

HW # 6: Theme: Stacks and Procedures

All main questions carry equal weight.

(Credit awarded to only those answers for which work has been shown.)

1. **[Procedures]** Write a main program which sets the registers BX and CX and calls a procedure *Add-Two*. The procedure *Add-Two* adds the values in registers BX and CX and returns the output (which is the sum) in AX. Single step through the program, displaying the value of the stack pointer so that you understand how the call and return are implemented.
code 18 points; set the registers, call Add-Two, add the values in BX and CX, return output in AX, 4* 4.5 points
the pointer value 3 * 5 points, it needs at least 3 steps to show how the call and return are implemented

```

; HW6 - 1

;
.model flat,stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:dword

INCLUDE Irvine32.inc

.data
count DWORD ?

.code
AddTwo PROC
    mov ax, bx
    add ax, cx
    ret
AddTwo ENDP

main PROC
    mov eax, 0
    mov ebx, 0
    mov ecx, 0
    mov bx, 1
    mov cx, 5
    call AddTwo
    call WriteInt
exit
main ENDP
END main

```

Microsoft Visual Studio Debug Console

```
+6  
C:\Users\Ziyan Tian\Desktop\AssemblyLanguagePro
```

```

14M - 00000000 EIP = 00000000 EIA = 00000000 EIS = 00000000 EIPF = 00000000 EIPF = 00000000 EIPF = 00000000 EIPF = 00000000
UE = 0 P = 0 R = 1 P1 = 0 CR = 1 AC = 0 P1 = 1 CY = 0

1
2 1
3 1
4 100
5 pushl $0x10000000
6 movl $0x10000000, %eax
7 ExitProcess PROCEDURE, defaulCode dword
8
9 INCLD64 1x10000000
10
11 .data
12 count: DQWORD 0
13
14 .code
15 Address PROC
16     mov ecx, 0
17     add ecx, 10
18     ret
19 Address ENDP
20
21 main PROC
22     mov ecx, 0
23     mov ecx, 0
24     mov ecx, 1
25     mov ecx, 1
26     call GetProcessId
27     call GetProcessId
28     exit
29     main ENDP
30     PROC main

```

```
00000000 EIP = 00000000 EAX = 0000000A ECX = 0000000A EDI = 00000000 ESP = 00000008 EBP = 00000000  
IN = 0 IP = 0 EIP - 1 PC = 0 PC - 1 CR = 0 PC - 1 CF = 0
```

```
1  j    RET     1  
2  
3      l      1  
4      [RDI]  
5      movl   $1,%eax  
6      rax    RDI  
7      call   PROC7, do_all_code_dump  
8  
9      INCLUDE Irvine32.inc  
10  
11     .data  
12     count DWORD ?  
13  
14     .code  
15     AddFun PROC  
16         mov esi, ecx ;ecx changed  
17         add eax, esi  
18         ret  
19     AddFun ENDP  
20  
21     main PROC  
22         mov eax, 0  
23         mov ebx, 0  
24         mov edx, 0  
25         mov hex, 5  
26         mov cx, 5  
27         call AddFun  
28         call WriteInt  
29     exit  
30     main ENDP  
31     END
```

```
EAX = 00000000 EAX = 00000000 ECX = 00000000 EDI = 0040100A ESI = 0040100A ESP = 0040107F ESP = 0040107E ESP = 0040107D ESP = 0040107C  
0 IF 0 IF = 0 IF = 1 IF = 2 IF = 3 AC = 8 FE = 1 CY = 0  
  
[C:\...]  
1 | j 796 - 1  
2  
3 |  
4 | JMP  
5 .model flat,stdcall  
6 .stack 4096  
7 ExitProcess PROTO, dwExitCode dword  
8  
9 INCLUDE Irvine32.inc  
10  
11  
12 .data  
13 count DWORD ?  
14  
15 .code  
16 _start PROC  
17 mov esi, 8n  
18 add esp, 4n  
19 ret  
20 _start ENDP  
21  
22 main PROC  
23 mov eax, 0  
24 mov ebx, 0  
25 mov ecx, 0  
26 mov edx, 1  
27 mov esi, 5  
28  
29 call WriteInt ; I'm confused  
30 exit  
31 main ENDP
```

2. [Arrays] Write a program that:

- 1) Prompts the user for integer input 5 times 8 points
- 2) Stores these inputs in a stack using the Push instruction 8 points
- 3) After the storing is complete in the Step 2, pop the stored values and display them on the screen using WriteInt (**not** DumpRegs). 8 points

