CSC 3210

Computer Organization and Programming

Lab 8

Answer Sheet

Student Name: **Aparna Mandapaka**

Section:

Debug through each line of code and explain the register content.

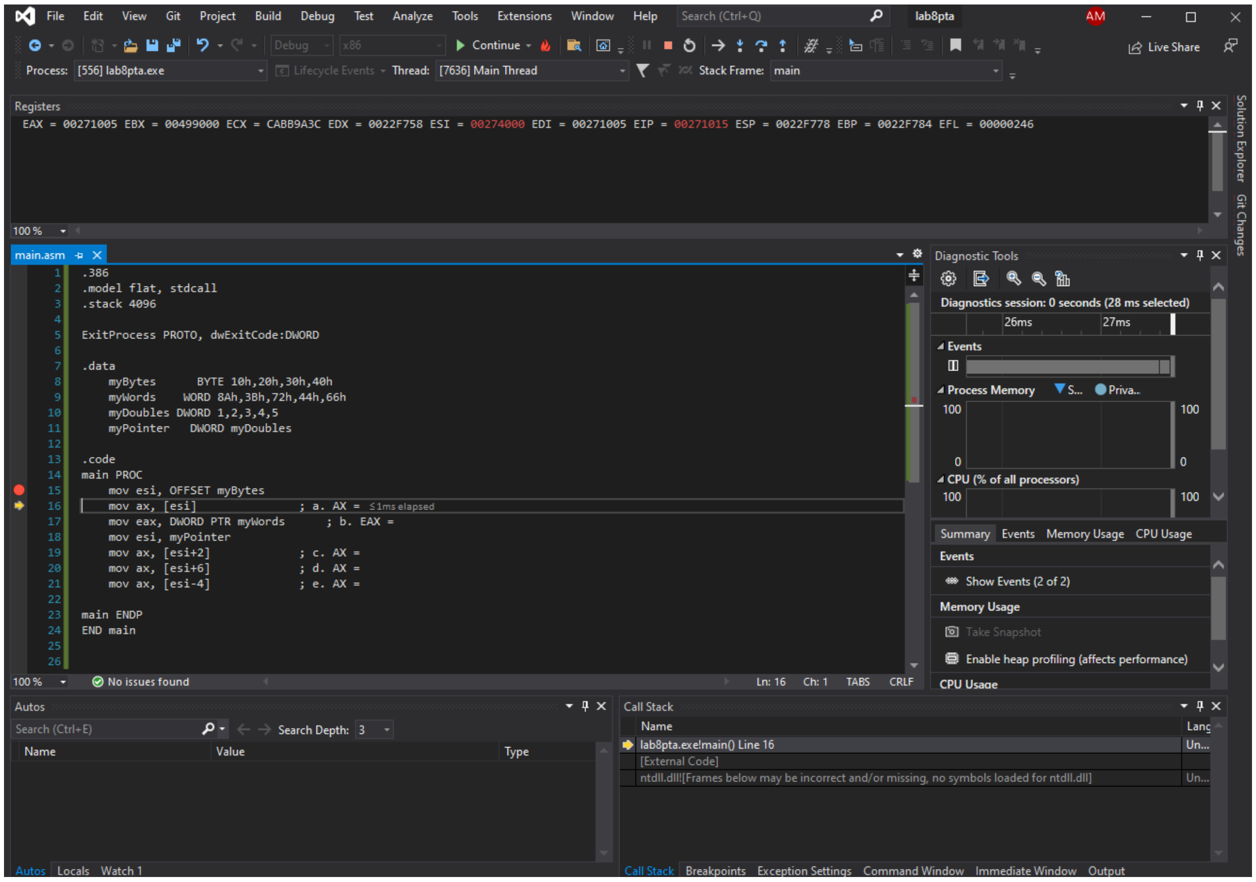
Lab 8a

Line:15

Instruction: mov esi, OFFSET myBytes

Register value: ESI = 00274000

Screenshot:



```
EAX = 00271005 EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 00274000 EDI = 00271005 EIP = 00271015 ESP = 0022F778 EBP = 0022F784 EFL = 00000246
```

```
 1  .386
 2  .model flat, stdcall
 3  .stack 4096
 4
 5  ExitProcess PROTO, dwExitCode:DWORD
 6
 7  .data
 8      myBytes      BYTE 10h,20h,30h,40h
 9      myWords    WORD 8Ah,3Bh,72h,44h,66h
10      myDoubles DWORD 1,2,3,4,5
11      myPointer   DWORD myDoubles
12
13  .code
14  main PROC
15      mov esi, OFFSET myBytes
16      mov ax, [esi]               ; a. AX =  ≤1ms elapsed
17      mov eax, DWORD PTR myWords      ; b. EAX =
18      mov esi, myPointer
19      mov ax, [esi+2]             ; c. AX =
20      mov ax, [esi+6]             ; d. AX =
21      mov ax, [esi-4]             ; e. AX =
22
23  main ENDP
24  END main
25
26
```

```
Name
lab8pta.exe!main() Line 16
[External Code]
ntdll.dll![Frames below may be incorrect and/or missing, no symbols loaded for ntdll.dll]
```

