Name:	Pankhania Aanandi R.
Roll No:	IT081
Batch:	I1

Experiment 5

AIM: To Study the response of Type-0 interrupt.

1. Write an assembly language program of dividing four numbers. If the result of the division is too large to fit in the quotient register then the 8086 will do a type 0 interrupt immediately after the divide instruction finishes.
   - Write two programs one is main line program which contains div instruction and second program is interrupt service routine which handles the type 0 interrupt.

Rules for Operands:

1. You have to use following values as dividend
   DIVIDEND DW 00ABh,0CDEh,7FFFh,0FFFFh

2. You have to use the ASCII value (in hex) of the first 1-letters of your name as a **DIVISOR**.
   E.g. According to my surname (VITHLANI), my DIVISOR is: 56h

LETTER (use UPPERCASE letters)	ASCII Value in Hex
V	56h

3. Clearly mention ASCII values of surname and then write your program.

Write your code here:

According to my surname (PANKHANIA), my DIVISOR is: 56h

LETTER (use UPPERCASE letters)	ASCII Value in Hex
P	50h

---

1. **isrexp.asm**

```
DATA_HERE SEGMENT WORD PUBLIC
        INPUT DW 00ABH,0CDEH,7FFFH,0FFFFH
        QUOTIENTS DB 4 DUP(0)
        DIVISOR DB 50H ; DIVISOR P(50H)
        FLAGS DB 4 DUP (0)
        EFLAG DB 0 ; ERROR FLAG
DATA_HERE ENDS
```

```
STACK_HERE SEGMENT STACK
        DW 100 DUP(0)
        STACK1 LABEL WORD
STACK_HERE ENDS

PUBLIC EFLAG

        PROC_HERE SEGMENT WORD PUBLIC
        EXTRN DIV_PROC : FAR
        PROC_HERE ENDS

CODE_HERE SEGMENT WORD PUBLIC

                ASSUME CS:CODE_HERE , DS:DATA_HERE , SS:STACK_HERE
START:          MOV AX , STACK_HERE
                MOV SS , AX
                MOV SP , OFFSET STACK1
                MOV AX , DATA_HERE
                MOV DS , AX
                MOV AX,0000
                MOV ES, AX

                ;CHANGE INTERRUPT TYPE0

                MOV WORD PTR ES:0002,SEG DIV_PROC
                MOV WORD PTR ES:0000,OFFSET DIV_PROC
                MOV SI,OFFSET INPUT
                MOV BX,OFFSET QUOTIENTS
                MOV DI,OFFSET FLAGS
                MOV CX,0004
NEXT:  MOV AX,[SI]
                DIV DIVISOR
                CMP EFLAG,01
                JNE NXT
                MOV BYTE PTR[BX],00
                MOV BYTE PTR[DI],01
                JMP NXT1
NXT:    MOV [BX],AL
                MOV BYTE PTR[DI],00
NXT1:  MOV EFLAG,00
                ADD SI,02H
                INC BX
                INC DI
                LOOP NEXT
STOP: NOP
CODE_HERE ENDS
END START
```

2. **isrdiv.asm**

```
        DATA_HERE SEGMENT WORD PUBLIC
        EXTRN EFLAG: BYTE
DATA_HERE ENDS

PUBLIC DIV_PROC

PROC_HERE SEGMENT WORD PUBLIC
        DIV_PROC PROC FAR
                ASSUME CS:PROC_HERE, DS:DATA_HERE
                PUSH AX
                PUSH DS
                PUSH BX

                MOV AX, DATA_HERE
                MOV DS,AX

                MOV BP, SP ; INCREMENT IP BY 4
                MOV BX, WORD PTR [BP+6]
                ADD BX, 04H
                MOV [BP+6],BX

                MOV EFLAG,01 ; SET EFLAG(ERROR FLAG) 1

                POP BX
                POP DS
                POP AX
                IRET
            DIV_PROC ENDP
        PROC_HERE ENDS
```

END

Compilation /Running and Debugging steps:

- Clearly mention each step

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Pro...
A:\>tasm isrdiv.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   isrdiv.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  476k


A:\>tasm isrexp.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   isrexp.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  475k


A:\>tlink isrexp.obj isrdiv.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
Warning: No stack

A:\>_
```

```
A:\>tasm isrexp
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   isrexp.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  475k


A:\>
```

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Pro...

A:\>debug isrexp.exe
-t

AX=076C  BX=0000  CX=0158  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=077A  IP=0007   NV UP EI PL NZ NA PO NC
077A:0007 8ED0          MOV     SS,AX
-t

AX=076C  BX=0000  CX=0158  DX=0000  SP=00C8  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076C  CS=077A  IP=000C   NV UP EI PL NZ NA PO NC
077A:000C B86A07        MOV     AX,076A
-t

AX=076A  BX=0000  CX=0158  DX=0000  SP=00C8  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076C  CS=077A  IP=000F   NV UP EI PL NZ NA PO NC
077A:000F 8ED8          MOV     DS,AX
-t

AX=076A  BX=0000  CX=0158  DX=0000  SP=00C8  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076C  CS=077A  IP=0011   NV UP EI PL NZ NA PO NC
077A:0011 B80000        MOV     AX,0000
-
```

- Put a screenshot of the mapping file. (Generated after linkage of object files)

```
ISREXP.MAP

Start  Stop   Length Name              Class

00000H 00011H 00012H DATA_HERE
00020H 000E7H 000C8H STACK_HERE
000E8H 00103H 0001CH PROC_HERE
00104H 00157H 00054H CODE_HERE
00158H 00158H 00000H DATA

Error: Fixup overflow in module ISRDIV.ASM at PROC_HERE:0015, target = EF
Program entry point at 0010:0004
Warning: No stack
```

```
A:\>type isrexp.map

 Start  Stop   Length Name              Class

 00000H 00011H 00012H DATA_HERE
 00020H 000E7H 000C8H STACK_HERE
 000E8H 00103H 0001CH PROC_HERE
 00104H 00157H 00054H CODE_HERE

Program entry point at 0010:0004
Warning: No stack


A:\>
```

Output:

1. Screenshot of memory after each iteration of loop. (See below screenshot for more clarification)

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Pro...

