HW # 5:  Theme: Data Definitions, Addressing Modes, Arrays

*All main questions carry equal weight.*

*(Credit awarded to only those answers for which work has been shown.)*

1. [Memory Map] Fill in the following memory diagram with the data provided below. Please assume that the data segment begins at 0x0065A300.

.data  
Rose       BYTE       0B9h  
Magnolia   WORD       0E54Eh  
Cannas   DWORD     3985B637h, 0ECh

|  |  |  |
| --- | --- | --- |
| Variable | Address | Data |
| Rose | 0x0065A300 | B9h |
| Magnolia | 0x0065A301 | 4Eh |
| Magnolia + 1 | 0x0065A302 | E5h |
| Cannas | 0x0065A303 | 37h |
| Cannas + 1 | 0x0065A304 | B6h |
| Cannas + 2 | 0x0065A305 | 85h |
| Cannas + 3 | 0x0065A306 | 39h |
| Cannas + 4 | 0x0065A307 | ECh |
| Cannas + 5 | 0x0065A308 | 00h |
| Cannas + 6 | 0x0065A309 | 00h |
| Cannas + 7 | 0x0065A30A | 00h |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

2. [Addressing Modes] Copy the following code into your assembly development environment and single-step through it.  For each single step execution, submit the screenshot.  For those instructions referencing memory, do the linear address computation by hand and typewrite it.

\*Note -

TITLE Addressing Modes              (main.asm)  
  
INCLUDE Irvine32.inc  
.data  
   alpha       DWORD       0A1B1C1D1h, 87654321h    
   beta        DWORD      67EED9FCh, 21A220C2h  
  gamma       DWORD       0BCB1D44Fh  
.code         
  
main PROC     
      mov eax, 1C2Fh;         Immediate ; does not refer to memory  
      mov ecx, eax;           Register to Register ; does not refer to memory     
      mov edi, OFFSET beta;   Immediate   ; does not refer to memory  
      mov [gamma], eax;       Direct    ; address: 0x00404010  
      mov esi, gamma;         Direct    ; address: 0x00404010    
      mov esi, 4;             Immediate  ; does not refer to memory  
      mov eax, beta[esi];     Indirect-offset ; beta + esi = 0x00404008 + 0x00000004

= 0x0040400C

     mov ebx, OFFSET alpha;  Immediate   ; does not refer to memory   
      mov eax, 4[ebx];        Indirect-displacement; 4 + ebx = 0x00000004 + 0x00404000

= 0x00404004  
 mov eax, [ebx];         Indirect   ; refers to alpha; 0x00404000    
      mov eax,4[ebx][esi]; Base-Indirect-displacement;

4 + ebx + esi = 0x00000004 + 0x00404000 + 0x00000004 = 0x00404008  
exit  
main ENDP  
END main

Screenshots for problem 2:

