NATIONAL INSTITUTE OFTECHNOLOGY KURUKSHETRA

Microprocessors

Lab Programs

Submitted To:
Manju Mam
Assistant Professor
Department of Computer Engineering
NIT KURUKSHETRA

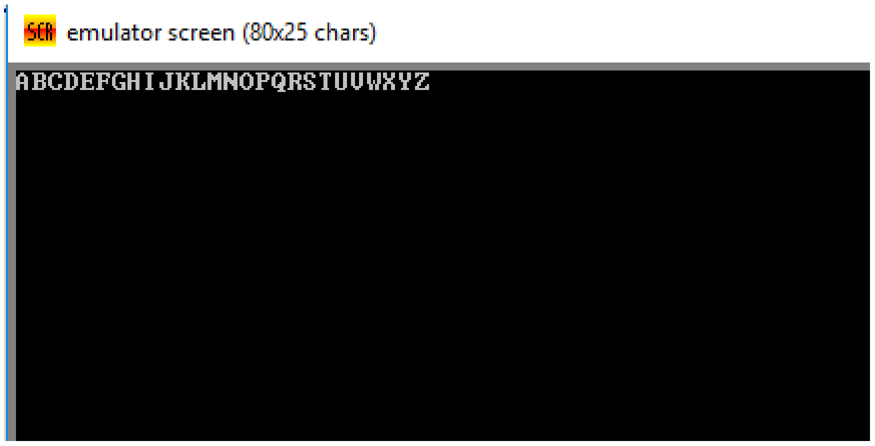
Submitted By:
Ginni Garg
11610559
B.Tech 2nd year (CO-4)
(COT-214)

INDEX

/*WALP to print the Capital A-Z*/

```
.MODEL SMALL
.DATA
.CODE
START:
MOV AX,@DATA
MOV DS,AX
MOV DX,65
MOV CX,26
L:
MOV AH,02H
INT 21H
INC DX
LOOP L
END START
```

OUTPUT::

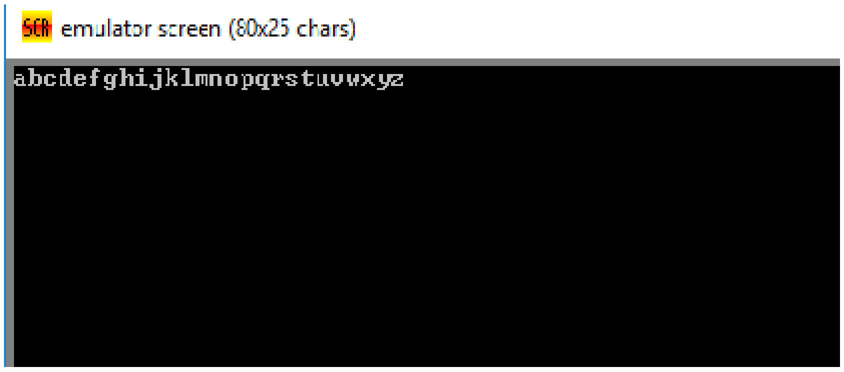


```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

/*WALP to print the Small a-z*/

```
.MODEL SMALL
.DATA
.CODE
START:
MOV AX,@DATA
MOV DS,AX
MOV DX,97
MOV CX,26
L:
MOV AH,02H
INT 21H
INC DX
LOOP L
END START
```

OUTPUT::

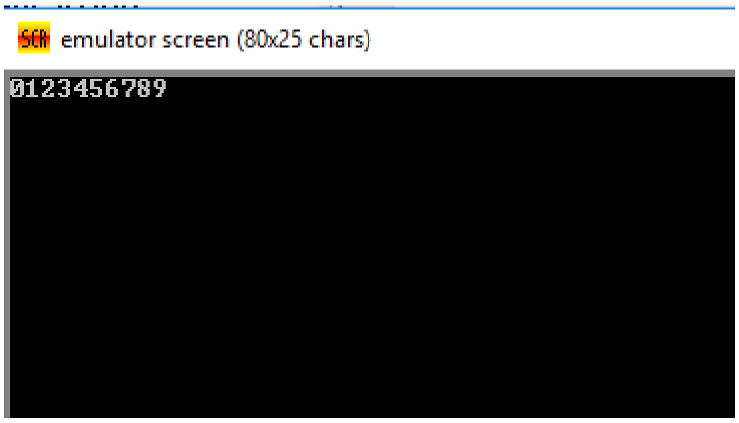

emulator screen (80x25 chars)

```
abcdefghijklmnopqrstuvwxyz
```

/*WALP to print the 0-9 Numbers*/

```
.MODEL SMALL
.DATA
.CODE
START:
MOV AX,@DATA
MOV DS,AX
MOV DX,48
MOV CX,10
L:
MOV AH,02H
INT 21H
INC DX
LOOP L
END START
```

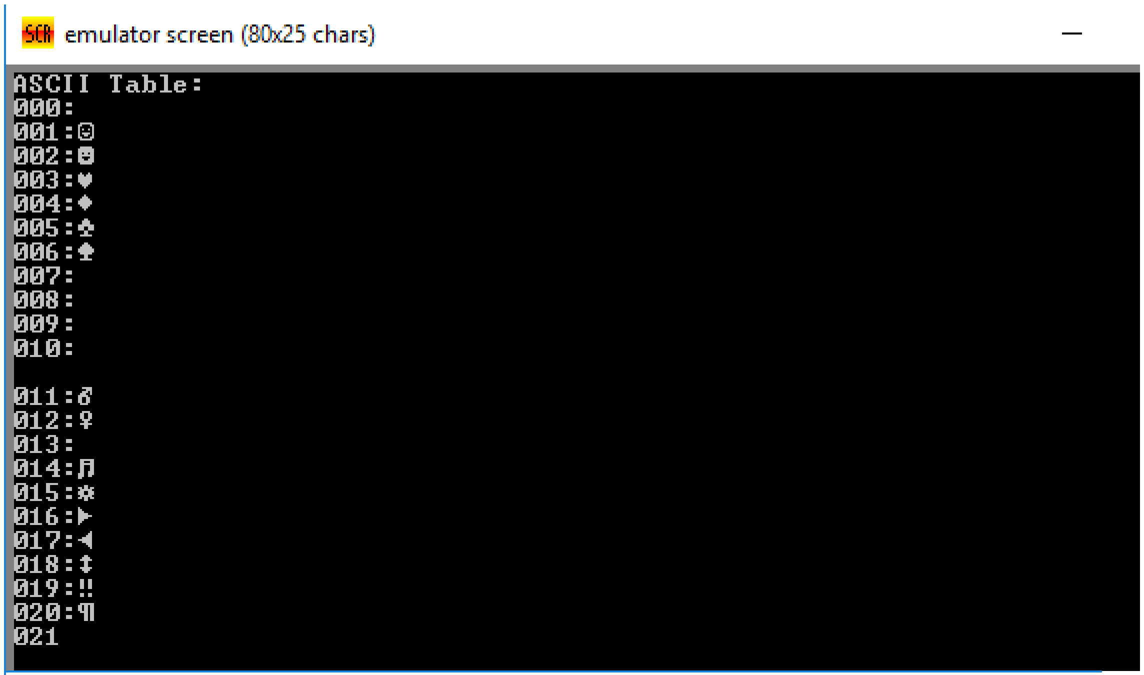
OUTPUT::


emulator screen (80x25 chars)

