LAB #11
Interrupts

INT Interrupt:

An interrupt is either a hardware-generated CALL (externally derived from a hardware signal) or a software-generated CALL (internally derived from the execution of an instruction or by some other internal event).

It is a method of creating a temporary halt during the program execution to allow peripheral devices to access the microprocessor. Microprocessor responds to that interrupt with interrupt Service Routine (ISR), a short program to instruct the microprocessor that how to handle the interrupt.

The microprocessor has three different interrupt instructions that are available to the programmer: INT, INTO, and INT 3. In the real mode, each of these instructions fetches a vector from the vector table, and then calls the procedure stored at the location addressed by the vector.

Syntax: INT type or Value

Example: INT 21H,

There are 256 different software interrupt instructions (INTs) available to the programmer.

For Examples:

INT 100 uses interrupt vector 100, which appears at memory address 190H–193H.

```
MOV ah, 09H          ; Print a string (for printing, put msg or address in DX)
INT 21H
```

```
MOV ah, 0Ah          ; Input a string
INT 21H

MOV AH, 00                ; Clear screen
MOV AL, 02
INT 10H

                          ; To place a cursor on screen
Mov DH, 10                ; ROW
Mov dl, 27           ; Column
Mov AH, 02
Mov BH, 0            ; Page 0
INT 10H
```

The INT 10H instruction calls the interrupt service procedure whose address is stored at some memory location

Whenever a software interrupt instruction executes, it

(1) pushes the flags onto the stack,
(2) clears flag bits,
(3) pushes CS onto the stack
(4) fetches the new value for CS from the interrupt vector,
(5) pushes IP/EIP onto the stack,
(6) fetches the new value for IP/EIP from the vector, and
(7) jumps to the new location addressed by CS and IP/EIP.

Interrupts are used to get the attention of the microprocessor. In the 8086/88 there are a total of 256 interrupts: INT 00, INT 01, INT 02, ..., INT FF.

As mentioned above, the address that has an interrupt jumps in always four times the value of the interrupt number. For example, INT03 will jump to

memory address 0000CH (4 x 03=12=0CH). Table below shows a partial list of interrupts, commonly referred to as the interrupt vector table.

INT # (hex)	Physical Address	Logical Address
INT 00	00000	0000:0000
INT 01	00004	0000:0004
INT 02	00008	0000:0008
INT 03	0000C	0000:000C
INT 04	00010	0000:0010
INT 05	00014	0000:0014
...	...	...
INT FF	003FC	0000:03FC

Every interrupt has a program associated with it called the interrupt service routine (ISR). When an interrupt is invoked, the CS: IP address of its ISR is retrieved from the vector table (shown above). The lowest 1024 bytes (256 x 4= 1024) of RAM are set aside for the interrupt vector table and must not be used for any other function.

Example: -

Find the physical and logical addresses of the vector table associated with a. INT 14H the physical address for INT 14h is 0050H-0053H according to the formula described above i.e. 4 X 14H = 50H. This gives the logical address of 0000:0050H to 0000:0053H.

TYPE 0 interrupt represents division by zero situation.
TYPE 1 interrupt represents single-step execution during the debugging of a program.
TYPE 2 interrupt represents non-mask-able NMI interrupt.
TYPE 3 interrupt represents break-point interrupt.
TYPE 4 interrupt represents overflow interrupt.

This applies for any number between1-9

INT 21h / AH=1 - read character from standard input, with echo, result is stored in **AL**. If there is no character in the keyboard buffer, the function waits until any key is pressed.

```
mov ah, 5
mov dl, 'a'
int 21h
```

With this chunk of code, you can actually output the value in dl to the screen

AAA (ASCII AJDUST AFTER ADDITION)

Corrects result in AH and AL after addition when working with BCD values.

Algorithm: -

If low nibble of
AL>9 or AF=1
then: AL=AL+6

AH=AH+1
AF=1
CF=1 else
AF=0 CF=0
In both the cases clear the high nibble of AL.

Example:

```
MOV AL, 31H        ; AL=31 THE ASCII CODE FOR 1
ADD AL, 37H        ; ADD 37 (ASCII FOR 7) TO AL; AL=68H
AAA                ; AL=08 AND CF=0
```

In the example above, ASCII (31) is added to ASCII 7 (37). After the AAA instruction, AL will contain 8 in BCD and CF =0. The following example shows another ASCII addition and then the adjustment.

