**CA LAB PRACTICALS**

**COMPUTER ENGINEERING AND INFORMATION TECHNOLOGY**

**COMPUTER ARCHITECTURE LAB – CP352**

**B.TECH CSE Vth SEMESTER**

**SURESH GYAN VIHAR UNIVERSITY, JAIPUR**

**Practicals are performed in 8086 simulator for exp1 to exp9 and MARS45 Simulator for exp10.**

**Exp.1. Write a Program to enter a number.**

**Code:**

.model small

.stack 100h

.data

msg db 10, 13, "enter a numbers $"

bufferSize db 21 ; 20 char + RETURN

inputLength db 0 ; number of count numbers

buffer db 21 DUP("$") ; actual buffer

.code

main proc

mov ax, @data

mov ds, ax

lea dx, msg

mov ah, 09h ;output

int 21h

mov dx, offset bufferSize ; load our pointer to the beginning of the structure

mov ah, 10 ; GetLine function

int 21h

MOV dl, 10

MOV ah, 02h

INT 21h

MOV dl, 13

MOV ah, 02h

INT 21h

mov ax, @data

mov ds , ax

lea dx, buffer

mov ah, 09 ;output

int 21h

endp

end main

**OUTPUT:**



**Exp.2. Write a Program to get factorial of a number.**

**Code:**

.MODEL SMALL

.STACK 1000h

.DATA

decstr DB 16 DUP ('$') ; String is $-terminated

.CODE

main PROC

mov ax, @DATA ; Initialize DS

mov ds, ax

mov bx, 9 ; Factorial 9! = 362880

xor dx, dx ; DX:AX=1 (first multiplicand)

mov ax, 1 ; Begin with 1

; for (dx:ax = 1, cx = 2; cx <= 10; cx++)

mov cx, 2 ; Incrementing multiplicator

L1:

call mul\_dword\_word ; DX:AX \* CX -> DX:AX

inc cx

cmp cx, bx

jbe L1 ; While cx <= 10

; Print result

mov di, OFFSET decstr

call dword\_to\_dec

mov dx, OFFSET decstr

mov ah, 9

int 21h

; Exit

mov ax, 4C00h

int 21h

main ENDP

mul\_dword\_word PROC ; DX:AX multiplicand, CX multiplier

push dx

mul cx ; AX \* CX -> DX:AX

mov si, dx ; Store high result

mov di, ax ; Low result won't be changed anymore

pop ax ; High word

mul cx ; AX \* CX -> DX:AX

add ax, si ; Add high result from last mul to low result here

adc dx, 0

mov si, dx ; SI:DX:AX return value

mov dx, ax

mov ax, di

ret ; RET: SI:DX:AX result

mul\_dword\_word ENDP

dword\_to\_dec PROC ; ARG DX:AX DWORD, DI: offset of string

mov cs:target, di

mov si, ax

mov di, dx

; First Loop: get digits and push them

mov cs:counter, 0

mov bx, 10

LL1:

inc cs:counter

xor dx, dx

mov ax, di ; High WORD

mov cx, ax

div bx ; DX:AX / BX -> AX Remainder DX

mov di, ax ; Store new high word

mul bx ; AX \* BX -> DX:AX

sub cx, ax ; sub highest CX-divisible value

mov dx, cx

mov ax, si ; Low WORD

div bx ; DX:AX / BX -> AX Remainder DX

or dl, 30h ; Convert remainder to ASCII

push dx ; Store remainder

mov si, ax ; Store new low WORD

or ax, di ; Anything more to process?

jnz LL1 ; yes: jump to LL1 above

; Second Loop: get back digits in reversed order

mov di, cs:target

mov cx, cs:counter

LL2:

pop ax

mov [di], al

inc di

loop LL2

mov BYTE PTR [di], '$' ; Terminator for INT 21h/09h

ret

counter dw 0

target dw 0

dword\_to\_dec ENDP

**OUTPUT:**



**Exp.3. Write a Program to reversing a string.**

**Code:**

print macro m

mov ah,09h

mov dx,offset m

int 21h

endm

.model small

;\*\*\*\*\*\* Data Segment \*\*\*\*\*\*

.data

empty db 10,13, " $"

str1 db 25,?,25 dup('$')

str2 db 25,?,25 dup('$')

mstring db 10,13, "Enter the string: $"

mstring2 db 10,13, "Enter second string: $"

mreverse db 10,13, "Reversed string: $"