```
0123456789
```

```
/*Program to Print the ASCII Table*/
.model small
.data
print db "ASCII Table:$"
newline db 0ah,0dh,"$"
value db 5 dup('$')
.code
start:
mov ax,@data
mov ds,ax
mov dx,offset print
mov ah,09h
int 21h
mov si,offset value
mov cx,128
mov bx,0
again:
mov dx,offset newline
mov ah,09h
int 21h
mov ax,bx
mov dl,10
div dl
mov [si],ah
add [si],30h
mov ah,00h
div dl
inc si
mov [si],ah
add [si],30h
inc si
mov [si],al
add [si],30h
mov dx,[si]
mov ah,02h
int 21h
dec si
mov dx,[si]
mov ah,02h
int 21h
dec si
mov dx,[si]
mov ah,02h
int 21h
mov dx,':'
mov ah,02h
int 21h
mov dx,bx
mov ah,02h
int 21h
inc bx
loop again
end start
```

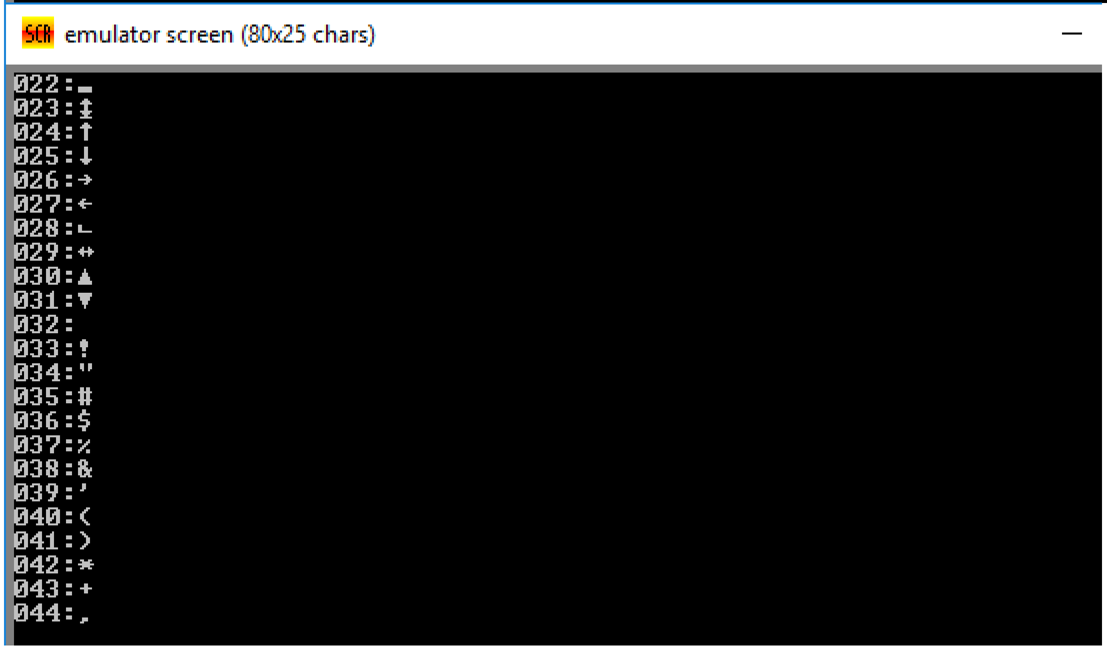
OUTPUT::



emulator screen (80x25 chars)

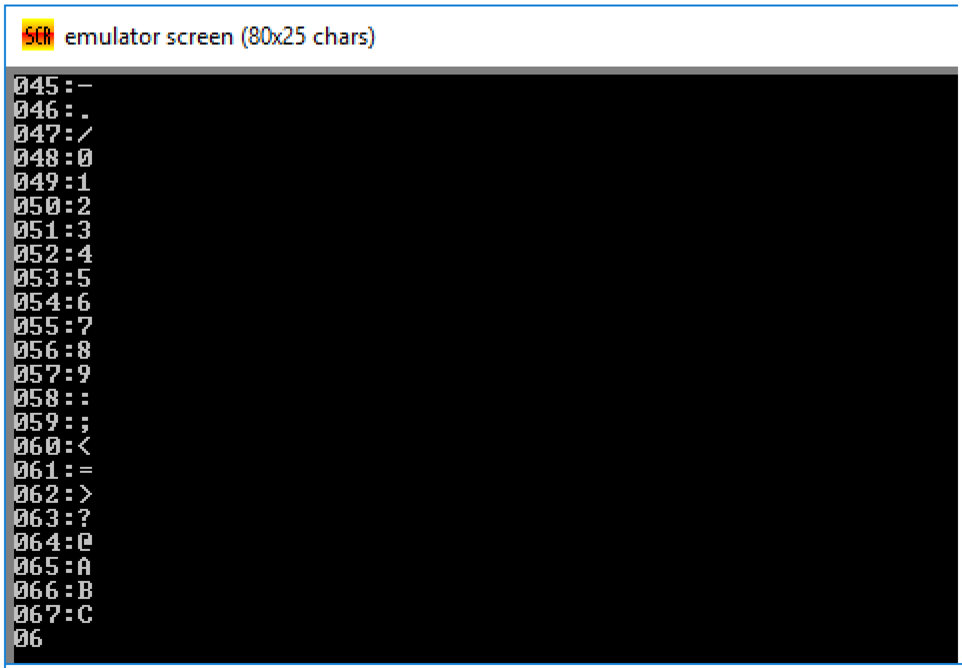
```
ASCII Table:
000:
001:☺
002:☻
003:♥
004:♦
005:♣
006:♠
007:
008:
009:
010:

011:♂
012:♀
013:
014:♫
015:☼
016:►
017:◄
018:↕
019:‼
020:¶
021
```

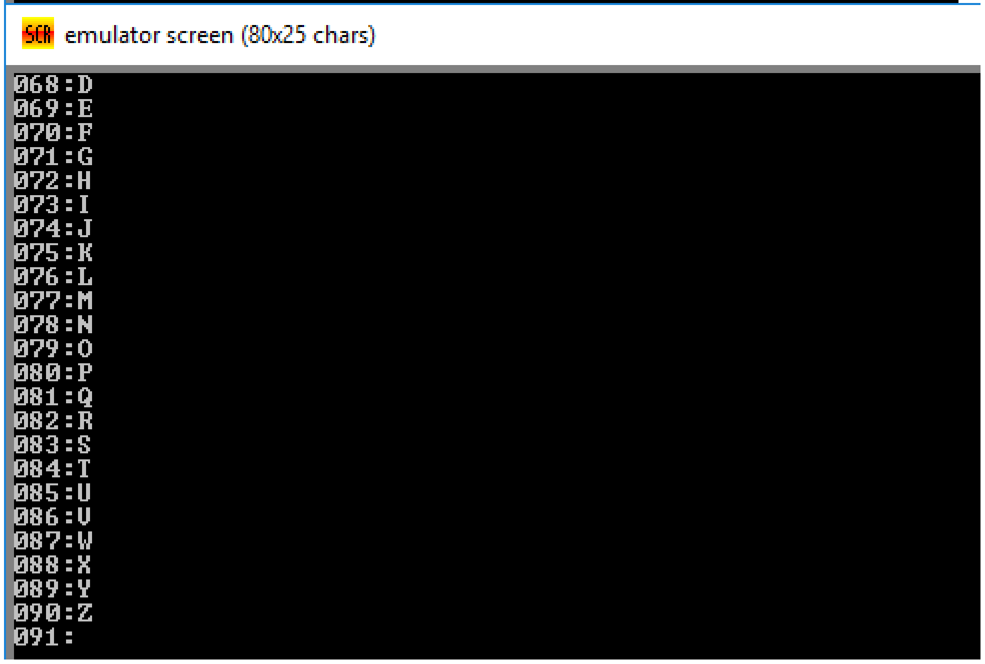


emulator screen (80x25 chars)

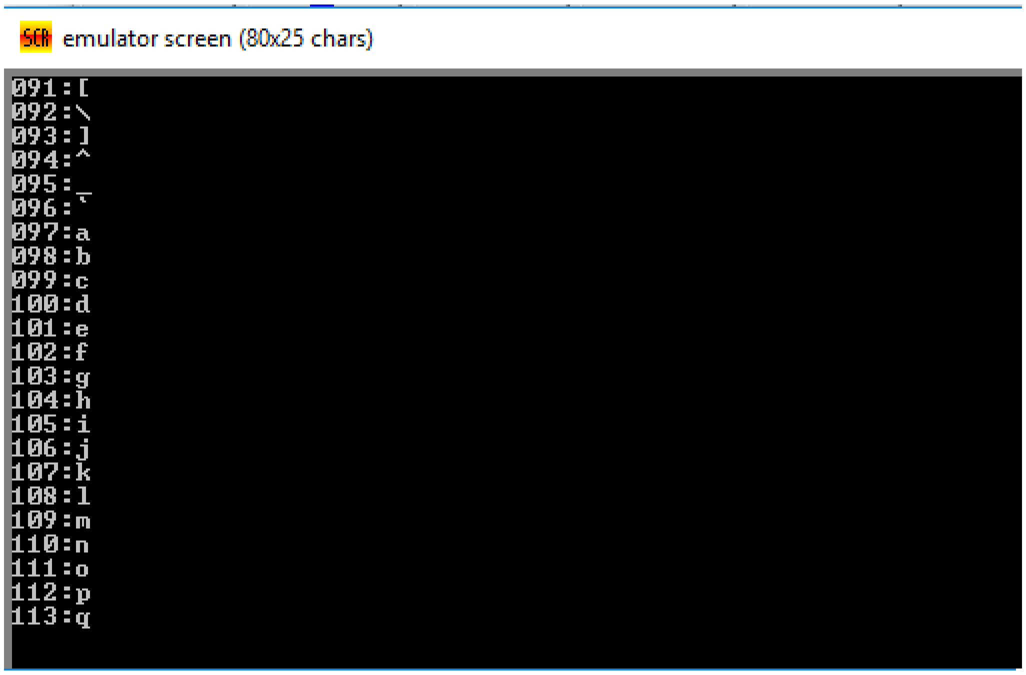
```
022:▬
023:↨
024:↑
025:↓
026:→
027:←
028:∟
029:↔
030:▲
031:▼
032:
033:!
034:"
035:#
036:$
037:%
038:&
039:'
040:(
041:)
042:*
043:+
044:,
```


emulator screen (80x25 chars)

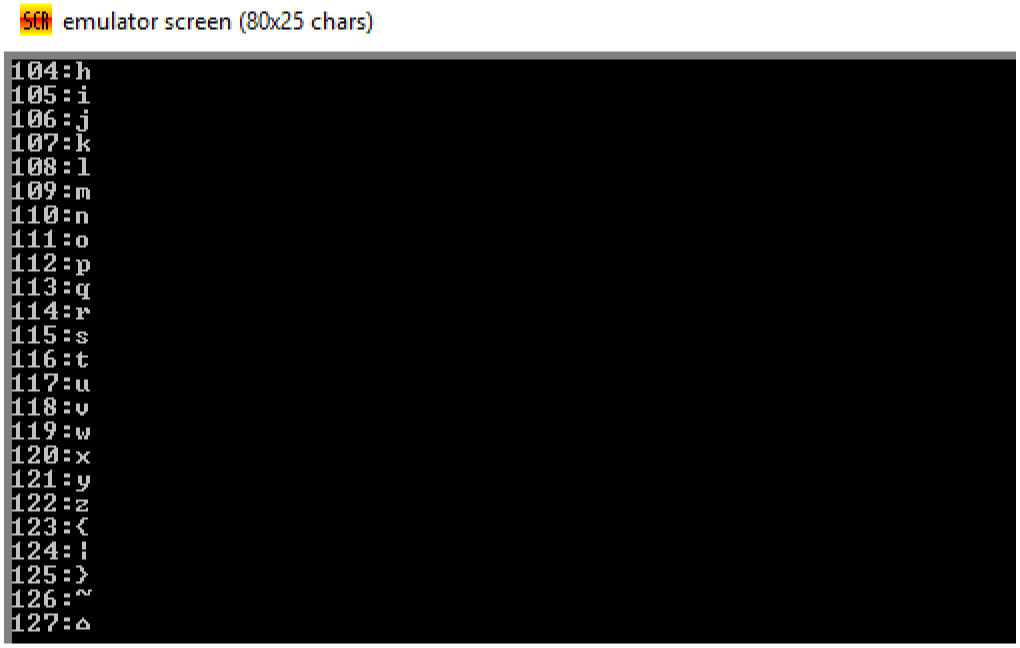
```
045:-
046:.
047:/
048:0
049:1
050:2
051:3
052:4
053:5
054:6
055:7
056:8
057:9
058::
059:;
060:<
061:=
062:>
063:?
064:@
065:A
066:B
067:C
06
```


emulator screen (80x25 chars)

```
068:D
069:E
070:F
071:G
072:H
073:I
074:J
075:K
076:L
077:M
078:N
079:O
080:P
081:Q
082:R
083:S
084:T
085:U
086:V
087:W
088:X
089:Y
090:Z
091:
```


emulator screen (80x25 chars)

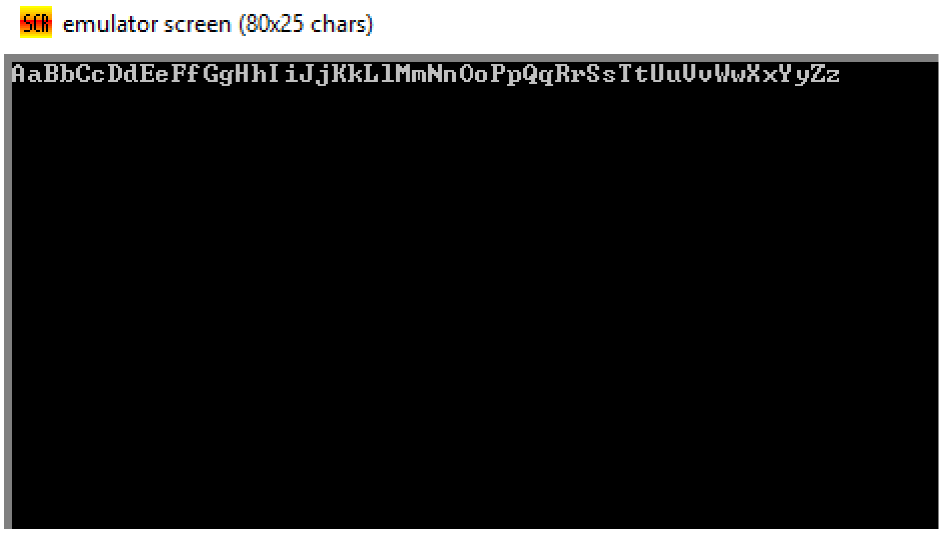
```
091:[
092:\
093:]
094:^
095:_
096:`
097:a
098:b
099:c
100:d
101:e
102:f
103:g
104:h
105:i
106:j
107:k
108:l
109:m
110:n
111:o
112:p
113:q
```


emulator screen (80x25 chars)

