Krish Jindal
12012104 CSB6
Microprocessor Final Lab File
Date: 21-04-2022

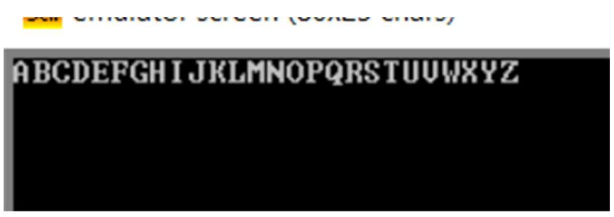
Submitted to: Mohit Dua Sir

Signature

Assignment-1

1. **Code for A-Z**

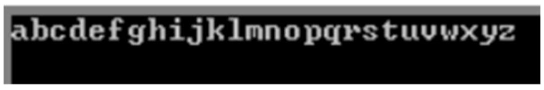
```
start:

mov cl,26

mov dl,'A'

L:

  mov ah,02h

  int 21h

  inc dl

  Loop L

end start
```



```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

2. **Code for a-z**

```
start:

mov cl,26

mov dl,'a'

L:

  mov ah,02h

  int 21h

  inc dl

  Loop L

end start
```



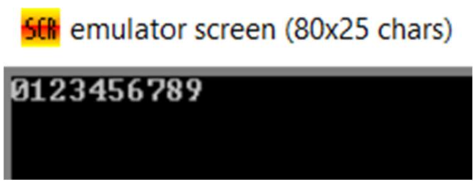
```
abcdefghijklmnopqrstuvwxyz
```

Assignment-2

1. Program to print **0-9**

```
start:
mov cl,10,
mov dl,'0'
L:
  mov ah,02h
  int 21h
  inc dl
  loop L
  end start
```

OUTPUT



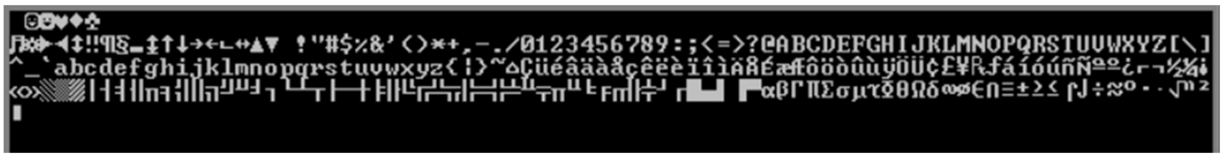
emulator screen (80x25 chars)

```
0123456789
```

2. Program to print **ascii table**

```
start:
mov cx,256,
mov dl,0
L:
   mov ah,02h
   int 21h
   inc dl
   loop L
   end start
```

OUTPUT

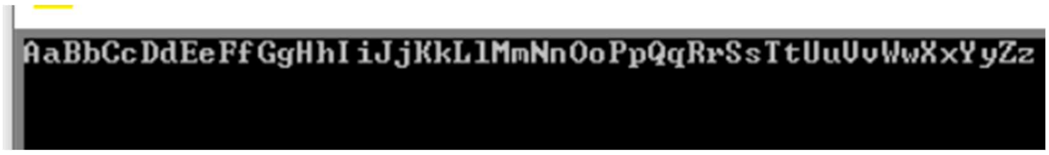


```
 ☺☻♥♦♣
♫☼►◄↕‼¶§▬↨↑↓→←∟↔▲▼ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
^_`abcdefghijklmnopqrstuvwxyz{|}~⌂ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖÜ¢£¥₧ƒáíóúñÑªº¿⌐¬½¼¡
«»░▒▓│┤╡╢╖╕╣║╗╝╜╛┐└┴┬├─┼╞╟╚╔╩╦╠═╬╧╨╤╥╙╘╒╓╫╪┘┌█▄▌▐▀αßΓπΣσµτΦΘΩδ∞φε∩≡±≥≤⌠⌡÷≈°∙·√ⁿ²
```

3. Program to print **AaBbCc…….**

```
start:

mov cl,26,

mov bl,'A'

mov bh,'a'