;\*\*\*\*\*\*\*\*\*\* Code Segment \*\*\*\*\*\*\*\*\*\*\*\*

.code

start:

mov ax,@data

mov ds,ax

print mstring

call accept\_string

mov si,offset str1 ;point si to start of string1

mov di,offset str2 ;point di to start of string2

mov al,[si] ;copy first two locations of string1 to string2

mov [di],al ;since these contain the size and length of the string

inc si ;which are same in reverse string also

inc di

mov al,[si]

mov [di],al

inc si

inc di

mov cl,str1+1 ; copy length in cl

mov ch,00

add si,cx ;add length of string1 to si to move it to last location

dec si ;si at last location of string1

move\_more: mov al,[si] ;copying character one by one from string1 pointed by si

mov [di],al ; to string2 pointed by "di" in reverse order as si moves

dec si ; from last character to first character

inc di

dec cl

jnz move\_more

print mreverse

print str2+2 ; printing the reversed string

print empty

exit:

mov ah,4ch ;exit the program

int 21h

;accept procedure

accept proc near

mov ah,01

int 21h

ret

accept endp

display1 proc near

mov al,bl

mov bl,al

and al,0f0h

mov cl,04

rol al,cl

cmp al,09

jbe number

add al,07

number: add al,30h

mov dl,al

mov ah,02

int 21h

mov al,bl

and al,00fh

cmp al,09

jbe number2

add al,07

number2: add al,30h

mov dl,al

mov ah,02

int 21h

ret

display1 endp

accept\_string proc near

mov ah,0ah ;accept string from user function

mov dx,offset str1 ; store the string in memory pointed by "DX"

int 21h

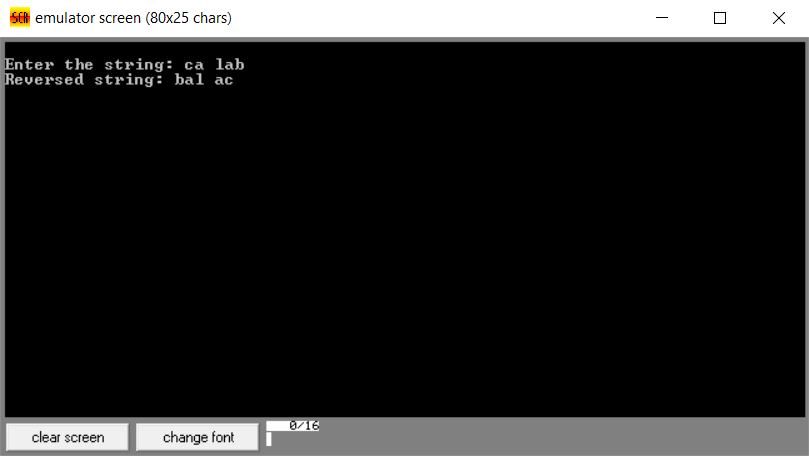
ret

accept\_string endp

end start

end

**OUTPUT:**



**Exp.4. Write a Program for integer to convert to bin to hex.**

**Code:**

.8086

.model small

.stack 256

data segment

pbuff1 db 'Input a decimal number:','$'

pbuff2 db 'To Hex:','$'

pbuff3 db ' <-- STOP: Wrong key pressed.',0dh,0ah,'$'

data ends

code segment

assume cs:code,ds:data

rpt:

mov ax,@data

mov ds,ax

lea dx,pbuff1

mov ah,9

int 21h

call dec2bin

push ax

mov dx,13

mov ah,2

int 21h

mov dx,10

mov ah,2

int 21h

lea dx,pbuff2

mov ah,9

int 21h

pop ax

call crlf

call bin2hex

call crlf

jmp rpt

dec2bin proc near

xor bx,bx

newchar:

mov ah,1

int 21h

cmp al,27d

je quit

cmp al,13d

je continue

sub al,30h

jl exit

cmp al,9

jg exit

cbw

xchg ax,bx

mov cx,10

mul cx

xchg ax,bx

add bx,ax

jmp newchar

exit:

push ax

lea dx,pbuff3

mov ah,9

int 21h

pop ax

continue:

ret

dec2bin endp

bin2hex proc near

mov ch,4

rotate:mov cl,4

rol bx,cl

mov al,bl

and al,0fh

add al,30h

cmp al,3ah

jl printit

add al,07h

printit:

mov dl,al

mov ah,2

int 21h

dec ch

jne rotate

ret

bin2hex endp

crlf proc near

mov dl,0dh

mov ah,2

int 21h

mov dl,0ah

mov ah,2

int 21h

ret

crlf endp

quit proc near

mov ah,4ch

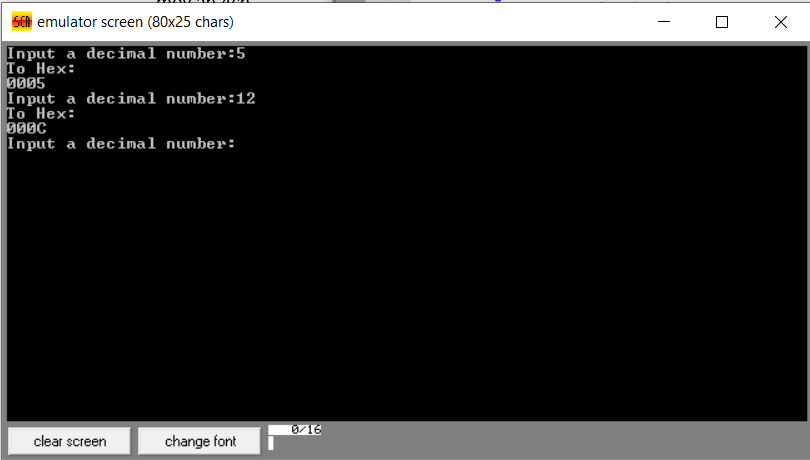
int 21h

quit endp

code ends

end rpt

**OUTPUT:**



**Exp.5. Write a Program for a count.**

**Code:**

.MODEL SMALL

.STACK 100H

.DATA

MSG1 DB 0AH, 0DH, "ENTER A STRING--> $"

MSG2 DB 0AH, 0DH, "NUMBER OF CHARACTERS ENTERED--> $"

INMSG DB 25 DUP('0')

.CODE

MAIN:

MOV AX, @DATA

MOV DS, AX

;----------------DISPLAY THE MESSAGE------------------

AGAIN:

LEA DX, MSG1

MOV AH, 09H

INT 21H

;-----------------------------------------------------

;-----------TAKE STRING INPUT------------------------

LEA SI, INMSG

MOV DX, SI

MOV AH, 0AH

INT 21H

MOV SI, 02H

CMP INMSG+SI, 0DH

JE EXIT

;---------------------------------------------------

LEA DX, MSG2

MOV AH, 09H

INT 21H

;----------SET SOURCE INDEX(SI) VALUE TO 1----------

;----------BECAUSE FIRST INDEX TELLS NUMBER OF BYTES READ--------

MOV SI,01H

MOV AL, INMSG+SI

MOV AH, 00H

MOV BL, 0AH ;

AAD ;

DIV BL ;BREAKING THE VALUE IF IT EXCEEDS 10

;--------------------------------------------------------

MOV DX, AX

ADD DL, 30H

MOV AH, 02H

INT 21H

MOV DL, DH

ADD DL, 30H

MOV AH, 02H

INT 21H

JMP AGAIN

EXIT:

MOV AH, 04CH

INT 21H

END MAIN

**OUTPUT:**



**Exp.6. Write a Program to lower the string.**

**Code:**

.MODEL SMALL

.DATA

MSG DB 0DH,0AH, ' ENTER THE STRING :-----> : $'

MSG2 DB 0DH,0AH, ' YOUR STRING IS :-----> : $'

STR1 DB 255 DUP(?)

