**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Lab 10** | |
| **Topic** | 1. Stack operations 2. Implementation of subroutine 3. Passing parameters to subroutine via stack |

**PART 1**

## *Stack Example:*

**ADD FIRST Two Number that are pushed in stack without POP**

MOV AX, 5

MOV BX, 7

MOV CX,8

PUSH AX

PUSH BX

PUSH CX

MOV BP, SP ; SP CURRENT ADDREESS IS STORED IN BP

MOV AX, [BP+2]; AX=5

MOV BX, [BP+4]; BX=7

|  |  |
| --- | --- |
| Memory Address | Value in stack |
| FFFE | FF00 |
| FFFC | 0005 |
| FFFA | 0007 |
| FFF8 | 0008 SP |

## *Subroutine Example(1):*

USING BP (Base Pointer)

JMP START

NUM1 DW 5

FACT DW ?

MyFUNCTION:

MOV BP,SP ; TOP OF THE STACK WILL HAVE RETURNING ADDRESS.

MOV CX,[BP+4] ;CX=5

MOV DI,[BP+2] ; DI=2

MOV AX,1

L1:

MUL CX

LOOP L1

MOV [DI],AX

RET

START

PUSH NUM1 ; PUSHING NUMBER IN STACK

LEA BX,FACT ; PUSHING ADDRESS OF VARIABLE IN WHICH RESULT WILL BE SAVED

CALL MYFUNCTION ; CALLING THE FUNCTION

.EXIT

## *Subroutine Example(2):*

WITHOUT USING BP (Base Pointer)

JMP START

|  |  |
| --- | --- |
| Memory Address | Value in stack |
| FFFE | FF00 |
| FFFC | 0005 |
| FFFA | 0002 SP |

NUM1 DW 5

FACT DW ?

MyFUNCTION:

Pop SI ; SAVING RETURNING ADDRESS

POP DI ; DI=2

POP CX ; CX=5

MOV AX,1

|  |  |
| --- | --- |
| Memory Address | Value in stack |
| FFFE | FF00 |
| FFF8 | 001F SP |

L1:

MUL CX ***AFTER COMPLETE EXECUTION THE STACK STATE WILL BE:***

LOOP L1

MOV [DI],AX

PUSH SI ; YOU HAVE TO PUSH VALUE OF RETURNING ADDRESS AGAIN

RET

START:

PUSH NUM1 ; PUSHING NUMBER IN STACK

LEA BX,FACT ; PUSHING ADDRESS OF VARIABLE IN WHICH RESULT WILL BE SAVED

PUSH BX

CALL MYFUNCTION ; CALLING THE FUNCTION

.EXIT

Subroutine Example(3):

RETURNING OUTPUT

JMP START

|  |  |
| --- | --- |
| Memory Address | Value in stack |
| FFFE | FF00 |
| FFFC | 0005 SP |

NUM1 DW 5

FACT DW ?

MyFUNCTION:

POP SI ; SAVING RETURNING ADDRESS

POP CX ; CX=5

MOV AX,1

|  |  |
| --- | --- |
| Memory Address | Value in stack |
| FFFE | FF00 |
| FFFC | 0078 |
| FFFA | 001F SP |

L1:

MUL CX

LOOP L1

PUSH AX

PUSH SI ; YOU HAVE TO PUSH VALUE OF RETURNING ADDRESS AGAIN

RET

START:

PUSH NUM1 ; PUSHING NUMBER IN STACK

CALL MYFUNCTION ; CALLING THE FUNCTION

POP AX ; POP THE RESULT IN AX

MOV FACT,AX ; MOVING RESULT INTO VARIABLE

.EXIT

CHARACTER ASCII EXAMPLE:

jmp start

sampleword db 'UCP FALL 2017'

character db ?

start:

lea bx, sampleword

mov al,[bx] ;55 in hex is the ascii of ‘U’

mov cl,[bx+12] ;37 in hex is the ascii of ‘7’

mov caharacter,cl

ret

**PART 2**

## *NOTE: USE STACK TO PASS THE ARGUMENTS TO A SUBROUTINE*

## Task 1:

Write a subroutine that checks whether the number is prime or not. If it is prime then set DX=0xABCD else set DX=0xDCBA.

