National University of Computer and Emerging Sciences



Laboratory Manual

for

Computer Organization and Assembly Language

**Department of Computer Science**

# COAL Lab 7 Manual

|  |
| --- |
| Objectives:  * Revision * Problems & Assignments |

## Problem(s) / Assignment(s)

|  |  |
| --- | --- |
| **Discussion & Practice** | **Estimated completion time: 1 hr, 30 mins** |

|  |  |
| --- | --- |
| **Problem 7.1:** *Reverse an Array*  Use a loop with indirect or indexed addressing to reverse the elements of an integer array in place. Do not copy the elements to any other array. Use the SIZEOF, TYPE, and LENGTHOF operators to make the program as flexible as possible if the array size and type should be changed in the future. Display the modified array by calling the DumpMem. | **Estimated completion time:20 mins** |

|  |  |
| --- | --- |
| **Problem 7.2:** *Reversing a String*  Write a program using the LOOP instruction with indirect addressing that copies a string from source to target, reversing the character order in the process. Use the following variables:  **source BYTE "This is the source string",0**  **target BYTE SIZEOF source DUP('#')**  Use DumpMem to display the string. If your program works correctly, it will display the following sequence of hexadecimal bytes:  67 6E 69 72 74 73 20 65 63 72 75 6F 73 20 65 68  74 20 73 69 20 73 69 68 54 | **Estimated completion time:15 mins** |

Problem 7.1: Reverse an Array

Use a loop with indirect or indexed addressing to reverse the elements of an integer array in place. Do not copy the elements to any other array. Use the SIZEOF, TYPE, and LENGTHOF operators to make the program as flexible as possible if the array size and type should be changed in the future. Display the modified array by calling the DumpMem.

***CODE:***

include irvine32.inc

.data

arra DB 1,2,3,4

temp DB ?

.code

main proc

mov esi, OFFSET arra

mov bl,[esi]

mov ECX, LENGTHOF arra

DEC CX

l1:

ADD esi,1

loop l1

MOV Al, [esi]

mov [esi],bl

mov esi,OFFSET arra

mov [esi],Al

ADD esi,1

mov bl,[esi]

mov ecx, LENGTHOF arra

sub esi,1

DEC CX

l2:

ADD esi,1

loop l2

sub esi,1

mov Al,[esi]

mov [esi],bl

mov esi, OFFSET arra

ADD esi ,1

mov [esi], Al

mov esi, OFFSET arra

mov ecx, LENGTHOF arra

mov ebx, TYPE arra

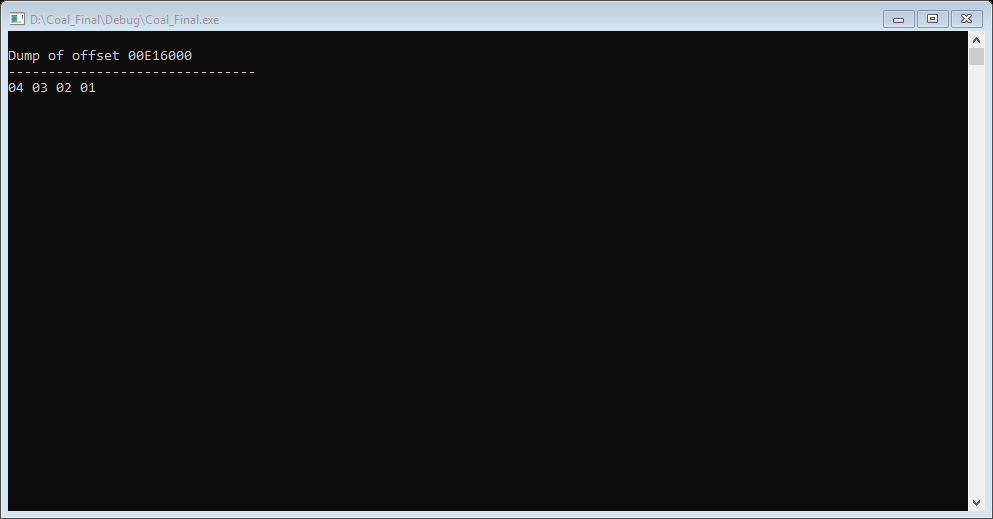
call DumpMem

exit

main endp

end main

**SCREENSHOT:**

****

***CODE:***

include irvine32.inc

.data

source BYTE "This is a source string",0

target BYTE SIZEOF source DUP ('#')

.code

main proc

mov esi, OFFSET source

mov edi, OFFSET target

mov ecx, LENGTHOF source

Dec cx

l1:

ADD esi,1

loop l1

mov ECX,LENGTHOF source

l2:

mov al,[edi]

mov bl,[esi]

xchg al,bl

ADD edi,1

sub edi,1

loop l2

mov esi, OFFSET target

mov ecx, LENGTHOF target

mov ebx, TYPE target

call DumpMem

exit

main endp

end main