CSC 3210

Spring 22

TTA

Set B

Solution

1. Consider the following code. What is the content of AX and BX register after executing the code. Explain your answer. (2 points for correct answer, 5 points for explanation)

.data

Array WORD 3344h, 1122h, 8866h

.code

mov esi, 0

mov edi, OFFSET Array+4

mov ax, word PTR Array+3

mov bx, [edi]

Answer:

First store the array items in Little Endian order. Below each box contains 8-bit or 1 byte data.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | 33 | 22 | 11 | 66 | 88 |  |  |

Esi contains 0.

Mov edi, OFFSET Array+4

Array+4 makes a 4-byte jump from the beginning of the array and reach 66.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | 33 | 22 | 11 | 66 | 88 |  |  |

Offset operator returns the address of 66 and then mov operation stores that in edi register.

Edi register contains the address of 66.

Mov ax, word ptr Array+3

Here Array+1 takes you to the 11.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | 33 | 22 | 11 | 66 | 88 |  |  |

Then word ptr extracts 2 bytes from 11.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | 33 | 22 | 11 | 66 | 88 |  |  |

Because computer is little endian, it will read the data in reverse order: 66 11

Ax contains 66 11

mov bx, [edi]

Here you dereference the edi register that contains the address of 66. The destination in the mov operation is bx (2-byte long). So it will take you to the 22 and extract 2 bytes from there.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | 33 | 22 | 11 | 66 | 88 |  |  |

Because computer is little endian, it reads the numbers in reverse order: 88 66

Bx contains 8866

1. Translate the following code into assembly code(masm). You can use Mov,Loop, Add, Sub instructions to complete the translation (8 points).

int var1 = 10

int var2 = 0;

for (i = 7; i > 0; i--){

var2 = 0;

for (j=10; j > 0; j--){

var1 = var1 + var2

}

Var1 = var2 + 10

}

Solution:

You can store the data as either byte, word or dword

.data

var1 DWORD 10

var2 DWORD 0

.code

mov ecx, 7 ; initialize outer loop counter

L1:

mov edx, ecx ; keep a back up of ecx

mov ecx, 10 ; initialize inner loop counter

L2:

mov eax, var2

add var1, eax

Loop L2

mov ecx, edx ; restore outer loop counter

mov eax, var2

add eax, 10

mov var1, eax

Loop L1