In your submission, please embed the full program (.asm and .lst file) and one screen shot with at least one positive and one negative input value. Use the following: files 6 points, screenshot 3 points

```
.data
PromptUser BYTE "Please enter a value:", 0

.data
PromptUser BYTE "Please enter a value:", 0

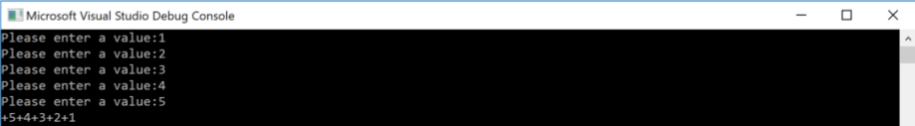
.code
main PROC

    ; 1) Prompt the user for integer 5 times
    mov ecx, 5
L1:
    mov edx, OFFSET PromptUser
    call WriteString

    ; 2) Store the input in a stack
    call ReadInt
    push eax
    loop L1

    ; 3) Displays the stored values
    mov ecx, 5
L2: pop eax
    call WriteInt
    loop L2

exit
main ENDP
END main
```



The screenshot shows the Microsoft Visual Studio Debug Console window. It displays the output of the program, which consists of five prompts: "Please enter a value:". The user has entered the values 1, 2, 3, 4, and 5 in sequence. The output shows the values 1, 2, 3, 4, and 5 being displayed on separate lines. At the bottom of the console, there is a calculation: +5+4+3+2+1.

3. [Compares, Procedures] Write a procedure, *Search* which searches the stack for the value that you provide in register AX and returns its index, assuming the first value is stored in index 0. Write a main program that fills the stack with positive values, sets AX and calls *Search* and prints the index at which the value was found.

For example, if the inputs are: 5, 6, 1, 10, 44, 79

and AX is set in the main to be 1, then the expected output of your code is:

The target value is 1, and is located at index:2

In cases where more than one element has the same value, you only have to output one of them. If the value is not found, print -1.

Use the following:

```
.data
prompt      BYTE "Please input a value: ", 0
spacing     BYTE ", ", 0;
String2     BYTE "The target value is," 0
String2     BYTE "and is located at index: ", 0
```

In your submission, please embed the full program (.asm and .lst file) and one screen shot showing the values found. Please test several sets of positive and negative values

code 24 points, files 6 points, screen shot 3points

```
.code
main PROC

    ; 1) Prompt the user for integer 6 times
    mov ecx, 6
L1: mov edx, OFFSET prompt
    call WriteString

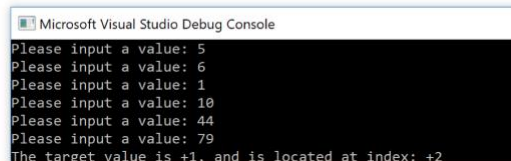
    ; 2) Store the input in a stack
    call ReadInt
    push eax
    loop L1

    ; AX = 1,
    mov eax, 1

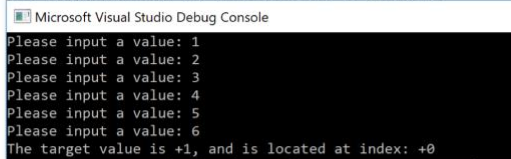
; Search
    mov ecx, 6
L2: pop ebx
    cmp eax, ebx
    jz L3
    loop L2

L3: cmp ecx, 0
    jnz L5
L4: cmp eax, ebx
    jnz L5
    add ecx, 1
L5: mov edx, OFFSET String1
    call WriteString
    call WriteInt
    mov edx, OFFSET spacing
    call WriteString
    mov edx, OFFSET String2
    call WriteString
    sub ecx, 1
    mov eax, ecx
    call WriteInt

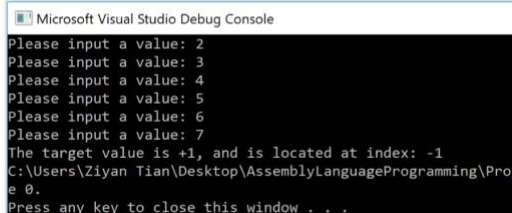
exit
main ENDP
END main
```



```
Microsoft Visual Studio Debug Console
Please input a value: 5
Please input a value: 6
Please input a value: 1
Please input a value: 10
Please input a value: 44
Please input a value: 79
The target value is +1, and is located at index: +2
```



```
Microsoft Visual Studio Debug Console
Please input a value: 1
Please input a value: 2
Please input a value: 3
Please input a value: 4
Please input a value: 5
Please input a value: 6
The target value is +1, and is located at index: +0
```



```
Microsoft Visual Studio Debug Console
Please input a value: 2
Please input a value: 3
Please input a value: 4
Please input a value: 5
Please input a value: 6
Please input a value: 7
The target value is +1, and is located at index: -1
C:\Users\Ziyan Tian\Desktop\AssemblyLanguageProgramming\Pro
e 0.
Press any key to close this window . . .
```