Manas Pratim Biswas

IT-A1-025 **ASM Assignment-2**

---

1. **Write an 8086 assembly language program to calculate the GCD of two unsigned 16-bit numbers.**

```
.model small
.stack 100h
.data
num1 dw ? ;first number
num2 dw ? ;second number
count dw 0 ;printnum - digit counter
intg dw 0 ;printnum - I/O for inputnum and printnum
digt dw 0 ;printnum - extracted digit
var1 dw 0 ;printnum - reversed number
msg1 db 'Enter first number : $'
msg2 db 'Enter second number : $'
msg3 db 'Greatest common divisor (GCD) = $'
.code
main proc
mov ax,@data
mov ds,ax
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in num1
call inputnum
mov bx,intg
mov num1,bx
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;taking input in num2
call inputnum
mov bx,intg
mov num2,bx
;calculating GCD
call getgcd
;printing msg3
mov dx,offset msg3
mov ah,9
int 21h
;printing GCD
call printnum
;exit program
mov ah,4ch
int 21h
main endp
getgcd proc ;calculates GCD of two integers
mov bx,num1
mov var1,bx
cmp bx,num2
je endsolve
jl solve ;else swap num1 and num2
push num1
push num2
pop num1
pop num2
```

```
solve: ;here num1 < num2, always
mov ax,num2
mov dx,0
div var1
cmp dx,0
jne skip
mov ax,num1
mov dx,0
div var1
cmp dx,0
je endsolve
skip:
dec var1
jmp solve
endsolve:
mov bx,var1
mov intg,bx
ret
getgcd endp
inputnum proc ;inputs an integer
mov intg,0
iter1:
mov ah,1
int 21h
cmp al,13
je enditer1
sub al,48
mov ah,0
mov digt,ax
mov ax,intg
mov bx,10
```

```
mul bx
add ax,digt
mov intg,ax
jmp iter1
enditer1:
ret
inputnum endp
printnum proc ;prints an integer
mov var1,0
mov count,0
iter2:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
mov al,dl
mov digt,ax
mov ax,var1
mov bx,10
mul bx
add ax,digt
mov var1,ax
inc count
cmp intg,0
jle enditer2
jmp iter2
enditer2:
mov bx,var1
mov intg,bx
mov cx,count
iter3:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
add dl,48
mov ah,2
int 21h
loop iter3
ret
printnum endp
end main
```

2. **Write an 8086 assembly language program to generate all prime numbers for a given range. For e.g. if anyone inputs the range 10 to 20, the program will return 11, 13, 17, 19.**

```
.model small
.stack 100h
.data
check db 'T' ;true or false
count dw 0 ;printnum - digit counter
intg dw 0 ;printnum - I/O for inputnum and printnum
digt dw 0 ;printnum - extracted digit
var1 dw 0 ;printnum - reversed number
num dw 0 ;number to be checked
```

```
rnum db 0 ;starting number
enum db 0 ;ending number
countprime db 0 ;number of primes found
newl db 10,13,'$'
msg1 db 'Enter starting number of range : $'
msg2 db 'Enter ending number of range : $'
msg3 db 'The prime numbers in the range are
:',10,13,'$'
msg4 db 'There is no prime number in the range.$'
.code
main proc
mov ax,@data
mov ds,ax
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in num1 or starting number
call inputnum
mov num1,al
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;taking input in num2 or ending number
call inputnum
mov num2,al
;printing prime numbers in the range
mov bh,0
loop1:
mov check,'T'
mov bl,num1
mov num,bx
call isprime
cmp check,'T'
jne skiploop
cmp countprime,0
jne displaycomma
;printing msg3
mov dx,offset msg3
mov ah,9
int 21h
jmp displaynum
displaycomma:
mov dl,','
mov ah,2
int 21h
displaynum:
call printnum
inc countprime
skiploop:
inc num1
mov bl,num1
mov cl,num2
cmp bl,cl
jg exitloop
jmp loop1
exitloop:
cmp countprime,0
jne printdot
;printing msg4
mov dx,offset msg4
mov ah,9
int 21h
jmp endprog
printdot:
mov dl,'.'
