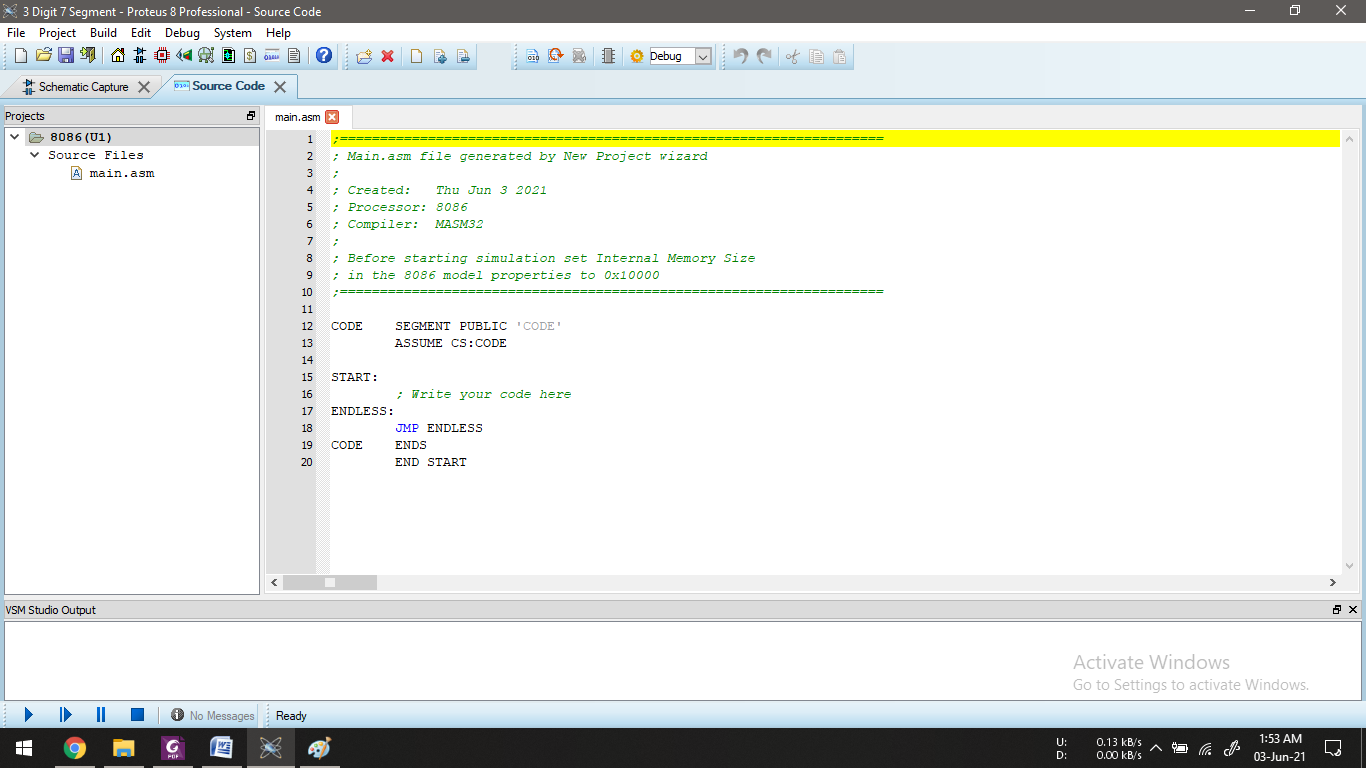


Fig: Source code (assembly) screen

**Step 2: Circuit Schematic Creation**

By pressing ***P*** on the keyboard, parts library shows up on top of the schematics. From there we have to find and add the following parts (double clicking on part names adds them to the components list):

