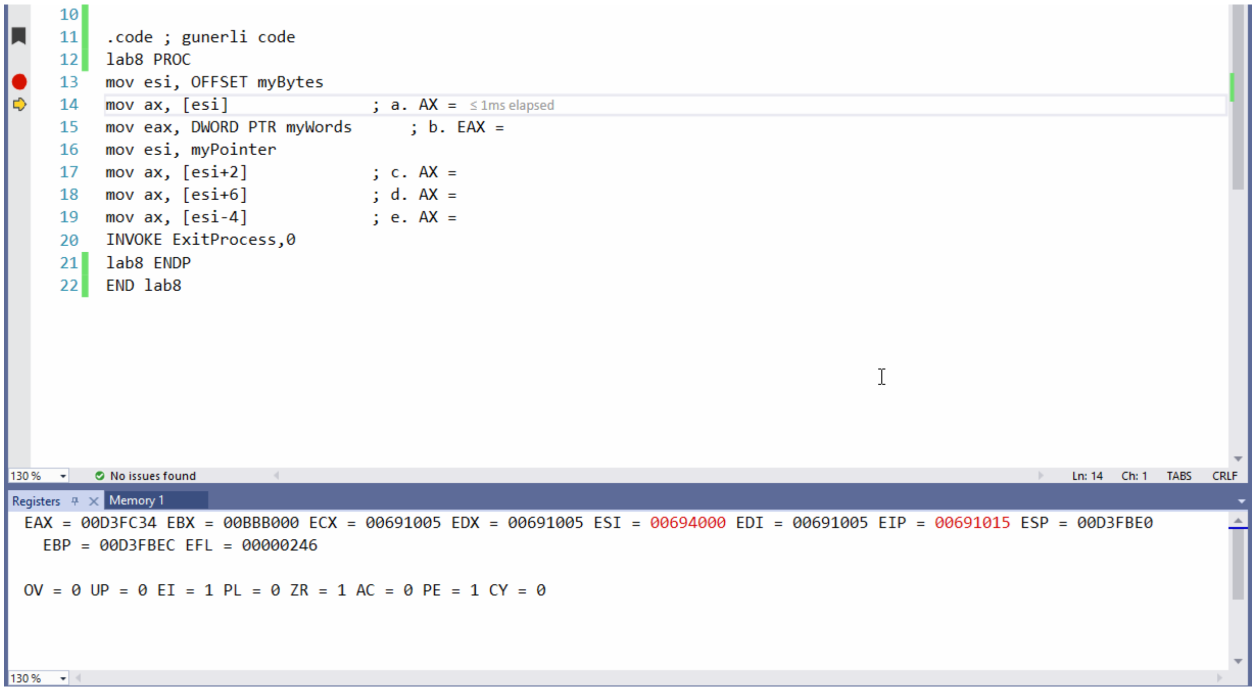
CSC 3210
Computer Organization and Programming
Lab 8
Answer Sheet

Student Name: Hakan Gunerli
Section: Tuesday-Thursday 3:45PM-5PM

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.

Line number: 13
Instruction: `mov esi, OFFSET myBytes`
Register values: `ESI = 00664000`
Screenshot:



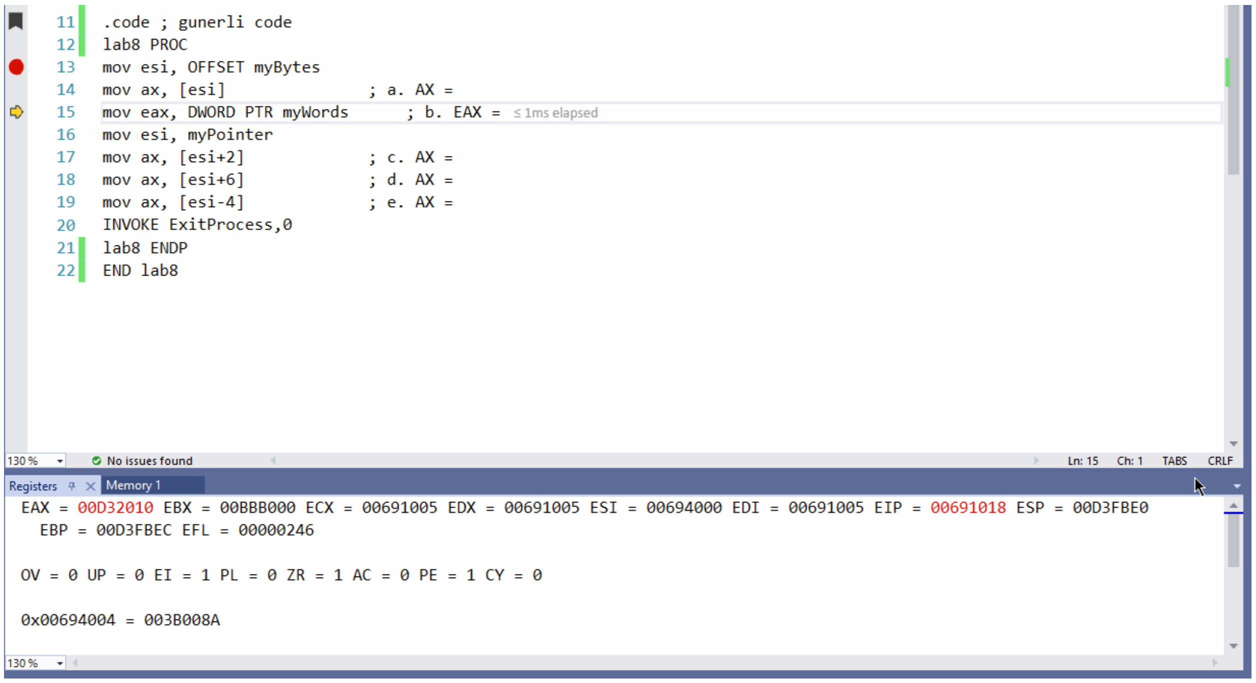
```
.code ; gunerli code
lab8 PROC
mov esi, OFFSET myBytes
mov ax, [esi]              ; a. AX =  ≤ 1ms elapsed
mov eax, DWORD PTR myWords     ; b. EAX =
mov esi, myPointer
mov ax, [esi+2]            ; c. AX =
mov ax, [esi+6]            ; d. AX =
mov ax, [esi-4]            ; e. AX =
INVOKE ExitProcess,0
lab8 ENDP
END lab8
```

```
EAX = 00D3FC34 EBX = 00BBB000 ECX = 00691005 EDX = 00691005 ESI = 00694000 EDI = 00691005 EIP = 00691015 ESP = 00D3FBE0
  EBP = 00D3FBEC EFL = 00000246

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

Explanation:
Offset operator gives the address of the myBytes array and stores it in ESI.

Line number: 14
Instruction: `mov ax, [esi]`
Register values: `EAX = …..2010 ,ESI = 00664000`
Screenshot:



```
.code ; gunerli code
lab8 PROC
mov esi, OFFSET myBytes
mov ax, [esi]               ; a. AX =
mov eax, DWORD PTR myWords      ; b. EAX =  ≤ 1ms elapsed
mov esi, myPointer
mov ax, [esi+2]             ; c. AX =
mov ax, [esi+6]             ; d. AX =
mov ax, [esi-4]             ; e. AX =
INVOKE ExitProcess,0
lab8 ENDP
END lab8
```

```
EAX = 00D32010 EBX = 00BBB000 ECX = 00691005 EDX = 00691005 ESI = 00694000 EDI = 00691005 EIP = 00691018 ESP = 00D3FBE0
  EBP = 00D3FBEC EFL = 00000246

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

0x00694004 = 003B008A
```

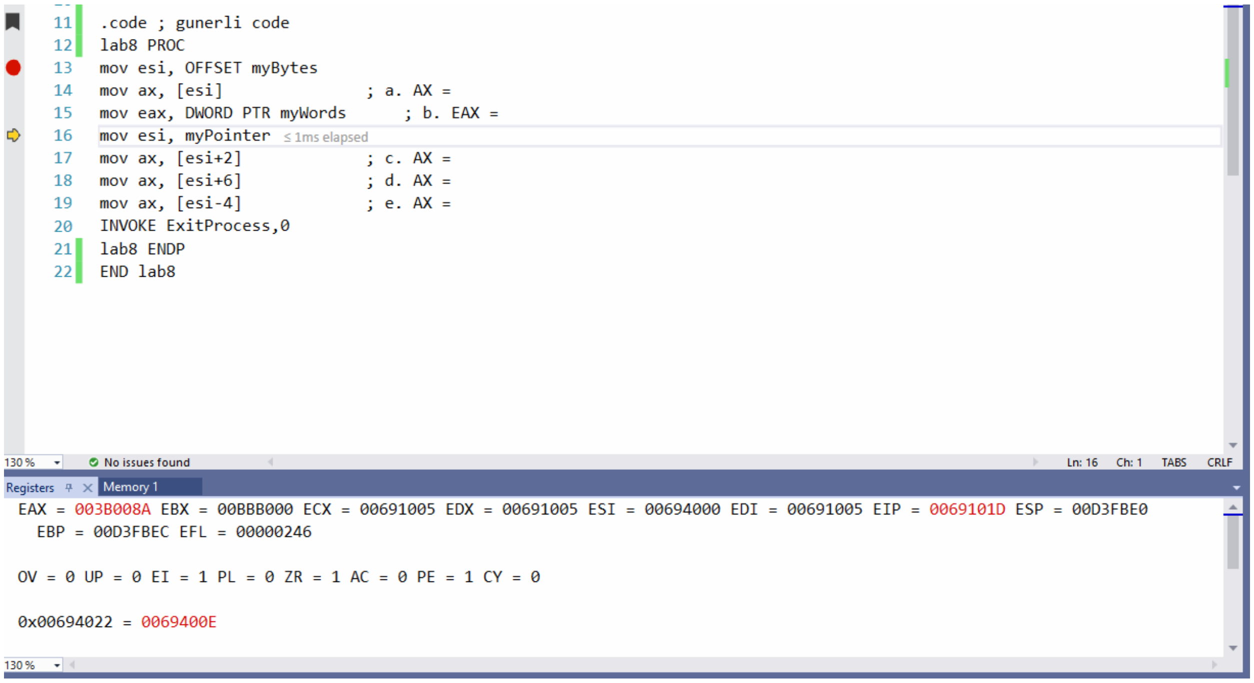
Explanation:
First 16 bits of myBytes, in little endian order

Line number: 15

Instruction: `mov eax, DWORD PTR myWords`

Register values: `EAX = 003B008A, ESI = 005C4000A`

Screenshot:



```
.code ; gunerli code
lab8 PROC
mov esi, OFFSET myBytes
mov ax, [esi]              ; a. AX =
mov eax, DWORD PTR myWords      ; b. EAX =
mov esi, myPointer  ≤ 1ms elapsed
mov ax, [esi+2]            ; c. AX =
mov ax, [esi+6]            ; d. AX =
mov ax, [esi-4]            ; e. AX =
INVOKE ExitProcess,0
lab8 ENDP
END lab8
```

```
EAX = 003B008A EBX = 00BBB000 ECX = 00691005 EDX = 00691005 ESI = 00694000 EDI = 00691005 EIP = 0069101D ESP = 00D3FBE0
  EBP = 00D3FBEC EFL = 00000246

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

0x00694022 = 0069400E
```

Explanation:

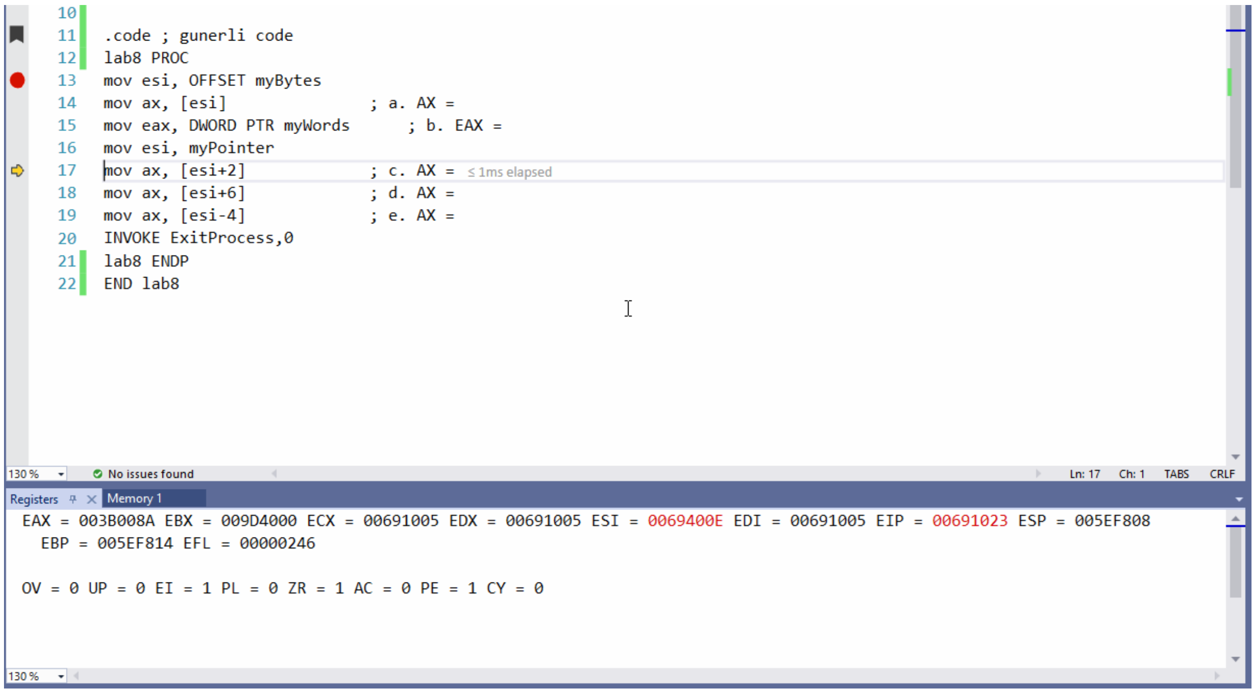
32 bits myBytes access, size overiden by PTR is stored in EAX in little endian order.

Line number: 16

Instruction: `mov esi, myPointer`

Register values: `ESI = 0066400E`

Screenshot:



```
10
11  .code ; gunerli code
12  lab8 PROC
13  mov esi, OFFSET myBytes
14  mov ax, [esi]                ; a. AX =
15  mov eax, DWORD PTR myWords       ; b. EAX =
16  mov esi, myPointer
17  mov ax, [esi+2]              ; c. AX =  ≤ 1ms elapsed
18  mov ax, [esi+6]              ; d. AX =
19  mov ax, [esi-4]              ; e. AX =
20  INVOKE ExitProcess,0
21  lab8 ENDP
22  END lab8
```

130 % · No issues found · Ln: 17 · Ch: 1 · TABS · CRLF

Registers · Memory 1

```
EAX = 003B008A EBX = 009D4000 ECX = 00691005 EDX = 00691005 ESI = 0069400E EDI = 00691005 EIP = 00691023 ESP = 005EF808
  EBP = 005EF814 EFL = 00000246

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

130 %

Explanation: move the address of myPointer to ESI.

Line number: 17-20

Instruction:

mov ax, [esi+2]

mov ax, [esi+6]

mov ax, [esi-4]

Register values:

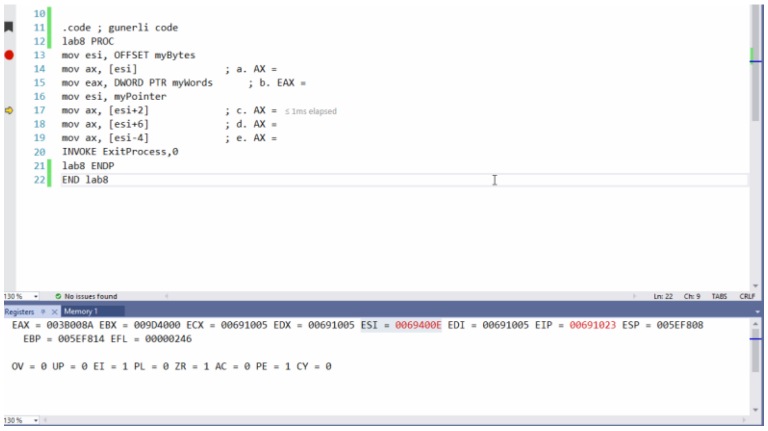
ESI = 0069400E

EAX = 003B0044

EAX = 003B0000

EAX = 003B0044

Screenshot:



```
10
11  .code ; gunerli code
12  lab8 PROC
13  mov esi, OFFSET myBytes
14  mov ax, [esi]                ; a. AX =
15  mov eax, DWORD PTR myWords       ; b. EAX =
16  mov esi, myPointer
17  mov ax, [esi+2]              ; c. AX =  ≤ 1ms elapsed
18  mov ax, [esi+6]              ; d. AX =
19  mov ax, [esi-4]              ; e. AX =
20  INVOKE ExitProcess,0
21  lab8 ENDP
22  END lab8
```

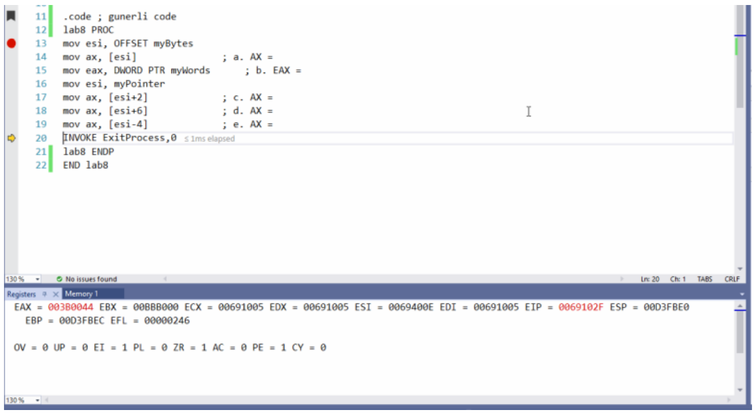
130 % · No issues found · Ln: 22 · Ch: 9 · TABS · CRLF

Registers · Memory 1

```
EAX = 003B008A EBX = 009D4000 ECX = 00691005 EDX = 00691005 ESI = 0069400E EDI = 00691005 EIP = 00691023 ESP = 005EF808
  EBP = 005EF814 EFL = 00000246

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

130 %



```
.code ; gunerli code
lab8 PROC
mov esi, OFFSET myBytes
mov ax, [esi]                ; a. AX =
mov eax, DWORD PTR myWords      ; b. EAX =
mov esi, myPointer
mov ax, [esi+2]              ; c. AX =
mov ax, [esi+6]              ; d. AX =
mov ax, [esi-4]              ; e. AX =
INVOKE ExitProcess,0  ≤ 1ms elapsed
lab8 ENDP
END lab8
```

