Electrical and Computer Systems Engineering Department

Computer Design Laboratory

ENCS411

Report 3

Experiment 10: 8259 Programmable Interrupt Controller Application

Prepared by

Anas Nimer_1180180

Instructor:

Dr. Abdallatif Abuissa

Teacher assistant:

Eng. Motasem Diab

BIRZEIT

August – 2021

1. Abstract:

In this experiment, we will illustrate 8086 Interrupt capabilities using Intel 8259 PIC that includes reviewing Intel 8259 control that we learn in microprocessor course also how to use PPI 8255 & PIT 8253/4, we will discuss modes, read/write and the other pins.

Table Of Contents:

1. Abstract: II
2. Theory: 1
   2.1. 8259 PIC Microprocessor: 1
      2.1.1. Features of 8259 PIC microprocessor 1
      2.1.2. 8259 Pins: 2
         2.1.2.1. Data Bus Buffer (D7-D0): 2
         2.1.2.2. Read/Write: 2
         2.1.2.3. A0: 2
         2.1.2.4. Chip Select (CS): 2
         2.1.2.5. Vcc: 2
         2.1.2.6. GND: 2
         2.1.2.7. INT: 2
         2.1.2.8. INTA: 2
         2.1.2.9. Interrupt request register (IRR): 2
         2.1.2.10. Interrupt service register (ISR): 2
         2.1.2.11. Interrupt mask register (IMR): 2
         2.1.2.12. Priority resolver: 2
         2.1.2.13. Cascade buffer: 3
         2.1.2.14. Slave Program (SP) / Enable Buffer (EB): 3
3. Procedure & Discussion: 4
   3.1. 3x8 Decoder: 4
   3.2. 8086: 4
   3.3. Experiment Code [A1]: 5
   4. In Lab To Do (code in [A2]): 8
5. Conclusion: 10
6. References: 11
7. Appendix 12
   7.1. A.1: 12
   7.2. A.2: 16

Table of figure:

Figure 1:8259 Block Diagram 1
Figure 2:EXP 10 Counter Circuit 4
Figure 3: 7 bit segment Array 5
Figure 4:8259 Code 5
Figure 5:Program PPI Code 5
Figure 6:Program 8253 Code 6
Figure 7:Interrupt Routine 6
Figure 8:Interrupt Setup Code 6
Figure 9:Show 7 segment 7
Figure 10:In Lab To Do Circuit 8
Figure 11:Mask 8
Figure 12:Decrease Interrupt Routine 9
Figure 13:Connect With Interrupt 9

2. Theory:

2.1. 8259 PIC Microprocessor:

Programmable Interrupt Controller (PIC) microprocessor is the name given to the 8259 microprocessors. In 8085 and 8086, there are 5 hardware interrupts and 2 hardware interrupts, respectively. We will improve the interrupt handling capabilities of the 8259 by connecting it to the CPU. The 8259 converts several interrupt inputs into a single interrupt output the single PIC interface provides 8 interrupt inputs ranging from (IR0-IR7).

