**PROGRAM No. 1**

# Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.

**; === Define a macro to display a string ===**

disp macro msg

lea dx, msg ; Load address of message into DX mov ah, 9 ; DOS function 09h – display string int 21h ; Interrupt to call DOS function

endm

.model small ; Define memory model

.stack ; Define stack segment

.data ; Data segment begins

# ; Predefined message strings (with carriage return and line feed)

m1 db 10,13,"enter string 1:$" m2 db 10,13,"enter string 2:$"

m3 db 10,13,"length of string 1 is:$" m4 db 10,13,"length of string 2 is:$" m5 db 10,13,"string1 equal to string2$"

m6 db 10,13,"string1 not equal to string2$"

# ; Input buffers for strings (DOS format: MaxLen, ActualLen, Data...)

str1 db 80 dup(40) str2 db 80 dup(40)

# ; Variables to hold string lengths

l1 db ? l2 db ?

.code ; Code segment begins

# ; Initialize data segment registers

mov ax, @data mov ds, ax mov es, ax

# ; Prompt user for first string

disp m1

lea dx, str1 ; Load address of str1 buffer

call read ; Call read procedure to take input

# ; Prompt user for second string

disp m2

lea dx, str2 ; Load address of str2 buffer

call read ; Call read procedure to take input

# ; Store the actual length of string 1

mov al, [str1+1] mov l1, al

# ; Store the actual length of string 2

mov al, [str2+1] mov l2, al

# ; Compare lengths of the two strings

cmp al, l1

jne strnote ; If lengths differ, jump to not equal

; Set up for comparing strings character by character mov ch, 0

mov cl, l1 ; Length of string to compare lea si, str1+2 ; SI points to first char of str1 lea di, str2+2 ; DI points to first char of str2

cld ; Clear direction flag (increment)

repe cmpsb ; Repeat compare while equal

jne strnote ; If any character mismatched, not equal

; If equal

disp m5 ; Display "strings are equal"

jmp next ; Skip to displaying lengths

strnote: ; If strings are not equal

disp m6 ; Display "strings are not equal" next:

# ; Display length of string 1

disp m3 mov al, l1

call displ ; Call procedure to display 2-digit number

# ; Display length of string 2

disp m4 mov al, l2

call displ ; Call procedure to display 2-digit number

; Exit program mov ah, 4ch

int 21h

# ; === Procedure to read a string using DOS Function 0Ah ===

read proc

mov ah, 0ah ; DOS buffered input int 21h

ret read endp

# ; === Procedure to display 2-digit decimal number ===

displ proc

aam ; Convert AL into two decimal digits (AH=tens, AL=units) mov bx, ax

add bx, 3030h ; Convert digits to ASCII

mov ah, 2 ; int 21h / AH=2 prints a single character (DL = char). mov dl, bh ; Display tens digit

int 21h

mov dl, bl ; Display units digit int 21h

ret displ endp

end ; End of program

Procedure

Masm filename.asm link filename.obj cv filename.exe

F5 to run the code,or F8 to run step by step File exit to see the result

**PROGRAM No. 2**

# Simulate a Decimal Up-counter to display 00-99.

.model small ; Use the small memory model (code and data fit in one segment)

.stack ; Define the stack segment (default size)

.data ; Start of data segment

msg db "press any key to exit$" ; Message to display on screen

.code ; Start of code segment

start:

mov ax, @data ; Load address of data segment into AX mov ds, ax ; Initialize DS with data segment address call clear ; Clear the screen

lea dx, msg ; Load the address of the message into DX mov ah, 9 ; DOS function to print string

int 21h ; Call DOS to print message

mov ax, 00h ; Initialize AX with 0, used as a counter nxtnum:

push ax ; Save current count value on stack (to preserve across subroutines) call setcursor ; Set the cursor position (row=12, col=40)

call disp ; Display the current value in AX call delay ; Delay so it doesn't count too fast

mov ah, 01h ; Check for key press (non-blocking) int 16h ; BIOS interrupt

jnz exit ; If a key was pressed, jump to exit

pop ax ; Restore the previous AX value from the stack add ax, 1 ; Increment the counter

daa ; Decimal Adjust AL (optional for BCD representation) cmp ax, 0 ; Loop always unless overflowed to 0 (unlikely)