```
104:h
105:i
106:j
107:k
108:l
109:m
110:n
111:o
112:p
113:q
114:r
115:s
116:t
117:u
118:v
119:w
120:x
121:y
122:z
123:{
124:|
125:}
126:~
127:⌂
```

```
/*WALP to print the pattern AaBb…..*/
.MODEL SMALL
.DATA
.CODE
START:
MOV AX,@DATA
MOV DS,AX
MOV DX,65
MOV CX,26
L:
MOV AH,02H
INT 21H
ADD DX,32
MOV AH,02H
INT 21H
SUB DX,32
INC DX
LOOP L
END START
```

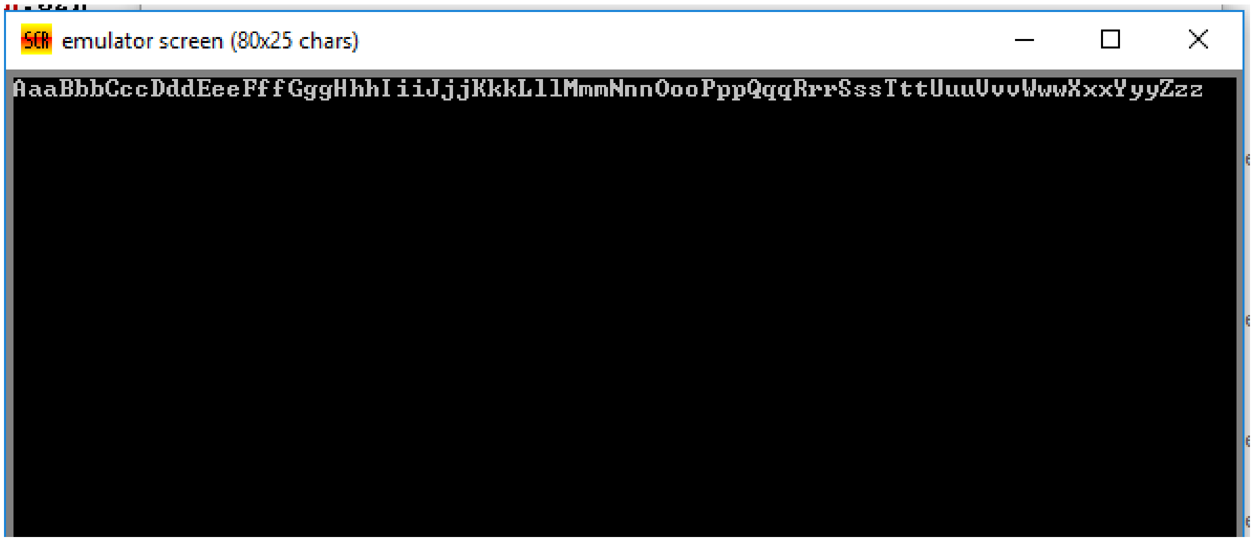
OUTPUT::


emulator screen (80x25 chars)

```
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
```

```
/*WALP to print the pattern AaaBbb.....*/
.MODEL SMALL .DATA
.CODE
START:
MOV AX,@DATA
MOV DS,AX
MOV DX,65
MOV CX,26
L:
MOV AH,02H
INT 21H
ADD DX,32
MOV AH,02H
INT 21H
mov ah,02h
int 21h
SUB DX,32
INC DX
LOOP L
END START
```

OUTPUT::

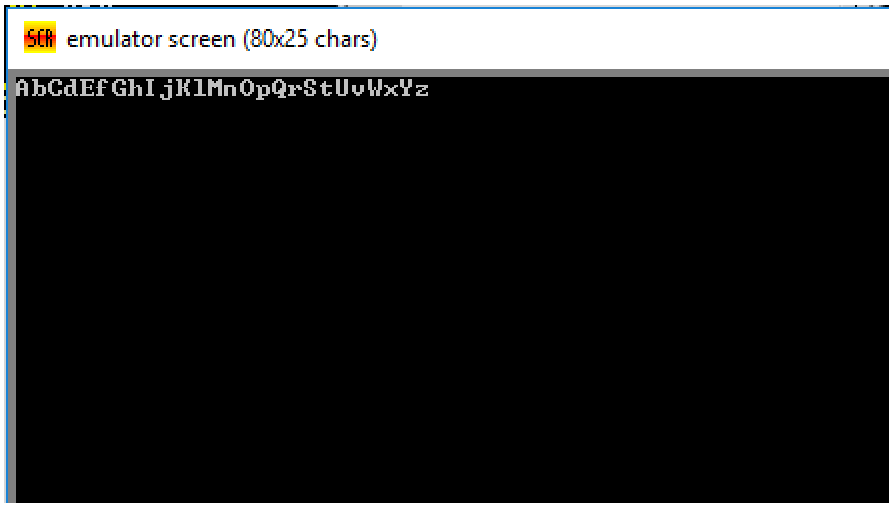


```
AaaBbbCccDddEeeFffGggHhhIiiJjjKkkLllMmmNnnOooPppQqqRrrSssTttUuuVvvWwwXxxYyyZzz
```

/*WALP to print the pattern AbCd....*/

```
.MODEL SMALL
.DATA
.CODE
START:
MOV AX,@DATA
MOV DS,AX
MOV DX,65
MOV CX,13
L:
MOV AH,02H
INT 21H
ADD DX,33
MOV AH,02H
INT 21H
SUB DX,32
INC DX
LOOP L
END START
```

OUTPUT::

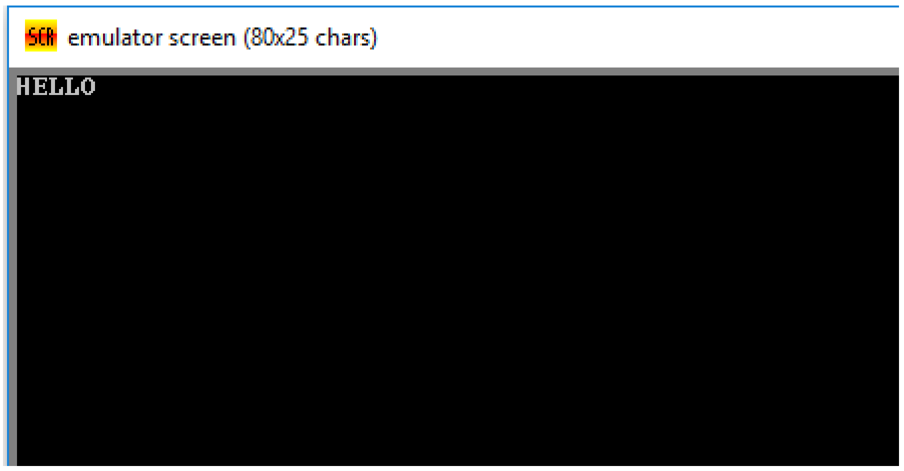