mov ah,2
int 21h
;exit program
endprog:
mov ah,4ch
int 21h
main endp
isprime proc ;checks if a number is prime or not
cmp num,1
je notprime
cmp num,2
je endproc
mov bh,0
mov bl,2
l1:
mov ax,num
div bl
cmp ah,0
je notprime
inc bl
cmp bx,num
jge endproc
jmp l1
notprime:
mov check,'F'
endproc:
```

```
ret
isprime endp
inputnum proc ;inputs an integer
mov intg,0
iter1:
mov ah,1
int 21h
cmp al,13
je enditer1
sub al,48
mov ah,0
mov digt,ax
mov ax,intg
mov bx,10
mul bx
add ax,digt
mov intg,ax
jmp iter1
enditer1:
mov ax,intg
mov ah,0
ret
inputnum endp
printnum proc ;prints an integer
mov bx,num
mov intg,bx
mov var1,0
mov count,0
iter2:
mov ax,intg
mov bl,10
```

```
div bl
mov dl,ah
mov ah,0
mov intg,ax
mov al,dl
mov digt,ax
mov ax,var1
mov bx,10
mul bx
add ax,digt
mov var1,ax
inc count
cmp intg,0
jle enditer2
jmp iter2
enditer2:
mov bx,var1
mov intg,bx
mov cx,count
iter3:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
add dl,48
mov ah,2
int 21h
loop iter3
ret
```

```
printnum endp
end main
```

3. Write an 8086 alp to sort in ascending order using Insertion Sort algorithm, a given set of 16-bit unsigned numbers in memory.

```
.model small
.stack 100h
.data
count dw 0 ;digit counter
intg dw 0 ;I/O for inputnum and printnum
digt dw 0 ;extracted digit
var1 dw 0 ;reversed number
n dw 8 ;number of entries
cc dw 0 ;loop counter
i dw 0 ;inner loop control of insertion sort
j dw 0 ;outer loop control of insertion sort
key dw 0 ;key of insertion sort
m dw 0 ;index counter
array dw 20 dup(0) ;array
ind dw 0 ;array index
val dw 0 ;array value
temp dw 0;temporary variable
htab db ' $' ;horizontal tab
newl db 10,13,'$' ;new line
msg1 db 'Enter the number of entries : $'
msg2 db 'Enter $'
msg3 db ' unsigned numbers :',10,13,'$'
msg4 db 'Original array : $'
msg5 db 'Sorted array : $'
.code
main proc
```

```
mov ax,@data
mov ds,ax
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in n
call inputnum
mov bx,intg
mov n,bx
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;printing n
mov bx,n
mov intg,bx
call printnum
;printing msg3
mov dx,offset msg3
mov ah,9
int 21h
;taking inputs in array
mov cc,0
mov si,offset array
loop1:
call inputnum
mov dx,intg
mov [si],dx
add si,2
inc cc
```

```
mov bx,cc
cmp bx,n
jl loop1
;printing msg4
mov dx,offset msg4
mov ah,9
int 21h
;printing original array
mov cc,0
mov si,offset array
loop2:
mov dx,[si]
mov intg,dx
push si
call printnum
mov dx,offset htab
mov ah,9
int 21h
pop si
add si,2
inc cc
mov bx,cc
cmp bx,n
jl loop2
;printing newl
mov dx,offset newl
mov ah,9
int 21h
;performing insertion sort
mov j,1
loop3:
```

```
mov bx,j
mov i,bx
dec i
mov ind,bx
call getvalue
mov bx,val
mov key,bx
loop4:
cmp i,0
jl exitloop4
mov bx,i
mov ind,bx
call getvalue
mov bx,val
cmp bx,key
jle exitloop4
mov bx,i
inc bx
mov ind,bx
call setvalue
dec i
jmp loop4
exitloop4:
mov bx,key
mov val,bx
mov bx,i
inc bx
mov ind,bx
call setvalue
inc j
mov bx,j
```

```
cmp bx,n
jl loop3
;printing msg5
mov dx,offset msg5
mov ah,9
int 21h