ONE DB ?

TWO DB ?

.CODE

BEGIN:

MOV AX,@DATA

MOV DS,AX

LEA DX,MSG

MOV AH,09H

INT 21H

LEA SI,STR1

MOV AH,01H

READ:

INT 21H

MOV BL,AL

CMP AL,0DH

JE DISPLAY

XOR AL,20H

MOV [SI],AL

INC SI

;CMP BL,0DH

JMP READ

DISPLAY:

MOV AL,'$'

MOV [SI],AL

LEA DX,MSG2

MOV AH,09H

INT 21H

LEA DX,STR1

MOV AH,09H

INT 21H

; MOV AH,4CH

; INT 21H

.EXIT

END BEGIN

**OUTPUT:**

****

**Exp.7. Write a Program a simple procedure call.**

**Code: (For finding Maximum of numbers)**

DIM EQU 5

.model SMALL

.stack

.data

ARRAY DB DUP 4, 2, 1, 5

N DW ?

MAX DB ?

.code

.startup

LEA SI, ARRAY

MOV N, DIM

CALL MAXIMUM

.exit

MAXIMUM PROC NEAR

PUSH AX

PUSH BX

PUSH CX

PUSH DX

XOR AX, AX

MOV CX, N

MOV AL, [SI]

MOV MAX, AL

INC SI

SEARCH\_MAX: XOR AX, AX

MOV AL, [SI]

CMP AL, MAX

JA CHANGE\_MAX

CONTINUE\_MAX: INC SI

DEC CX

JNZ SEARCH\_MAX

JMP EXIT\_MAX

CHANGE\_MAX: MOV MAX, AL

JMP CONTINUE\_MAX

EXIT\_MAX:

POP DX

POP CX

POP BX

POP AX

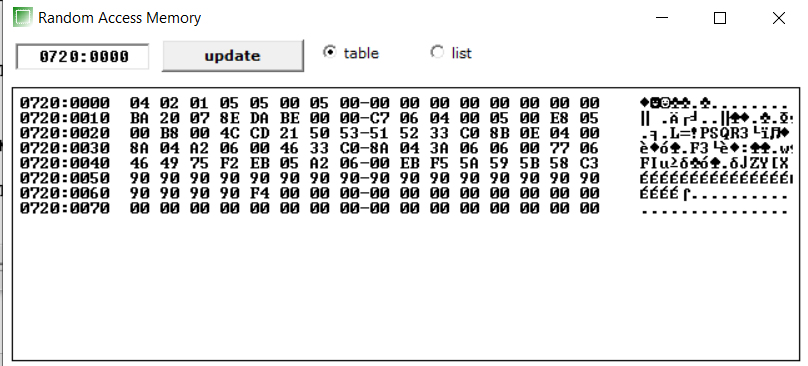
RET

MAXIMUM ENDP

end

**OUTPUT:**

**AUX🡪Memory**



**Exp.8. Write a Program to implement bubble sort.**

**Code:**

data segment

list db 15h,03h,08h,01h,14h

count equ 4

data ends

code segment

assume cs:code,ds:data

start:

mov ax,data

mov ds,ax

mov dx,count

call sort

mov ah,4ch

int 21h

sort proc near

rep1:

mov cx,dx

lea si,list

rep2:

mov al,[si]

cmp al,[si+1]

