Interrupts

MDA-8086 Kit

Programmable Interrupt Controller Application

Birzeit University

Information Technology Faculty

Computer Systems Engineering Department

Abstract

This experiment aims at understanding and expanding 8086 Interrupt capabilities using Intel 8259 PIC that includes reviewing Intel 8259 control, its initialization and operational modes.

PART I Theoretical Introduction

The 8086 interrupts can be classified into three types. These are

1. Predefined interrupts
2. User-defined software interrupts
3. User-defined hardware interrupts

The interrupt vector address f or all the 8086 interrupts are determined from a table stored in locations 00000H through 003FFH. The starting addresses for the service routines for the interrupts are obtained by the 8086 using this table. Four bytes of the table are assigned to each interrupt: two bytes for IP and two bytes for CS. The table may contain up to 256 8-bit vectors. If fewer than 256 interrupts are defined in the system, the user need only provide enough memory for the interrupt pointer table for obtaining the defined interrupts. The interrupt address vector (contents of IP and CS) for all the interrupts of the 8086 assigns every interrupt a type code for identifying the interrupt. There are 256 type codes associated with 256 table entries. Each entry consists of two addresses, one for storing the IP contents and the other for storing the CS contents. Each 8086 interrupt physical address vector is 20 bits wide and is computed from the 16-bit contents of IP and CS. For obtaining an interrupt address vector, the 8086 calculates two addresses in the pointer table where IP and CS are stored for a particular interrupt type. For example, for the interrupt type nn (instruction INT nn), the table address for IP=4×nn and the table address for CS=4×nn+2. For servicing the 8086's nonmaskable interrupt (NMI pin), the 8086 assigns the type code 2 to this interrupt. The 8086 automatically executes the INT2 instruction internally to obtain the interrupt address vector as follows:

Address for IP = 4 × 2 = 00008H

Address for CS = 4 × 2 + 2 = 0000AH

The 8086 loads the values of IP and CS from the 20-bit physical address 00008H and 0000AH in the pointer table. The user must store the desired 16-bit values o f IP and CS in these locations. Similarly, the I P and CS values for other interrupts are calculated. The 8086 interrupt pointer table layout is shown in Figure 1.

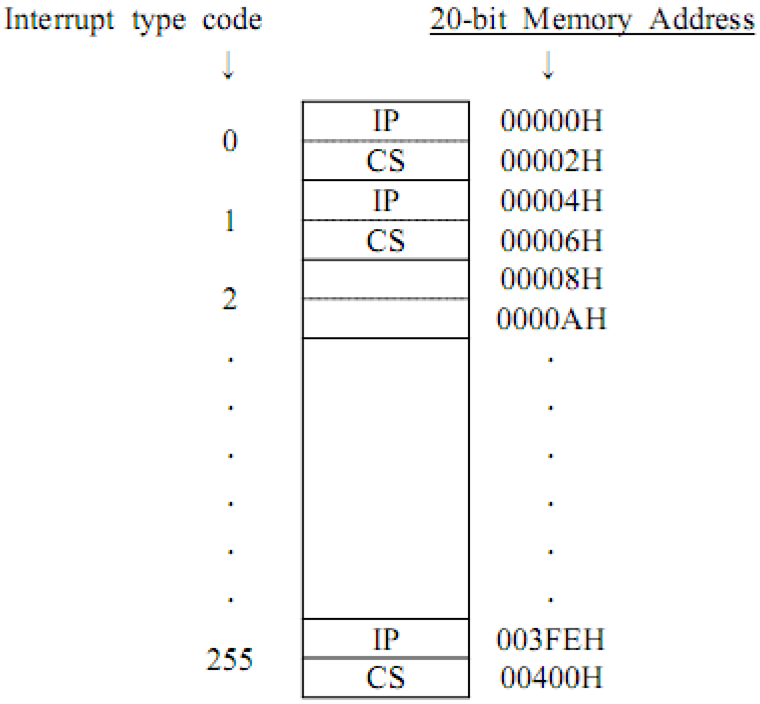


Figure 1 Interrupt Vector Table

In response to an interrupt, the 8086 pushes flags, CS, and IP onto the stack, clears TF and IF flags, and then loads IP and CS from the pointer table using the type code. Interrupt service routine should be terminated with the IRET (Interrupt Return) instruction which pops the top three s tack words into IP, CS, and flags, thus returning to the right place in the main program. The 256 interrupt type codes are assigned as follows;

- Types 0 to 4 are for the predefined interrupts.
- Types 5 to 31 are reserved by Intel for future use.
- Types 32 t o 255 are available for maskable interrupts.