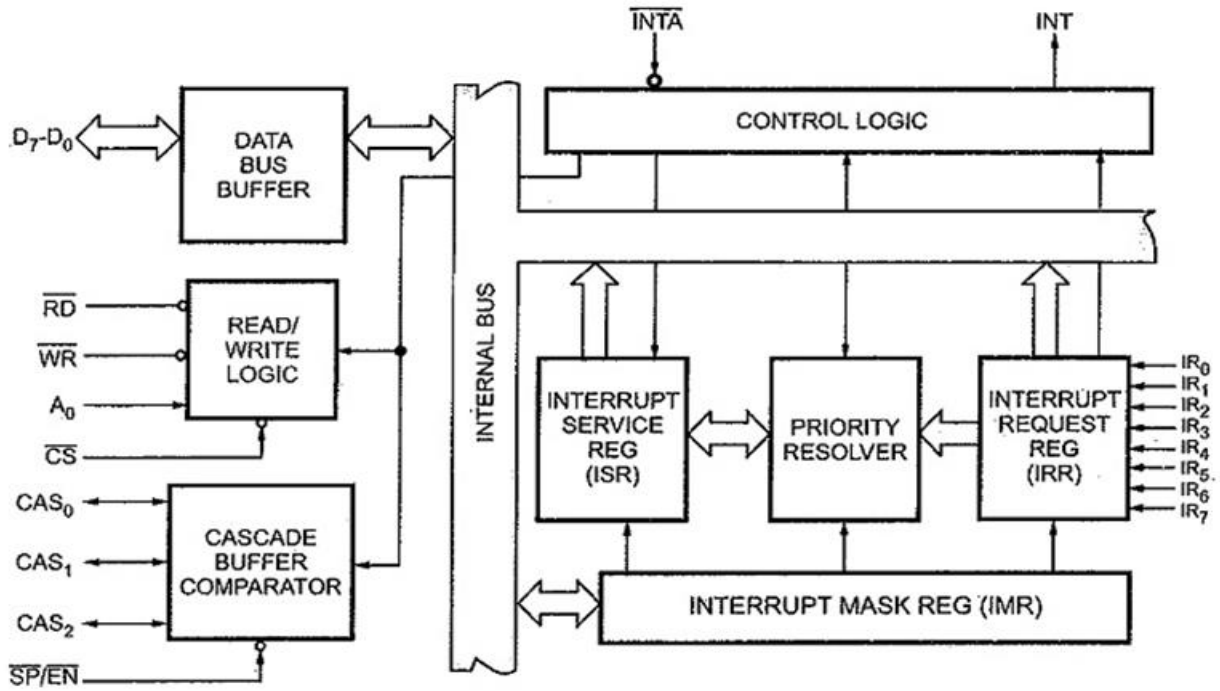

Figure 1:8259 Block Diagram

2.1.1. Features of 8259 PIC microprocessor – [1]:

- Intel 8259 is designed for Intel 8085 and Intel 8086 microprocessor.
- It can be programmed either in level triggered or in edge triggered interrupt level.
- We can masked individual bits of interrupt request register.
- We can increase interrupt handling capability up to 64 interrupt level by cascading further 8259 PIC.
- Clock cycle is not required.

2.1.2. 8259 Pins:

2.1.2.1. Data Bus Buffer (D7-D0):

By acting as a buffer, this block acts as a mediator between the 8259 and the 8085/8086 microprocessor.

2.1.2.2. Read/Write:

Active low inputs activate by the processor to read or write the input data from the 8259.

2.1.2.3. A0:

Input address used with RD & WR to identify the different command words.

2.1.2.4. Chip Select (CS):

Active low input used to select the chip it's used.

2.1.2.5. Vcc:

It's input voltage or the power supply which is equal to 5V in 8259 chip.

2.1.2.6. GND:

Ground pin V= 0.

2.1.2.7. INT:

Active high output which is interrupt the processor when and it's always connect to INTR of 8085.

2.1.2.8. INTA:

It is termed as an active low-input pin. The 8259 receives the signal from INTA* to the output of 8085. 8085 sends the three consecutive INTA* signals, the 8259sends a 3-byte CALL instruction to the 8085 via D7-0 pins. The two bytes termed as second and third bytes of the CALL instruction contains the ISS address which depends on the IR input of 8259 that is going to be serviced.[2]

2.1.2.9. Interrupt request register (IRR):

It stores all the interrupt level which are requesting for Interrupt services.[1]

2.1.2.10. Interrupt service register (ISR):

It stores the interrupt level which are currently being executed.[1]

2.1.2.11. Interrupt mask register (IMR):

It stores the interrupt level which have to be masked by storing the masking bits of the interrupt level.[1]

2.1.2.12. Priority resolver:

It examines all the three registers and set the priority of interrupts and according to the priority of the interrupts, interrupt with highest priority is set in ISR register. Also, it reset the interrupt level which is already been serviced in IRR.[1]

2.1.2.13. Cascade buffer:

To increase the Interrupt handling capability, we can further cascade more number of pins by using cascade buffer. So, during increment of interrupt capability, CSA lines are used to control multiple interrupt structure.[1]

2.1.2.14. Slave Program (SP) / Enable Buffer (EB):

SP*/EN* stands for “slave program/enable buffer”. This pin serves dual function. When it is used as EN* pin it provides an active low-output pin that controls the buffer transceivers in the buffer mode.[2]