```
EAX = 003B0044 EBX = 00BBB000 ECX = 00691005 EDX = 00691005 ESI = 0069400E EDI = 00691005 EIP = 0069102F ESP = 00D3FBE0
  EBP = 00D3FBEC EFL = 00000246

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

Explanation:

Esi+2
moves the value 2 places after the address of myPointer to AX in Little Endian order

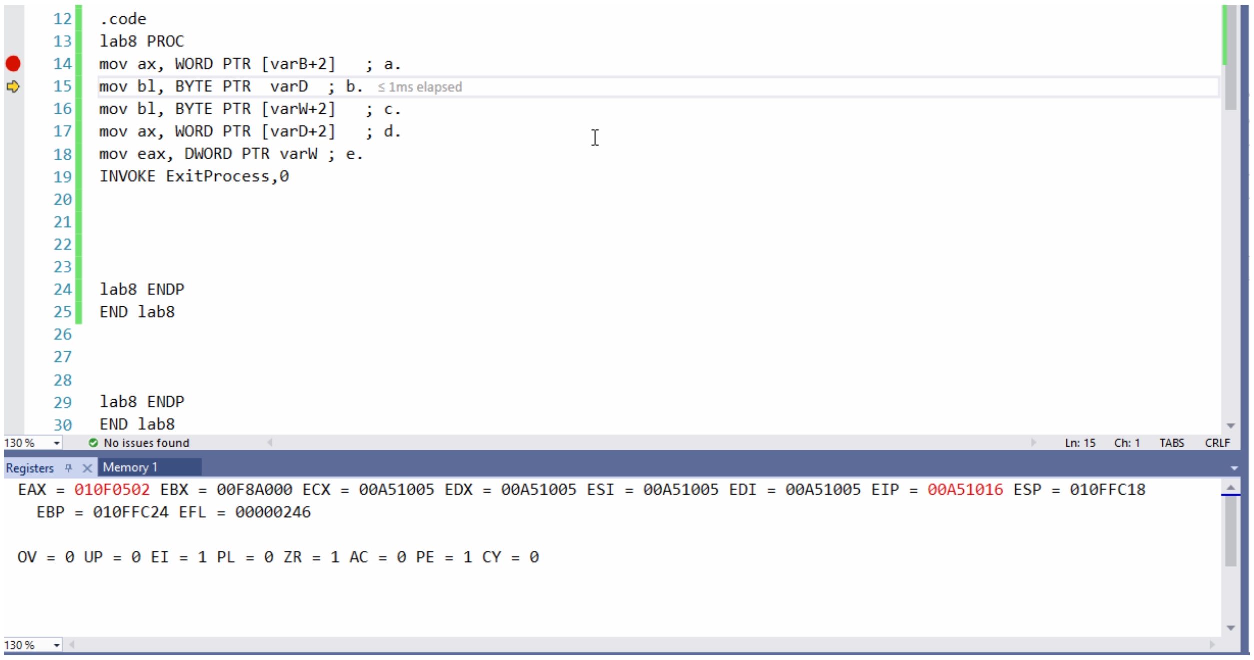
Esi+6, moves the value 6 places after the address of myPointer to AX in Little Endian order (it doesn't matter here as it only has zeroes).

Esi-4 moves the value 4 places before the address of myPointer (value is 4400h) and moves it to AX in little Endian order.

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.

Line number: 14
Instruction: mov ax, WORD PTR [varB+2]
Register values: EAX=….0502
Screenshot:



```
12  .code
13  lab8 PROC
14  mov ax, WORD PTR [varB+2]   ; a.
15  mov bl, BYTE PTR  varD  ; b.  ≤ 1ms elapsed
16  mov bl, BYTE PTR [varW+2]   ; c.
17  mov ax, WORD PTR [varD+2]   ; d.
18  mov eax, DWORD PTR varW ; e.
19  INVOKE ExitProcess,0
20
21
22
23
24  lab8 ENDP
25  END lab8
26
27
28
29  lab8 ENDP
30  END lab8
```

130 %   No issues found   Ln: 15   Ch: 1   TABS   CRLF

Registers   Memory 1

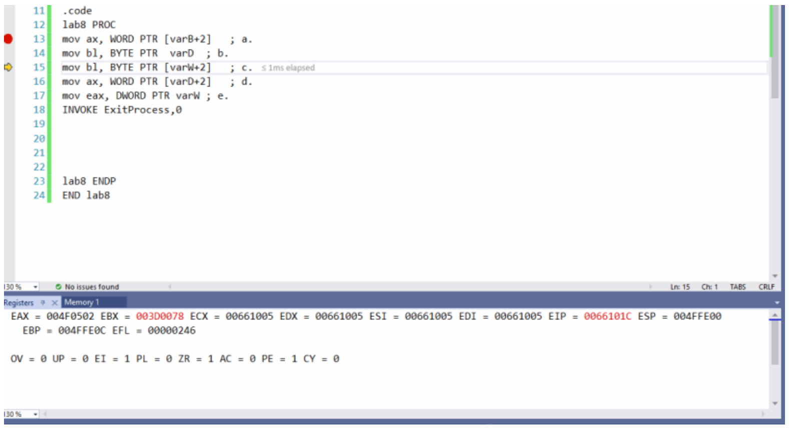
```
 EAX = 010F0502 EBX = 00F8A000 ECX = 00A51005 EDX = 00A51005 ESI = 00A51005 EDI = 00A51005 EIP = 00A51016 ESP = 010FFC18
   EBP = 010FFC24 EFL = 00000246

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

130 %

Explanation:
Moves the last 2 elements of varB to ax in little endian order.

Line number: 15
Instruction: `mov bl, BYTE PTR  varD`
Register values: EAX=….0502, EBX = ……78
Screenshot:



```
11  .code
12  lab8 PROC
13  mov ax, WORD PTR [varB+2]   ; a.
14  mov bl, BYTE PTR  varD  ; b.
15  mov bl, BYTE PTR [varW+2]   ; c.  ≤ 1ms elapsed
16  mov ax, WORD PTR [varD+2]   ; d.
17  mov eax, DWORD PTR varW ; e.
18  INVOKE ExitProcess,0
19
20
21
22
23  lab8 ENDP
24  END lab8
```

```
EAX = 004F0502 EBX = 003D0078 ECX = 00661005 EDX = 00661005 ESI = 00661005 EDI = 00661005 EIP = 0066101C ESP = 004FFE00
  EBP = 004FFE0C EFL = 00000246

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

Explanation:

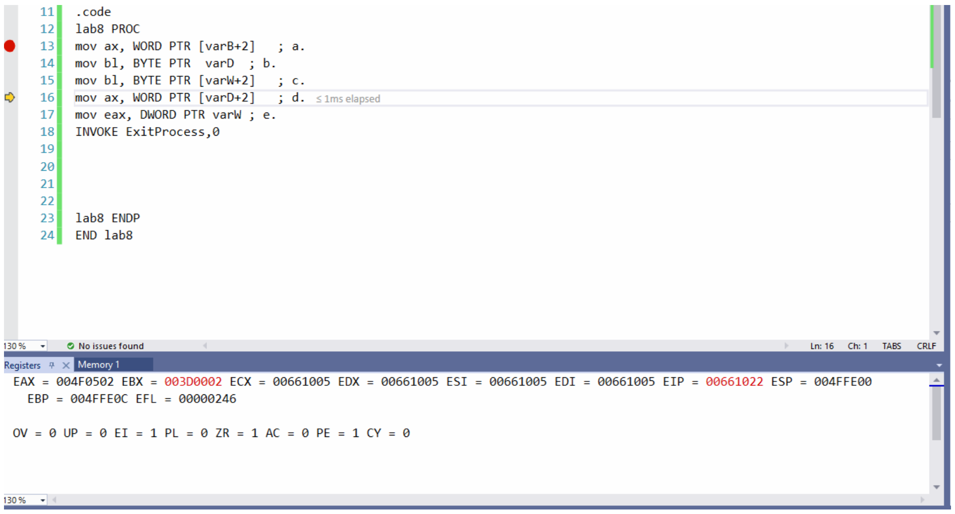
Moves the last two digits of varD into EAX (78)

Line number: 16

Instruction: `mov bl, BYTE PTR [varW+2]`

Register values: EAX=….0502, EBX = ……02

Screenshot:



```
11  .code
12  lab8 PROC
13  mov ax, WORD PTR [varB+2]   ; a.
14  mov bl, BYTE PTR  varD  ; b.
15  mov bl, BYTE PTR [varW+2]   ; c.
16  mov ax, WORD PTR [varD+2]   ; d.  ≤ 1ms elapsed
17  mov eax, DWORD PTR varW ; e.
18  INVOKE ExitProcess,0
19
20
21
22
23  lab8 ENDP
24  END lab8
```

```
EAX = 004F0502 EBX = 003D0002 ECX = 00661005 EDX = 00661005 ESI = 00661005 EDI = 00661005 EIP = 00661022 ESP = 004FFE00
  EBP = 004FFE0C EFL = 00000246

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

Explanation:

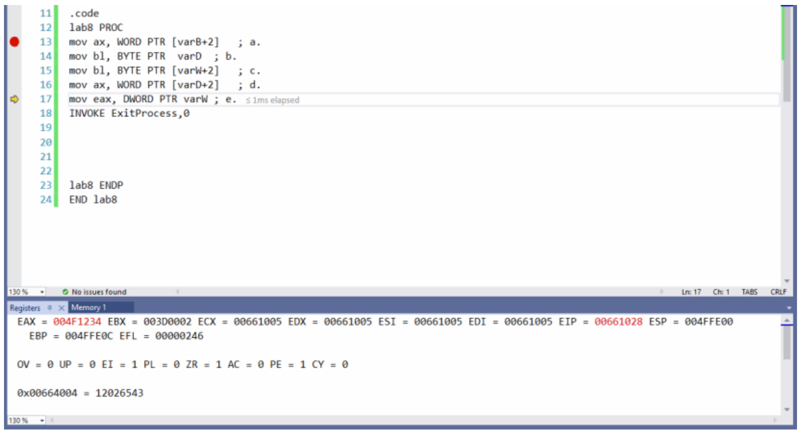
varB is turned into BYTE and moved to BL. (last 2 digits 02)

Line number: 17

Instruction: `mov ax, WORD PTR [varD+2]`

Register values: EAX=….1234, EBX = ……02

Screenshot:



```
11  .code
12  lab8 PROC
13  mov ax, WORD PTR [varB+2]   ; a.
14  mov bl, BYTE PTR  varD  ; b.
15  mov bl, BYTE PTR [varW+2]   ; c.
16  mov ax, WORD PTR [varD+2]   ; d.
17  mov eax, DWORD PTR varW ; e.  ≤ 1ms elapsed
18  INVOKE ExitProcess,0
19
20
21
22
23  lab8 ENDP
24  END lab8
```

```
EAX = 004F1234 EBX = 003D0002 ECX = 00661005 EDX = 00661005 ESI = 00661005 EDI = 00661005 EIP = 00661028 ESP = 004FFE00
  EBP = 004FFE0C EFL = 00000246

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

0x00664004 = 12026543
```

Explanation:

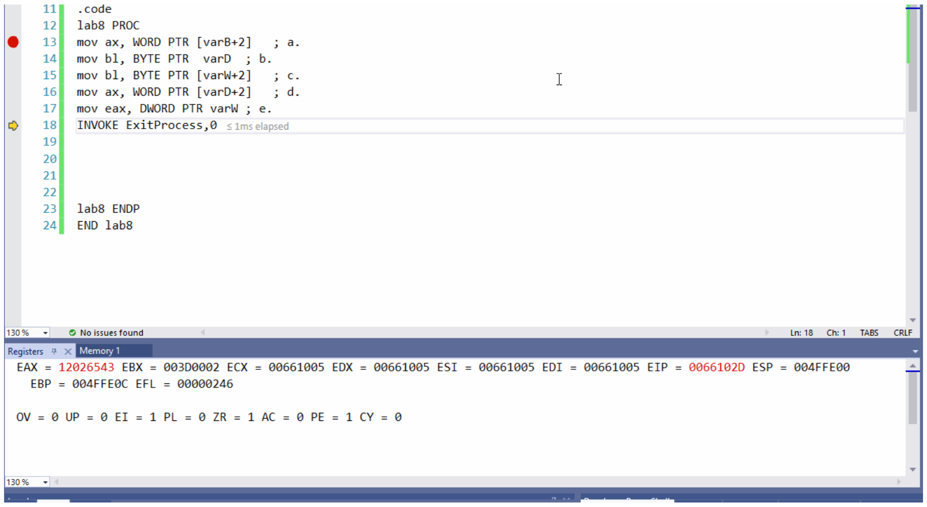
Moves the values of varD into AX, the first 4 digits.

Line number: 18

Instruction: `mov eax, DWORD PTR varW`

Register values: EAX= 12026543 , EBX = ……02

Screenshot:



```
11  .code
12  lab8 PROC
13  mov ax, WORD PTR [varB+2]   ; a.
14  mov bl, BYTE PTR  varD  ; b.
15  mov bl, BYTE PTR [varW+2]   ; c.
16  mov ax, WORD PTR [varD+2]   ; d.
17  mov eax, DWORD PTR varW ; e.
18  INVOKE ExitProcess,0  ≤ 1ms elapsed
19
20
21
22
23  lab8 ENDP
24  END lab8
```

```
EAX = 12026543 EBX = 003D0002 ECX = 00661005 EDX = 00661005 ESI = 00661005 EDI = 00661005 EIP = 0066102D ESP = 004FFE00
  EBP = 004FFE0C EFL = 00000246

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

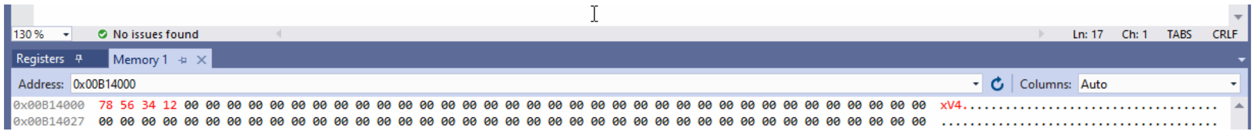
Explanation:

This moved varW values into EAX in little endian order

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.

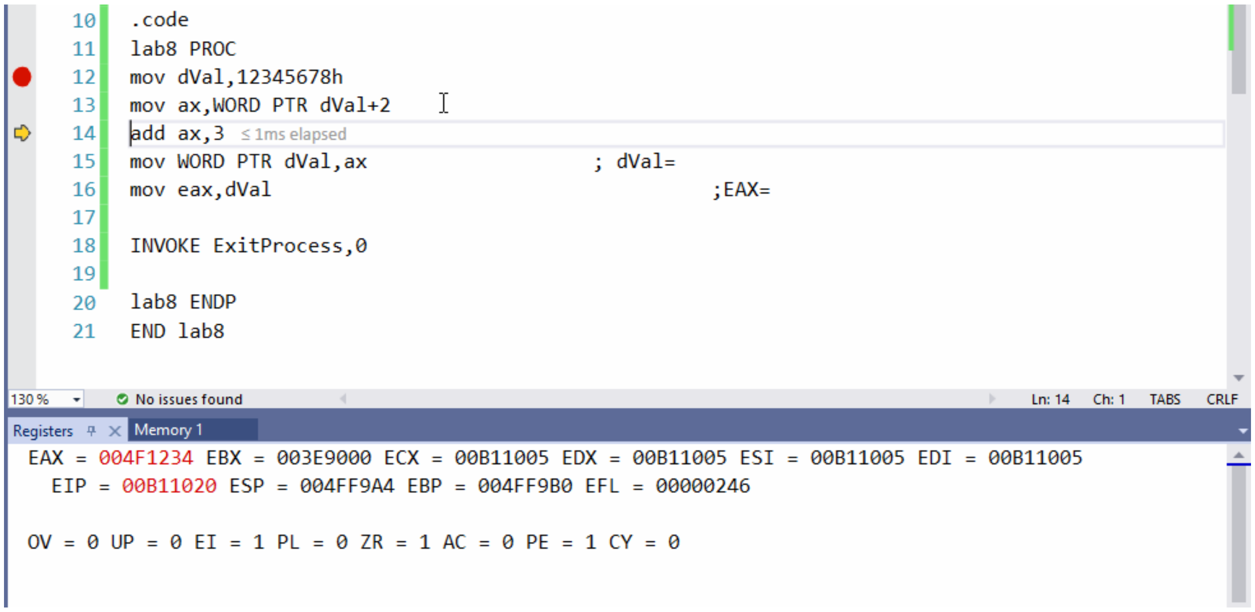
Line number: 12
Instruction: mov dVal, 12345678
Register / variable content: dVal= 12345678
Screenshot:



```
Address: 0x00B14000                Columns: Auto
0x00B14000  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  xV4....................................
0x00B14027  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  .......................................
```

Explanation:
Values 2 digits at a time in little endian order

Line number: 13
Instruction: mov ax, WORD PTR dVal+2
Register / variable content: EAX = _ _ _ _ 1234h, dVal = 12345678h
Screenshot:



```
10  .code
11  lab8 PROC
12  mov dVal,12345678h
13  mov ax,WORD PTR dVal+2
14  add ax,3
15  mov WORD PTR dVal,ax                ; dVal=
16  mov eax,dVal                                  ;EAX=
17
18  INVOKE ExitProcess,0
19
20  lab8 ENDP
21  END lab8
```