Example 1:

Program that accept two values from the user and show the sum in AL. mov ax,0h

```
mov ah, 01h                    ; accepts the character in al
int 21h
AAA
mov dl, al
mov ax,0h
mov ah,01
int 21h                        ; (take input from user and store in AL with ascii number)
AAA
add al, dl
```

Example 2:

Program that takes the value from user and show the output value of DX register in AL.

```
mov ax,0h
mov ah, 01h                    ; accepts the character in al
int 21h
aaa
mov dl, al                     ; outputs the character from dl
mov ah,02h
int 21h
```

Example:

```
org 100h

MOV ah, 0Ah                    ; Input a string
INT 21H
```

```
mov ah,02h
mov dl,0Ah      ; new line
INT 21H
mov dl,0dh
int 21h


MOV ah, 01h             ; Input a number
INT 21H

MOV ah, 02h             ; Output a number
mov dl, '4'
INT 21H


MOV AH, 00h                     ; Clear screen
MOV AL, 02h
INT 10H


lea dx, msg
mov ah, 09h
int 21h

msg db "Hello INT",'$'
```

Example to take input from user and add the numbers

```
org 100h

.data
a db 0ah,0dh, "Enter 1st Number: $"

b db 0ah,0dh, "Enter 2nd Number: $"

c db 0ah,0dh, "Your answer is: $"
```

```
.code

mov ax, @data
mov ds, ax

lea dx,a

mov ah, 09      ; Display
int 21h

mov ah, 01h     ; input 1st number
int 21h

mov bl, al

lea dx,b

mov ah, 09      ; Display
int 21h

mov ah, 01h     ; input 2nd number
int 21h

sub al, 30h
sub bl, 30h

add bl, al
add bl, 30h

mov ah,00h
mov al, 02h
int 10h

lea dx,c

mov ah, 09      ; Display
int 21h
```

```
mov dl, bl
mov ah, 02
int 21h

ret
```

Lab Tasks

Execute the following tasks.

Task 1:

Take input from the user, increment that and show the incremented value at the output Take the input from the user, decrement it and show the decremented value at the output.

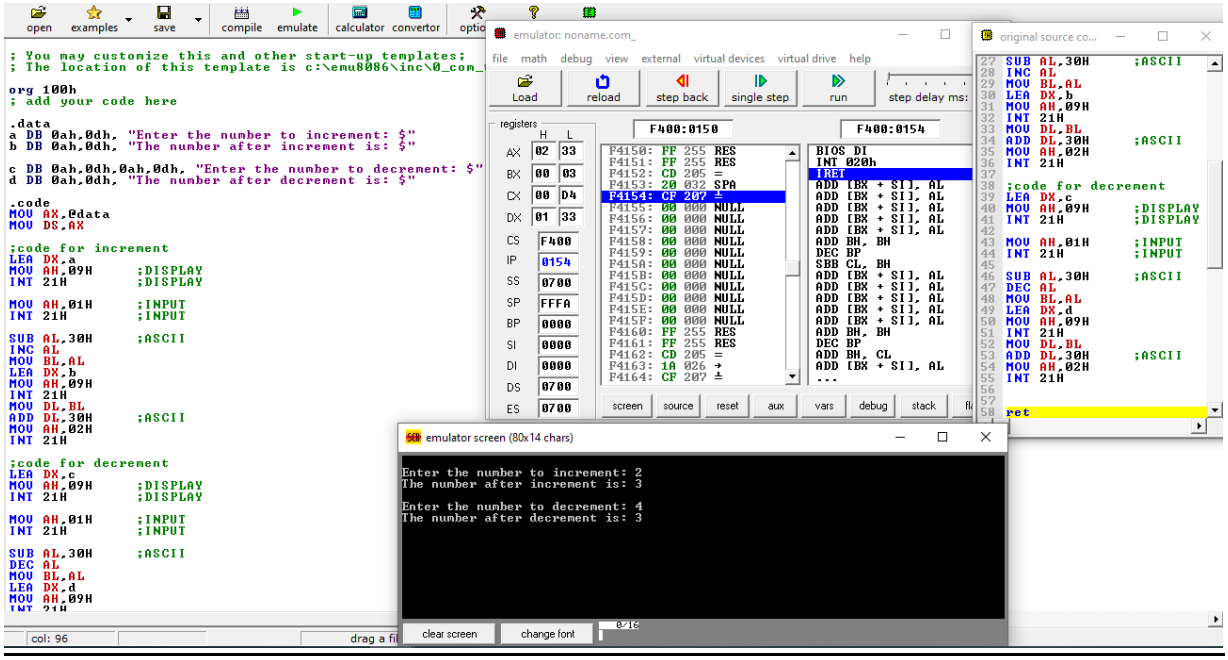
SOURCE CODE:

```
org 100h
; add your code here

.data
a DB 0ah,0dh, "Enter the number to increment: $"
b DB 0ah,0dh, "The number after increment is: $"

c DB 0ah,0dh,0ah,0dh, "Enter the number to decrement: $"
d DB 0ah,0dh, "The number after decrement is: $"

.code
MOV AX,@data
MOV DS,AX

;code for increment
```

```
LEA DX,a
MOV AH,09H    ;DISPLAY
INT 21H       ;DISPLAY

MOV AH,01H    ;INPUT
INT 21H       ;INPUT

SUB AL,30H    ;ASCII
INC AL
MOV BL,AL
LEA DX,b
MOV AH,09H
INT 21H
MOV DL,BL
ADD DL,30H    ;ASCII
MOV AH,02H
INT 21H

;code for decrement
LEA DX,c
MOV AH,09H    ;DISPLAY
INT 21H       ;DISPLAY

MOV AH,01H    ;INPUT
INT 21H       ;INPUT

SUB AL,30H    ;ASCII
DEC AL
MOV BL,AL
LEA DX,d
MOV AH,09H
INT 21H
MOV DL,BL
```

```
ADD DL,30H     ;ASCII
MOV AH,02H
INT 21H

ret
```