Our focus in this lab is on **software interrupts**.

PART II Pre-Lab

(This part should be handed on to the teaching assistant in your Lab)

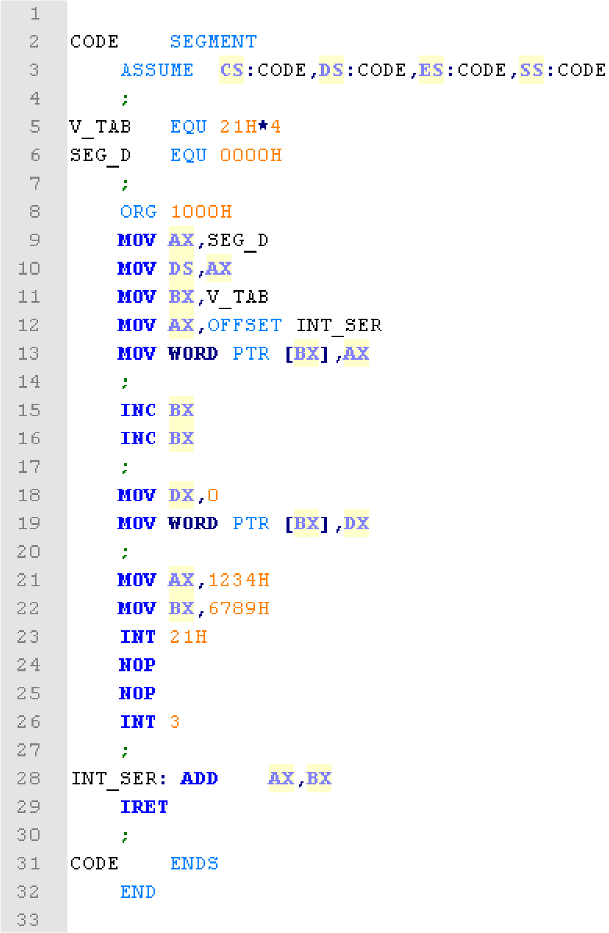
1. Review Intel 8259 Programmable Interrupt Controller and its modes of operation. Make sure you read the datasheet.
2. What would be the I/O ports for the 8259 if direct addressing mode is used with only 8086 A4 being "1" and 8086 A1 being connected to A0 of 8259?
3. Study the TO-DO Practices and write down the values for ICW1, ICW2, and ICW4?
4. What values of OCWs are needed?

PART III Practices

3.1 PRACTIC I: User Defined Software Interrupts

The user can generate an interrupt by executing a two-byte interrupt instruction "**INT nn**"[1]. The "**INT nn**" instruction is not maskable by the interrupt enable flag (IF). The "**INT nn**" instruction can be used to test an interrupt service routine for external interrupts. Type codes 0 to 255 can be used. If predefined interrupt is not used in a system, the associated type code can be utilized with the "**INT nn**" instruction to generate software (internal) interrupts.

Setp1: Write down the following code and save it to an Assembly file.



```

CODE    SEGMENT
    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
    ;
V_TAB   EQU 21H*4
SEG_D   EQU 0000H
    ;
    ORG 1000H
    MOV AX,SEG_D
    MOV DS,AX
    MOV BX,V_TAB
    MOV AX,OFFSET INT_SER
    MOV WORD PTR [BX],AX
    ;
    INC BX
    INC BX
    ;
    MOV DX,0
    MOV WORD PTR [BX],DX
    ;
    MOV AX,1234H
    MOV BX,6789H
    INT 21H
    NOP
    NOP
    INT 3
    ;
INT_SER: ADD    AX,BX
    IRET
    ;
CODE    ENDS
    END

```

Figure 2 Code 1

Step2: Compile and build this ASM file and execute it on MDA-8086 kit. (How? Review Exp#1 Intro. To MDA Kit)

[1] nn: Interrupt number

TASKS:

1. Explain what does this code do?
2. What does INT 21 do? (after executing the code)
3. What is the content of AX after executing the code/

3.2 PRACTICE II: 8259a Interrupt Control (Polling Technique)

The Intel 8259A Programmable interrupt controller handles up to eight sectored priority interrupts for the CPU. It is cascade-able for up to 64 vectored probabilities interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single + 5V supply. Circuitry is static, requiring no clock input. The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements. Refer to 8259A data sheet for more detail. The 8259A and MDA-8086 interface is shown in Figure 3.

