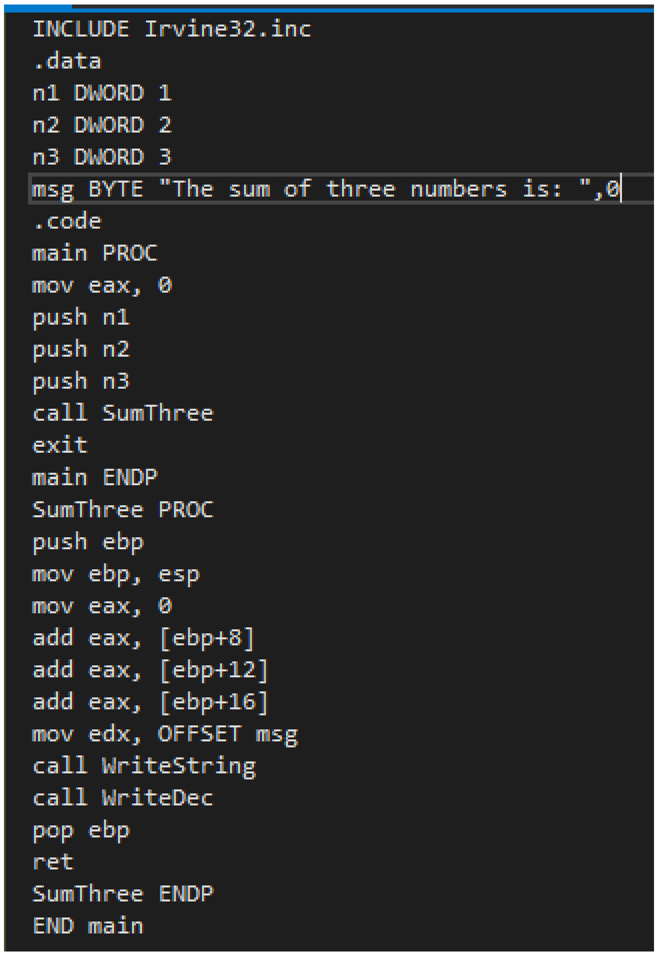
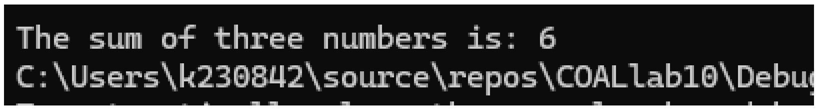
TASK 1

Write a program containing a procedure named **SumThree** that displays the sum of three numeric parameters passed through the stack.