* 74HC373
* 8255A
* 7-SEG-COM-ANODE

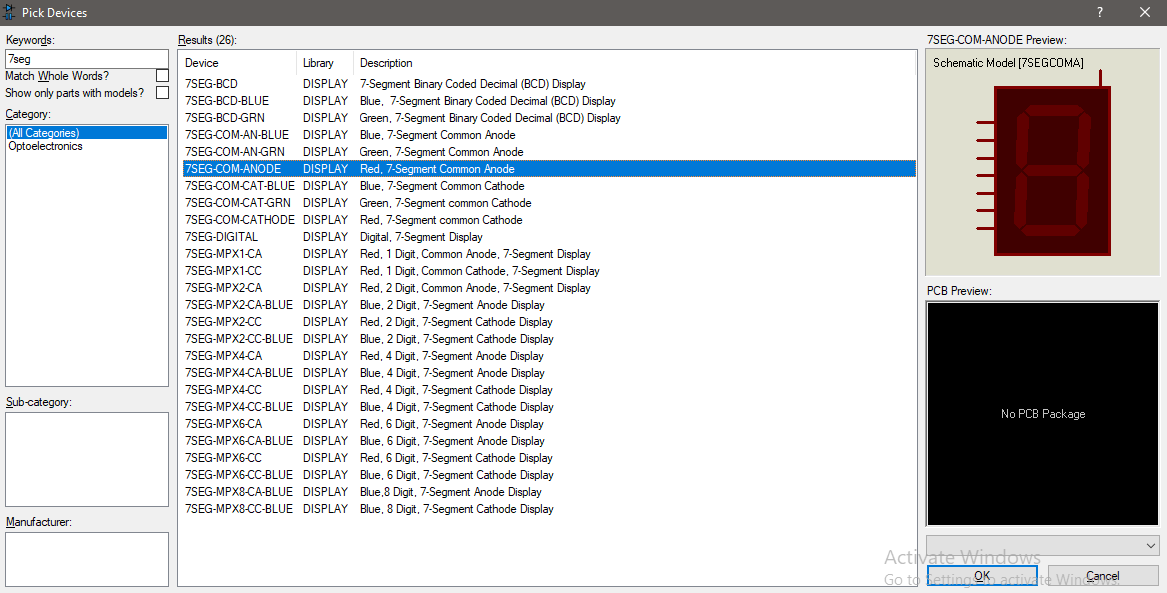


Fig: Getting components from library

From the components list, we have to pick and place the circuit components on the schematic sheet.

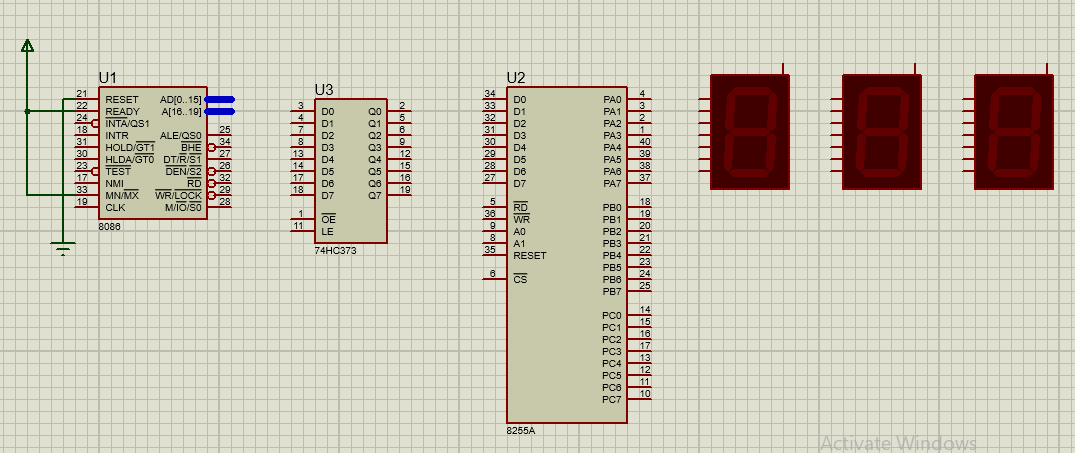


Fig: Components required for the simulation

After placing, we have to attach named wires and terminals to each of the components

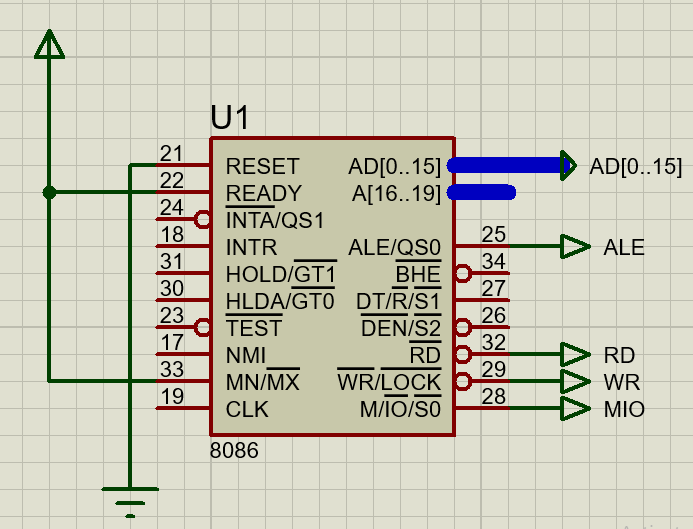


Fig: 8086 connections

For successful simulation, the internal memory size of the 8086 processor has to be defined from properties.