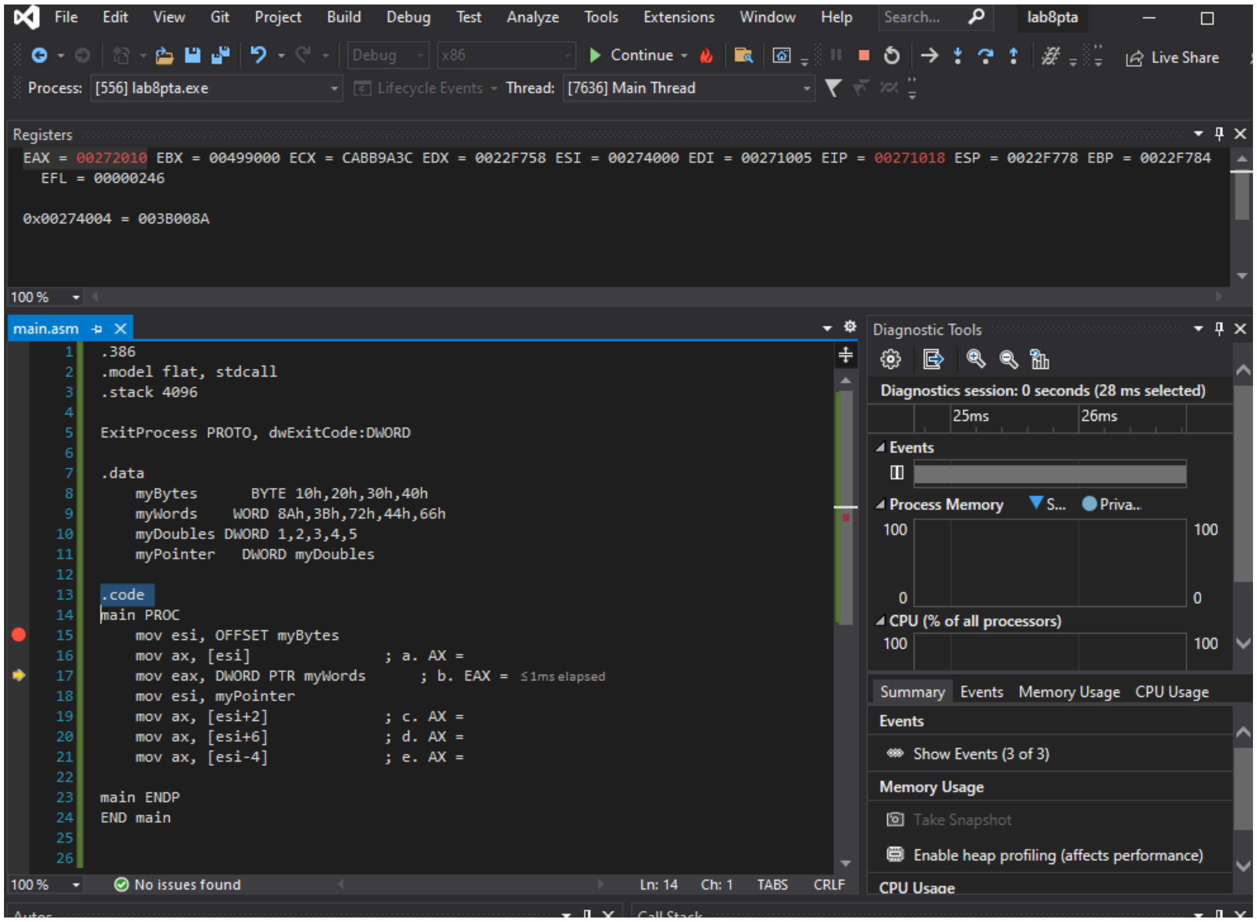
Explanation: In the instruction, the OFFSET is basically the address of the location added to a base address to get a specific address. Here in this instruction, we can see that the OFFSET gives the address of the myBYTES arrays and stores that address value in the ESI.

Line: 16

Instruction: mov ax, [esi]

Register value: EAX = 00272010

Screenshot:

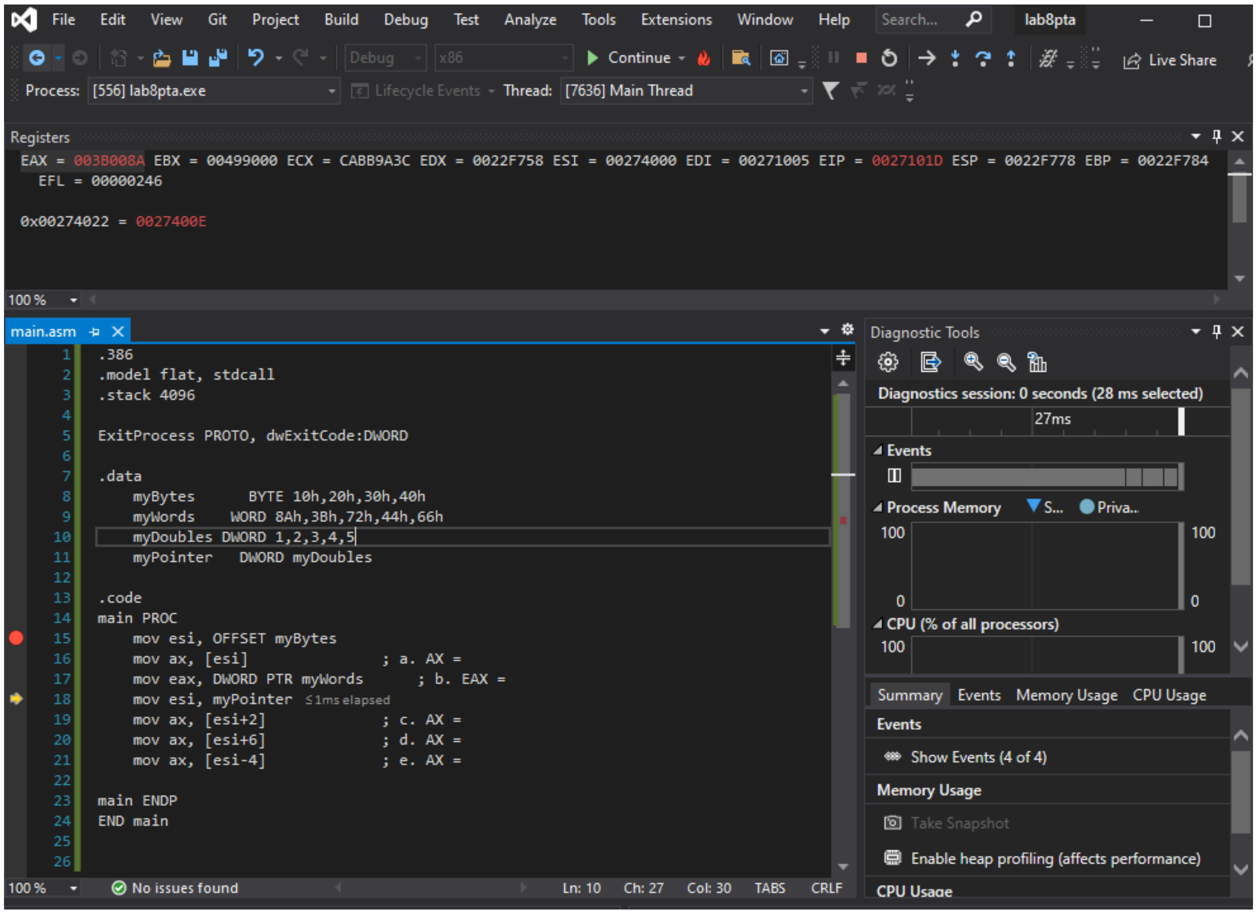


```
EAX = 00272010 EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 00274000 EDI = 00271005 EIP = 00271018 ESP = 0022F778 EBP = 0022F784
  EFL = 00000246

0x00274004 = 003B008A
```

```
.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 1,2,3,4,5
    myPointer   DWORD myDoubles

.code
main 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 =

main ENDP
END main
```

Explanation: This instruction is basically moving the first 16 bits of the myBYTES variable to the EAX register in the little-endian order.