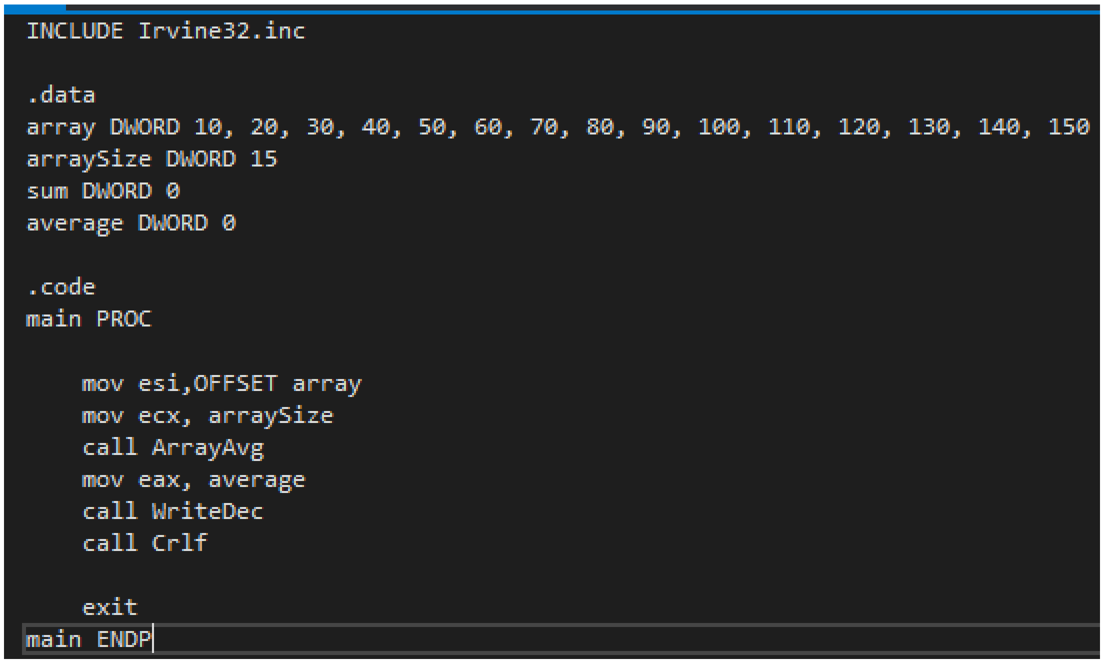
```
INCLUDE Irvine32.inc
.data
n1 DWORD 1
n2 DWORD 2
n3 DWORD 3
msg BYTE "The sum of three numbers is: ",0
.code
main PROC
mov eax, 0
push n1
push n2
push n3
call SumThree
exit
main ENDP
SumThree PROC
push ebp
mov ebp, esp
mov eax, 0
add eax, [ebp+8]
add eax, [ebp+12]
add eax, [ebp+16]
mov edx, OFFSET msg
call WriteString
call WriteDec
pop ebp
ret
SumThree ENDP
END main
```



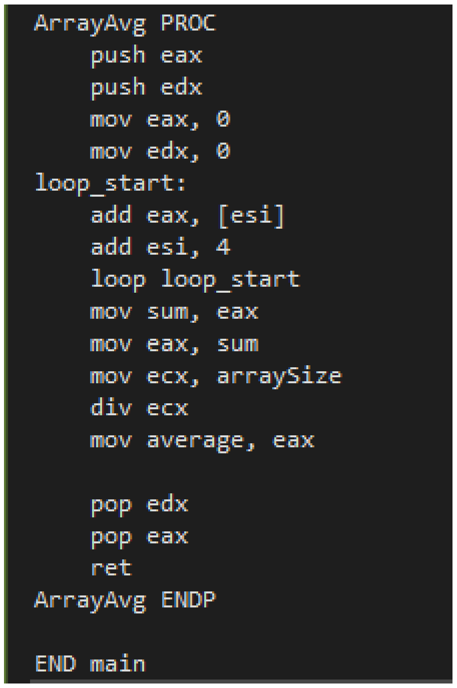
```
The sum of three numbers is: 6
C:\Users\k230842\source\repos\COALlab10\Debu
```

TASK 2

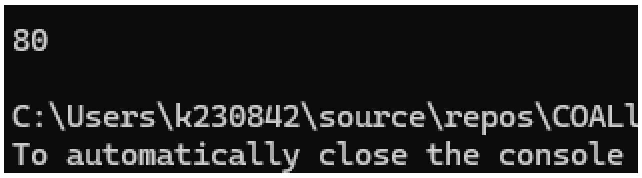
Write a program containing a procedure named **ArrayAvg** that calculates and displays the average value in an array. Pass a size-15 array by reference to this procedure.



```
INCLUDE Irvine32.inc

.data
array DWORD 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150
arraySize DWORD 15
sum DWORD 0
average DWORD 0

.code
main PROC

    mov esi,OFFSET array
    mov ecx, arraySize
    call ArrayAvg
    mov eax, average
    call WriteDec
    call Crlf

    exit
main ENDP
```



```
ArrayAvg PROC
    push eax
    push edx
    mov eax, 0
    mov edx, 0
loop_start:
    add eax, [esi]
    add esi, 4
    loop loop_start
    mov sum, eax
    mov eax, sum
    mov ecx, arraySize
    div ecx
    mov average, eax

    pop edx
    pop eax
    ret
ArrayAvg ENDP

END main
```

