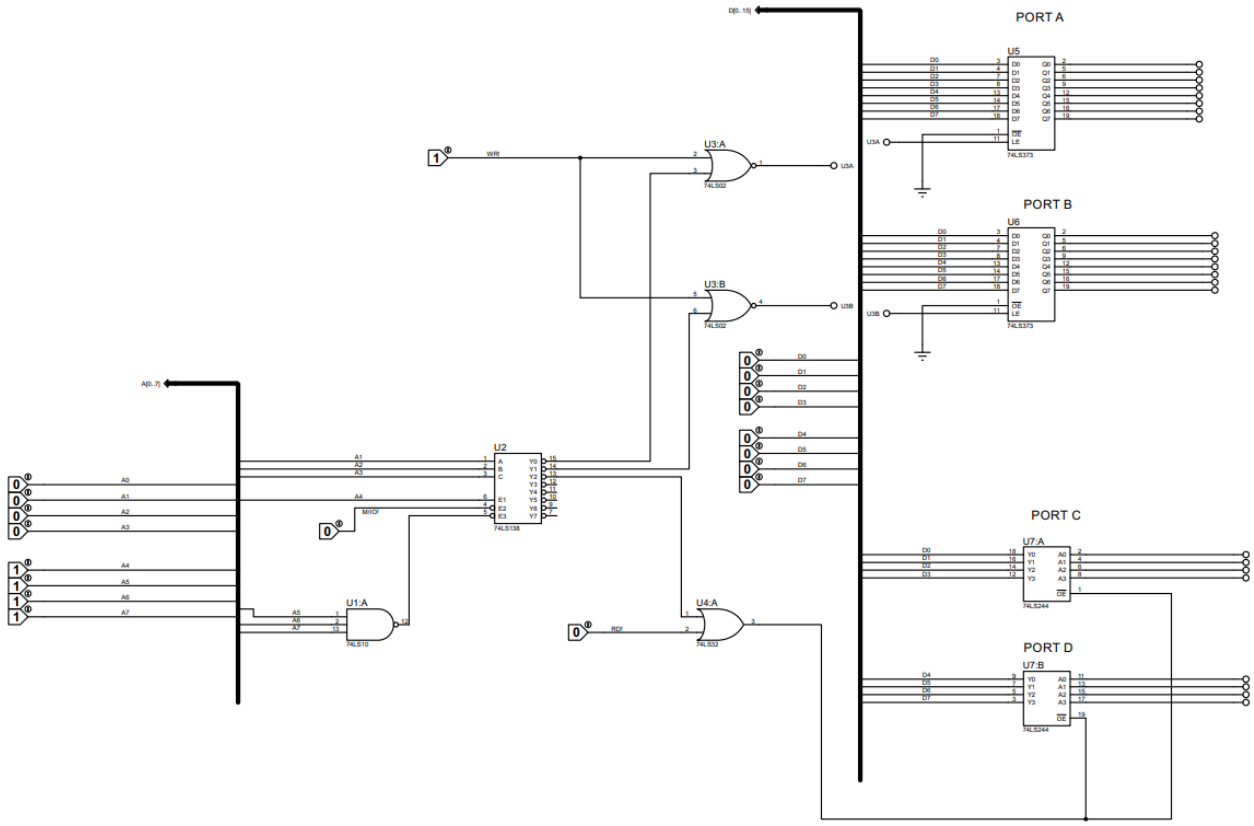
Laboratory Report

Laboratory Exercise No.:	5	**Date Performed:**	October 17, 2024
Laboratory Exercise Title:	I/O Interfacing (Address Decoding and I/O Ports)		
Name of Student:	Ivor Louisetyne Canque May G. Ochia	**Document Version:**	1.0