;printing sorted array
mov cc,0
mov si,offset array
loop9:
mov dx,[si]
mov intg,dx
push si
call printnum
mov dx,offset htab
mov ah,9
int 21h
pop si
add si,2
inc cc
mov bx,cc
cmp bx,n
jl loop9
;exit program
mov ah,4ch
int 21h
main endp
getvalue proc ;val=array[ind]
mov m,0
mov cx,n
mov si,offset array
```

```
loopget:
mov bx,ind
cmp bx,m ;checking if ind==m
jne skipget
mov bx,[si]
mov val,bx
mov cx,0
jmp exitget
skipget:
add si,2
inc m
loop loopget
exitget:
ret
getvalue endp
setvalue proc ;array[ind]=val
mov m,0
mov cx,n
mov si,offset array
loopset:
mov bx,ind
cmp bx,m ;checking if ind==m
jne skipset
mov bx,val
mov [si],bx
mov cx,0
jmp exitset
skipset:
add si,2
inc m
loop loopset
```

```
exitset:
ret
setvalue endp
inputnum proc ;inputs an integer
mov intg,0
iter1:
mov ah,1
int 21h
cmp al,13
je enditer1
sub al,48
mov ah,0
mov digt,ax
mov ax,intg
mov bx,10
mul bx
add ax,digt
mov intg,ax
jmp iter1
enditer1:
ret
inputnum endp
printnum proc ;prints an integer
mov var1,0
mov count,0
iter2:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
mov al,dl
mov digt,ax
mov ax,var1
mov bx,10
mul bx
add ax,digt
mov var1,ax
inc count
cmp intg,0
jle enditer2
jmp iter2
enditer2:
mov bx,var1
mov intg,bx
mov cx,count
iter3:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
add dl,48
mov ah,2
int 21h
loop iter3
ret
printnum endp
end main
```

```
push si
call printnum
mov dx,offset htab
mov ah,9
int 21h
pop si
add si,2
inc cc
mov bx,cc
cmp bx,n
jl loop2
;printing newl
mov dx,offset newl
mov ah,9
int 21h
;performing insertion sort
mov j,1
loop3:
mov bx,j
mov i,bx
dec i
mov ind,bx
call getvalue
mov bx,val
mov key,bx
loop4:
cmp i,0
jl exitloop4
mov bx,i
mov ind,bx
```

```
call getvalue
mov bx,val
cmp bx,key
jle exitloop4
mov bx,i
inc bx
mov ind,bx
call setvalue
dec i
jmp loop4
exitloop4:
mov bx,key
mov val,bx
mov bx,i
inc bx
mov ind,bx
call setvalue
inc j
mov bx,j
cmp bx,n
jl loop3
;printing msg5
mov dx,offset msg5
mov ah,9
int 21h
;printing sorted array
mov cc,0
mov si,offset array
loop9:
mov dx,[si]
mov intg,dx
```

```
push si
call printnum
mov dx,offset htab
mov ah,9
int 21h
pop si
add si,2
inc cc
mov bx,cc
cmp bx,n
jl loop9
;exit program
mov ah,4ch
int 21h
main endp
getvalue proc ;val=array[ind]
mov m,0
mov cx,n
mov si,offset array
loopget:
mov bx,ind
cmp bx,m ;checking if ind==m
jne skipget
mov bx,[si]
mov val,bx
mov cx,0
jmp exitget
skipget:
add si,2
inc m
loop loopget
```

```
exitget:
ret
getvalue endp
setvalue proc ;array[ind]=val
mov m,0
mov cx,n
mov si,offset array
loopset:
mov bx,ind
cmp bx,m ;checking if ind==m
jne skipset
mov bx,val
mov [si],bx
mov cx,0
jmp exitset
skipset:
add si,2
inc m
loop loopset
exitset:
ret
setvalue endp
inputnum proc ;inputs an integer
mov intg,0
iter1:
mov ah,1
int 21h
cmp al,13
je enditer1
sub al,48
mov ah,0
mov digt,ax
mov ax,intg
mov bx,10
mul bx
add ax,digt
mov intg,ax
jmp iter1
enditer1:
ret
inputnum endp
printnum proc ;prints an integer
mov var1,0
mov count,0
iter2:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