A:\>debug isrexp.exe
-t

AX=076C  BX=0000  CX=0158  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=077A  IP=0007   NV UP EI PL NZ NA PO NC
077A:0007 8ED0          MOV     SS,AX
-t

AX=076C  BX=0000  CX=0158  DX=0000  SP=00C8  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076C  CS=077A  IP=000C   NV UP EI PL NZ NA PO NC
077A:000C B86A07        MOV     AX,076A
-t
```

2. Our 1st number is 00ABh, so in the 1st screenshot highlight this number from memory and its quotient & division flag stored in memory. As below screenshot.

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Pro...
-t

AX=0B02  BX=0008  CX=0004  DX=0000  SP=00C8  BP=0000  SI=0000  DI=000D
DS=076A  ES=0000  SS=076C  CS=077A  IP=0050   NV UP EI NG NZ AC PE CY
077A:0050 83C602        ADD     SI,+02
-t

AX=0B02  BX=0008  CX=0004  DX=0000  SP=00C8  BP=0000  SI=0002  DI=000D
DS=076A  ES=0000  SS=076C  CS=077A  IP=0053   NV UP EI PL NZ NA PO NC
077A:0053 43            INC     BX
-t

AX=0B02  BX=0009  CX=0004  DX=0000  SP=00C8  BP=0000  SI=0002  DI=000D
DS=076A  ES=0000  SS=076C  CS=077A  IP=0054   NV UP EI PL NZ NA PE NC
077A:0054 47            INC     DI
-t

AX=0B02  BX=0009  CX=0004  DX=0000  SP=00C8  BP=0000  SI=0002  DI=000E
DS=076A  ES=0000  SS=076C  CS=077A  IP=0055   NV UP EI PL NZ NA PO NC
077A:0055 E2D9          LOOP    0030
-d ds:0000 20
076A:0000  AB 00 DE 0C FF 7F FF FF-02 00 00 00 50 00 00 00   ............P...
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00                                                .
-
```

3. Our 2nd number is 0CDEh, so in the 2nd screen shot highlight this number from memory and its quotient & division flag stored in memory. As below screenshot.

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Pro...
077A:0050 83C602        ADD     SI,+02
-t

AX=0E29  BX=0009  CX=0003  DX=0000  SP=00C8  BP=0000  SI=0004  DI=000E
DS=076A  ES=0000  SS=076C  CS=077A  IP=0053   NV UP EI PL NZ NA PO NC
077A:0053 43            INC     BX
-t

AX=0E29  BX=000A  CX=0003  DX=0000  SP=00C8  BP=0000  SI=0004  DI=000E
DS=076A  ES=0000  SS=076C  CS=077A  IP=0054   NV UP EI PL NZ NA PE NC
077A:0054 47            INC     DI
-t

AX=0E29  BX=000A  CX=0003  DX=0000  SP=00C8  BP=0000  SI=0004  DI=000F
DS=076A  ES=0000  SS=076C  CS=077A  IP=0055   NV UP EI PL NZ NA PE NC
077A:0055 E2D9          LOOP    0030
-t

AX=0E29  BX=000A  CX=0002  DX=0000  SP=00C8  BP=0000  SI=0004  DI=000F
DS=076A  ES=0000  SS=076C  CS=077A  IP=0030   NV UP EI PL NZ NA PE NC
077A:0030 8B04          MOV     AX,[SI]                          DS:0004=7FFF
-d ds:0000 10
076A:0000  AB 00 DE 0C FF 7F FF FF-02 29 00 00 50 00 00 00   .........)..P...
076A:0010  00                                                .
-
```

4. Our 3rd number is 7FFFh, so in 3rd screen shot highlight this number from memory and its quotient & division flag stored in memory. As below screen shot.

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Pro...
-t

AX=7FFF  BX=000A  CX=0002  DX=0000  SP=00C8  BP=00BC  SI=0006  DI=000F
DS=076A  ES=0000  SS=076C  CS=077A  IP=0053   NV UP EI PL NZ NA PE NC
077A:0053 43            INC     BX
-t

AX=7FFF  BX=000B  CX=0002  DX=0000  SP=00C8  BP=00BC  SI=0006  DI=000F
DS=076A  ES=0000  SS=076C  CS=077A  IP=0054   NV UP EI PL NZ NA PO NC
077A:0054 47            INC     DI
-t

AX=7FFF  BX=000B  CX=0002  DX=0000  SP=00C8  BP=00BC  SI=0006  DI=0010
DS=076A  ES=0000  SS=076C  CS=077A  IP=0055   NV UP EI PL NZ AC PO NC
077A:0055 E2D9          LOOP    0030
-d ds:0000
076A:0000  AB 00 DE 0C FF 7F FF FF-02 29 00 00 50 00 00 01   .........)..P...
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

5. Our 4th number is 0FFFFh, so in 4th screen shot highlight this number from memory and its quotient & division flag stored in memory. As below screen shot.