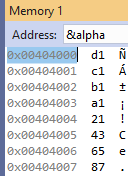


Figure : address of Alpha

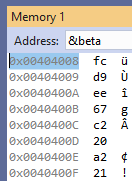


Figure Address of Beta

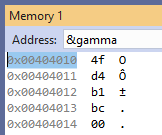
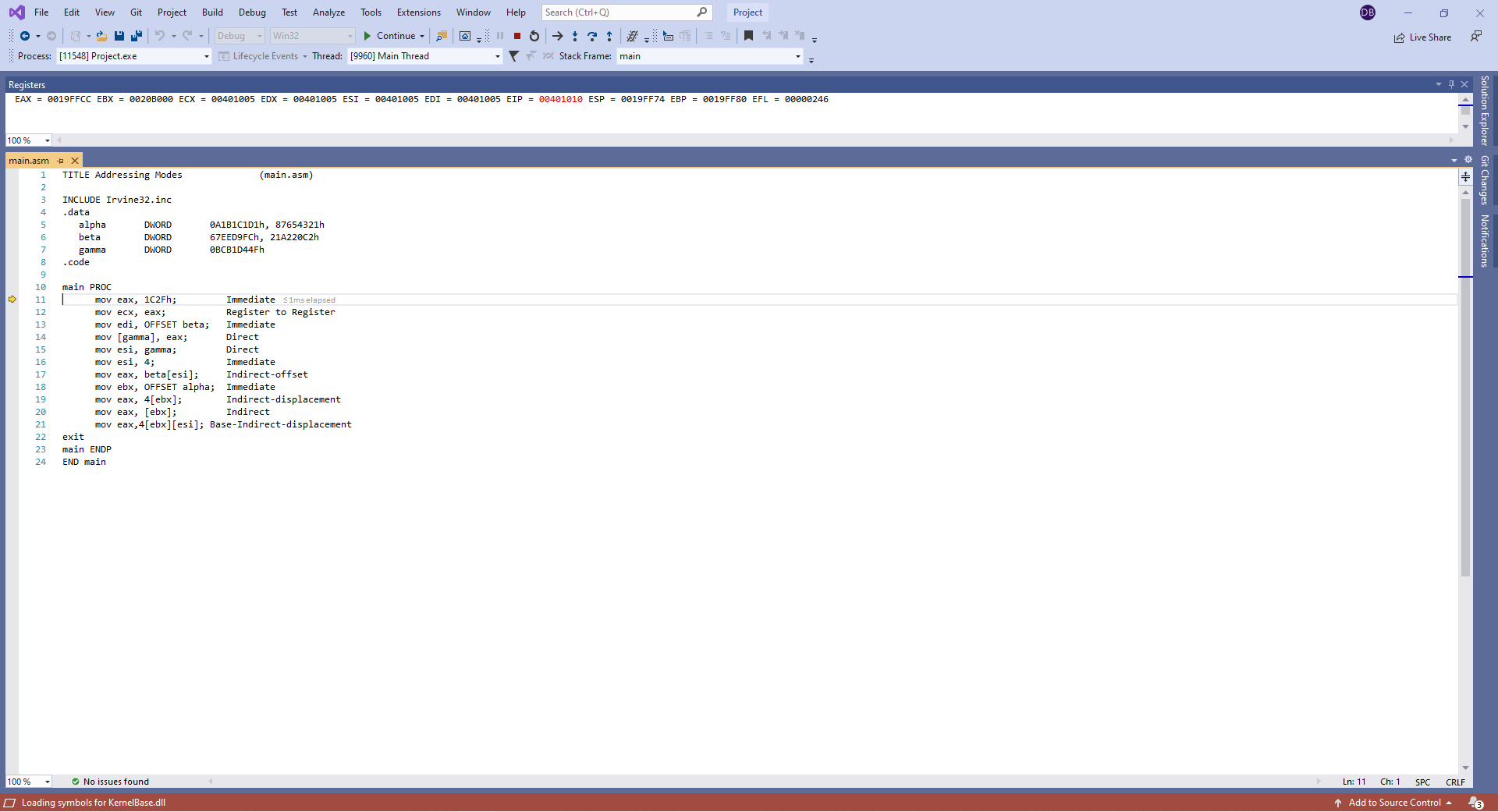
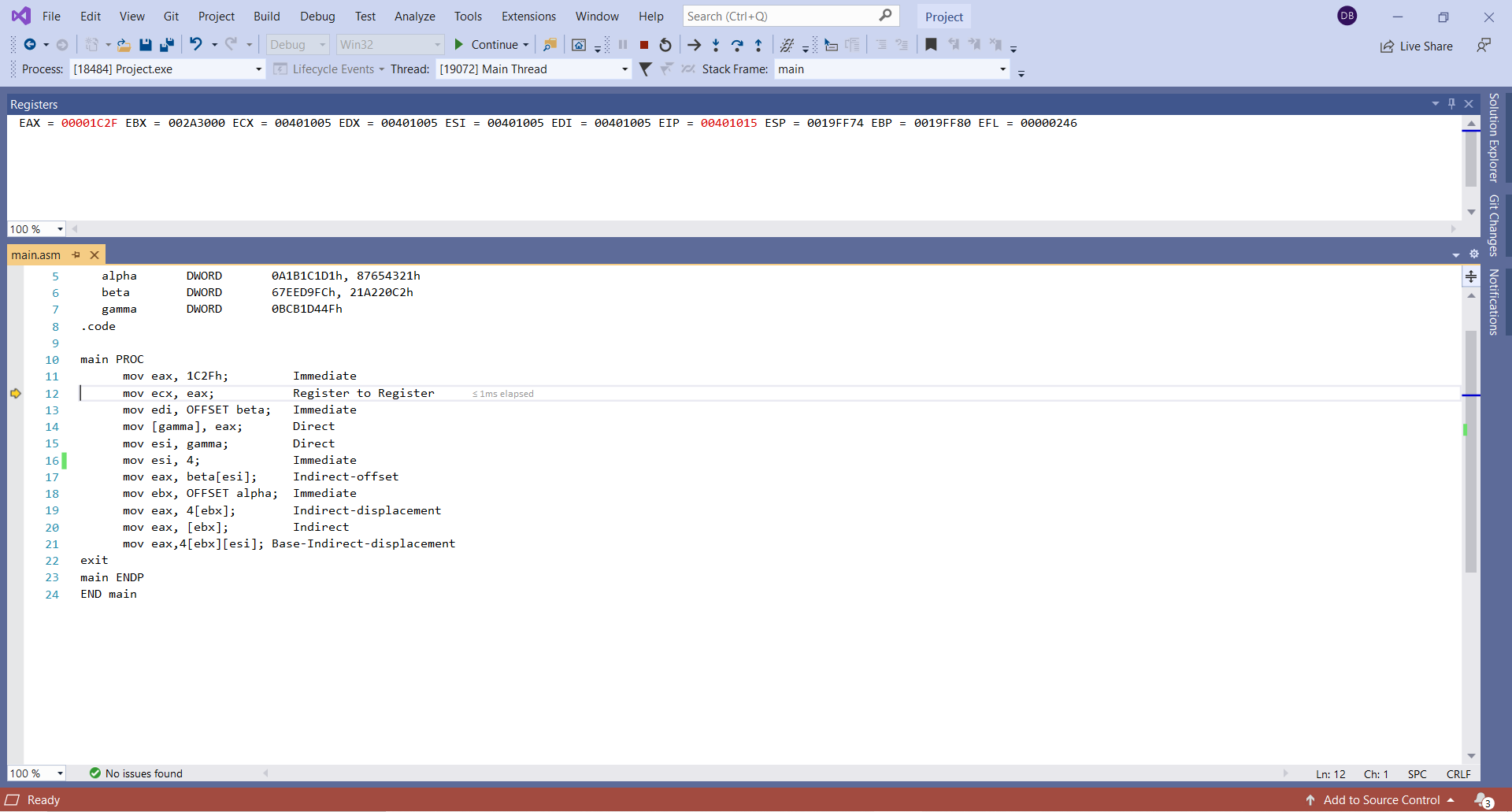
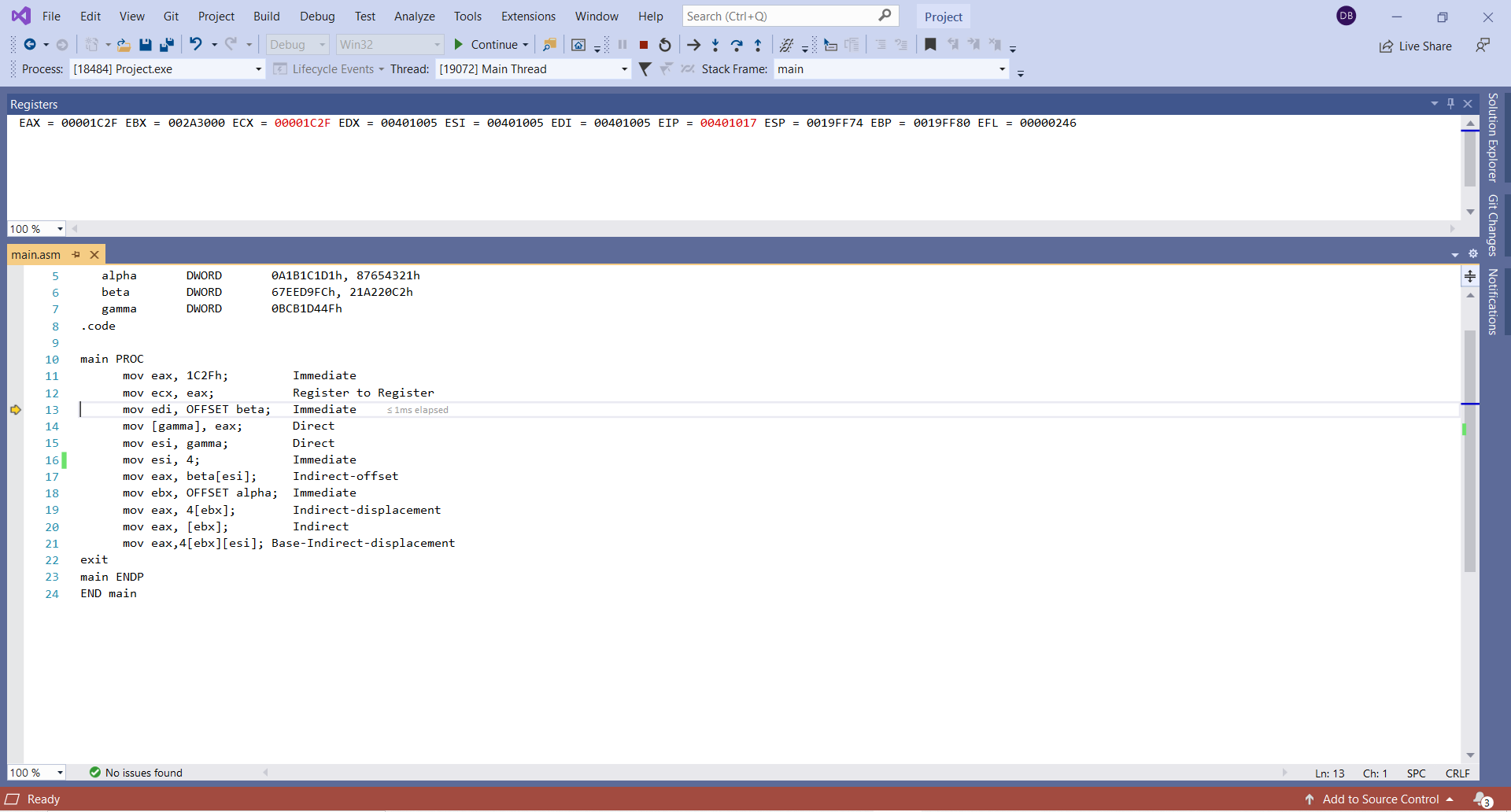