mov al,dl
mov digt,ax
mov ax,var1
mov bx,10
mul bx
add ax,digt
mov var1,ax
inc count
cmp intg,0
jle enditer2
jmp iter2
```

```
enditer2:
mov bx,var1
mov intg,bx
mov cx,count
iter3:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
add dl,48
mov ah,2
int 21h
loop iter3
ret
printnum endp
end main
```

4. **Assume that two variables x and y are stored in unpacked BCD format. Write an 8086 alp to add x and y without using DAA and display the result in unpacked BCD format.**

```
.model small
.stack 100h
.data
x dw ? ;first unpacked BCD number
y dw ? ;second unpacked BCD number
x1 dw ? ;first number in decimal form
y1 dw ? ;second number in decimal form
```

```
z1 dw ? ;sum of x1 and y1
zh db ? ;higher byte of unpacked BCD
zl db ? ;lower byte of unpacked BCD
cc db 0 ;loop counter variable
count dw 0 ;printnum - digit counter
intg dw 0 ;printnum - I/O for inputnum and printnum
digt dw 0 ;printnum - extracted digit
var1 dw 0 ;printnum - reversed number
newl db 10,13,'$'
msg1 db 'Enter first unpacked BCD number : (BIN) $'
msg2 db 'Enter second unpacked BCD number : (BIN) $'
msg3 db 'Addition in decimal :',10,13,'$'
msg4 db ' + $'
msg5 db ' = $'
msg6 db 'Addition in unpacked BCD :',10,13,'$'
.code
main proc
mov ax,@data
mov ds,ax
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in x
call inputbcd
mov x,bx
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;taking input in y
```

```
call inputbcd
mov y,bx
;converting unpacked x to decimal x1
mov ax,x ;ax=x
mov bl,ah ;bl=ah
mov dh,0 ;dh=0
mov dl,al ;dl=al
mov ah,0 ;ah=0
mov al,bl ;al=bl
mov bl,10 ;bl=10
mul bl ;ax=al*bl
add ax,dx ;ax=ax+dx
mov x1,ax ;x1=ax
;converting unpacked y to decimal y1
mov ax,y ;ax=y
mov bl,ah ;bl=ah
mov dh,0 ;dh=0
mov dl,al ;dl=al
mov ah,0 ;ah=0
mov al,bl ;al=bl
mov bl,10 ;bl=10
mul bl ;ax=al*bl
add ax,dx ;ax=ax+dx
mov y1,ax ;y1=ax
;adding z1 = x1 + y1
mov ax,x1
mov bx,y1
add ax,bx
mov z1,ax
;printing msg3
mov dx,offset msg3
mov ah,9
int 21h
;printing x1 in decimal
mov bx,x1
mov intg,bx
call printnum
;printing msg4
mov dx,offset msg4
mov ah,9
int 21h
;printing y1 in decimal
mov bx,y1
mov intg,bx
call printnum
;printing msg5
mov dx,offset msg5
mov ah,9
int 21h
;printing z1 in decimal
mov bx,z1
mov intg,bx
call printnum
;printing newl
mov dx,offset newl
mov ah,9
int 21h
;printing msg6
mov dx,offset msg6
mov ah,9
int 21h
;printing x1 in BCD
```

```
mov bx,x1
call printbcd
;printing msg4
mov dx,offset msg4
mov ah,9
int 21h
;printing y1 in BCD
mov bx,y1
call printbcd
;printing msg5
mov dx,offset msg5
mov ah,9
int 21h
;printing z1 in BCD
mov bx,z1
call printbcd
;exit program
mov ah,4ch
int 21h
main endp
inputbcd proc ;inputs a BCD integer and stores it in BX
mov bx,0
mov cl,1
loop3:
mov ah,1
int 21h
cmp al,13
je endloop3
sub al,48
rol bx,cl
or bl,al
```

```
jmp loop3
endloop3:
ret
inputbcd endp
printbcd proc ;prints a BCD integer stored in BX
;extracting digits and unpacking
mov ax,bx
mov bl,10
div bl
mov zh,al
mov zl,ah
;printing higher byte in BCD format
