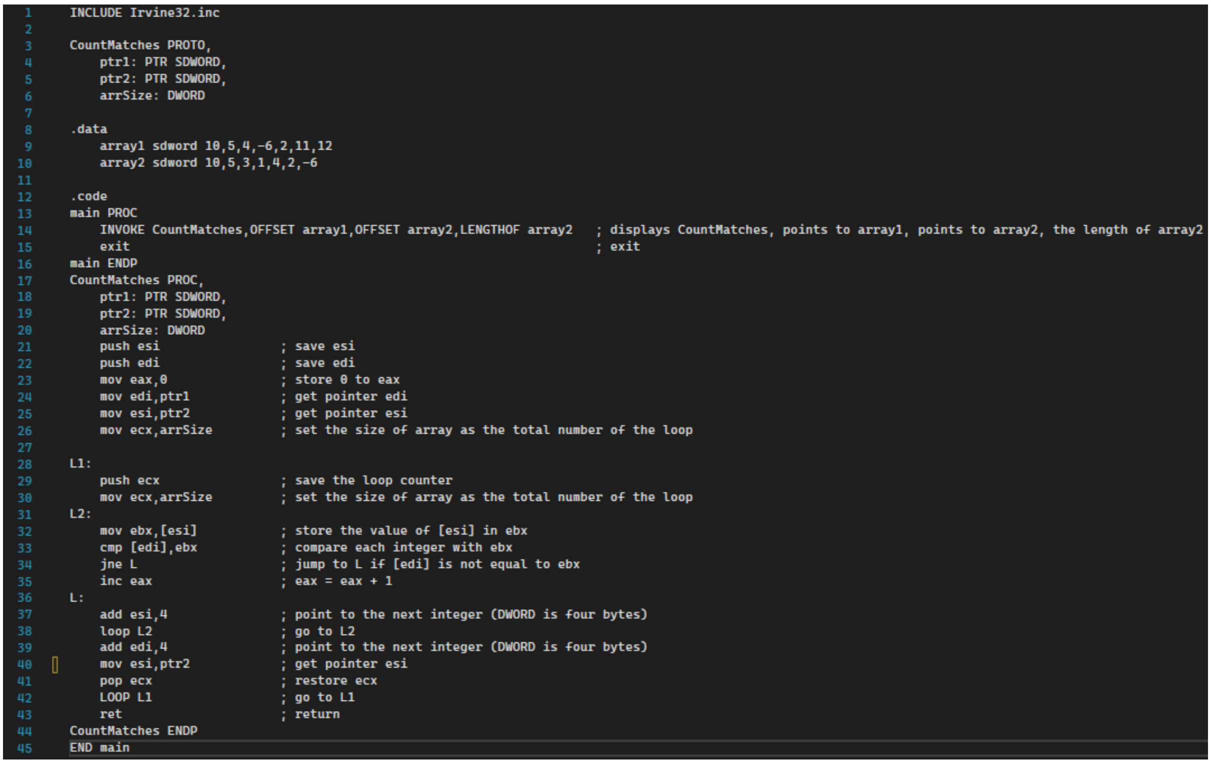
Title: Assembly Language HW3 Bonus Function
Name / student ID: 曾千芸109504501

- **Screenshots of result and code + explanations**



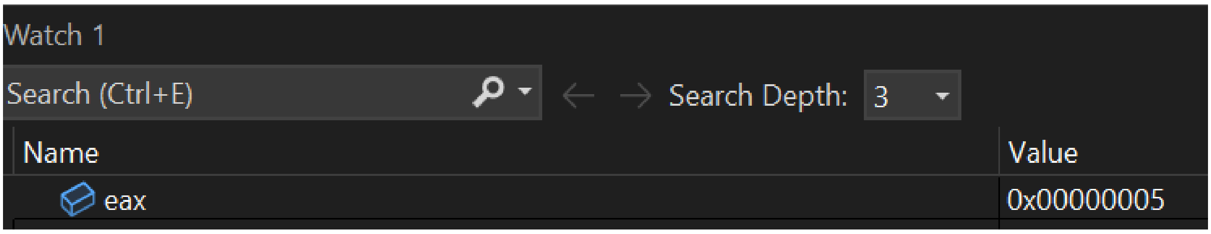
```
INCLUDE Irvine32.inc

CountMatches PROTO,
    ptr1: PTR SDWORD,
    ptr2: PTR SDWORD,
    arrSize: DWORD

.data
    array1 sdword 10,5,4,-6,2,11,12
    array2 sdword 10,5,3,1,4,2,-6

.code
main PROC
    INVOKE CountMatches,OFFSET array1,OFFSET array2,LENGTHOF array2   ; displays CountMatches, points to array1, points to array2, the length of array2
    exit                                                              ; exit
main ENDP
CountMatches PROC,
    ptr1: PTR SDWORD,
    ptr2: PTR SDWORD,
    arrSize: DWORD
    push esi                ; save esi
    push edi                ; save edi
    mov eax,0               ; store 0 to eax
    mov edi,ptr1            ; get pointer edi
    mov esi,ptr2            ; get pointer esi
    mov ecx,arrSize         ; set the size of array as the total number of the loop

L1:
    push ecx                ; save the loop counter
    mov ecx,arrSize         ; set the size of array as the total number of the loop
L2:
    mov ebx,[esi]           ; store the value of [esi] in ebx
    cmp [edi],ebx           ; compare each integer with ebx
    jne L                   ; jump to L if [edi] is not equal to ebx
    inc eax                 ; eax = eax + 1
L:
    add esi,4               ; point to the next integer (DWORD is four bytes)
    loop L2                 ; go to L2
    add edi,4               ; point to the next integer (DWORD is four bytes)
    mov esi,ptr2            ; get pointer esi
    pop ecx                 ; restore ecx
    LOOP L1                 ; go to L1
    ret                     ; return
CountMatches ENDP
END main
```



Name	Value
eax	0x00000005

- **main idea**

1. store array1, array2 and the size of the array to CountMatches function
2. apply two-dimensional loop to compare whether each position of the array are the same, then add one to eax

- **detailed explanation**

1. edi is where array1 point to
2. esi is where array2 point to
3. ecx is the total number of the loop which means how many times the program is going to compare
4. set eax to zero at the beginning of the program
5. After applying the external loop L1 which in the two-dimensional loop, the program will do the following steps. First, push ecx into the stack. Second, applying internal loop L2 until ecx equals to zero and end L2. Then, pop its value into ecx. Lastly, go to the next internal loop L2 until ecx in the external loop equals to zero.
6. after the comparison, if both value of esi and edi are the same, then add one to eax, else jump to L and go back to L2

- **Thoughts about the HW3 Bonus:**
  To be honest, I couldn’t figure out what is the difference between basic function and the bonus one. The program kept reporting only three matches no matter how I fix the code. At the end of the day, I finally figure out that I should try to think of two-dimensional loop. Thus, I started to solve it in a different way and eventually I succeeded.