L:

   mov dl,bl

   mov ah,02h

   int 21h

   inc bl


   mov dl,bh

   mov ah,02h

   int 21h

   inc bh

   loop L

   end start
```

OUTPUT

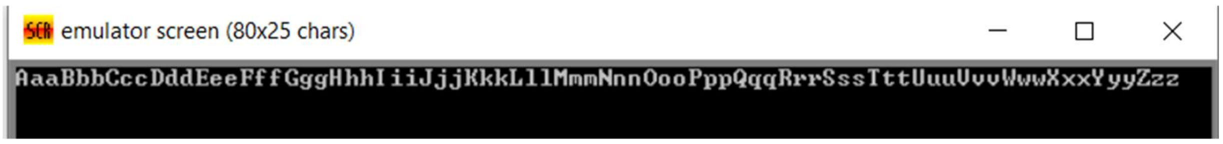


```
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
```

4. Program to print **AaaBbbCcc…..**

```
start:

mov cl,26,
mov bl,'A'
mov bh,'a'
L:
   mov dl,bl
   mov ah,02h
   int 21h
   inc bl

   mov dl,bh
   mov ah,02h
   int 21h
   int 21h
   inc bh
   loop L
   end start
```

OUTPUT



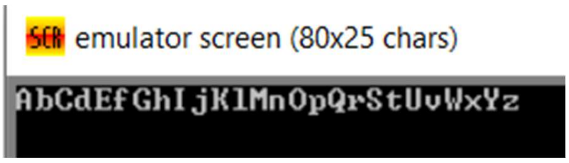
emulator screen (80x25 chars)

```
AaaBbbCccDddEeeFffGggHhhIiiJjjKkkLllMmmNnnOooPppQqqRrrSssTttUuuVvvWwwXxxYyyZzz
```

5. Program to print AbCdEf……….z

```
start:
mov cl,13,
mov bl,'A'
mov bh,'b'
L:
   mov dl,bl
   mov ah,02h
   int 21h
   inc bl
   inc bl
   mov dl,bh
   mov ah,02h
   int 21h
   inc bh
   inc bh
   loop L
   end start
```

OUTPUT



emulator screen (80x25 chars)

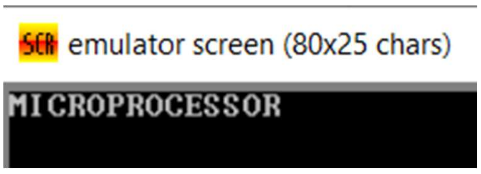
```
AbCdEfGhIjKlMnOpQrStUvWxYz
```

Assignment-3

1. Program to print **the string using 09h function**

```
.data

string db "MICROPROCESSOR$"

start:

mov ax,@data

mov ds,ax

mov dl,offset string

mov ah ,09h

int 21h

end start
```

OUTPUT

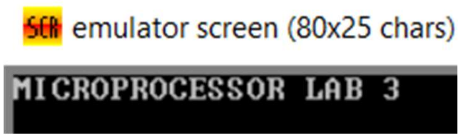


```
MICROPROCESSOR
```

2. Program to print **the string character wise**

```
.data

a db "MICROPROCESSOR LAB 3$"

start:

mov ax, @data

mov ds, ax

mov si, offset a

L:

   mov dl, [si]

   mov ah, 02h

   int 21h
   inc si
   cmp [si],'$'
   jnz L
end start
```

OUTPUT

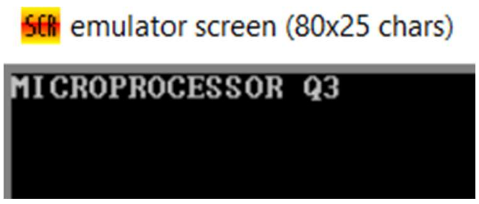

emulator screen (80x25 chars)

```
MICROPROCESSOR LAB 3
```

3. Program to print **the string using 09h function (16bit)**

```
.data
a dw "MICROPROCESSOR Q3$"
start:
mov ax, @data
mov ds,ax
mov dx,offset a
mov ah, 09h
int 21h
end start
```

OUTPUT

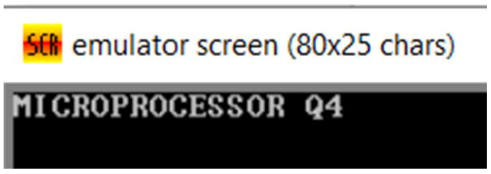

emulator screen (80x25 chars)

```
MICROPROCESSOR Q3
```

4. Program to print **the string character wise(16bit)**