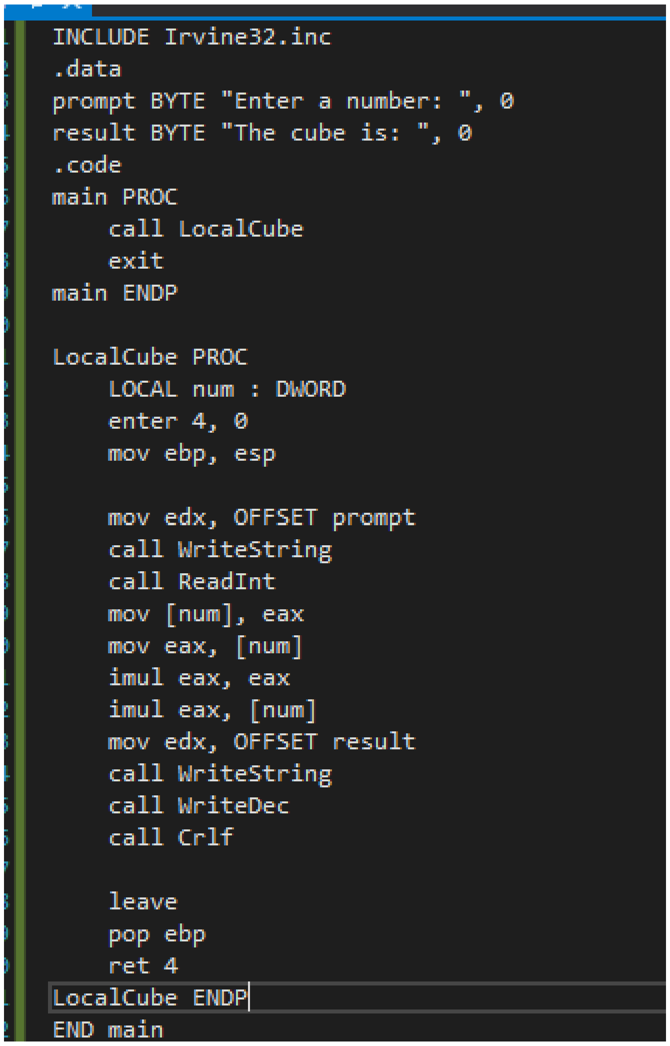


```
80

C:\Users\k230842\source\repos\COALl
To automatically close the console
```

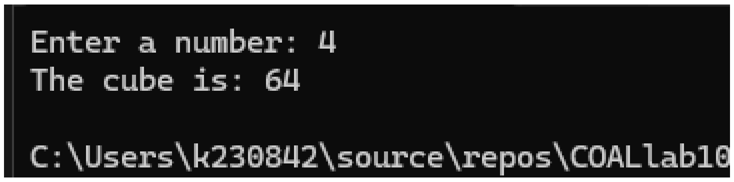
Average is: 80.

TASK 3

Write a program which contains a procedure named **LocalCube**. The procedure must declare a local variable. Initialize this variable by taking an input value from the user and then display its cube. Use **ENTER & *LEAVE*** instructions to allocate and de-allocate the local variable.



```
INCLUDE Irvine32.inc
.data
prompt BYTE "Enter a number: ", 0
result BYTE "The cube is: ", 0
.code
main PROC
    call LocalCube
    exit
main ENDP

LocalCube PROC
    LOCAL num : DWORD
    enter 4, 0
    mov ebp, esp

    mov edx, OFFSET prompt
    call WriteString
    call ReadInt
    mov [num], eax
    mov eax, [num]
    imul eax, eax
    imul eax, [num]
    mov edx, OFFSET result
    call WriteString
    call WriteDec
    call Crlf

    leave
    pop ebp
    ret 4
LocalCube ENDP
END main
```