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

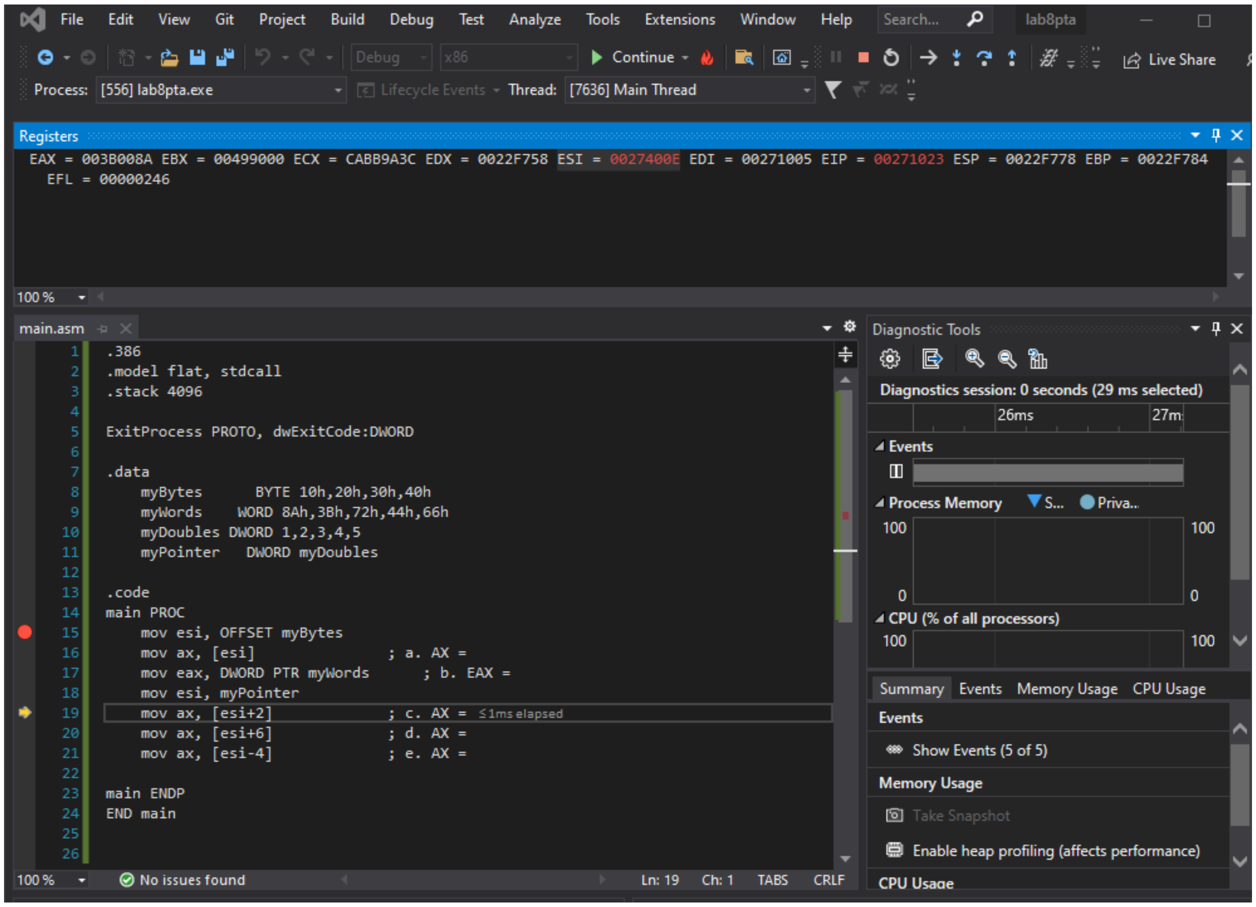


```
EAX = 003B008A EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 00274000 EDI = 00271005 EIP = 0027101D ESP = 0022F778 EBP = 0022F784
  EFL = 00000246

0x00274022 = 0027400E
```

```
 1  .386
 2  .model flat, stdcall
 3  .stack 4096
 4
 5  ExitProcess PROTO, dwExitCode:DWORD
 6
 7  .data
 8      myBytes      BYTE 10h,20h,30h,40h
 9      myWords    WORD 8Ah,3Bh,72h,44h,66h
10      myDoubles DWORD 1,2,3,4,5
11      myPointer   DWORD myDoubles
12
13  .code
14  main PROC
15      mov esi, OFFSET myBytes
16      mov ax, [esi]                ; a. AX =
17      mov eax, DWORD PTR myWords      ; b. EAX =
18      mov esi, myPointer  ≤1ms elapsed
19      mov ax, [esi+2]              ; c. AX =
20      mov ax, [esi+6]              ; d. AX =
21      mov ax, [esi-4]              ; e. AX =
22
23  main ENDP
24  END main
25
26
```

Explanation: In this instruction, we are basically changing the bits from 16 to 32 bits since it's a DWORD and this causes the size difference of the bytes and registers, so the PTR basically overrides the size and stores the value in the EAX register using the little-endian order.

Line: 18
Instruction: mov esi, myPointer
Register value: ESI = 0027400E
Screenshot:

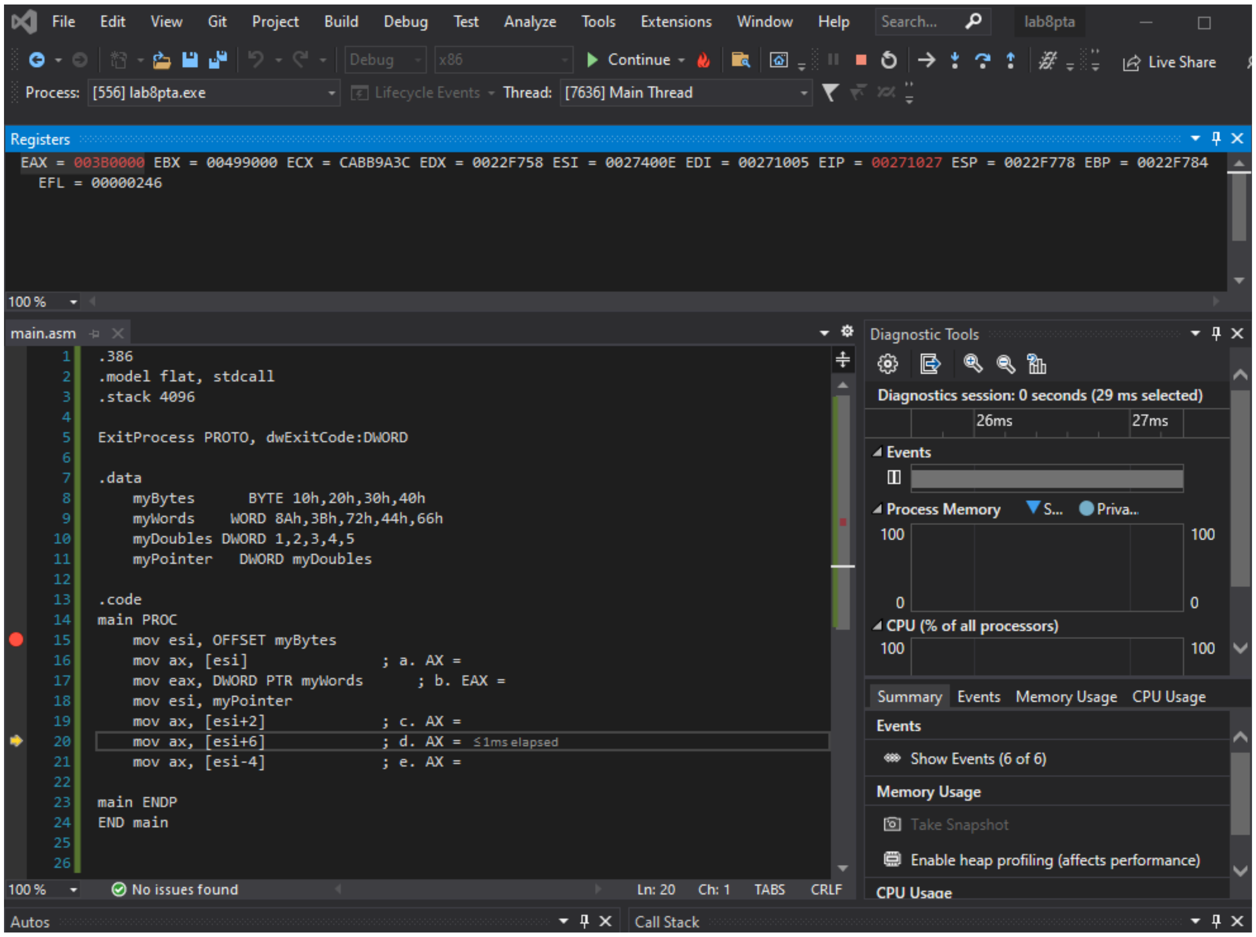


```
EAX = 003B008A EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 0027400E EDI = 00271005 EIP = 00271023 ESP = 0022F778 EBP = 0022F784
  EFL = 00000246
```

```
 1  .386
 2  .model flat, stdcall
 3  .stack 4096
 4
 5  ExitProcess PROTO, dwExitCode:DWORD
 6
 7  .data
 8      myBytes      BYTE 10h,20h,30h,40h
 9      myWords    WORD 8Ah,3Bh,72h,44h,66h
10      myDoubles DWORD 1,2,3,4,5
11      myPointer   DWORD myDoubles
12
13  .code
14  main PROC
15      mov esi, OFFSET myBytes
16      mov ax, [esi]              ; a. AX =
17      mov eax, DWORD PTR myWords      ; b. EAX =
18      mov esi, myPointer
19      mov ax, [esi+2]            ; c. AX =  ≤1ms elapsed
20      mov ax, [esi+6]            ; d. AX =
21      mov ax, [esi-4]            ; e. AX =
22
23  main ENDP
24  END main
25
26
```

Explanation: In this instruction, we are moving the value or the address of the myPointer variable and storing it into the ESI register

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

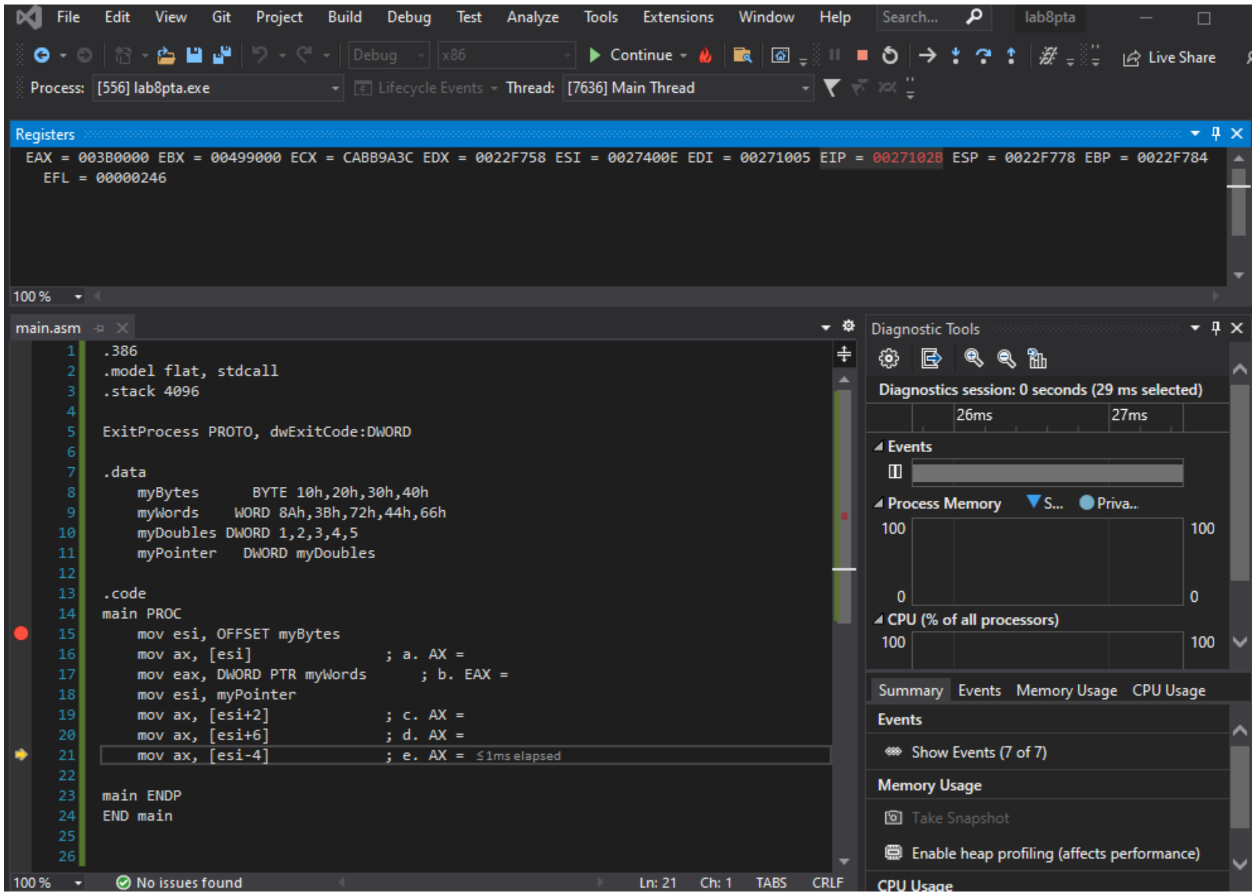


```
EAX = 003B0000 EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 0027400E EDI = 00271005 EIP = 00271027 ESP = 0022F778 EBP = 0022F784
  EFL = 00000246
```

```
.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 1,2,3,4,5
    myPointer   DWORD myDoubles

.code
main 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 =  ≤1ms elapsed
    mov ax, [esi-4]             ; e. AX =

main ENDP
END main
```

Explanation: In this instruction, we are adding +2 to the ESI register, which basically moves the value two places after the address of myPointer to the ax register in the little endian order.