```
.data
a dw "MICROPROCESSOR Q4$"
start:
mov ax, @data
mov ds, ax
mov si, offset a
L:
   mov dx, [si]
   mov ah, 02h
   int 21h
   nc sii
   cmp [si],'$'
   jnz L

end start
```

OUTPUT



emulator screen (80x25 chars)

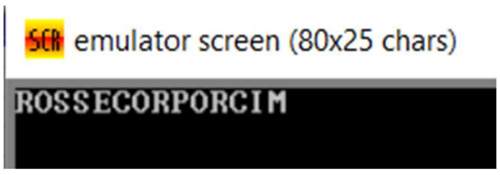
```
MICROPROCESSOR Q4
```

Assignment-4

1. Program to reverse a given String

```
.data
a db "MICROPROCESSOR$"
.code
start:
mov ax,@data
mov ds,ax
mov si,offset a
mov cx,0
L1:
   inc si
   inc cx
   cmp [si],'$'
   jne L1
L2:
   dec si
   mov dl,[si]
   mov ah,02h
   int 21h
   loop L2
end start
```

OUTPUT



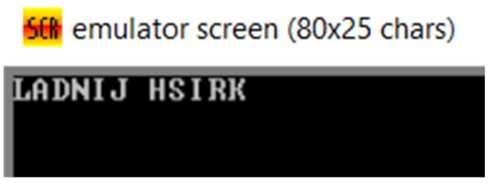
```
ROSSECORPORCIM
```

2. Program to reverse name

```
.data
```

```
a db "KRISH JINDAL$"
.code
start:
mov ax,@data
mov ds,ax
mov si,offset a
mov cx,0
L1:
   inc si
   inc cx
   cmp [si],'$'
   jne L1
L2:
   dec si
   mov dl,[si]
   mov ah,02h
   int 21h
   loop L2
end start
```

OUTPUT

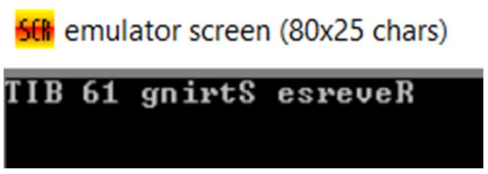

emulator screen (80x25 chars)

```
LADNIJ HSIRK
```

3. Program to reverse a given String 16 bit

```
.data
a db "Reverse String 16 BIT$"
.code
start:
mov ax,@data
mov ds,ax
mov si,offset a
mov cx,0
L1:
   inc si
   inc cx
   cmp [si],'$'
   jne L1
L2:
   dec si
   mov dx,[si]
   mov ah,02h
   int 21h
   loop L2
end start
```

OUTPUT

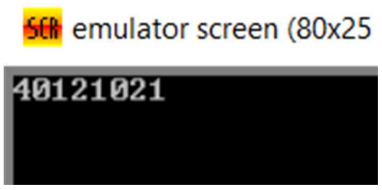

emulator screen (80x25 chars)

```
TIB 61 gnirtS esreveR
```

4. Program to reverse roll number.

```
.data
a db "12012104$"
.code
start:
mov ax,@data
mov ds,ax
mov si,offset a
mov cx,0
L1:
   inc si
   inc cx
   cmp [si],'$'
   jne L1
L2:
   dec si
   mov dl,[si]
   mov ah,02h
   int 21h
   loop L2
end start
```

OUTPUT



emulator screen (80x25

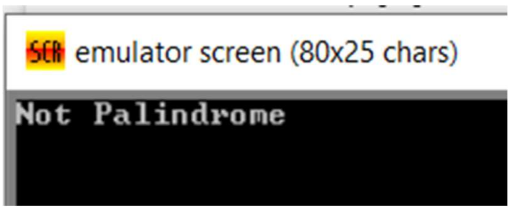
```
40121021
```

Assignment-5

1. **Write a program to check if the given string is a palindrome or not.**

```
.data
a db "ABCCB$"
b db "Palindrome$"
c db "Not Palindrome$"
.code
start:
mov ax,@data
mov ds,ax
lea si,a
lea di,a
mov cl,0
L1:
   inc si
   inc cl
   cmp [si],'$'
   jne L1
   dec cl
L2:
            dec si
             mov al,[si]
             mov bl,[di]
            cmp al,bl
              jne L3
   inc di
   loop L2
   lea dx,b
   mov ah,09h
   int 21h
   hlt
L3:
   lea dx,c
   mov ah,09h
   int 21h
end start
```

OUTPUT

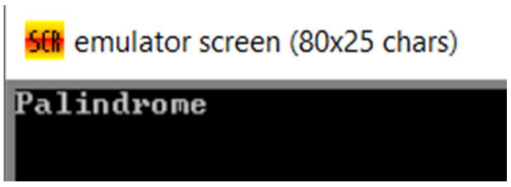


```
Not Palindrome
```

2. Repeat the above program for 16 bits.