```
AbCdEfGhIjKlMnOpQrStUvWxYz
```

/*WALP to print the string using interrupt 09h*/

```
.MODEL SMALL
.DATA
STRING DW "HELLO$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA DX,STRING
MOV AH,09H
INT 21H
END START
```

OUTPUT::

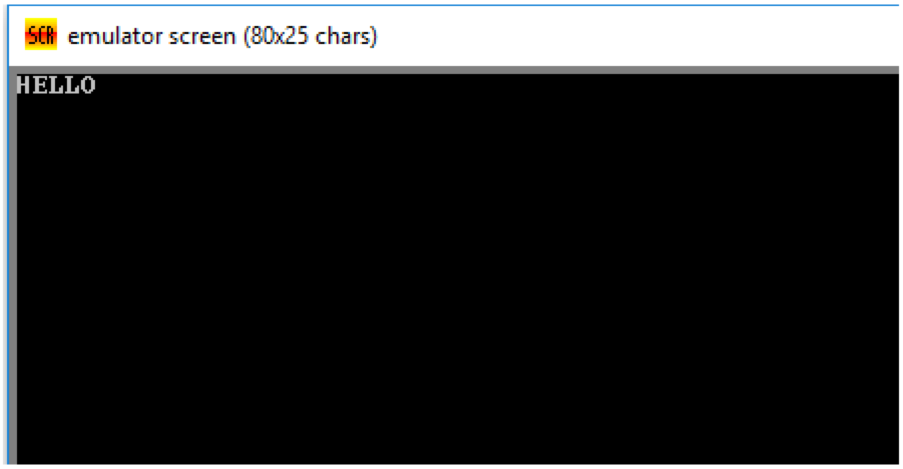


```
HELLO
```

/*WALP to print the string characterwise for the 8-bit (DB)*/

```
.MODEL SMALL
.DATA
STRING DB "HELLO$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA SI,STRING
L:
MOV DX,[SI]
MOV AH,02H
INT 21H
INC SI
CMP [SI],'$'
JNZ L
END START
```

OUTPUT::

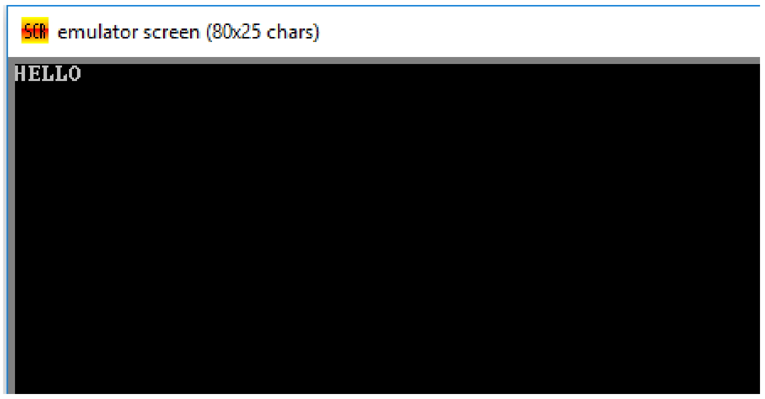

emulator screen (80x25 chars)

```
HELLO
```

/*WALP to print the string characterwise for the 16-bit (DW)*/

```
.MODEL SMALL
.DATA
STRING DW "HELLO$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA SI,STRING
L:
MOV DX,[SI]
MOV AH,02H
INT 21H
INC SI
CMP [SI],'$'
JNZ L
END START
```

OUTPUT::

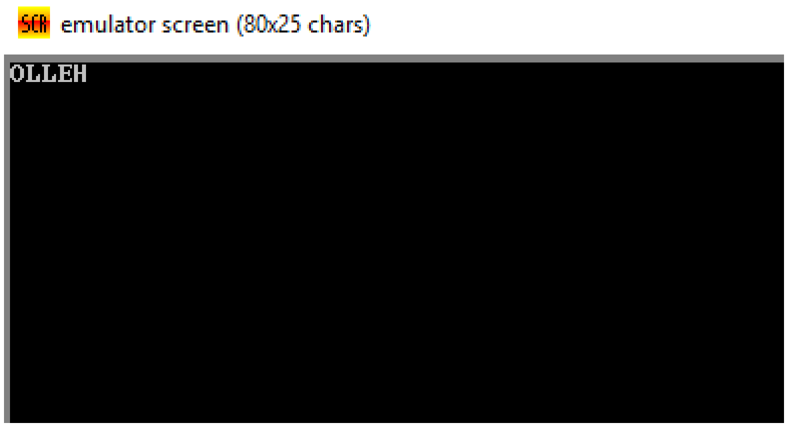

emulator screen (80x25 chars)

```
HELLO
```

/*WALP to reverse print the string characterwise for the 8-bit (DB)*/

```
.MODEL SMALL
.DATA
STRING DB "HELLO$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA SI,STRING
MOV CX,0
L:
INC SI
INC CX
CMP [SI],'$'
JNZ L
DEC SI
K:
MOV DX,[SI]
MOV AH,02H
INT 21H
DEC SI
LOOP K
END START
```

OUTPUT::

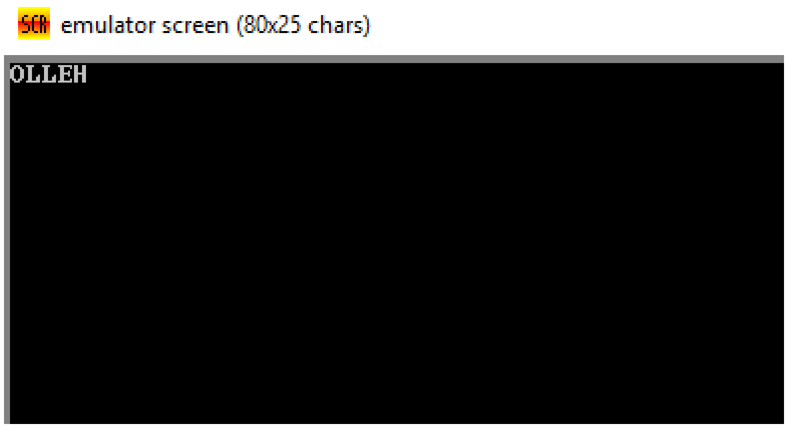

emulator screen (80x25 chars)

```
OLLEH
```

/*WALP to reverse print the string characterwise for the 16-bit (DW)*/

```
.MODEL SMALL
.DATA
STRING DW "HELLO$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA SI,STRING
MOV CX,0
L:
INC SI
INC CX
CMP [SI],'$'
JNZ L
DEC SI
K:
MOV DX,[SI]
MOV AH,02H
INT 21H
DEC SI
LOOP K
END START
```

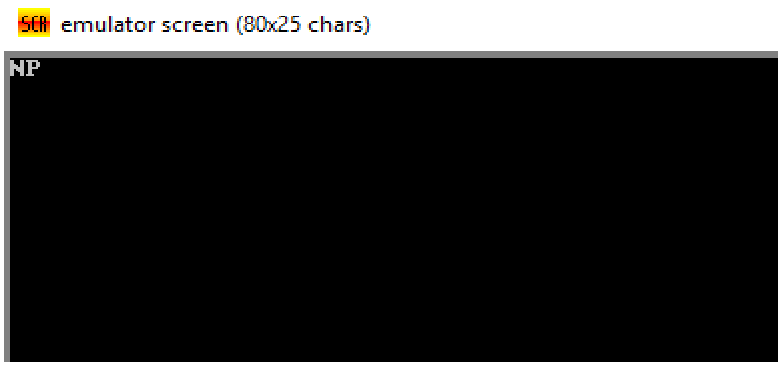
OUTPUT::