Line: 20
Instruction: mov ax, [esi+6]
Register value: EIP = 0027102B

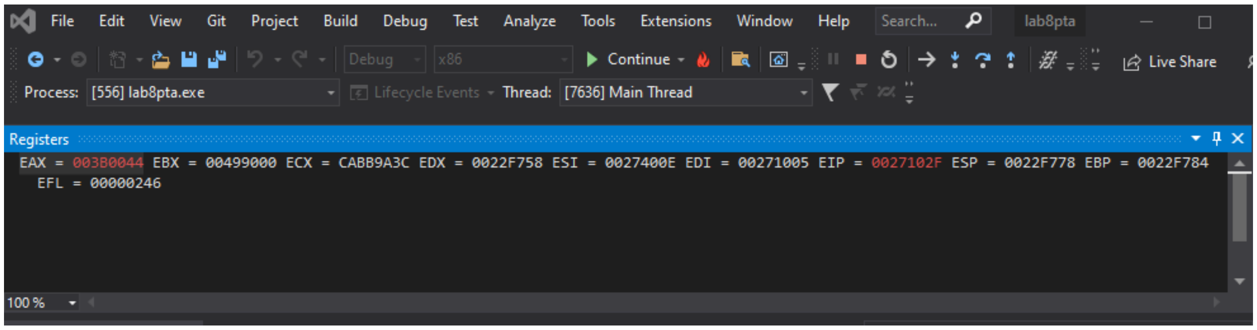


```
Registers
EAX = 003B0000 EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 0027400E EDI = 00271005 EIP = 0027102B ESP = 0022F778 EBP = 0022F784
  EFL = 00000246
```

```
main.asm
1   .386
2   .model flat, stdcall
3   .stack 4096
4
5   ExitProcess PROTO, dwExitCode:DWORD
6
7   .data
8       myBytes      BYTE 10h,20h,30h,40h
9       myWords    WORD 8Ah,3Bh,72h,44h,66h
10      myDoubles DWORD 1,2,3,4,5
11      myPointer   DWORD myDoubles
12
13  .code
14  main PROC
15      mov esi, OFFSET myBytes
16      mov ax, [esi]              ; a. AX =
17      mov eax, DWORD PTR myWords      ; b. EAX =
18      mov esi, myPointer
19      mov ax, [esi+2]            ; c. AX =
20      mov ax, [esi+6]            ; d. AX =
21      mov ax, [esi-4]            ; e. AX =  ≤1ms elapsed
22
23  main ENDP
24  END main
25
26
```

Explanation: In this instruction we are adding +6 to the ESI register and moving it to ax register. So we are basically moving the value to the sixth or six places after the address of myPointer to the ax register.

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

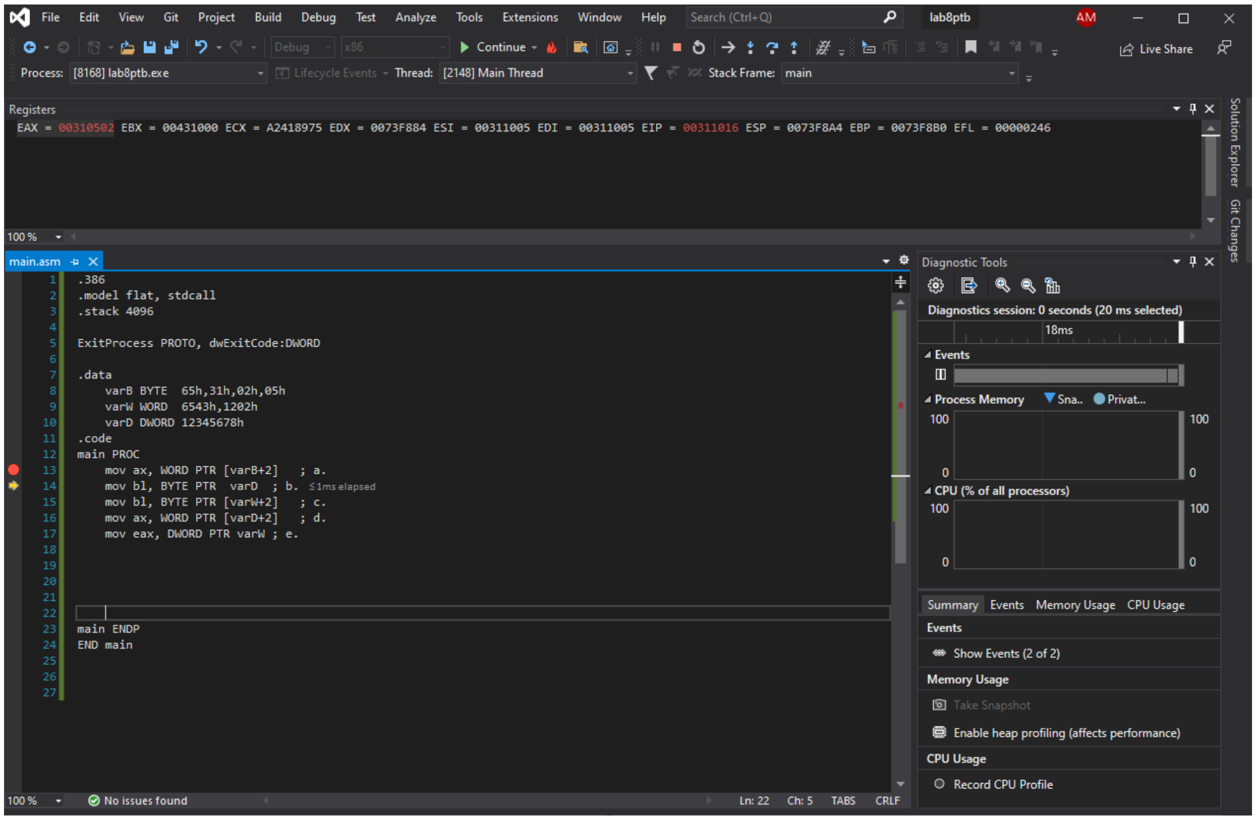


```
Registers
EAX = 003B0044 EBX = 00499000 ECX = CABB9A3C EDX = 0022F758 ESI = 0027400E EDI = 00271005 EIP = 0027102F ESP = 0022F778 EBP = 0022F784
  EFL = 00000246
```

Explanation: In this instruction, we are subtracting the -4 places in the esi register. So we are basically moving four places to the left of the address of myPointer and then that value is moved into the ax register.