```
.data
a dw "ABCBA$"
b dw "Palindrome$"
c dw "Not Palindrome$"
.code
start:
mov ax,@data
mov ds,ax
lea si,a
lea di,a
mov cl,0
L1:
   inc si
   inc cl
   cmp [si],'$'
   jne L1
   dec cl
L2:
   dec si
   mov al,[si]
   mov bl,[di]
   cmp al,bl
   jne L3
   inc di
   loop L2
   lea dx,b
   mov ah,09h
   int 21h
   hlt
L3:
   lea dx,c
   mov ah,09h
   int 21h
end start
```

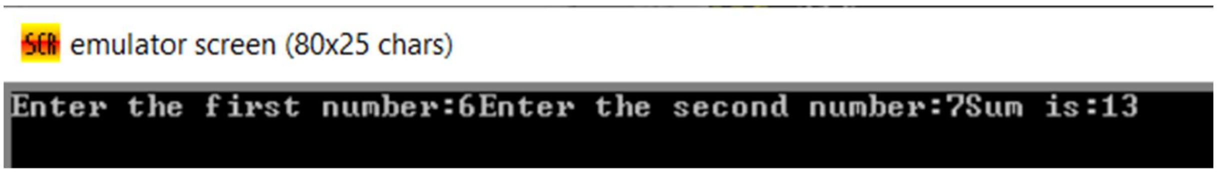
OUTPUT



```
Palindrome
```

3. Write a program to sum two 8 bit single digit numbers.

```
.data
a db "Enter the first number:$"
b db "Enter the second number:$"
c db  "Sum is:$"


.code


start:
mov ax,data
mov ds,ax
mov dx,offset a

mov ah,09h
int 21h
mov ah,01h
int 21h
mov bl,al

mov dx,offset b
mov ah,09h
int 21h
mov ah,01h
int 21h

add al,bl

mov ah,0
aaa
mov bx,ax
add bx,3030h

mov dx,offset c
mov ah,09h
int 21h
mov dl,bh
mov ah,02h
int 21h
mov dl,bl
mov ah,02h
int 21h
end start
```

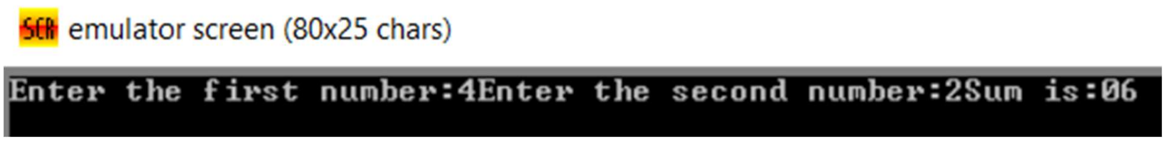


```
Enter the first number:6Enter the second number:7Sum is:13
```

4. Repeat the above program for 16 bit

```
.data

a db "Enter the first number:$"

b db "Enter the second number:$"

c db "Sum is:$"


.code


start:

mov ax,data

mov ds,ax


mov dx,offset a

mov ah,09h

int 21h

mov ah,01h

int 21h

mov bl,al


mov dx,offset b

mov ah,09h

int 21h

mov ah,01h

int 21h

add al,bl

mov ah,0
```

```
aaa

mov bx,ax

add bx,3030h


mov dx,offset c

mov ah,09h

int 21h

mov dl,bh

mov ah,02h

int 21h


mov dl,bl

mov ah,02h

int 21h


end start
```


emulator screen (80x25 chars)

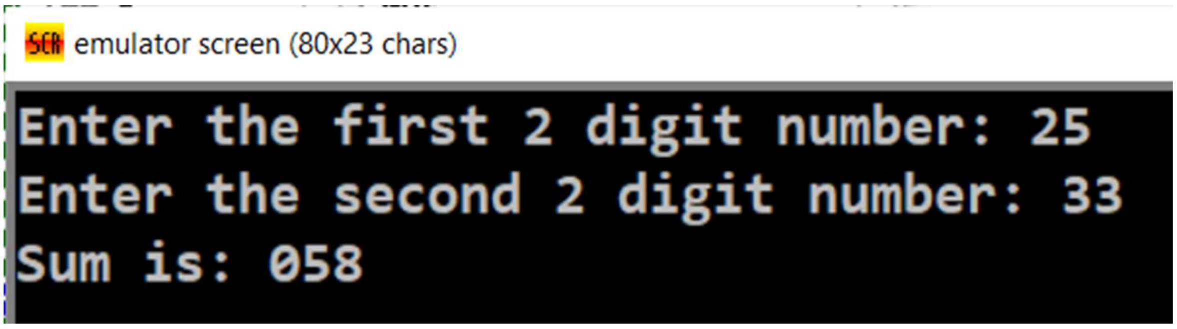
```
Enter the first number:4Enter the second number:2Sum is:06
```

5. Write a program to add two 8-bit multi digit numbers

```
.data

a db "Enter the first 2 digit number: $"

b db "Enter the second 2 digit number: $"

c db "Sum is: $"

n db 10

.code

start:

mov ax,@data

mov ds,ax

lea dx,a

mov ah,09h
```

```
int 21h
mov ah,01h
int 21h
sub al,30h
mov bh,al
int 21h
sub al,30h
mov bl,al          ; bh contains higher digit and bl contains lower digit
; 2nd number
mov ah,02h
mov dl,10
int 21h
mov dl,13
int 21h
lea dx,b
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov ch,al
int 21h
sub al,30h
mov cl,al          ; ch contains higher digit and cl contains lower digit
mov ah,02h
mov dl,10
int 21h
mov dl,13
int 21h
lea dx,c
mov ah,09h
int 21h
```

```
; Perform addition

mov ax,cx

add ax,bx

aaa

mov bl,al

mov al,ah

mov ah,0h

aaa

mov cx,ax

mov ah,02h

mov dl,ch

add dl,30h

int 21h

mov dl,cl

add dl,30h

int 21h

mov dl,bl

add dl,30h

int 21h

end start
```



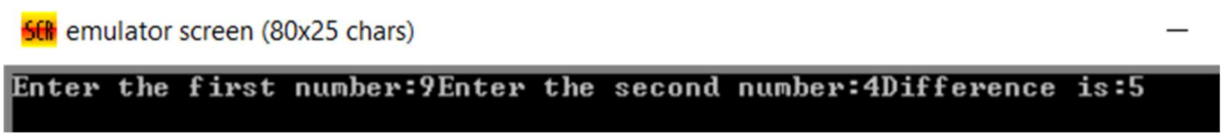
```
Enter the first 2 digit number: 25
Enter the second 2 digit number: 33
Sum is: 058
```

Assignment 6

1. Write a program to subtract two 8-bit single digit numbers

```
.data
a db "Enter the first number:$"
b db  “Enter the second number:$"
c db “Difference is:$"
.code
start:
mov ax,data
mov ds,ax
mov dx,offset a
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov bl,al
mov dx,offset b
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov cl,al
mov al,bl
sub al,cl
mov ah,0
aas
mov bl,al
add bl,30h
```

```
mov dx,offset c

mov ah,09h

int 21h

mov dl,bl

mov ah,02h

int 21h

end start
```



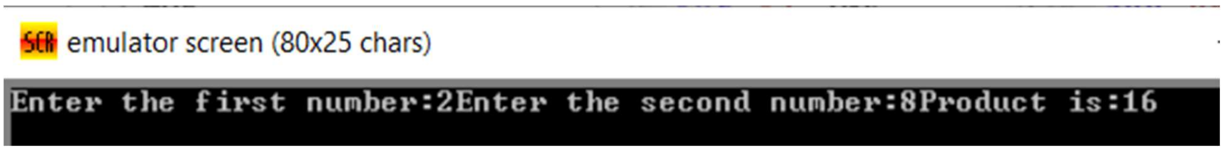
emulator screen (80x25 chars)

```
Enter the first number:9Enter the second number:4Difference is:5
```

2. Write a program to multiply two 8-bit single digit numbers

```
.data

a db "Enter the first number:$"

b db “Enter the second number:$"

c db “Product is:$"