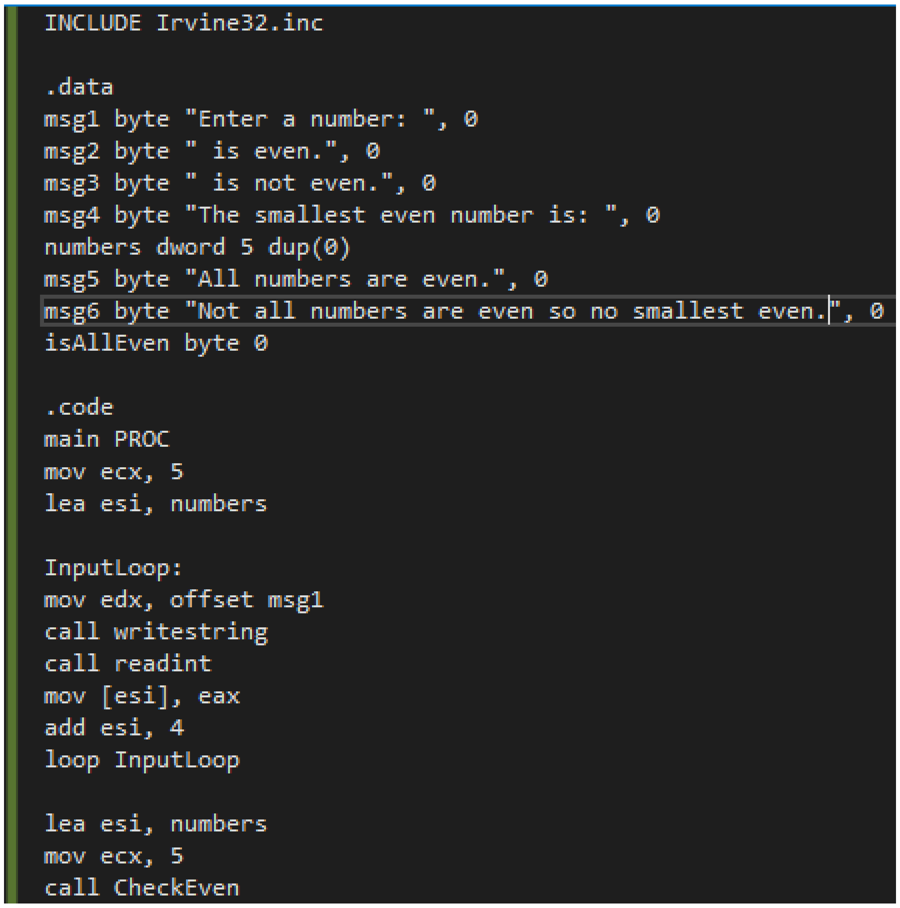


```
Enter a number: 4
The cube is: 64

C:\Users\k230842\source\repos\COALlab10
```

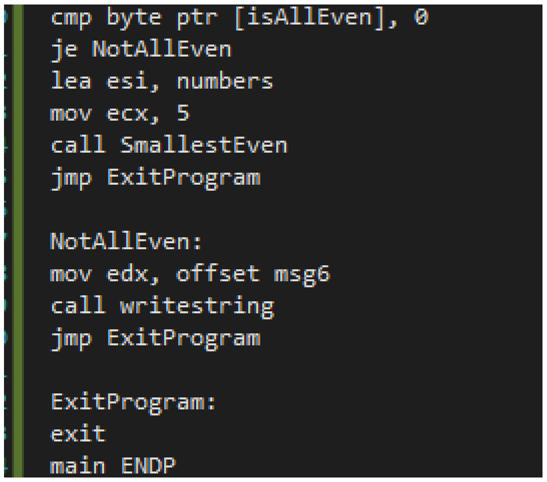
TASK 4

Write a program that takes 5 input numbers from the user. Then create two procedures, **CheckEven** and **SmallestEven**. The program should check if a given number is even. If all input numbers are even, the program should call the **SmallestEven** procedure.

