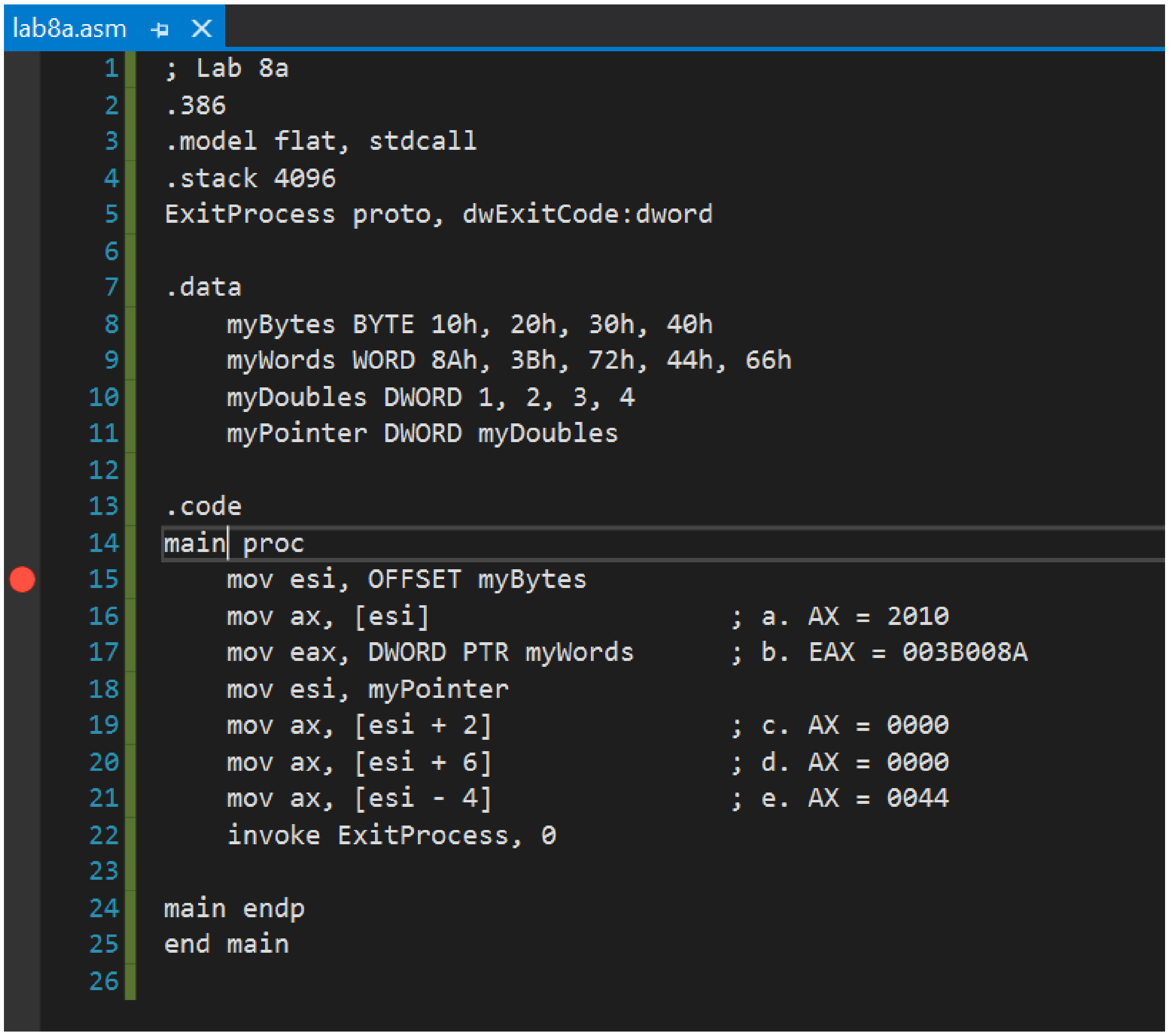
CSC 3210
Computer Organization and Programming
Lab 8
Answer Sheet

Student Name: Vivian Do
Section: CRN 90913; 11:00-12:40
Oct 15 2021

Lab 8(a)

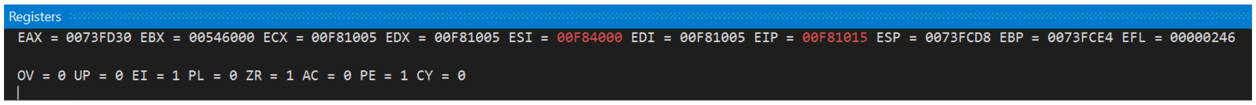
Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents.