Activity #1



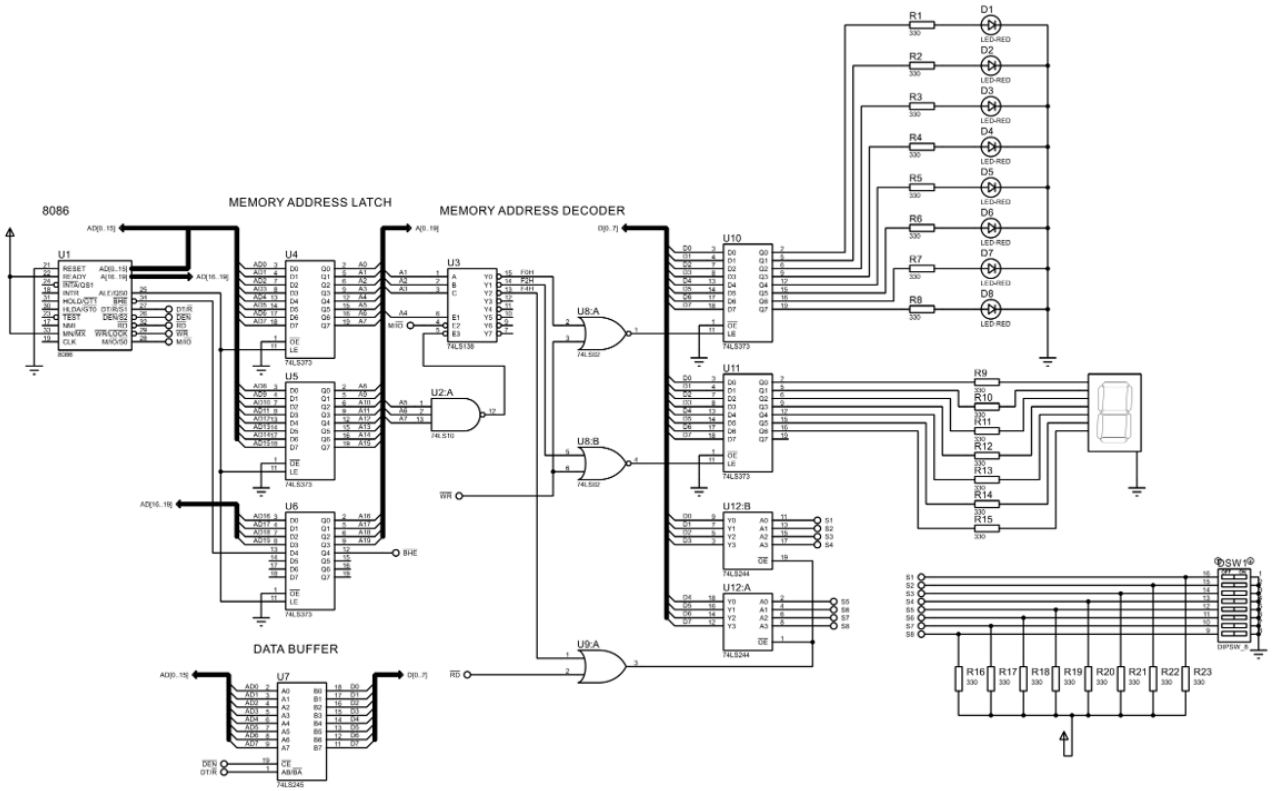
Address (A0-A7)	$\overline{WR}$	$\overline{RD}$	$M/\overline{IO}$	I/O Port enabled
F0H	0	1	0	PORT A (F0H)
F1H	1	0	0	None Enabled
F4H	0	1	0	None Enabled
F4H	0	1	1	None Enabled
F5H	1	0	0	PORT C (F4H)

Copyright 2024

F3H	0	1	0	PORT B (F2H)
F2H	0	1	1	None Enabled
02H	1	0	0	None Enabled
65H	0	1	1	None Enabled
F6H	1	0	0	Command Register

- Observe the data in Table 1. What is the role of the control lines , and in I/O address decoding?
    - The control lines determine which ports are to be enabled in regards with the lines enabled. M/IO’ must be at logic 0 for I/O operation to be done since it is an active low. Values may be placed at RD’ or WR’ for a read or write operation to be done.
- What do you think is the purpose of the latches and buffers?
    - Latches hold data until it's cleared. Buffers amplify the input signal's power without altering its value. These components are managed by addresses accessible via the address decoder. Because data and memory addresses share the same physical bus, mishandling could result in overwritten values, affecting the system's expected output.
- Based on the decoder circuit and I/O address range, is the I/O memory mapped or isolated? Why?
    - The I/O system in the circuit above is an isolated I/O thus it uses the same M/IO! control to determine if an operation is for I/O, the address range does not use the same range as that of the memory.