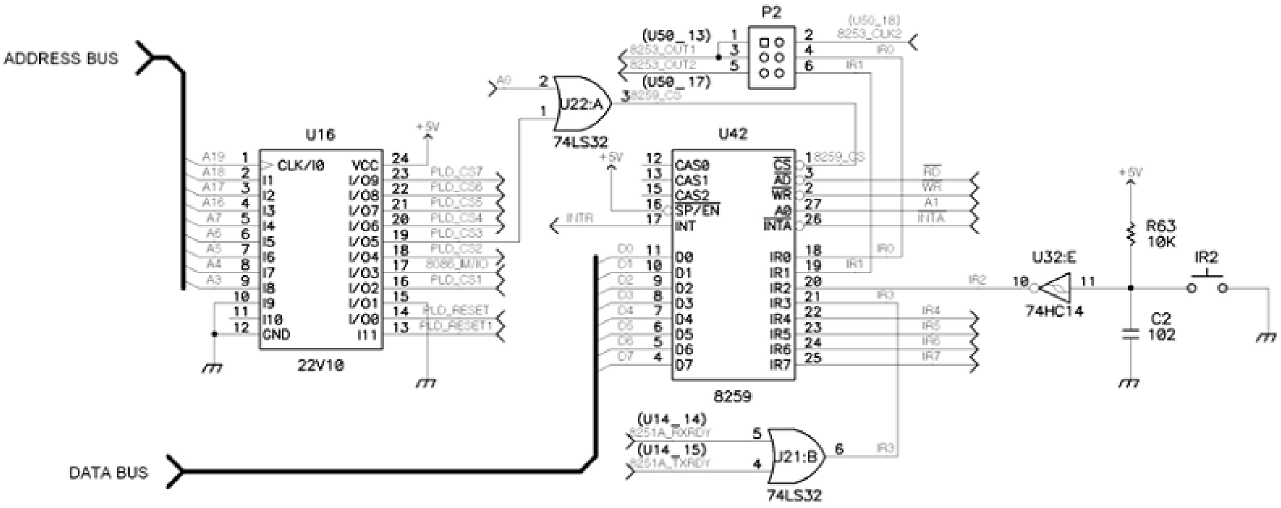


Figure 3 8259A and MDA-8086 interface

The IR2 interrupt request input of the 8259A is connected to press-button switch such that whenever it is pressed an interrupt request is generated to the CPU (active low input).

Figure 4 shows the PPI connectivity with the LED farm.

