Q1

INCLUDE Irvine32.inc

.data

str1 db "Assembly is Awesome"

len dd lengthof str1

.code

main PROC

mov esi, offset str1

mov edi,esi

add edi,len

mov ecx, 9

loop1:

mov al,[esi]

mov bl,[edi]

cmp al, bl

je Exitcode

mov[esi], bl

mov [edi], al

add esi,1

sub edi,1

loop loop1

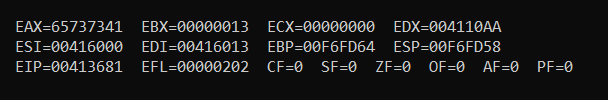
Exitcode:

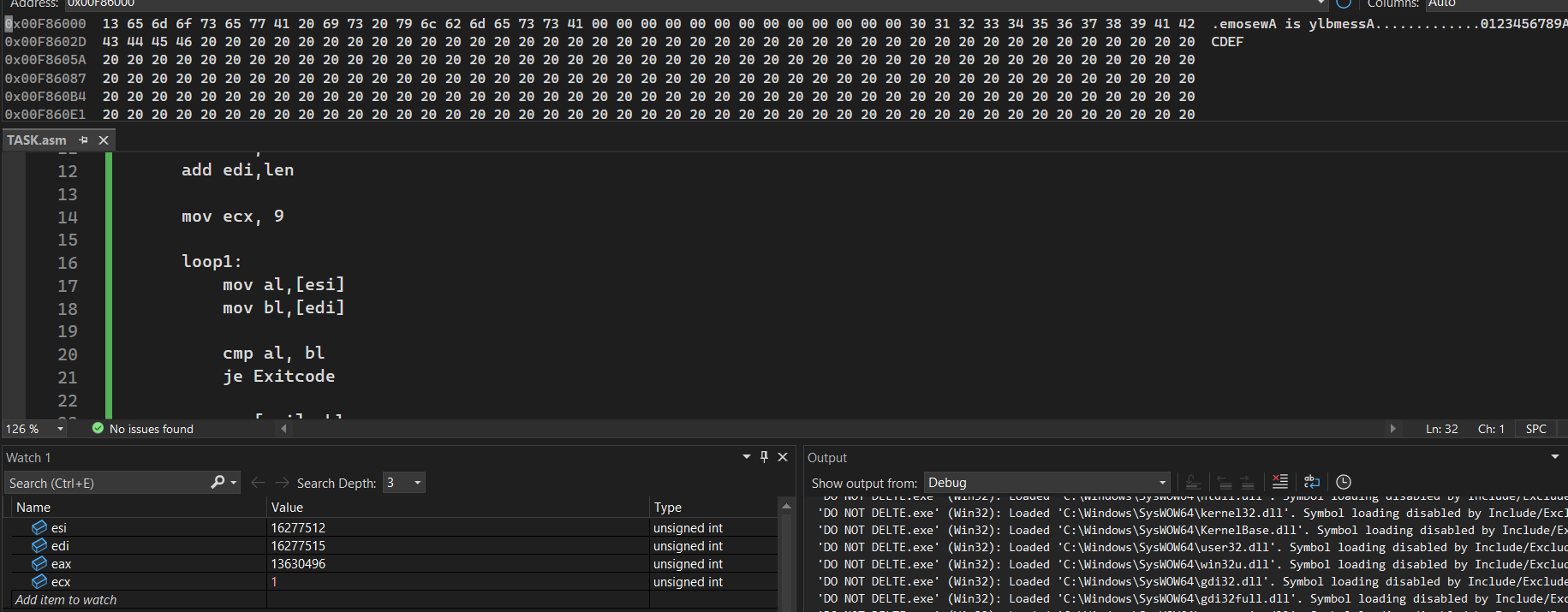
call DumpRegs

exit

main ENDP

END main





Q2

INCLUDE irvine32.inc

.data

str1 db "SmalL And LaRGE",0

.code

main PROC

mov esi, offset str1

loop1:

mov al, [esi]

cmp al,0

je exit\_label

cmp al, 'A'

jb next

cmp al, 'Z'

jbe uppercase

cmp al, 'a'

jb next

cmp al, 'z'

jbe lowercase

jmp next

uppercase:

add al, 32

mov [esi], al

jmp next

lowercase:

sub al, 32

mov [esi], al

next:

inc esi

jmp loop1

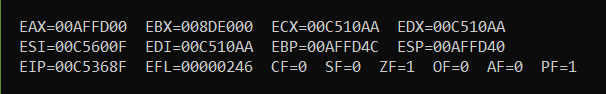
exit\_label:

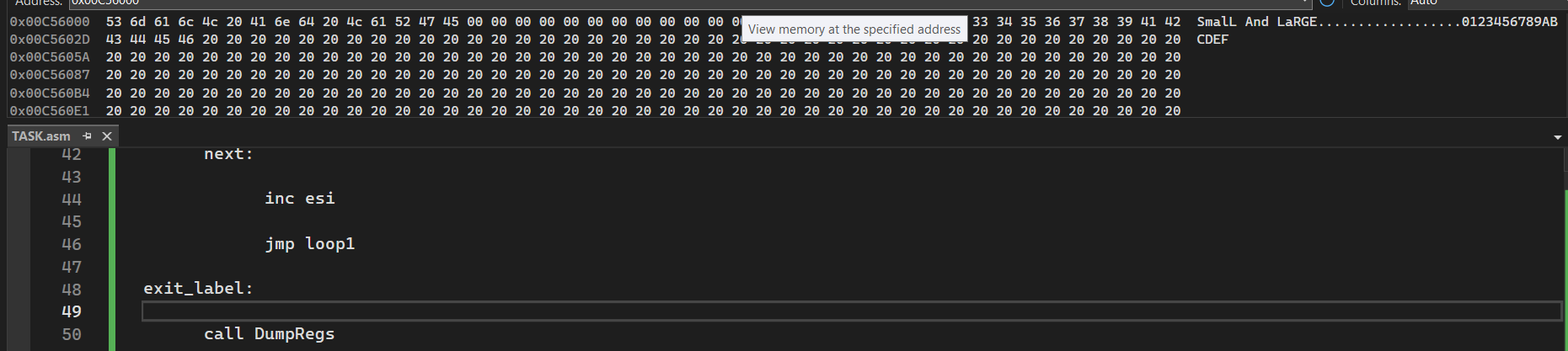
call DumpRegs

exit

main endp

end main





Q3

INCLUDE Irvine32.inc

.data

str1 db "Hello Assembly",0

str2 db "With Assembly",0

.code

main PROC

mov esi, OFFSET str1

mov edi, OFFSET str2

find\_end:

mov al, [esi]

cmp al, 0

je start\_copy ; found end of str1

inc esi

jmp find\_end

; Copy str2 into str1 starting from null terminator

start\_copy:

mov al, [edi]

mov [esi], al

inc edi

inc esi

cmp al, 0 ; stop after copying null

jne start\_copy

mov edx, OFFSET str1

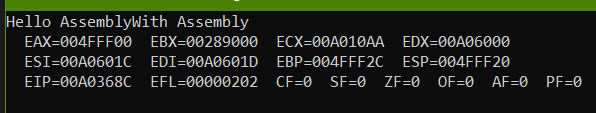
call WriteString

call DumpRegs

exit

main ENDP

END main



INCLUDE Irvine32.inc

.data

str1 db "Hello World",0

str2 db "Hello World",0

.code

main PROC

mov esi, OFFSET str1

mov edi, OFFSET str2

compare:

mov al, [esi]

mov bl, [edi]

cmp al, bl

jne not\_equal

cmp al, 0

je equal

inc esi

inc edi

jmp compare

equal:

mov al, 1

jmp done

not\_equal:

mov al, 0

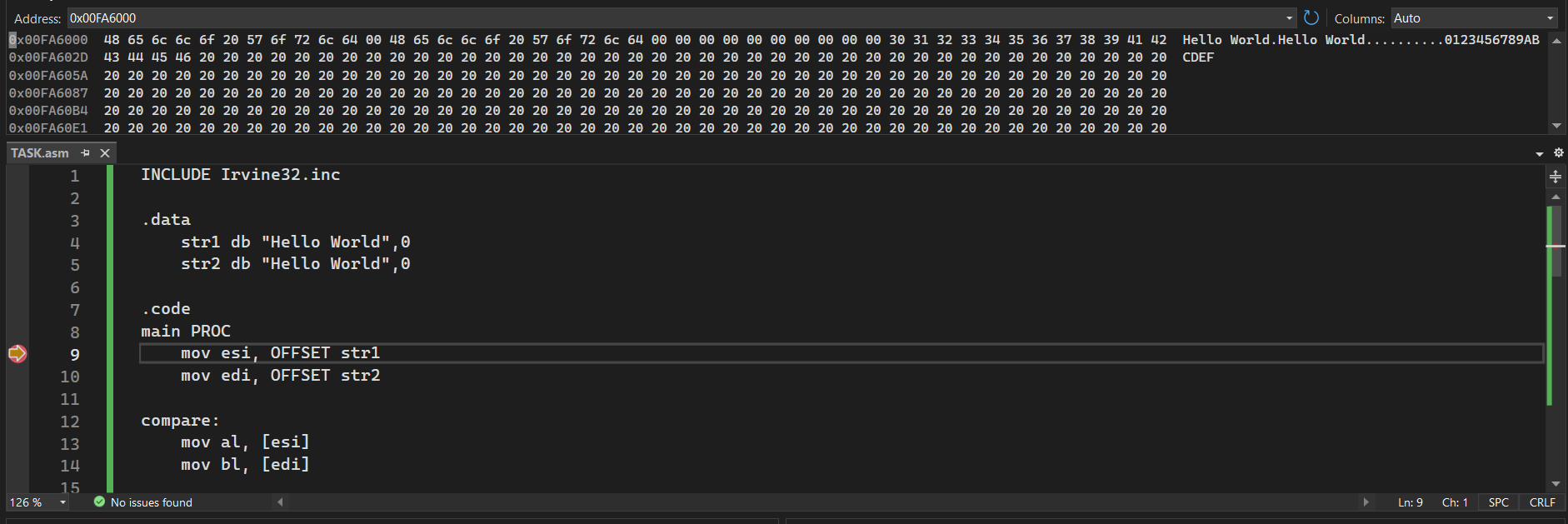
done:

call DumpRegs

exit

main ENDP

END main



Task 5:

INCLUDE Irvine32.inc

.data

arr WORD 1,5,7,3,7,5,1

.code

main PROC

mov esi, OFFSET arr

mov edi, OFFSET arr

add edi,12

l1:

mov ax, [esi]

mov bx, [edi]

cmp ax, bx

jne NOpalindrome

cmp esi, edi

jl palindrome

add esi, 2

sub edi, 2

NOpalindrome:

mov ax, 0

jmp done

palindrome:

mov ax, 1

jmp done

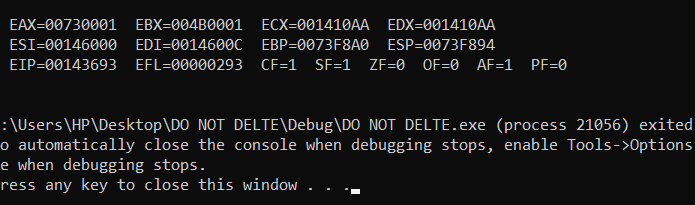
done:

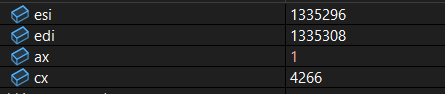
call DumpRegs

exit

main ENDP

END main





Q6 (bubble sort)

INCLUDE Irvine32.inc

.data

arr DWORD 2,1,5,0,4,8,3,7,1,4,7

.code

main PROC

mov ecx, 10 ; outer loop counter

mov esi,OFFSET arr

l1:

mov ebx,9

l2:

mov eax, [esi]

mov edx,[esi+4]

cmp eax,edx

jle NoSwap

mov[esi],edx

mov[esi+4],eax

NoSwap:

add esi, 4

dec ebx

cmp ebx, 0

jne l2

loop l1

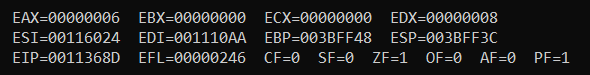
done:

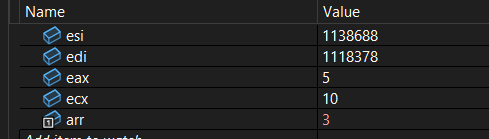
call DumpRegs

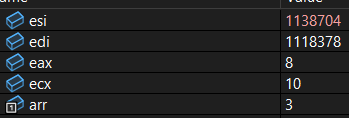
exit

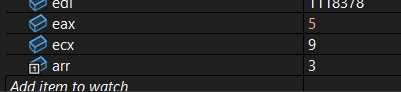
main ENDP

END main









Q7

INCLUDE Irvine32.inc

.data

arr DWORD 2,1,5,0,4,8,3,7,1,4,7

smallestsecond DWORD ?

GREATESTsecond DWORD ?

.code

main PROC

mov ecx, 10 ; outer loop counter

mov esi,OFFSET arr

l1:

mov ebx,9

l2:

mov eax, [esi]

mov edx,[esi+4]

cmp eax,edx

jle NoSwap

mov[esi],edx

mov[esi+4],eax

NoSwap:

add esi, 4

dec ebx

cmp ebx, 0

jne l2

loop l1

; now the array array is sorted just extract 2nd smallest and largest values

mov eax,[arr+4]

mov ebx,[arr+36]

mov smallestsecond ,eax

mov GREATESTsecond, ebx

done:

call DumpRegs

exit

main ENDP

END main