Activity #2

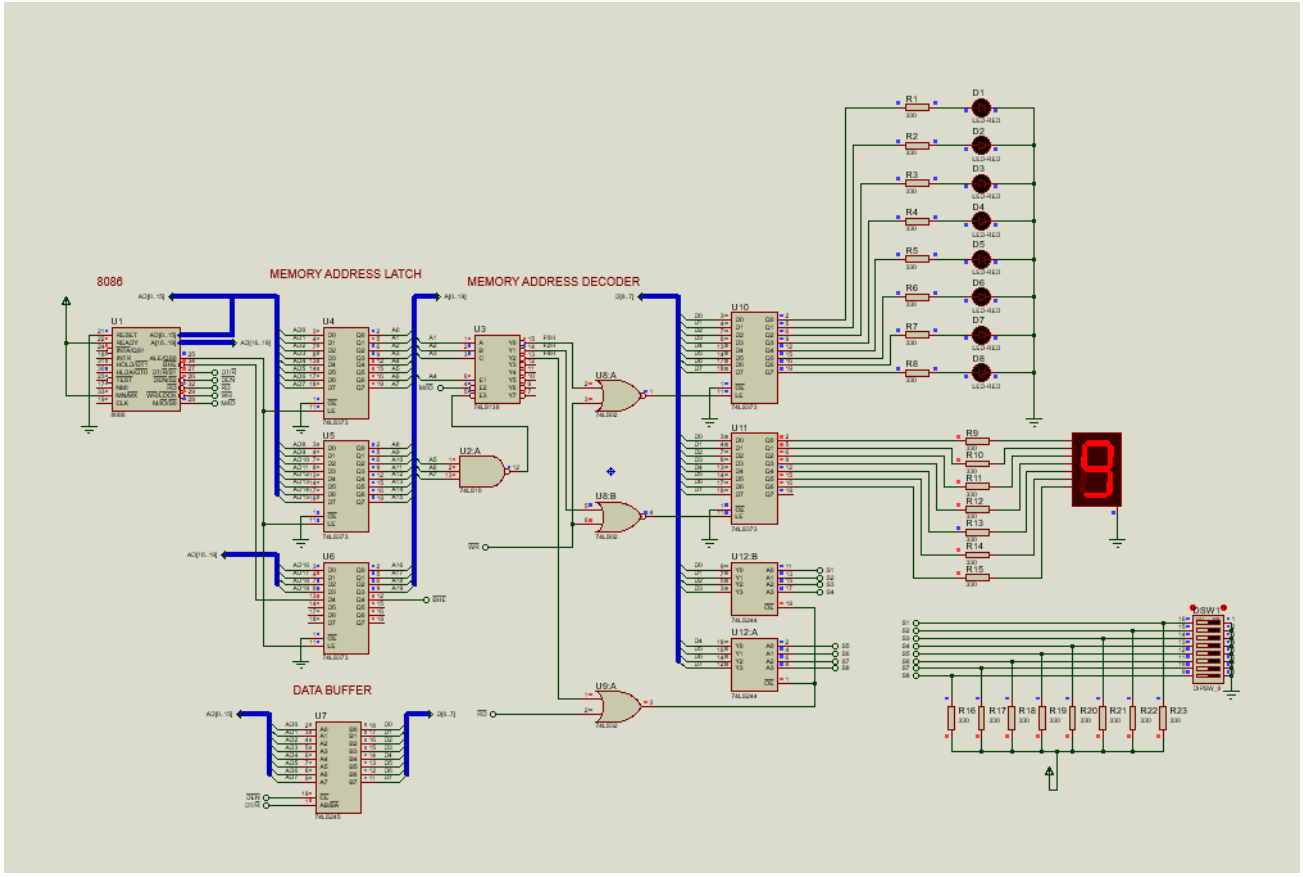


8086 I/O Interface using Latches and Buffer

main.asm

```
1   DATA SEGMENT
2
3    PORTA EQU 0F0H
4    PORTB EQU 0F2H
5    PORTC EQU 0F4H
6
7    DATA ENDS
8
9    CODE SEGMENT
10     MOV AX, DATA
11     MOV DS, AX
12     ORG 0000H
13
14    START:
15  MOV DX, PORTA
16  MOV AL, 00000000B
17  OUT DX, AL
18
19  MOV DX, PORTB
20  MOV AL, 01101111B
21  OUT DX, AL
22
23  HERE:
24  NOP
25  JMP HERE
26
27  CODE ENDS
28
29  END
```

