combined PDF of all the Practical (LAB-1 to LAB-11)

Name: Vaghani Smit Dhirubhai

Roll No: 166

College Id: 19CEUEG022

Lab 1

1. **Use the FILL command (F) to initialize the 10h storage locations starting at DS:10 with the value 11h, the 10h storage locations starting at address DS:30 with 22h, the 10h storage locations starting at address DS:50 with 33h, and the 10h storage locations starting at address DS:70 with 44h**

Ans:

- initialize the 10h storage locations starting at DS:10 with the value 11h

  ```
  -f 10L 0A 11
  ```

- initialize the 10h storage locations starting at address DS:30 with 22h

  ```
  -f 30L 0A 22
  ```

- initialize the 10h storage locations starting at address DS:50 with 33h

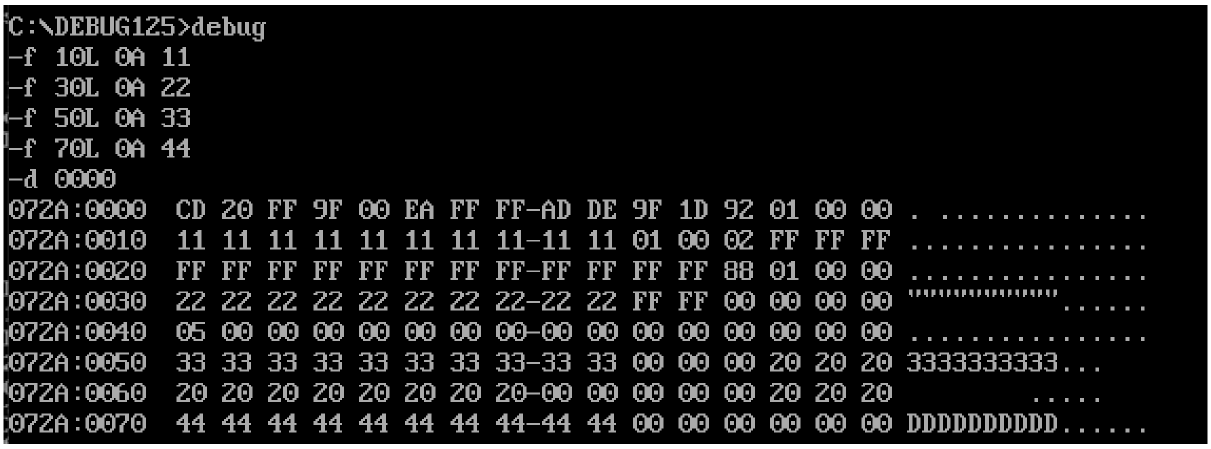
  ```
  -f 50L 0A 33
  ```

- initialize the 10h storage locations starting at address DS:70 with 44h

  ```
  -f 70L 0A 44
  ```

2. Verify the result of step 6 using the DUMP command.

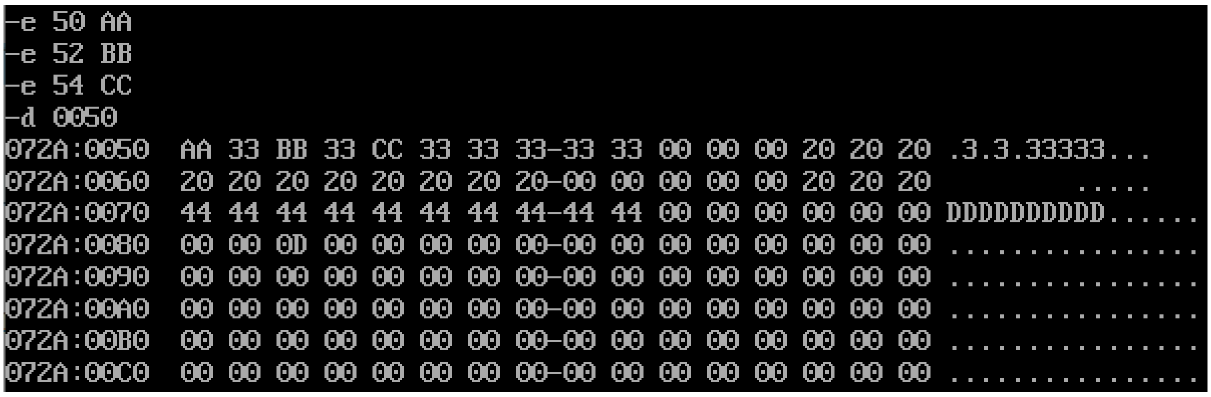
Ans:



```
C:\DEBUG125>debug
-f 10L 0A 11
-f 30L 0A 22
-f 50L 0A 33
-f 70L 0A 44
-d 0000
072A:0000  CD 20 FF 9F 00 EA FF FF-AD DE 9F 1D 92 01 00 00 . ..............
072A:0010  11 11 11 11 11 11 11 11-11 11 01 00 02 FF FF FF ................
072A:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 88 01 00 00 ................
072A:0030  22 22 22 22 22 22 22 22-22 22 FF FF 00 00 00 00 """"""""""......
072A:0040  05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
072A:0050  33 33 33 33 33 33 33 33-33 33 00 00 00 20 20 20 3333333333...
072A:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20         .....
072A:0070  44 44 44 44 44 44 44 44-44 44 00 00 00 00 00 00 DDDDDDDDDD......
```

3.Use the ENTER command (E) to load locations CS:50, CS:52, and CS:54 with AA, BB, and CC, respectively.

Ans:



```
-e 50 AA
-e 52 BB
-e 54 CC
-d 0050
072A:0050  AA 33 BB 33 CC 33 33 33-33 33 00 00 00 20 20 20 .3.3.33333...
072A:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20         .....
072A:0070  44 44 44 44 44 44 44 44-44 44 00 00 00 00 00 00 DDDDDDDDDD......
072A:0080  00 00 0D 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
072A:0090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
072A:00A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
072A:00B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
072A:00C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
```

4. What is the extension of the file produced by the linker?

Ans: .MAP is the extension of the file produced by the linker.

5. Which debug commands allows us to see the memory contents?

Ans: dump(-d) debug commands allows us to see the memory contents.

6. What is the difference between a logical address and a physical address?

Ans: The fundamental difference between logical and physical address is that logical address is generated by CPU during a program execution whereas, the physical address refers to a location in the memory unit.

7. Show how a physical address is generated from a logical address.

Ans: physical address= logical address +offset value

8. What are the following registers used for: DS, CS, SS, SP, IP, AX

Ans:

- **Data segment register (DS):**it points to the data segment of the memory where the data is stored.
- **Code segment Register (CS):** It points to the segment of the running program.
- **Stack Segment Register(SS):**It points to stack segment.
- **Stack Pointer Register(SP):**SP is used to point the current stack.
- **Instruction Pointer Register(IP):**IP denotes the current pointer of the running program. .
- **Accumulator Register(AX)(General Purpose Register):** Most of the arithmetic operation is done with AX.

9. Define the function each of the following flag bits in the flag register: Overflow, Carry, Sign, and Zero.

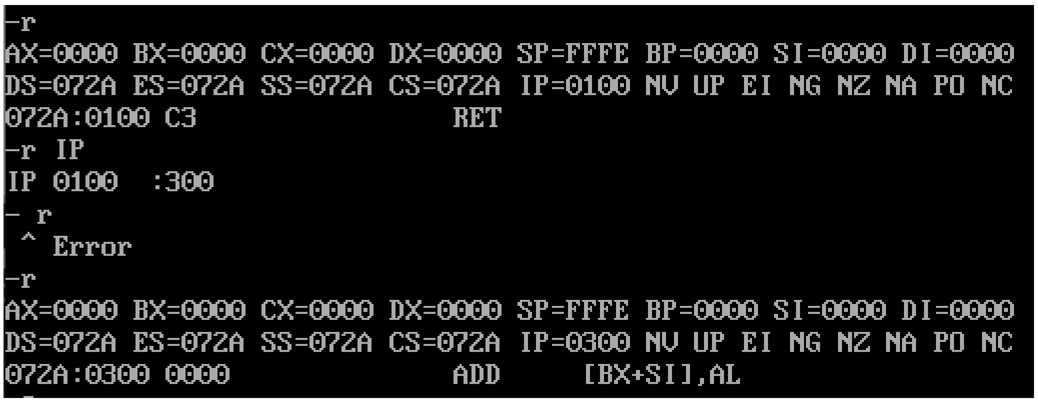
Ans:

- **Overflow flag:** this flag register set if arithmetic operation of two number is overflow otherwise reset.
- **Carry flag:** this flag register set if addition or subtraction of two number and carry or borrow generated otherwise reset.

- **Sign flag:** this flag register set if last operation result is negative otherwise reset.
- **Zero:** this flag register set if last operation result is zero otherwise reset.

10. Use a REGISTER command to first display the current contents of IP and then change this value to 0300h.

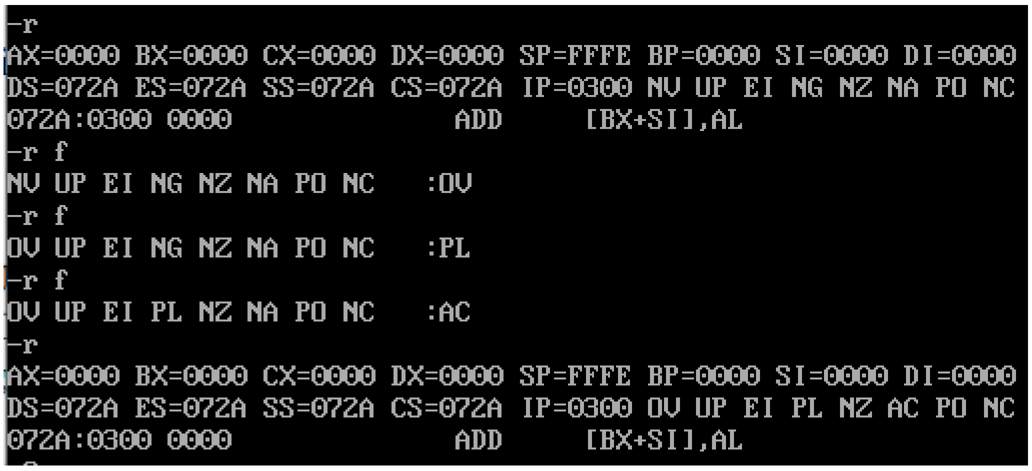
Ans:



```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0100 NV UP EI NG NZ NA PO NC
072A:0100 C3               RET
-r IP
IP 0100  :300
- r
 ^ Error
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0300 NV UP EI NG NZ NA PO NC
072A:0300 0000             ADD     [BX+SI],AL
```

11. Use a REGISTER command to first display the current contents of the flag register and then reset the overflow, sign, and auxiliary flags.

Ans:



```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0300 NV UP EI NG NZ NA PO NC
072A:0300 0000             ADD     [BX+SI],AL
-r f
NV UP EI NG NZ NA PO NC   :OV
-r f
OV UP EI NG NZ NA PO NC   :PL
-r f
OV UP EI PL NZ NA PO NC   :AC
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0300 OV UP EI PL NZ AC PO NC
072A:0300 0000             ADD     [BX+SI],AL
```

12. Using the ASSEMBLE command (A), load the program shown below into memory starting at address CS: 0100.

a. program

b. Verify the loading of the program by displaying it with the UNASSEMBLE (U) command.

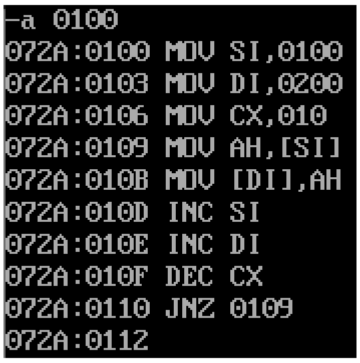
c. How many bytes of memory does the program take up?

d. What is the machine code for the DEC CX instruction?

e. What is the address offset for the label BACK?

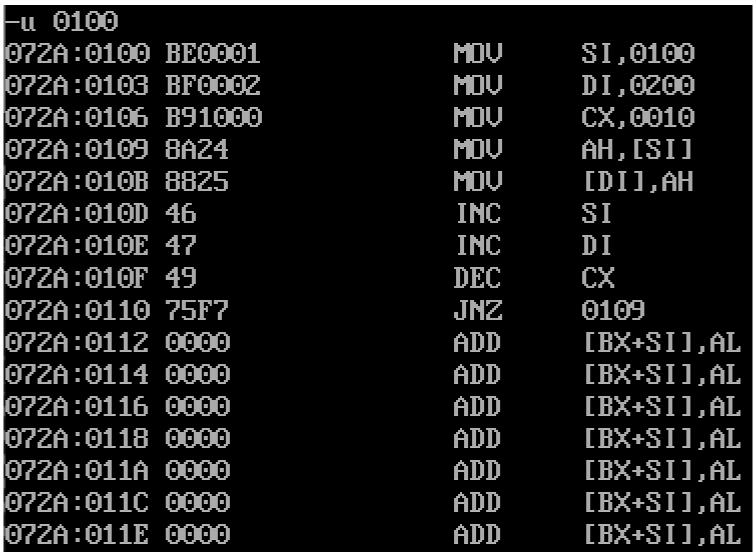
Ans:

- A.



```
-a 0100
072A:0100 MOV SI,0100
072A:0103 MOV DI,0200
072A:0106 MOV CX,010
072A:0109 MOV AH,[SI]
072A:010B MOV [DI],AH
072A:010D INC SI
072A:010E INC DI
072A:010F DEC CX
072A:0110 JNZ 0109
072A:0112
```

- B.



```
-u 0100
072A:0100 BE0001            MOV     SI,0100
072A:0103 BF0002            MOV     DI,0200
072A:0106 B91000            MOV     CX,0010
072A:0109 8A24              MOV     AH,[SI]
072A:010B 8825              MOV     [DI],AH
072A:010D 46                INC     SI
072A:010E 47                INC     DI
072A:010F 49                DEC     CX
072A:0110 75F7              JNZ     0109
072A:0112 0000              ADD     [BX+SI],AL
072A:0114 0000              ADD     [BX+SI],AL
072A:0116 0000              ADD     [BX+SI],AL
072A:0118 0000              ADD     [BX+SI],AL
072A:011A 0000              ADD     [BX+SI],AL
072A:011C 0000              ADD     [BX+SI],AL
072A:011E 0000              ADD     [BX+SI],AL
```

- C. Program takes 101E(11E-100) bytes memory take up.
- D. Machine code for DEC CX is 072A:010F.
- E. the address offset for the label BACK is 0110

13.What are the difference between T, G and P debug commands?

Ans:

- **T (Trace command):**

  while Go executes a whole block of code at one time, the Trace command executes instructions one at a time, displaying the registers after each instruction.

- **G (Go command):**

  The Go command is used to start program execution. It can be used to start the execution at any point in the program, and optionally stop at any of points (breakpoints) in the program. If no breakpoints are set, program execution continues until termination.

- **P (Proceed command):**

The P command executes one or more instructions or subroutines. Whereas the T command trace into subroutine calls, the P command simply executes subroutines.