3. Procedure & Discussion:

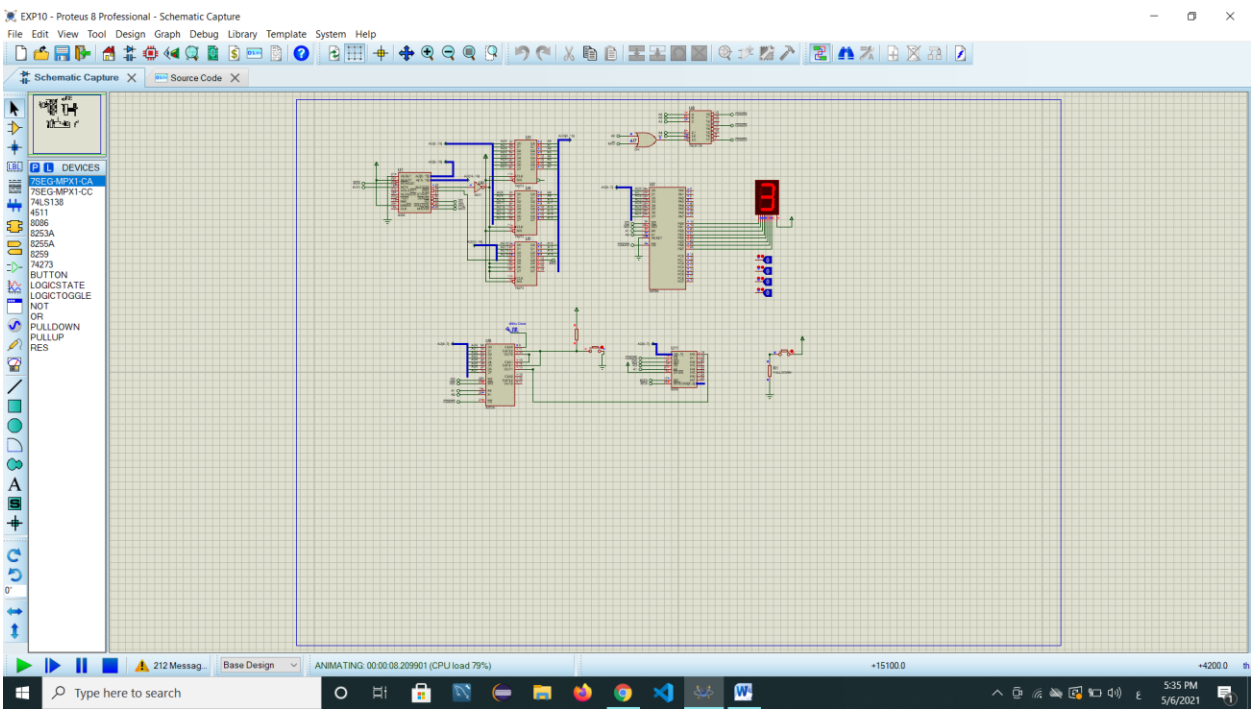

Figure 2:EXP 10 Counter Circuit

The counter circuit was built using Proteus program which is count from 0 to F using 7 segment which is connected to port B in the PPI, In the figure there's a lot of chips such as 3x8decoder, 8086, 8259, PPI, PIC controller that it will be discussed below:

3.1. 3x8 Decoder:

Used to select the chip depending on the address from the 8086, A3, A5 and A6 specify the chip when decoder is enabled where A4 = 1 active high, A7 & A0 & M/~(IO) = 0 active low, there's three possible outputs to the decoder: -

- ✓ PIC when A6, A5, A3 equal to 000 and it's start address is 10H
- ✓ PPI when A6, A5, A3 equal to 011 and it's start address is 70H
- ✓ PIT when A6, A5, A3 equal to 111 and it's start address is 78H
- ✓

3.2. 8086:

Address data buses of the 8086 connected with all chips in the circuit also ~BHE & ALE/QS0 used with the PPI, in the circuit the work done using low buses if we want to use the high buses the we replace low buses by high buses and replace A0 in the decoder with ~BHE.

3.3. Experiment Code [A1]:

```
DATA    SEGMENT PARA 'DATA'
; 0 for lighting, 1 otherwise
DIGITS  DB 0C0h, 0F9h, 0A4h, 0B0h, 99h, 92h, 82h, 0F8h, 80h, 90h, 088h, 083h, 0C6h, 0A1h, 86h, 8Eh  , 00 ; 0 to F
DATA    ENDS
```