.code

start:

mov ax,data

mov ds,ax

mov dx,offset a

mov ah,09h

int 21h

mov ah,01h

int 21h

sub al,30h

mov bl,al

mov dx,offset b

mov ah,09h
```

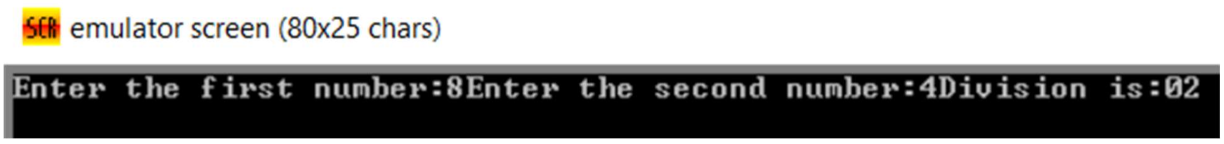
```
int 21h

mov ah,01h

int 21h

sub al,30h

mul bl

mov ah,0

aam

mov bx,ax

add bx,3030h

mov dx,offset c

mov ah,09h

int 21h

mov dl,bh

mov ah,02h

int 21h

mov dl,bl

mov ah,02h

int 21h

end start
```


emulator screen (80x25 chars)

```
Enter the first number:2Enter the second number:8Product is:16
```

3. Write a program to divide two 8-bit single digit numbers

```
.data

a db "Enter the first number:$"

b db “Enter the second number:$"

c db "Division is:$"

.code

start:

mov ax,@data

mov ds,ax
mov dx,offset a
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov bl,al
mov dx,offset b
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov cl,al
mov al,bl
mov ah,0
div cl
mov bx,ax
add bx,3030h
mov dx,offset c
mov ah,09h
int 21h
mov dl,bh
mov ah,02h
int 21h
mov dl,bl
mov ah,02h
int 21h
end start
```

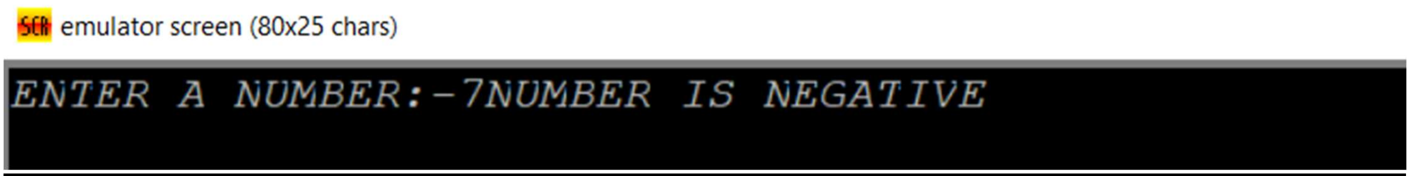


```
Enter the first number:8Enter the second number:4Division is:02
```

4. Write a program to find that 8-bit number is positive or negative

```
.DATA

MSG1 DW "ENTER A NUMBER:$"

MSG2 DW "NUMBER IS POSITIVE$"

MSG3 DW "NUMBER IS NEGATIVE$"

NUM1 DW 9925H

NUM2 DW 2851H


.CODE

START:

   MOV AX, @DATA

   MOV DS, AX

   MOV DX, OFFSET MSG1

   MOV AH, 09H

   INT 21H

   MOV AH, 01H

   INT 21H

   MOV BL, AL

   MOV AH, 01H

   INT 21H


   CMP BL, '-'

   JZ l

   ;PRINT POSITIVE

   MOV DX, OFFSET MSG2

   MOV AH, 09H

   INT 21H

   HLT
```

```
l:
   ;NEGATIVE
   MOV DX, OFFSET MSG3
   MOV AH, 09H
   INT 21H
END START
```

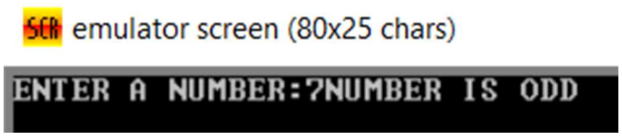


emulator screen (80x25 chars)

```
ENTER A NUMBER:-7NUMBER IS NEGATIVE
```

5. Write a program to find that 8-bit number is even or odd