Ex:

P 100 5	Execute 5 instructions starting at CS:0100
P 3	Execute the next 3 instructions

Lab 2

1. **Copy the data from source to destination variable using different data types and different addressing modes.**

Ans:

```
data segment
num db 10;
data ends
code segment
```

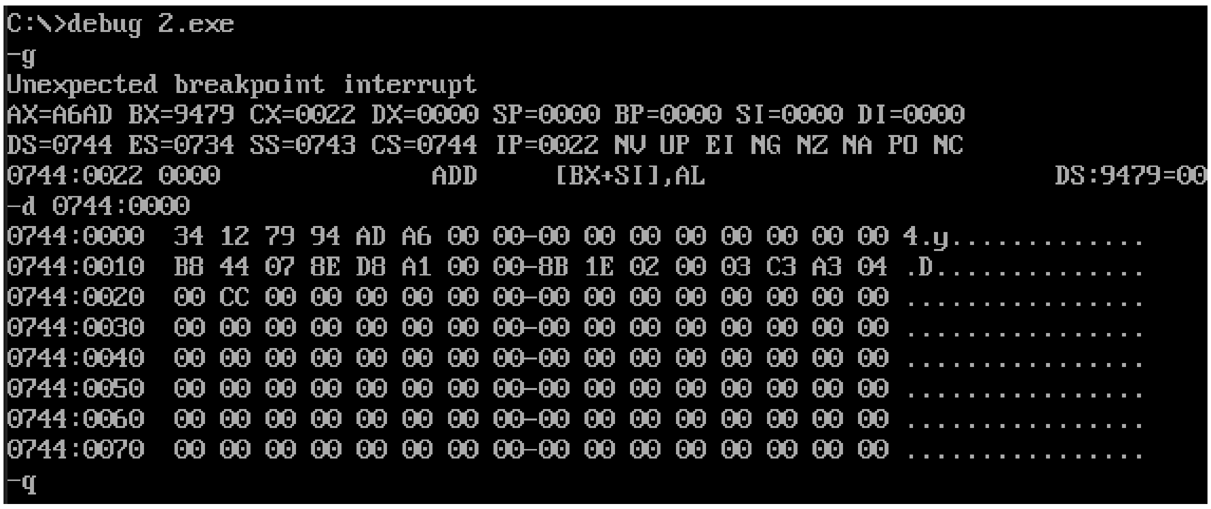
```
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov dl,num
        mov bX,5000H ;immediate addressing mode
        mov dX,bX    ;register addressing mode
        mov dx,[5000H];direct addressing mode
        mov dx,[bx];register indirect addressing mode
        mov dx,[bx+02];based addressing mode
        mov bx,[si+10]; index addressing mode
        mov ax,[si+bx]; based index addressing mode
                mov ax,[bx+di+07]
;based index with displacement addressing mode
        code ends
end
```

2. Add 2 16-bit numbers. The 16 bit numbers are stored into the data segment.

Ans:

```
data segment
        a dw 1234h
        b dw 9479h
        c dw ?
data ends
code segment
```

```
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov ax,a
        mov bx,b
        add ax,bx
        mov c,ax
        int 3
    code ends
    end
```



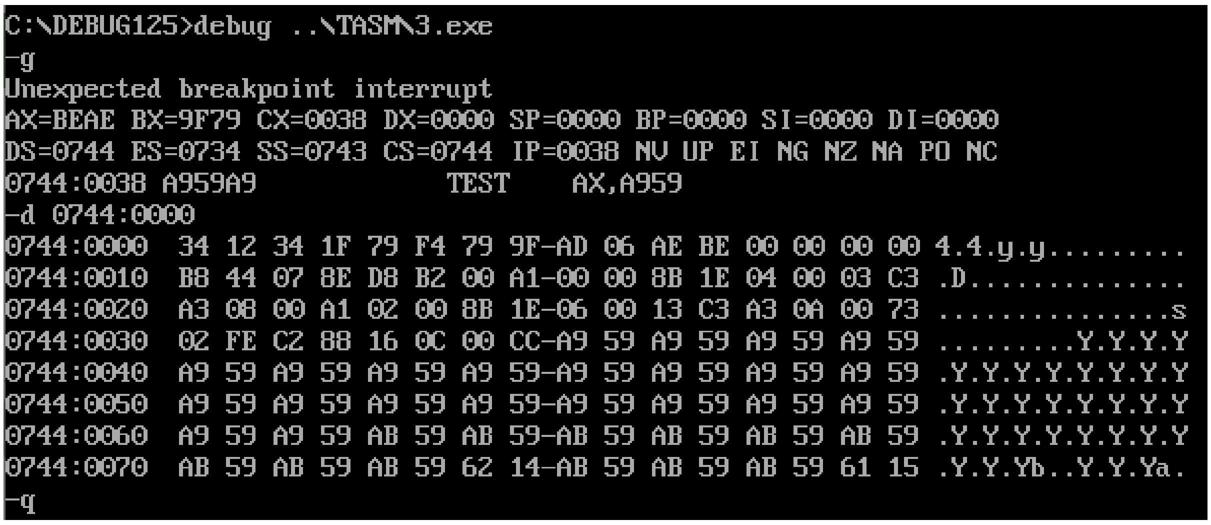
```
C:\>debug 2.exe
-g
Unexpected breakpoint interrupt
AX=A6AD BX=9479 CX=0022 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0022 NV UP EI NG NZ NA PO NC
0744:0022 0000          ADD     [BX+SI],AL                    DS:9479=00
-d 0744:0000
0744:0000  34 12 79 94 AD A6 00 00-00 00 00 00 00 00 00 00 4.y.............
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 03 C3 A3 04 .D..............
0744:0020  00 CC 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0744:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0744:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0744:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0744:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0744:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
-q
```

3. Add 2 32-bit numbers stored in the data segment.

Ans:

```
    data segment
        a dd 1F341234h
        b dd 9F79F479h
        c dw ?
```

```
data ends
code segment
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov dl,00H
        mov ax,word ptr a ;lsb of a
        mov bx,word ptr b ;lsb of b
        add ax,bx;
        mov word ptr c,ax ;lsb of answer
        mov ax,word ptr a+2 ;msb of a
        mov bx,word ptr b+2 ;msb of b
        adc ax,bx;
        mov word ptr c+2,ax;
        jnc jump
        inc dl
        jump:mov byte ptr c+4,dl
        int 3
code ends
end
```



```
C:\DEBUG125>debug ..\TASM\3.exe
-g
Unexpected breakpoint interrupt
AX=BEAE BX=9F79 CX=0038 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0038 NV UP EI NG NZ NA PO NC
0744:0038 A959A9            TEST    AX,A959
-d 0744:0000
0744:0000  34 12 34 1F 79 F4 79 9F-AD 06 AE BE 00 00 00 00 4.4.y.y.........
0744:0010  B8 44 07 8E D8 B2 00 A1-00 00 8B 1E 04 00 03 C3 .D..............
0744:0020  A3 08 00 A1 02 00 8B 1E-06 00 13 C3 A3 0A 00 73 ...............s
0744:0030  02 FE C2 88 16 0C 00 CC-A9 59 A9 59 A9 59 A9 59 .........Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
-q
```

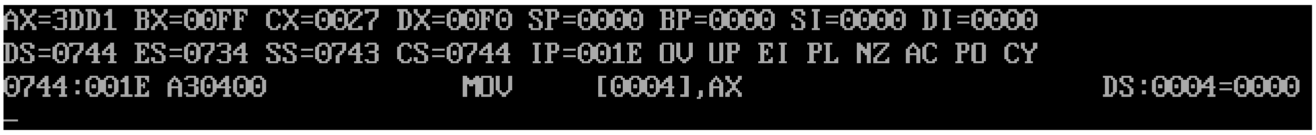
Lab 3

1. Program to Multiply two unsigned 16 bit numbers.

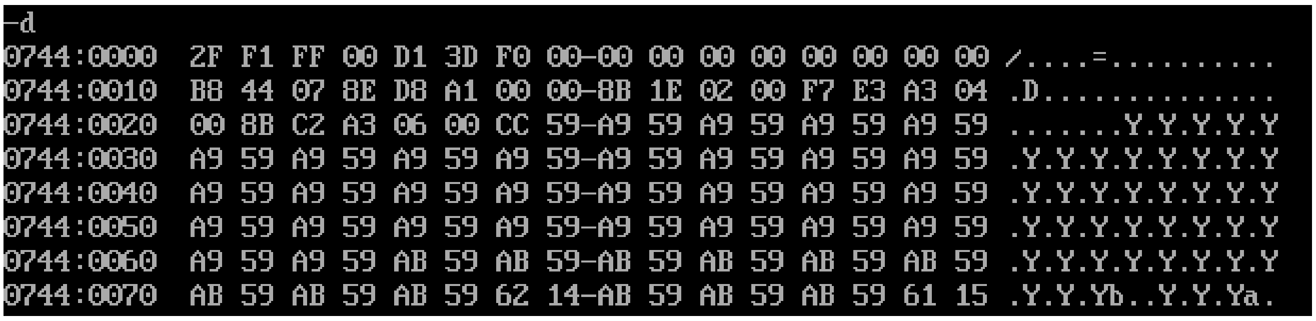
Ans:

```
data segment
        num1 dw 0F12FH
        num2 dw 000FFH
        answer dd ?
data ends
code segment
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov ax,num1
        mov bx,num2
        mul bx
        mov word ptr answer,ax

        mov ax,dx

        mov word ptr answer+2,ax

        int 03

code ends

end
```

ScreenShots:



```
AX=3DD1  BX=00FF  CX=0027  DX=00F0  SP=0000  BP=0000  SI=0000  DI=0000
DS=0744  ES=0734  SS=0743  CS=0744  IP=001E   OV UP EI PL NZ AC PO CY
0744:001E A30400        MOV     [0004],AX                       DS:0004=0000
-
```



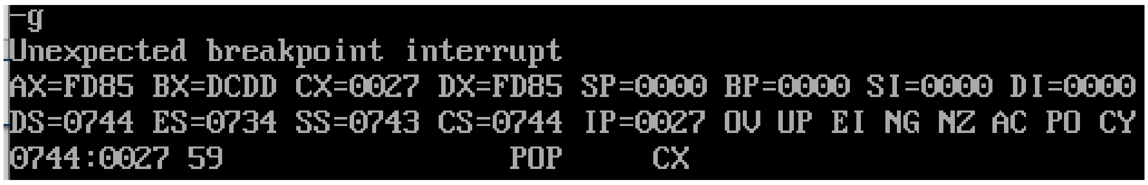
```
-d
0744:0000  2F F1 FF 00 D1 3D F0 00-00 00 00 00 00 00 00 00   /....=..........
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .D..............
0744:0020  00 8B C2 A3 06 00 CC 59-A9 59 A9 59 A9 59 A9 59   .......Y.Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59   .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59   .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59   .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59   .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15   .Y.Y.Yb..Y.Y.Ya.
```

2. Multiply Two 16 bit Sign number.

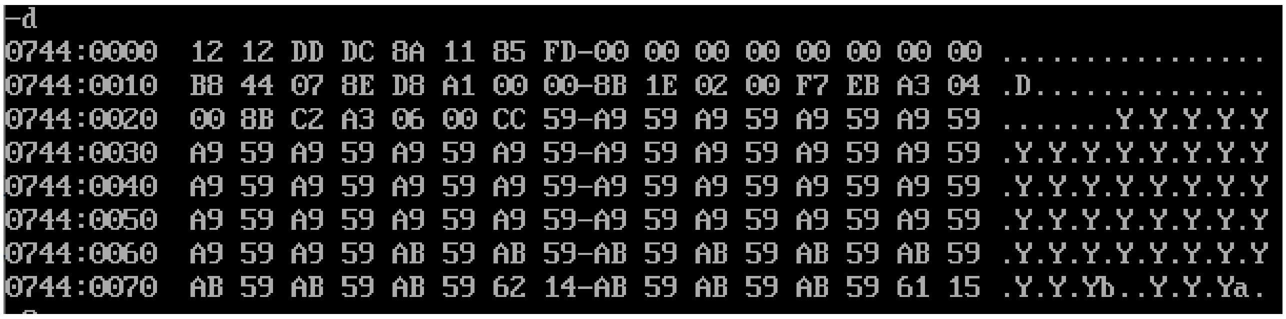
Ans:

```
data segment

    num1 dw 1212H

    num2 dw -2323H

    answer dd ?

data ends

code segment

    assume cs:code,ds:data

    mov ax,data
    mov ds,ax
    mov ax,num1
    mov bx,num2
    imul bx
    mov word ptr answer,ax
    mov ax,dx
    mov word ptr answer+2,ax
    int 03
code ends
end
```

Screenshots:



```
-g
Unexpected breakpoint interrupt
AX=FD85 BX=DCDD CX=0027 DX=FD85 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0027 OV UP EI NG NZ AC PO CY
0744:0027 59                POP     CX
```

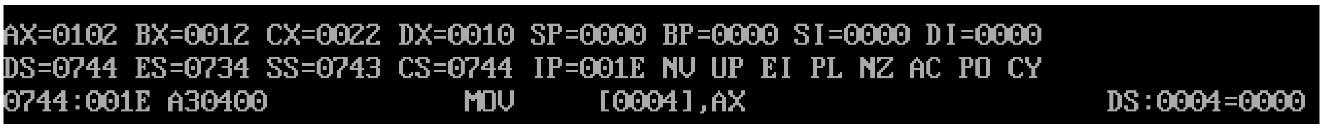


```
-d
0744:0000  12 12 DD DC 8A 11 85 FD-00 00 00 00 00 00 00 00 ................
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 F7 EB A3 04 .D..............
0744:0020  00 8B C2 A3 06 00 CC 59-A9 59 A9 59 A9 59 A9 59 .......Y.Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
```

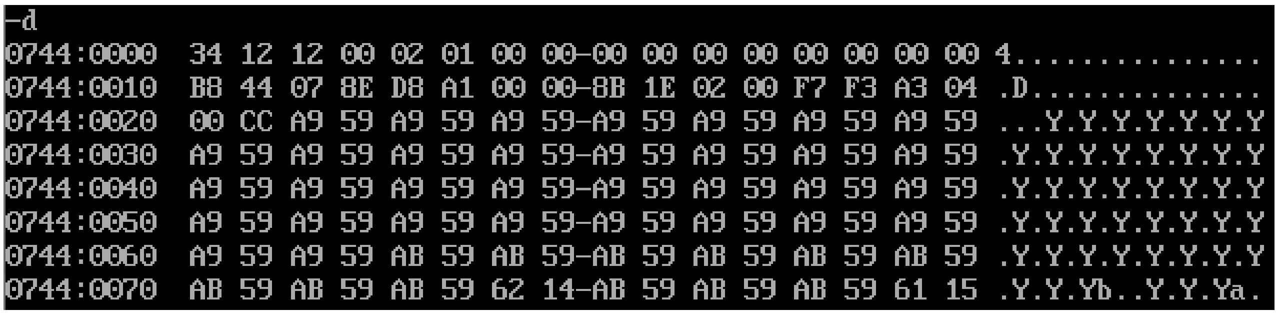
3. program to divide 16 bit sign/unsign number.