jl do

xchg [si+1],al

mov [si],al

do: inc si

loop rep2

dec dx

jnz rep1

ret

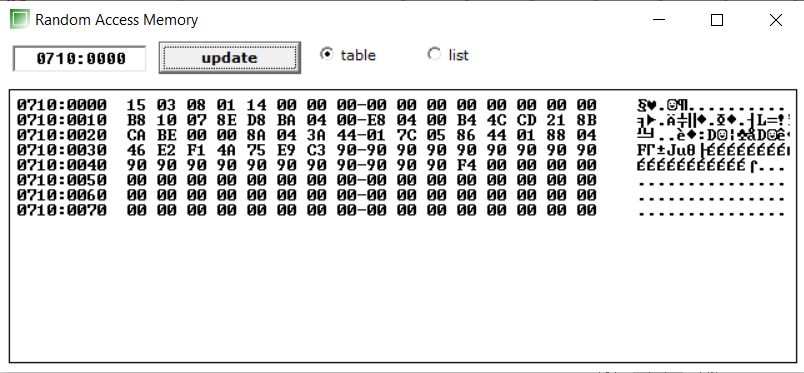
sort endp

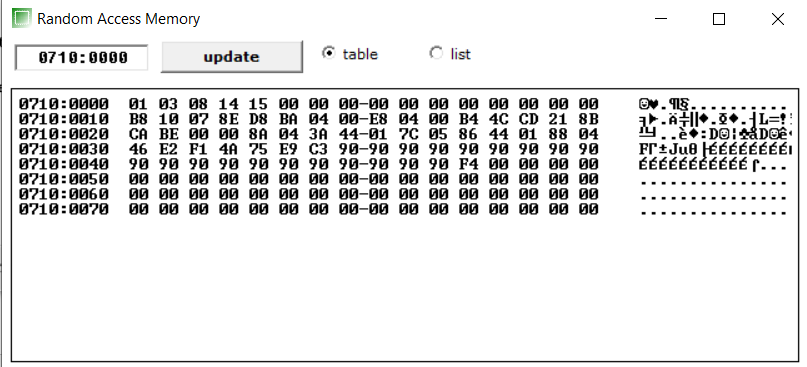
code ends

end start

**OUTPUT:**

**AUX🡪Memory**

****



**Exp.9. Write a Program a simple routine to demonstrate functions.**

**Code:**

; Concatenation of strings in 8086 ALP

;macro for printing a string

print macro m

mov ah,09h

mov dx,offset m

int 21h

endm

.model small

;\*\*\*\*\*\* Data Segment \*\*\*\*\*\*

.data

empty db 10,13, " $"

str1 db 25,?,25 dup('$')

str2 db 25,?,25 dup('$')

mstring db 10,13, "Enter the string: $"

mstring2 db 10,13, "Enter second string: $"

mconcat db 10,13, "Concatenated string: $"

;\*\*\*\*\*\*\*\*\*\* Code Segment \*\*\*\*\*\*\*\*\*\*\*\*

.code

start:

mov ax,@data

mov ds,ax

print mstring

call accept\_string

;storing string in str2

print mstring2

mov ah,0ah

lea dx,str2

int 21h

mov cl,str1+1 ;length of string1 in cl

mov si,offset str1

next: inc si

dec cl

jnz next

inc si

inc si

mov di,offset str2

inc di

inc di

mov cl,str2+1

move\_next:

mov al,[di]

mov [si],al

inc si

inc di

dec cl

jnz move\_next

print mconcat

print str1+2

exit:

mov ah,4ch ;exit the program

int 21h

;accept procedure

accept proc near

mov ah,01

int 21h

ret

accept endp

display1 proc near

mov al,bl

mov bl,al

and al,0f0h

mov cl,04

rol al,cl

cmp al,09

jbe number

add al,07

number: add al,30h

mov dl,al

mov ah,02

int 21h

mov al,bl

and al,00fh

cmp al,09

jbe number2

add al,07

number2: add al,30h

mov dl,al

mov ah,02

int 21h

ret

display1 endp

accept\_string proc near

mov ah,0ah ;accept string from user function

mov dx,offset str1 ; store the string in memory pointed by "DX"

