Assembly Homework 3

Department: CSIE 2-B

Student Number: 111502539

Name: 沈以捷

DATA SECTION:

array1, array2: two arrays to pass into *CountMatches.*

```
6   .data
7       array1 SDWORD 10, 5, 4, -6, 2, 11, 12
8       array2 SDWORD 10, 5, 3, 1, 4, 2, -6
```

THOUGHT:

Use two registers to store the pointers of the arrays, and compare them. If they are equal, increase the counter (***eax***).

MAIN PROC:

Print out the registers before and after calling *CountMatches* to see if the registers remain the same.

```
11  MAIN PROC
12      call DumpRegs       ; print registers
13
14      invoke CountMatches, OFFSET array1, OFFSET array2, LENGTHOF array1
15      call WriteInt       ; print eax
16      call Crlf           ; new line
17
18      call DumpRegs       ; print registers
19
20      call WaitMsg        ; wait for input
21      exit
22
23  MAIN ENDP
```

CountMatches PROC:

```
25  CountMatches PROC,
26      aptr1: PTR SDWORD, aptr2: PTR SDWORD, arraySize: DWORD
27
28      push esi    ; push esi to stack
29      push edi    ; push edi to stack
30      push ebx    ; push ebx to stack
31      push ecx    ; push ecx to stack
32
33      mov esi, aptr1
34      mov edi, aptr2
35      mov ecx, arraySize
36      mov eax, 0
37
38      L1:
39          mov ebx, [esi]      ; ebx = elements in array1
40          cmp ebx, [edi]      ; compare ebx with elements in array2
41          jne Final           ; jump to Final if not equal
42
43      Equal:
44          inc eax             ; eax += 1 if equal
45
46      Final:
47          add esi, 4          ; esi += 4
48          add edi, 4          ; edi += 4
49          LOOP L1
50
51      pop ecx     ; pop ecx from stack
52      pop ebx     ; pop ebx from stack
53      pop edi     ; pop edi from stack
54      pop esi     ; pop esi from stack
55
56      ret
57  CountMatches ENDP
```

Line 28-31, 33-36, 51-54:

Registers used:

esi: store the addresses of the elements in array1.

edi: store the addresses of the elements in array2.

eax: match count.

ebx: store elements in array1 for temporary comparison.

ecx: store the length of the array for loop count.

Push and pop the registers used by the procedure.

```
28      push esi    ; push esi to stack
29      push edi    ; push edi to stack
30      push ebx    ; push ebx to stack
31      push ecx    ; push ecx to stack
```

```
51      pop ecx     ; pop ecx from stack
52      pop ebx     ; pop ebx from stack
53      pop edi     ; pop edi from stack
54      pop esi     ; pop esi from stack
```

Move the arguments to registers

```
33      mov esi, aptr1
34      mov edi, aptr2
35      mov ecx, arraySize
36      mov eax, 0
```

Line 38-49:

Move the element in array1 to ***ebx***, then compare it with the element in array2. If they are equal, then increase ***eax*** by 1. If not, increase address registers by 4, since the size of double word is 4, then loop.

```
38      L1:
39          mov ebx, [esi]      ; ebx = elements in array1
40          cmp ebx, [edi]      ; compare ebx with elements in array2
41          jne Final           ; jump to Final if not equal
42
43      Equal:
44          inc eax             ; eax += 1 if equal
45
46      Final:
47          add esi, 4          ; esi += 4
48          add edi, 4          ; edi += 4
49          LOOP L1
```

RESULT:

4 registers used by the procedure remains the same. And ***eax*** is correctly printed out as 2.

```
  EAX=001BFDFC  EBX=005BF000  ECX=00291113  EDX=00291113
  ESI=00291113  EDI=00291113  EBP=001BFDB4  ESP=001BFDA8
  EIP=00293A75  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

+2

  EAX=00000002  EBX=005BF000  ECX=00291113  EDX=00291113
  ESI=00291113  EDI=00291113  EBP=001BFDB4  ESP=001BFDA8
  EIP=00293A95  EFL=00000202  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=0

Press any key to continue...
```