Program#3: Program that defines the procedure “addition” to add two numbers, passed through

stack, and return their sum.

.model small

.stack

.data

.code

Main proc

Mov ax,@data ; These instructions initialize the data segment register (DS) with the starting address of the data segment.

Mov ds,ax

Mov ax,5

Mov bx,10

Push ax ;pushing value of ax, which is 5 to stack

Push bx ;pushing value of bx, which is 10 to stack

Call addition ; calling procedure

Pop bx ; removing number 10 from stack

Pop bx ; These instructions pop the values from the stack back into the BX register. Since we pushed the values in the order AX, BX, we need to pop them in the reverse order to get the correct values back.

.exit

Main endp

Addition proc ;defining procedure

Push bp ;storing value of BP on stack so that we can restore it later

Mov bp,sp ;to access stack without pop instruction—moving TOS to bp

Mov ax,0 ;no need to push AX as it is safe to use AX register

Add ax,[bp+4] ; [bp+4] contains value 10

Add ax,[bp+6] ; [bp+6] contains value 5

Pop bp ; restoring value of BP from stack

Ret ; transferring control back to the calling procedure

Addition endp

Program#4: Program that defines the procedure “addition” to sum up an array of 5 elements and

return its sum. Note: Array is always passed by reference.

.model small

.data

array db 1,2,3,4,5

.code

Main proc

Mov ax,@data

Mov ds,ax

Mov bx,offset array ;base address of array

mov ax,5 ;size of array

Push bx These instructions load the offset of the **array** variable into the BX register, load the value 5 into the AX register, and then push the values in the BX and AX registers onto the stack.

Push ax

Call addition ;calling procedure

pop bx ;removing parameters from stack

pop bx ;removing parameters from stack

.exit

Main endp

Addition proc ;Defining procedure

push bp ;saving the value of BP on stack before using it

push cx ;saving the value of CX on stack before using it

push si ;saving the value of SI on stack before using it

mov bp,sp ;moving top of the stack to bp

These instructions set up the stack frame for the procedure, load the values from the stack into the CX and AX registers, and then initialize the AX and SI registers to 0.

mov cx,[bp+8] ;reading size of array

mov ax,[bp+10] ;reading base address of array

mov bp,ax ;moving base address to BP

mov ax,0 ; it is safe to use ax without pushing it

mov si,0 ;assigning index register with 0

l1:

Add al,ds:[bp+si] ;accessing value 10

inc si ;incrementing index by 1 as the array is of a byte type

loop l1

pop si ;Restoring original value of SI from stack

pop cx ;Restoring original value of CX from stack

pop bp ;Restoring original value of BP from stack

ret ;returning control back to calling procedure

Addition endp

.model small

.data

This directive is where you would define initialized data. In this case, we have three arrays: **Array** and **a2** contain the values to be added together, and **res** is a third array that will store the result. The **sum** variable will store the final result.

Array db 1,2,3,4,5

a2 db 2,4,6,8,9

res db 5 dup[?]

sum db 0

.code

Mov ax,@data

Mov ds,ax

These instructions set up the loop counter (CX) to 5, load the offset of the **Array** variable into the BX register, load the offset of the **a2** variable into the AX register, and then set the extra segment (ES) register to the value in the AX register. The SI register is initialized to 0, and the AL register is initialized to 0

Mov cx , 5

Mov bx, offset Array

Mov ax, offset a2

Mov es , ax

Mov si, 0

Mov al , 0

L1:

mov al, [bx + si]

This loop adds the elements of the **Array** and **a2** arrays together and stores the result in the **res** array. The loop continues until the CX register reaches 0.

add al, [es + si]

mov res[si] , al

mov ah , res[si]

Inc si

loop L1

mov sum , al

.exit

Loops

.model small

.stack 100h

.data

.code

main proc

mov cx,10

mov dx , 48 ; ascii code for 0

test:

mov ah,2

int 21h

add dx,1 ; increment

inc dx ; increment

loop test

Main endp

end main