Assembly Language Program for displaying ‘9’ in 7-segment Display

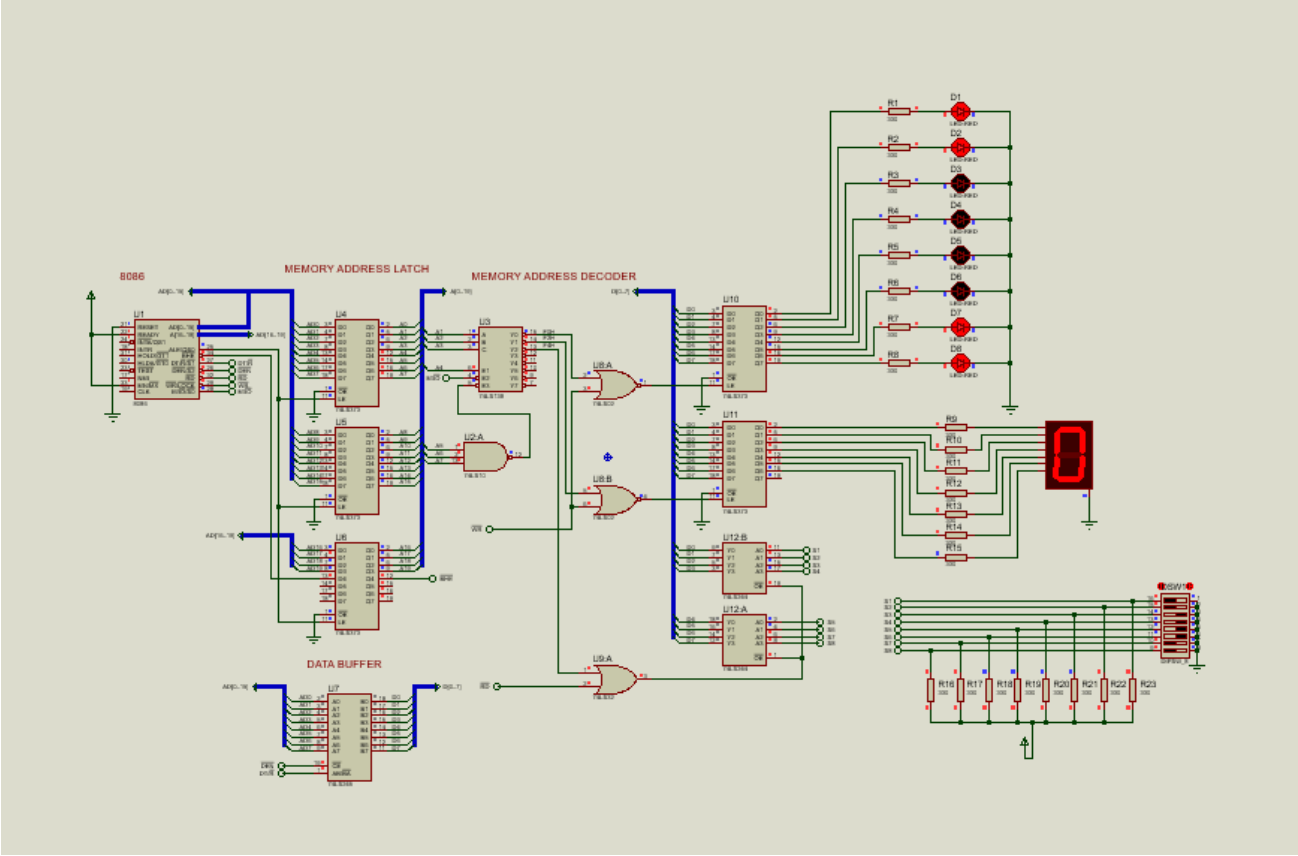


Output for the .asm Code displaying 9

main.asm

```
1   DATA SEGMENT
2   PORTA EQU 0F0H
3   PORTB EQU 0F2H
4   PORTC EQU 0F4H
5
6   DATA ENDS
7
8   CODE SEGMENT
9
10  MOV AX, DATA
11  MOV DS, AX
12  ORG 0000H
13
14  START:
15  MOV DX, PORTA
16  MOV AL, 00000000B
17  OUT DX,AL
18
19  MOV DX, PORTB
20  MOV AL, 00000000B
21  OUT DX, AL
22
23  MOV DX, PORTB
24  MOV AL, 00111111B
25  OUT DX, AL
26
27  HERE:
28
29  MOV DX, PORTC
30  IN AL, DX
31
32  MOV DX, PORTA
33  OUT DX, AL
34
35  JMP HERE
36
37  CODE ENDS
38
39  END
```

Assembly Language Program that Reads Data from Port C and Displays output at Port A



Output for the .asm that Reads Data from Port C and Displays output at Port A

main.asm

```
1   DATA SEGMENT
2       PORTA EQU 0F0H    ; Define PORTA address (used for LED control)
3       PORTB EQU 0F2H    ; Define PORTB address (used for 7-segment display)
4       PORTC EQU 0F4H    ; Define PORTC address (used for input detection)
5   DATA ENDS
6
7   CODE SEGMENT
8       MOV AX, DATA
9       MOV DS, AX
10      ORG 0000H
11
12  START:
13      ; Initialize PORTA (LED control port) to 0
14      MOV DX, PORTA
15      MOV AL, 00000000B ; Clear all LEDs initially
16      OUT DX, AL
17
18      ; Initialize PORTB (7-segment display control port) to 0
19      MOV DX, PORTB
20      MOV AL, 00000000B
21      OUT DX, AL
22
23  HERE:
24      MOV DX, PORTC
25      IN AL, DX       ; Read input from PORTC
26
27      ; If equal, jump to LED control
28      CMP AL, 01H
29      JE ON_LED
30
31      ; If equal, jump to 7-segment display control
32      CMP AL, 02H
33      JE ON_SEG
34
35      NOP           ; Do nothing if no input match
36      JMP HERE        ; Repeat the loop
37
38  ON_LED:
39      ; LED control sequence (shifts a single lit LED across PORTA)
40      MOV CX, 08H
41      MOV DX, PORTA
42      MOV AL, 10000000B ; Start with the leftmost LED turned on
43      OUT DX, AL
44
45      CALL DELAY
46
47  DISPLAY:
48      ; Shift LED to the right by one bit
49      SHR AL, 1H      ; Shift right one position
50      MOV DX, PORTA
51      OUT DX, AL
52      CALL DELAY
53      LOOP DISPLAY    ; Repeat until all LEDs have shifted
```

Assembly Language Input to Create a running LED pattern (single cycle) on Port A when data in Port C is 01H and display numbers 0-9 in Port B when data in Port C is 02H *(1/3)*

main.asm

```
56
57  ON_SEG:
58      ; 7-segment display control sequence (cycles through digits 0 to 9)
59      MOV DX, PORTB     ; Set up PORTB for display control
60
61      ; Display 0
62      MOV AL, 00111111B
63      OUT DX, AL
64      CALL DELAY
65
66      ; Display 1
67      MOV DX, PORTB
68      MOV AL, 00000110B
69      OUT DX, AL
70      CALL DELAY
71
72      ; Display 2
73      MOV DX, PORTB
74      MOV AL, 01011011B
75      OUT DX, AL
76      CALL DELAY
77
78      ; Display 3
79      MOV DX, PORTB
80      MOV AL, 01001111B
81      OUT DX, AL
82      CALL DELAY
83
84      ; Display 4
85      MOV DX, PORTB
86      MOV AL, 01100110B
87      OUT DX, AL
88      CALL DELAY
89
90      ; Display 5
91      MOV DX, PORTB
92      MOV AL, 01101101B
93      OUT DX, AL
94      CALL DELAY
95
96      ; Display 6
97      MOV DX, PORTB
98      MOV AL, 01111101B
99      OUT DX, AL
100     CALL DELAY
101
102     ; Display 7
103     MOV DX, PORTB
104     MOV AL, 00000111B
105     OUT DX, AL
106     CALL DELAY
107
108     ; Display 8
```