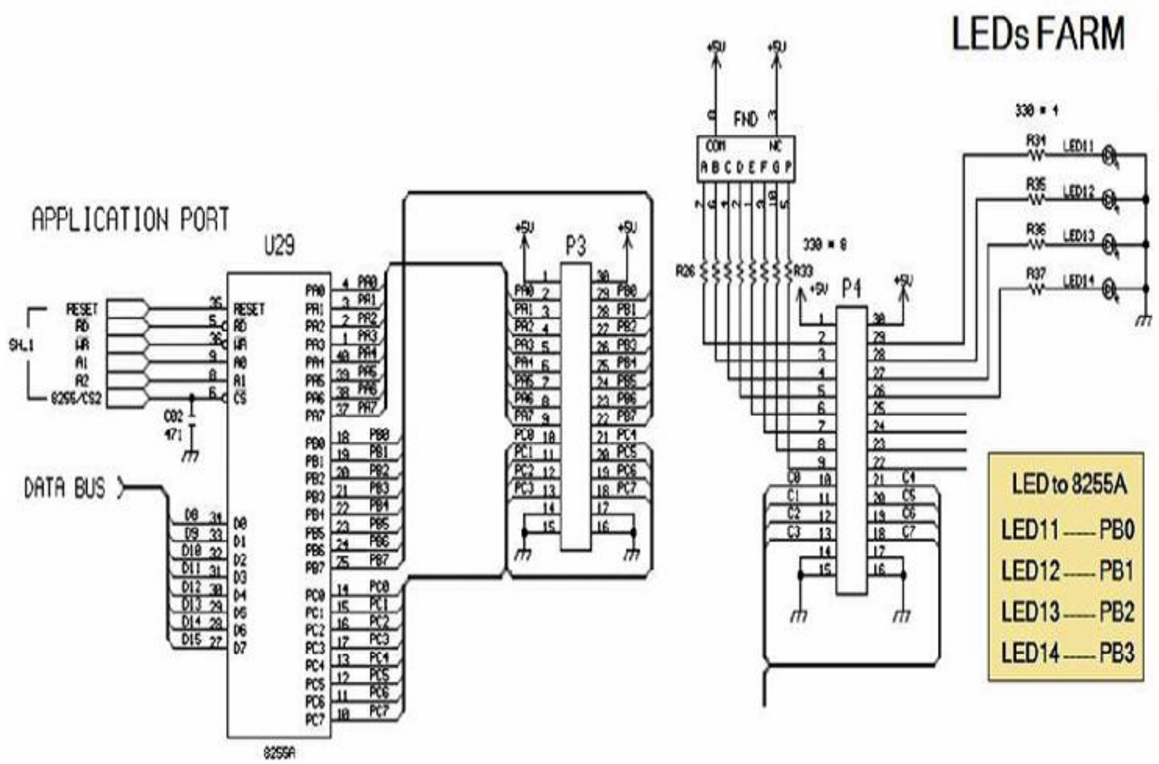


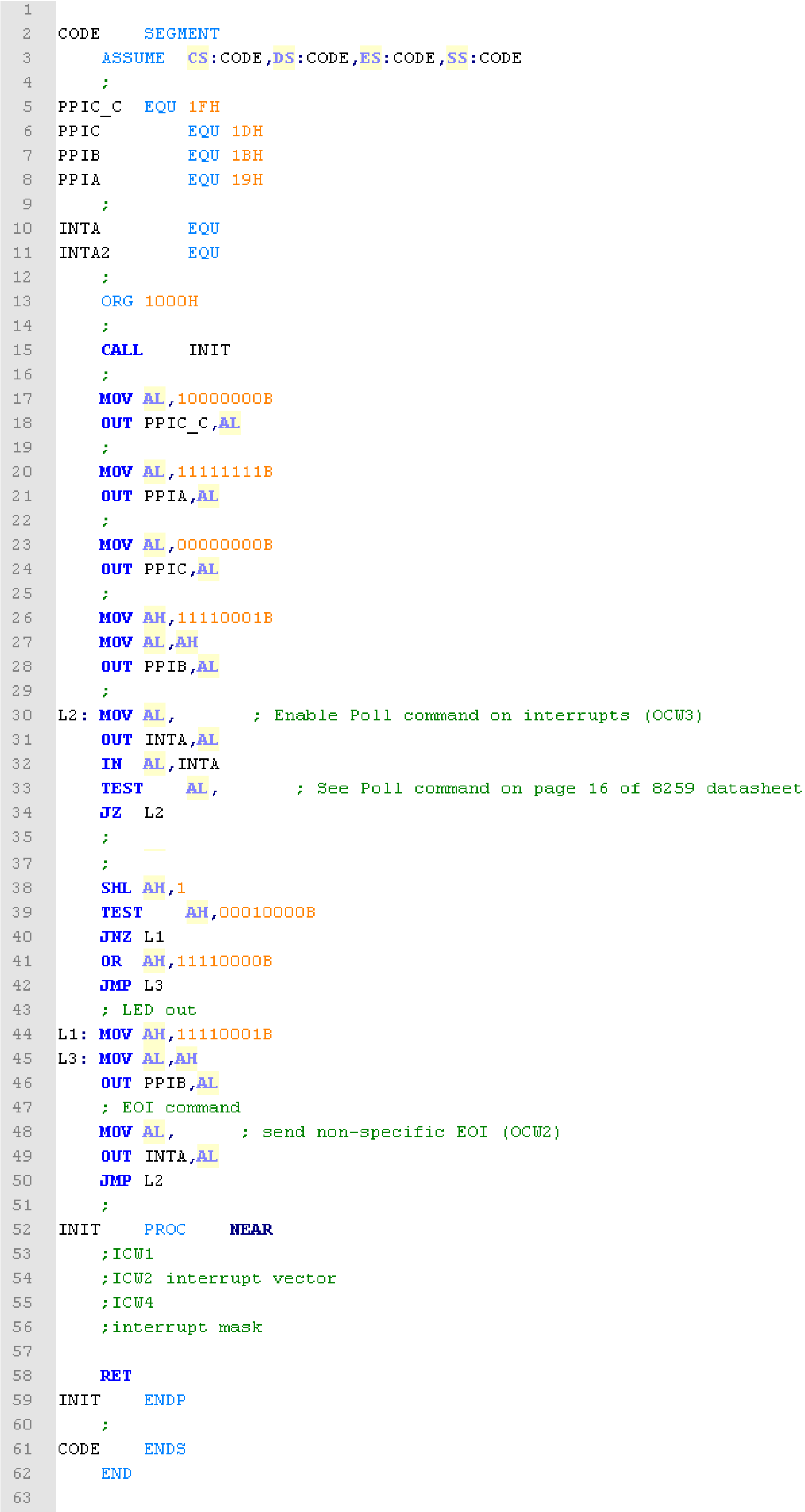
Figure 4 PPI Connectivity with the LED Farm