```
OLLEH
```

```
/*WALP to check whether a string is a palindrome  or not for 8-bit*/
.MODEL SMALL
.DATA
A DB "TUTs$"
B DB "PAL$"
C DB "NP$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA SI,A
LEA DI,A
MOV CL,0
L1:
INC CL
INC SI
CMP [SI],'$'
JNE L1
DEC CL
L2:
DEC SI
MOV AL,[SI]
MOV BL,[DI]
CMP AL,BL
JNZ L3
INC DI
LOOP L2
LEA DX,B
MOV AH,09H
INT 21H
HLT
L3:
LEA DX,C
MOV AH,09H
INT 21H
HLT
END START
```

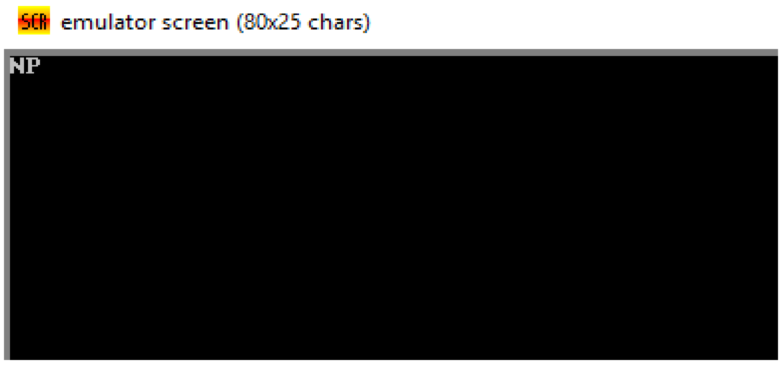
OUTPUT::



```
NP
```

```
/*WALP to check whether a string is a palindrome  or not for 8-bit*/
.MODEL SMALL
.DATA
A DW "TUTs$"
B DW "PAL$"
C DW "NP$"
.CODE
START:
MOV AX,@DATA
MOV DS,AX
LEA SI,A
LEA DI,A
MOV CL,0
L1:
INC CL
INC SI
CMP [SI],'$'
JNE L1
DEC CL
L2:
DEC SI
MOV AL,[SI]
MOV BL,[DI]
CMP AL,BL
JNZ L3
INC DI
LOOP L2
LEA DX,B
MOV AH,09H
INT 21H
HLT
L3:
LEA DX,C
MOV AH,09H
INT 21H
HLT
END START
```

OUTPUT::

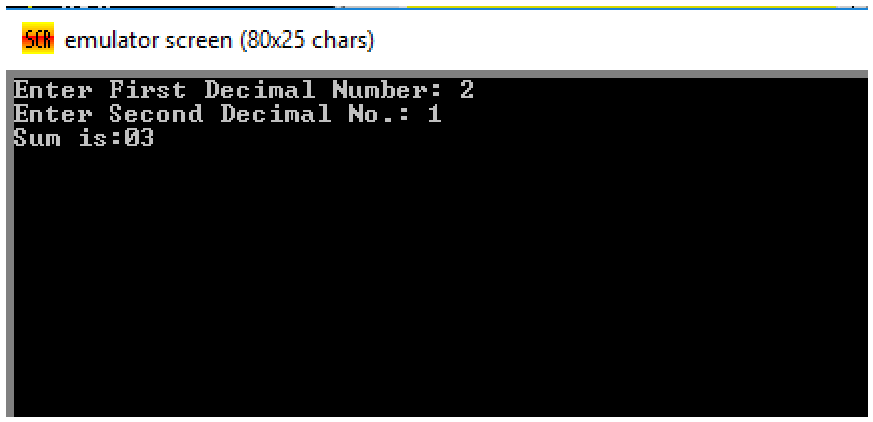

emulator screen (80x25 chars)

```
NP
```

/*Program for the addition of the single digit numbers*/

```
.model small
.data
a db "Enter First Decimal Number : $"
b db ,0dh,0ah,"Enter Second Decimal No. : $"
c db ,0dh,0ah,"Sum is:$"
.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
```

OUTPUT::



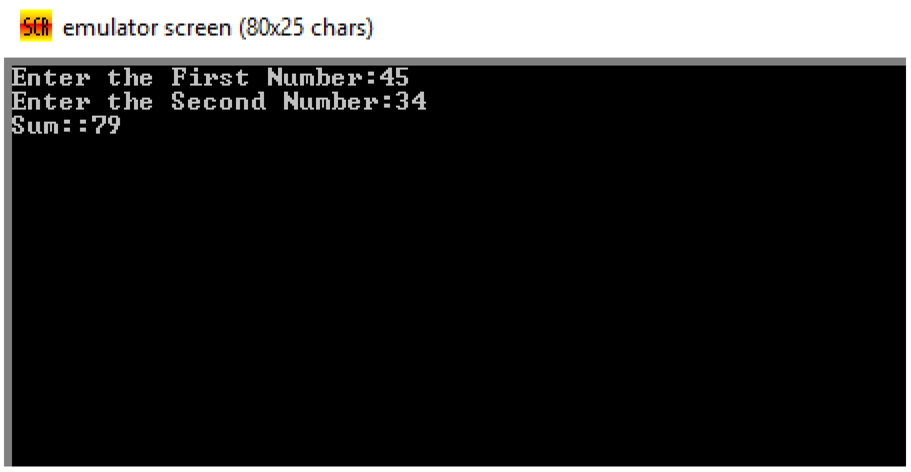
```
Enter First Decimal Number: 2
Enter Second Decimal No.: 1
Sum is:03
```

/*Program for the addition of the Double digit numbers*/

```
.model small
.data
enter1 db "Enter the First Number:$"
enter2 db "Enter the Second Number:$"
result db "Sum::$"
newline db 0ah,0dh,"$"
d1 db 5 dup('$')
d2 db 5 dup('$')
.code
start:
mov ax,@data
mov ds,ax
mov dx,offset enter1
mov ah,09h
int 21h
mov si,offset d1
mov di,offset d2
mov ah,01h
int 21h
sub ax,48
mov [si],al
inc si
mov ah,01h
int 21h
sub ax,48
mov [si],al
mov dx,offset newline
mov ah,09h
int 21h
mov dx,offset enter2
mov ah,09h
```

```
int 21h
mov ah,01h
int 21h
sub ax,48
mov [di],al
inc di
mov ah,01h
int 21h
sub ax,48
mov [di],al
mov dl,[di]
add [si],dl
add [si],48
dec si
dec di
mov dx,[di]
add [si],dl
add [si],48
mov dx,offset newline
mov ah,09h
int 21h
mov dx,offset result
mov ah,09h
int 21h
mov dx,offset d1
mov ah,09h
int 21h
end start
```

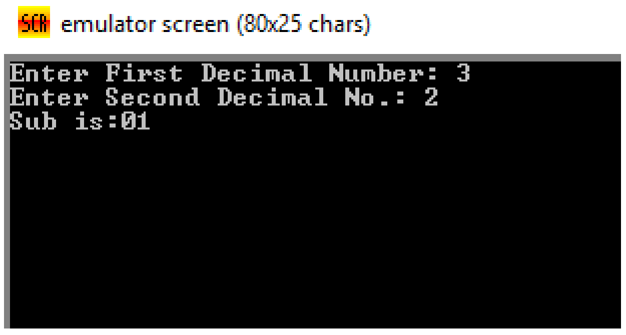
OUTPUT::