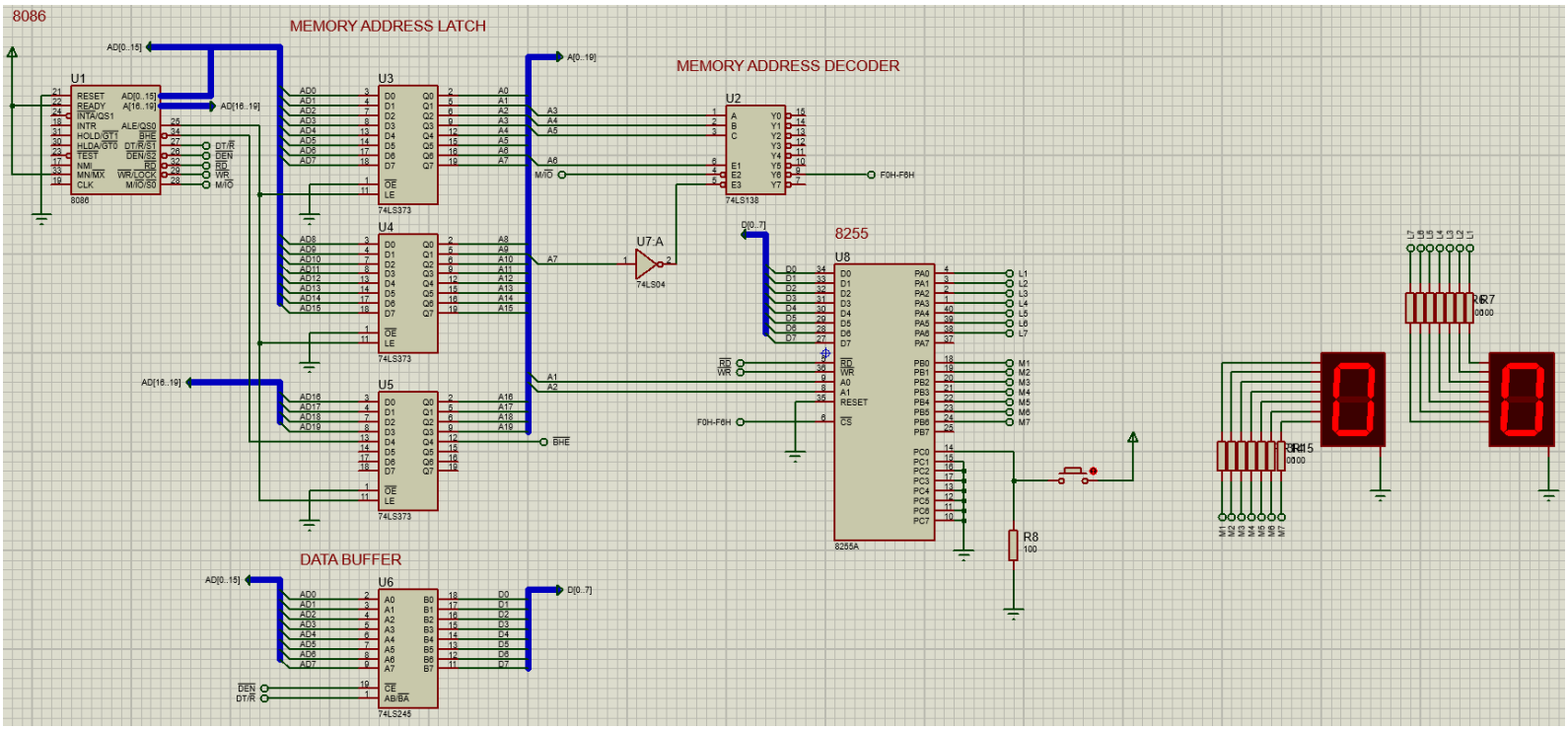
Assembly Language Input to Create a running LED pattern (single cycle) on Port A when data in Port C is 01H and display numbers 0-9 in Port B when data in Port C is 02H *(2/3)*

Copyright 2024

```
107
108         ; Display 8
109         MOV DX, PORTB
110         MOV AL, 01111111B
111         OUT DX, AL
112         CALL DELAY
113
114         ; Display 9
115         MOV DX, PORTB
116         MOV AL, 01101111B
117         OUT DX, AL
118         CALL DELAY
119
120         ; Clear the 7-segment display
121         MOV DX, PORTB
122         MOV AL, 00000000B ; Turn off all segments
123         OUT DX, AL
124
125         JMP HERE
126
127     ; Delay subroutine
128     DELAY PROC
129         MOV BX, 9FFFH
130       L1:
131           DEC BX
132         NOP
133         JNZ L1
134         RET
135     DELAY ENDP
136
137     CODE ENDS
138     END
139
```

Assembly Language Input to Create a running LED pattern (single cycle) on Port A when data in Port C is 01H and display numbers 0-9 in Port B when data in Port C is 02H *(3/3)*