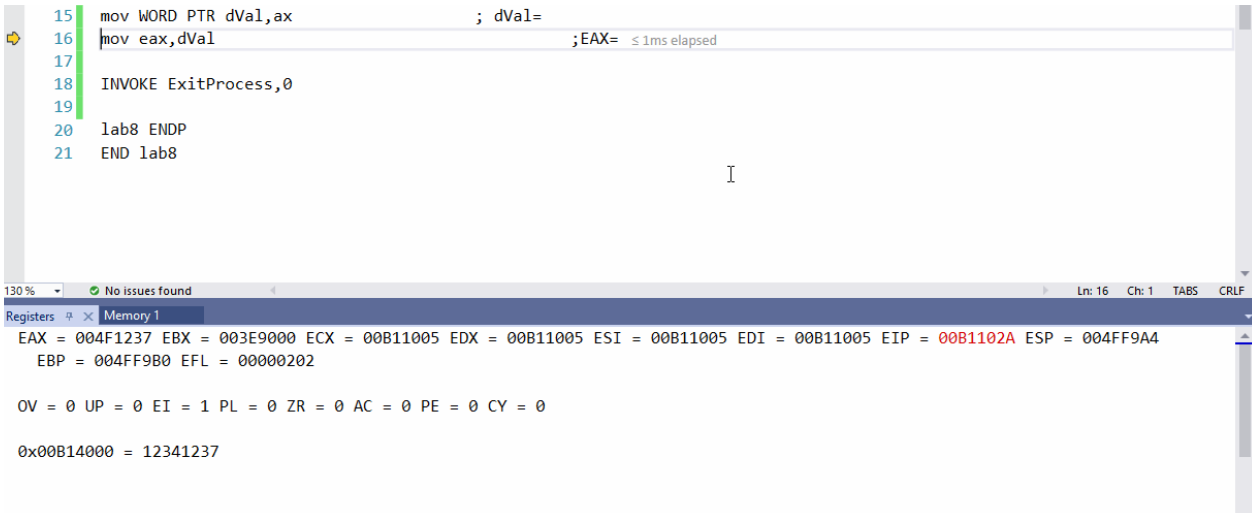
```
EAX = 004F1234 EBX = 003E9000 ECX = 00B11005 EDX = 00B11005 ESI = 00B11005 EDI = 00B11005
  EIP = 00B11020 ESP = 004FF9A4 EBP = 004FF9B0 EFL = 00000246

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

Explanation:
The second part of dVal and store in AX in little endian order.

Line number: 14
Instruction: add ax,3
Register / variable content: EAX = _ _ _ _ 1237h, dVal = 12345678h

Screenshot:



```
15  mov WORD PTR dVal,ax                ; dVal=
16  mov eax,dVal                                  ;EAX=  ≤ 1ms elapsed
17
18  INVOKE ExitProcess,0
19
20  lab8 ENDP
21  END lab8
```

```
130 %    No issues found    Ln: 16   Ch: 1   TABS   CRLF
Registers    Memory 1
 EAX = 004F1237 EBX = 003E9000 ECX = 00B11005 EDX = 00B11005 ESI = 00B11005 EDI = 00B11005 EIP = 00B1102A ESP = 004FF9A4
   EBP = 004FF9B0 EFL = 00000202

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

0x00B14000 = 12341237
```

Explanation:

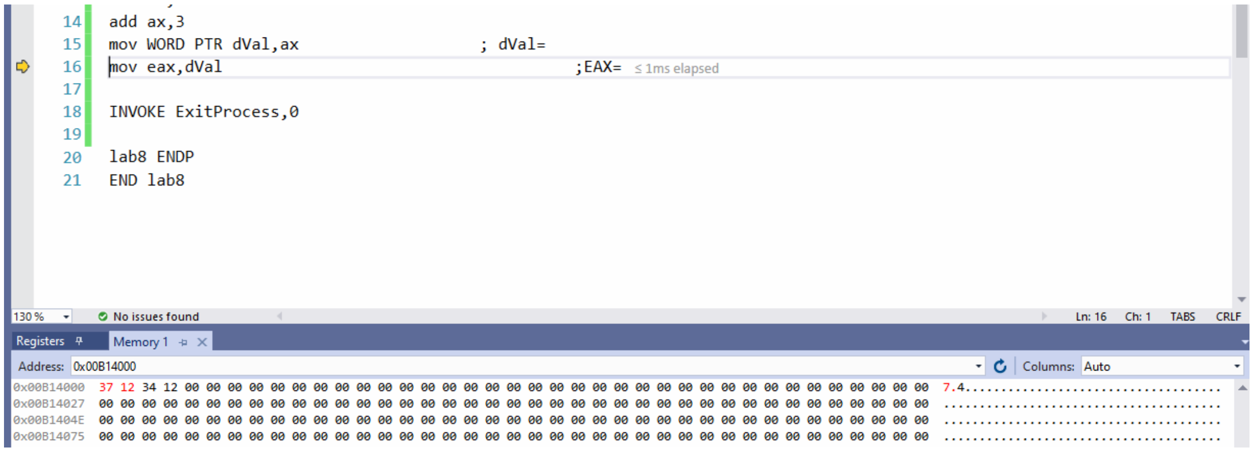
Adds 3 into AX, making the end 1237.

Line number: 15

Instruction: mov WORD PTR dVal, ax

Register / variable content: dVal = 12341237h, EAX = _ _ _ _ 1237h

Screenshot:



```
14  add ax,3
15  mov WORD PTR dVal,ax                ; dVal=
16  mov eax,dVal                                  ;EAX=  ≤ 1ms elapsed
17
18  INVOKE ExitProcess,0
19
20  lab8 ENDP
21  END lab8
```

```
130 %    No issues found    Ln: 16   Ch: 1   TABS   CRLF
Registers    Memory 1
Address: 0x00B14000                    Columns: Auto
0x00B14000  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  7.4....................................
0x00B14027  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  .......................................
0x00B1404E  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  .......................................
0x00B14075  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  .......................................
```

Explanation:

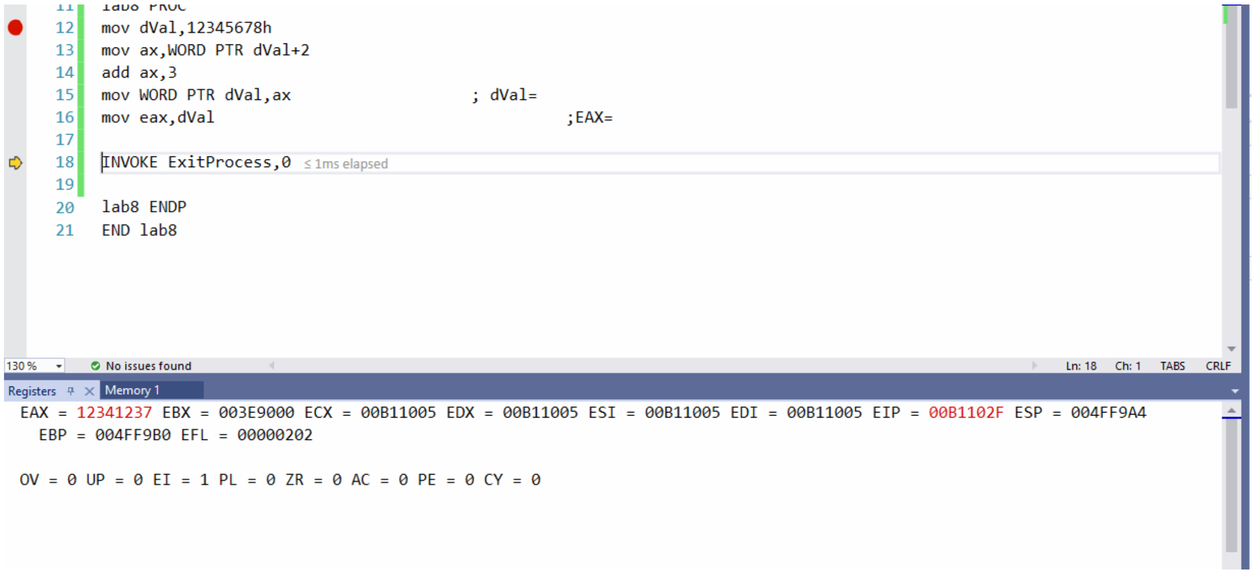
Last 4 digits and store it in little endian order.

Line number: 17

Instruction: mov eax, dVal

Register / variable content: dVal = 12341237h, EAX = 12341237h

Screenshot:



```
lab8 PROC
mov dVal,12345678h
mov ax,WORD PTR dVal+2
add ax,3
mov WORD PTR dVal,ax                    ; dVal=
mov eax,dVal                                     ;EAX=

INVOKE ExitProcess,0  ≤ 1ms elapsed

lab8 ENDP
END lab8
```

130 %   No issues found   Ln: 18   Ch: 1   TABS   CRLF

Registers   Memory 1

```
 EAX = 12341237 EBX = 003E9000 ECX = 00B11005 EDX = 00B11005 ESI = 00B11005 EDI = 00B11005 EIP = 00B1102F ESP = 004FF9A4
   EBP = 004FF9B0 EFL = 00000202