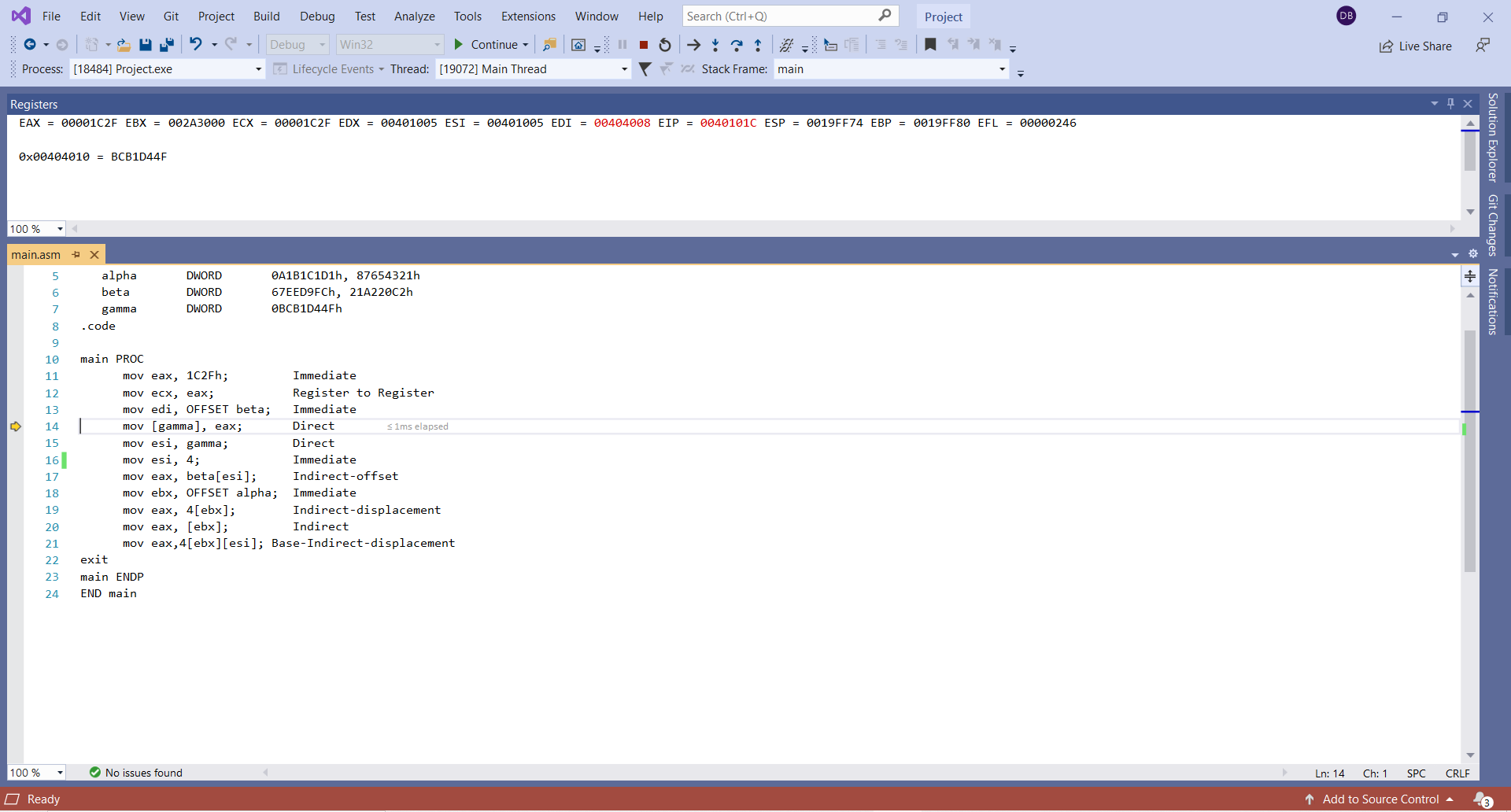
Figure Address of Gamma

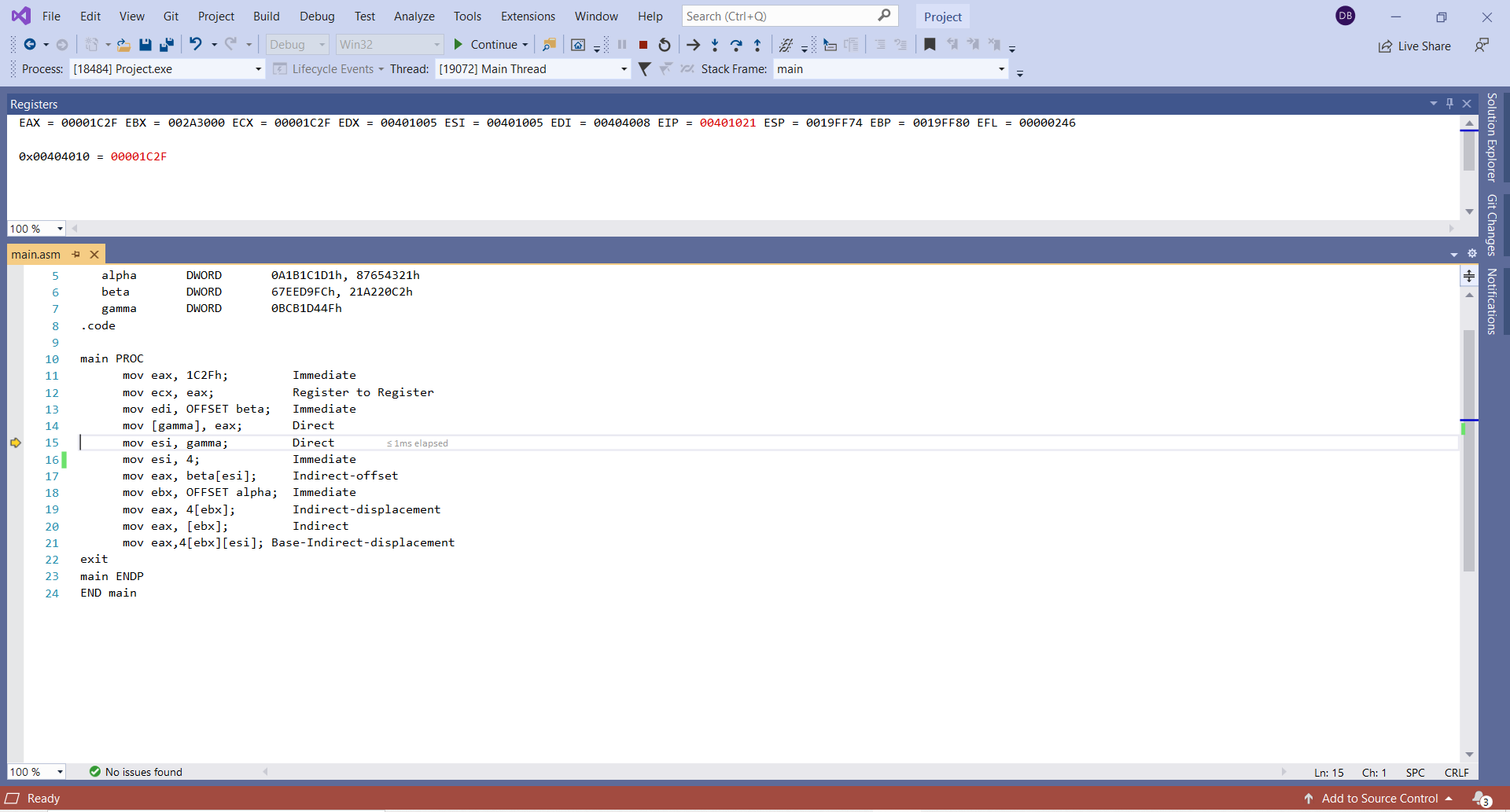
Single-step execution screenshots:

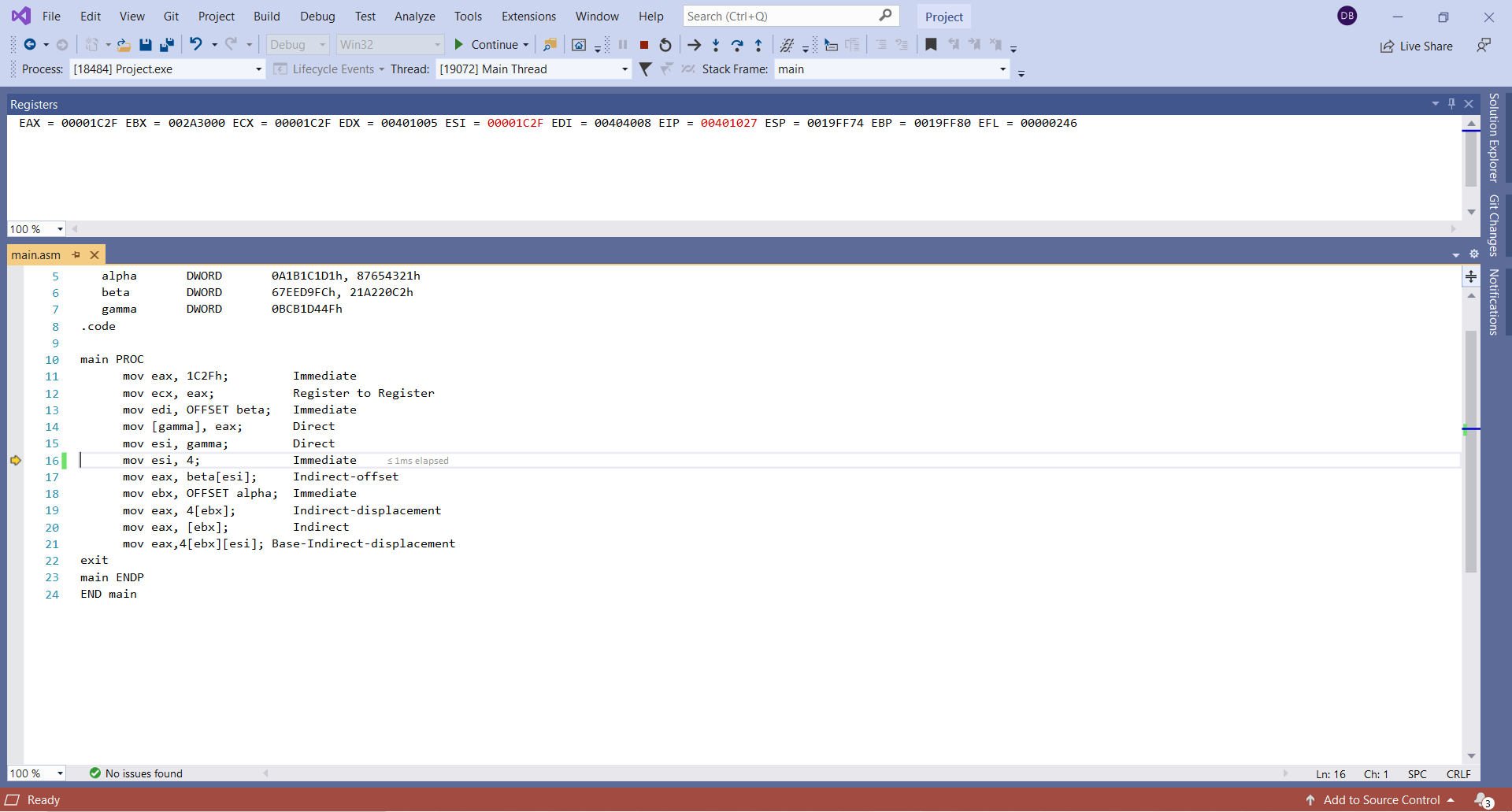


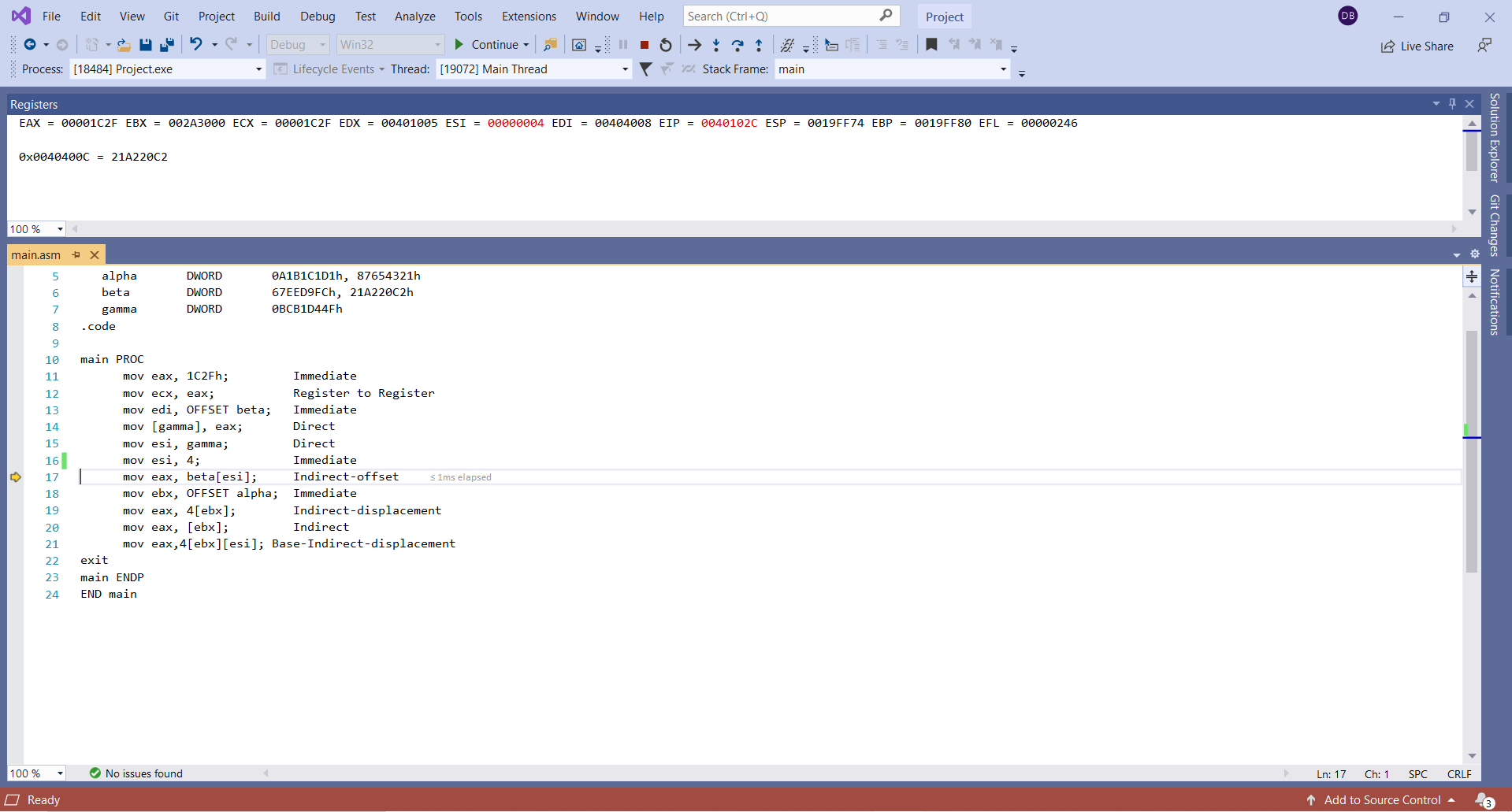


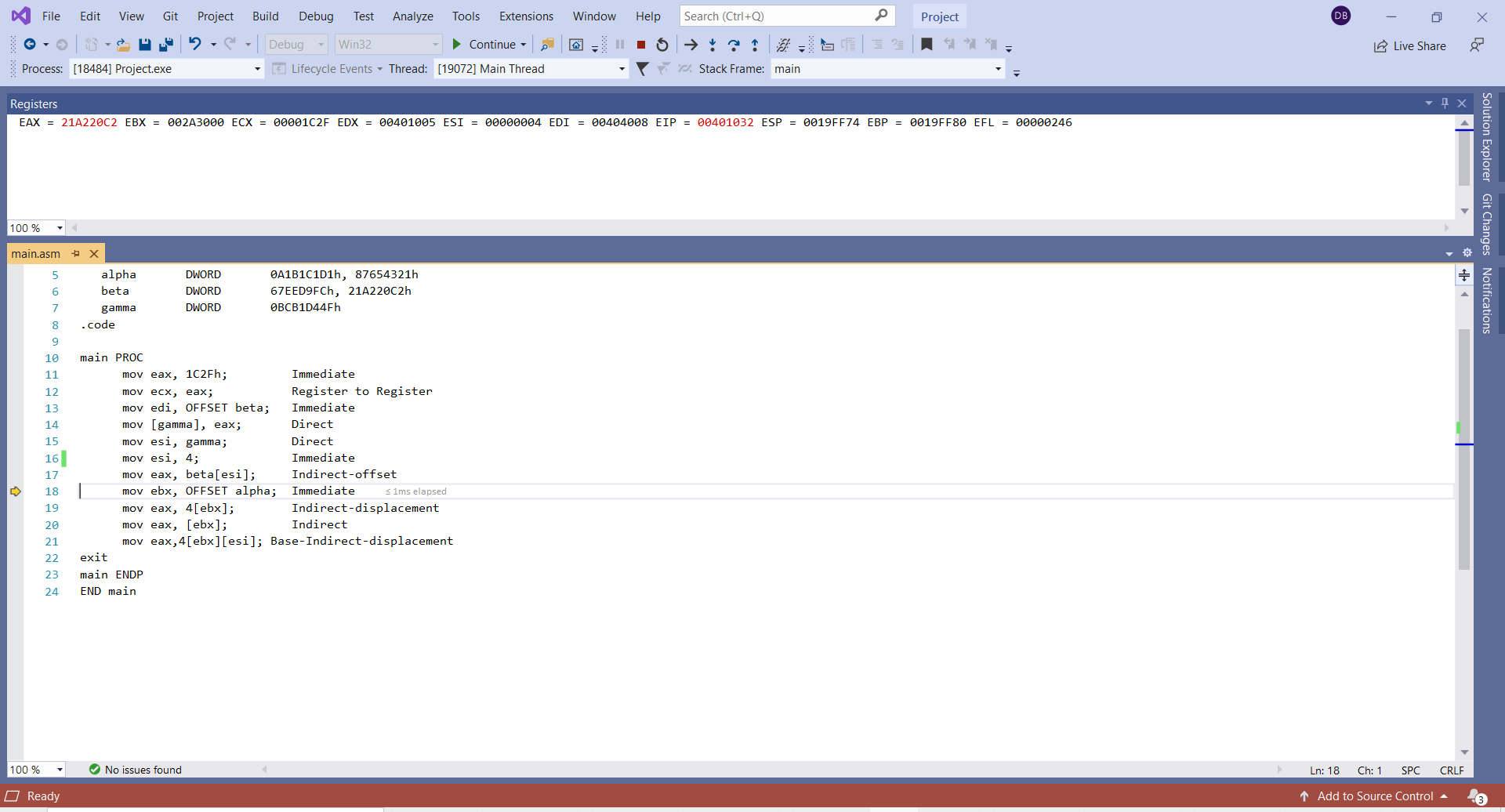


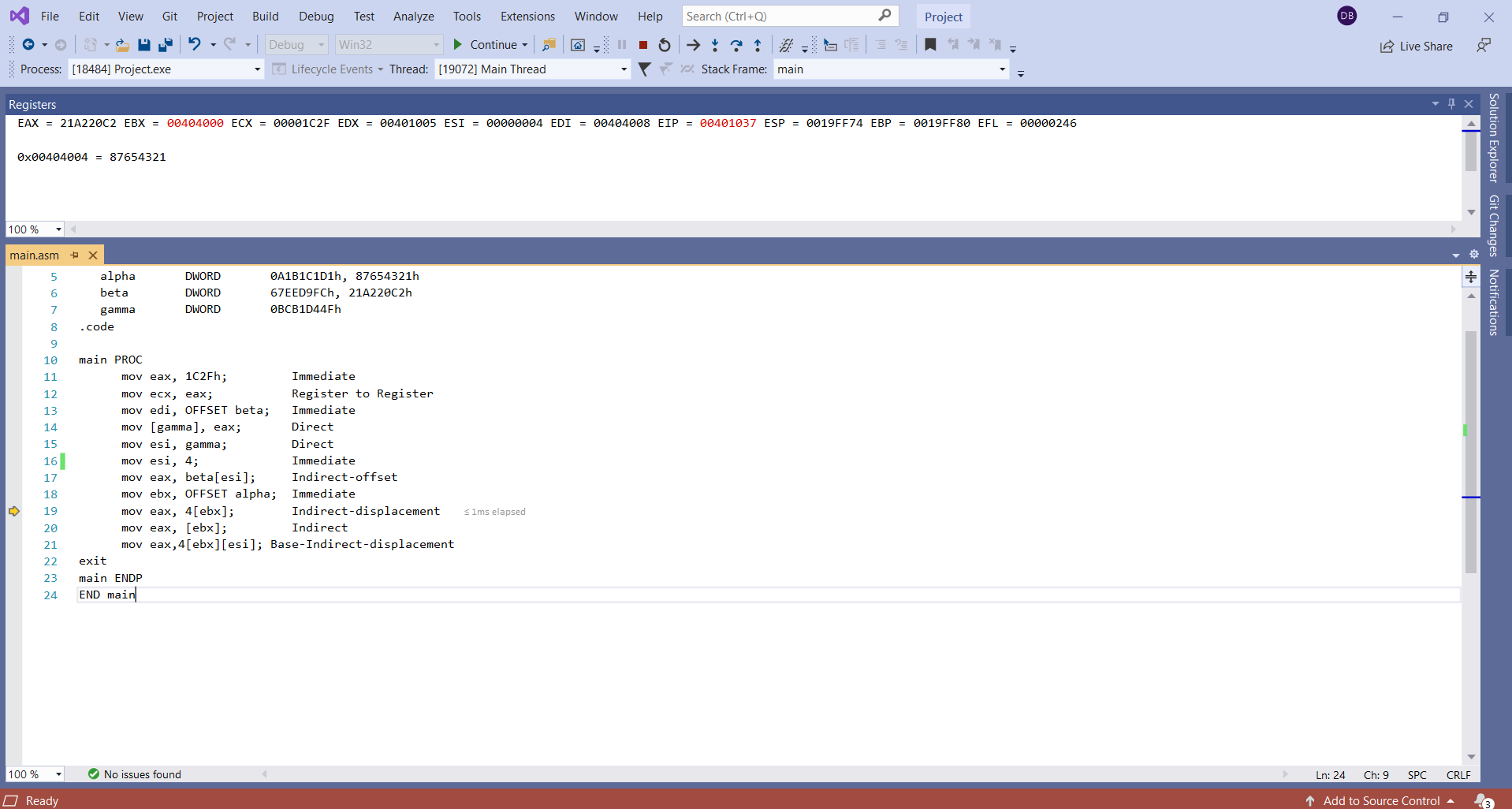


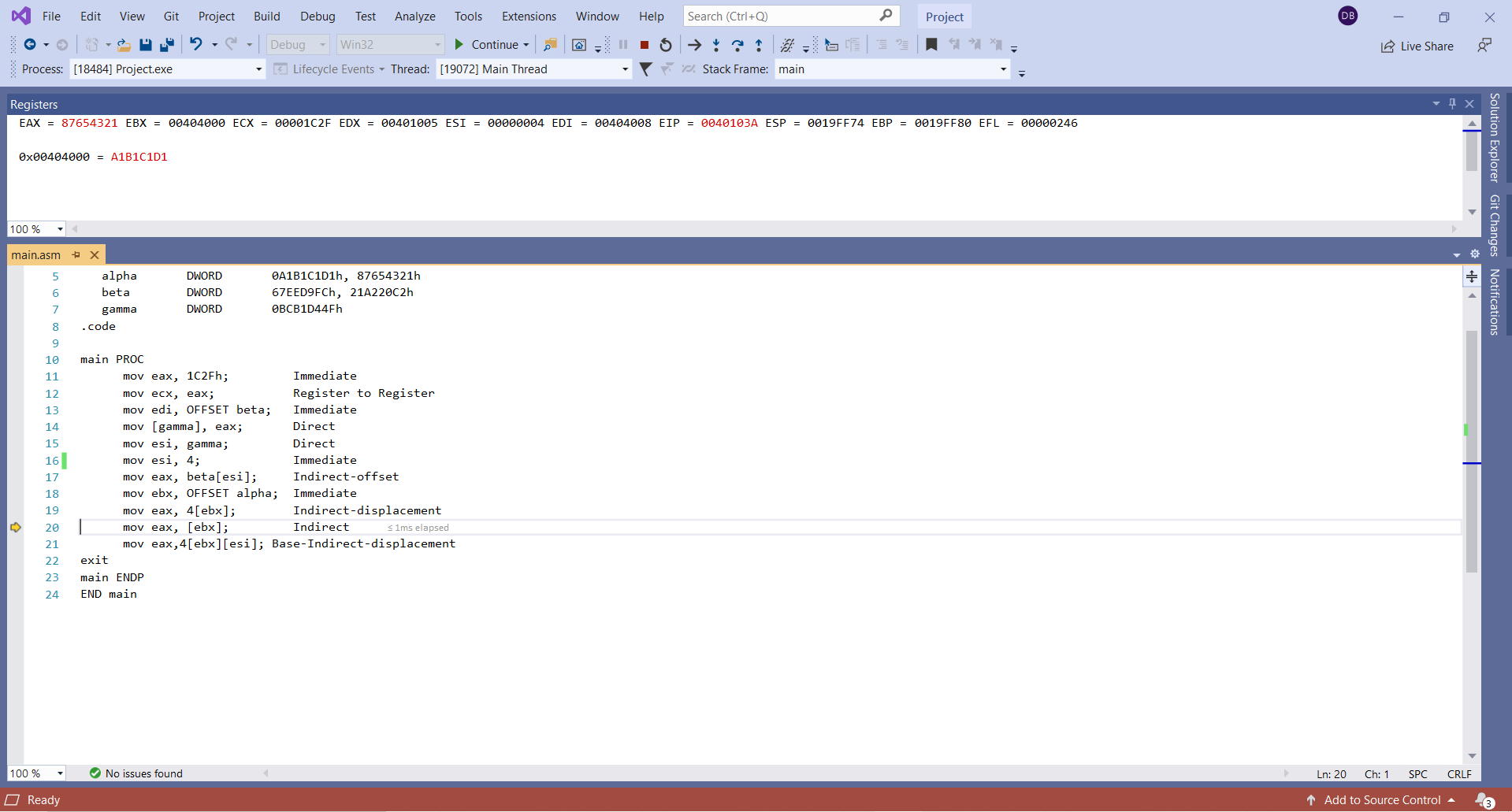


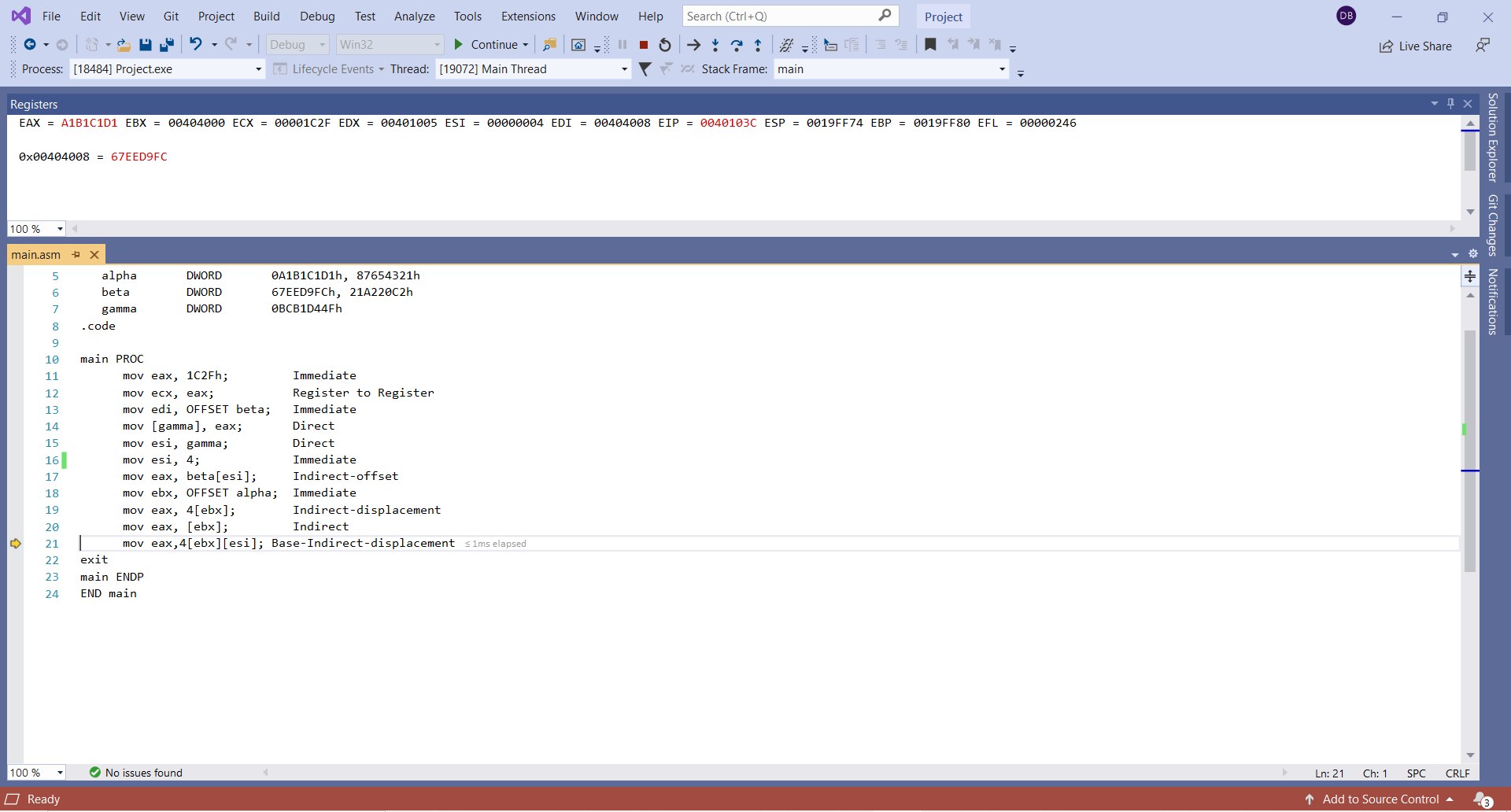


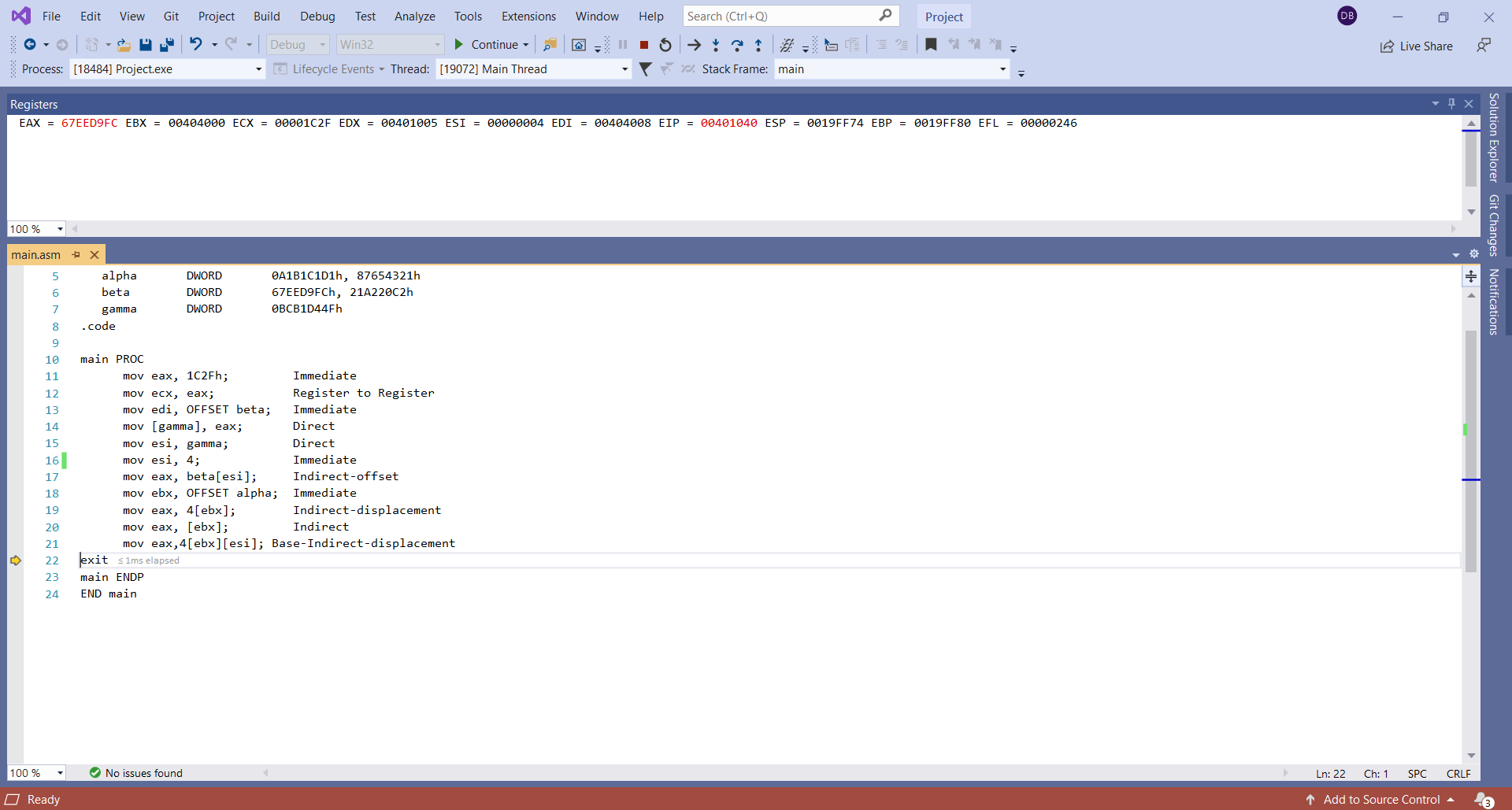


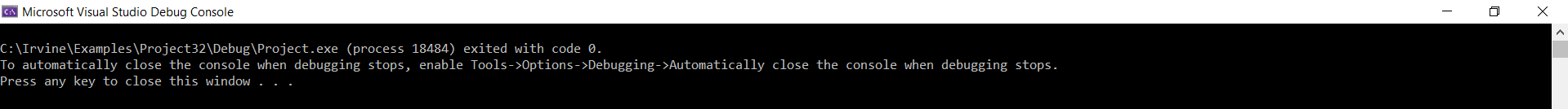








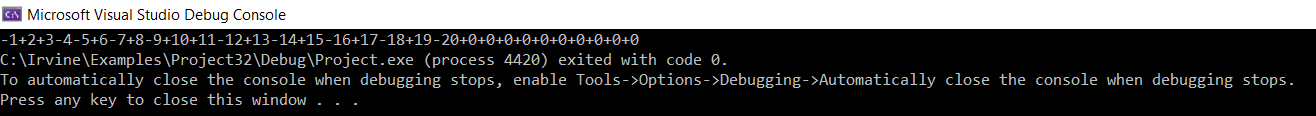




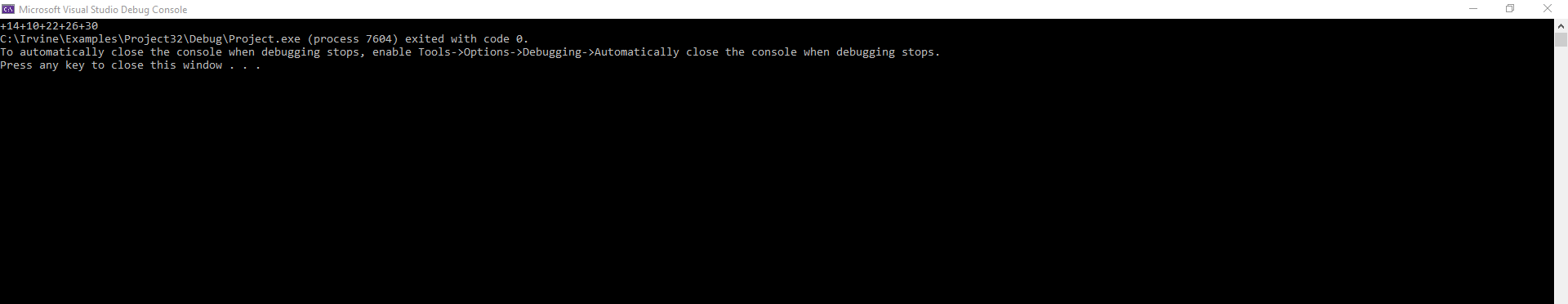
3. [Indirect addressing] Write a program that first displays  all the elements of Array1, Array2 and Array3.  