Practice Goal: Write an Assembly program that will control the LEDs such that only one LED is lit every time you send an interrupt to the CPU by requesting a service via the IR2 input to the 8259 controller? The LEDs circulate one at a time in response to an interrupt.

Needed Info: The 8259 is initialized with the following features:

1. ICW4 is needed
2. Edge triggered mode
3. An address interval of 8
4. Single mode
5. Interrupt vector of 40H
6. Normal end of interrupt
7. Non-buffered mode
8. Not SFNM

Setp1: Write down the following code and save it to an Assembly file. (Code is not complete. Complete it as required)



```
CODE    SEGMENT
    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
    ;
PPIC_C  EQU 1FH
PPIC        EQU 1DH
PPIB        EQU 1BH
PPIA        EQU 19H
    ;
INTA        EQU
INTA2       EQU
    ;
    ORG 1000H
    ;
    CALL    INIT
    ;
    MOV AL,10000000B
    OUT PPIC_C,AL
    ;
    MOV AL,11111111B
    OUT PPIA,AL
    ;
    MOV AL,00000000B
    OUT PPIC,AL
    ;
    MOV AH,11110001B
    MOV AL,AH
    OUT PPIB,AL
    ;
L2: MOV AL,       ; Enable Poll command on interrupts (OCW3)
    OUT INTA,AL
    IN  AL,INTA
    TEST    AL,       ; See Poll command on page 16 of 8259 datasheet
    JZ  L2
    ;
    ;
    SHL AH,1
    TEST    AH,00010000B
    JNZ L1
    OR  AH,11110000B
    JMP L3
    ; LED out
L1: MOV AH,11110001B
L3: MOV AL,AH
    OUT PPIB,AL
    ; EOI command
    MOV AL,      ; send non-specific EOI (OCW2)
    OUT INTA,AL
    JMP L2
    ;
INIT    PROC    NEAR
    ;ICW1
    ;ICW2 interrupt vector
    ;ICW4
    ;interrupt mask

    RET
INIT    ENDP
    ;
CODE    ENDS
    END
```

Figure 5 Code 2

Step2: Compile and build this ASM file and execute it on MDA-8086 kit. (How? Review Exp#1 Intro. To MDA Kit)

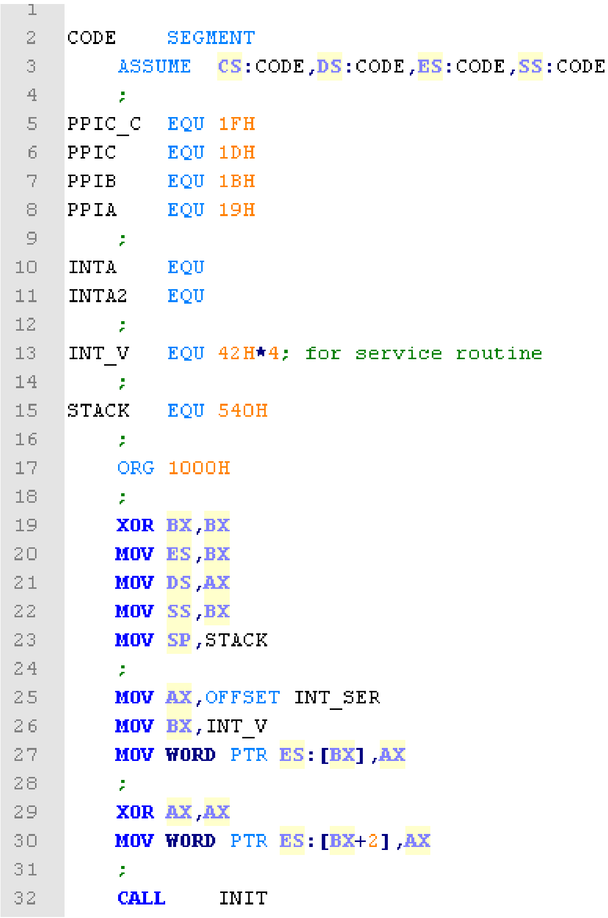
TASKS:

1. Explain what does this code do?
2. What do we mean by Polling? Why polling is used?
3. What does EOI assembly instruction do?