Lab 8b

Line: 13
Instruction: mov ax, WORD PTR [varB+2]
Register value: EAX = 00310502
Screenshot:

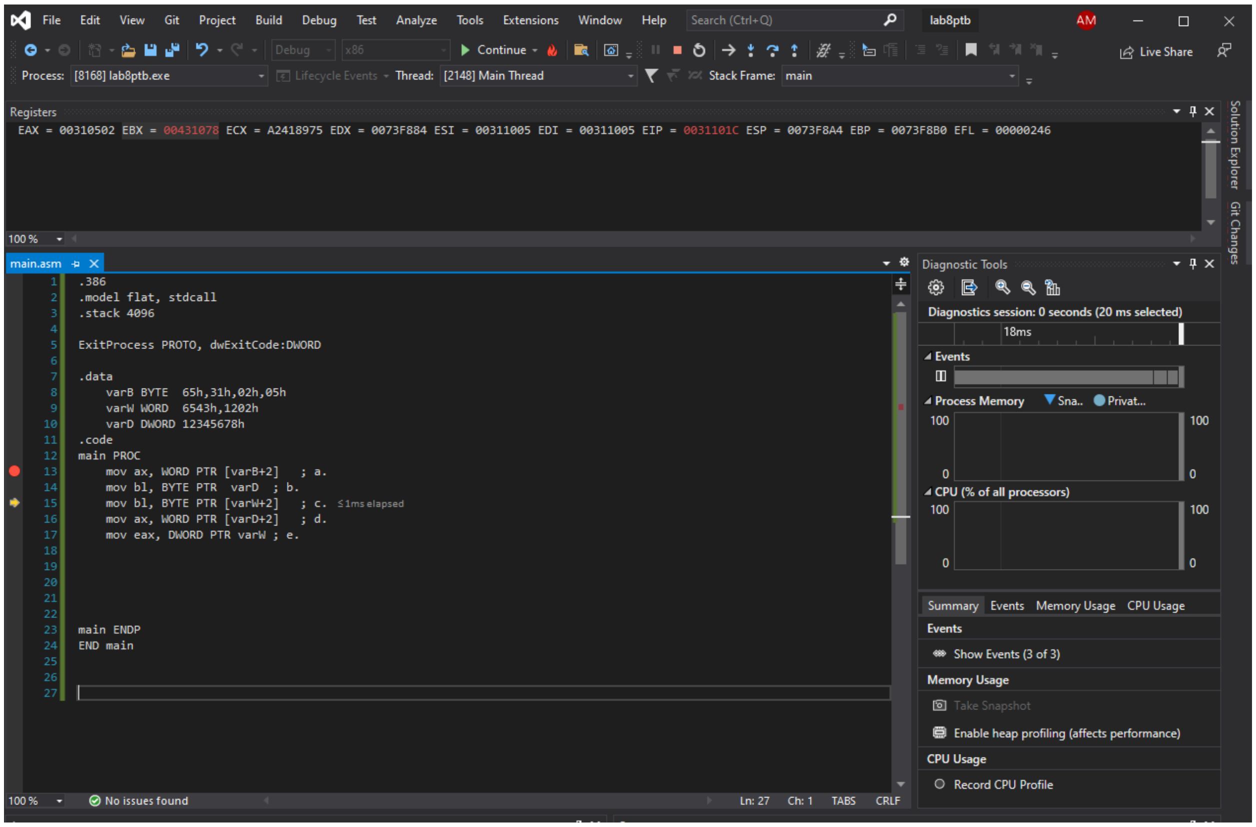


```
EAX = 00310502 EBX = 00431000 ECX = A2418975 EDX = 0073F884 ESI = 00311005 EDI = 00311005 EIP = 00311016 ESP = 0073F8A4 EBP = 0073F8B0 EFL = 00000246
```

```
 1  .386
 2  .model flat, stdcall
 3  .stack 4096
 4
 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  .code
12  main PROC
13      mov ax, WORD PTR [varB+2]   ; a.
14      mov bl, BYTE PTR  varD  ; b.  ≤1ms elapsed
15      mov bl, BYTE PTR [varW+2]   ; c.
16      mov ax, WORD PTR [varD+2]   ; d.
17      mov eax, DWORD PTR varW ; e.
18
19
20
21
22
23  main ENDP
24  END main
25
26
27
```

Explanation: In this instruction, we are adding +2 to the varB variable which basically moves to the last two elements of that variable and then stores that value in the EAX register, in the little endian order.

Line: 14
Instruction: mov bl, BYTE PTR varD
Register value: EBX = 00431078
Screenshot:



```
EAX = 00310502 EBX = 00431078 ECX = A2418975 EDX = 0073F884 ESI = 00311005 EDI = 00311005 EIP = 0031101C ESP = 0073F8A4 EBP = 0073F8B0 EFL = 00000246
```

```
1   .386
2   .model flat, stdcall
3   .stack 4096
4
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  .code
12  main 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
19
20
21
22
23  main ENDP
24  END main
25
26
27
```

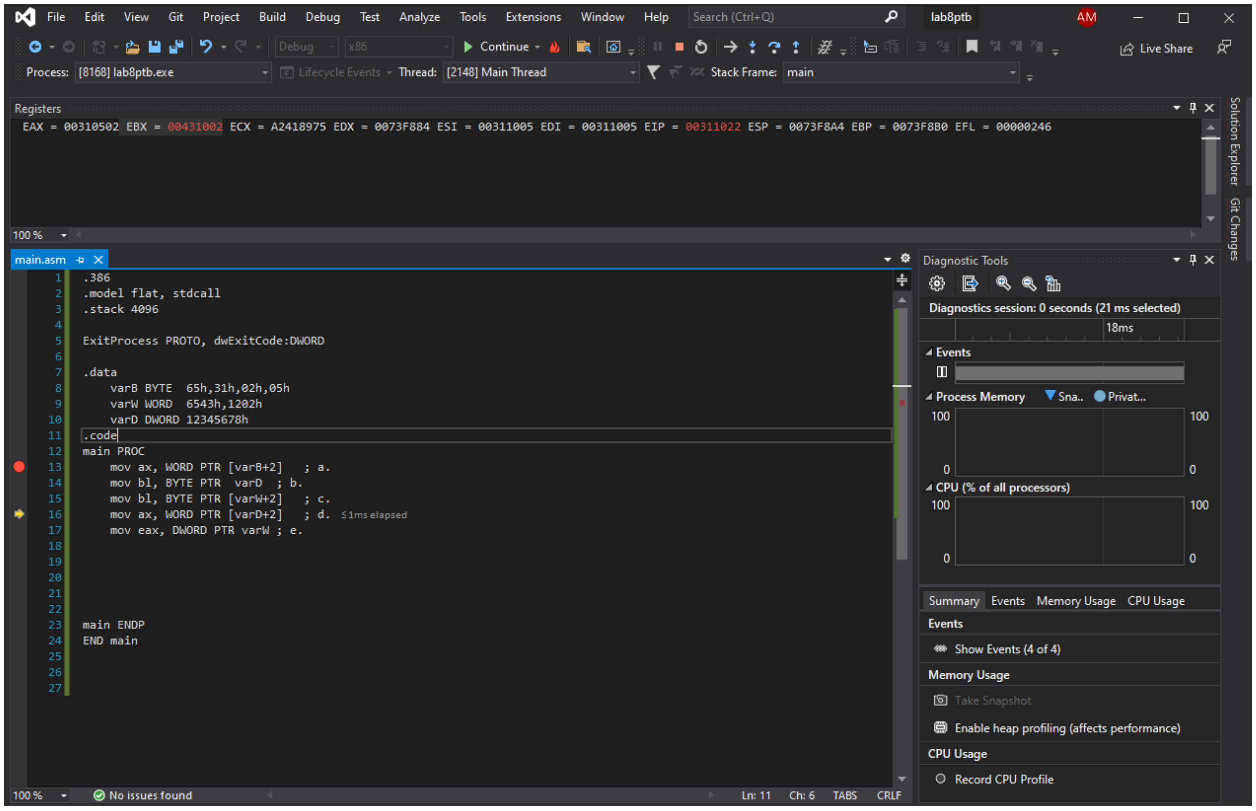
Explanation: PTR basically specifies the operand address of a word and it can be used to override the operands as well. In this instruction, we are basically moving the last two digits of the variable of varD and storing that last two-digit values in the EBX register.