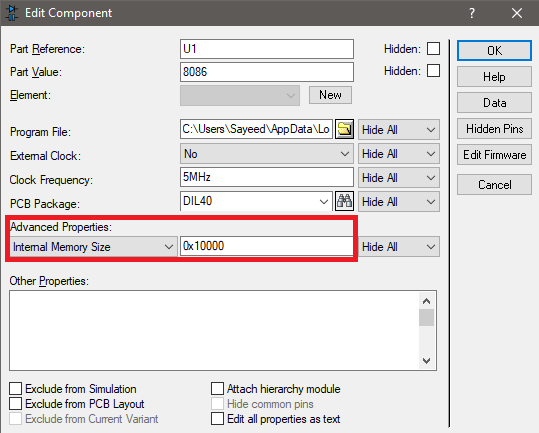


Fig: Setting internal memory for 8086 processor

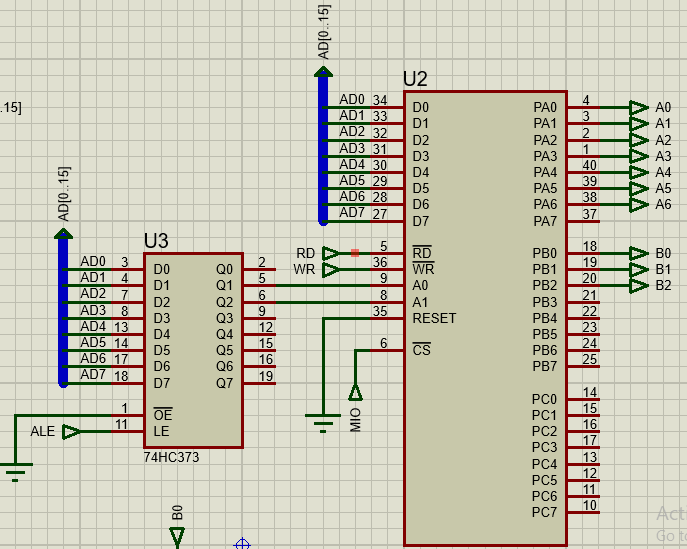


Fig: Buffer and 8255 Connections

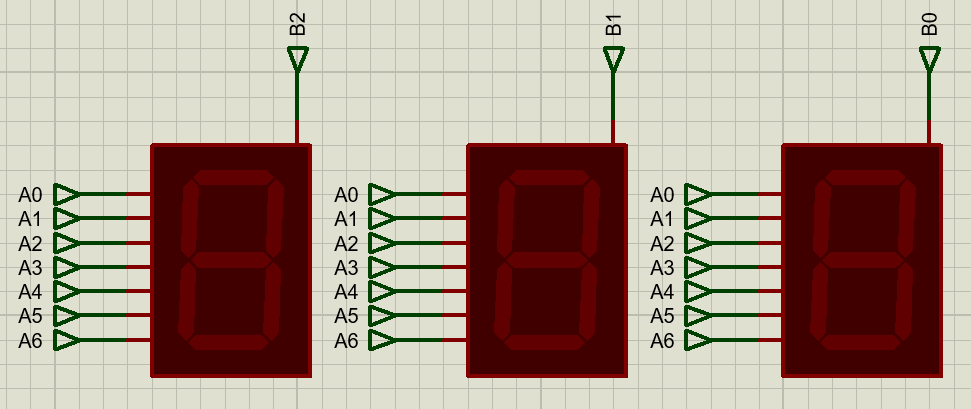


Fig: Seven segments connection for displaying 3 digits

With this, the circuit schematic is complete, and source code has to be provided for the 8086 processor.

**Step 3: Assembly Code**

;====================================================================

; Main.asm file generated by New Project wizard

;

; Created: Thu Jun 3 2021

; Processor: 8086

; Compiler: MASM32

;

; Before starting simulation set Internal Memory Size

; in the 8086 model properties to 0x10000

;====================================================================

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, ES:CODE, SS:CODE

; ----------------- SETUP ----------------- ;

ORG 0000H

MOV AX, DATA

MOV DS, AX

MOV AL, 10000000B ; configure all ports as output in mode 0

OUT PORT\_CON, AL

; ----------------- LOOP ----------------- ;

FOREVER:

MOV AL, FONT[3] ; selected digit 0 of roll '003'

OUT PORTA, AL ; output data through portA

MOV AL, 00000001B ; selecting anode 0

OUT PORTB, AL

MOV AL, FONT[0] ; selected digit 1 of roll '003'

