Experiment #6

Interrupts in Personal Computers

Birzeit University
Faculty of Engineering and Technology
Department of Electrical and Computer Engineering

Abstract

This experiment aims at understanding and expanding the concept of interruption. Students will learn how to use and call Interrupt Service Routines (ISR) based on MS Windows XP OS. Besides, creating a new ISR using debug and TASM will be a main task of the experiment.

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 non-maskable 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 of 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.

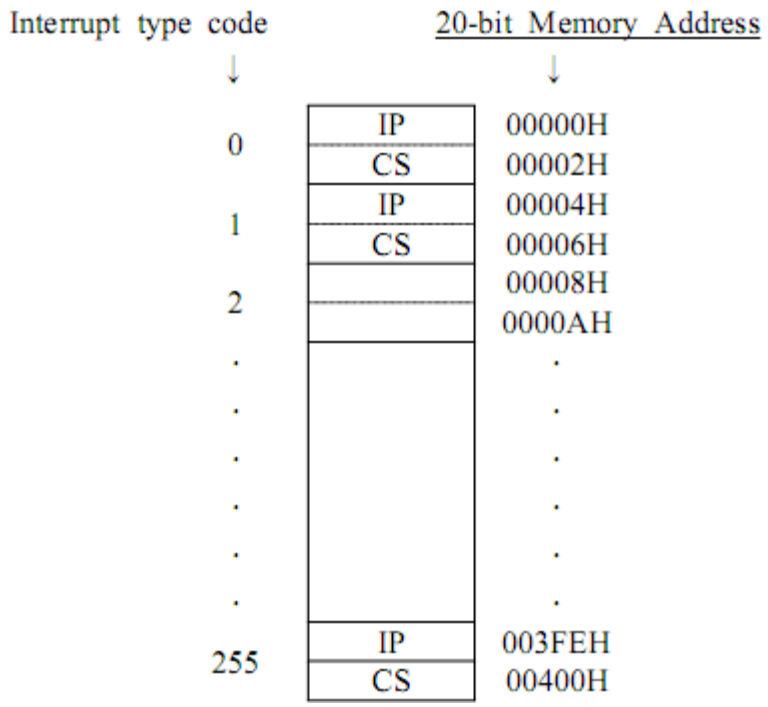

Figure 1 Interrupt Vector Table

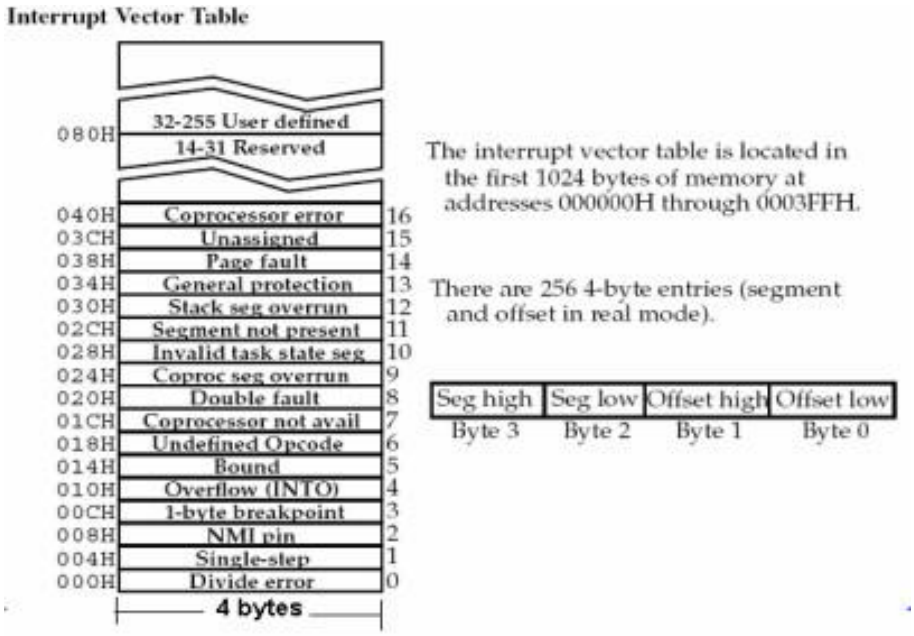

Figure 2 Interrupt Vector Table

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 to 255 are available for maskable interrupts.

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

PART II Pre-Lab

1. Briefly explain the following terms:
   - ISR [Interrupt Service Routine]
   - IVT [Interrupt Vector Table]

2. What's the address of CS & IP of the ISR of INT 62H?

PART III Practices

3.1 PRACTIC I: Activating the Interrupt Service Routine Manually

You have notices that Interrupt Vector table contains addresses for Interrupt Service Routines (ISPs). These Routines can be activated by executing their code. One way to do so is using "**INT n**[1]" Assembly instruction. These instructions activate the ISP its address loaded in vector number"**n**".

Step 1: Open Command Dos Terminal and execute "Debug".