int 21h

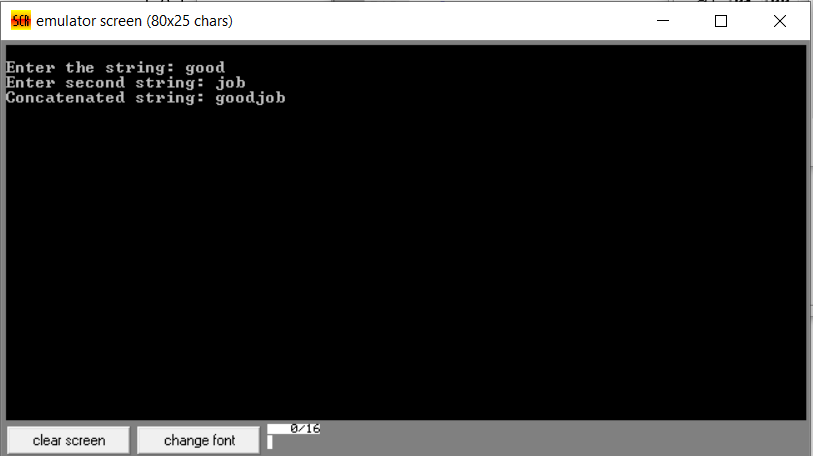
ret

accept\_string endp

end start

end

**OUTPUT:**



**Exp.10. Write a Program for Ackermann function.**

**Code:**

.data

.align 2

prmptM:.asciiz "\nEnter m value to calculate A(m,n): "

prmptN:.asciiz "\nEnter n value to calculate A(m,n): "

result:.asciiz "\nAckermann's function with M and N = "

#--------------------------------

.text

.globl main

main:

move $s0, $0 # s0 : computed A(m,n) value

# get user input M

la $a0, prmptM #load prmpt for M

li $v0, 4 #code to print string

syscall #print

li $v0, 5 #take int input

syscall

move $s1, $v0 # s1 = user input "m"

# get user input N

la $a0, prmptN #load prmpt for N

li $v0, 4 #code to print string

syscall #print

li $v0, 5 #take int input

syscall

move $s2, $v0 # s2 = user input "n"

# call Ackman func

move $a0, $s1

move $a1, $s2

jal ackman

move $s0, $v0 # s0 = result from Ackman func

#------ Display results and exit ---------------------------------

la $a0, result #load display string

li $v0, 4 #code to print string

syscall #print

li $v0, 1 #code to print int

move $a0, $s0 #load computed A(m,n)

syscall #print

#----------------- Exit ---------------------

li $v0, 10

syscall

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# ackman function

#

# a0 - user input "m"

# a1 - user input "n"

#

# v0 - computed A(m,n)

ackman:

#--------------- Usual stuff at function beginning ---------------

addi $sp, $sp, -24

sw $ra, 20($sp)

sw $s0, 16($sp)

sw $s1, 12($sp)

sw $s2, 8($sp)

sw $s3, 4($sp)

sw $s4, 0($sp)

#-------------------------- function body -------------------------

move $s0, $a0 # s0 : "M"

move $s1, $a1 # s1 : "N"

move $s2, $0 # s2 : computed A(m,n)

# base case if m = 0

bne $s0, 0, cont1 # if (M == 0)

addi $s2, $s1, 1 # s2 = (n+1)

# else if m > 0 and n = 0

cont1: ble $s0, 0, cont2 # if (M > 0)

bne $s1, 0, cont2 # if (N == 0)

addi $a0, $s0, -1 # a0 : M = (m-1)

addi $a1, $0, 1 # a1 : N = 1

jal ackman # compute

move $s2, $v0 # s2 = A(m-1,1)

# else if m > 0 and n > 0 ----- A(m-1, A(m,n-1))

cont2: ble $s0, 0, done # if (M > 0)

ble $s1, 0, done # if (N > 0)

#inner

move $a0, $s0 # a0 : M = m

addi $a1, $s1, -1 # a1 : N = (n-1)

jal ackman

move $a1, $v0 # a1 : N = A(m,n-1)

#outer

addi $a0, $s0, -1 # a0 : M = (m-1)

jal ackman # compute

move $s2, $v0 # s2 = A(m-1, A(m,n-1))

done: move $v0, $s2

#-------------------- Usual stuff at function end -----------------

lw $ra, 20($sp)

lw $s0, 16($sp)

lw $s1, 12($sp)

lw $s2, 8($sp)

lw $s3, 4($sp)

lw $s4, 0($sp)

addi $sp, $sp, 24

jr $ra

**OUTPUT:**