OUT PORTA, AL ; output data through portA

MOV AL, 00000010B ; selecting anode 1

OUT PORTB, AL

MOV AL, FONT[0] ; selected digit 2 of roll '003'

OUT PORTA, AL ; output data through portA

MOV AL, 00000100B ; selecting anode 2

OUT PORTB, AL

JMP FOREVER

CODE ENDS

DATA SEGMENT

PORTA EQU 00H

PORTB EQU 02H

PORTC EQU 04H

PORT\_CON EQU 06H

; ----FONTS---- ;

FONT DB 11000000B ; 0

DB 11111001B ; 1

DB 10100100B ; 2

DB 10110000B ; 3

DB 10011001B ; 4

DB 10010010B ; 5

DB 10000010B ; 6

DB 11011000B ; 7

DB 10000000B ; 8

DB 10010000B ; 9

DB 10001000B ; A

DB 10000011B ; B

DB 11000110B ; C

DB 10100001B ; D

DB 10000110B ; E

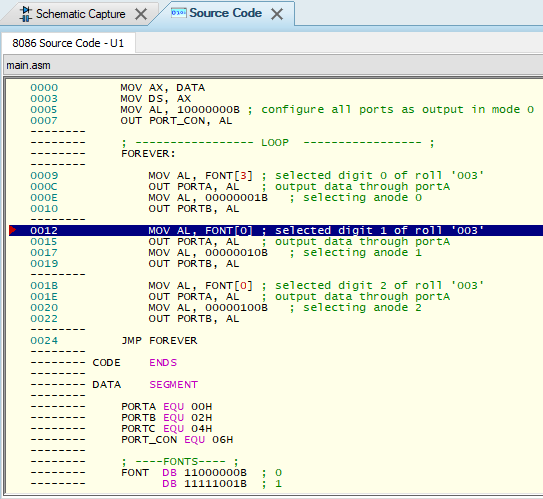
DB 10001110B ; F

DATA ENDS

END

**Result:**

The simulation does not show the digits in real time due to load on the laptop processor. Thus we show the simulation output in debug step mode to prove that it is indeed working.