Code:



```
; Lab 8a
.386
.model flat, stdcall
.stack 4096
ExitProcess proto, dwExitCode:dword

.data
    myBytes BYTE 10h, 20h, 30h, 40h
    myWords WORD 8Ah, 3Bh, 72h, 44h, 66h
    myDoubles DWORD 1, 2, 3, 4
    myPointer DWORD myDoubles

.code
main proc
    mov esi, OFFSET myBytes
    mov ax, [esi]                   ; a. AX = 2010
    mov eax, DWORD PTR myWords      ; b. EAX = 003B008A
    mov esi, myPointer
    mov ax, [esi + 2]               ; c. AX = 0000
    mov ax, [esi + 6]               ; d. AX = 0000
    mov ax, [esi - 4]               ; e. AX = 0044
    invoke ExitProcess, 0

main endp
end main
```

Line number: 15
Instruction: mov esi, OFFSET myBytes
Register Values: ESI = 00DD4000
Screenshot:

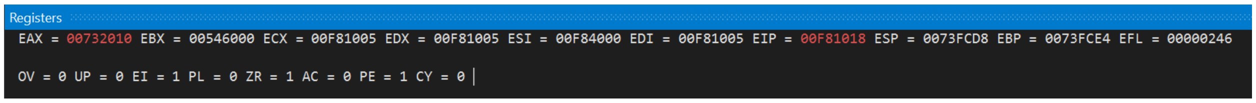

Registers

```
EAX = 0073FD30 EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F84000 EDI = 00F81005 EIP = 00F81015 ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: N/A

(Copy this format as needed)

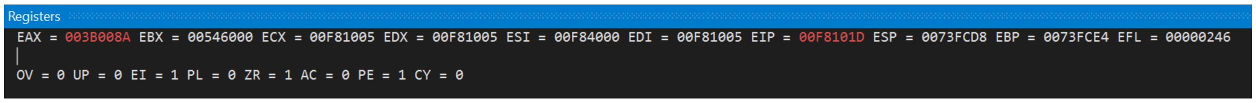
Line number: 16
Instruction: mov ax, [esi]
Register Values: EAX = 00CF2010
Screenshot:


Registers

```
EAX = 00732010 EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F84000 EDI = 00F81005 EIP = 00F81018 ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is 32-bit long with a signed integer variable. The register is updated with 2010 when the first element in ESI is moved to AX.

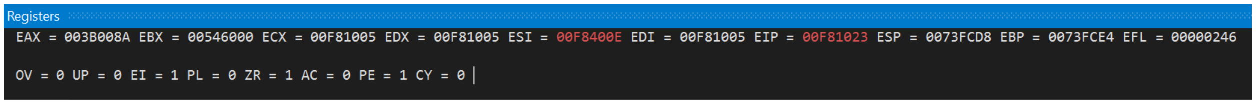
Line number: 17
Instruction: mov eax, DWORD PTR myWords
Register Values: EAX = 003B008A
Screenshot:


Registers

```
EAX = 003B008A EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F84000 EDI = 00F81005 EIP = 00F8101D ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated with 003B008A.

Line number: 18
Instruction: mov esi, myPointer
Register Values: ESI = 00F8400E
Screenshot:

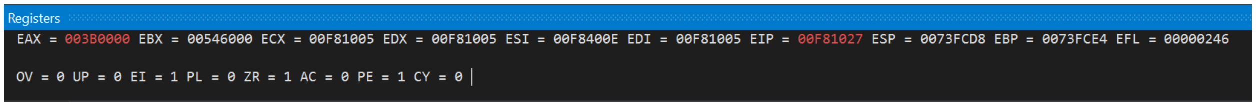

Registers

```
EAX = 003B008A EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F8400E EDI = 00F81005 EIP = 00F81023 ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: N/A

Line number: 19
Instruction: mov ax, [esi + 2]
Register Values: EAX = 003B0000

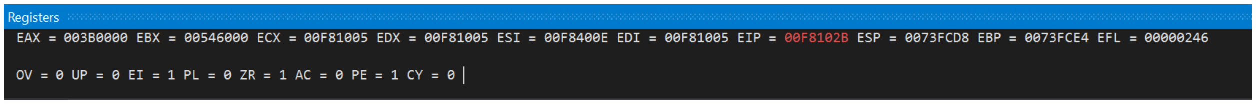
Screenshot:



```
Registers
EAX = 003B0000 EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F8400E EDI = 00F81005 EIP = 00F81027 ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated with 0000 in ax when ESI + 2 is moved to AX.

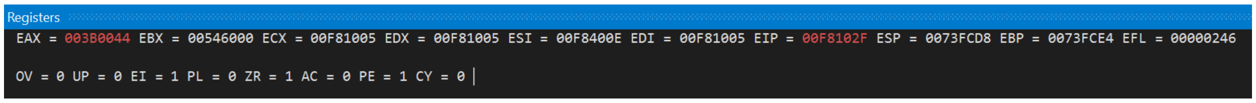
Line number: 20
Instruction: mov ax, [esi + 6]
Register Values: EAX = 003B0000
Screenshot:



```
Registers
EAX = 003B0000 EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F8400E EDI = 00F81005 EIP = 00F8102B ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: Nothing has changed in the EAX register when ESI + 6 is moved to AX.