Ans:

```
data segment
    num1 dw 1234H
    num2 dw 0012H

    answer dw ?

data ends

code segment

    assume ds:data,cs:code

    mov ax,data

    mov ds,ax

    mov ax,num1

    mov bx,num2

    div bx

    mov answer,ax

    int 03

code ends

end
```



```
AX=0102 BX=0012 CX=0022 DX=0010 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=001E NV UP EI PL NZ AC PO CY
0744:001E A30400            MOV     [0004],AX                    DS:0004=0000
```

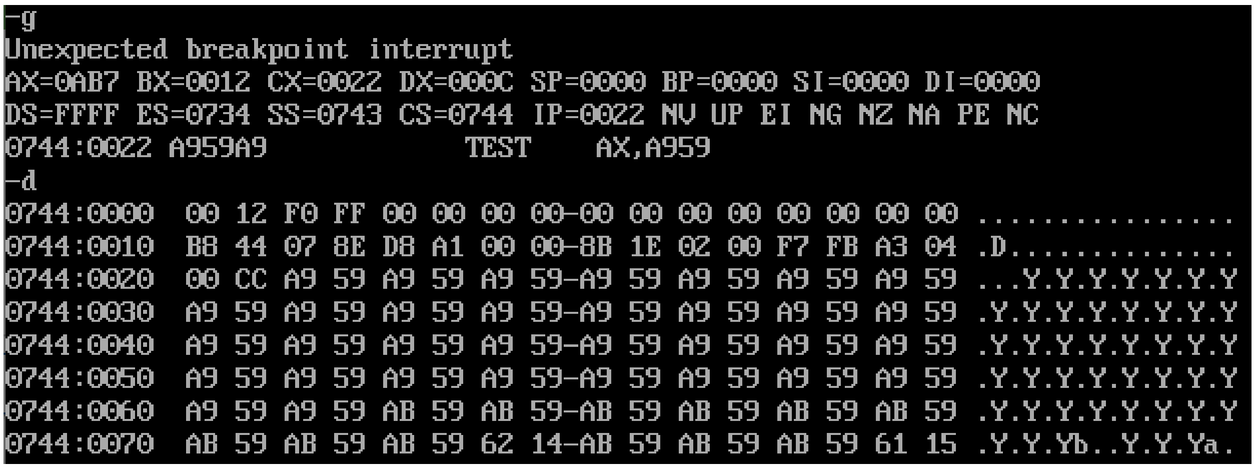


```
-d
0744:0000  34 12 12 00 02 01 00 00-00 00 00 00 00 00 00 00 4...............
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 F7 F3 A3 04 .D..............
0744:0020  00 CC A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 ...Y.Y.Y.Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
```

```
data segment

    num1 dw 1200H

    num2 dw -10H

    answer dw ?
```

```
data ends
code segment
    assume ds:data,cs:code
    mov ax,data
    mov ds,ax
    mov ax,num1
    mov bx,num2
    idiv bx
    mov answer,ax
    int 03
code ends
end
```

Screenshots:



```
-g
Unexpected breakpoint interrupt
AX=0AB7 BX=0012 CX=0022 DX=000C SP=0000 BP=0000 SI=0000 DI=0000
DS=FFFF ES=0734 SS=0743 CS=0744 IP=0022 NV UP EI NG NZ NA PE NC
0744:0022 A959A9            TEST    AX,A959
-d
0744:0000  00 12 F0 FF 00 00 00 00-00 00 00 00 00 00 00 00 ................
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 F7 FB A3 04 .D..............
0744:0020  00 CC A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 ...Y.Y.Y.Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
```

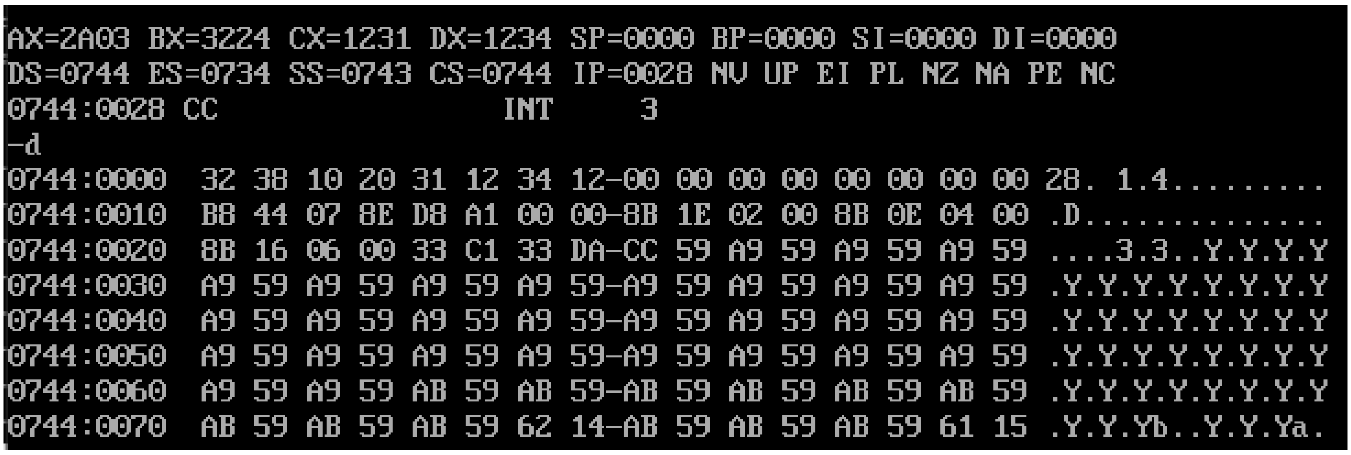
4. Find the Checksum and verify validity of a 32 bit data.

Ans:

```
data segment
    num1 dd 20103832H
    num2 dd 12341231H
```

```
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov ax,word ptr num1
    mov bx,word ptr num1+2
    mov cx,word ptr num2
    mov dx,word ptr num2+2
    xor ax,cx
    xor bx,dx
    int 03H
code ends
end
```

ScreenShots:



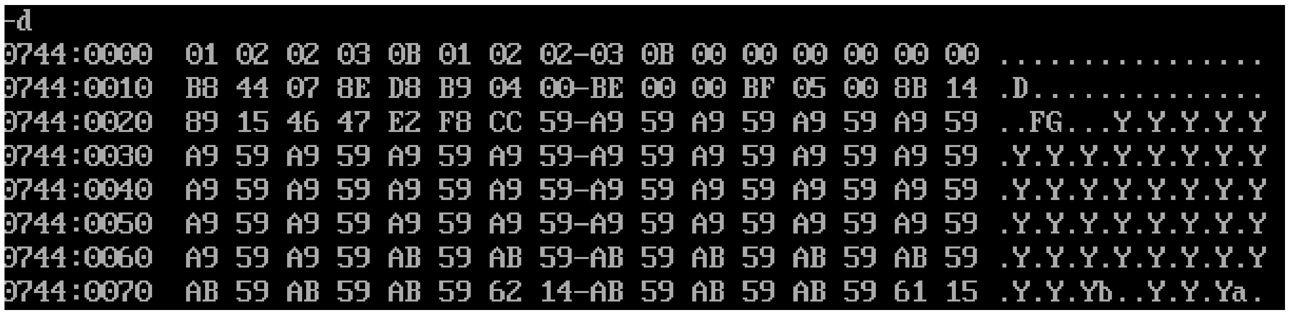
```
AX=2A03 BX=3224 CX=1231 DX=1234 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0028 NV UP EI PL NZ NA PE NC
0744:0028 CC              INT     3
-d
0744:0000  32 38 10 20 31 12 34 12-00 00 00 00 00 00 00 00 28. 1.4.........
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 8B 0E 04 00 .D..............
0744:0020  8B 16 06 00 33 C1 33 DA-CC 59 A9 59 A9 59 A9 59 ....3.3..Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
```

5. Program to copy an array of bytes/words from the variable “SOURCE” to variable “DEST” which are defined in data segment.

Ans:

```
data segment
    src db 1,2,2,3,11
    dest db 5 dup(?)
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov cx,4
    mov si,offset src
    mov di,offset dest
    l1:mov dx,[si]
            mov [di],dx
            inc si
            inc di
            loop l1
    int 03H
code ends
end
```

ScreenShot:



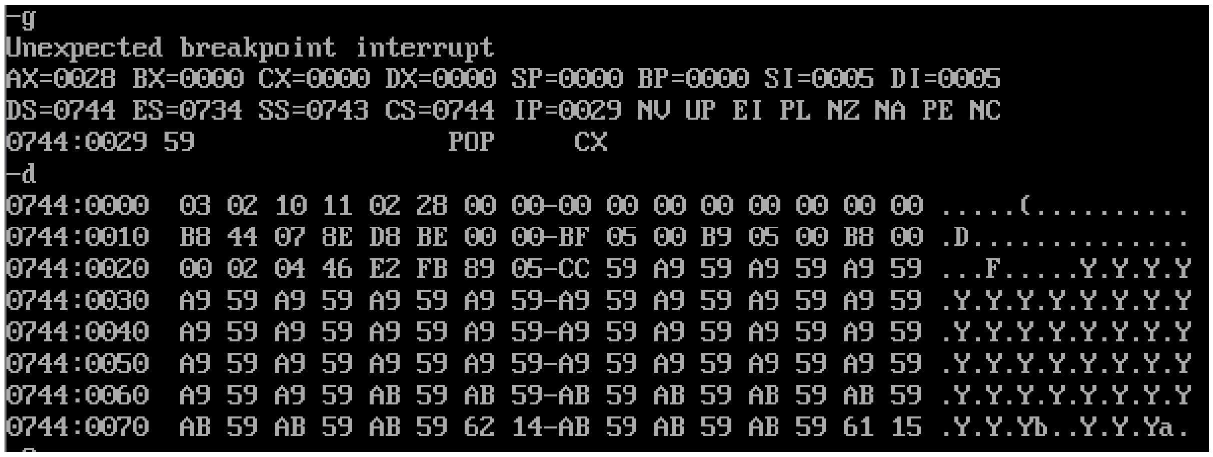
```
-d
0744:0000  01 02 02 03 0B 01 02 02-03 0B 00 00 00 00 00 00  ................
0744:0010  B8 44 07 8E D8 B9 04 00-BE 00 00 BF 05 00 8B 14  .D..............
0744:0020  89 15 46 47 E2 F8 CC 59-A9 59 A9 59 A9 59 A9 59  ..FG...Y.Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15  .Y.Y.Yb..Y.Y.Ya.
```

6.find the sum of array

Ans:

```
data segment
        arr db 3H,2H,10H,11H,2H
        ans db ?
data ends
code segment
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov si,offset arr
        mov di,offset ans
        mov cx,5
        mov ax,0000H
        l1:add al,[si]
                inc si
                loop l1
        mov [di],ax
        int 03h
code ends
end
```

Screenshot:



```
-g
Unexpected breakpoint interrupt
AX=0028 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0005 DI=0005
DS=0744 ES=0734 SS=0743 CS=0744 IP=0029 NV UP EI PL NZ NA PE NC
0744:0029 59                POP     CX
-d
0744:0000  03 02 10 11 02 28 00 00-00 00 00 00 00 00 00 00  .....(..........
0744:0010  B8 44 07 8E D8 BE 00 00-BF 05 00 B9 05 00 B8 00  .D..............
0744:0020  00 02 04 46 E2 FB 89 05-CC 59 A9 59 A9 59 A9 59  ...F.....Y.Y.Y.Y
0744:0030  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15  .Y.Y.Yb..Y.Y.Ya.
```

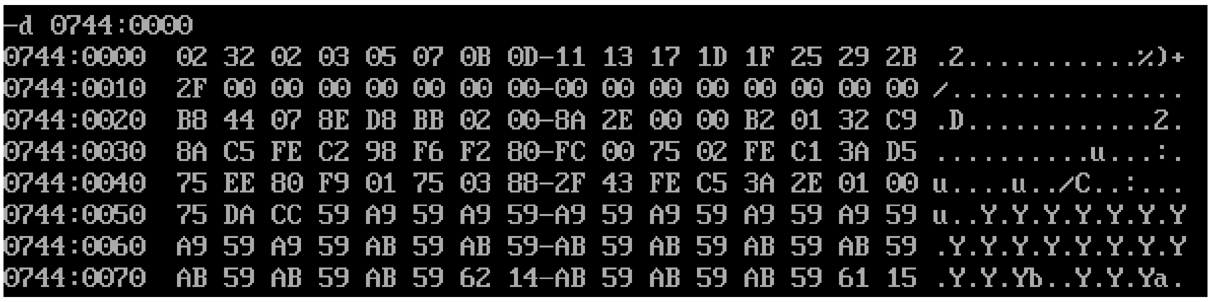
Lab 4

1.Write a program to find all the prime numbers between 1 to 50.

Code:

```
data segment
        start db 02H
        ending db 32H
        answer db 18 dup(0)
data ends
code segment
        assume cs:code, ds:data
        mov ax,data
        mov ds,ax
        mov bx,offset answer
        mov ch,start
NEXT:
         mov dl,01H
        xor cl,cl
LOOP2:
        mov al,ch
        inc dl
        cbw
        div dl
        cmp ah,00
        JNZ SKIP
        inc cl
SKIP:
        cmp dl,ch
        JNZ LOOP2
        cmp cl,1H
        JNZ SKIP1
        mov byte ptr[bx],ch
        inc bx
SKIP1:
        inc ch;
        cmp ch,ending
        JNZ NEXT
        int 03
code ends
end
```

Screenshot:



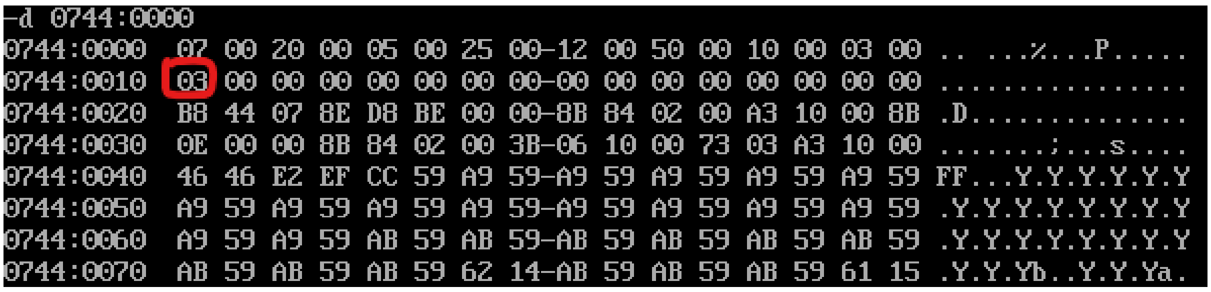
```
-d 0744:0000
0744:0000  02 32 02 03 05 07 0B 0D-11 13 17 1D 1F 25 29 2B .2...........%)+
0744:0010  2F 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 /...............
0744:0020  B8 44 07 8E D8 BB 02 00-8A 2E 00 00 B2 01 32 C9 .D............2.
0744:0030  8A C5 FE C2 98 F6 F2 80-FC 00 75 02 FE C1 3A D5 ..........u...:.
0744:0040  75 EE 80 F9 01 75 03 88-2F 43 FE C5 3A 2E 01 00 u....u../C..:...
0744:0050  75 DA CC 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 u..Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
```

2. Write a program to find the smallest number in an array of words and store it in the variable named small of the data segment.

