|  |  |  |
| --- | --- | --- |
| Week | Reverse Engineering Malware | Duration |
| 3 | Data Transfers, Addressing, and Arithmetic | 120 mins |

Marks allocation: 8/100 for CA practical submission

**Lesson Objectives**

* Write and execute assembly language programs using Arrays and Loops

1. Integer Expression Calculation

Using the AddTwo program as a reference, write a program that calculates the following expression, using registers: A = (A + B) − (C + D). Assign integer values to the EAX, EBX, ECX, and EDX registers.

**.386**

**.model flat,stdcall**

**.stack 4096**

**ExitProcess proto,dwExitCode:dword**

**.data**

**varA dword 30**

**varB dword 30**

**varC dword 20**

**varD dword 20**

**.code**

**main proc**

**mov eax, varA**

**mov ebx, varB**

**mov ecx, varC**

**mov edx, varD**

**add eax, ebx**

**add ecx , edx**

**sub eax , ecx**

**invoke ExitProcess,0**

**main endp**

**end main**

2. Exchanging Pairs of Array Values

Write a program with a loop and indexed addressing that exchanges every pair of values in an array with an even number of elements. Therefore, item i will exchange with item i+1, and item i+2 will exchange with item i+3, and so on.

**.386**

**.model flat,stdcall**

**.stack 4096**

**ExitProcess proto,dwExitCode:dword**

**.data**

**array DWORD 1,2,3,4,5,6,7,8,9,10,11,12**

**.code**

**main PROC**

**; point to the first two array elements**

**mov esi,0 ; beginning of array**

**mov ecx,LENGTHOF array / 2 ; loop (N / 2) times**

**; The loop swaps adjacent array elements, gradually**

**; moving towards the last pair.**

**L1:**

**; exchange array[esi] with array[edi], using indexed addressing.**

**mov eax,array[esi]**

**xchg eax,array[esi+4]**

**mov array[esi],eax**

**add esi,TYPE array\*2 ; pointer moves forward**

**loop L1**

**invoke ExitProcess,0**

**main endp**

**end main**

3. Copying a Word Array to a DoubleWord array

Write a program that uses a loop to copy all the elements from an unsigned Word (16-bit) array into an unsigned doubleword (32-bit) array.

**.386**

**.model flat,stdcall**

**.stack 4096**

**ExitProcess proto,dwExitCode:dword**

**.data**

**source word 1,2,3,4,5,6,7,8,9,10**

**count = LENGTHOF source**

**target dword count DUP(?)**

**.code**

**main proc**

**mov ecx,count**

**mov esi,0 ; source**

**mov edi,0 ; target**

**L1:**

**movzx eax,source[esi]**

**mov target[edi],eax**

**add esi,2**

**add edi,4**

**loop L1**

**invoke ExitProcess,0**

**main endp**

**end main**

END