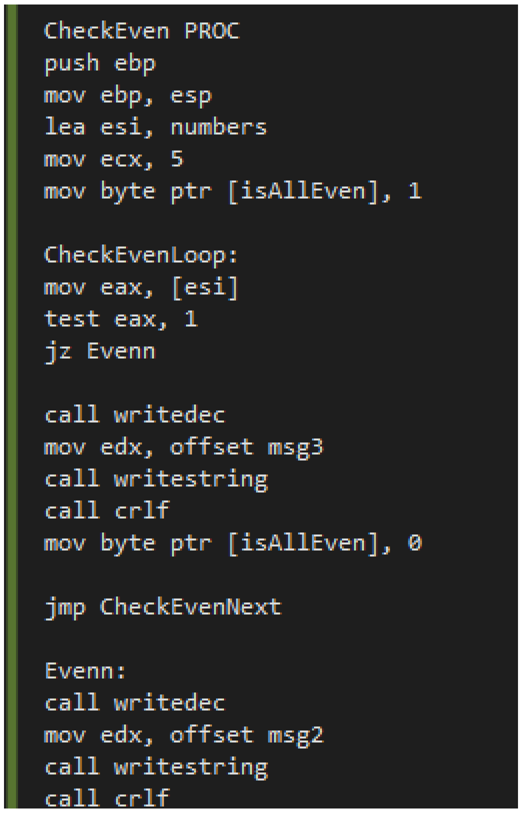
- **CheckEven**: Tests if a number is even or not.
- **SmallestEven**: Finds and displays the smallest of the five even numbers.



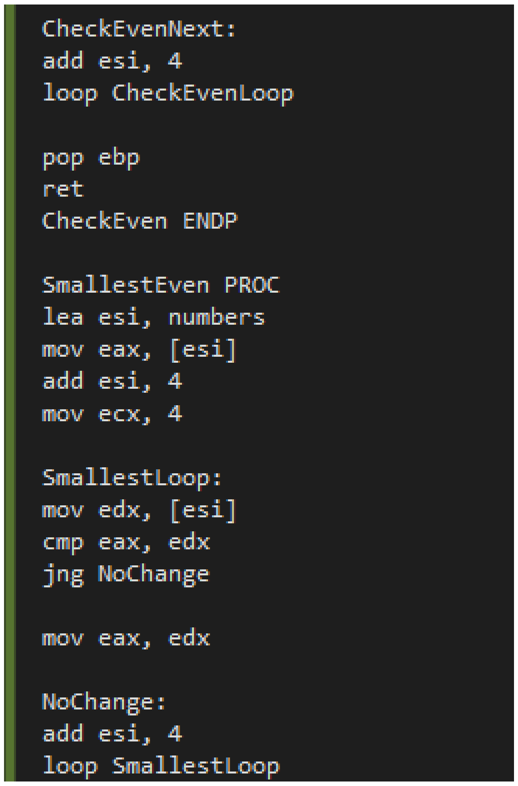
```
INCLUDE Irvine32.inc

.data
msg1 byte "Enter a number: ", 0
msg2 byte " is even.", 0
msg3 byte " is not even.", 0
msg4 byte "The smallest even number is: ", 0
numbers dword 5 dup(0)
msg5 byte "All numbers are even.", 0
msg6 byte "Not all numbers are even so no smallest even.", 0
isAllEven byte 0

.code
main PROC
mov ecx, 5
lea esi, numbers

InputLoop:
mov edx, offset msg1
call writestring
call readint
mov [esi], eax
add esi, 4
loop InputLoop

lea esi, numbers
mov ecx, 5
call CheckEven
```