mov bx,0
mov bl,zh
mov cc,0
loop4:
mov bh,0
mov cl,1
rol bx,cl
mov dl,bh
add dl,48
mov ah,2
int 21h
inc cc
cmp cc,8
jl loop4
;printing lower byte in BCD format
mov bx,0
mov bl,zl
mov cc,0
loop5:
```

```
mov bh,0
mov cl,1
rol bx,cl
mov dl,bh
add dl,48
mov ah,2
int 21h
inc cc
cmp cc,8
jl loop5
ret
printbcd endp
printnum proc ;prints an integer
mov var1,0
mov count,0
iter2:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
mov al,dl
mov digt,ax
mov ax,var1
mov bx,10
mul bx
add ax,digt
mov var1,ax
inc count
cmp intg,0
```

```
jle enditer2
jmp iter2
enditer2:
mov bx,var1
mov intg,bx
mov cx,count
iter3:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
add dl,48
mov ah,2
int 21h
loop iter3
ret
printnum endp
end main
```

5. **Write an 8086 alp to find out if a substring (which would be taken from the keyboard) is present in a given string (which would be taken from the keyboard) or not. If the substring is present it will return the location (or index) from where it matches otherwise print a negative message.**

```
.model small
.stack 100h
```

```
.data
i db 0 ;outer loop control
j db 0 ;outer loop upper limit
k db 0 ;inner loop control
f db 0 ;flag
n db 0 ;i+k
ch1 db ' ' ;character from original string
ch2 db ' ' ;character from substring
len1 db 0 ;length of original string
len2 db 0 ;length of substring
str1 db 100 dup('0') ;original string
str2 db 100 dup('0') ;substring
newl db 10,13,'$' ;new line
msg1 db 'Enter the original string : $'
msg2 db 'Enter the substring : $'
msg3 db 'The substring is PRESENT in the original
string.',10,13,'$'
msg4 db 'The substring is NOT PRESENT in the original
string.',10,13,'$'
msg5 db 'Starting index of substring in original string
= $'
msg6 db 'Ending index of substring in original string =
$'
count dw 0 ;printnum - digit counter
intg dw 0 ;printnum - I/O for inputnum and printnum
digt dw 0 ;printnum - extracted digit
var1 dw 0 ;printnum - reversed number
.code
main proc
mov ax,@data
mov ds,ax
```

```
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in str1
mov si,offset str1
loop1:
mov ah,1
int 21h
cmp al,13
je exitloop1
mov [si],al
inc si
inc len1
jmp loop1
exitloop1:
mov bl,'$'
mov [si],bl
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;taking input in str2
mov si,offset str2
loop2:
mov ah,1
int 21h
cmp al,13
je exitloop2
mov [si],al
inc si
inc len2
jmp loop2
exitloop2:
mov bl,'$'
mov [si],bl
;checking
mov i,0 ;i=0
mov bl,len1
sub bl,len2
mov j,bl ;j=len1-len2
for1: ;outer loop
mov f,0 ;f=0
mov k,0 ;k=0
for2: ;inner loop
;extracting (i+k)th character of str1
mov bl,i
add bl,k
mov n,bl
mov cl,n
add cl,1 ;cl=i+k+1
mov si,offset str1
extr1:
mov bl,[si]
inc si
loop extr1
mov ch1,bl
;extracting (k)th character of str2
mov cl,k
add cl,1 ;cl=k+1
mov si,offset str2
extr2:
```

```
mov bl,[si]
inc si
loop extr2
mov ch2,bl
;comparing the two characters
mov bl,ch1
cmp bl,ch2 ;ch1==ch2?
je continue2
mov f,1 ;f=1
jmp break2
continue2:
inc k ;k++
mov bl,k
cmp bl,len2 ;k<len2?