OUTPUT:



Task 2:

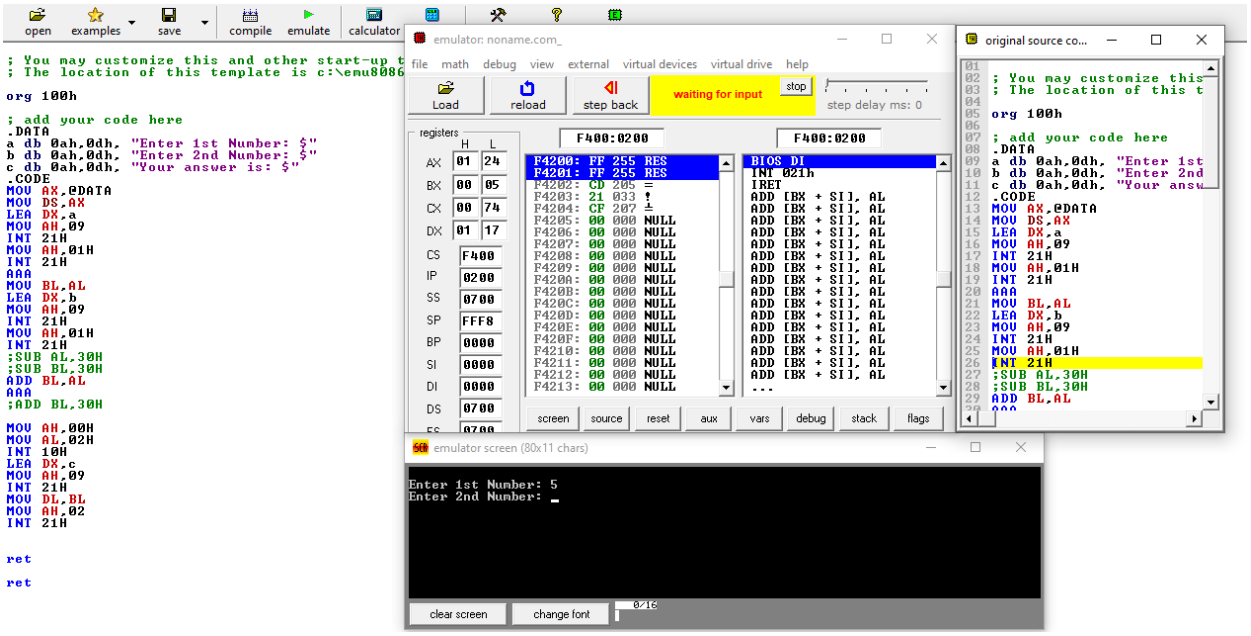
Take the ASCII value 31 and 35 from user and show the result of their sum?