Line number: 21
Instruction: mov ax, [esi - 4]
Register Values: EAX = 003B0044
Screenshot:



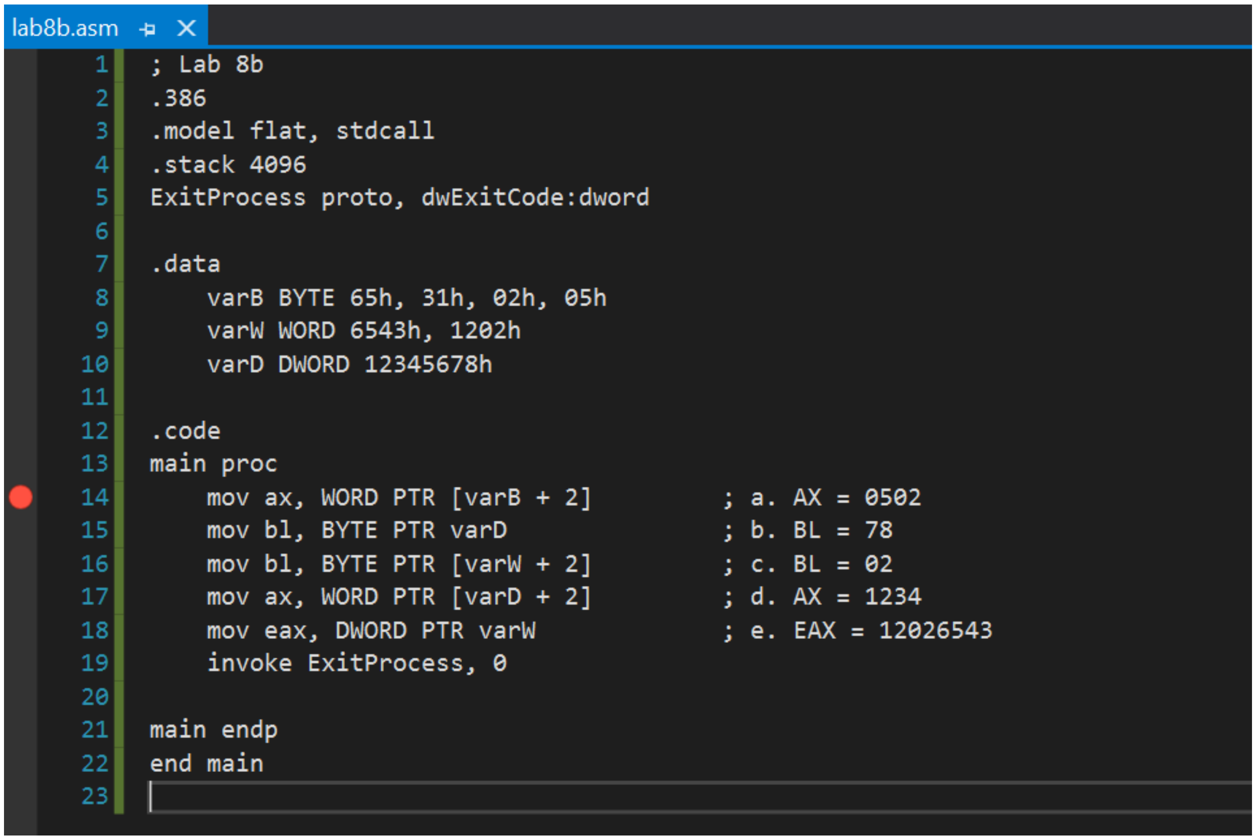
```
Registers
EAX = 003B0044 EBX = 00546000 ECX = 00F81005 EDX = 00F81005 ESI = 00F8400E EDI = 00F81005 EIP = 00F8102F ESP = 0073FCD8 EBP = 0073FCE4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated with 0044 when ESI - 4 is moved to AX.

Lab 8(b)

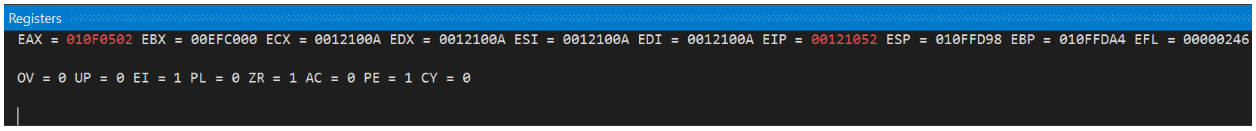
Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents.

Code:


lab8b.asm

```
 1  ; Lab 8b
 2  .386
 3  .model flat, stdcall
 4  .stack 4096
 5  ExitProcess proto, dwExitCode:dword
 6
 7  .data
 8      varB BYTE 65h, 31h, 02h, 05h
 9      varW WORD 6543h, 1202h
10      varD DWORD 12345678h
11
12  .code
13  main proc
14      mov ax, WORD PTR [varB + 2]         ; a. AX = 0502
15      mov bl, BYTE PTR varD               ; b. BL = 78
16      mov bl, BYTE PTR [varW + 2]         ; c. BL = 02
17      mov ax, WORD PTR [varD + 2]         ; d. AX = 1234
18      mov eax, DWORD PTR varW             ; e. EAX = 12026543
19      invoke ExitProcess, 0
20
21  main endp
22  end main
23
```

Line number: 14
Instruction: mov ax, WORD PTR [varB + 2]
Register Values: EAX = 010F0502
Screenshot:

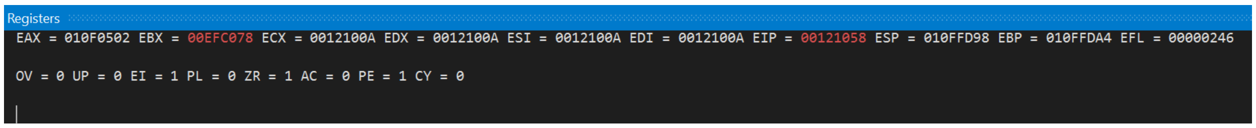

Registers

```
EAX = 010F0502 EBX = 00EFC000 ECX = 0012100A EDX = 0012100A ESI = 0012100A EDI = 0012100A EIP = 00121052 ESP = 010FFD98 EBP = 010FFDA4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: AX in EAX register is updated with 0502.