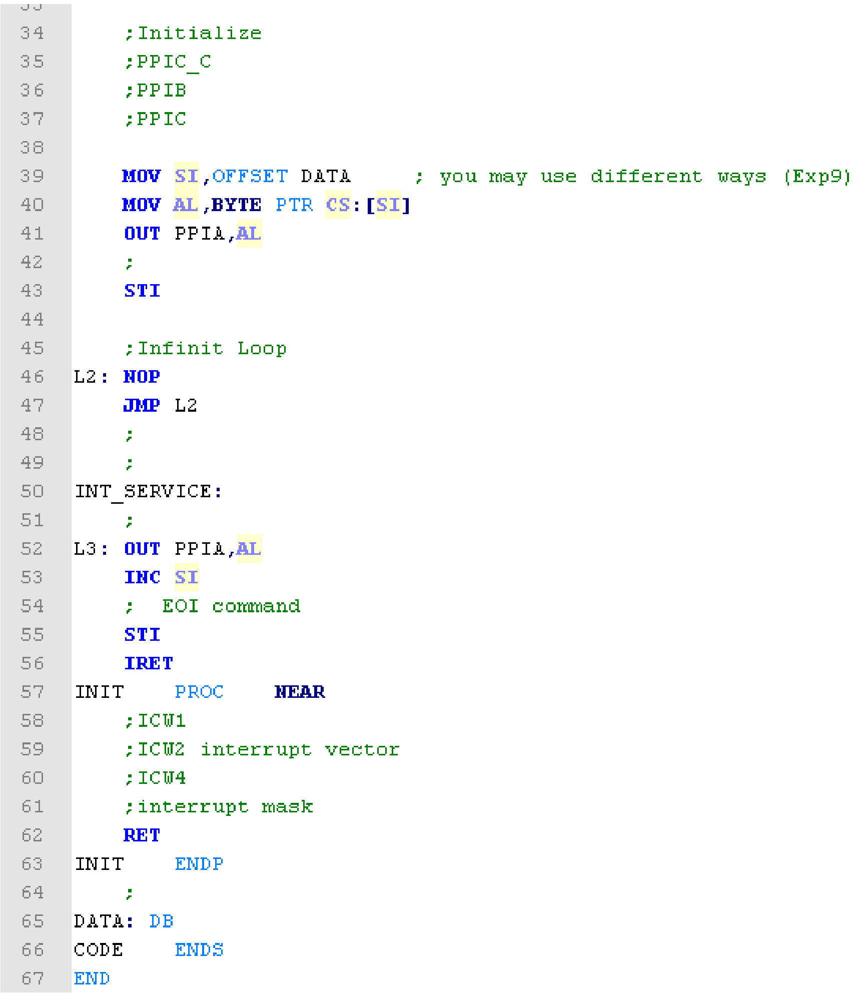
3.3 PRACTICE III: 8259a Interrupt Control (Interruption Technique)

Practice Goal: The task now is to control the seven-segment display to count from 0 to 9 and back to 0. The display can advance from one digit to the next only when IR2 switch is pressed. You may reference back to your work from experiment 9.

