**Lab Manual 9**

## Subroutine Example:

JMP START

ARR1 DW 2,3,4,5,6,7

ARR2 DW 3,4,8,7,12,9

SQUARE:

L1:

mov ax,[bx+si]

mul ax

mov [bx+si],ax

add si,2

loop L1

mov si,0

RET

START:

mov cx,6

lea bx,ARR1

CALL SQUARE

mov cx,6

lea bx,ARR2

CALL SQUARE

.exit

## Stack Example:

ORG 100h

MOV AX, 1234h

PUSH AX ; store value of AX in stack.

MOV AX, 5678h ; modify the AX value.

POP AX ; restore the original value of AX.

RET

## Graded Task

## NOTE:

**Use stack to take input and output parameter from subroutine.**

## Task 1:

Write a subroutine which takes address of an array as parameter and output sum of that array. **Note:** Array values remain unchanged

SOL:

Jmp start

result dw 0

arr dw 1,2,3,4,5

count db 5

SUM:

PUSH bp

mov bp,sp

mov bx,[bp+4] ;copy address of array from stack

mov cl,count ;size of the array elements

L1:

add dx,[bx]

add bx,2 ;array is type word so increment of 2 for each element.

loop L1

mov [bp+6],dx ;saving answer in space reserved for return value

POP bp

ret 2 ;return pops the value at top in IP

; 2 after ret indicates that there is one argument of function.

;2 X no. arguments after ret statement so that stack points at our

;reserved location where the answer is stored.

start:

sub sp,2 ;reserving space for the answer in stack

lea bx,arr

push bx ;pushing the address of the array

call SUM

pop result

.exit

## Task 2:

Write a subroutine which takes address of variable and position of bit as parameter and set or resets bit at that position accordingly and output that modified number.

**Note**: variable value remain unchanged

SOL:

Jmp start

var dw 0xABCD

;bit dw 5

bit dw 6

res dw 0

SET:

Push bp

mov bp,sp

mov cx,[bp+4] ;reading the nth position from stack

mov ax,1 ;to set at nth position that bit should be 1 and must be 'or' with the no. while other bits should be 0

shl ax,cl ;shifting left n times

mov bx,[bp+6] ;reading the variable address

mov bx,[bx] ;getting the value from address

or bx,ax ;logical operation on variable and modified nth position value

Pop bp

ret

RESET:

Push bp

mov bp,sp

mov cx,[bp+4] ;reading the nth position from stack

mov ax,0xFFFE ;to reset at nth position, that bit should be 0 and must be 'and' with the no. while other bits should be 1

rol ax,cl ;rotating left not shifting left because shifting inserts 0 automatically but

;we want 0 at specific nth position to reset our number.

mov bx,[bp+6] ;reading the variable address

mov bx,[bx] ;getting the value from address

and bx,ax ;logical operation on variable and modified nth position value

Pop bp

ret

start:

lea si,var

Push si ;push address of variable

Push bit ;push the nth position of bit to set/reset

;Call SET ;choose between set / reset function

Call RESET

pop bit

pop si

mov res,bx

.exit

## Task 3:

Write a subroutine that takes address of an array as parameter, and counts the odd elements.

SOL:

Jmp start

arr dw 1,2,3,4,5,6,7,8,9,0xA

size dw 10

count db 0

Odd:

Push bp

mov bp,sp

mov cx,[bp+4]

mov bx,[bp+6]

L1:

mov ax,[bx]

shr ax,1

JNC next ;if there is no carry out after shift right then skip to next element

INC count ;if carry exist then increment the no.

next:

add bx,2 ;to get next element add 2

loop L1

pop bp

ret

start:

lea bx,arr

Push bx

Push size

Call Odd

pop size

pop bx

ret

## Task 4:

Write a subroutine that takes address of an array as parameter, and counts the elements which are in the form. .

**Note:** Array values remain unchanged

SOL:

Jmp start

arr dw 2,4,15,30,128

size db 5

count db 0

Form2N:

Push bp

mov bp,sp

mov cl,[bp+4]

mov bx,[bp+6]

L1:

mov ax,[bx]

mov dx,0

L2:

shr ax,1 ;shifting the number to get bit in carry

JNC noAdd

INC dx

noAdd:

Cmp ax,0 ;checking if the bits in the number are finished or not.

JNZ L2 ;continue process till the bits are finished

add bx,2 ;when the bits are finished get the next element

Cmp dx,1 ;in 2^n there should be only single occurrence of 1.

JNE next ;if more than 1 bits are found then skip to the next element

INC count ;if dx=1 then incrementing the counter

next:

loop L1

pop bp

ret

start:

lea bx,arr

Push bx

mov bl,size

mov bh,0

Push bx

Call Form2N

pop bx

pop bx

ret

## Task 5:

Write a subroutine that takes address of an array of integers, size of array, direction of rotation of array and number of rotations as parameters through stack, and rotates the array left or right the number of times specified. Type of array is ‘word’.

|  |  |  |
| --- | --- | --- |
| 1 | Address of array |  |
| 2 | Size of array | A positive integer |
| 3 | Direction of Rotation | 0 or 1  0= rotate left  1=rotate right |
| 4 | Number of rotations | Any positive integer |

jmp start

rotate\_array:

pop di ;saving the return IP value from stack

pop cx ;get number of rotations to be performed on array.

pop dx

pop ax

pop bx

dec ax

mov temp , ax

rotate\_times\_loop:

mov ax , temp

mov si , 0

cmp dx , 1

je rotate\_right

push [bx] ;saving the first element in stack

rotate\_left:

mov bp , [bx + si + 2] ;copying the values in memory-> n-1=n

mov [bx + si] , bp

add si , 2 ;controlling the next element

dec ax ;number of elements remaining to be rotated in array

cmp ax , 0

jnz rotate\_left

pop [bx+si] ;copy the value saved in stack back into the array

jmp exit

rotate\_right:

add si , ax

add si , ax

push [bx+si]

rotate\_right\_loop:

mov bp , [bx + si - 2]

mov [bx + si] , bp

sub si , 2

dec ax

cmp ax , 0

jnz rotate\_right\_loop

pop [bx]

exit:

loop rotate\_times\_loop ;repeat the rotation till the value of variable (no. of rotations)

push di ;pushing the IP return value saved at the start of the function

ret

start:

lea bx , arr

mov ax , size

mov dx , direction

mov cx , rotations

push bx

push ax

push dx

push cx

mov bx , 0

mov cx , 0

mov dx , 0

mov ax , 0

call rotate\_array

.exit

arr dw 1,2,3,4,5

size dw 5

direction dw 0

rotations dw 3

temp dw 0