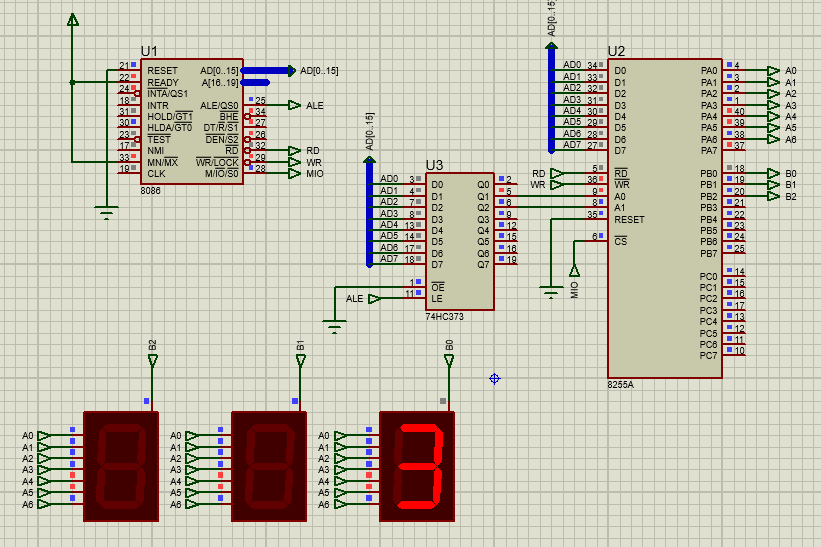


Fig: Digit no 0 showing on screen after instruction 0010

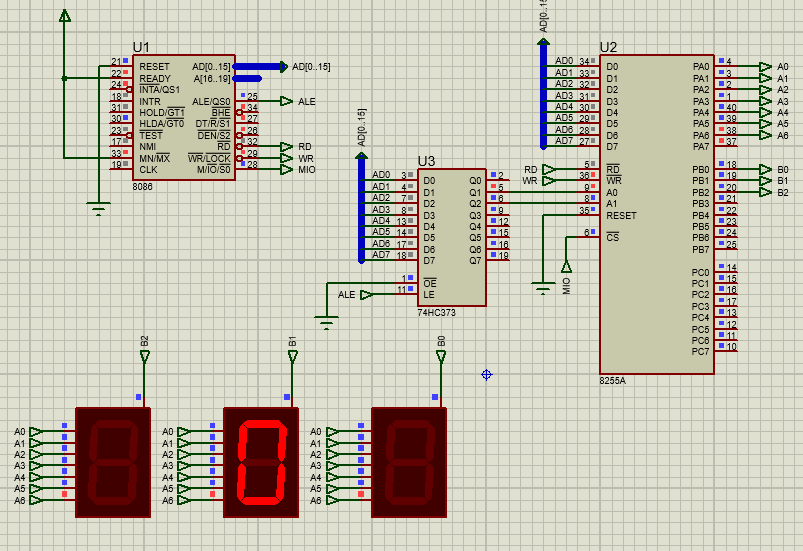
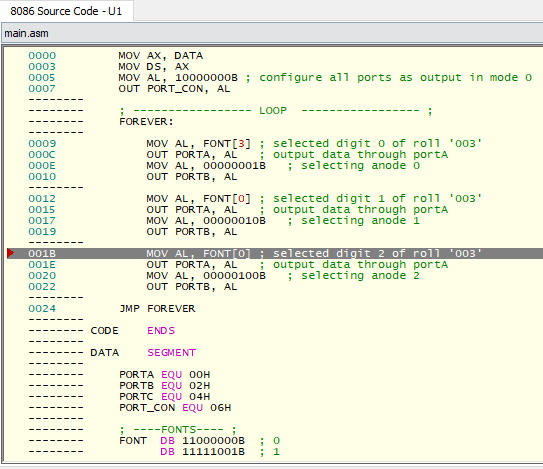


Fig: Digit no 0 showing on screen after instruction 0010

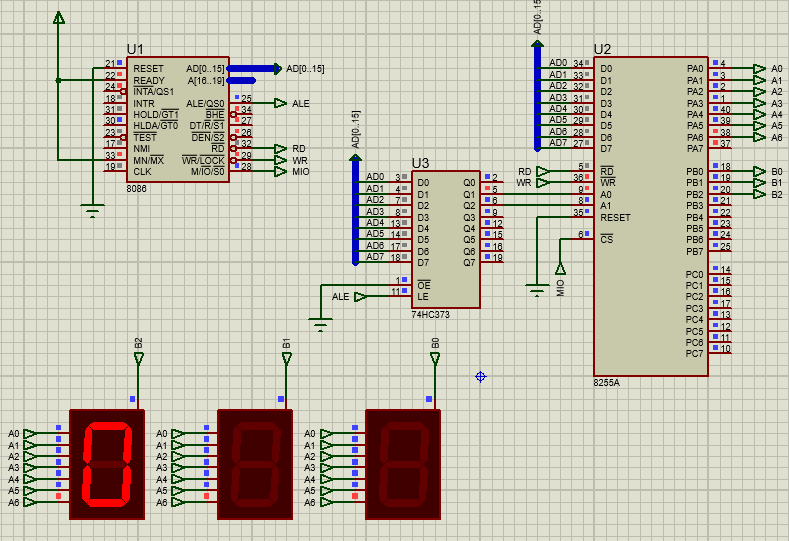
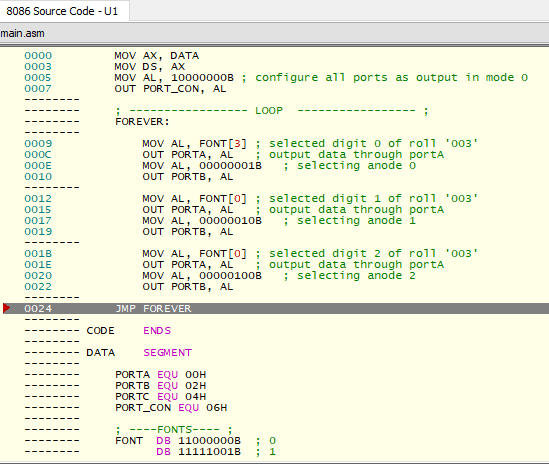


Fig: Digit no 0 showing on screen after instruction 0010

On real hardware, the process of switching between digits is really fast and the difference is not observable to the human eye, and as a result a continuous “003” seems to be appearing on the 3 seven segment displays. On simulation, the switching interval is not consistent and at the same time the simulation speed slows down drastically, so it could not be seen in the “Run Simulation” mode. However the circuit is indeed working.

**Discussion:**

The simulation despite being correct cannot show the full picture of the circuit in question due to the simulation lag. Although this would not cause issue on real hardware, the results could be improved in simulation using some other approaches. One such approach would be to connect the three segments to the 3 separate ports of the 8255 IC so that the LEDs do not turn off when we are switching segments. This would provide a consistent result. Another solution would be to use shift registers on the output stage of the 8255 IC that would simply load the values from PORTA and hold them until next loading. This would also give a consistent output. However this solution would increase circuit complexity.