Figure 3: 7 bit segment Array

Digits array express the numbers from 0 to F in 7 bit segment so to covert from number to the next one array index must increase and that's make the code easily and less complexity.

```
SETUP_8259 PROC NEAR

MOV AL, 00010011b       ; ICW1 : Edge Triggered - 1 Master - ICW4 Needed
OUT 10h, AL             ; SET InputControlWord1

MOV AL, 40h             ; ICW2 : 40h
OUT 12h, AL             ; SET InputControlWord2

MOV AL, 03h             ; AEOI(Automatic Interrupt) = 1, 8086= 1
OUT 12h, AL             ; SET InputControlWord4

RET
SETUP_8259 ENDP
;-------------------------------
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Figure 4:8259 Code

00010011 moved to 10H which is the starting address of ICW1 then we send 40H as interrupt 0 to the end address (12H), after that in ICW4 Automatic end interrupt & 8086 processor ones and the others zeros so its command word = 0000 0011 = 3H moved to 12H end address.

```
;; Address of 8253: 78h
;; Addrss of  8255: 70h
;; Address of 8259: 10h
MOV AL, 80h             ; 1000 0000b
OUT 76h, AL             ; Setup 8255 as all ports mo0 output
CALL SETUP_8253
```

Figure 5:Program PPI Code

Enable it to sample input output and suppose all others zeros (outputs) then, move it to the control register output 76H and call setup_8253.

```
SETUP_8253 PROC NEAR
    ;; SETUP 8253

    ; --> Used to Divide 4Mhz Clock to 2000 so that we get 2Khz Output
    MOV AL, 00110111b   ; 00 [counter 0], 11[ 2 byte data], 010 [ with mod 3], 1 [BCD]
    OUT 7Eh, AL         ; configure the counter 0 of 8253

    ; --> Pulse Wave  :: MOD 2 of 8253
    MOV AL, 01110111b   ; 01 [counter 1], 11[ 2 byte data], 010 [ with mod 3], 1 [BCD]
    OUT 7Eh, AL         ; configure the counter 1 of 8253

    MOV AL, 00h
    OUT 78h, AL         ; Send first byte of Counter 0
    MOV AL, 20h
    OUT 78h, AL         ; Send second byte of Counter 0
    ;;--> Counter0 : 2000h      ; send BCD 2000 to counter 0

    MOV AL, 00h
    OUT 7Ah, AL         ; Send first byte of Counter 1
    MOV AL, 5h
    OUT 7Ah, AL         ; Send second byte of Counter 1
    ;;--> Counter1   : 5h ; send BCD 5 to counter 1
RET
SETUP_8253 ENDP
```

Figure 6:Program 8253 Code

The code in figure 6 enable 2 counters cascade firstly program counter 0 and output it on 7EH then the same thing but with counter 1 on 7EH, then program timer by divide over 2000h in the counter 0 and over 500h in counter 1.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Interrupt Routine
INC_AX PROC FAR
  INC AX
IRET
INC_AX ENDP
................................
```

Figure 7:Interrupt Routine

In figure 7 AX increased by 1 where AX express 7 segment Array index when it's increase that's means go to the next digit in the array.

```
INSERT_ISR_INTO_VECTOR_TABLE PROC NEAR
  XOR AX, AX
  MOV ES, AX
  MOV AL, 40H
  MOV AH, 4
  MUL AH
  MOV BX, AX
  LEA AX, INC_AX
  MOV WORD PTR ES:[BX], AX
  MOV AX, CS
  MOV WORD PTR ES:[BX+2], AX
RET
INSERT_ISR_INTO_VECTOR_TABLE ENDP
```

Figure 8:Interrupt Setup Code

We multiply 40 H, which is the interrupt 0 address, by 4 to get the offset of the interrupt routine loop to the end of multiplying 4 by 40, and we also transfer the code section.

```
SHOW_AX PROC NEAR
  PUSH AX

  XOR BX, BX
  MOV BL, AL
  MOV AL, DIGITS[BX]
  OUT 72h, AL                    ; 8255's Port B at 72h

  POP AX
RET
SHOW_AX ENDP
```

Figure 9:Show 7 segment

The code in Figure 9 shows the digits of the 7 segment bits when start simulation on the screen.