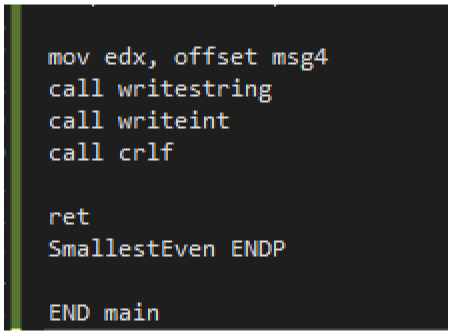
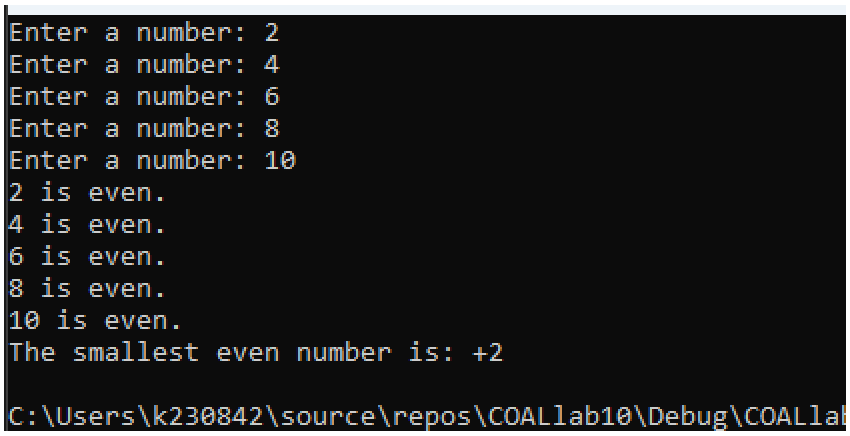
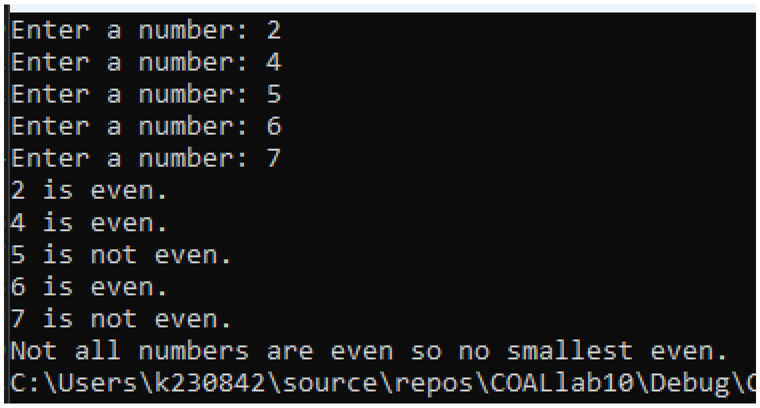
```
cmp byte ptr [isAllEven], 0
je NotAllEven
lea esi, numbers
mov ecx, 5
call SmallestEven
jmp ExitProgram

NotAllEven:
mov edx, offset msg6
call writestring
jmp ExitProgram

ExitProgram:
exit
main ENDP
```



```
CheckEven PROC
push ebp
mov ebp, esp
lea esi, numbers
mov ecx, 5
mov byte ptr [isAllEven], 1

CheckEvenLoop:
mov eax, [esi]
test eax, 1
jz Evenn

call writedec
mov edx, offset msg3
call writestring
call crlf
mov byte ptr [isAllEven], 0

jmp CheckEvenNext

Evenn:
call writedec
mov edx, offset msg2
call writestring
call crlf
```