Activity #3



Address: **PORTA:** F0H **PORTB:** F2H

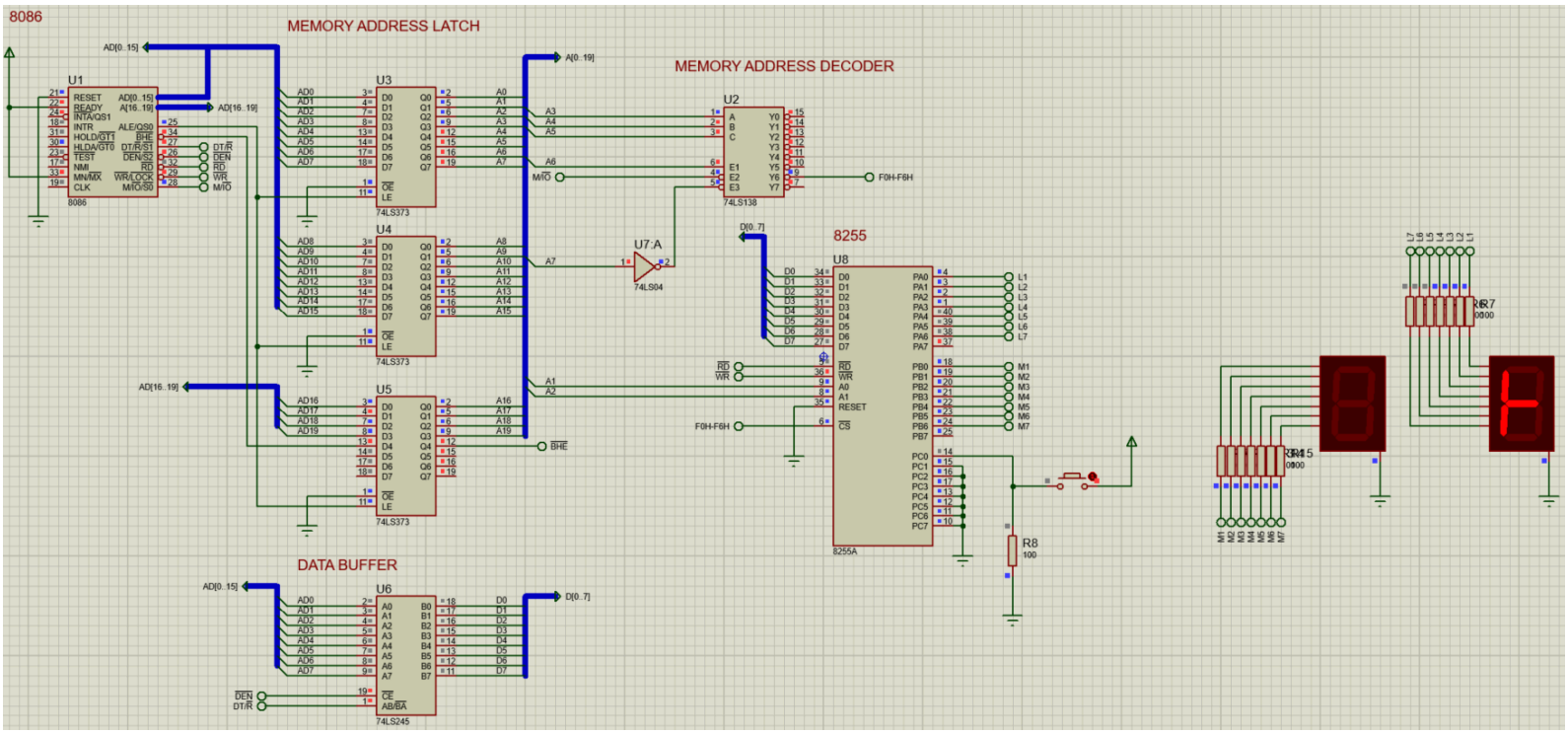
PORTC: F4H **COM_REG:** F6H

Command Byte for PORTA and PORTB as output and PORTC as input: 10001001B (89H)