Code:

```
data segment
        arrsize dw 7
        array dw 20h,5h,25h,12h,50h,10h,3h
        smallNum dw ?
data ends
code segment
        assume cs: code, ds: data
        mov ax, data
        mov ds, ax
        mov si,0
        mov ax, array[si]
        mov smallNum, ax
        mov cx, arrsize
next:
        mov ax, array[si]
        cmp ax, smallNum
        jnc skip
        mov smallNum, ax
skip:
        inc si
        inc si
        loop next
        int 03
code ends
end
```

Screenshot:



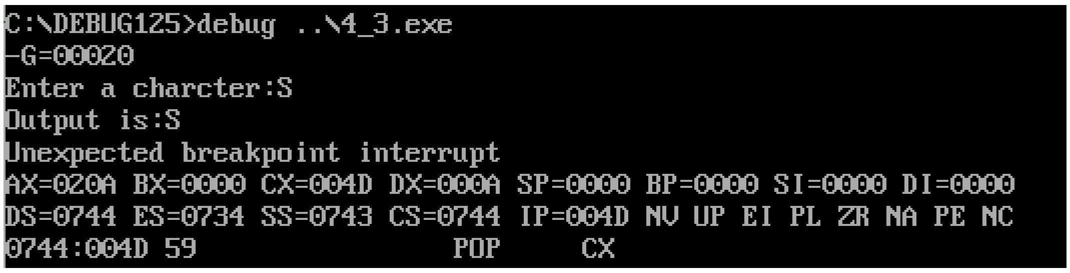
```
-d 0744:0000
0744:0000  07 00 20 00 05 00 25 00-12 00 50 00 10 00 03 00  .. ...%...P.....
0744:0010  03 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0744:0020  B8 44 07 8E D8 BE 00 00-8B 84 02 00 A3 10 00 8B  .D..............
0744:0030  0E 00 00 8B 84 02 00 3B-06 10 00 73 03 A3 10 00  .......;...s....
0744:0040  46 46 E2 EF CC 59 A9 59-A9 59 A9 59 A9 59 A9 59  FF...Y.Y.Y.Y.Y.Y
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15  .Y.Y.Yb..Y.Y.Ya.
```

3. Write a program to read a character and Display the character on the screen.

Code:

```
data segment
        msg1 db "Enter a charcter:$"
        msg2 db "Output is:$"
        char db ?;
data ends
code segment
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov dx,offset msg1
        mov ah,09 ;print msg1 string
        int 21h
        mov ah,1 ;take i/p from user and store in al
        int 21h
        mov char,al
        mov dl,10
        mov ah,2 ;print new line character
        int 21h
        mov dx,offset msg2
        mov ah,09
        int 21h
        mov dl,char
        mov ah,2 ;print character which store in char
        int 21h
        mov dl,0AH
        int 21h
        int 03
code ends
end
```

Screenshot:



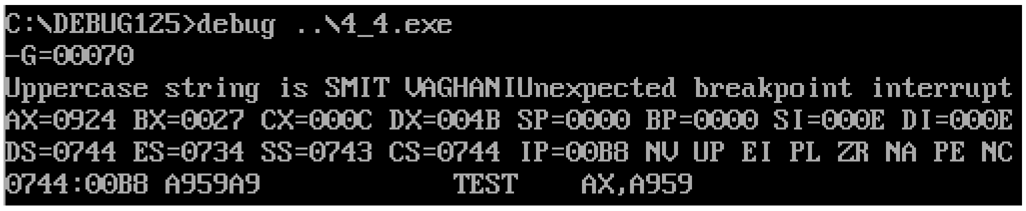
```
C:\DEBUG125>debug ..\4_3.exe
-G=00020
Enter a charcter:S
Output is:S
Unexpected breakpoint interrupt
AX=020A BX=0000 CX=004D DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=004D NV UP EI PL ZR NA PE NC
0744:004D 59                POP     CX
```

4. Write a program to read a string of lower case letters and convert to upper case and display the string on the console.

Code:

```
data segment
        mes1 db "Enter a lowercase string $"
        lower db 25,25 dup(0)
        mes2 db "Uppercase string is $"
        upper db 25,25 dup('$')
data ends
code segment
        assume ds:data,cs:code
        mov ax,data
        mov ds,ax
        mov ah,09
        mov dx,offset mes1
        int 21h
        mov ah,0ah
        mov dx,offset lower
        int 21h
        mov ah,09
        mov dx,offset mes2
        int 21h
        mov bx,offset lower
        inc bx
        mov ch,00
        mov cl,[bx]
        add bx,cx
        mov byte ptr[bx+1],'$'
        mov si,00
        mov di,00
label1:
        mov al,lower[si]
        cmp al,'$'
        jz label2
        sub al,20h
        mov upper[di],al
        inc si
        inc di
        jmp label1
label2:
        mov ah,09
        mov dx,offset upper+2
        int 21h
        int 03
code ends
end
```

Screenshot:



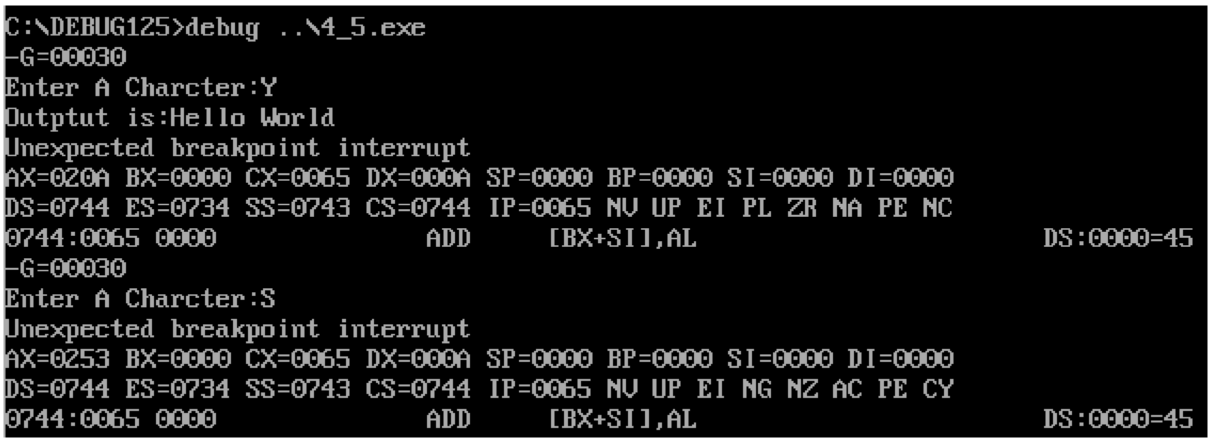
```
C:\DEBUG125>debug ..\4_4.exe
-G=00070
Uppercase string is SMIT VAGHANIUnexpected breakpoint interrupt
AX=0924 BX=0027 CX=000C DX=004B SP=0000 BP=0000 SI=000E DI=000E
DS=0744 ES=0734 SS=0743 CS=0744 IP=00B8 NV UP EI PL ZR NA PE NC
0744:00B8 A959A9            TEST    AX,A959
```

5. Write a program to display the string “Hello world” when the character ‘Y’ is pressed.

Code:

```
data segment
        msg1 db "Enter A Charcter:$"
        msg2 db "Outptut is:$"
        output db "Hello World$"
        char db ?
data ends
code segment
        assume cs:code,ds:data
        mov ax,data
        mov ds,ax
        mov dx,offset msg1
        mov ah,09
        int 21h
        mov ah,01
        int 21h
        mov char,al
        mov dl,10
        mov ah,02
        int 21h
        mov al,char
        cmp al,'Y'
        jnz lable1
        mov dx,offset msg2
        mov ah,09
        int 21h
        mov dx,offset output
        mov ah,09
        int 21h
        mov dl,10
        mov ah,02
        int 21h
lable1: int 03
code ends
end
```

Screenshot:



```
C:\DEBUG125>debug ..\4_5.exe
-G=00030
Enter A Charcter:Y
Outptut is:Hello World
Unexpected breakpoint interrupt
AX=020A BX=0000 CX=0065 DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0065 NV UP EI PL ZR NA PE NC
0744:0065 0000          ADD     [BX+SI],AL                 DS:0000=45
-G=00030
Enter A Charcter:S
Unexpected breakpoint interrupt
AX=0253 BX=0000 CX=0065 DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0065 NV UP EI NG NZ AC PE CY
0744:0065 0000          ADD     [BX+SI],AL                 DS:0000=45
```

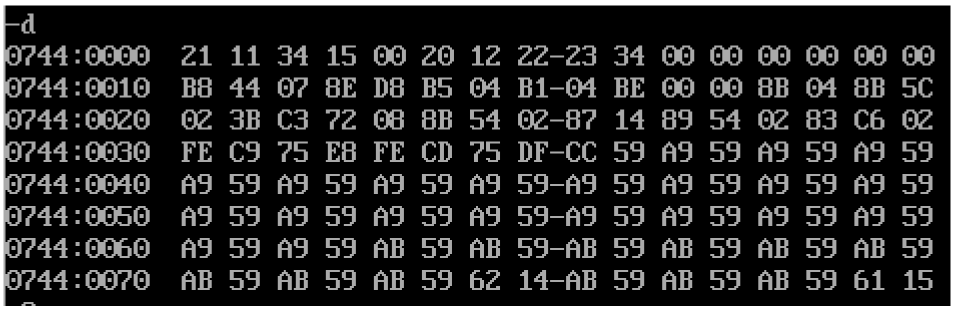
Lab 5

1. **Write a program to sort an array of words in the data segment.**

Code:

```
data segment
    arr dw 2000H,1121H,2212H,3423H,1534H
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov ch,04h;for outer loop
    l:
            mov cl,04h;for inner loop
            mov si,offset arr
    l1:
            mov ax,[si]
            mov bx,[si+2]
            cmp ax,bx
            jc skip;if ax<bx then skip
            mov dx,[si+2]
            xchg [si],dx
            mov [si+2],dx
    skip:
            add si,2
            dec cl

            jnz l1

            dec ch

            jnz l

            int 03

code ends

end
```

Screenshot:



```
-d
0744:0000  21 11 34 15 00 20 12 22-23 34 00 00 00 00 00 00
0744:0010  B8 44 07 8E D8 B5 04 B1-04 BE 00 00 8B 04 8B 5C
0744:0020  02 3B C3 72 08 8B 54 02-87 14 89 54 02 83 C6 02
0744:0030  FE C9 75 E8 FE CD 75 DF-CC 59 A9 59 A9 59 A9 59
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15
```

2. **Consider an array(array1) of 20 random numbers ranging between 1 to 4. Write a program to Count the number of 1’s, 2’s, 3’s and 4’s in the array1 and store the result of the count in one more array(array 2) of size 4 elements. At first location of array2 it has to store the count of 1’s, at second location it has to store count of 2’s and so on...**

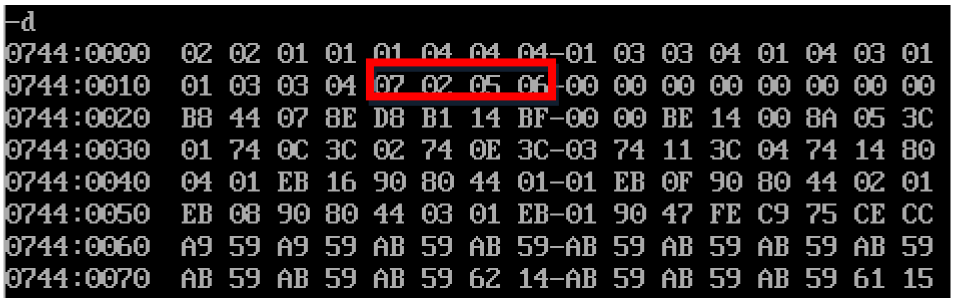
Code:

```
data segment

    arr db 2,2,1,1,1,4,4,4,1,3,3,4,1,4,3,1,1,3,3,4

    count db 4 dup(0)

data ends

code segment

    assume cs:code,ds:data
```

```
mov ax,data
mov ds,ax
mov cl,20
mov di,offset arr
mov si,offset count
l:mov al,[di]
cmp al,1
jz skip
cmp al,2
jz skip1
cmp al,3
jz skip2
cmp al,4
jz skip3
skip:
        add byte ptr[si],1
        jmp l1
skip1:add byte ptr[si+1],1
        jmp l1
skip2:add byte ptr[si+2],1
        jmp l1
skip3:add byte ptr[si+3],1
        jmp l1
l1:inc di
```

```
            dec cl
            jnz l
    int 03
code ends
end
```

Screenshot:



```
-d
0744:0000  02 02 01 01 01 04 04 04-01 03 03 04 01 04 03 01
0744:0010  01 03 03 04 07 02 05 06-00 00 00 00 00 00 00 00
0744:0020  B8 44 07 8E D8 B1 14 BF-00 00 BE 14 00 8A 05 3C
0744:0030  01 74 0C 3C 02 74 0E 3C-03 74 11 3C 04 74 14 80
0744:0040  04 01 EB 16 90 80 44 01-01 EB 0F 90 80 44 02 01
0744:0050  EB 08 90 80 44 03 01 EB-01 90 47 FE C9 75 CE CC
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15
```

3. **Write a program to create a file and write 10 bytes of data into the file. Create one more file and make a copy of the first file.(i.e Read from the first file and write into the second file.)**

Code:

```
data segment
    file db "file.txt",0
    handler dw ?
    string db "helloWorld"
    file1 db "file1.txt",0
    handler1 dw ?
    buffer db ?
data ends
code segment
```

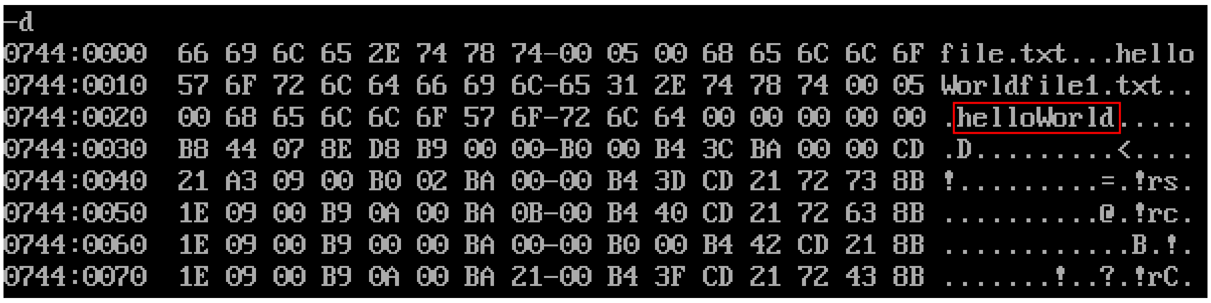
```
assume cs:code,ds:data

mov ax,data

mov ds,ax


;file creation

mov cx,0000H ;no attribute

mov al,00H

mov ah,3CH ;file creation

mov dx,offset file

int 21h

mov handler,ax


;file open

mov al,2 ;open file in read/write

mov dx,offset file

mov ah,3DH ;file open

int 21H

jc skip


;file write

mov bx,handler

mov cx,10 ;10 byte write

mov dx,offset string ;data write in file

mov ah,40H ;write in file
```

```
int 21H
jc skip

;seek to stating of file
mov bx,handler
mov cx,0
mov dx,0
mov al,0 ;stating of file
mov ah,42H;seek
int 21H

;file read
mov bx,handler
mov cx,10 ;10 bytes read
mov dx,offset buffer
mov ah,3FH
int 21H
jc skip

;close file
mov bx,handler
mov ah,3EH
int 21H
jc skip
```

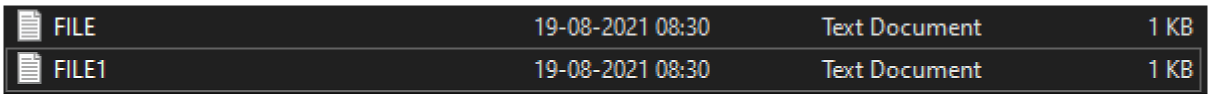
```
;file1 creation
mov cx,0000H ;no attribute
mov al,00H
mov ah,3CH ;file creation
mov dx,offset file1
int 21h
jc skip
mov handler1,ax

;file1 open
mov al,2 ;open file in read/write
mov dx,offset file1
mov ah,3DH ;file open
int 21H
jc skip


;file1 write
mov bx,handler1
mov cx,10
mov dx,offset buffer
mov ah,40H
int 21H
```

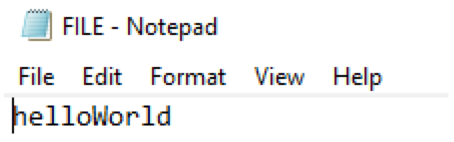
```
    jc skip

    ;close file1
    mov bx,handler1
    mov ah,3EH
    int 21H
    jc skip
    jmp k

    ;if error occurs
    skip:mov ah,4CH
    int 21H

    ;terminate
    k:int 03H
code ends
end
```

Screenshot:

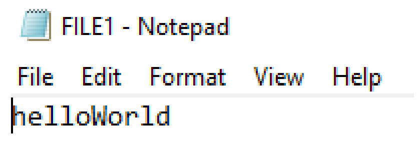


```
-d
0744:0000  66 69 6C 65 2E 74 78 74-00 05 00 68 65 6C 6C 6F file.txt...hello
0744:0010  57 6F 72 6C 64 66 69 6C-65 31 2E 74 78 74 00 05 Worldfile1.txt..
0744:0020  00 68 65 6C 6C 6F 57 6F-72 6C 64 00 00 00 00 00 .helloWorld.....
0744:0030  B8 44 07 8E D8 B9 00 00-B0 00 B4 3C BA 00 00 CD .D.........<....
0744:0040  21 A3 09 00 B0 02 BA 00-00 B4 3D CD 21 72 73 8B !.........=.!rs.
0744:0050  1E 09 00 B9 0A 00 BA 0B-00 B4 40 CD 21 72 63 8B ..........@.!rc.
0744:0060  1E 09 00 B9 00 00 BA 00-00 B0 00 B4 42 CD 21 8B ............B.!.
0744:0070  1E 09 00 B9 0A 00 BA 21-00 B4 3F CD 21 72 43 8B .......!..?.!rC.
```


```
FILE     19-08-2021 08:30    Text Document    1 KB
FILE1    19-08-2021 08:30    Text Document    1 KB
```



```
helloWorld
```



```
helloWorld
```

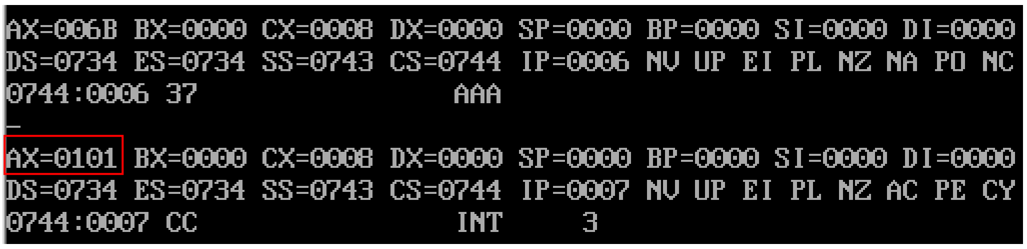
Lab 6

1. **Write the programs to verify the instructions AAA, AAS, AAM, AAD, DAA and DAS instructions.**

Code:

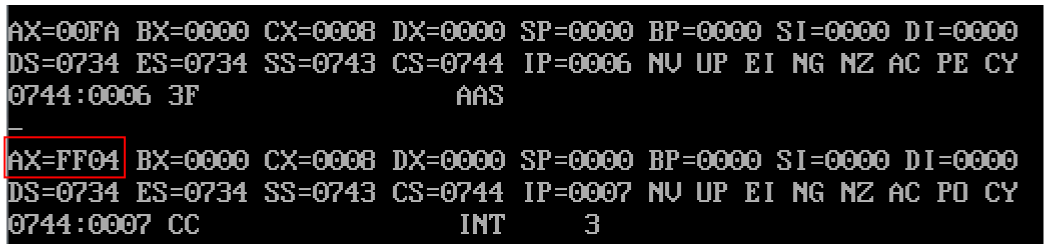
```
code segment
    assume cs:code
    mov ah,0
    mov al,'6'
    add al,'5'
    aaa
    int 03
code ends
end
```

Screenshot:


```
AX=006B BX=0000 CX=0008 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0006 NV UP EI PL NZ NA PO NC
0744:0006 37                AAA
-
AX=0101 BX=0000 CX=0008 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0007 NV UP EI PL NZ AC PE CY
0744:0007 CC                INT     3
```

```
code segment
    assume cs:code
    mov ah,0
    mov al,'3'
    sub al,'9'
    aas
    int 03
code ends
end
```

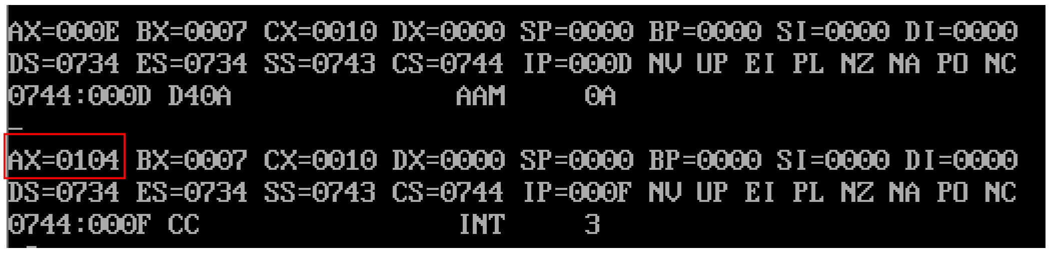
Screenshot:



```
AX=00FA BX=0000 CX=0008 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0006 NV UP EI NG NZ AC PE CY
0744:0006 3F            AAS
-
AX=FF04 BX=0000 CX=0008 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0007 NV UP EI NG NZ AC PO CY
0744:0007 CC            INT     3
```

```
code segment
    assume cs:code
    mov ah,0
    mov al,'2'
    mov bl,'7'
    and al,0fh
    and bl,0fh
    mul bl
    aam
    int 03
code ends
end
```

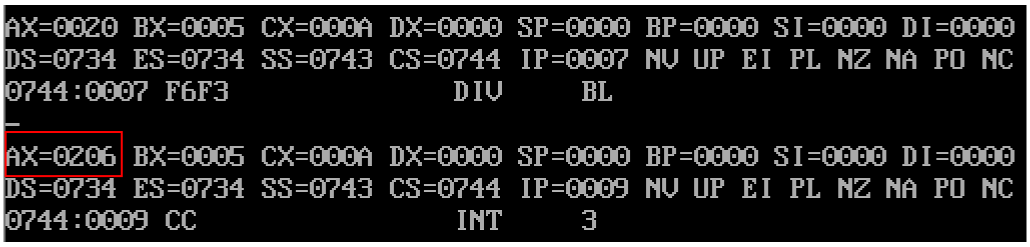
Screenshot:



```
AX=000E BX=0007 CX=0010 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=000D NV UP EI PL NZ NA PO NC
0744:000D D40A          AAM     0A
-
AX=0104 BX=0007 CX=0010 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=000F NV UP EI PL NZ NA PO NC
0744:000F CC            INT     3
```

```
code segment
    assume cs:code
    mov ax,0302h
    aad
    mov bl,5
    div bl
    int 03
code ends
end
```

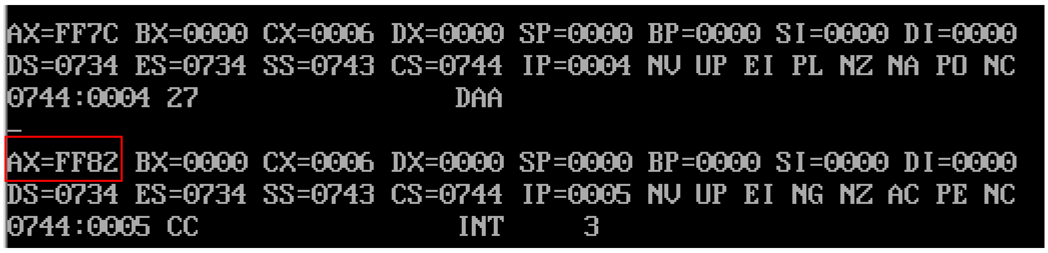
Screenshot:



```
AX=0020 BX=0005 CX=000A DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0007 NV UP EI PL NZ NA PO NC
0744:0007 F6F3              DIV     BL
-
AX=0206 BX=0005 CX=000A DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0009 NV UP EI PL NZ NA PO NC
0744:0009 CC                INT     3
```

```
code segment
    assume cs:code
    mov al,34h
    add al,48h
    daa
    int 03
code ends
end
```

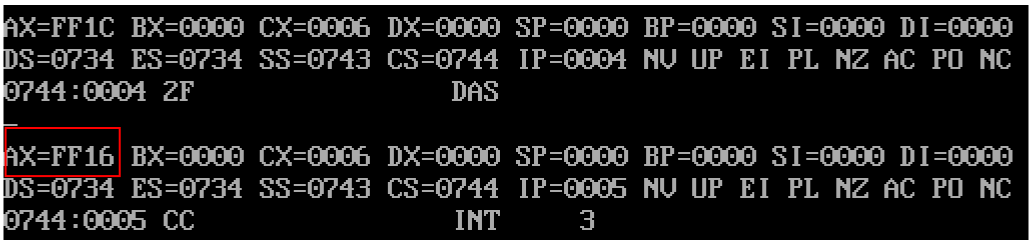
Screenshot:



```
AX=FF7C BX=0000 CX=0006 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0004 NV UP EI PL NZ NA PO NC
0744:0004 27                DAA
-
AX=FF82 BX=0000 CX=0006 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0005 NV UP EI NG NZ AC PE NC
0744:0005 CC                INT     3
```

```
code segment
    assume cs:code
    mov al,34h
    sub al,18h
    das
    int 03
code ends
end
```

Screenshot:



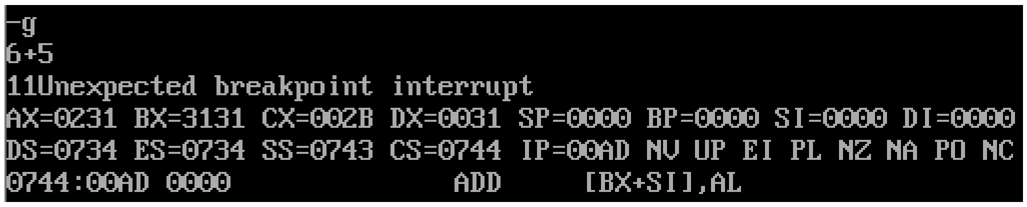
```
AX=FF1C BX=0000 CX=0006 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0004 NV UP EI PL NZ AC PO NC
0744:0004 2F            DAS
-
AX=FF16 BX=0000 CX=0006 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0005 NV UP EI PL NZ AC PO NC
0744:0005 CC            INT     3
```

2. **Implement a calculator for single digit numbers, which takes the input through the keyboard and display the result on the screen.**

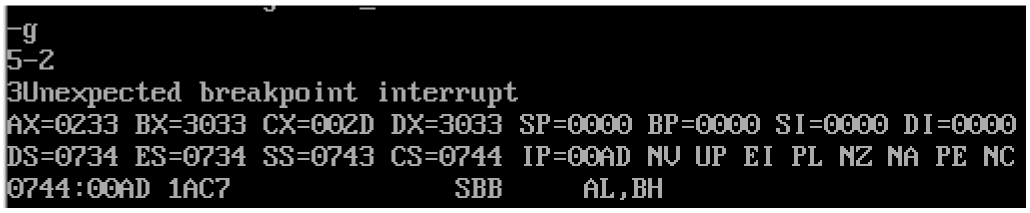
Code:

```
code segment
    assume cs:code

    xor ax,ax
    mov ah,1;first operand
    int 21h
    mov bl,al

    xor ax,ax
    mov ah,1;operation
    int 21h
    mov cl,al

    xor ax,ax
    mov ah,1;second operand
    int 21h
    mov bh,al

    cmp cl,'+'
    jz addition

    cmp cl,'-'
    jz subtraction

    cmp cl,'*'
    jz multiplication

    cmp cl,'/'
    jz division

addition:
    xor ax,ax
    mov al,bh
    add al,bl
    aaa
    or ax,3030h
    mov bx,ax

    mov dl,0ah ;new line
    mov ah,2
    int 21h

    mov dl,bh
    mov ah,2
    int 21h

    mov dl,bl
    mov ah,2
    int 21h
    jmp exit

subtraction:
    xor ax,ax
    mov al,bl
    sub al,bh
    aas
    or ax,3030h
    mov bx,ax

    mov dl,0ah ;new line
    mov ah,2
    int 21h

    mov dx,bx
    mov ah,2
    int 21h
    jmp exit

multiplication:
    xor ax,ax
    and bx,0f0fh
    mov al,bh
    mul bl
    aam
    or ax,3030h
    mov bx,ax

    mov dl,0ah ;new line
    mov ah,2
    int 21h

    mov dl,bh
    mov ah,2
    int 21h

    mov dl,bl
    mov ah,2
    int 21h
    jmp exit

division:
    xor ax,ax
    and bx,0f0fh
    mov al,bl
    div bh
    or ax,3030h
    mov bx,ax

    mov dl,0ah ;new line
    mov ah,2
    int 21h

    mov dx,bx
    mov ah,2
    int 21h
    jmp exit

exit:int 3

code ends
end
```

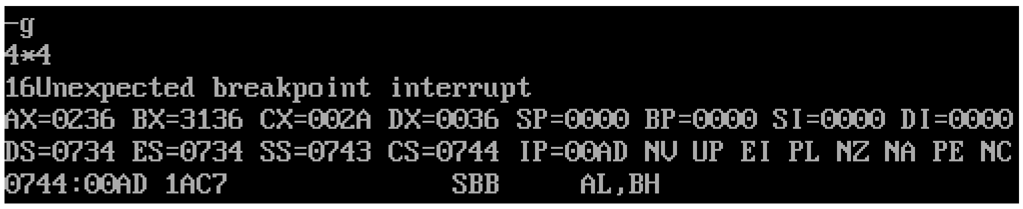
Screenshots:



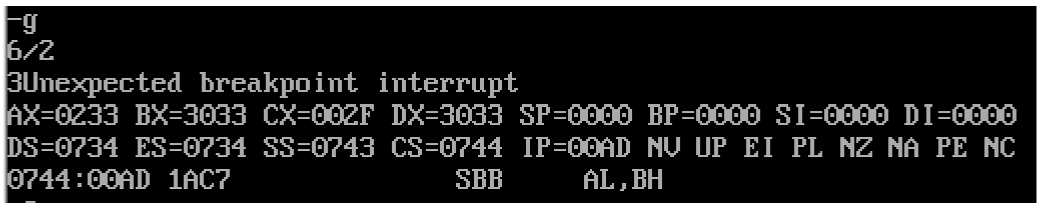
```
-g
6+5
11Unexpected breakpoint interrupt
AX=0231 BX=3131 CX=002B DX=0031 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=00AD NV UP EI PL NZ NA PO NC
0744:00AD 0000           ADD     [BX+SI],AL
```



```
-g
5-2
3Unexpected breakpoint interrupt
AX=0233 BX=3033 CX=002D DX=3033 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=00AD NV UP EI PL NZ NA PE NC
0744:00AD 1AC7           SBB     AL,BH
```



```
-g
4*4
16Unexpected breakpoint interrupt
AX=0236 BX=3136 CX=002A DX=0036 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=00AD NV UP EI PL NZ NA PE NC
0744:00AD 1AC7           SBB     AL,BH
```



```
-g
6/2
3Unexpected breakpoint interrupt
AX=0233 BX=3033 CX=002F DX=3033 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=00AD NV UP EI PL NZ NA PE NC
0744:00AD 1AC7           SBB     AL,BH
```

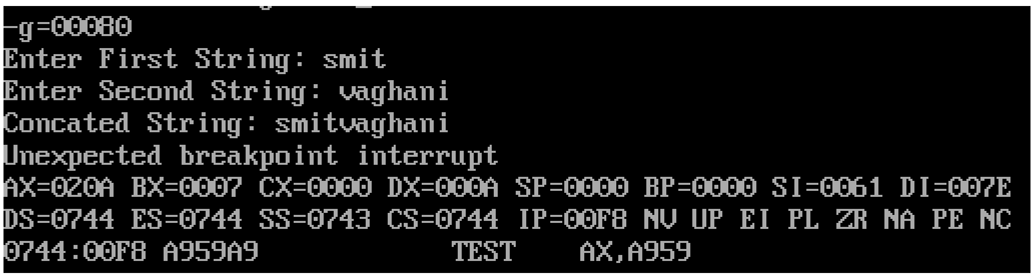
3. **Write a program to concatenate two strings STR1 and STR2 and store the result in the string STR3. Try to input the string through the keyboard.**