Setp1: Write down the following code and save it to an Assembly file to achieve the required goal. (P.S.: Code is not complete. Complete it as required)



```

CODE    SEGMENT
    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
    ;
PPIC_C  EQU 1FH
PPIC    EQU 1DH
PPIB    EQU 1BH
PPIA    EQU 19H
    ;
INTA    EQU
INTA2   EQU
    ;
INT_V   EQU 42H*4; for service routine
    ;
STACK   EQU 540H
    ;
    ORG 1000H
    ;
    XOR BX,BX
    MOV ES,BX
    MOV DS,AX
    MOV SS,BX
    MOV SP,STACK
    ;
    MOV AX,OFFSET INT_SER
    MOV BX,INT_V
    MOV WORD PTR ES:[BX],AX
    ;
    XOR AX,AX
    MOV WORD PTR ES:[BX+2],AX
    ;
    CALL    INIT
```



```
34      ;Initialize
35      ;PPIC_C
36      ;PPIB
37      ;PPIC
38
39      MOV SI,OFFSET DATA     ; you may use different ways (Exp9)
40      MOV AL,BYTE PTR CS:[SI]
41      OUT PPIA,AL
42      ;
43      STI
44
45      ;Infinit Loop
46  L2: NOP
47      JMP L2
48      ;
49      ;
50  INT_SERVICE:
51      ;
52  L3: OUT PPIA,AL
53      INC SI
54      ;  EOI command
55      STI
56      IRET
57  INIT    PROC    NEAR
58      ;ICW1
59      ;ICW2 interrupt vector
60      ;ICW4
61      ;interrupt mask
62      RET
63  INIT    ENDP
64      ;
65  DATA: DB
66  CODE    ENDS
67  END
```

Figure 6 Code 3

Step2: Compile and build this ASM file and execute it on MDA-8086 kit. (How? Review Exp#1 Intro. To MDA Kit)

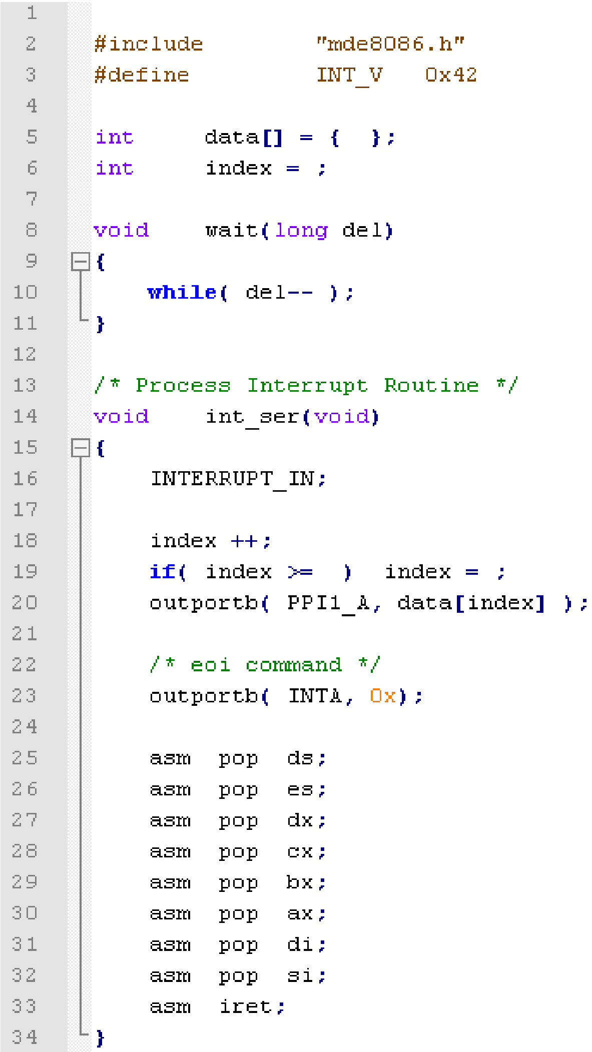
TASKS:

1. Explain what does this code do?
2. What is the difference between Code 3 and Code 2, are they checking the Interrupt occurrence in the same way?
3. How come results changes on the Seven Segment while the CPU enters an infinite loop? (Look @ line 46)