(Copy this format as needed)

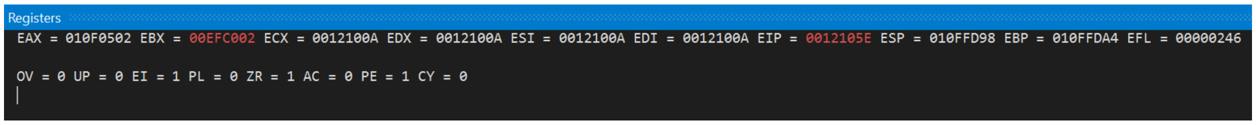
Line number: 15
Instruction: mov bl, BYTE PTR varD
Register Values: EBX = 00EFC078
Screenshot:


Registers

```
EAX = 010F0502 EBX = 00EFC078 ECX = 0012100A EDX = 0012100A ESI = 0012100A EDI = 0012100A EIP = 00121058 ESP = 010FFD98 EBP = 010FFDA4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: BL in EBX register is updated with the last 2 numbers of varD (12345678h).

Line number: 16
Instruction: mov bl, BYTE PTR [varW + 2]
Register Values: EBX = 00EFC002
Screenshot:

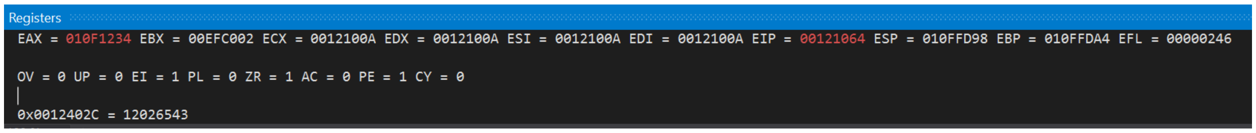


Registers

```
EAX = 010F0502 EBX = 00EFC002 ECX = 0012100A EDX = 0012100A ESI = 0012100A EDI = 0012100A EIP = 0012105E ESP = 010FFD98 EBP = 010FFDA4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: BL in EBX register is updated with the last two numbers of the second element in varW (1202h)

Line number: 17
Instruction: mov ax, WORD PTR [varD + 2]
Register Values: EAX = 010F1234
Screenshot:

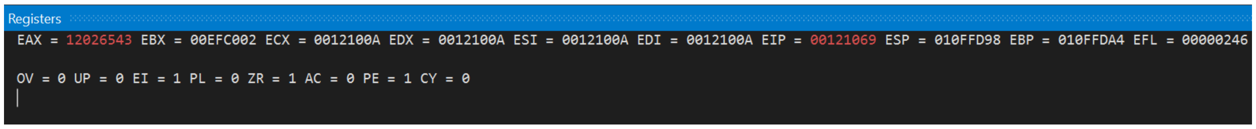


Registers

```
EAX = 010F1234 EBX = 00EFC002 ECX = 0012100A EDX = 0012100A ESI = 0012100A EDI = 0012100A EIP = 00121064 ESP = 010FFD98 EBP = 010FFDA4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0

0x0012402C = 12026543
```

Explanation: AX in EAX register is updated with the first 4 numbers of the second element in varD (12345678).

Line number: 18
Instruction: mov eax, DWORD PTR varW
Register Values: EAX = 12026543
Screenshot:



Registers

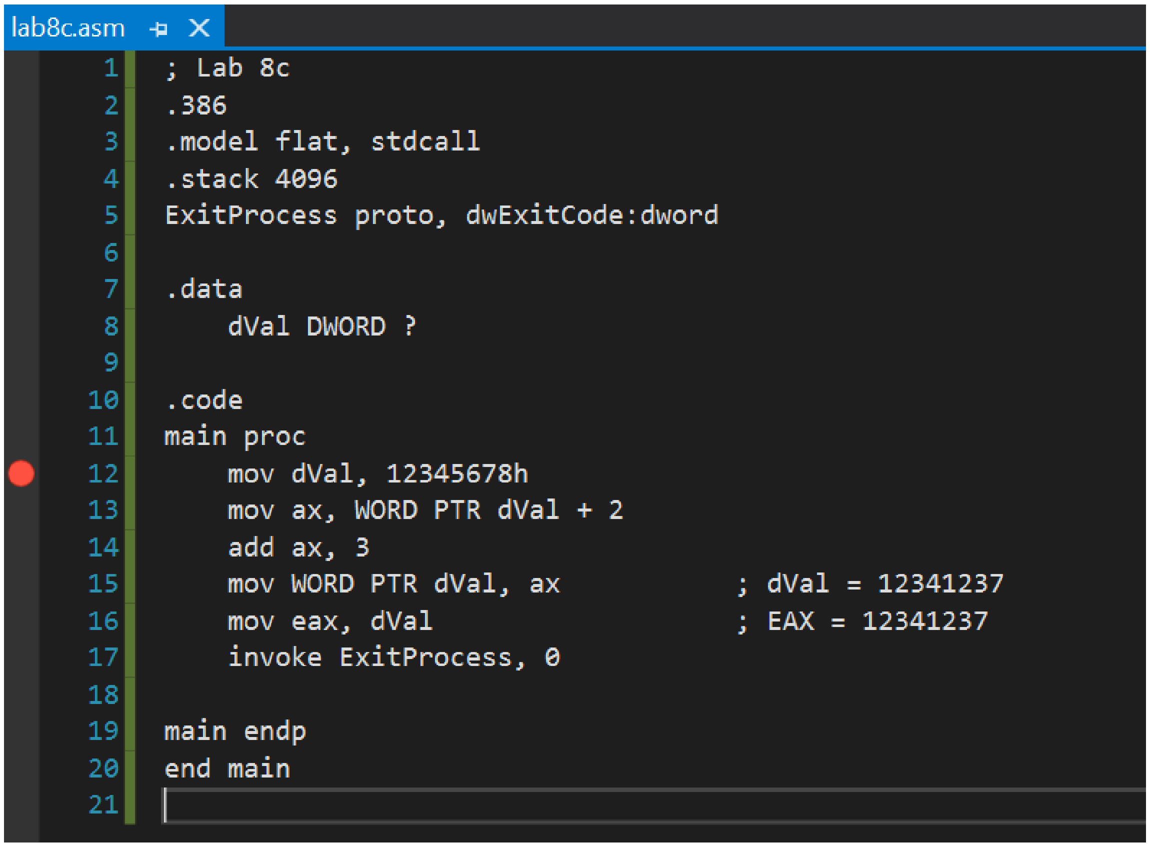
```
EAX = 12026543 EBX = 00EFC002 ECX = 0012100A EDX = 0012100A ESI = 0012100A EDI = 0012100A EIP = 00121069 ESP = 010FFD98 EBP = 010FFDA4 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated with varW, but in descending order. 1202 is the first half, and 6543 is the second half of the register.

Lab 8(c)

Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents.

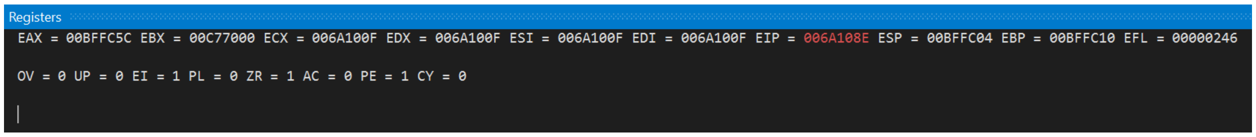
Code:



lab8c.asm

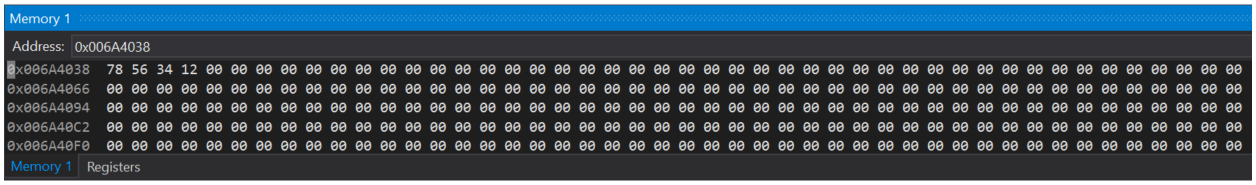
```
 1  ; Lab 8c
 2  .386
 3  .model flat, stdcall
 4  .stack 4096
 5  ExitProcess proto, dwExitCode:dword
 6
 7  .data
 8      dVal DWORD ?
 9
10  .code
11  main proc
12      mov dVal, 12345678h
13      mov ax, WORD PTR dVal + 2
14      add ax, 3
15      mov WORD PTR dVal, ax             ; dVal = 12341237
16      mov eax, dVal                     ; EAX = 12341237
17      invoke ExitProcess, 0
18
19  main endp
20  end main
21
```

Line number: 12
Instruction: mov dVal, 12345678h
Register / variable content: EIP = 006A108E
Screenshot:



Registers

```
EAX = 00BFFC5C EBX = 00C77000 ECX = 006A100F EDX = 006A100F ESI = 006A100F EDI = 006A100F EIP = 006A108E ESP = 00BFFC04 EBP = 00BFFC10 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```