CODE FOR PROGRAMMING 8255

```
MOV DX, COM_REG
MOV AL, 89H                ; can also be 10001001B
OUT DX, AL
```

SENDING 8-BIT DATA TO PORTA

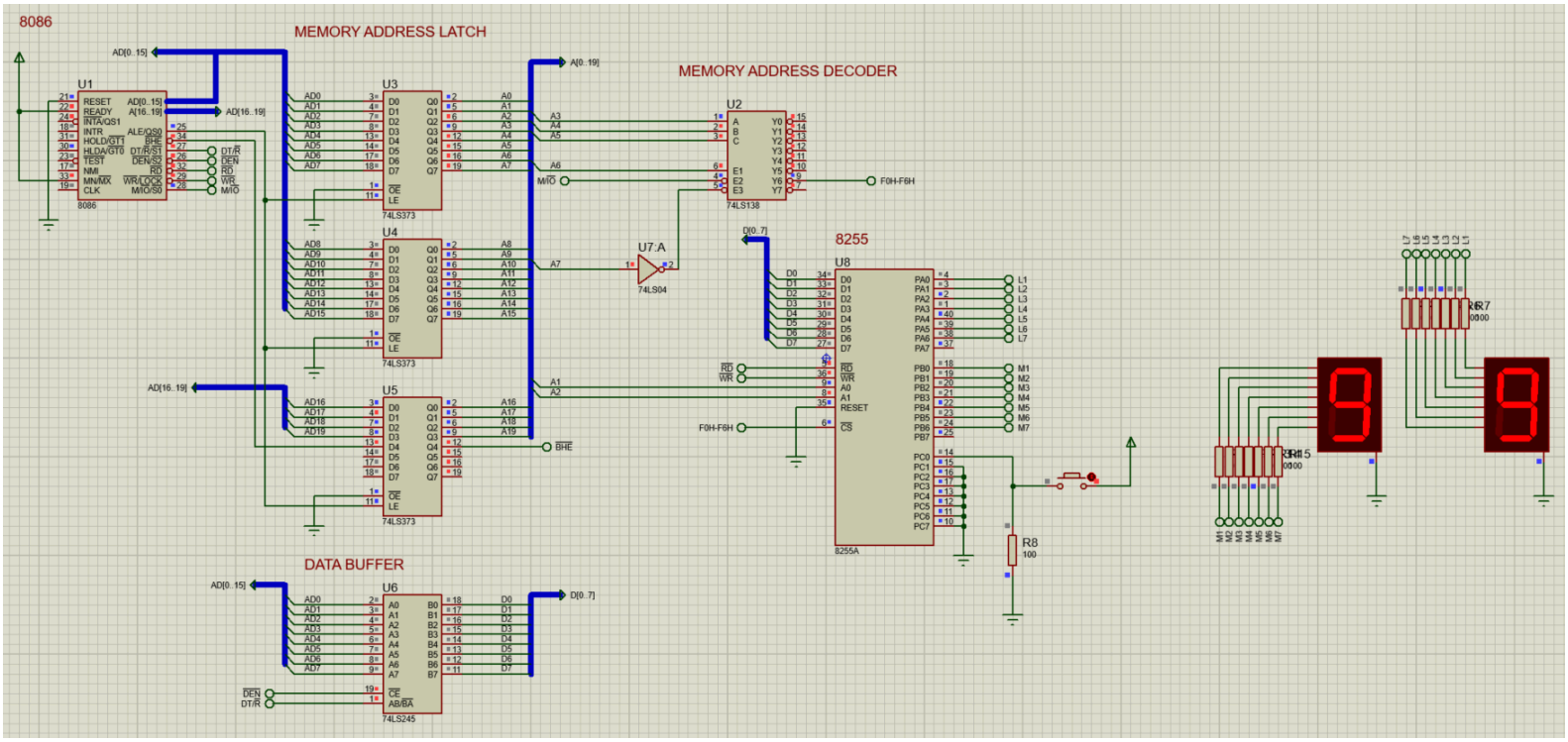


CODE FOR SENDING 8-BIT DATA TO PORTA

```
 DATA SEGMENT
    PORTA EQU 0F0H      ; PORTA address
    PORTB EQU 0F2H      ; PORTB address
    PORTC EQU 0F4H      ; PORTC address
    COM_REG EQU 0F6H    ; Command Register address
DATA ENDS

CODE SEGMENT
    MOV AX, DATA
    MOV DS, AX      ; set the Data Segment address
    ORG 0000H       ; write code below starting at address 0000H

START:
    MOV DX, COM_REG
    MOV AL, 89H
    OUT DX, AL
```

```
    MOV DX, PORTA       ; set port address of PORTA
    MOV AL, 11110000B
    OUT DX, AL          ; send 11110000B to PORTA

HERE:
    NOP                 ; do nothing
    JMP HERE
CODE ENDS
END
```