Line: 15

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

Register value: EBX = 00431002

Screenshot:



```
EAX = 00310502 EBX = 00431002 ECX = A2418975 EDX = 0073F884 ESI = 00311005 EDI = 00311005 EIP = 00311022 ESP = 0073F8A4 EBP = 0073F8B0 EFL = 00000246
```

```
1   .386
2   .model flat, stdcall
3   .stack 4096
4
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  .code
12  main 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
19
20
21
22
23  main ENDP
24  END main
25
26
27
```

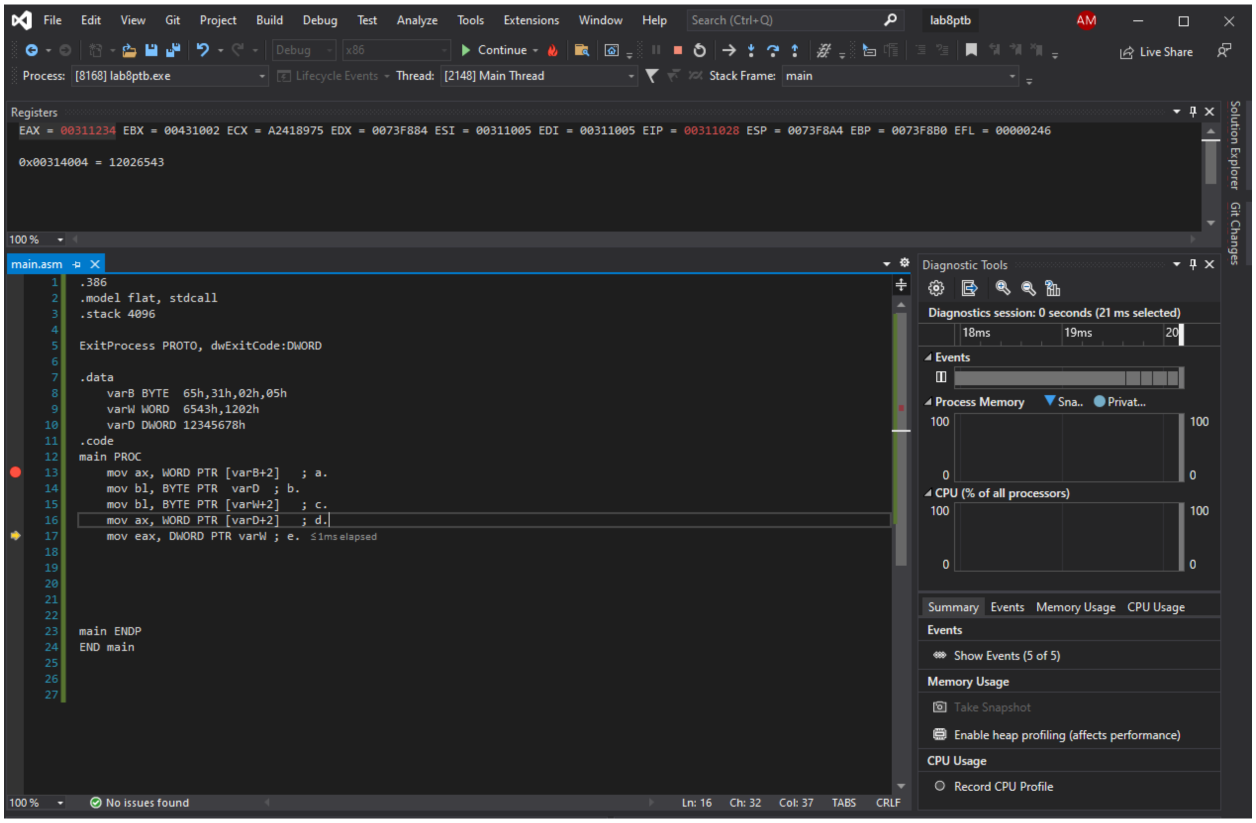
Explanation:In this instruction, we are adding +2 to the variable of varW, where it gors through the last two digits of the variable of varW. In here the variable varB is turned into the byte and then that value is moved into the bl register

Line: 16

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

Register value: EAX = 00311234

Screenshot:



```
Registers
EAX = 00311234 EBX = 00431002 ECX = A2418975 EDX = 0073F884 ESI = 00311005 EDI = 00311005 EIP = 00311028 ESP = 0073F8A4 EBP = 0073F8B0 EFL = 00000246

0x00314004 = 12026543
```

```
1  .386
2  .model flat, stdcall
3  .stack 4096
4
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 .code
12 main 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
19
20
21
22
23 main ENDP
24 END main
25
26
27
```