Memory 1

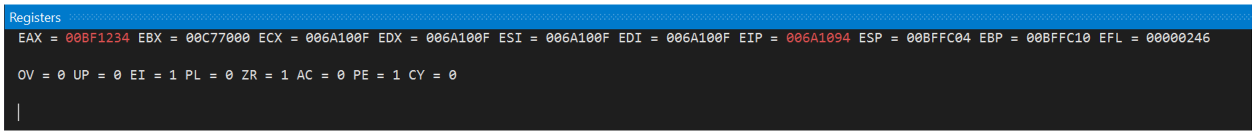
Address: 0x006A4038

```
0x006A4038  78 56 34 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A4066  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A4094  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A40C2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A40F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Memory 1 Registers

Explanation: Memory is updated from 0’s to 12345678h in Little Endian order when dVal was moved.

Line number: 13
Instruction: mov ax, WORD PTR dVal + 2
Register / variable content: EAX = 00BF1234
Screenshot:

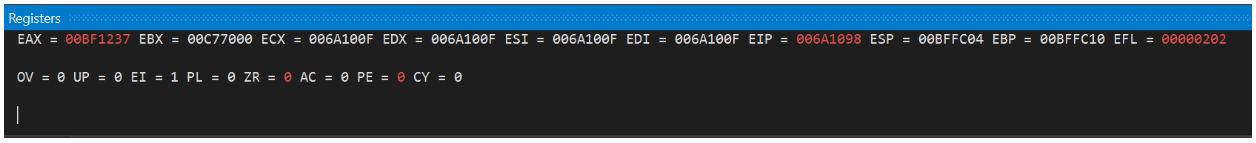


```
Registers
EAX = 00BF1234 EBX = 00C77000 ECX = 006A100F EDX = 006A100F ESI = 006A100F EDI = 006A100F EIP = 006A1094 ESP = 00BFFC04 EBP = 00BFFC10 EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: AX in EAX register is updated with the values from dVal (12345678).

(Copy this format as needed)

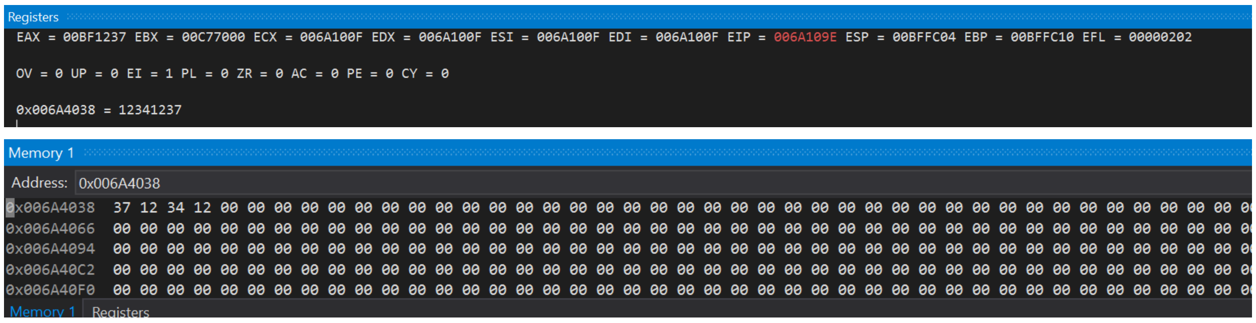
Line number: 14
Instruction: add ax, 3
Register / variable content: EAX = 00BF1237
Screenshot:



```
Registers
EAX = 00BF1237 EBX = 00C77000 ECX = 006A100F EDX = 006A100F ESI = 006A100F EDI = 006A100F EIP = 006A1098 ESP = 00BFFC04 EBP = 00BFFC10 EFL = 00000202

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0
```

Explanation: 3 is added to AX in EAX register.

Line number: 15
Instruction: mov WORD PTR dVal, ax
Register / variable content: 12341237
Screenshot:

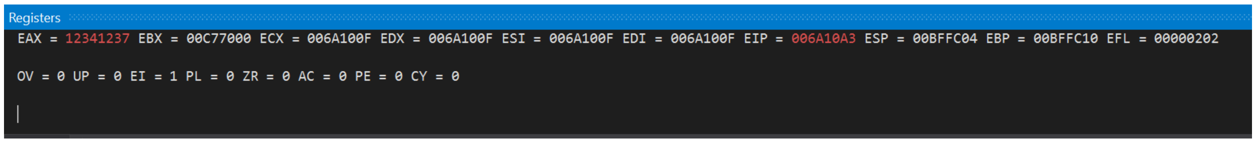


```
Registers
EAX = 00BF1237 EBX = 00C77000 ECX = 006A100F EDX = 006A100F ESI = 006A100F EDI = 006A100F EIP = 006A109E ESP = 00BFFC04 EBP = 00BFFC10 EFL = 00000202

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0

0x006A4038 = 12341237
```

```
Memory 1
Address: 0x006A4038
0x006A4038  37 12 34 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A4066  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A4094  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A40C2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x006A40F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Memory 1   Registers
```

Explanation: The first two elements of Memory/dVal are updated with AX in Little Endian. The overall value of Memory is no 12341237.