Step 2: Assemble the following code to offset 100.

-INT 0

Step 3: Run the program using G (Make sure the IP value is 100).

TASKS:

1. Explain what does "INT 0" do?

3.2 PRACTICE II: Activating the Interrupt Service Routine Automatically

Sometimes, ISP can be called automatically as a response of a specific situation or operation. For example, ISP 0 is activated whenever there is a division over zero.

Step 1: Run the Debug program.

Step2: Assemble the following program

```
MOV AX, 1234
MOV CL,0
DIV CL
INT 3
```

Step 3: Run the program using G.

TASKS:

1. Explain the result (Notice Figure 3)?
2. INT 3 used to set break point and stops the executions. If this instruction was removed, will there be any changes in the result?

[1] n: Interrupt number

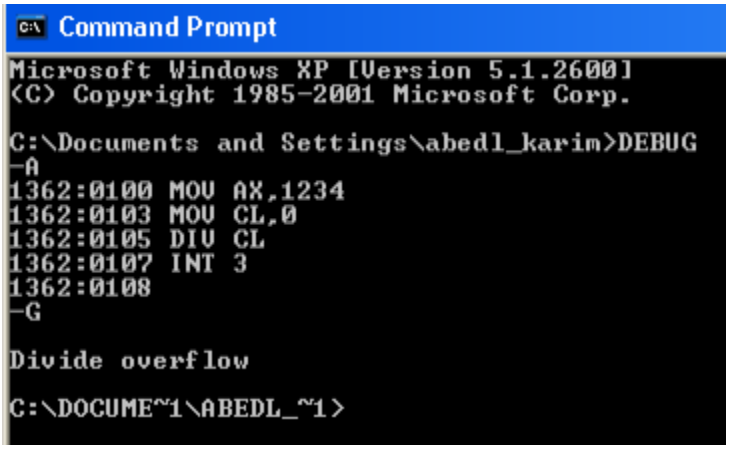

Figure 3 Activating ISP Automatically

3.3 PRACTICE III: Activating Interrupt Service Routine Manually (Another Way)

What happened in the first practice was calling the ISP using an assembly instruction. This instruction actually refers to the Interrupt Vector table each time it is executed. It goes there and gets the address for the specified interrupt number, all what it needs is getting the Segment address and the offset.

What we are trying to do in this practice is changing the **CS** and **IP** to the address of the ISP number 0 and executing using G. This supposed to work as it works with "**INT 0**" instruction.

Step 1: Run the debug program.

Step 2: Run the R command and check the content of the registers.

Finding Out the Address for ISP # 0

Step 3: The vector table is located in the address **0:0** on the memory, Display the content of memory at this address (Hint: use D command).

Step 4: The address of interrupt service routine corresponding to INT 0 is located the range 0-3 as shown in table below

3	Segment (high)
2	Segment (low)
1	Offset (high)

0	Offset (low)

Find out the values for ISP # 0 address.

(**P.S.:** The values of segment and offset below are not necessary the same on your PC)

Step 5: We can run the interrupt service routine by making the CS register points to the code segment that the interrupt service routine located in it and making the IP register points to the start address of the interrupt service routine. (Hint: Use R command)

Step 6: Use G to execute the program start at address **CS:IP**.

TASKS:

1. You should notice that the results of this practice and Practice I are the Same.
2. What is the routine address of INT F stored in the Interrupt Vector Table?

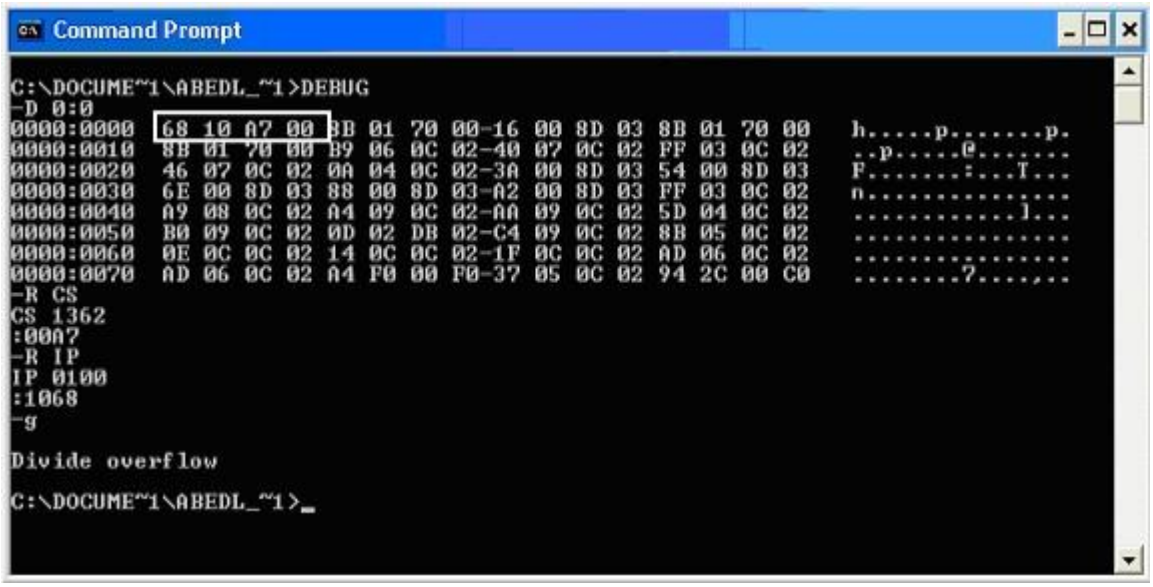

Figure 4 Activating the ISP Manually (Another Way)