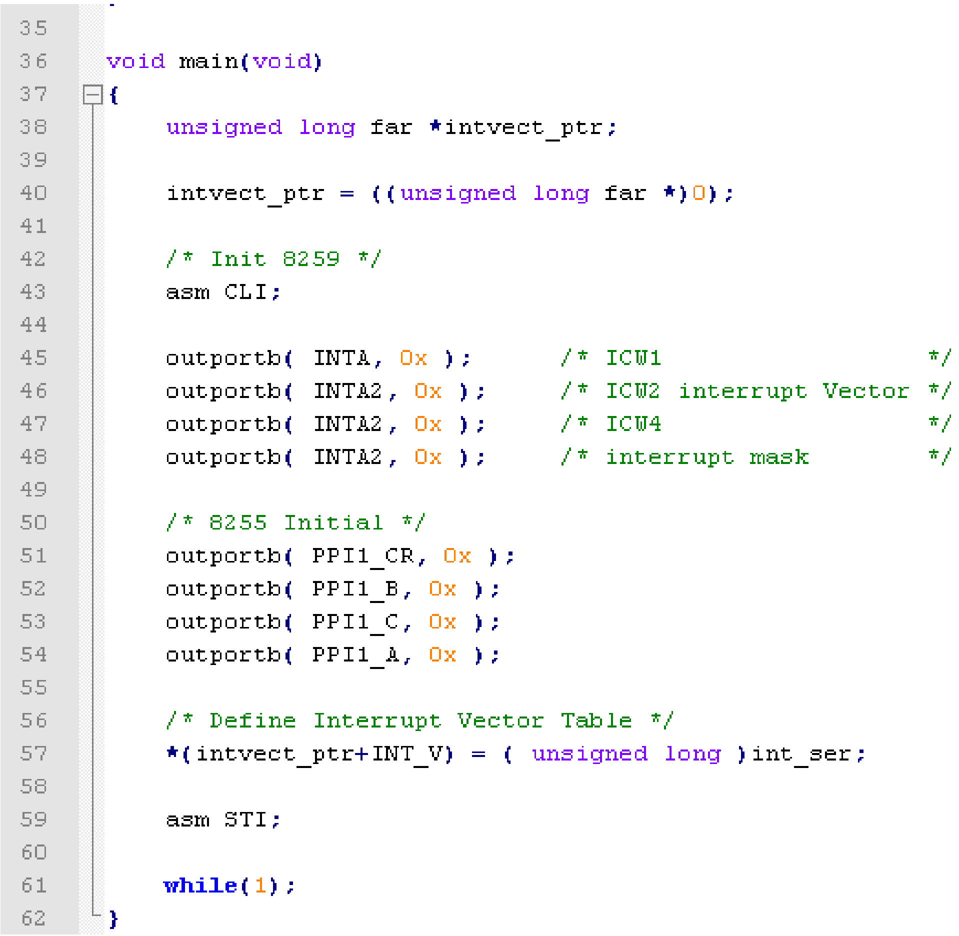
3.4 PRACTICE IV: 8259a Interrupt Control (Interruption Technique) (C Code)

Practice Goal: We need to use C programming to perform Practice III, but for counting from A to F?

Setp1: Write down the needed code and save it to C file to achieve the required goal. (P.S.: User the following code. Complete it as required)



```

#include        "mde8086.h"
#define         INT_V   0x42

int     data[] = {  };
int     index = ;

void    wait(long del)
{
    while( del-- );
}

/* Process Interrupt Routine */
void    int_ser(void)
{
    INTERRUPT_IN;

    index ++;
    if( index >=  )  index = ;
    outportb( PPI1_A, data[index] );

    /* eoi command */
    outportb( INTA, 0x);

    asm  pop  ds;
    asm  pop  es;
    asm  pop  dx;
    asm  pop  cx;
    asm  pop  bx;
    asm  pop  ax;
    asm  pop  di;
    asm  pop  si;
    asm  iret;
}
```



```

void main(void)
{
    unsigned long far *intvect_ptr;

    intvect_ptr = ((unsigned long far *)0);

    /* Init 8259 */
    asm CLI;

    outportb( INTA, 0x );       /* ICW1                   */
    outportb( INTA2, 0x );      /* ICW2 interrupt Vector */
    outportb( INTA2, 0x );      /* ICW4                   */
    outportb( INTA2, 0x );      /* interrupt mask         */

    /* 8255 Initial */
    outportb( PPI1_CR, 0x );
    outportb( PPI1_B, 0x );
    outportb( PPI1_C, 0x );
    outportb( PPI1_A, 0x );

    /* Define Interrupt Vector Table */
    *(intvect_ptr+INT_V) = ( unsigned long )int_ser;

    asm STI;

    while(1);
}
```

Figure 7 Code 7

Step2: Compile and build this ASM file and execute it on MDA-8086 kit. (How? Review Exp#1 Intro. To MDA Kit)

TASKS:

1. Explain what does this code do (**Line by Line**)?
2. What does "**asm**" Instruction mean?
3. Explain how can you find what does "INTURREPT_IN" instruction does?

Bibliography

Tech., MEDAS. 2008. *MDA 8086 Kit User Manual.* Korea : s.n., 2008.