Q6:

INCLUDE Irvine32.inc

.data

str1 BYTE "hello",0

str2 BYTE "hello",0

.code

main PROC

mov esi, OFFSET str1

mov edi, OFFSET str2

mov ecx, LENGTHOF str1

CompareLoop:

mov al, [esi]

cmp al, [edi]

jne NotEqual

inc esi

inc edi

loopne CompareLoop

cmp ecx, 0

jne NotEqual

mov eax, 1

jmp Done

NotEqual:

mov eax, 0

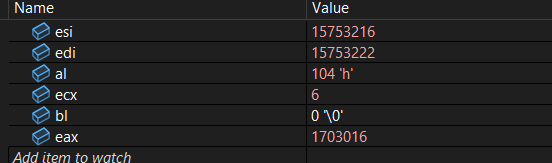
Done:

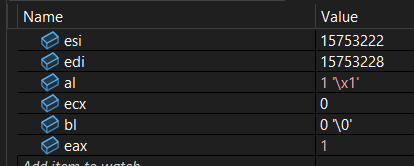
call DumpRegs

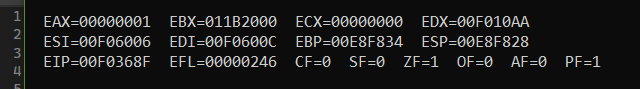
exit

main ENDP

END main







Q1:

INCLUDE Irvine32.inc

.data

arr BYTE "hello world",0

arr2 BYTE ?

.code

main PROC

mov esi, OFFSET arr

mov edi, OFFSET arr2

NextWord:

mov al, [esi]

;----------- end loop -----------

cmp al, 0

je Done

;--------- skip spaces -------

cmp al, ' '

je PrintSpace

;--------- find start of word

mov ebx, esi

FindEnd:

mov al, [esi]

cmp al, ' '

je Reverse

cmp al, 0

je Reverse

inc esi

jmp FindEnd

Reverse:

mov edi, esi ; EDI = end of word (space or null)

dec edi ; step back to last character

PrintBackward:

mov al, [edi]

cmp edi, ebx ; reached start?

je AfterWord

dec edi

jmp PrintBackward

AfterWord:

cmp byte ptr [esi], 0

je Done ; if end of string, exit

inc esi ; skip the space

jmp NextWord

PrintSpace:

inc esi

jmp NextWord

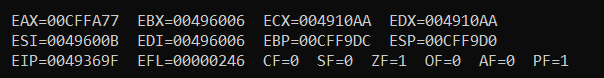
Done:

call DumpRegs

exit

main ENDP

END main



include irvine32.inc

.data

arr DWORD 1,51,2,35,2,4,3,5,6,33

arr1 DWORD 1,1,1,1,1,1,1,1,1,1

.code

main PROC

mov ecx, 10

mov esi,OFFSET arr

mov edi,OFFSET arr1

l1:

mov ebx,[esi]

cmp ebx,0

jl Negative

add esi,4

Positive:

add ebx,5

jmp Store

Negative:

sub ebx,3

Store:

mov [edi],ebx

add edi,4

add esi,4

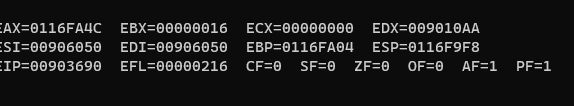
loop l1

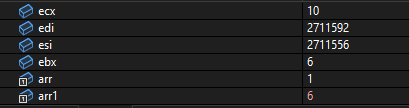
call DumpRegs

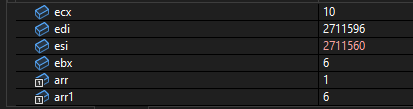
exit

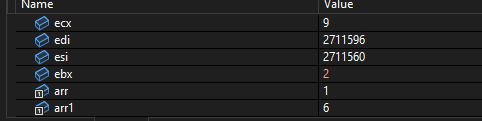
main ENDP

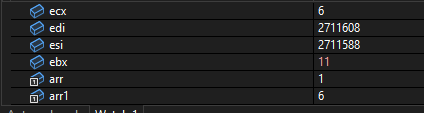
END main



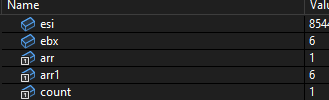








Count:

****

INCLUDE Irvine32.inc

.data

ALIGN 8

num1 DWORD 0AABBCCDDh

num2 WORD 0EEFFh

num3 BYTE 11h

newVal DWORD ?

.code

main PROC

mov eax, num1

mov BYTE PTR [num1+1], 00h

mov ebx, 0

mov bx, num2

mov eax, 256

mul ebx ; eax = ebx \* 256

mov ebx, eax

mov ecx, 0

mov cl, num3

or ebx, ecx

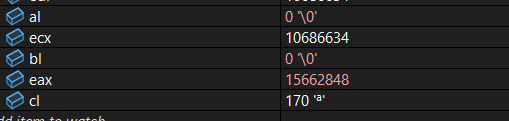
mov newVal, ebx

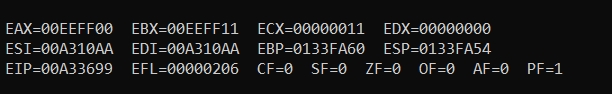
call DumpRegs

exit

main ENDP

END main

****

****