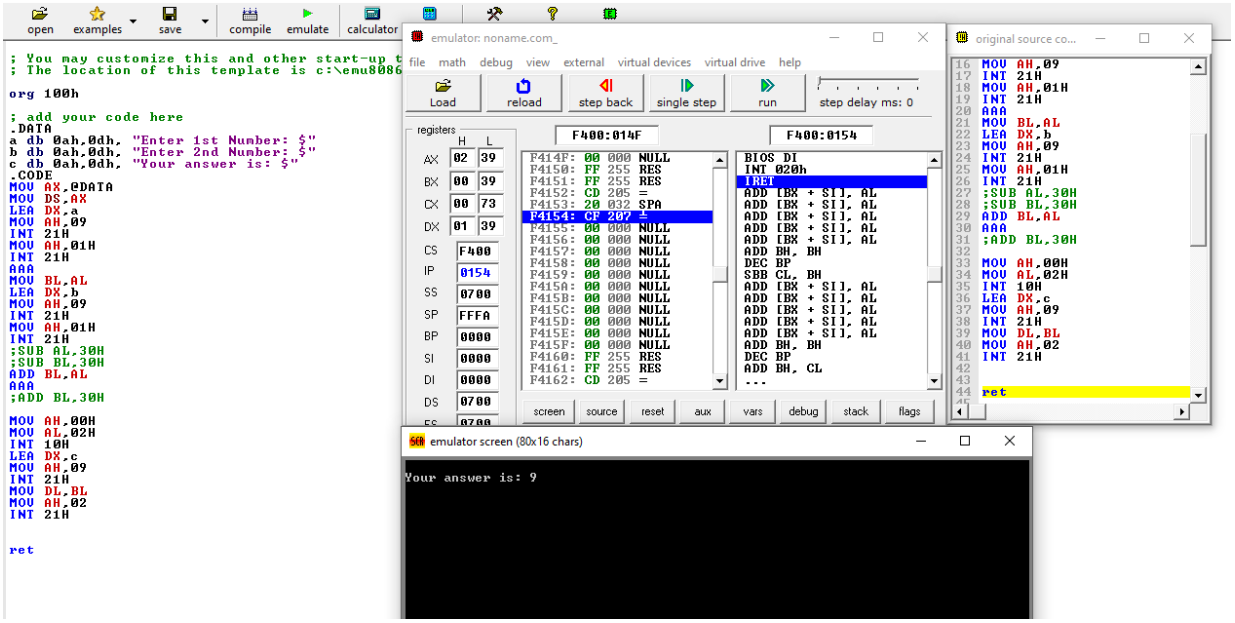
SOURCE CODE:

```
org 100h

; add your code here
.DATA
a db 0ah,0dh, "Enter 1st Number: $"
b db 0ah,0dh, "Enter 2nd Number: $"
c db 0ah,0dh, "Your answer is: $"
```

```
.CODE
MOV AX,@DATA
MOV DS,AX
LEA DX,a
MOV AH,09
INT 21H
MOV AH,01H
INT 21H
AAA
MOV BL,AL
LEA DX,b
MOV AH,09
INT 21H
MOV AH,01H
INT 21H
;SUB AL,30H
;SUB BL,30H
ADD BL,AL
AAA
;ADD BL,30H

MOV AH,00H
MOV AL,02H
INT 10H
LEA DX,c
MOV AH,09
INT 21H
MOV DL,BL
MOV AH,02
INT 21H


ret
```

OUTPUT:





Task 3:

Design a calculator that takes three input numbers from user and perform addition, subtraction, division and multiplication on two of them and return the output value to the user. Third value is

used to decide which operation it has to perform e.g. 1=add, 2=sub…..

SOURCE CODE:

```
org 100h

; add your code here
.DATA
a DB 0ah,0dh,'For Addition type:'1'$'
b DB 0ah,0dh,'For Subtraction type:'2'$'
c DB 0ah,0dh,'For Multiplication type:'3'$'
d DB 0ah,0dh,'For Division type:'4'$'
e DB 0ah,0dh,'Choose Any One:$'
f DB 0ah,0dh,'Enter 1st Number:$'
g DB 0ah,0dh,'Enter 2nd Number:$'
h DB 0ah,0dh,'The Result is:$'
NUM1 DB ?
NUM2 DB ?
RESULT DB ?
.CODE
CALCULATOR PROC
 MOV AX,@DATA
 MOV DS,AX
 LEA DX,a
 MOV AH,9
 INT 21H
 LEA DX,b
 MOV AH,9
 INT 21H
 LEA DX,c
 MOV AH,9
 INT 21H
```

```
LEA DX,d
MOV AH,9
INT 21H
LEA DX,e
MOV AH,9
INT 21H
MOV AH,1
INT 21H
MOV BH,AL
SUB BH,30H
CMP BH,1
JE ADD
CMP BH,2
JE SUB
CMP BH,3
JE MUL
CMP BH,4
JE DIV

ADD:
LEA DX,f  ;ENTER 1ST NUMBER
MOV AH,09
INT 21H
MOV AH,01
INT 21H
MOV BL,AL
LEA DX,g    ;ENTER 2ND NUMBER
MOV AH,09
INT 21H
MOV AH,01
INT 21H
MOV CL,AL
ADD AL,BL
```