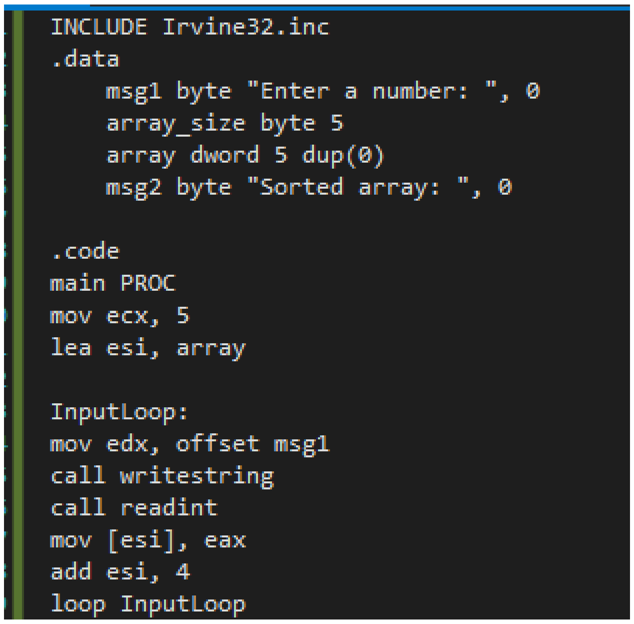


```
CheckEvenNext:
add esi, 4
loop CheckEvenLoop

pop ebp
ret
CheckEven ENDP

SmallestEven PROC
lea esi, numbers
mov eax, [esi]
add esi, 4
mov ecx, 4

SmallestLoop:
mov edx, [esi]
cmp eax, edx
jng NoChange

mov eax, edx

NoChange:
add esi, 4
loop SmallestLoop
```

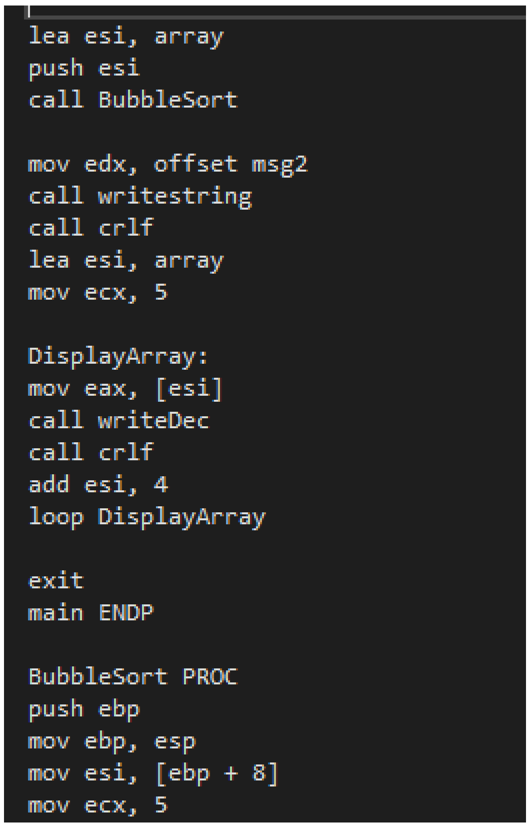

```
mov edx, offset msg4
call writestring
call writeint
call crlf

ret
SmallestEven ENDP

END main
```

```
Enter a number: 2
Enter a number: 4
Enter a number: 6
Enter a number: 8
Enter a number: 10
2 is even.
4 is even.
6 is even.
8 is even.
10 is even.
The smallest even number is: +2

C:\Users\k230842\source\repos\COALlab10\Debug\COALlal
```

```
Enter a number: 2
Enter a number: 4
Enter a number: 5
Enter a number: 6
Enter a number: 7
2 is even.
4 is even.
5 is not even.
6 is even.
7 is not even.
Not all numbers are even so no smallest even.
C:\Users\k230842\source\repos\COALlab10\Debug\C
```