jl for2 ;end of inner for
jmp break2
for21:
jmp for1
break2:
cmp f,0 ;f==0?
je break1
inc i ;i++
mov bl,i
cmp bl,j ;i<j?
jle for21 ;end of outer for
break1:
cmp f,1 ;f==1?
je negative
positive: ;present
mov dx,offset msg3
mov ah,9
```

```
int 21h
;printing starting index
mov dx,offset msg5
mov ah,9
int 21h
mov dh,0
mov dl,i
add dl,1
mov intg,dx
call printnum
mov dx,offset newl
mov ah,9
int 21h
;printing ending index
mov dx,offset msg6
mov ah,9
int 21h
mov dh,0
mov dl,i
add dl,len2
mov intg,dx
call printnum
jmp exitprog
negative: ;not present
mov dx,offset msg4
mov ah,9
int 21h
exitprog:
mov ah,4ch
int 21h
main endp
```

```
printnum proc ;prints an integer
mov var1,0
mov count,0
iter2:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
mov al,dl
mov digt,ax
mov ax,var1
mov bx,10
mul bx
add ax,digt
mov var1,ax
inc count
cmp intg,0
jle enditer2
jmp iter2
enditer2:
mov bx,var1
mov intg,bx
mov cx,count
iter3:
mov ax,intg
mov bl,10
div bl
mov dl,ah
mov ah,0
mov intg,ax
add dl,48
mov ah,2
int 21h
loop iter3
ret
printnum endp
end main
```

6. **Write a program to multiply two unsigned word integers and display the product. The integers are stored in two variables in the data segment, namely X and Y each of which is a "word" data. Store the product in a variable that is 4 bytes long. Handle overflow case.**

```
.model small
.stack 100h
.data
x dw ? ;first word integer
y dw ? ;second word integer
p dw 2 dup(?) ;product in 32-bit/4-byte array form
str db ? ;product in string form for printing
fl db 0 ;flag to detect leading zeros
msg1 db 'Enter first number : (HEX) $'
msg2 db 'Enter second number : (HEX) $'
msg3 db 'The product is : (HEX) $'
.code
main proc
```

```
mov ax,@data
mov ds,ax
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in x
call inputhex
mov x,bx
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;taking input in y
call inputhex
mov y,bx
;calculating product of x and y
mov dx,0
mov ax,x
mov bx,y
mul bx
;storing product in p (array of size 32-bit/4-byte)
mov si,offset p
mov [si],dx ;higher word of product
add si,2
mov [si],ax ;lower word of product
;storing product in string str for printing
mov di,offset p
mov bx,[di] ;set bx to higher word of product
add di,2
mov si,[di] ;set si of lower word of product
mov di,offset str
mov dh,2 ;two 2-byte-word parts, hence iterate loop1
twice
loop1:
mov ch,4 ;ch = count of digits to be displayed
mov cl,4 ;cl = count of bits to be rotate
loop2:
rol bx,cl ;rotate bl towards left, i.e., MSB comes to
LSB
mov dl,bl ;dl = data to be displayed
and dl,0fh ;extract and keep only LSB
cmp dl,9 ;check if character is digit (0-9) or letter
(A-F)
jbe skip1
add dl,7 ;if letter then add 37h = 55 (07h + 37h)
skip1:
add dl,30h ;else add only 30h = 48
cmp fl,0
jne skip2
cmp dl,'0'
je skip3
mov fl,1
skip2:
mov [di],dl
inc di
skip3:
dec ch ;decrement count of digits to be displayed by 1
jnz loop2
dec dh ;loop1 control variable
cmp dh,0
mov bx,si ;set bx to lower word of product
```

```
jnz loop1
mov bl,'$'
mov [di],bl
;printing msg3
mov dx,offset msg3
mov ah,9
int 21h
;printing product from str
mov dx,offset str
mov ah,9
int 21h
;exit program
mov ah,4ch
int 21h
main endp
inputhex proc ;inputs a hexadecimal integer and stores
it in BX
mov bx,0
mov cl,4
loop3:
mov ah,1
int 21h
cmp al,13
je endloop3
cmp al,57
jle skip4
sub al,7
