MUHAMMAD HAMMAD

23I0544

LAB 10

BCS-3D

TASK 01:  
include irvine32.inc

.data

var1 DWORD 3

var2 DWORD 4

var3 DWORD 5

.code

ThreeProd PROC

push ebp

mov ebp,esp

mov eax, [ebp+8]

mov ebx,[ebp+12]

mul ebx

mov ebx,[ebp+16]

mul ebx

call writeint

pop ebp

comment!

push EBP

mov EBP, ESP

mov eax,DWORD PTR [EBP + 12]

mov ebx,DWORD PTR [EBP + 8]

mul ebx

mov ebx,DWORD PTR [EBP + 4]

mul ebx

call WriteInt

pop EBP!

ret

ThreeProd ENDP

main PROC

push var3

push var2

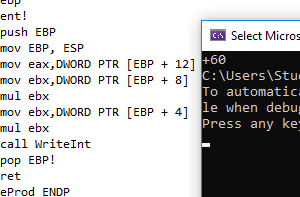
push var1

call ThreeProd

Exit

main ENDP

END main



TASK 02:

;2. Write a program which contains a procedure named MinMaxArray that displays the

;minimum & maximum values in an array. Pass a size-20 array by reference to this

;procedure.

include irvine32.inc

.data

arr DWORD 5, 23, 8, 11, 42, 3, 16, 9, 27, 6, 12, 35, 18, 7, 4, 19, 30, 10, 25, 2

ASize DWORD 20

prompt1 BYTE "Min: ", 0

prompt2 BYTE "Max: ", 0

.code

main PROC

push ASize

push OFFSET arr

call MinMaxArray

exit

main ENDP

MinMaxArray PROC

push ebp

mov ebp, esp

pushad

mov esi, [ebp + 8]

mov ecx, [ebp + 12]

mov eax, [esi]

mov ebx, [esi]

add esi, 4

dec ecx

L1:

mov edx, [esi]

cmp edx, eax

jl updateMin

cmp edx, ebx

jg updateMax

jmp Next

updateMin:

mov eax, edx

jmp Next

updateMax:

mov ebx, edx

Next:

add esi, 4

loop L1

call crlf

mov edx, OFFSET prompt1

call writestring

mov eax, eax

call writedec

call crlf

mov edx, OFFSET prompt2

call writestring

mov eax, ebx

call writedec

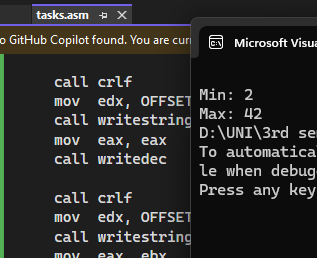
popad

pop ebp

ret 8

MinMaxArray ENDP

END main



TASK 03:

include irvine32.inc

.data

prompt1 BYTE "Enter number: ", 0

prompt2 BYTE "The square is: ", 0

.code

main PROC

call Square

exit

main ENDP

Square PROC

enter 4, 0

mov edx, OFFSET prompt1

call writestring

call readint

mov [ebp - 4], eax

mov eax, [ebp - 4]

imul eax, eax

mov edx, OFFSET prompt2

call writestring

call writedec

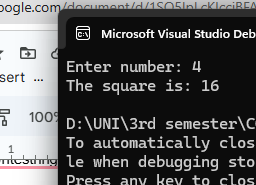
call crlf

leave

ret

Square ENDP

END main



TASK 04:  
include irvine32.inc

.data

prompt1 BYTE "Enter number: ", 0

prompt2 BYTE "Not all are prime.", 0

prompt3 BYTE "The largest prime number is: ", 0

nums DWORD 4 DUP(0)

.code

main PROC

mov ecx, 4

mov esi, OFFSET nums

Iloop:

mov edx, OFFSET prompt1

call writestring

call readint

mov [esi], eax

add esi, 4

loop Iloop

mov ecx, 4

mov esi, OFFSET nums

checkp:

mov eax, [esi]

call CheckPrime

cmp eax, 0

je notp

add esi, 4

loop checkp

call LargestPrime

exit

notp:

mov edx, OFFSET prompt2

call writestring

call crlf

exit

main ENDP

CheckPrime PROC

mov ebx, eax

cmp ebx, 2

je isprime

cmp ebx, 1

jle isnotprime

mov ecx, 2

check\_loop:

mov eax, ebx

cdq

div ecx

cmp edx, 0

je isnotprime

inc ecx

mov eax, ecx

mul eax

cmp eax, ebx

jb check\_loop

isprime:

mov eax, 1

ret

isnotprime:

mov eax, 0

ret

CheckPrime ENDP

LargestPrime PROC

mov ecx, 4

mov esi, OFFSET nums

mov eax, 0

findlargest:

mov ebx, [esi]

call CheckPrime

cmp eax, 1

jne skip

cmp ebx, eax

jle skip

mov eax, ebx

skip:

add esi, 4

loop findlargest

cmp eax, 0

je primeNotfound

mov edx, OFFSET prompt3

call writestring

call writeint

call crlf

ret

primeNotfound:

mov edx, OFFSET prompt2

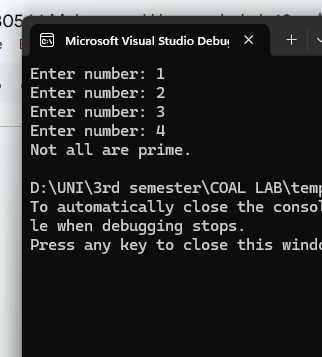
call writestring

call crlf

ret

LargestPrime ENDP

end main



TASK 05:  
include irvine32.inc

.data

ASize DWORD 5

array DWORD 9, 5, 8, 3, 2

prompt1 BYTE "Unsorted: ", 0

prompt2 BYTE "Sorted: ", 0

.code

main PROC

mov edx, OFFSET prompt1

call writestring

call Displayarr

call crlf

push OFFSET array

push ASize

call BubbleSort

mov edx, OFFSET prompt2

call writestring

call Displayarr

call crlf

exit

main ENDP

BubbleSort PROC

push ebp

mov ebp, esp

mov ecx, [ebp+8]

mov esi, [ebp+12]

dec ecx

outerL:

mov edx, ecx

mov edi, esi

innerL:

mov eax, [edi]

mov ebx, [edi+4]

cmp eax, ebx

jle nswap

mov [edi], ebx

mov [edi+4], eax

nswap:

add edi, 4

dec edx

jnz innerL

dec ecx

jnz outerL

pop ebp

ret

BubbleSort ENDP

Displayarr PROC

push ebp

mov ebp, esp

mov ecx, ASize

mov esi, OFFSET array

displayL:

mov eax, [esi]

call writedec

call crlf

add esi, 4

loop displayL

pop ebp

ret

Displayarr ENDP

END main  