3.4 PRACTICE IV: Creating Your Own ISP

Now, to change the INT response. What about replacing the code of ISP 0? or maybe creating a new code and replace the vector values with its address? We can write our routine and make it run when dived by zero occur by putting the address (segment and offset) of our new routine in memory location 0-3.

Out task: is to rewrite the interrupt routine for INT 0 to display AAAA when a divide by zero occur.

Step 1: Run the debug program.

Step 2: Assemble the following program to offset 100

```
MOV CX,4
L1: MOV DL,41
MOV AH,6
INT 21
LOOP L1
INT 3
```

Notice that the first instruction should be at address CS:0100.

Step 3: Change the address of interrupt service routine corresponding to INT 0 To be CS:0100 (Hint: Use E command) (Notice Figure 5).

Figure 5 Changing the Values of INT 0 Vector Entry

Step 4: Now we can test the new interrupt routine by making a program that makes a division by zero (e.g. Practice II)

Code Example:

```
MOV CL,0
MOV AX,5043
DIV CL
INT 3
```

Notice that when we execute the program AAAA is displayed instead of divide overflow (Notice Figure 6).

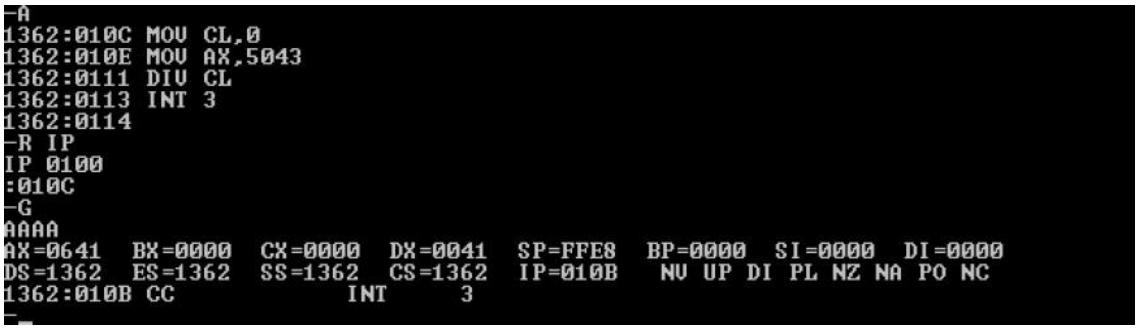

Figure 6 Test the new INT 0

3.4 PRACTICE V: Creating Your Own ISP Using TASM

In this practice we would work on installing a new interrupt service routine for interrupt number 62.

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

```
.MODEL TINY
.CODE
.STACK 100H

JMP INSTALL
MESS DB "WELCOME TO COOLEST LAB ON EARTH ENCS411 :D $"

MYINT PROC

   MOV BX,OFFSET MESS
   MOV DL,44  ;column 20 (# of mess chars)

L1:
   MOV AL ,CS:[BX]
   CMP al, '$'
   JZ L2

   PUSH BX

   MOV BH,0 ;page number 0
   MOV AH,2   ;set the curser position
   MOV DH,20 ;row 0
   INC DL
   INT 10H   ;to access  video memory

   MOV AH,9H ;Write character and attribute at cursor position

   MOV BH,0 ;page number 0
   MOV BL,84H ;color of the character
   MOV CX,1 ;number of times to write the character
   INT 10H ;video memory access

   POP BX
   INC BX
   JMP L1

L2:
   IRET
MYINT ENDP

INSTALL:
 ;install new interrupt vector
   MOV AH,25H ;set interrupt vector
   MOV AL,62H ;desired interrupt vector
   MOV DX,CS
   MOV DS,DX
   MOV DX, OFFSET MYINT
   INT 21H

; Terminate & stay resident

   MOV AX,3100H
   INT 21H

END
```

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

Step 3: To test the code

- Open debug.
- Execute the instruction "INT 62". What is the result?

Another way to test the code:

- Write the following code to an assembly file

```
.MODEL SMALL
.STACK 100
.CODE
   INT 62H
   mov AH,4CH
   INT 21H
END
```

- Compile and run it. What is the result?

TASKS:

1. Explain what does this code do (Line by line)?
2. What does INT 21 do?