jnz nxtnum ; Repeat loop exit:

mov ah, 4Ch ; DOS function to terminate program int 21h ; Return control to DOS

# ; setcursor: Moves the cursor to row 12, column 40

setcursor proc

mov ah, 2 ; BIOS function to set cursor position

mov dh, 12 ; Row 12

mov dl, 40 ; Column 40

int 10h ; BIOS video interrupt ret

setcursor endp

# ; disp: Displays the 2-digit number in AL

disp proc

mov bl, al ; Save original AL value in BL

mov dl, al ; Copy AL to DL for upper nibble

mov cl, 4 ; Prepare to shift 4 bits

shr dl, cl ; Shift DL right 4 bits (upper nibble)

add dl, 30h ; Convert high nibble to ASCII

mov ah, 2 ; DOS function to print character

int 21h ; Print high digit

mov dl, bl ; Get original value back

and dl, 0Fh ; Mask to get low nibble

add dl, 30h ; Convert to ASCII

int 21h ; Print low digit ret

disp endp

# ; delay: Creates a time delay using nested loops

delay proc

mov bx, 00FFh ; Outer loop counter b2:

mov cx, 0FFFFh ; Inner loop counter b1:

loop b1 ; Decrement CX and loop if not zero

dec bx ; Decrement BX

jnz b2 ; Repeat outer loop if BX != 0 ret

delay endp

# ; clear: Clears the screen using BIOS scroll function

clear proc

mov al, 0 ; Number of lines to scroll (0 = clear entire window) mov ah, 6 ; BIOS function to scroll window up

mov ch, 0 ; Upper-left row = 0 mov cl, 0 ; Upper-left column = 0 mov dh, 24 ; Bottom-right row = 24

mov dl, 79 ; Bottom-right column = 79 mov bh, 7 ; Attribute (gray on black) int 10h ; BIOS video interrupt

ret clear endp

end start ; Mark program end and entry point

Procedure

Masm filename.asm link filename.obj debug filename.exe

**PROGRAM No. 3**

# Compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non- negative integers.

.model small ; Use small memory model

.stack ; Define default stack segment

.data ; Start of data segment

n dw 4 ; n = 4 (can be changed)

r dw 2 ; r = 2 (can be changed)

ncr dw 0 ; Result of nCr will be stored here msg db "ncr= $" ; Message to display result

.code ; Start of code segment start:

mov ax, @data ; Load data segment address into AX mov ds, ax ; Set DS with the data segment address

mov ax, n ; Load n into AX

mov bx, r ; Load r into BX

call ncrpro ; Call recursive procedure to calculate nCr

mov ax, ncr ; Move result into AX

mov bx, ax ; Copy to BX for printing later

lea dx, msg ; Load address of message into DX

mov ah, 9 ; DOS function to display string

int 21h ; Call DOS

mov ax, bx ; Move nCr result into AX again for display

aam ; Adjust AX into unpacked BCD (AH = tens, AL = ones)

mov bx, ax ; Store result in BX

add bx, 3030h ; Convert digits to ASCII ('0' = 30h)

mov dl, bh ; Move high digit (tens) to DL

mov ah, 2 ; DOS function to display character int 21h

mov dl, bl ; Move low digit (ones) to DL

int 21h ; Display it

mov ah, 4Ch ; Terminate program int 21h

# ; ===== Recursive Procedure to Calculate nCr =====

; Uses Pascal's identity:

; nCr = (n-1)Cr + (n-1)C(r-1)

ncrpro proc near

cmp bx, ax ; if r == n

je res1 ; then result is 1

cmp bx, 0 ; if r == 0

je res1 ; then result is 1

cmp bx, 1 ; if r == 1

je resn ; then result is n

dec ax ; Calculate (n-1) cmp bx, ax ; if r == (n-1)

je incr ; then result is n

# ; First recursive call: (n-1)Cr

push ax ; Save current ax and bx push bx

call ncrpro ; Recursive call

pop bx ; Restore bx and ax pop ax

# ; Second recursive call: (n-1)C(r-1)

dec bx ; r = r - 1 push ax

push bx call ncrpro

pop bx ; Restore registers pop ax

ret

# ; === Result cases ===

res1:

inc ncr ; Increment result (1 added) ret

incr:

inc ncr ; For case when r == (n - 1), result is n resn:

add ncr, ax ; For r == 1, result is n ret

ncrpro endp

end ; End of program

Procedure

Masm filename.asm link filename.obj cv filename.exe

F5 to run the code,or F8 to run step by step

e ds:……. (location of ncr check in the code)