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

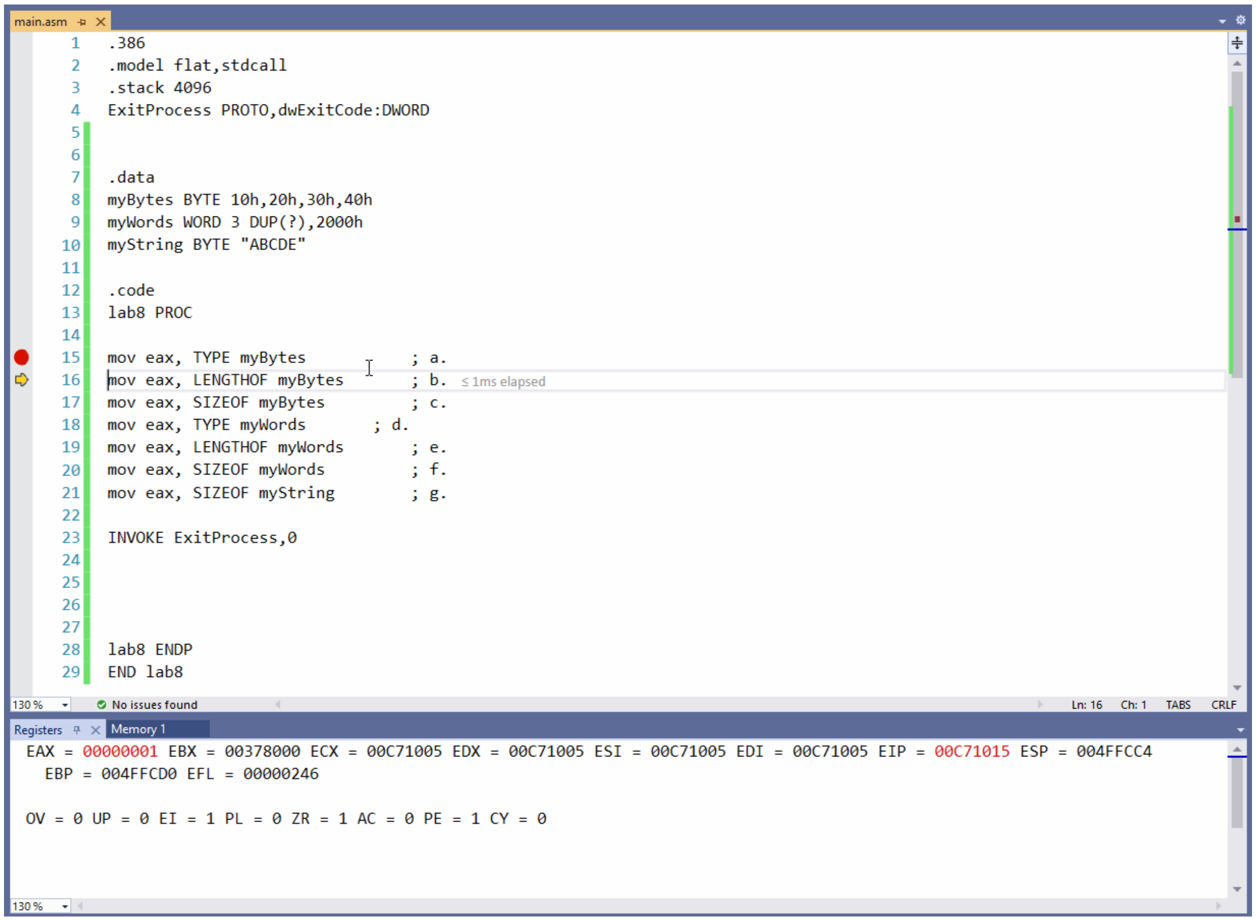
Explanation:

value in dVal to EAX in Little Endian order.

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.

Line number: 15
Instruction: `mov eax, TYPE myBytes`
Register values: `EAX = 00000001`
Screenshot:



main.asm

```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes          ; a.
mov eax, LENGTHOF myBytes      ; b.  ≤ 1ms elapsed
mov eax, SIZEOF myBytes        ; c.
mov eax, TYPE myWords      ; d.
mov eax, LENGTHOF myWords      ; e.
mov eax, SIZEOF myWords        ; f.
mov eax, SIZEOF myString       ; g.

INVOKE ExitProcess,0




lab8 ENDP
END lab8
```

130 % No issues found Ln: 16 Ch: 1 TABS CRLF

Registers Memory 1

```
EAX = 00000001 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C71015 ESP = 004FFCC4
  EBP = 004FFCD0 EFL = 00000246

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

130 %

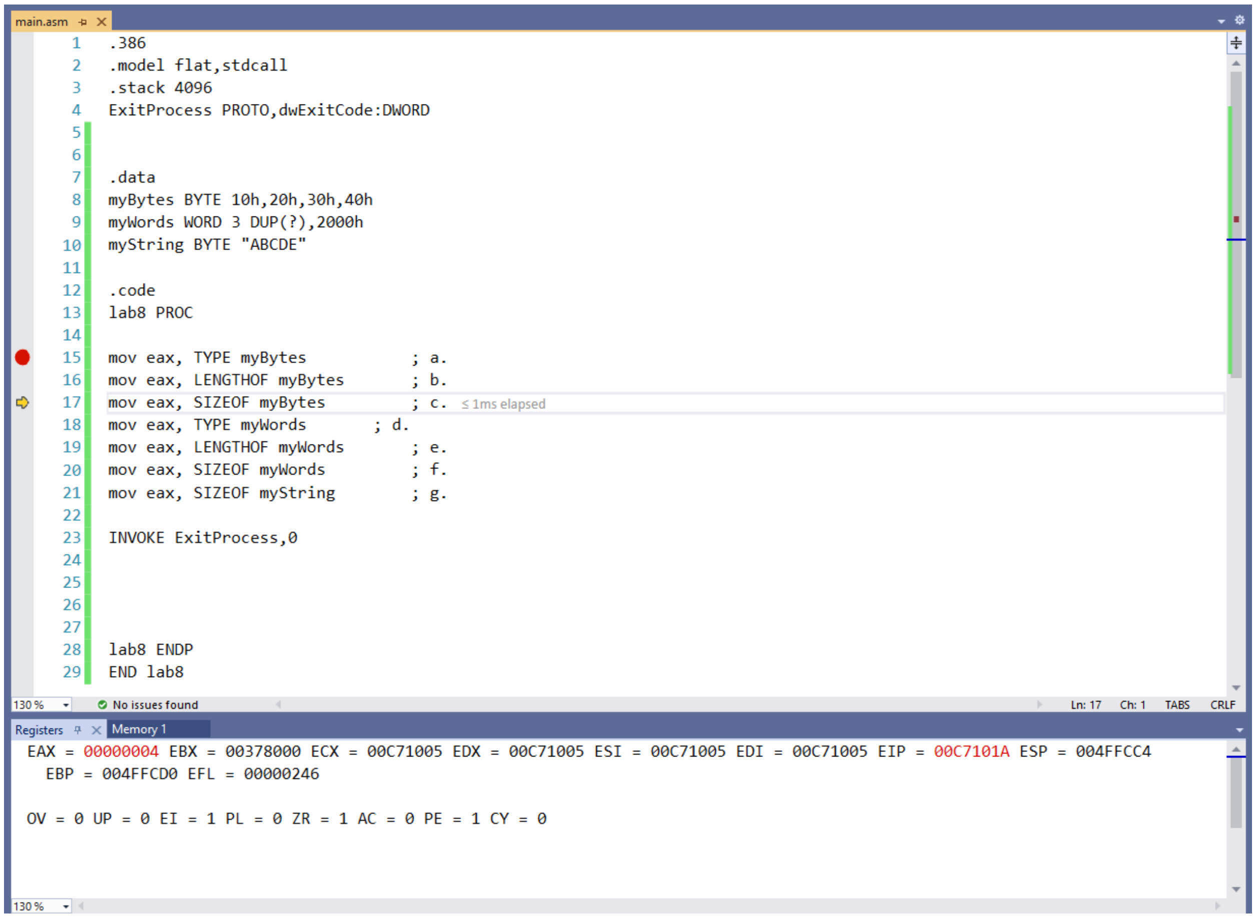
Explanation:
The value BYTE is 1 bytes. Word is 2, therefore the value stored for myBytes is 1.

Line number: 16

Instruction: `mov eax, LENGTHOF myBytes`

Register values: `EAX = 00000004`

Screenshot:



```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes           ; a.
mov eax, LENGTHOF myBytes       ; b.
mov eax, SIZEOF myBytes         ; c.
mov eax, TYPE myWords       ; d.
mov eax, LENGTHOF myWords       ; e.
mov eax, SIZEOF myWords         ; f.
mov eax, SIZEOF myString        ; g.

INVOKE ExitProcess,0




lab8 ENDP
END lab8
```

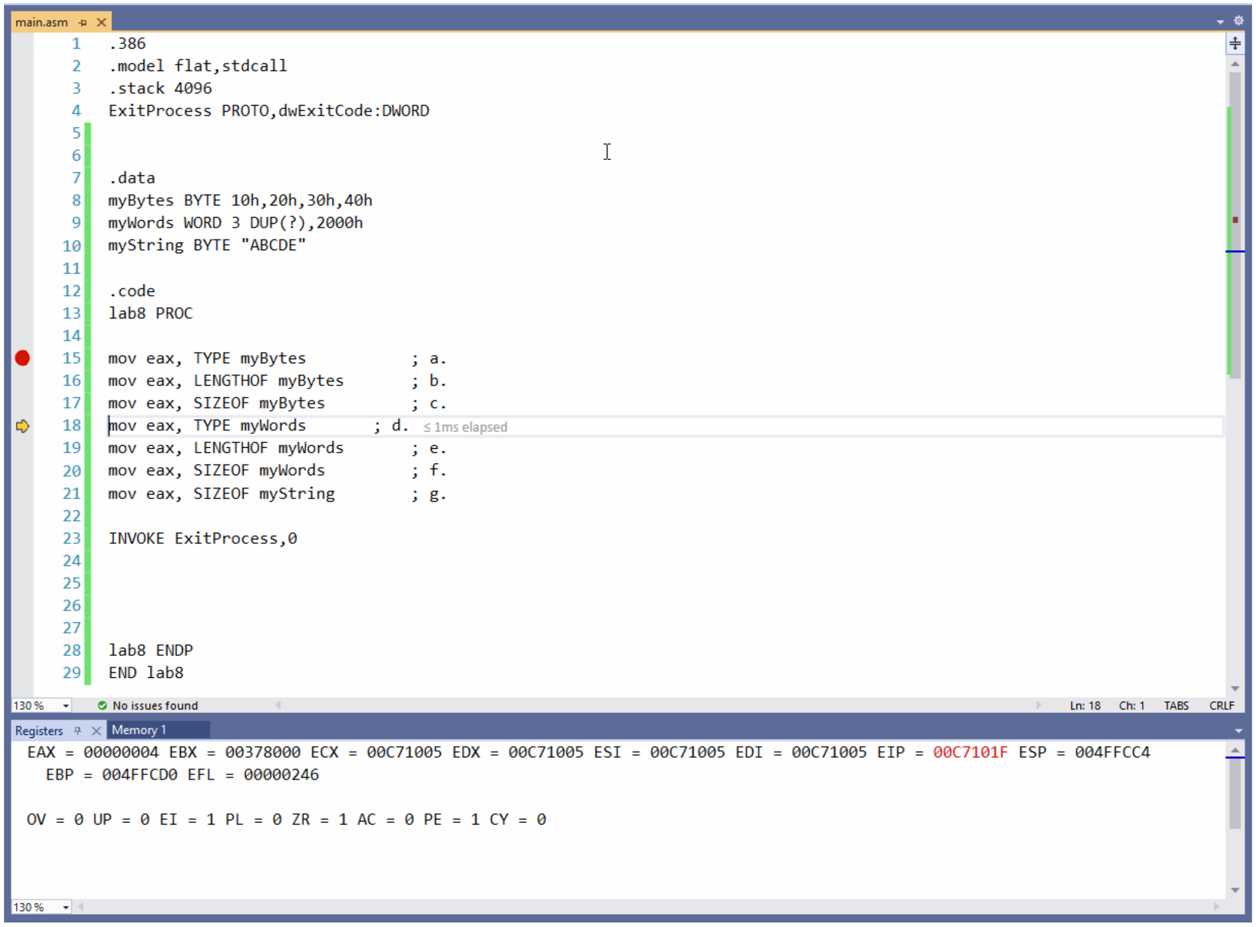
```
EAX = 00000004 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C7101A ESP = 004FFCC4
  EBP = 004FFCD0 EFL = 00000246