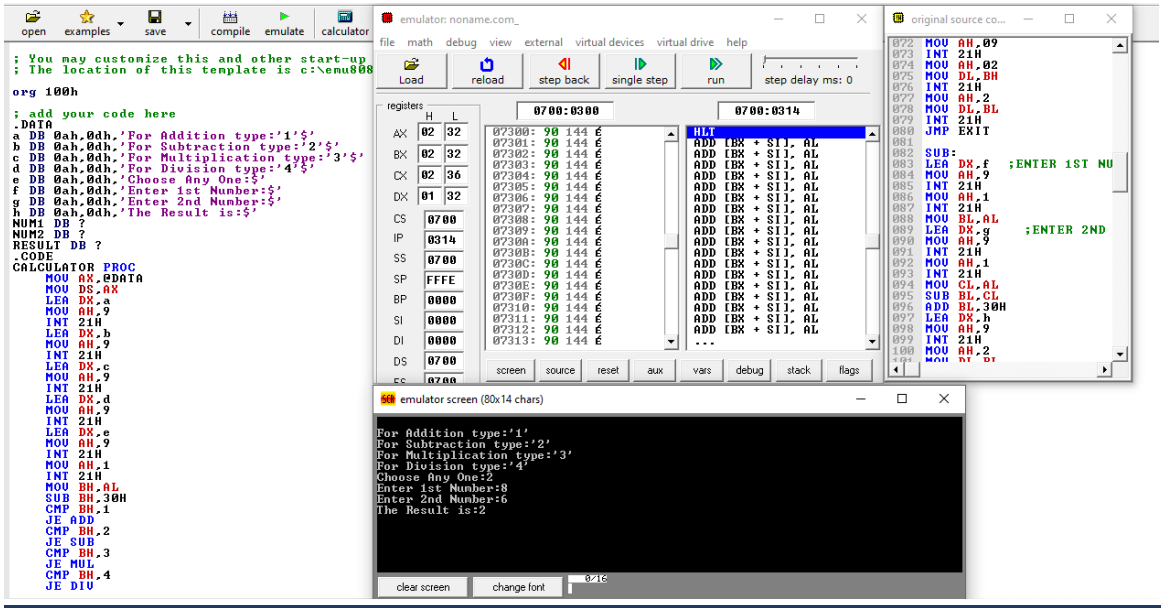
```
MOV AH,0
AAA
MOV BX,AX
ADD BH,30H
ADD BL,30H
LEA DX,h
MOV AH,09
INT 21H
MOV AH,02
MOV DL,BH
INT 21H
MOV AH,2
MOV DL,BL
INT 21H
JMP EXIT

SUB:
LEA DX,f  ;ENTER 1ST NUMBER
MOV AH,9
INT 21H
MOV AH,1
INT 21H
MOV BL,AL
LEA DX,g    ;ENTER 2ND NUMBER
MOV AH,9
INT 21H
MOV AH,1
INT 21H
MOV CL,AL
SUB BL,CL
ADD BL,30H
LEA DX,h
MOV AH,9
```

```
INT 21H
MOV AH,2
MOV DL,BL
INT 21H
JMP EXIT
MUL:
LEA DX,f
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV NUM1,AL
LEA DX,g
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV NUM2,AL
MUL NUM1
MOV RESULT,AL
AAA
ADD AH,30H
ADD AL,30H
MOV BX,AX
LEA DX,h
MOV AH,9
INT 21H
MOV AH,2
MOV DL,BH
INT 21H
MOV AH,2
```

```
MOV DL,BL
INT 21H
JMP EXIT
DIV:
LEA DX,f
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV NUM1,AL
LEA DX,g
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV NUM2,AL
MOV CL,NUM1
MOV CH,00
MOV AX,CX
DIV NUM2
MOV RESULT,AL
MOV AH, 00
AAD
ADD AH,30H
ADD AL,30H
MOV BX,AX
LEA DX,h
MOV AH,9
INT 21H
MOV AH,2
MOV DL,BH
```

```
INT 21H
MOV AH,2
MOV DL,BL
INT 21H
JMP EXIT
EXIT:
CALCULATOR ENDP
END CALCULATOR

ret
```

OUTPUT:



----------THE END----------