```
Enter the First Number:45
Enter the Second Number:34
Sum::79
```

```
/*Program to Subtract the single digit numbers*/
.model small
.data
a db "Enter First Decimal Number : $"
b db ,0dh,0ah,"Enter Second Decimal No. : $"
c db ,0dh,0ah,"Sub is:$"
.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
mov cl,al
mov al,bl
sub al,cl
mov ah,0
aas
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
```

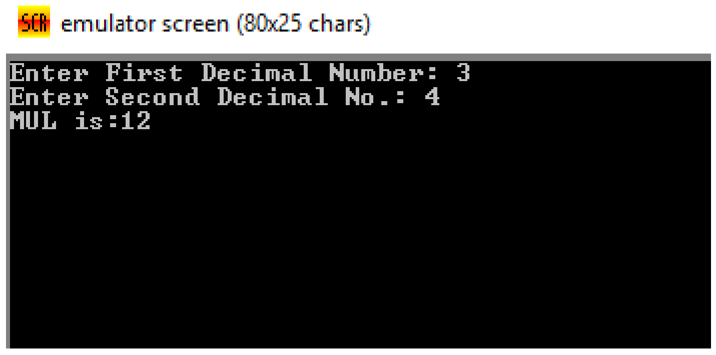
OUTPUT::



```
Enter First Decimal Number: 3
Enter Second Decimal No.: 2
Sub is:01
```

```
/*Program for the MUL of the single digit numbers*/
.model small
.data
a db "Enter First Decimal Number : $"
b db ,0dh,0ah,"Enter Second Decimal No. : $"
c db ,0dh,0ah,"MUL is:$"
.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 ah,0
mul bl
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
```

OUTPUT::



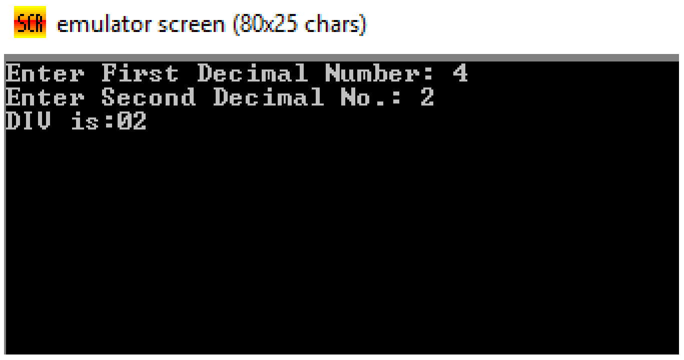
```
Enter First Decimal Number: 3
Enter Second Decimal No.: 4
MUL is:12
```

/*Program for the DIV of the single digit numbers*/

```
.model small
.data
a db "Enter First Decimal Number : $"
b db ,0dh,0ah,"Enter Second Decimal No. : $"
c db ,0dh,0ah,"DIV is:$"
.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
```

OUTPUT::

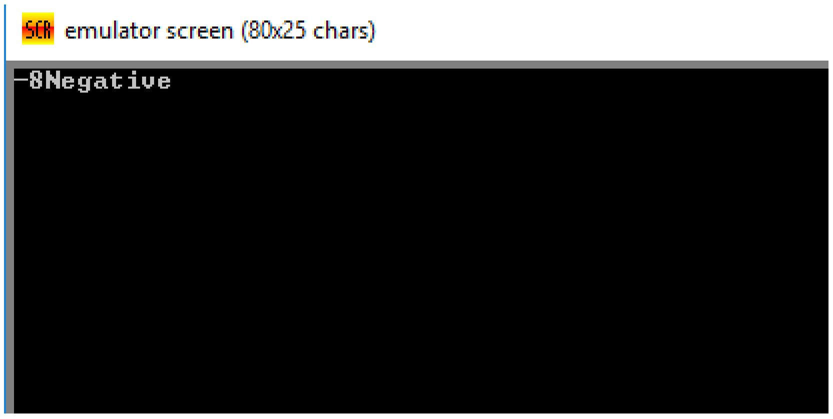


```
Enter First Decimal Number: 4
Enter Second Decimal No.: 2
DIV is:02
```

/*Program to check whether Number is Positive or Negative*/

```
.model small
.data
a db "Positive$"
b db "Negative$"
.code
start:
mov ax,@data
mov ds,ax
mov ah,1h
int 21h
mov bl,al
mov ah,1h
int 21h
cmp bl,'-'
jne pos
mov dx,offset b
mov ah,09h
int 21h
hlt
pos:
mov dx,offset a
mov ah,09h
int 21h
hlt
end start
```

OUTPUT::

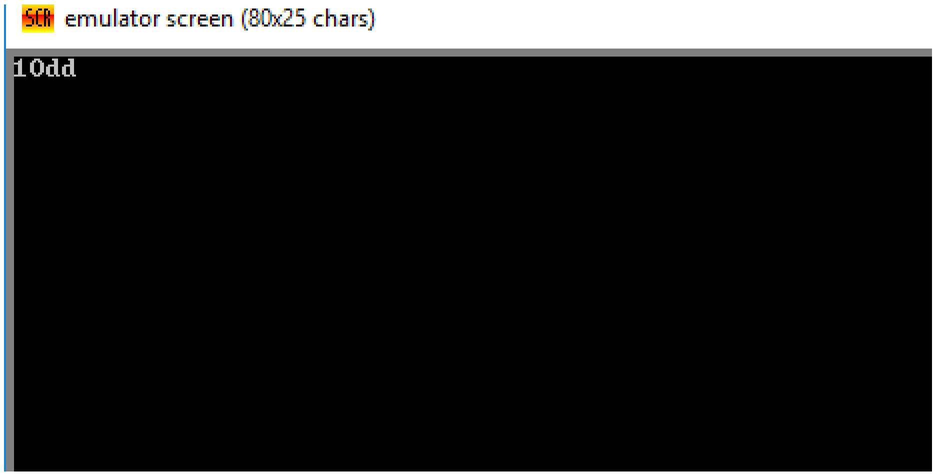


```
-8Negative
```

/*Program to Check whether a number is even or odd*/

```
.model small
.data
a dw "Even$"
b dw "Odd$"
.code
start:
mov ax,@data
mov ds,ax
mov ah,00h
mov ah,1h
int 21h
sub al,30h
mov ah,00h
mov bx,2
div bx
add dl,30h
cmp dl,'0'
jnz odd
mov dx,offset a
mov ah,09h
int 21h
hlt
odd:
mov dx,offset b
mov ah,09h
int 21h
hlt
end start
```

OUTPUT::



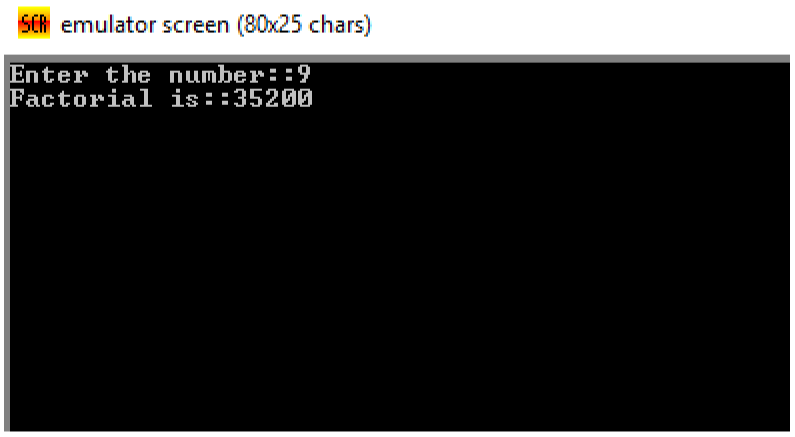
```
1Odd
```

/*Program to find the Factorial of a Number*/