**PROGRAM No. 4**

# Sort a given set of ‘n’ numbers in ascending and descending orders using the Bubble Sort algorithm.

.model small ; Use small memory model (single code & data segment)

.stack 100 ; Reserve 100 bytes for stack

.data ; Data segment

a db 10,6,8,0,4,2 ; Array of 6 elements to sort

len dw ($ - a) ; Calculate length of array (6 bytes here)

.code ; Start of code segment start:

mov ax, @data ; Load data segment address into AX mov ds, ax ; Initialize DS with data segment address mov bx, len ; Load length of array into BX (BX = 6)

dec bx ; BX = len - 1 = 5 (number of outer loop passes)

outloop: ; Outer loop for Bubble Sort (5 passes needed for 6 elements) mov cx, bx ; CX = number of inner loop iterations (decreases each pass) mov si, 0 ; SI = index into array (starting at 0)

inloop:

mov al, a[si] ; Load current element into AL cmp al, a[si+1] ; Compare AL with next element

jb next ; If AL < next element, skip swap (already in correct order)

# ; Swap a[si] and a[si+1]

xchg al, a[si+1] ; Exchange AL with a[si+1]

mov a[si], al ; Store the original a[si+1] into a[si] next:

inc si ; Move to next index

loop inloop ; Decrease CX and repeat inner loop if CX != 0 dec bx ; Decrement outer loop counter

jnz outloop ; Repeat outer loop if BX != 0

; End of program

mov ah, 4Ch ; Terminate program int 21h

end start ; End of code, entry point is "start"

Procedure

Masm filename.asm link filename.obj cv filename.exe

F5 to run the code,or F8 to run step by step

e ds:……. (location of specified in mov al,a[si] check in the code)

**PROGRAM No. 5**

# Read the current time from the system and display it in the standard format on the screen.

.model small ; Define small memory model (1 code + 1 data segment)

.stack ; Allocate default stack segment

.data

msg db 10,13,"current time is $" ; Message to display. 10,13 = newline, $ = terminator

.code start:

mov ax, @data ; Load the address of the data segment into AX

mov ds, ax ; Initialize DS with the address in AX so we can access variables

# ; Display the message: "current time is"

lea dx, msg ; Load address of the message into DX

mov ah, 9 ; DOS function 09h: display string at DS:DX ending with '$' int 21h ; Call DOS interrupt

# ; Get the current time from system

mov ah, 2Ch ; DOS function 2Ch: get system time

int 21h ; Returns time in CH (hour), CL (minute), DH (second), DL (hundredths)

# ; Display the hour

mov al, ch ; Move hour into AL

call disp ; Call disp procedure to display 2-digit number

# ; Display ':'

mov dl, ':' ; Load colon character into DL

mov ah, 2 ; DOS function 02h: display character in DL int 21h ; Call DOS interrupt

# ; Display the minute

mov al, cl ; Move minute into AL

call disp ; Display minute

# ; Display ':'

mov dl, ':' ; Display another colon mov ah, 2

int 21h

# ; Display the second

mov al, dh ; Move second into AL

call disp ; Display seconds

# ; Display '.'

mov dl, '.' ; Optional stylistic period mov ah, 2

int 21h

# ; Terminate program and return to DOS

mov ah, 4Ch ; DOS function 4Ch: exit program int 21h

# ; --- Display Procedure: Converts binary in AL to two ASCII digits and prints them ---

disp proc near

aam ; Adjust AL to unpack BCD (e.g., 25 → AH=2, AL=5) add ax, 3030h ; Convert AH and AL to ASCII digits by adding '0' (30h) mov bx, ax ; Copy AX to BX

mov dl, bh ; Move high digit (tens) to DL

mov ah, 2 ; DOS function 02h

int 21h ; Display first digit

mov dl, bl ; Move low digit (ones) to DL

int 21h ; Display second digit

ret ; Return from procedure disp endp

end start ; End of program; entry point is 'start'

Procedure

Masm filename.asm link filename.obj debug filename.exe