Lab: 9

Fahid Imran

Roll No: 23i-0061

COAL

Instructor

Mr. Sulaman Saboor

Fast NUCES Islamabad

Campus

Task1:

Code:

```
include Irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte 'Enter a number: ', 0
line byte '<------------------------------------>', 0
msg1 byte 'Push: ', 0
msg2 byte 'Pop: ', 0
num dword ?

.code
main PROC

mov edx , offset line
call writestring
call crlf

mov ecx , 8
label1:
  mov edx , offset msg
  call writestring
  call readint

  push eax
  mov edx , offset msg1
  call writestring
  call writedec
  call crlf

loop label1

mov edx , offset line
call writestring
call crlf
mov edx , offset line
call writestring
```

```
    call crlf

    mov ecx , 8
    label2:

      pop eax
      mov edx , offset msg2
      call writestring
      call writedec
      call crlf

    loop label2

    mov edx , offset line
    call writestring
    call crlf

    exit
    main endp
end main
```

Output:

```
<------------------------------------>
Enter a number: 1
Push: 1
Enter a number: 2
Push: 2
Enter a number: 3
Push: 3
Enter a number: 4
Push: 4
Enter a number: 5
Push: 5
Enter a number: 6
Push: 6
Enter a number: 7
Push: 7
Enter a number: 8
Push: 8
<------------------------------------>
<------------------------------------>
Pop: 8
Pop: 7
Pop: 6
Pop: 5
Pop: 4
Pop: 3
Pop: 2
Pop: 1
<------------------------------------>
```

Task2:

Code:

```
include Irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte 'Enter a number: ', 0
line byte '<---------------------------------->', 0
msg1 byte 'Push: ', 0
msg2 byte 'Pop: ', 0
```

```
num dword ?

.code

Push_proc Proc
  pop esi
  mov ecx , 5
  label1:
        mov edx , offset msg
        call writestring
        call readint
        push eax
        call display1


  loop label1
  push esi
  ret
Push_proc endp

pop_proc Proc
  pop esi
  mov ecx , 5
  label2:

        pop eax

        call display2

  loop label2
  push esi
  ret
pop_proc endp

display1 Proc

  mov edx , offset line
  call writestring
  call crlf
  mov edx , offset msg1
  call writestring
  call writedec
```

```
        call crlf
        mov edx , offset line
        call writestring
        call crlf
        ret
    display1 endp

    display2 Proc
        mov edx , offset line
        call writestring
        call crlf
        mov edx , offset msg2
        call writestring
        call writedec
        call crlf
        mov edx , offset line
        call writestring
        call crlf
        ret
    display2 endp
    main PROC
        call Push_proc
        call pop_proc


    exit
    main endp
end main
```

Output:

```
Push: 2
<------------------------------------>
Enter a number: 3
<------------------------------------>
Push: 3
<------------------------------------>
Enter a number: 4
<------------------------------------>
Push: 4
<------------------------------------>
Enter a number: 5
<------------------------------------>
Push: 5
<------------------------------------>
<------------------------------------>
Pop: 5
<------------------------------------>
<------------------------------------>
Pop: 4
<------------------------------------>
<------------------------------------>
Pop: 3
<------------------------------------>
<------------------------------------>
Pop: 2
<------------------------------------>
<------------------------------------>
Pop: 1
<------------------------------------>
```

Task3:

Code:

```
include Irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte 'Enter a String: ', 0
line byte '<------------------------------------>', 0
msg1 byte 'Original String: ', 0
msg2 byte 'Reversed String: ', 0
num dword ?
```

```
var dword ?

str1 byte 'My hobby is Painting and crafting.', 0

.code

display_String PROC
  pop esi
  pop ebx
  pop ecx

  label1:

        mov al , byte ptr [ebx]
        call writechar
        inc ebx

  loop label1

  push esi
  ret
display_String endp

reverse_String PROC
  pop esi
  pop ebx
  pop ecx ; length of string
  mov num, ecx
  mov var, ebx

  goto1:
        mov eax, 0
        mov al , [ebx]
        inc ebx
        push eax

  loop goto1

  mov ecx, num
  mov ebx, var

  goto2:
```

```
        pop eax
        ;call writechar
        mov byte ptr [ebx], al
        inc ebx
  loop goto2

  push esi
  ret
reverse_String endp

main PROC

  mov edx, offset line
  call writestring
  call crlf

  mov edx, offset msg1
  call writestring

  push lengthof str1
  push offset str1
  call display_String
  call crlf

  mov edx, offset line
  call writestring
  call crlf

  push lengthof str1
  push offset str1
  call reverse_String

  mov edx, offset line
  call writestring
  call crlf

  mov edx, offset msg2
  call writestring

  push lengthof str1
  push offset str1
  call display_String
```

```
    call crlf

    mov edx, offset line
    call writestring
    call crlf


  exit
  main endp
end main
```

Output:

```
<------------------------------------>
Original String: My hobby is Painting and crafting.
<------------------------------------>
<------------------------------------>
Reversed String: .gnitfarc dna gnitniaP si ybboh yM
<------------------------------------>
```

Task4:

Code:

```
include Irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte 'Enter a String: ', 0
line byte '<---------------------------------->', 0
msg1 byte 'Original String: ', 0
msg2 byte 'Reversed String: ', 0
num dword ?
var dword ?

str1 byte 50 DUP (0)

.code

display_String PROC
  pop esi
```

```
  pop ebx
  pop ecx

  label1:

        mov al , byte ptr [ebx]
        call writechar
        inc ebx

  loop label1

  push esi
  ret
display_String endp

reverse_String PROC
  pop esi
  pop ebx
  pop ecx ; length of string
  mov num, ecx
  mov var, ebx

  goto1:
        mov eax, 0
        mov al , [ebx]
        inc ebx
        push eax

  loop goto1

  mov ecx, num
  mov ebx, var

  goto2:
        pop eax
        ;call writechar
        mov byte ptr [ebx], al
        inc ebx
  loop goto2

  push esi
  ret
```

```
reverse_String endp

main PROC

  mov edx, offset msg
  call writestring
  mov ecx, lengthof str1
  mov edx, offset str1
  call readstring
  call crlf

  mov edx, offset line
  call writestring
  call crlf

  mov edx, offset msg1
  call writestring

  push lengthof str1
  push offset str1
  call display_String

  mov edx, offset line
  call writestring
  call crlf

  push lengthof str1
  push offset str1
  call reverse_String

  mov edx, offset line
  call writestring
  call crlf

  mov edx, offset msg2
  call writestring

  push lengthof str1
  push offset str1
  call display_String
  call crlf
```

```
    mov edx, offset line
    call writestring
    call crlf


    exit
    main endp
end main
```

Output:

```
Enter a String: Pakistan is my country.

<---------------------------------------->
Original String: Pakistan is my country.
<---------------------------------------->
<---------------------------------------->
Reversed String:
.yrtnuoc ym si natsikaP
<---------------------------------------->
```

Task5:

Code:

```
include Irvine32.inc
.386
.model flat, stdcall
.stack 4096

.data
msg byte 'Enter a String: ', 0
line byte '<---------------------------------------->', 0
line2 byte '------------------------------------------------------', 0
line3 byte '<--------------------( Stack )--------------------->', 0
msg1 byte 'Message: Stack Overflow.', 0
msg2 byte 'Message: Empty Stack.', 0
msg3 byte 'Message: Element Pushed on stack.', 0
msg4 byte 'Message: Element poped from stack.', 0
str2 byte '| __ ', 0
str3 byte '| ', 0
str4 byte ' ', 0
str5 byte '|', 0
num dword ?
var dword ?
```

```
option1 byte 'Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: ',
0
input1 byte 'Enter a number: ', 0


Stack_arr dword 10 dup (0)
top_pointer dword offset Stack_arr
length_of_stack dword lengthof Stack_arr
no_of_elements dword 0



.code

Push_element PROC

  mov ecx, length_of_stack
  cmp ecx, no_of_elements
  je goto1
  jmp goEnd1
  goto1:
        mov edx, offset msg1
        call writestring
        call crlf
        jmp goEnd2
  goEnd1:
        mov ebx, top_pointer
        mov dword ptr [ebx], eax
        add top_pointer, 4
        inc no_of_elements
        mov edx, offset msg3
        call writestring
        call crlf
  goEnd2:

  ret
Push_element endp

Pop_element PROC

  cmp no_of_elements, 0
  je goto1
```

```
  jmp goEnd1

  goto1:
        mov edx, offset msg2
        call writestring
        call crlf
        jmp goEnd2
  goEnd1:
        sub top_pointer, 4
        mov ebx, top_pointer
        mov eax, dword ptr [ebx]
        mov dword ptr [ebx], 0
        dec no_of_elements
        mov edx, offset msg4
        call writestring
        call crlf
  goEnd2:

  ret
Pop_element endp

display_Stack PROC
  pop esi

  pop ecx
  pop ebx

  mov edx, offset line3
  call writestring
  call crlf

  mov edx, offset line2
  call writestring
  call crlf
  mov ecx, length_of_stack
  label1:

        mov eax , dword ptr [ebx]
        cmp eax, 0
        je goto1
        jmp goto2
        goto1:
```

```
                mov edx, offset str2
                call writestring
                jmp goto3
        goto2:
        mov edx, offset str3
        call writestring
        call writedec
        mov edx, offset str4
        call writestring
        goto3:
        add ebx, 4

  loop label1
  mov edx, offset str5
  call writestring
  call crlf

  mov edx, offset line2
  call writestring
  call crlf
  push esi
  ret
display_Stack endp

main PROC

  mov var, 1

  menu_loop:
        mov edx, offset option1
        call writestring
        call readint

        cmp eax, 0
        je goto0
        cmp eax, 1
        je goto1
        cmp eax, 2
        je goto2
        cmp eax, 3
        je goto3
        jmp goEnd
```

```
            goto0:
                  mov var, 0
                  jmp goEnd
            goto1:
                  mov edx, offset input1
                  call writestring
                  call readint
                  call push_element
                  jmp goEnd
            goto2:
                  call pop_element
                  jmp goEnd
            goto3:
                  push offset stack_arr
                  push lengthof stack_arr
                  call display_Stack
            goEnd:

      cmp var, 0
      jne menu_loop


    exit
    main endp
end main
```

Output:

```
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 12
Message: Element Pushed on stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | __ | __ | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 89
Message: Element Pushed on stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | 89 | __ | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 67
Message: Element Pushed on stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | 89 | 67 | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 89
Message: Element Pushed on stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
```

```
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 34
Message: Element Pushed on stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | 89 | 67 | 89 | 67 | 74 | 83 | 94 | 34 | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 56
Message: Element Pushed on stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | 89 | 67 | 89 | 67 | 74 | 83 | 94 | 34 | 56 |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 1
Enter a number: 89
Message: Stack Overflow.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit:
```

```
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | 89 | 67 | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 2
Message: Element poped from stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | 89 | __ | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 2
Message: Element poped from stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| 12 | __ | __ | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 2
Message: Element poped from stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
<--------------------( Stack )-------------------->
-------------------------------------------------
| __ | __ | __ | __ | __ | __ | __ | __ | __ | __ |
-------------------------------------------------
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 2
Message: Empty Stack.
Enter 1 to push, 2 to pop, 3 to see stack, 0 to exit: 3
```