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

Explanation:

The length of the myBytes array is 4 elements.

Line number: 17
Instruction: `mov eax, SIZEOF myBytes`
Register values: `EAX = 00000004`
Screenshot:



main.asm

```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes          ; a.
mov eax, LENGTHOF myBytes      ; b.
mov eax, SIZEOF myBytes        ; c.
mov eax, TYPE myWords      ; d.  ≤ 1ms elapsed
mov eax, LENGTHOF myWords      ; e.
mov eax, SIZEOF myWords        ; f.
mov eax, SIZEOF myString       ; g.

INVOKE ExitProcess,0




lab8 ENDP
END lab8
```

130 % No issues found Ln: 18 Ch: 1 TABS CRLF

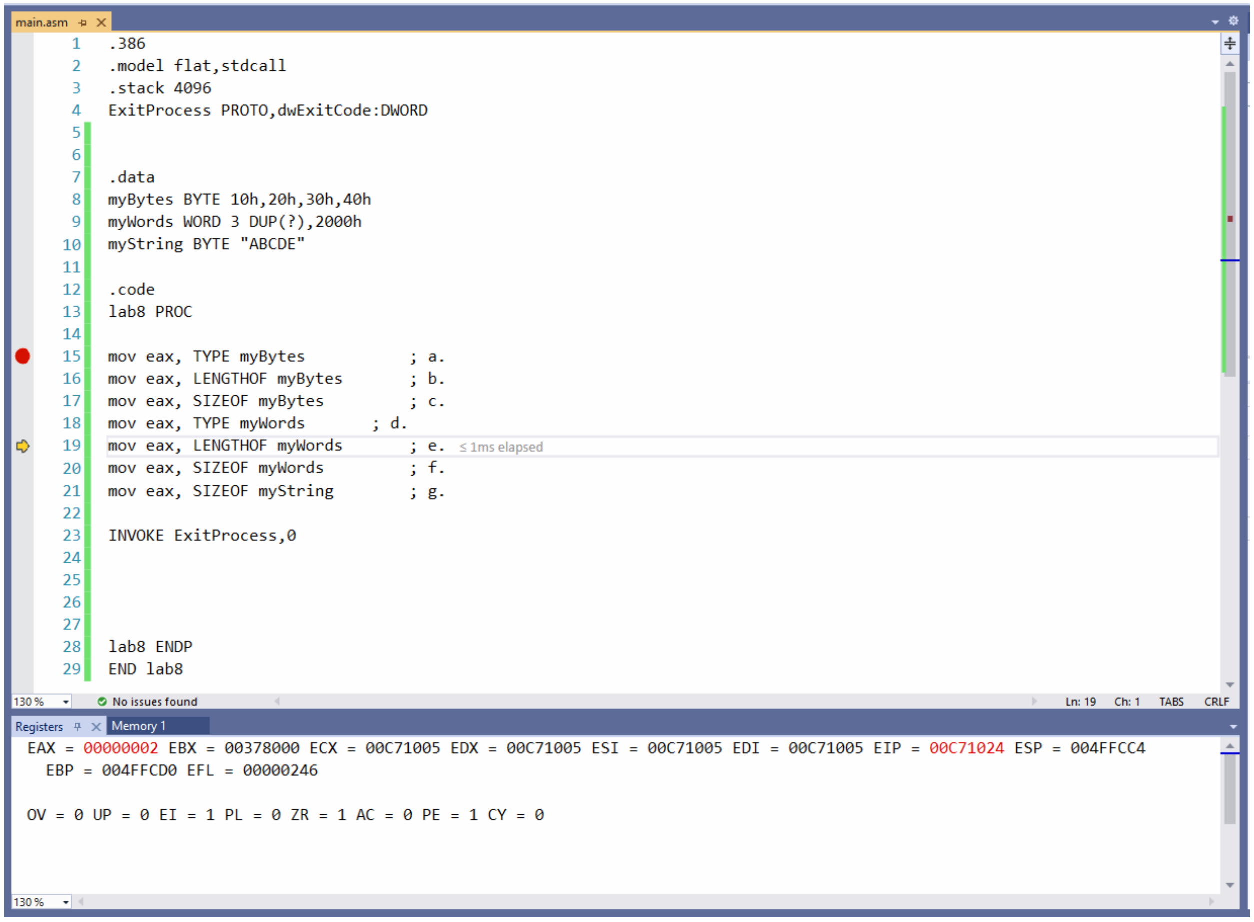
Registers Memory 1

```
EAX = 00000004 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C7101F ESP = 004FFCC4
  EBP = 004FFCD0 EFL = 00000246

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

Explanation:
Moves the size of all elements of myBtets into eax.

Line number: 18
Instruction: `mov eax, TYPE myWords`
Register values: `EAX = 00000002`
Screenshot:



```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes          ; a.
mov eax, LENGTHOF myBytes      ; b.
mov eax, SIZEOF myBytes        ; c.
mov eax, TYPE myWords      ; d.
mov eax, LENGTHOF myWords      ; e.
mov eax, SIZEOF myWords        ; f.
mov eax, SIZEOF myString       ; g.

INVOKE ExitProcess,0




lab8 ENDP
END lab8
```

```
EAX = 00000002 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C71024 ESP = 004FFCC4
  EBP = 004FFCD0 EFL = 00000246

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

Explanation:

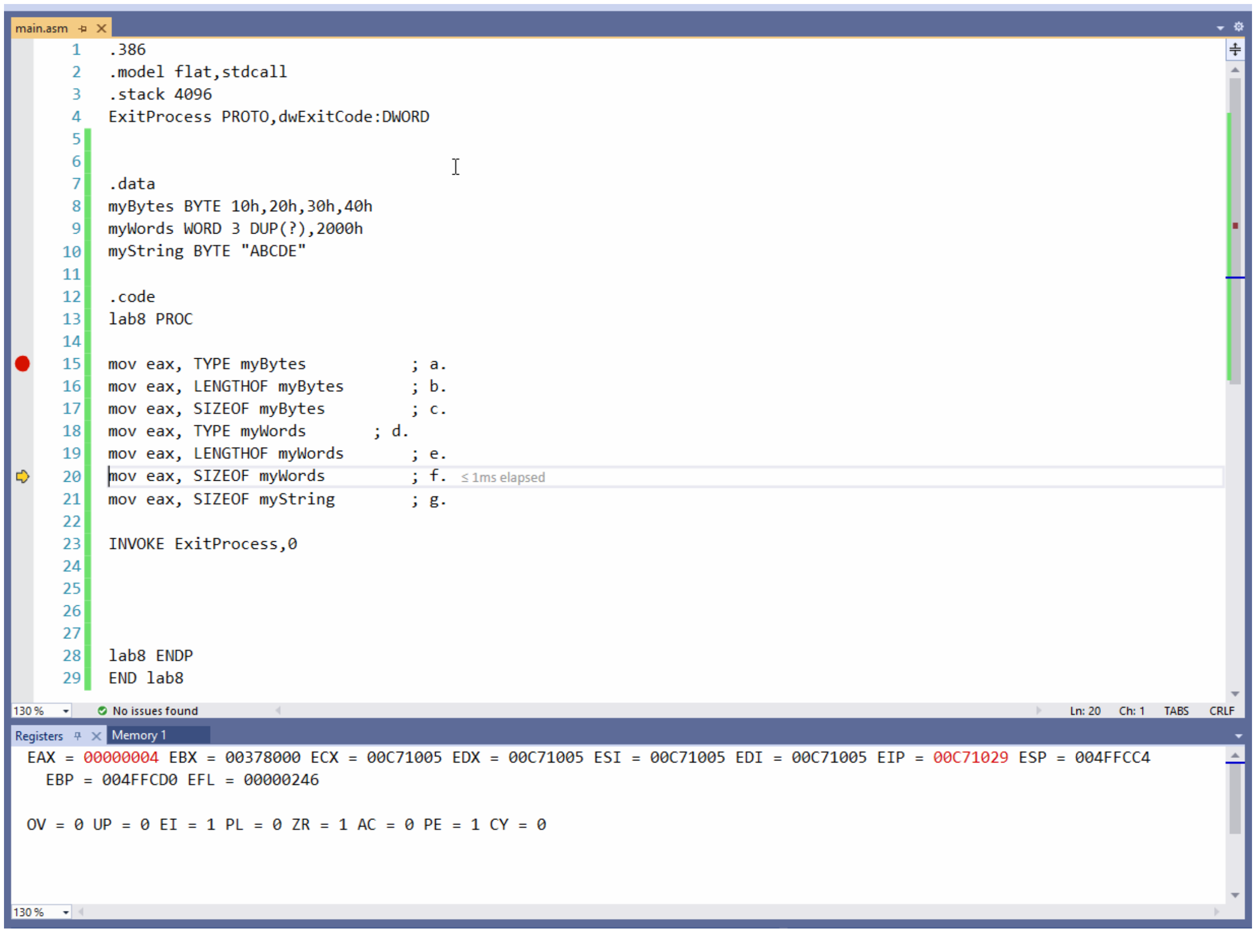
Size of each element into eax.