skip4:
sub al,48
rol bx,cl
or bl,al
```

```
jmp loop3
endloop3:
ret
inputhex endp
end main
```

7. **Write an 8086 ALP to check for the password using DOS interrupt. If the entry does not match with the password display "Wrong Password! Try Again" and remain in the loop, else display "You are authorized person" and come out. Set a counter so that a maximum of three tries is possible, otherwise, it will exit from the program.**

```
.model small
.stack 100h
.data
pw db 'afternoon$' ;original password
str db 100 dup(0) ;entered password
plen db 9 ;length of original password
len db 0 ;length of entered password
f db 0 ;flag
try db 0 ;tries
msg1 db 'Enter password : $'
msg2 db 'Wrong password! Try again. $'
msg3 db ' attempt(s) left.',10,13,'$'
msg4 db 'You are authorized person.$'
msg5 db 'Maximum tries exceeded.$'
.code
```

```
main proc
mov ax,@data
mov ds,ax
loop1:
inc try
mov f,0
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in str
mov len,0
mov si,offset str
loop2:
mov ah,1
int 21h
cmp al,13
je exitloop2
mov [si],al
inc si
inc len
jmp loop2
exitloop2:
mov bl,'$'
mov [si],bl
;comparing plen=9 and len
cmp len,9
je skiptocompare
mov f,1
jmp break3
skiptocompare:
```

```
;comparing pw and str
mov si,offset pw
mov di,offset str
mov cl,plen
loop3:
mov bl,[si]
mov dl,[di]
cmp bl,dl
je continue3
mov f,1
jmp break3
continue3:
inc si
inc di
loop loop3
break3:
cmp f,0
je positive
cmp try,3
je negative
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
;printing number of tries left (3-try)
mov dh,0
mov dl,51 ;3+48
sub dl,try
mov ah,2
int 21h
;printing msg3
```

```
mov dx,offset msg3
mov ah,9
int 21h
jmp loop1
positive:
;printing msg4
mov dx,offset msg4
mov ah,9
int 21h
jmp exitprog
negative:
;printing msg5
mov dx,offset msg5
mov ah,9
int 21h
exitprog:
mov ah,4ch
int 21h
main endp
end main
```

8. **Write an 8086 alp to read a string of 10 characters from the keyboard and display the same at screen position (x1, y1) using BIOS interrupts. (x1, y1 can be stored value.)**

```
.model small
.stack 100h
.data
i db 0 ;loop control variable
```

```
x1 db 16 ;stored value for row number
y1 db 36 ;stored value for column number
len db 0 ;length of entered string
str db 100 dup(0) ;entered string
msg1 db 'Enter the string of 10 characters : $'
msg2 db 'INVALID INPUT : The string does not have 10
characters.$'
.code
main proc
mov ax,@data
mov ds,ax
;printing msg1
mov dx,offset msg1
mov ah,9
int 21h
;taking input in str
mov si,offset str
mov cx,10
loop1:
mov ah,1
int 21h
cmp al,13
je exitloop1
mov [si],al
inc si
inc len
loop loop1
exitloop1:
mov bl,'$'
mov [si],bl
cmp len,10
```

```
jne invalid
;displaying str at (x1,y1)
mov si,offset str
mov dh,x1 ;row number
mov dl,y1 ;column number (initial)
mov bh,0 ;page number
loop2:
mov ah,02h ;setting cursor position at (dh,dl)
int 10h
mov ah,08h ;reading character and attribute at cursor
position
int 10h
mov bl,ah ;attributes to be written (from attributes
returned)
mov al,[si] ;character to be written
mov cx,1 ;number of times to write character
mov ah,09h ;writing character and attribute at cursor
position
int 10h
inc dl ;increasing column number
inc si
inc i
cmp i,10
jl loop2
jmp endprog
invalid:
;printing msg2
mov dx,offset msg2
mov ah,9
int 21h
endprog:
```

```
;end of program
mov ah,4ch
int 21h
main endp
end main
```