```
.model small
.stack 100h
.data
a db "Enter the number::$"
b db ,0dh,0ah,"Factorial is ::$"
c db 100 dup('$')
.code
start:
mov ax,@data
mov ds,ax
mov di,offset c
mov dx,offset a
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov bl,al
mov ax,1
mov cx,bx
l:mul bx
dec bx
loop l
mov bx,10
mov cx,0
m:mov dx,0
div bx
add dx,30h
push dx
inc cx
```

```
cmp ax,0
jne m
n:pop dx
mov [di],dx
inc di
loop n
mov dx,offset b
mov ah,09h
int 21h
mov dx,offset c
mov ah,09h
int 21h
end start
```

OUTPUT::



emulator screen (80x25 chars)

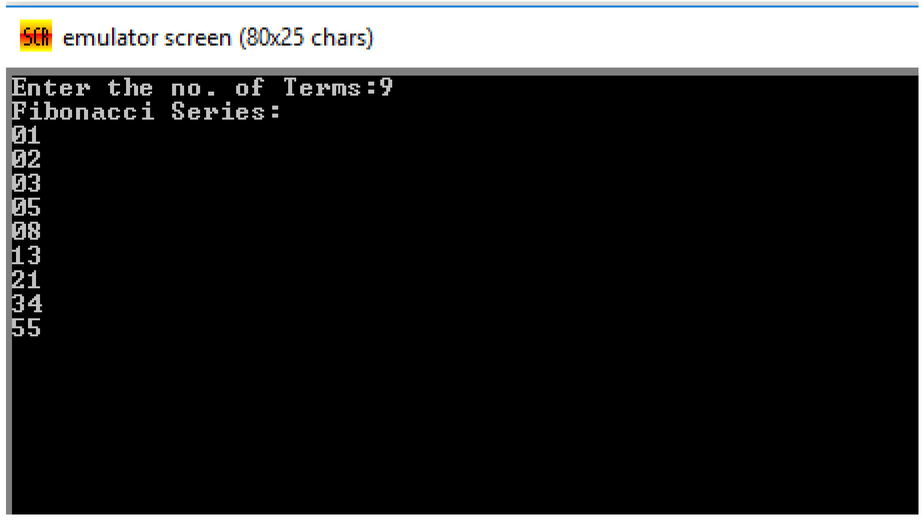
```
Enter the number::9
Factorial is::35200
```

/*Program to Print the Fibonacci Series*/

```
.model small
.data
print db "Enter the no. of Terms:$"
ls db "Fibonacci Series:$"
newline db 0ah,0dh,"$"
result db 100 dup('$')
.code
start:
mov ax,@data
mov ds,ax
mov dx,offset print
mov ah,09h
int 21h
mov si,offset result
mov ah,01h
int 21h
sub ax,48
mov ah,00h
mov cx,ax
```

```
cmp cx,0
je last
mov bl,0
mov bh,1
mov dx,offset newline
mov ah,09h
int 21h
mov dx,offset ls
mov ah,09h
int 21h
control:
mov dx,offset newline
mov ah,09h
int 21h
mov dl,bh
add bh,bl
mov bl,dl
mov ah,00h
mov al,bh
mov dx,10
div dl
add ax,3030h
mov [si],al
inc si
mov [si],ah
mov dx,offset result
mov ah,09h
int 21h
mov si,offset result
loop control
last:
end start
```

OUTPUT::


emulator screen (80x25 chars)

```
Enter the no. of Terms:9
Fibonacci Series:
01
02
03
05
08
13
21
34
55
```

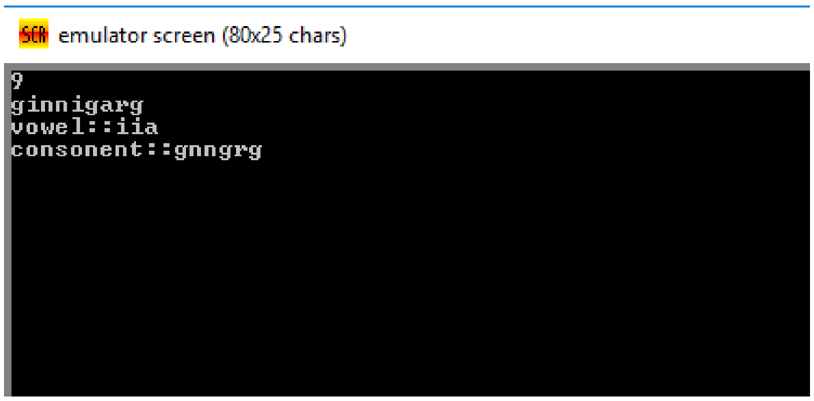
```
/*Program to print the Vowels and Consonants Distinctly from a String*/
.model small
.data
a db "vowel$"
b db "consonent$"
w db 100 dup('$')
d db 20 dup('$')
f db 20 dup('$')
q db ,0ah,0dh, "$"
.code
start:
mov ax,@data
mov ds,ax
mov di,offset w
mov ah,01h
int 21h
mov ah,00h
mov cx,ax
mov ch,00h
mov bx,cx
mov dx,offset q
mov ah,09h
int 21h
l:
mov ah,01h
int 21h
mov [di],al
inc di
dec cx
cmp cx,'0'
jnz l
mov si,offset d
mov di,offset w

dec di
mov cx,bx
mov ch,00h
mov bx,offset f
again:
inc di
mov al,[di]
cmp al,'a'
jnz e
mov [si],al
inc si
dec cx
cmp cx,'0'
```

```
jnz again
cmp cx,'-'
jnz last
e:
cmp al,'e'
jnz i
mov [si],al
inc si
dec cx
cmp cx,'0'
jnz again
cmp cx,'-'
jnz last
i:
cmp al,'i'
jnz o
mov [si],al
inc si
dec cx
cmp cx,'0'
jnz again
cmp cx,'-'
jnz last
o:
cmp al,'o'
jnz u
mov [si],al
inc si
dec cx
cmp cx,'0'
jnz again
cmp cx,'-'
jnz last
u:
cmp al,'u'
jnz c
mov [si],al
inc si
dec cx
cmp cx,'0'
jnz again
cmp cx,'-'
jnz last
c:
mov [bx],al
inc bl
dec cx
cmp cx,'0'
```

```
jnz again
last:
mov dx,offset q
mov ah,09h
int 21h
mov dx,offset d
mov ah,09h
int 21h
mov dx,offset q
mov ah,09h
int 21h
mov dx,offset f
mov ah,09h
int 21h
end start
```

OUTPUT::



```
9
ginnigarg
vowel::iia
consonent::gnngrg
```