Line number: 19

Instruction: `mov eax, LENGTHOF myWords`

Register values: `EAX = 00000004`

Screenshot:


main.asm

```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes          ; a.
mov eax, LENGTHOF myBytes      ; b.
mov eax, SIZEOF myBytes        ; c.
mov eax, TYPE myWords      ; d.
mov eax, LENGTHOF myWords      ; e.
mov eax, SIZEOF myWords        ; f.  ≤ 1ms elapsed
mov eax, SIZEOF myString       ; g.

INVOKE ExitProcess,0




lab8 ENDP
END lab8
```

130 %   No issues found   Ln: 20   Ch: 1   TABS   CRLF

Registers   Memory 1

```
 EAX = 00000004 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C71029 ESP = 004FFCC4
   EBP = 004FFCD0 EFL = 00000246

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

Explanation:

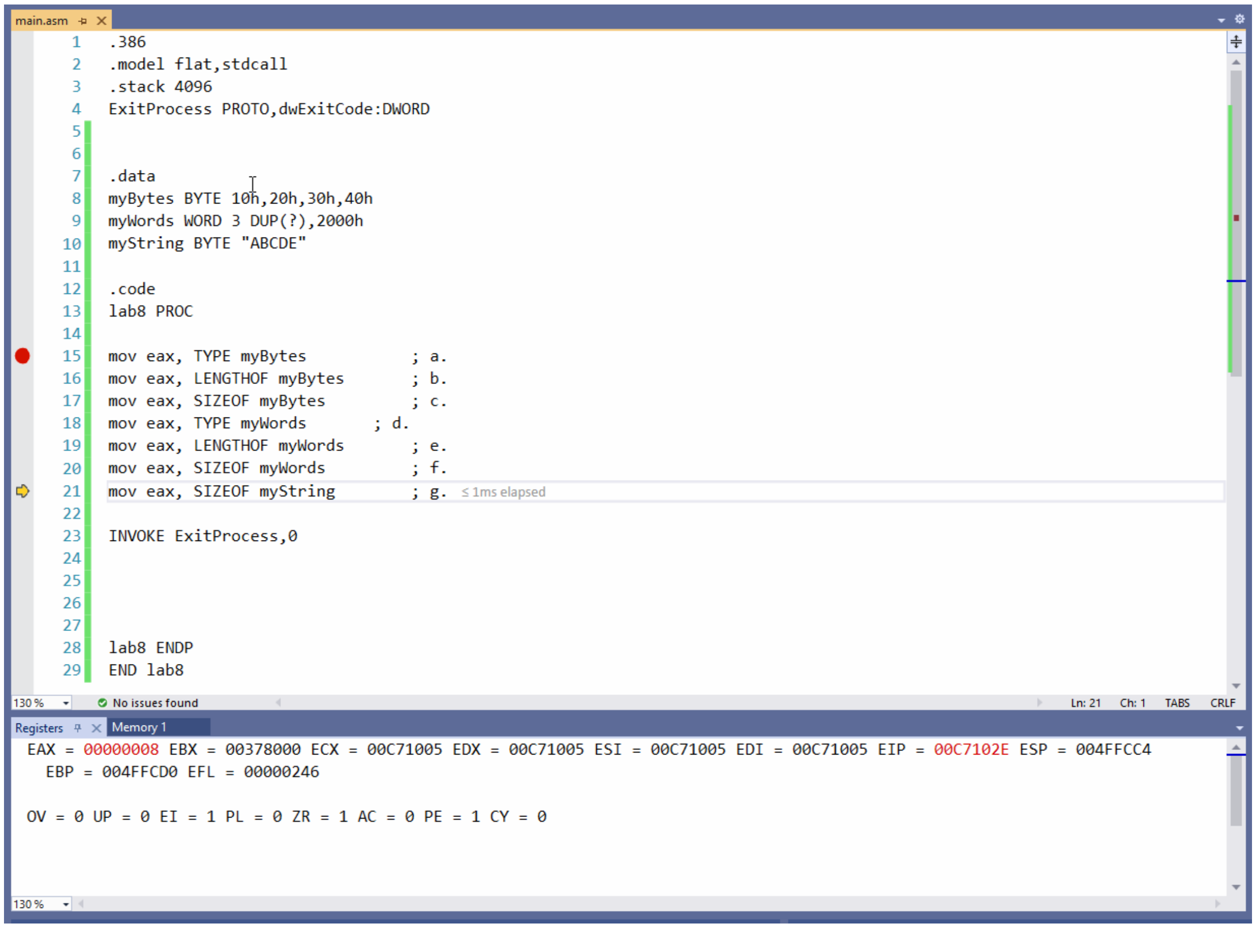
The amount of elements in myWords moved to eax.

Line number: 20

Instruction: `mov eax, SIZEOF myWords`

Register values: `EAX = 00000008`

Screenshot:



```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes          ; a.
mov eax, LENGTHOF myBytes      ; b.
mov eax, SIZEOF myBytes        ; c.
mov eax, TYPE myWords      ; d.
mov eax, LENGTHOF myWords      ; e.
mov eax, SIZEOF myWords        ; f.
mov eax, SIZEOF myString       ; g.

INVOKE ExitProcess,0




lab8 ENDP
END lab8
```

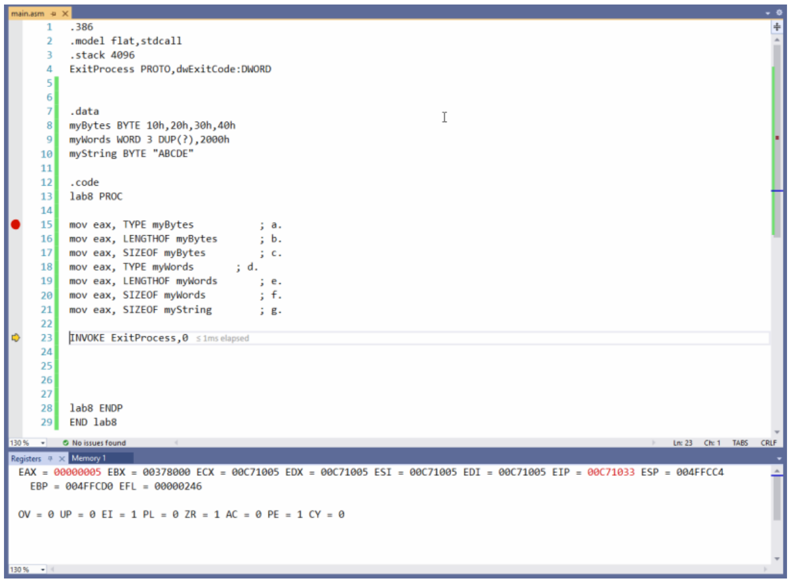
```
EAX = 00000008 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C7102E ESP = 004FFCC4
  EBP = 004FFCD0 EFL = 00000246

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

Explanation:

Size of myWords array into eax.

Line number: 21
Instruction: `mov eax, SIZEOF myString`
Register values: `EAX = 00000005`
Screenshot:



```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD


.data
myBytes BYTE 10h,20h,30h,40h
myWords WORD 3 DUP(?),2000h
myString BYTE "ABCDE"

.code
lab8 PROC

mov eax, TYPE myBytes           ; a.
mov eax, LENGTHOF myBytes       ; b.
mov eax, SIZEOF myBytes         ; c.
mov eax, TYPE myWords       ; d.
mov eax, LENGTHOF myWords       ; e.
mov eax, SIZEOF myWords         ; f.
mov eax, SIZEOF myString        ; g.

INVOKE ExitProcess,0  ≤ 1ms elapsed




lab8 ENDP
END lab8
```

```
EAX = 00000005 EBX = 00378000 ECX = 00C71005 EDX = 00C71005 ESI = 00C71005 EDI = 00C71005 EIP = 00C71033 ESP = 004FFCC4
  EBP = 004FFCD0 EFL = 00000246

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

Explanation:

Move the length of abcde into eax.