TASK 5

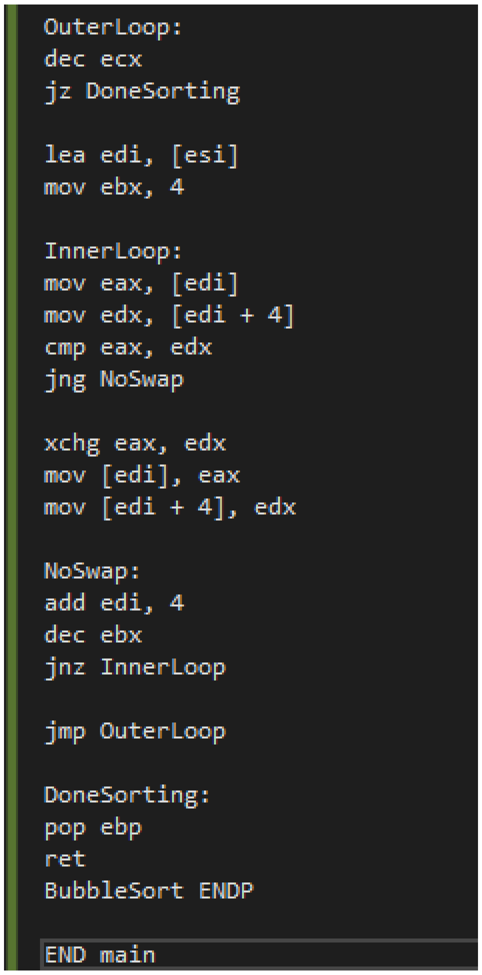
Write a program which contains a procedure named **BubbleSort** that sorts an array which is passed through a stack using indirect addressing.

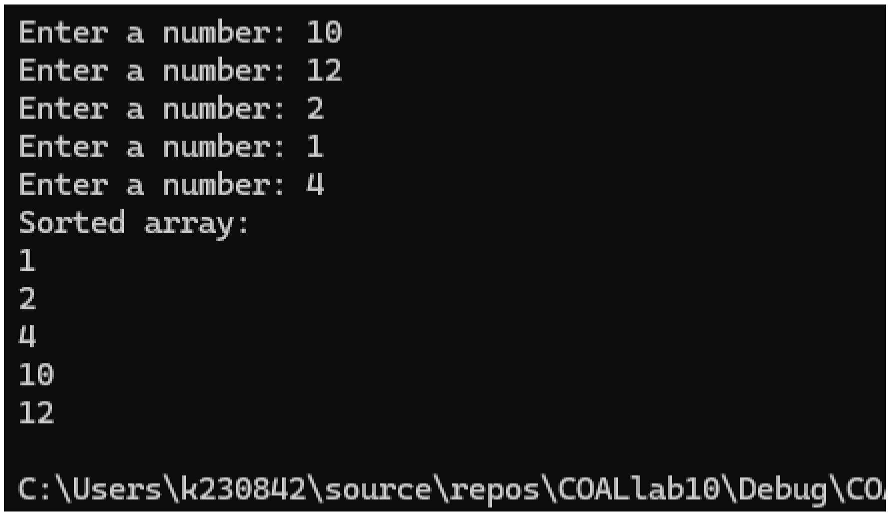


```
INCLUDE Irvine32.inc
.data
    msg1 byte "Enter a number: ", 0
    array_size byte 5
    array dword 5 dup(0)
    msg2 byte "Sorted array: ", 0

.code
main PROC
mov ecx, 5
lea esi, array

InputLoop:
mov edx, offset msg1
call writestring
call readint
mov [esi], eax
add esi, 4
loop InputLoop
```



```
lea esi, array
push esi
call BubbleSort

mov edx, offset msg2
call writestring
call crlf
lea esi, array
mov ecx, 5

DisplayArray:
mov eax, [esi]
call writeDec
call crlf
add esi, 4
loop DisplayArray

exit
main ENDP

BubbleSort PROC
push ebp
mov ebp, esp
mov esi, [ebp + 8]
mov ecx, 5
```


```
OuterLoop:
dec ecx
jz DoneSorting

lea edi, [esi]
mov ebx, 4

InnerLoop:
mov eax, [edi]
mov edx, [edi + 4]
cmp eax, edx
jng NoSwap

xchg eax, edx
mov [edi], eax
mov [edi + 4], edx

NoSwap:
add edi, 4
dec ebx
jnz InnerLoop

jmp OuterLoop

DoneSorting:
pop ebp
ret
BubbleSort ENDP

END main
```


```
Enter a number: 10
Enter a number: 12
Enter a number: 2
Enter a number: 1
Enter a number: 4
Sorted array:
1
2
4
10
12

C:\Users\k230842\source\repos\COALlab10\Debug\CO
```
Asking user for input values of array and then sorting them.