COUNT FROM 00 TO 99



CODE FOR COUNTING FROM 00 TO 99

main.asm

```
DATA SEGMENT

        PORTA EQU 0F0H      ; Address of PORTA
        PORTB EQU 0F2H      ; Address of PORTB
        PORTC EQU 0F4H      ; Address of PORTC
        COM_REG EQU 0F6H    ; Address of Command Register
        NUMB0 EQU 00111111B ; Binary code for displaying 0
        NUMB1 EQU 00000110B ; Binary code for displaying 1
        NUMB2 EQU 01011011B ; Binary code for displaying 2
        NUMB3 EQU 01001111B ; Binary code for displaying 3
        NUMB4 EQU 01100110B ; Binary code for displaying 4
        NUMB5 EQU 01101101B ; Binary code for displaying 5
        NUMB6 EQU 01111101B ; Binary code for displaying 6
        NUMB7 EQU 00000111B ; Binary code for displaying 7
        NUMB8 EQU 01111111B ; Binary code for displaying 8
        NUMB9 EQU 01101111B ; Binary code for displaying 9

DATA ENDS

CODE SEGMENT

        MOV AX, DATA      ; Load address of DATA segment into AX
        MOV DS, AX        ; Set Data Segment register (DS) to AX
        ORG 0000H         ; Start the code at memory address 0000H

START:
        MOV DX, COM_REG   ; Set Command Register address in DX
        MOV AL, 89H       ; Load control value into AL (initialization)
        OUT DX, AL        ; Send control value to Command Register

RESET:
        MOV DX, PORTA     ; Set address for PORTA
        MOV AL, NUMB0     ; Load value for displaying 0 into AL
        OUT DX, AL        ; Output value to PORTA

        MOV DX, PORTB     ; Set address for PORTB
        MOV AL, NUMB0     ; Load value for displaying 0 into AL
        OUT DX, AL        ; Output value to PORTB

        MOV CX, 0000H     ; Initialize CX register for counting

HERE:
        MOV DX, PORTC     ; Set address for PORTC
        IN AL, DX         ; Read input from PORTC into AL
        CMP AL, 01H       ; Compare AL with 1 (button press signal)
        JE LSDIG_A        ; Jump to LSDIG_A if equal
        JMP HERE          ; Otherwise, loop back to HERE

LSDIG_A:
        CALL DELAY        ; Call delay procedure
        CALL DELAY        ; Add a second delay for proper timing
        CMP CX, 0909H     ; Check if the count has reached 99
        JE RESET          ; Reset to 00 if CX = 99

        CMP CL, 09H       ; Check if lower byte of CX (CL) has reached 9
        JE MSDIG_B        ; Jump to MSDIG_B if CL = 9

        INC CL            ; Increment CL (units counter)
```

```
59
60          ; Update PORTA with the corresponding number based on CL
61          LSDIG_A1:
62          CMP CL, 01H         ; Check if CL = 1
63          JNE LSDIG_A2            ; Jump to LSDIG_A2 if not
64          MOV DX, PORTA       ; Set PORTA address
65          MOV AL, NUMB1       ; Load value for 1 into AL
66          OUT DX, AL          ; Output value to PORTA
67          JMP HERE
68
69          LSDIG_A2:
70          CMP CL, 02H         ; Check if CL = 2
71          JNE LSDIG_A3            ; Jump to LSDIG_A3 if not
72          MOV DX, PORTA
73          MOV AL, NUMB2
74          OUT DX, AL
75          JMP HERE
76
77          LSDIG_A3:
78          CMP CL, 03H
79          JNE LSDIG_A4
80          MOV DX, PORTA
81          MOV AL, NUMB3
82          OUT DX, AL
83          JMP HERE
84
85          LSDIG_A4:
86          CMP CL, 04H
87          JNE LSDIG_A5
88          MOV DX, PORTA
89          MOV AL, NUMB4
90          OUT DX, AL
91          JMP HERE
92
93          LSDIG_A5:
94          CMP CL, 05H
95          JNE LSDIG_A6
96          MOV DX, PORTA
97          MOV AL, NUMB5
98          OUT DX, AL
99          JMP HERE
100
101         LSDIG_A6:
102         CMP CL, 06H
103         JNE LSDIG_A7
104         MOV DX, PORTA
105         MOV AL, NUMB6
106         OUT DX, AL
107         JMP HERE
108
109         LSDIG_A7:
110         CMP CL, 07H
111         JNE LSDIG_A8
112         MOV DX, PORTA
113         MOV AL, NUMB7
114         OUT DX, AL
115         JMP HERE
116
```

```
117             LSDIG_A8:
118             CMP CL, 08H
119             JNE LSDIG_A9
120             MOV DX, PORTA
121             MOV AL, NUMB8
122             OUT DX, AL
123             JMP HERE
124
125             LSDIG_A9:
126             CMP CL, 09H         ; Check if CL = 9
127             MOV DX, PORTA
128             MOV AL, NUMB9
129             OUT DX, AL
130             JMP HERE
131
132     MSDIG_B:
133             MOV CL, 00H         ; Reset CL (units counter) to 0
134             MOV DX, PORTA
135             MOV AL, NUMB0       ; Display 0 on PORTA
136             OUT DX, AL
137
138             INC CH              ; Increment CH (tens counter)
139
140             ; Update PORTB with the corresponding number based on CH
141             HLSDIG_A1:
142             CMP CH, 01H
143             JNE HLSDIG_A2
144             MOV DX, PORTB
145             MOV AL, NUMB1
146             OUT DX, AL
147             JMP HERE
148
149             HLSDIG_A2:
150             CMP CH, 02H
151             JNE HLSDIG_A3
152             MOV DX, PORTB
153             MOV AL, NUMB2
154             OUT DX, AL
155             JMP HERE
156
157             HLSDIG_A3:
158             CMP CH, 03H
159             JNE HLSDIG_A4
160             MOV DX, PORTB
161             MOV AL, NUMB3
162             OUT DX, AL
163             JMP HERE
164
165             HLSDIG_A4:
166             CMP CH, 04H
167             JNE HLSDIG_A5
168             MOV DX, PORTB
169             MOV AL, NUMB4
170             OUT DX, AL
171             JMP HERE
```

```
172
173             HLSDIG_A5:
174             CMP CH, 05H
175             JNE HLSDIG_A6
176             MOV DX, PORTB
177             MOV AL, NUMB5
178             OUT DX, AL
179             JMP HERE
180
181             HLSDIG_A6:
182             CMP CH, 06H
183             JNE HLSDIG_A7
184             MOV DX, PORTB
185             MOV AL, NUMB6
186             OUT DX, AL
187             JMP HERE
188
189             HLSDIG_A7:
190             CMP CH, 07H
191             JNE HLSDIG_A8
192             MOV DX, PORTB
193             MOV AL, NUMB7
194             OUT DX, AL
195             JMP HERE
196
197             HLSDIG_A8:
198             CMP CH, 08H
199             JNE HLSDIG_A9
200             MOV DX, PORTB
201             MOV AL, NUMB8
202             OUT DX, AL
203             JMP HERE
204
205             HLSDIG_A9:
206             CMP CH, 09H         ; Check if CH = 9
207             MOV DX, PORTB
208             MOV AL, NUMB9
209             OUT DX, AL
210             JMP HERE
211
212 DELAY PROC                  ; Delay subroutine to slow down the counting
213               MOV BX, 1BE4H   ; BX controls the delay duration
214             L1:
215               DEC BX          ; Decrement BX until it reaches 0
216               NOP             ; No operation (waste one clock cycle)
217               JNZ L1          ; Jump to L1 if BX is not zero
218               RET             ; Return from subroutine
219 DELAY ENDP
220
221 CODE ENDS
222 END
```

References