Then, the program should subtract all the odd indexed elements of Array2 from the odd indexed elements of Array1 and store the result in Array3; e.g. Array3 [7] = Array1 [7] - Array2 [7].  Next, it must add the even indexed elements of Array1 and Array 2 and store them in the corresponding even indexed elements of Array3, e.g. Array3 [4] = Array1 [4] + Array2 [4]. Next, display the elements of all the arrays after these operations.  Submit screenshots of the displays of the elements of all the arrays.  You can use WriteInt or WriteHex to display the elements of the arrays.  Fill in Array1 and Array2 each by your own ten numbers each using both positive and negative integers.

.data  
Array1      SWORD -1, 2, 3, -4, -5, 6, -7, 8, -9, 10  
Array2      SWORD 11, -12, 13, -14, 15, -16, 17, -18, 19, -20  
Array3      SWORD 10 DUP (?)

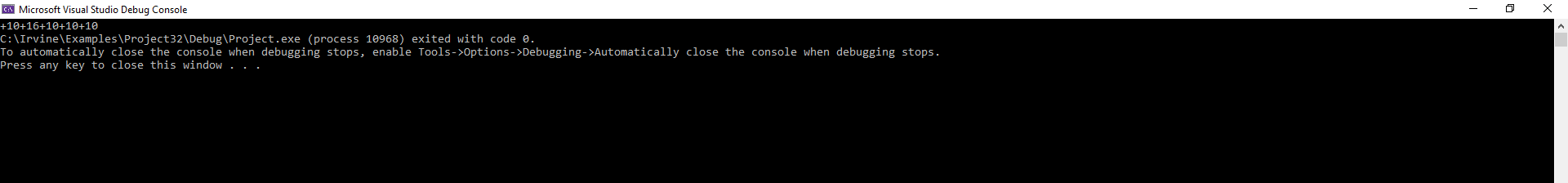
First Displaying All The Elements of Array1, Array2, and Array3:



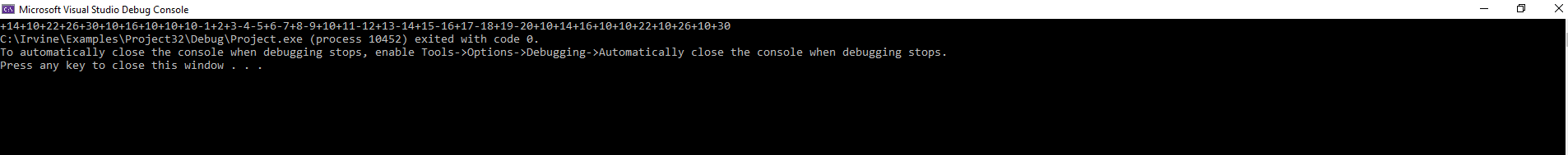
;Subtracting all the odd indexed elements of Array2 from the odd indexed elements of Array1 and store the result in Array3



;Add the even indexed elements of Array1 and Array 2 and store them in the corresponding even indexed elements of Array3



;displaying the elements of all the arrays after these operations



4. [Loops] Declare a signed word array. Write a program to print on screen the firs*t n* odd indexed elements of the array, using the Loop instruction.

* 1. Prompt user for integer *n*,
  2. Read the value of *n* from user input

Please use the “WriteInt” procedure, not “DumpRegs”. Other relevant procedures: “ReadInt” and “WriteString.” In your homework submission, please embed both the code and one screen shot for *n = 6*.

; PrintSWORDArray.asm

INCLUDE Irvine32.inc

.386

.stack 4096

ExitProcess proto,dwExitCode:dword

.data

;Declare a signed word array

Array1 SWORD 1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12

;Prompting user for integer n

promptRead BYTE "Enter an integer n: ",0

;integer n

n DWORD ?

.code

main proc

mov edx, OFFSET promptRead

call WriteString

call Crlf

call ReadInt

mov n, eax ; stores user input as n

mov esi, 2

mov ecx, n ; sets loop counter to n

; Prints on screen the first n odd indexed elements of the array, using the Loop instruction.

read:

mov ax, Array1[esi]

movsx eax, ax

Call WriteInt

add esi, 4

loop read

invoke ExitProcess,0

main endp

end main