Code:

```
data segment
 ins1 db "Enter First String: ",'$'
 ins2 db "Enter Second String: ",'$'
 ins3 db "Concated String: ",'$'

 str1 db 25,?,25 dup(0)
 str2 db 25,?,25 dup(0)
 str3 db ?
data ends
code segment
 assume cs:code,ds:data
 mov ax,data
 mov ds,ax
 mov es,ax

 ;print first string
 mov ah,09h
 lea dx,ins1
 int 21h
 lea dx,str1
 mov ah,0ah
 int 21h

 xor bx,bx ;put terminate character
 mov bl,str1[1]
 mov str1[bx+02h],'$'
 mov dl,0ah ;new line
 mov ah,02h
 int 21h

 ;print second string
 mov ah,09h
 lea dx,ins2
 int 21h

 lea dx,str2
 mov ah,0ah
 int 21h

 xor bx,bx ;put terminate character
 mov bl,str2[1]
 mov str2[bx+02h],'$'

 mov dl,0ah ;new line
 mov ah,02h
 int 21h

 lea dx,ins3
 mov ah,09h
 int 21h

 lea si,str1[2]
 lea di,str3
 xor cx,cx
 mov cl,str1[1]
 cld
```

```
 rep movsb

 lea si,str2[2]
 xor cx,cx
 mov cl,str2[1]
 cld
 rep movsb

 mov byte ptr [di],'$'

 lea dx,str3
 mov ah,09h
 int 21h

 ;newline
 mov dl,0ah
 mov ah,02h
 int 21h

 int 03
code ends
end
```

Screenshot:



```
-g=00080
Enter First String: smit
Enter Second String: vaghani
Concated String: smitvaghani
Unexpected breakpoint interrupt
AX=020A BX=0007 CX=0000 DX=000A SP=0000 BP=0000 SI=0061 DI=007E
DS=0744 ES=0744 SS=0743 CS=0744 IP=00F8 NV UP EI PL ZR NA PE NC
0744:00F8 A959A9            TEST    AX,A959
```

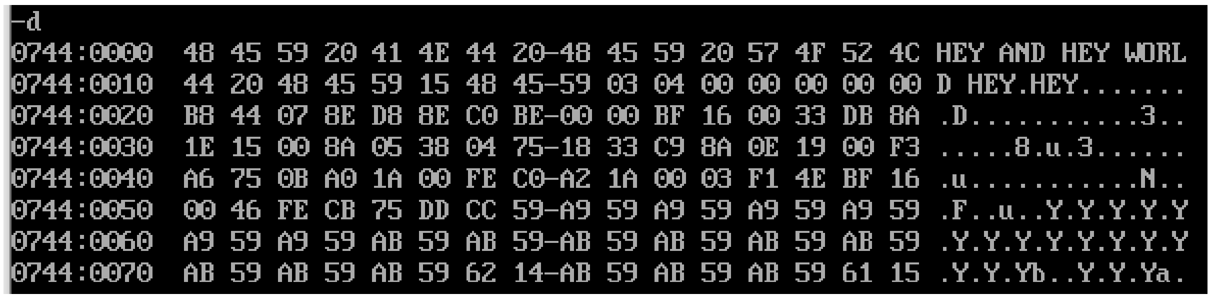
Lab 7

1.Write an assembly language to count the number of occurrence of a substring in a given string.

Code:

```
data segment
    str1 db "HEY AND HEY WORLD HEY"
    len1 db $-str1
    str2 db "HEY"
    len2 db $-str2
    count db 0H
data ends
code segment
    assume cs:code, ds:data
    mov ax,data
    mov ds,ax
    mov es,ax
    lea si,str1
    lea di,str2
    xor bx,bx
    mov bl,len1
    BACK:
    mov al,[di]
    cmp [si],al
    JNZ skip
    xor cx,cx
    mov cl,len2
    REPE cmpsb
    JNZ skip1
    mov al,count
    inc al
    mov count,al
    add si,cx
    dec si
 skip1:
    lea di,str2
 skip:
    inc si
    dec bl
    JNZ BACK
    int 03H
code ends
end
```

Screenshot:



```
-d
0744:0000  48 45 59 20 41 4E 44 20-48 45 59 20 57 4F 52 4C HEY AND HEY WORL
0744:0010  44 20 48 45 59 15 48 45-59 03 04 00 00 00 00 00 D HEY.HEY.......
0744:0020  B8 44 07 8E D8 8E C0 BE-00 00 BF 16 00 33 DB 8A .D...........3..
0744:0030  1E 15 00 8A 05 38 04 75-18 33 C9 8A 0E 19 00 F3 .....8.u.3......
0744:0040  A6 75 0B A0 1A 00 FE C0-A2 1A 00 03 F1 4E BF 16 .u...........N..
0744:0050  00 46 FE CB 75 DD CC 59-A9 59 A9 59 A9 59 A9 59 .F..u..Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.Yb..Y.Y.Ya.
```

2.In a string “Good Morning”, write a program to replace the substring “Morning” with “Evening” and display “Good Evening”.

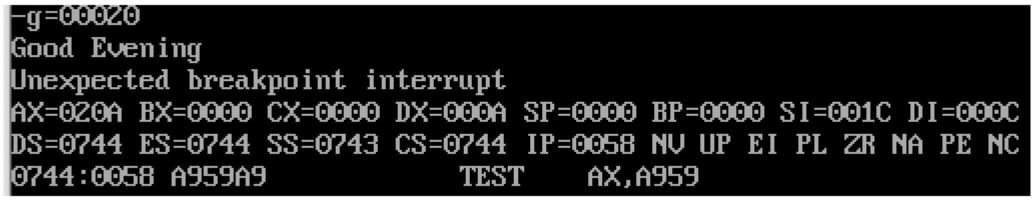
Code:

```
data segment
    str1 db "Good Morning$"
    len1 db $-str1
    str2 db "Morning"
    str3 db "Evening"
    len3 db $-str3
```

```
data ends
code segment
    assume cs:code, ds:data
    mov ax,data
    mov ds,ax
    mov es,ax
    lea di,str1
    lea si,str2
    cld
    sub cx,cx
    mov cl,len1
    lodsb
l2: scasb
    jz l1
    loop l2
l1: dec di
    lea si,str3
    sub cx,cx
    mov cl,len3
    cld
l3: lodsb
    stosb
    loop l3
    mov dx, offset str1
    mov ah, 9
    int 21h
    mov dx,10
    mov ah,2
    int 21h
    int 03H
```

```
code ends
end
```

Screenshot:



```
-g=00020
Good Evening
Unexpected breakpoint interrupt
AX=020A BX=0000 CX=0000 DX=000A SP=0000 BP=0000 SI=001C DI=000C
DS=0744 ES=0744 SS=0743 CS=0744 IP=0058 NV UP EI PL ZR NA PE NC
0744:0058 A959A9            TEST    AX,A959
```

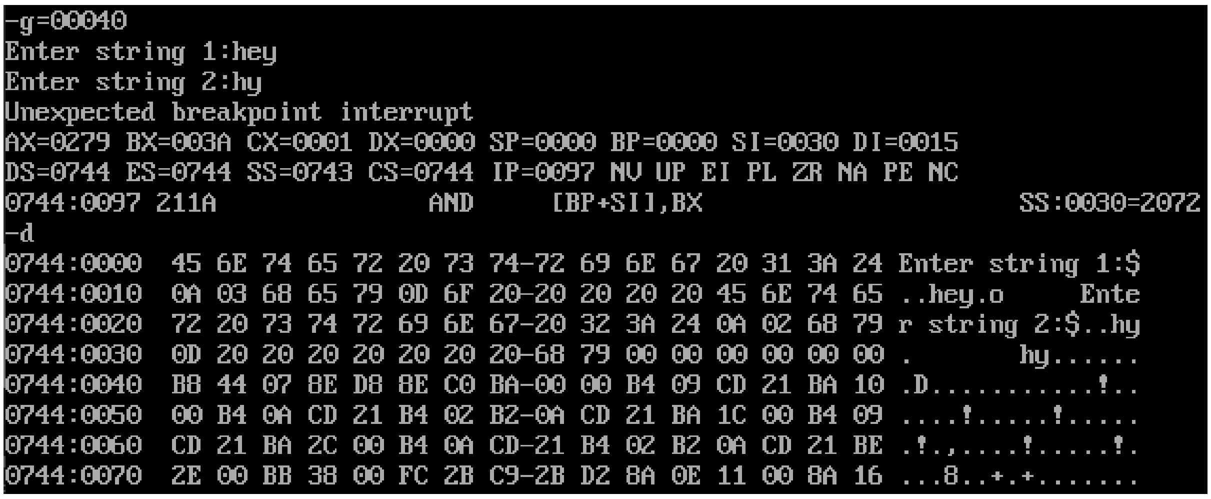
3. Program to enter two strings, Find the characters that match in both the strings, store these Characters

Code:

```
data segment
    str1 db "Good Morning$"
    len1 db $-str1
    str2 db "Morning"
    str3 db "Evening"
    len3 db $-str3
data ends
code segment
    assume cs:code, ds:data
    mov ax,data
    mov ds,ax
    mov es,ax
    lea di,str1
    lea si,str2
    cld
    sub cx,cx
    mov cl,len1
    lodsb
l2: scasb
```

```
    jz l1
    loop l2
l1: dec di
    lea si,str3
    sub cx,cx
    mov cl,len3
    cld
l3: lodsb
    stosb
    loop l3
    mov dx, offset str1
    mov ah, 9
    int 21h
    mov dx,10
    mov ah,2
    int 21h
    int 03H
code ends
end
```

Screenshot:



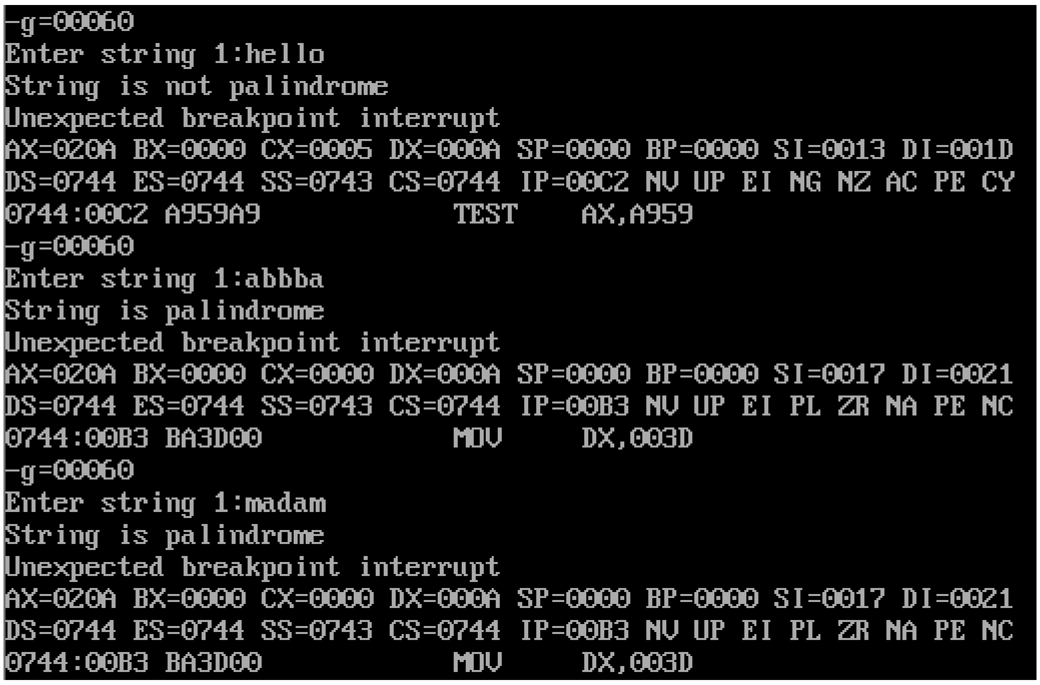
```
-g=00040
Enter string 1:hey
Enter string 2:hy
Unexpected breakpoint interrupt
AX=0279 BX=003A CX=0001 DX=0000 SP=0000 BP=0000 SI=0030 DI=0015
DS=0744 ES=0744 SS=0743 CS=0744 IP=0097 NV UP EI PL ZR NA PE NC
0744:0097 211A              AND     [BP+SI],BX                  SS:0030=2072
-d
0744:0000  45 6E 74 65 72 20 73 74-72 69 6E 67 20 31 3A 24 Enter string 1:$
0744:0010  0A 03 68 65 79 0D 6F 20-20 20 20 20 45 6E 74 65 ..hey.o     Ente
0744:0020  72 20 73 74 72 69 6E 67-20 32 3A 24 0A 02 68 79 r string 2:$..hy
0744:0030  0D 20 20 20 20 20 20 20-68 79 00 00 00 00 00 00 .       hy......
0744:0040  B8 44 07 8E D8 8E C0 BA-00 00 B4 09 CD 21 BA 10 .D..........!..
0744:0050  00 B4 0A CD 21 B4 02 B2-0A CD 21 BA 1C 00 B4 09 ....!.....!.....
0744:0060  CD 21 BA 2C 00 B4 0A CD-21 B4 02 B2 0A CD 21 BE .!.,....!.....!.
0744:0070  2E 00 BB 38 00 FC 2B C9-2B D2 8A 0E 11 00 8A 16 ...8..+.+.......
```

4. Write a Program to check whether the input string is palindrome or not. Get the string through the user.

Code:

```
data segment
 msg1 db "Enter string 1:$"
 str1 db 10,?, 10 dup(' ')
 str2 db 10,?, 10 dup(' ')
 success_msg db "String is palindrome$"
 fail_msg db "String is not palindrome$"
data ends
code segment
    assume cs:code, ds:data
    mov ax,data
    mov ds,ax
    mov es,ax
    mov dx,offset msg1
    mov ah,09H
    int 21H
    mov dx, offset str1
    mov ah, 0ah
    int 21h
    mov ah, 2
    mov dl, 0AH
    int 21h
    lea si,str1+2
    lea di,str2
    sub cx,cx
    mov cl,str1[1]
    add si,cx
    dec si
    REVERSE:mov al,[si]
        mov [di],al
        inc di
        dec si
        loop REVERSE
        lea si,str1+2
        lea di,str2
        cld
        sub cx,cx
        mov cl,str1[1]
    l2: cmpsb
        jnz l1
        loop l2
        mov dx,offset success_msg
        mov ah,09H
        int 21H
        mov dx,10
        mov ah,2
        int 21h
        int 03h
    l1 : mov dx,offset fail_msg
        mov ah,09H
        int 21H
        mov dx,10
        mov ah,2
        int 21h
        int 03h
code ends
end
```

Screenshot:



```
-g=00060
Enter string 1:hello
String is not palindrome
Unexpected breakpoint interrupt
AX=020A BX=0000 CX=0005 DX=000A SP=0000 BP=0000 SI=0013 DI=001D
DS=0744 ES=0744 SS=0743 CS=0744 IP=00C2 NV UP EI NG NZ AC PE CY
0744:00C2 A959A9            TEST    AX,A959
-g=00060
Enter string 1:abbba
String is palindrome
Unexpected breakpoint interrupt
AX=020A BX=0000 CX=0000 DX=000A SP=0000 BP=0000 SI=0017 DI=0021
DS=0744 ES=0744 SS=0743 CS=0744 IP=00B3 NV UP EI PL ZR NA PE NC
0744:00B3 BA3D00            MOV     DX,003D
-g=00060
Enter string 1:madam
String is palindrome
Unexpected breakpoint interrupt
AX=020A BX=0000 CX=0000 DX=000A SP=0000 BP=0000 SI=0017 DI=0021
DS=0744 ES=0744 SS=0743 CS=0744 IP=00B3 NV UP EI PL ZR NA PE NC
0744:00B3 BA3D00            MOV     DX,003D
```