Line number: 16
Instruction: mov eax, dVal
Register / variable content: EAX = 12341237
Screenshot:



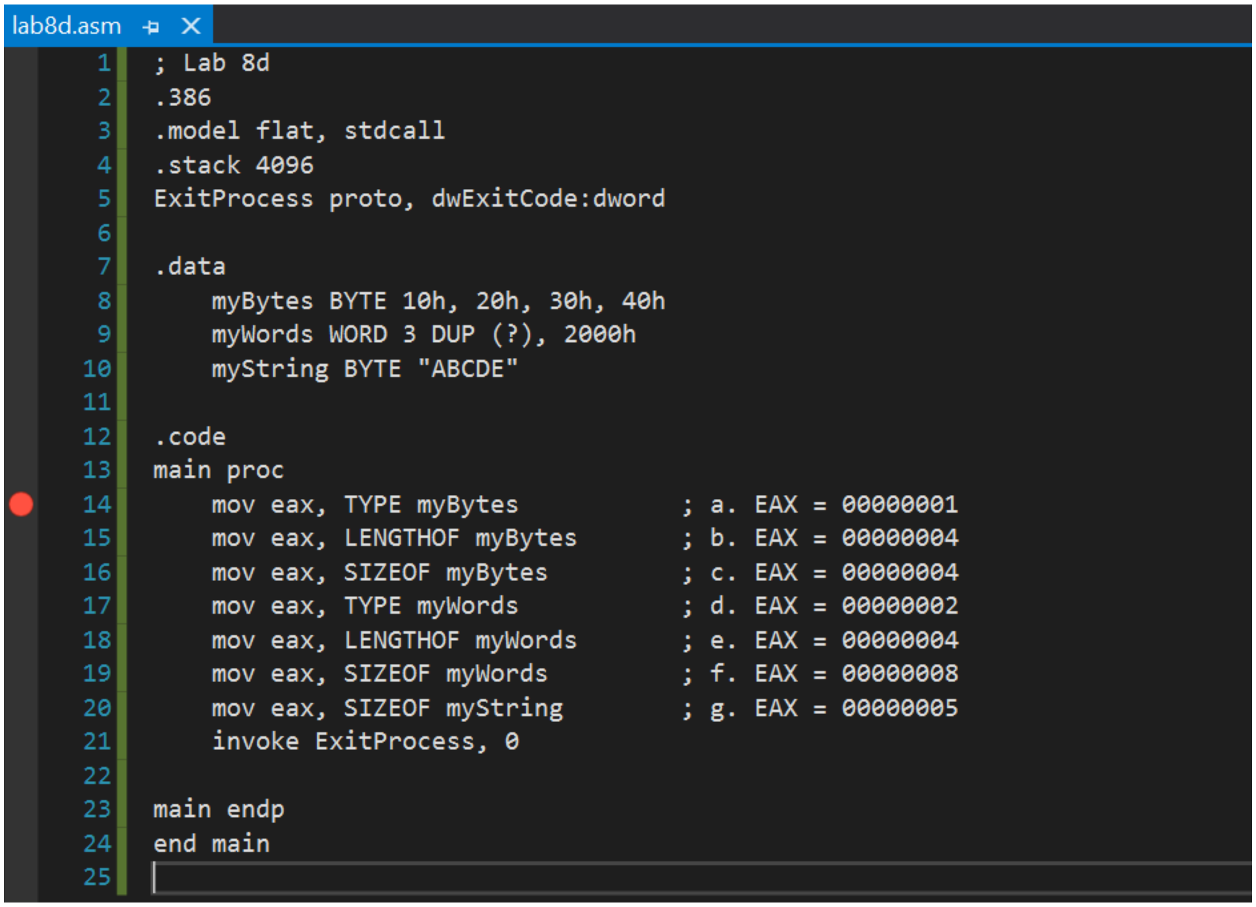
```
Registers
EAX = 12341237 EBX = 00C77000 ECX = 006A100F EDX = 006A100F ESI = 006A100F EDI = 006A100F EIP = 006A10A3 ESP = 00BFFC04 EBP = 00BFFC10 EFL = 00000202

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 0 AC = 0 PE = 0 CY = 0
```

Explanation: EAX register is updated with the value in Memory/dVal.

Lab 8(d)

Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents.

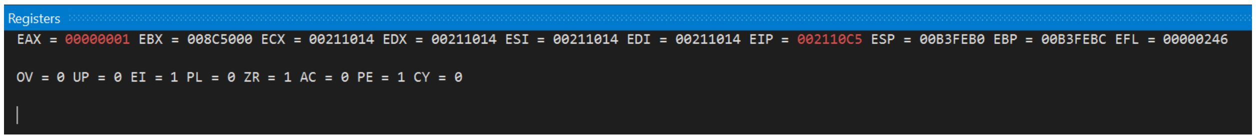
Code:



lab8d.asm

```
 1  ; Lab 8d
 2  .386
 3  .model flat, stdcall
 4  .stack 4096
 5  ExitProcess proto, dwExitCode:dword
 6
 7  .data
 8      myBytes BYTE 10h, 20h, 30h, 40h
 9      myWords WORD 3 DUP (?), 2000h
10      myString BYTE "ABCDE"
11
12  .code
13  main proc
14      mov eax, TYPE myBytes           ; a. EAX = 00000001
15      mov eax, LENGTHOF myBytes       ; b. EAX = 00000004
16      mov eax, SIZEOF myBytes         ; c. EAX = 00000004
17      mov eax, TYPE myWords           ; d. EAX = 00000002
18      mov eax, LENGTHOF myWords       ; e. EAX = 00000004
19      mov eax, SIZEOF myWords         ; f. EAX = 00000008
20      mov eax, SIZEOF myString        ; g. EAX = 00000005
21      invoke ExitProcess, 0
22
23  main endp
24  end main
25
```