SOL:

Jmp start

num dw 5

checkPrime:

mov bp,sp

Mov bx,2

L1:

mov ax,[bp+2]

mov dx,0

div bx

cmp dx,0

JZ notPrime

inc bx

cmp bx,[bp+2]

JNZ L1

mov dx,0xABCD

Jmp L2

notPrime:

mov dx,0xDCBA

L2:

ret

start:

push num

Call checkPrime

pop ax

ret

## Task 2:

Write a subroutine that calculates the discriminant? In C++ the function will be

int discr(int a,int b,int c)

;return (b\*b-(4\*a\*c))

SOL:

Jmp start

a dw 2

b dw 3

c dw 1

res dd 0

Discr:

pop si

pop cx

pop ax

pop bx

mul ax

push dx

push ax

mov ax,cx

mul bx

mov bx,4

mul bx

pop cx

pop bx

sub cx,ax

sub bx,dx

push cx

push bx

push si

ret

start:

push a

push b

push c

Call Discr

pop dx

pop ax

mov [res],ax

mov [res+2],dx

ret

## Task 3:

W rite a subroutine that checks whether the number is an Armstrong Number or not.

***An Armstrong Number is a Number which is equal to it’s sum of digit’s cube.***

For example : 153 is an Armstrong number.

* 153 = (1\*1\*1) + (5\*5\*5) + (3\*3\*3).

1+125+27=153

* 371= (3\*3\*3) + (7\*7\*7) + (1\*1\*1).

27+343+1=371

If the number is Armstrong then move 1 to AX else -1.

SOL:

Jmp start

num dw 153

arr dw 0,0,0

checkArmstrong:

pop si

mov bp,sp

mov ax,[bp]

mov cx,3

mov bx,10

L1:

div bl

mov dh,0

mov dl,ah

push dx

mov ah,0

dec cx

JNZ L1

mov cx,3

mov di,arr

L2:

pop ax

mov bx,ax

mul bl

mul bl

mov [di],ax

INC di

INC di

loop L2

lea di,arr

mov ax,0

mov cx,3

L3:

add ax,[di]

add di,2

loop L3

mov bp,sp

cmp ax,[bp]

JNE notequal

mov ax,1

Jmp ex

notequal:

mov ax,-1

ex:

push si

ret

start:

push num

Call checkArmstrong

pop bx

ret

## Task 4:

Write a subroutine to calculate the sum of the Fibonacci series which takes one argument. Calculate series till first n values.

For example if n=10 then series will be 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Then SUM will be 88.

SOL:

Jmp start

num dw 10

res dw 0

Fab:

pop si

mov ax,0

mov bx,1

add dx,ax

add dx,bx

add di,dx

INC di

pop cx

sub cx,3

L1:

mov ax,dx

add dx,bx

add di,dx

mov bx,ax

loop L1

push di

push si

ret

start:

push num

CALL Fab

pop ax

mov res,ax

ret

## Task 5:

Write a subroutine that takes address of an array of character as parameter and counts vowels.

SOL:

Jmp start

arr db 'The solutions are for practise',0

res dw 0

vowels:

pop si

pop bx

L1:

mov al,[bx]

Cmp al,0

JE ex

Cmp al,'a'

JE vow

Cmp al,'e'

JE vow

Cmp al,'i'

JE vow

Cmp al,'o'

JE vow

Cmp al,'u'

JNE next

vow:

INC cx

next:

Inc bx

Jmp L1

ex:

push cx

push si

ret

start:

lea bx,arr

push bx

CALL vowels

pop cx

mov res,cx

ret

## Task 6 :

Write a subroutine that takes the address of a variable as an argument and if the word is palindrome then count the number of vowel characters in that word.

For example :

* ujaeieaju

Count=7 vowels

* abcdecb(Not a palindrome)

Count=0

SOL:

jmp start

data db 'ujaeieaju',0

count dw 0

palindrome:

mov bp,sp

mov cx,0

mov al,0

mov di,[bp+2]

L1:

CMP ax,[di]

JZ L2

Inc cx

INc di

jmp L1

L2:

mov di,[bp+2]

mov bx,[bp+2]

add di,cx

dec di

mov ax,cx

mov bl,2

div bl

mov cl,al

mov ch,0

L3:

mov al,[di]

CMP al,[bx]

JNE ex

dec di

inc bx

loop L3

mov bx,[bp+2]

L4:

mov al,[bx]

cmp al,0

JE ex

Cmp al,'a'

JE vow

Cmp al,'e'

JE vow

cmp al,'i'

JE vow

Cmp al,'o'

JE vow

cmp al,'u'

JNE next

vow:

INC count

next:

Inc bx

Jmp L4

ex:

ret

start:

lea bx,data

Push bx

CALL palindrome

pop bx

ret

## Task 7(BONUS):

Write an assembly language program to invert the case of the characters within a word.

For example:

Vriable1: AbcDe

Outuput: aBCdE

SOL:

Jmp start

var db 'aBcDe',0

invert:

pop si

pop bx

L1:

mov al,[bx]

CMP al,0

JE ex

xor al,0x20

mov [bx],al

INC bx

JMP L1

ex:

push si

ret

start:

lea bx,var

push bx

CALL invert

ret