Lab 8

1. **Write an assembly language to convert the code of a 2 digit BCD number to hexadecimal and viceversa.**

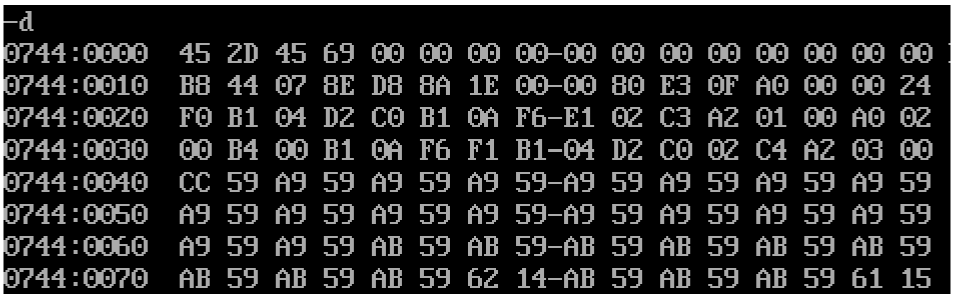
Code:

```
data segment
    n1 db 45h
    n2 db ?
    n3 db 45h
    n4 db ?
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov bl,n1
    and bl,0Fh
    mov al,n1
    and al,0f0h
    mov cl,04h
    rol al,cl
    mov cl,0Ah
    mul cl
    add al,bl
    mov n2,al
    mov al,n3
    mov ah,00h
    mov cl,0ah
    div cl
    mov cl,04h
    rol al,cl
    add al,ah
    mov n4,al
    int 3
code ends
end
```

Screenshot:

Bcd(45) => Hex(2D)

Hex(45) => Bcd(69)



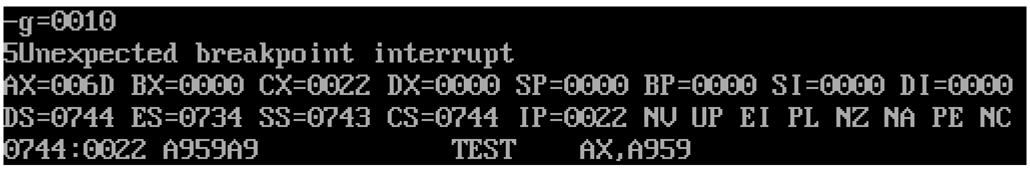
```
-d
0744:0000  45 2D 45 69 00 00 00 00-00 00 00 00 00 00 00
0744:0010  B8 44 07 8E D8 8A 1E 00-00 80 E3 0F A0 00 00 24
0744:0020  F0 B1 04 D2 C0 B1 0A F6-E1 02 C3 A2 01 00 A0 02
0744:0030  00 B4 00 B1 0A F6 F1 B1-04 D2 C0 02 C4 A2 03 00
0744:0040  CC 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15
```

2. **Write an assembly language to convert a number entered through keyboard to 7 segment display.**

Code:

```
data segment
    lookup db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh
,6fh
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov bx,offset lookup
    ;read number
    mov ah,01h
    int 21h
    and al,0fh
    xlat
    mov ah,00h
    int 3
code ends
end
```

Screenshot:



```
-g=0010
5Unexpected breakpoint interrupt
AX=006D BX=0000 CX=0022 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0022 NV UP EI PL NZ NA PE NC
0744:0022 A959A9            TEST    AX,A959
```

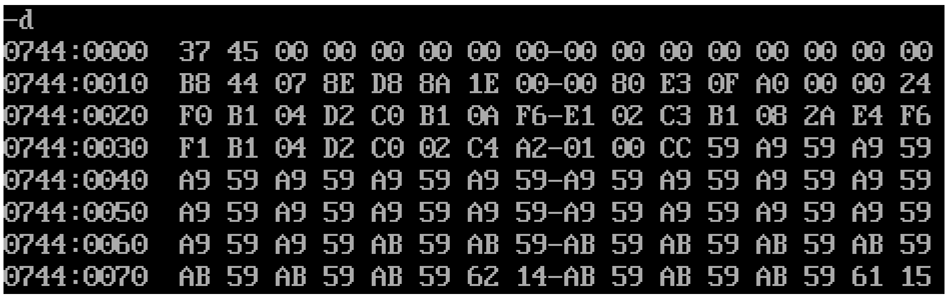
3. **Write an assembly program to convert an 8 bit BCD number to corresponding octal number.**

Code:

```
data segment
    n1 db 37h
    n2 db ?
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov bl,n1
    and bl,0Fh
    mov al,n1
    and al,0f0h
    mov cl,04h
    rol al,cl
    mov cl,0Ah
    mul cl
    add al,bl
    mov cl,08
    sub ah,ah
    div cl
    mov cl,04
    rol al,cl
    add al,ah
    mov n2,al
    int 3
code ends
end
```

Screenshot:

Bcd(37) => hex(45)



```
-d
0744:0000  37 45 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0744:0010  B8 44 07 8E D8 8A 1E 00-00 80 E3 0F A0 00 00 24
0744:0020  F0 B1 04 D2 C0 B1 0A F6-E1 02 C3 B1 08 2A E4 F6
0744:0030  F1 B1 04 D2 C0 02 C4 A2-01 00 CC 59 A9 59 A9 59
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15
```

4. Write a program to check whether two strings are same irrespective of their case.

Code:

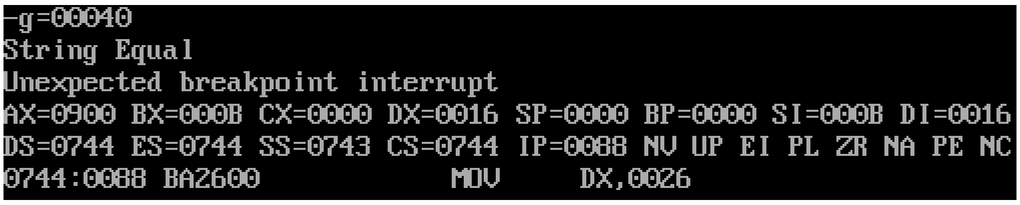
```
data segment
    str1 db "smitVaghani"
    strlen1 = $-str1
    str2 db "DiruVaghani"
    strlen2 = $-str2
    msg1 db "String Equal ",13,10,'$'
    msg2 db "String Not Equal",13,10,'$'
data ends
code segment
assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    mov es,ax
    lea si,str1
    lea di,str2
    mov al,strlen1
    mov bl,strlen2
    cmp al,bl
    jnz notEqual
    mov cl,61h
    loop1:
```

```
    cmp [si],cl
    jae upper_si
return:
    cmp [di],cl
    jae upper_di
    inc si
    inc di
    dec al
    jnz loop1
    lea si,str1
    lea di,str2
    xor cx,cx
    mov cl,strlen1
    repe cmpsb
    jnz notEqual
    jmp equal
upper_si:
    sub byte ptr[si],20h
    jmp return
upper_di:
    sub byte ptr[di],20h
    jmp return
equal:
    lea dx,msg1
    mov ah,09h
    int 21h
    int 03h
notEqual:
    lea dx,msg2
    mov ah,09h
```

```
        int 21h
        int 03h
code ends
end
```

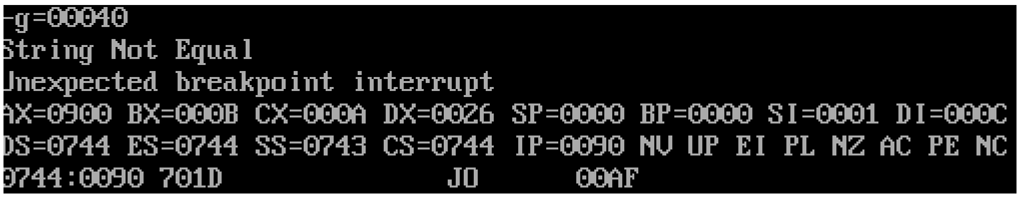
Screenshots:

1.strings are “smitVaghani” and “SmitVaghani”:



```
-g=00040
String Equal
Unexpected breakpoint interrupt
AX=0900 BX=000B CX=0000 DX=0016 SP=0000 BP=0000 SI=000B DI=0016
DS=0744 ES=0744 SS=0743 CS=0744 IP=0088 NV UP EI PL ZR NA PE NC
0744:0088 BA2600            MOV     DX,0026
```

2.strings are “smitVaghani” and “DiruVaghani”:



```
-g=00040
String Not Equal
Unexpected breakpoint interrupt
AX=0900 BX=000B CX=000A DX=0026 SP=0000 BP=0000 SI=0001 DI=000C
DS=0744 ES=0744 SS=0743 CS=0744 IP=0090 NV UP EI PL NZ AC PE NC
0744:0090 701D              JO      00AF
```

Lab 9

1. **Write an assembly language to find the LCM of two numbers. (Pass the parameters to the procedure using registers, pointer and stack)**

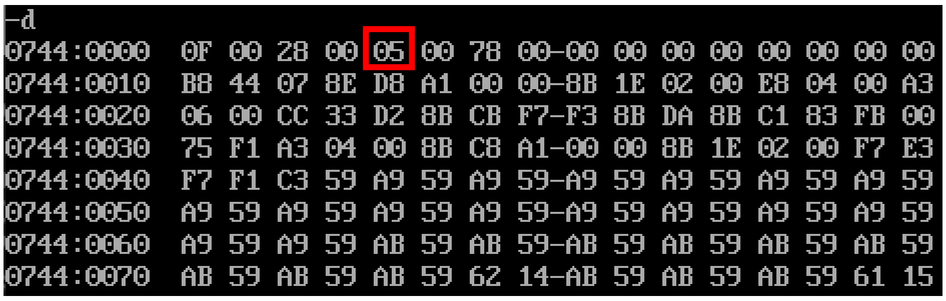
Code:

```
;Perameter passed by register
data segment
    num1 dw 0015
    num2 dw 0040
    gcd dw ?
    lcm dw ?
data ends
```

```
code segment
    assume ds:data,cs:code
    mov ax,data
    mov ds,ax

    mov ax,num1
    mov bx,num2
    call calcLCM
    mov lcm,ax
    int 03h

    calcLCM proc near
    l1:
        xor dx,dx ;Algorithm:
        mov cx,bx ;while(num!=0){
        div bx  ; temp=num;
        mov bx,dx ; num=(a%num);
        mov ax,cx; ; a=temp;
        cmp bx,0 ;}
        jne l1

        mov gcd ,ax ;gcd=a;
        mov cx,ax ;lcm=(a*b)/gcd
        mov ax,num1
        mov bx,num2
        mul bx
        div cx
        RET
    calcLCM endp
code ends
```

```
end
```

Screenshot:



```
-d
0744:0000  0F 00 28 00 05 00 78 00-00 00 00 00 00 00 00 00
0744:0010  B8 44 07 8E D8 A1 00 00-8B 1E 02 00 E8 04 00 A3
0744:0020  06 00 CC 33 D2 8B CB F7-F3 8B DA 8B C1 83 FB 00
0744:0030  75 F1 A3 04 00 8B C8 A1-00 00 8B 1E 02 00 F7 E3
0744:0040  F7 F1 C3 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15
```

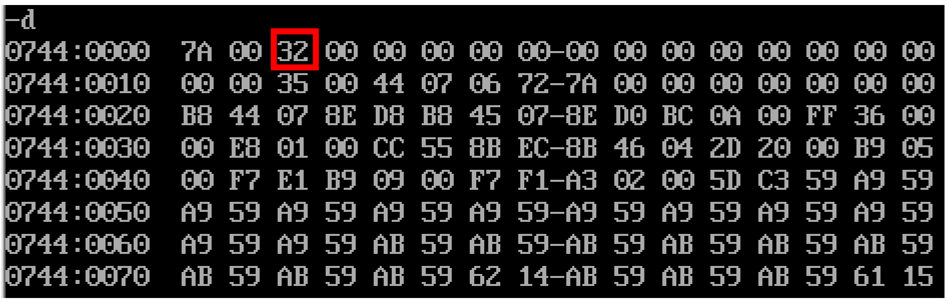
2. **Write an assembly language program to convert temperature from Fahrenheit to Celsius. [(F − 32) × 5/9 = °C]**

Code:

```
data segment
    Fahrenheit dw 122
    Celsius dw ? ;It should be 50
data ends
mystack segment stack
    dw 5 dup(0)
    stack_top label word
mystack ends
code segment
    assume cs:code,ds:data,ss:mystack
    mov ax,data
    mov ds,ax
    mov ax,mystack
    mov ss,ax

    mov sp,offset stack_top
    push Fahrenheit
    call calcCelsius
    int 03

    calcCelsius proc near
    push bp
    mov bp,sp
    mov ax,[bp+4]

    sub ax,32
    mov cx,0005
    mul cx
    mov cx,0009
    div cx
    mov Celsius,ax
    pop bp
    RET
    calcCelsius endp
code ends
end
```

Screenshot:

32H=50



```
-d
0744:0000  7A 00 32 00 00 00 00 00-00 00 00 00 00 00 00 00
0744:0010  00 00 35 00 44 07 06 72-7A 00 00 00 00 00 00 00
0744:0020  B8 44 07 8E D8 B8 45 07-8E D0 BC 0A 00 FF 36 00
0744:0030  00 E8 01 00 CC 55 8B EC-8B 46 04 2D 20 00 B9 05
0744:0040  00 F7 E1 B9 09 00 F7 F1-A3 02 00 5D C3 59 A9 59
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15
```

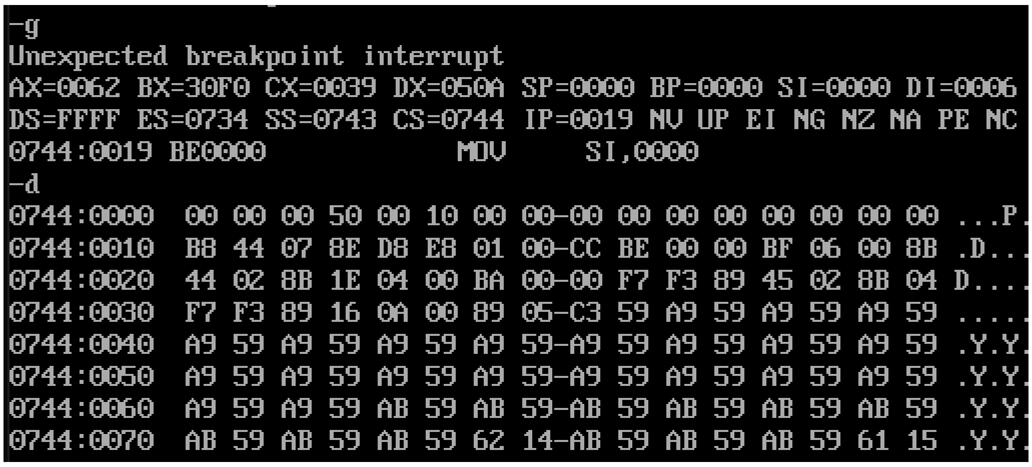
Lab 10

1.Using near procedure write an assembly language to divide a 32 bit number by 16 bit number. The program should handle the quotient size exceeds 16 bits.

code:

```
data segment
    num1 dd 50000000h
    num2 dw 1000h
    quotient dd ?
    remainder dw ?
data ends
code segment
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    call divide
    int 03h

    divide proc near
        lea si,num1
        lea di,quotient
        mov ax,[si + 2]
        mov bx,num2
        mov dx,0h
        div bx
        mov [di+2],ax
        mov ax,[si]
        div bx
        mov remainder,dx
        mov [di],ax
        RET
    divide endp
code ends
end
```