4. In Lab To Do (code in [A2]):

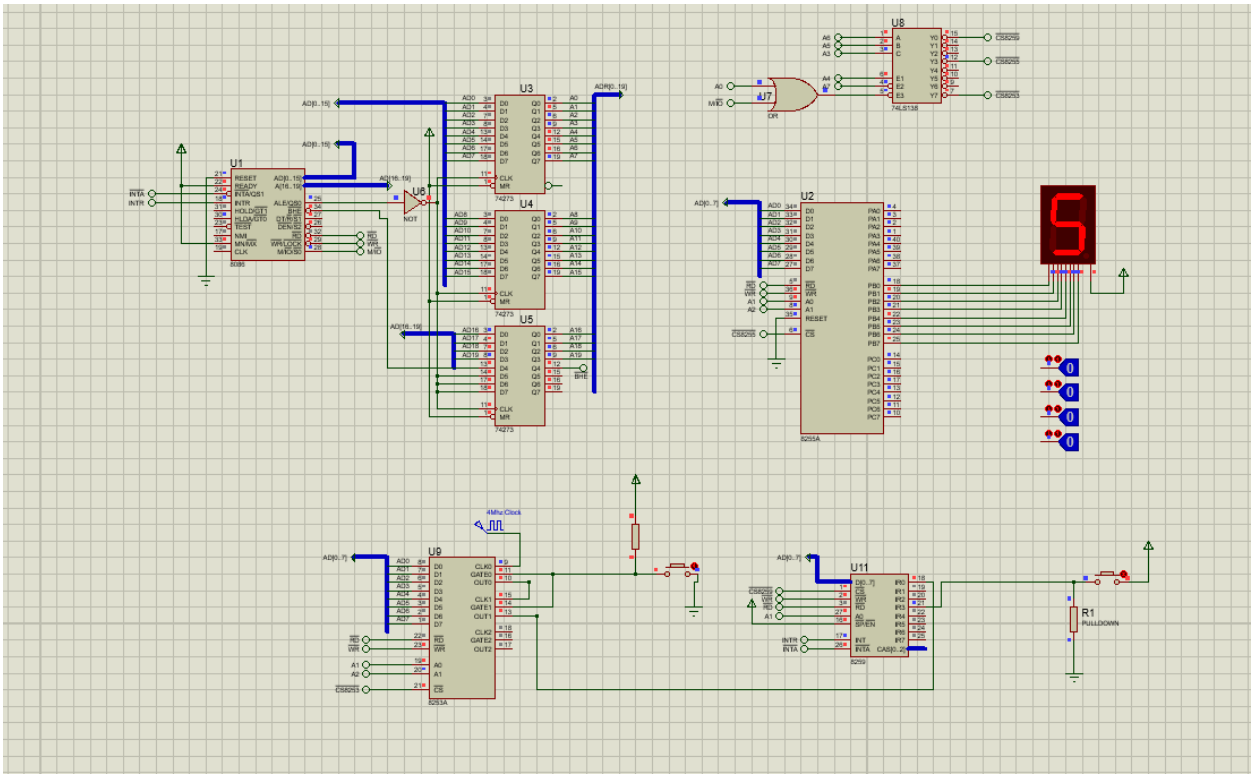

Figure 10:In Lab To Do Circuit

The aim of this to do to make counter count in reverse order when we click on the switch connected to IR3 in 8259 chip.

```
MOV AL,1110110B ; mask
OUT 12H,AL
```

Figure 11:Mask

In figure 11 everything mask except IR0.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Interrupt Routine
DEC_AX PROC FAR
   DEC AX
CMP AX, 0FFFFh
      JNZ BACK2
      MOV AX, 0Fh

BACK2:
IRET
DEC_AX ENDP
```

Figure 12:Decrease Interrupt Routine

In figure 12 when AX reach zero and when AX PROC FAR called it will decreased and its value become -1 which is equal to FFFFH then FH moved to register AX which is express the index of the 8 bit segment array.

```
XOR AX, AX
  MOV ES, AX
  MOV AL, 43H
  MOV AH, 4
  MUL AH
  MOV BX, AX
  LEA AX, DEC_AX
  MOV WORD PTR ES:[BX], AX
  MOV AX, CS
  MOV WORD PTR ES:[BX+2], AX
```

Figure 13:Connect With Interrupt

In this code we create another interrupt in 43H address this interrupt used to decrease the numbers to counter 7 segment reversely.

5. Conclusion:

In this experiment we learn about 8259A and how to use it with other chips like 8086, PPI and other chips, we learn how to build counter count in order and reverse also how to program each chip and work with interrupts.

6. References:

[1] https://www.geeksforgeeks.org/8259-pic-microprocessor/ [Accessed 4:34 Aug 2021].

[2] https://www.tutorialspoint.com/pins-of-8259 [Accessed 5:56 Aug 2021].

7. Appendix

7.1. A.1:

```
CODE    SEGMENT PARA 'CODE'
        ASSUME CS:CODE, DS:DATA, SS:STAK

STAK    SEGMENT PARA STACK 'STACK'
        DW 20 DUP(?)
STAK    ENDS

DATA    SEGMENT PARA 'DATA'
; 0 for lighting, 1 otherwise
DIGITS  DB 0C0h, 0F9h, 0A4h, 0B0h, 99h, 92h, 82h, 0F8h, 80h, 90h, 088h, 083h, 0C6h, 0A1h,
86h, 8Eh          , 00     ; 0 to F
DATA    ENDS
START PROC
   MOV AX, DATA
   MOV DS, AX

CALL INSERT_ISR_INTO_VECTOR_TABLE
      CALL SETUP_8259
      ;; Address of 8253: 78h
      ;; Addrss of  8255: 70h
      ;; Address of 8259: 10h
   MOV AL, 80h                    ; 1000 0000b
      OUT 76h, AL                 ; Setup 8255 as all ports mo0 output
      CALL SETUP_8253
```

```
    STI
    XOR AX,AX
ENDLESS:
    CALL SHOW_AX

    CMP AX, 10h
    JNZ ENDLESS
    MOV AX, 0h              ; MAke AX 0 after reaching 16

    JMP ENDLESS
RET
START ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;          Common Anode Seven Segmeent     ; 0 for lighting, 1 otherwise
SHOW_AX PROC NEAR
  PUSH AX

  XOR BX, BX
  MOV BL, AL
  MOV AL, DIGITS[BX]
  OUT 72h, AL                         ; 8255's Port B at 72h
  POP AX
RET
SHOW_AX ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SETUP_8253 PROC NEAR
```

```
        ;; SETUP 8253

        ; --> Used to Divide 4Mhz Clock to 2000 so that we get 2Khz Output
      MOV AL, 00110111b    ; 00 [counter 0], 11[ 2 byte data], 010 [ with mod 3], 1 [BCD]
        OUT 7Eh, AL          ; configure the counter 0 of 8253

        ; --> Pulse Wave  :: MOD 2 of 8253
        MOV AL, 01110111b; 01 [counter 1], 11[ 2 byte data], 010 [ with mod 3], 1 [BCD]
        OUT 7Eh, AL          ; configure the counter 1 of 8253

        MOV AL, 00h
        OUT 78h, AL          ; Send first byte of Counter 0
        MOV AL, 20h
        OUT 78h, AL          ; Send second byte of Counter 0
        ;;--> Counter0 : 2000h          ; send BCD 2000 to counter 0

        MOV AL, 00h
        OUT 7Ah, AL          ; Send first byte of Counter 1
        MOV AL, 5h
        OUT 7Ah, AL          ; Send second byte of Counter 1
        ;;--> Counter1 : 5h ; send BCD 5 to counter 1
RET
SETUP_8253 ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SETUP_8259 PROC NEAR
```

```
MOV AL, 00010011b       ; ICW1 : Edge Triggered - 1 Master - ICW4 Needed
OUT 10h, AL             ; SET InputControlWord1

MOV AL, 40h             ; ICW2 : 40h
OUT 12h, AL             ; SET InputControlWord2

MOV AL, 03h             ; AEOI(Automatic Interrupt) = 1, 8086= 1
OUT 12h, AL             ; SET InputControlWord4

RET
SETUP_8259 ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INSERT_ISR_INTO_VECTOR_TABLE PROC NEAR
  XOR AX, AX
  MOV ES, AX
  MOV AL, 40H
  MOV AH, 4
  MUL AH
  MOV BX, AX
 LEA AX, INC_AX
  MOV WORD PTR ES:[BX], AX
  MOV AX, CS
  MOV WORD PTR ES:[BX+2], AX
RET
INSERT_ISR_INTO_VECTOR_TABLE ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Interrupt Routine
INC_AX PROC FAR
  INC AX
IRET
INC_AX ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CODE    ENDS
        END START
```

7.2. A.2:

```
CODE    SEGMENT PARA 'CODE'
        ASSUME CS:CODE, DS:DATA, SS:STAK

STAK    SEGMENT PARA STACK 'STACK'
        DW 20 DUP(?)
STAK    ENDS

DATA    SEGMENT PARA 'DATA'
; 0 for lighting, 1 otherwise
DIGITS  DB 0C0h, 0F9h, 0A4h, 0B0h, 99h, 92h, 82h, 0F8h, 80h, 90h, 088h, 083h, 0C6h, 0A1h,
86h, 8Eh          , 00     ; 0 to F
DATA    ENDS

START PROC
  MOV AX, DATA
  MOV DS, AX
```

```
CALL INSERT_ISR_INTO_VECTOR_TABLE
    CALL SETUP_8259
    ;; Address of 8253: 78h
    ;; Addrss of  8255: 70h
    ;; Address of 8259: 10h
    MOV AL, 80h                     ; 1000 0000b
    OUT 76h, AL                     ; Setup 8255 as all ports mo0 output
    CALL SETUP_8253
    STI
    XOR AX,AX
ENDLESS:
    CALL SHOW_AX

    ;CMP AX, 10h
    ;JNZ ENDLESS
    ;MOV AX, 0h             ; MAke AX 0 after reaching 16

    JMP ENDLESS
RET
START ENDP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;            Common Anode Seven Segmeent     ; 0 for lighting, 1 otherwise
SHOW_AX PROC NEAR
  PUSH AX

  XOR BX, BX
  MOV BL, AL
  MOV AL, DIGITS[BX]
```

```
  OUT 72h, AL                          ; 8255's Port B at 72h

  POP AX
RET
SHOW_AX ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SETUP_8253 PROC NEAR
        ;; SETUP 8253

        ; --> Used to Divide 4Mhz Clock to 2000 so that we get 2Khz Output
      MOV AL, 00110111b    ; 00 [counter 0], 11[ 2 byte data], 010 [ with mod 3], 1 [BCD]
        OUT 7Eh, AL          ; configure the counter 0 of 8253

        ; --> Pulse Wave  :: MOD 2 of 8253
        MOV AL, 01110111b; 01 [counter 1], 11[ 2 byte data], 010 [ with mod 3], 1 [BCD]
        OUT 7Eh, AL          ; configure the counter 1 of 8253

        MOV AL, 00h
        OUT 78h, AL          ; Send first byte of Counter 0
        MOV AL, 10h
        OUT 78h, AL          ; Send second byte of Counter 0
        ;;--> Counter0 : 2000h          ; send BCD 2000 to counter 0

        MOV AL, 00h
        OUT 7Ah, AL          ; Send first byte of Counter 1
        MOV AL, 10h
        OUT 7Ah, AL          ; Send second byte of Counter 1
```

```
        ;;--> Counter1 : 5h ; send BCD 5 to counter 1
RET
SETUP_8253 ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SETUP_8259 PROC NEAR

MOV AL, 00010011b       ; ICW1 : Edge Triggered - 1 Master - ICW4 Needed
OUT 10h, AL             ; SET InputControlWord1

MOV AL, 40h             ; ICW2 : 40h
OUT 12h, AL             ; SET InputControlWord2

MOV AL, 03h             ; AEOI(Automatic Interrupt) = 1, 8086= 1
OUT 12h, AL             ; SET InputControlWord4

MOV AL,1110110B; mask
OUT 12H,AL

RET
SETUP_8259 ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INSERT_ISR_INTO_VECTOR_TABLE PROC NEAR
  XOR AX, AX
  MOV ES, AX
  MOV AL, 40H
  MOV AH, 4
```

```
  MUL AH
  MOV BX, AX
LEA AX, INC_AX
  MOV WORD PTR ES:[BX], AX
  MOV AX, CS
  MOV WORD PTR ES:[BX+2], AX


XOR AX, AX
  MOV ES, AX
  MOV AL, 43H
  MOV AH, 4
  MUL AH
  MOV BX, AX
  LEA AX, DEC_AX
  MOV WORD PTR ES:[BX], AX
  MOV AX, CS
  MOV WORD PTR ES:[BX+2], AX

RET
INSERT_ISR_INTO_VECTOR_TABLE ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Interrupt Routine
INC_AX PROC FAR
  INC AX
CMP AX, 10h
    JNZ BACK1
    MOV AX, 0h
```

```
BACK1:
IRET
INC_AX ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Interrupt Routine
DEC_AX PROC FAR
  DEC AX
CMP AX, 0FFFFh
    JNZ BACK2
    MOV AX, 0Fh

BACK2:
IRET
DEC_AX ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


CODE    ENDS
      END START
```