```
.DATA
MSG1 DW "ENTER A NUMBER:$"
MSG2 DW "NUMBER IS EVEN$"
MSG3 DW “NUMBER IS ODD$"
.CODE
START:
   MOV AX, @DATA
   MOV DS, AX

   MOV DX, OFFSET MSG1
   MOV AH, 09H
   INT 21H

   MOV AH, 01H
   INT 21H

   MOV DX, 0H
   MOV BX, 02H
   DIV BX
   CMP DX, 0H
   JNZ LABEL
   ;PRINT EVEN
   MOV DX, OFFSET MSG2
   MOV AH, 09H
   INT 21H
   HLT
LABEL:
   ;ODD
   MOV DX, OFFSET MSG3
   MOV AH, 09H
   INT 21H
    END START
```

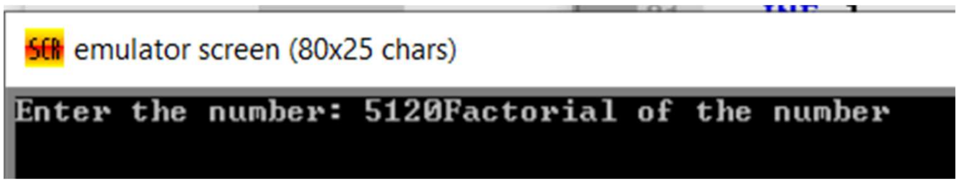

emulator screen (80x25 chars)

```
ENTER A NUMBER:7NUMBER IS ODD
```

6. Write a program to find the factorial of a given number

```
.STACK 100h
.data
a DB "Enter the number: $"
b DB "Factorial of the number $"
.code
start:
MOV AX,@data
MOV DS,AX
MOV DX,OFFSET a
MOV AH,09h
INT 21h
MOV AH,01h
INT 21h
SUB AL,30h
MOV CH,0
MOV CL,AL
MOV AX,1
```

```
l:MUL CX
  DEC CX
  CMP CX,0
  JNE l
MOV BX,10
MOV CL,0
m:MOV DX,0
DIV BX
PUSH DX
INC CL
CMP AX,0
JNE m
n:POP DX
ADD DX,30h
MOV AH,02h
INT 21h
DEC CL
CMP CL,0
JNE n
MOV DX,OFFSET b
MOV AH,09h
INT 21h
END start
```



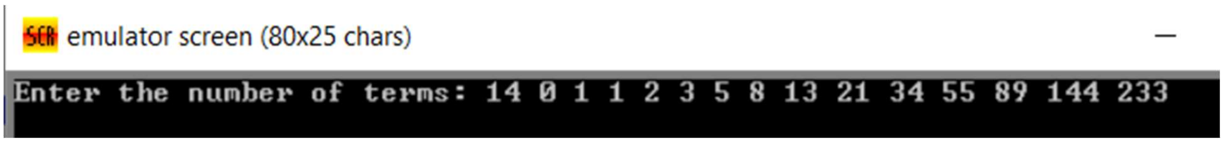
```
Enter the number: 5120Factorial of the number
```

7. Write a program to print the Fibonacci series up to 233

```
.data
a DB "Enter the number of terms: $"
```

```
.code
start:
MOV AX,@data
MOV DS,AX
MOV DX,OFFSET a
MOV AH,09h
INT 21h
MOV AH,01h
INT 21h
MOV BH,AL
MOV AH,01h
INT 21h
MOV AH,BH
SUB AX,3030h
AAD
MOV BH,AL
MOV DL,32
MOV AH,02h
INT 21h
MOV DL,48
MOV AH,02h
INT 21h
MOV DL,32
MOV AH,02h
INT 21h
MOV DL,49
MOV AH,02h
INT 21h
DEC BH
DEC BH
MOV CX,01
MOV SI,00
```

```
l:MOV DI,CX
ADD CX,SI
MOV SI,DI
MOV AX,CX
MOV DI,10
MOV BL,0
m:MOV DX,0
DIV DI
ADD DX,48
PUSH DX
INC BL
CMP AX,0
JNE m
MOV DL,32
MOV AH,02h
INT 21h
p:POP DX
MOV AH,02h
INT 21h
DEC BL
CMP BL,0
JNE p
DEC BH
CMP BH,0
JNE l
END start
```



```
Enter the number of terms: 14 0 1 1 2 3 5 8 13 21 34 55 89 144 233
```