Screenshot:



```
-g
Unexpected breakpoint interrupt
AX=0062 BX=30F0 CX=0039 DX=050A SP=0000 BP=0000 SI=0000 DI=0006
DS=FFFF ES=0734 SS=0743 CS=0744 IP=0019 NV UP EI NG NZ NA PE NC
0744:0019 BE0000          MOV     SI,0000
-d
0744:0000  00 00 00 50 00 10 00 00-00 00 00 00 00 00 00 00 ...P.
0744:0010  B8 44 07 8E D8 E8 01 00-CC BE 00 00 BF 06 00 8B .D...
0744:0020  44 02 8B 1E 04 00 BA 00-00 F7 F3 89 45 02 8B 04 D....
0744:0030  F7 F3 89 16 0A 00 89 05-C3 59 A9 59 A9 59 A9 59 .....
0744:0040  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.
0744:0050  A9 59 A9 59 A9 59 A9 59-A9 59 A9 59 A9 59 A9 59 .Y.Y.
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59 .Y.Y.
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15 .Y.Y.
```

2.Write an assembly language program using far procedure to find whether a given number is Armstrong or not. Display appropriate message using procedure.

code:

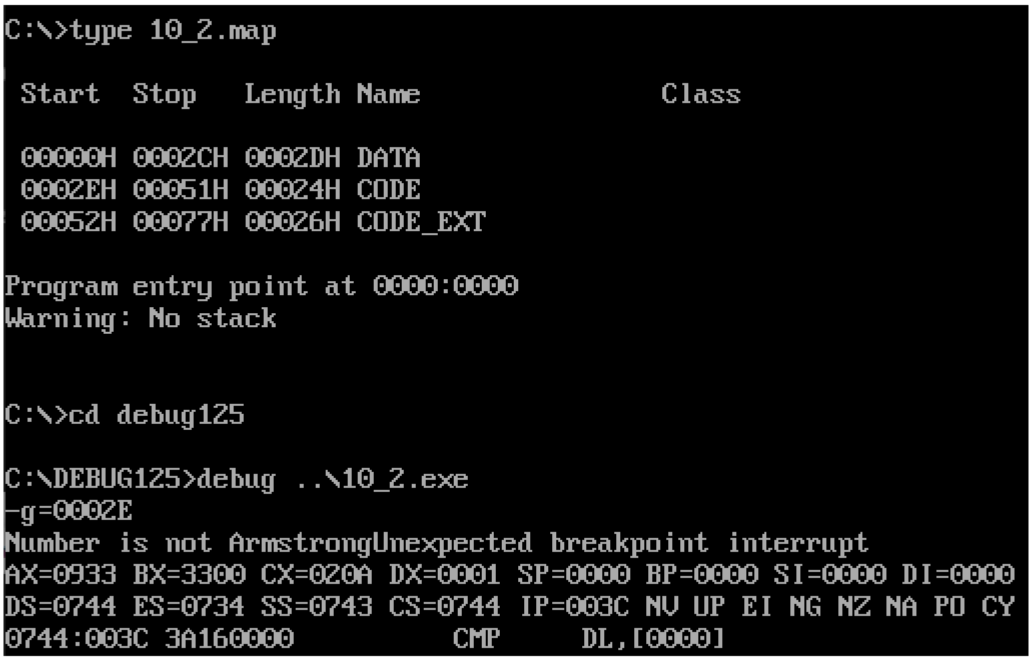
Driver Code

```
public num1
extrn armstrong:far
data segment word public
    num1 db 53h
    str1 db "Number is not Armstrong$"
    str2 db "Number is Armstrong$"
data ends
code segment word public
    assume cs:code,ds:data
    mov ax,data
    mov ds,ax
    call armstrong
    call arm
    int 03h
    arm proc near
        cmp dl,num1
        jnz loop2
        mov dx, offset str2
        mov ah, 9
        int 21h
        RET
        loop2:
            mov dx, offset str1
            mov ah, 9
            int 21h
            RET
    arm endp
code ends
end
```

external Code:

```
public armstrong
extrn num1:byte
code_ext segment word public
    assume cs:code_ext
    armstrong proc far
        xor ax,ax
        xor cx,cx
        xor bx,bx
        xor dx,dx
        mov al,num1
        mov cl,0ah
        mov bl,03h
        loop1:
            div cl
            mov bh,al
            mov al,ah
            mov ch,al
            sub ah,ah
            mul ch
            mul ch
            add dl,al
            mov al,bh
            dec bl
            jnz loop1
            RET
    armstrong endp
code_ext ends
end
```

Screenshot:



```
C:\>type 10_2.map

 Start  Stop   Length Name               Class

 00000H 0002CH 0002DH DATA
 0002EH 00051H 00024H CODE
 00052H 00077H 00026H CODE_EXT

Program entry point at 0000:0000
Warning: No stack


C:\>cd debug125

C:\DEBUG125>debug ..\10_2.exe
-g=0002E
Number is not ArmstrongUnexpected breakpoint interrupt
AX=0933 BX=3300 CX=020A DX=0001 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=003C NV UP EI NG NZ NA PO CY
0744:003C 3A160000          CMP     DL,[0000]
```

3.Write a program using Maco to add two numbers. Macro should be written in another file.

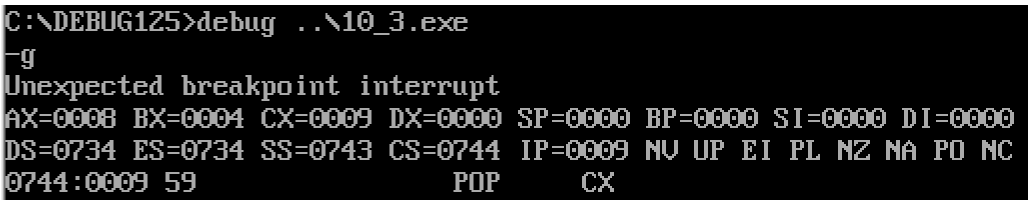
code:

```
INCLUDE macro.h
code segment
    assume cs:code
    xor ax,ax
    sum 4,4
    int 03h
code ends
end
```

micro.h

```
sum macro num1,num2
    mov al,num1
    mov bl,num2
    add al,bl
endm
```

Screenshot:



```
C:\DEBUG125>debug ..\10_3.exe
-g
Unexpected breakpoint interrupt
AX=0008 BX=0004 CX=0009 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0009 NV UP EI PL NZ NA PO NC
0744:0009 59            POP     CX
```

Lab 11

1.Write an assembly language to handle the divide by zero interrupt.

Code:

11_1.asm

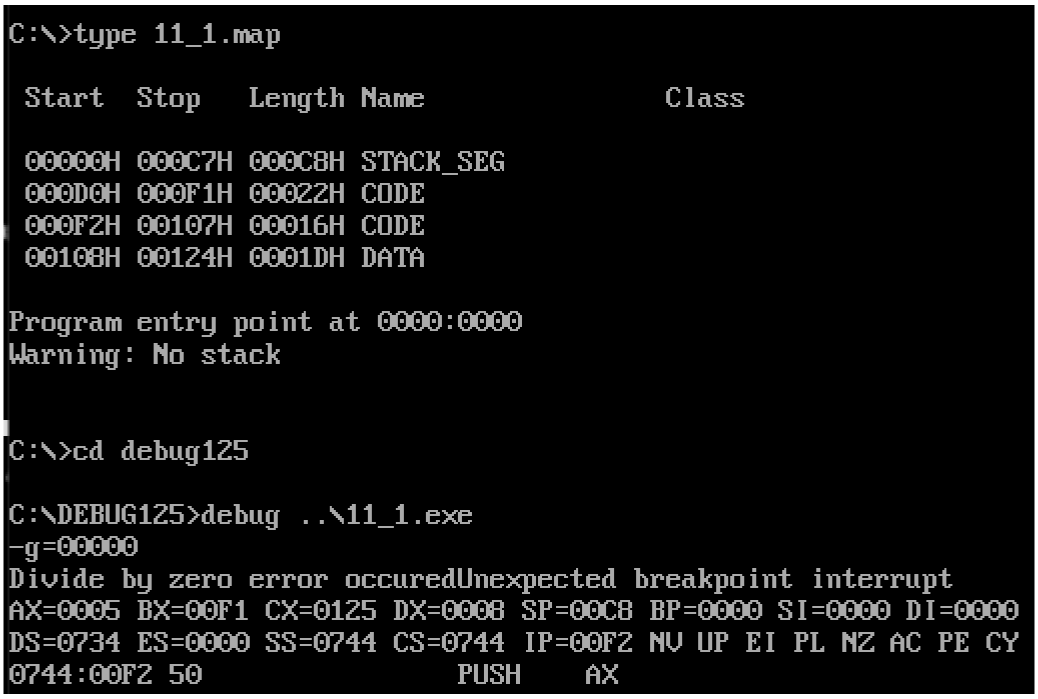
```
stack_seg segment stack
    dw 100 dup(0)
    top_st label word
    stack_seg ends
    extrn bad_div:far
code segment
    assume cs:code,ss:stack_seg
    mov ax,stack_seg
    mov ss,ax
    mov sp,offset top_st
    mov ax,0000
    mov es,ax
    mov al,05
    mov bl,00
    mov word ptr es:0000,offset bad_div
    mov word ptr es:0002,seg bad_div
    div bl
    int 03
code ends
end
```

11_1_1.asm

```
data segment word public
    str1 db "Divide by zero error occured$"
```

```
data ends
public bad_div
code segment word public
    bad_div proc far
        assume cs:code,ds:data
        push ax
        push ds
        mov ax,data
        mov ds,ax
        lea dx,str1
        mov ah,09
        int 21h
        pop ds
        pop ax
        pop bx
        add bx,2
        push bx
        iret
    bad_div endp
code ends
end
```

Screenshot:



```
C:\>type 11_1.map

 Start  Stop   Length Name               Class

 00000H 000C7H 000C8H STACK_SEG
 000D0H 000F1H 00022H CODE
 000F2H 00107H 00016H CODE
 00108H 00124H 0001DH DATA

Program entry point at 0000:0000
Warning: No stack


C:\>cd debug125

C:\DEBUG125>debug ..\11_1.exe
-g=00000
Divide by zero error occuredUnexpected breakpoint interrupt
AX=0005 BX=00F1 CX=0125 DX=0008 SP=00C8 BP=0000 SI=0000 DI=0000
DS=0734 ES=0000 SS=0744 CS=0744 IP=00F2 NV UP EI PL NZ AC PE CY
0744:00F2 50            PUSH    AX
```

2.Write an assembly language to override the overflow interrupt.

Code:

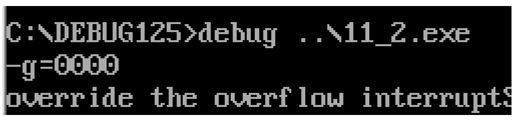
11_2.asm

```
stack_seg segment stack
    dw 100 dup(0)
    top_st label word
stack_seg ends
extrn bad_overflow:far
code segment
    assume cs:code,ss:stack_seg
    mov ax,stack_seg
    mov ss,ax
    mov sp,offset top_st
    mov ax,0000
    mov es,ax
    mov word ptr es:0010,offset bad_overflow
    mov word ptr es:0012,seg bad_overflow
    int 04
code ends
end
```

11_2_2.asm

```
data segment word public
    str1 db "override the overflow interrupt$"
data ends
public bad_overflow
code segment word public
    bad_overflow proc far
        assume cs:code,ds:data
        push ax
        push ds
        mov ax,data
        mov ds,ax
        lea dx,str1
        mov ah,09
        int 21h
        pop ds
        pop ax
        iret
    bad_overflow endp
code ends
end
```

Screenshot:



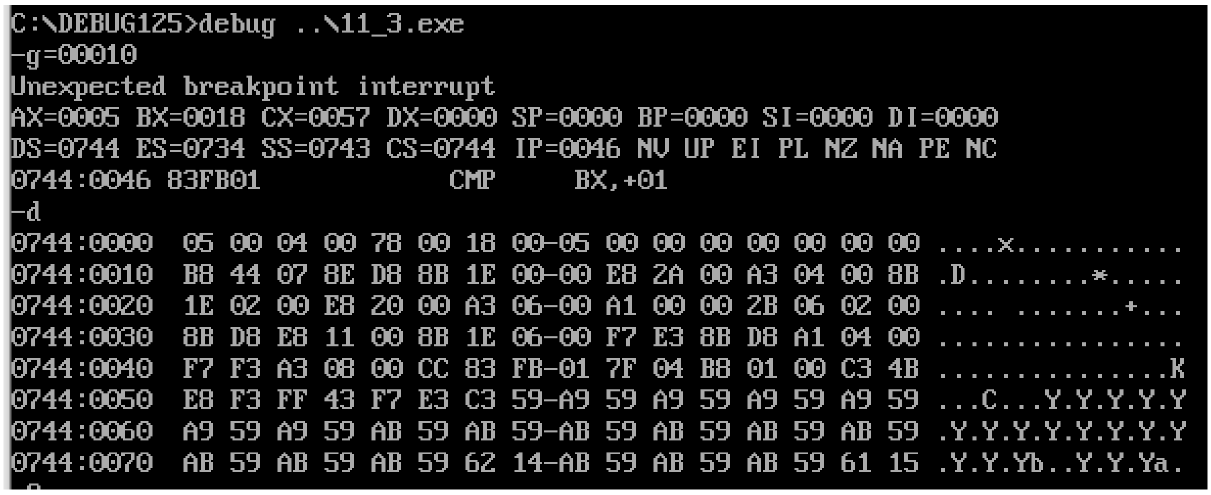
```
C:\DEBUG125>debug ..\11_2.exe
-g=0000
override the overflow interrupt$
```

3. Write an assembly language program to find the factorial recursively and find the nCr.

Code:

```
data segment
    n dw 5
    r dw 4
    resultn dw ?
    resultr dw ?
    result dw ?
data ends
code segment
    assume cs:code, ds:data
    mov ax,data
    mov ds,ax
    mov bx,n
    call factorial
    mov resultn,ax
    mov bx,r
    call factorial
    mov resultr,ax
    mov ax,n
    sub ax,r
    mov bx,ax
    call factorial
    mov bx,resultr
    mul bx
    mov bx,ax
    mov ax,resultn
    div bx
    mov result,ax
    int 03h

    factorial PROC NEAR
        cmp bx,1
        jg l1
        mov ax,1
        RET
        l1: dec bx
        call factorial
        inc bx
        mul bx
        RET
    factorial ENDP
code ends
End
```

Screenshot:



```
C:\DEBUG125>debug ..\11_3.exe
-g=00010
Unexpected breakpoint interrupt
AX=0005 BX=0018 CX=0057 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=0046 NV UP EI PL NZ NA PE NC
0744:0046 83FB01            CMP     BX,+01
-d
0744:0000  05 00 04 00 78 00 18 00-05 00 00 00 00 00 00 00  ....x...........
0744:0010  B8 44 07 8E D8 8B 1E 00-00 E8 2A 00 A3 04 00 8B  .D........*.....
0744:0020  1E 02 00 E8 20 00 A3 06-00 A1 00 00 2B 06 02 00  .... .......+...
0744:0030  8B D8 E8 11 00 8B 1E 06-00 F7 E3 8B D8 A1 04 00  ................
0744:0040  F7 F3 A3 08 00 CC 83 FB-01 7F 04 B8 01 00 C3 4B  ...............K
0744:0050  E8 F3 FF 43 F7 E3 C3 59-A9 59 A9 59 A9 59 A9 59  ...C...Y.Y.Y.Y.Y
0744:0060  A9 59 A9 59 AB 59 AB 59-AB 59 AB 59 AB 59 AB 59  .Y.Y.Y.Y.Y.Y.Y.Y
0744:0070  AB 59 AB 59 AB 59 62 14-AB 59 AB 59 AB 59 61 15  .Y.Y.Yb..Y.Y.Ya.
```