Line number: 14
Instruction: mov eax, TYPE myBytes
Register Values: EAX = 00000001
Screenshot:

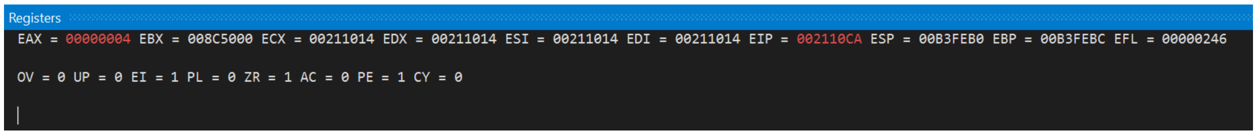


Registers

```
EAX = 00000001 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110C5 ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning with the size of myBytes (1 byte).

Line number: 15
Instruction: mov eax, LENGTHOF myBytes
Register Values: EAX = 00000004
Screenshot:

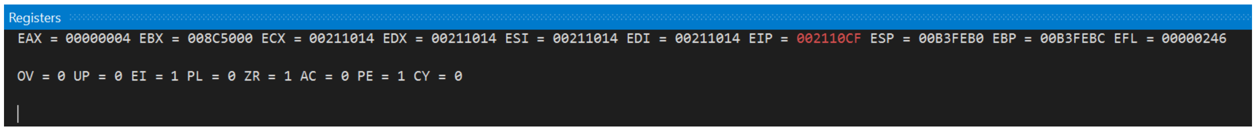


```
EAX = 00000004 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110CA ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning the number of elements in myBytes (4).

(Copy this format as needed)

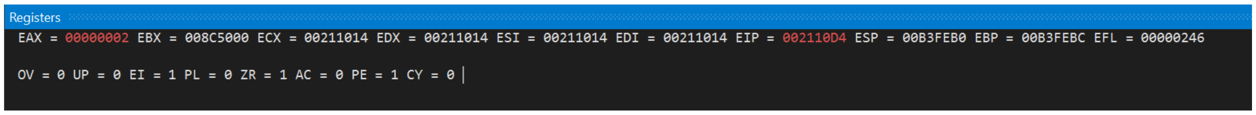
Line number: 16
Instruction: mov eax, SIZEOF myBytes
Register Values: EAX = 00000004
Screenshot:



```
EAX = 00000004 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110CF ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning the size of myBytes (4). In this case, EAX register did not change.

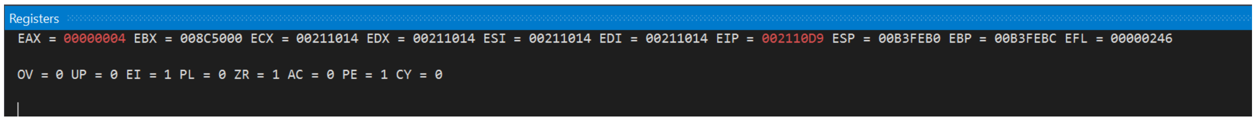
Line number: 17
Instruction: mov eax, TYPE myWords
Register Values: EAX = 00000002
Screenshot:



```
EAX = 00000002 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110D4 ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning with the size of myWords (2 bytes).

Line number: 18
Instruction: mov eax, LENGTHOF myWords
Register Values: EAX = 00000004
Screenshot:

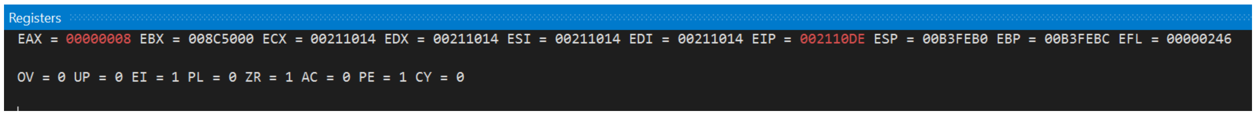


```
EAX = 00000004 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110D9 ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning the number of elements in myWords (4).

Line number: 19

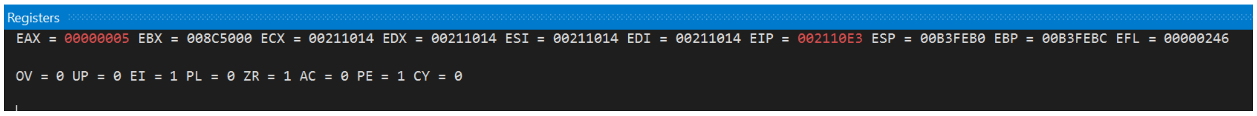
Instruction: mov eax, SIZEOF myWORDs
Register Values: EAX = 00000008
Screenshot:



```
Registers
EAX = 00000008 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110DE ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning the size of myWords (8).

Line number: 20
Instruction: mov eax, SIZEOF myString
Register Values: EAX = 00000005
Screenshot:



```
Registers
EAX = 00000005 EBX = 008C5000 ECX = 00211014 EDX = 00211014 ESI = 00211014 EDI = 00211014 EIP = 002110E3 ESP = 00B3FEB0 EBP = 00B3FEBC EFL = 00000246

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

Explanation: EAX register is updated, returning the size of myString (5).