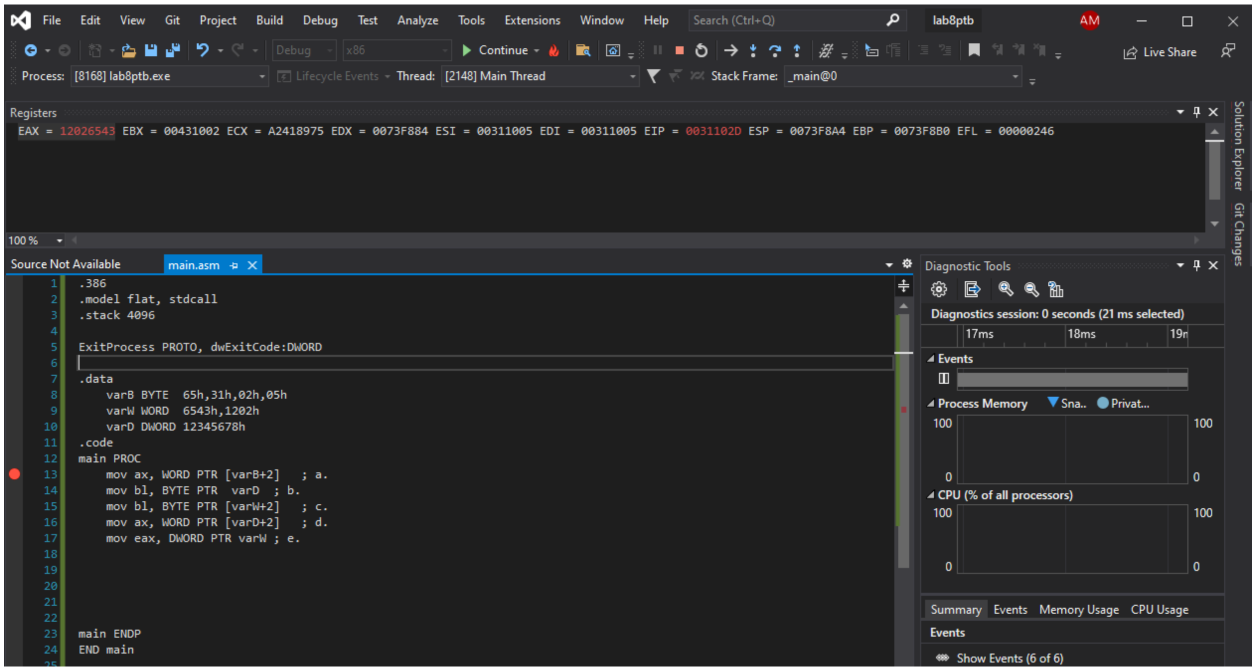
Explanation: here we are adding +2 to the varD variable which basically moves the first four digit of the varD variable into the eax register.

Line: 17

Instruction: mov eax, DWORD PTR varW

Register value: EAX = 12026543

Screenshot:



```
EAX = 12026543 EBX = 00431002 ECX = A2418975 EDX = 0073F884 ESI = 00311005 EDI = 00311005 EIP = 0031102D ESP = 0073F8A4 EBP = 0073F8B0 EFL = 00000246
```

```
.386
.model flat, stdcall
.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.data
    varB BYTE  65h,31h,02h,05h
    varW WORD  6543h,1202h
    varD DWORD 12345678h
.code
main PROC
    mov ax, WORD PTR [varB+2]   ; a.
    mov bl, BYTE PTR  varD  ; b.
    mov bl, BYTE PTR [varW+2]   ; c.
    mov ax, WORD PTR [varD+2]   ; d.
    mov eax, DWORD PTR varW ; e.





main ENDP
END main
```

Explanation: In this instruction, we can see that in the eax register it's storing the value of variable varD, so it first stores the value 1202, which is declared as 1202h in the varD variable and then it stores the other declared value 6543, which is stored as 6543h.So it is basically moving the values of varW into the eax register in the little endian order.

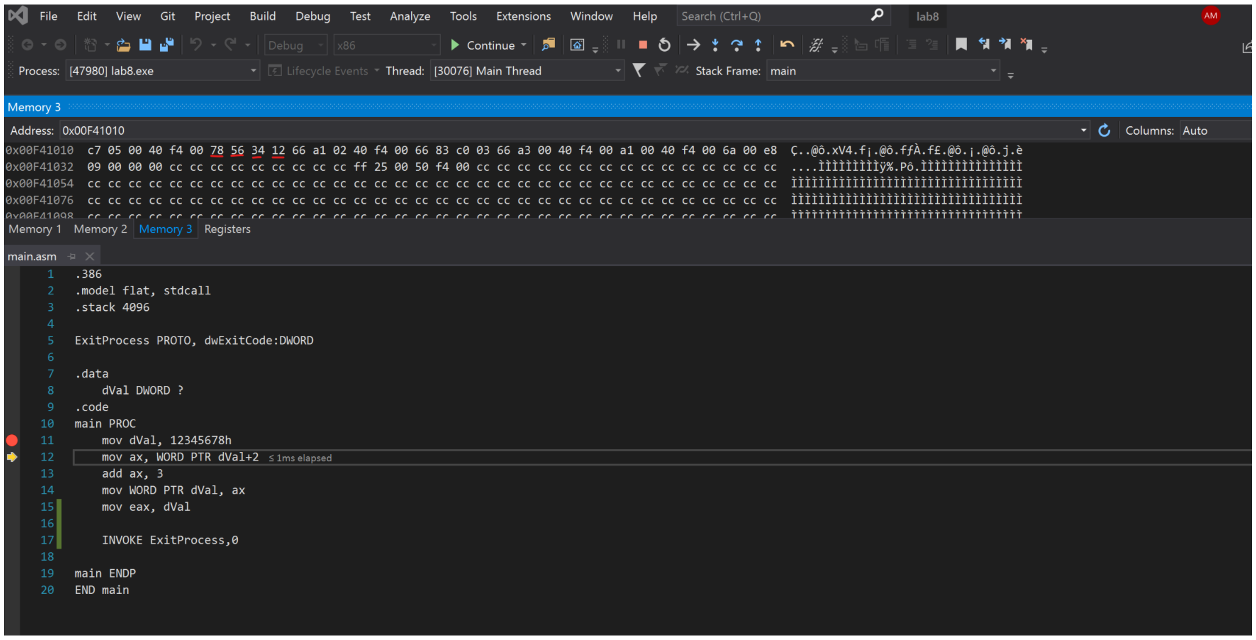
Lab 8c

Line: 11

Instruction: mov dVal, 12345678h

Register value: EIP: 00BD101A , **dVal= 12345678**

Screenshot:



```
Process: [47980] lab8.exe    Thread: [30076] Main Thread    Stack Frame: main

Memory 3
Address: 0x00F41010                                                                                  Columns: Auto
0x00F41010  c7 05 00 40 f4 00 78 56 34 12 66 a1 02 40 f4 00 66 83 c0 03 66 a3 00 40 f4 00 a1 00 40 f4 00 6a 00 e8  Ç..@ô.xV4.f¡.@ô.ƒÀ.f£.@ô.¡.@ô.j.è
0x00F41032  09 00 00 00 cc cc cc cc cc cc cc cc cc ff 25 00 50 f4 00 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ....ÌÌÌÌÌÌÌÌÌÿ%.Pô.ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00F41054  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00F41076  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00F41098  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
Memory 1  Memory 2  Memory 3  Registers
```

```
main.asm
 1  .386
 2  .model flat, stdcall
 3  .stack 4096
 4
 5  ExitProcess PROTO, dwExitCode:DWORD
 6
 7  .data
 8      dVal DWORD ?
 9  .code
10  main PROC
11      mov dVal, 12345678h
12      mov ax, WORD PTR dVal+2  ≤ 1ms elapsed
13      add ax, 3
14      mov WORD PTR dVal, ax
15      mov eax, dVal
16
17      INVOKE ExitProcess,0
18
19  main ENDP
20  END main
```

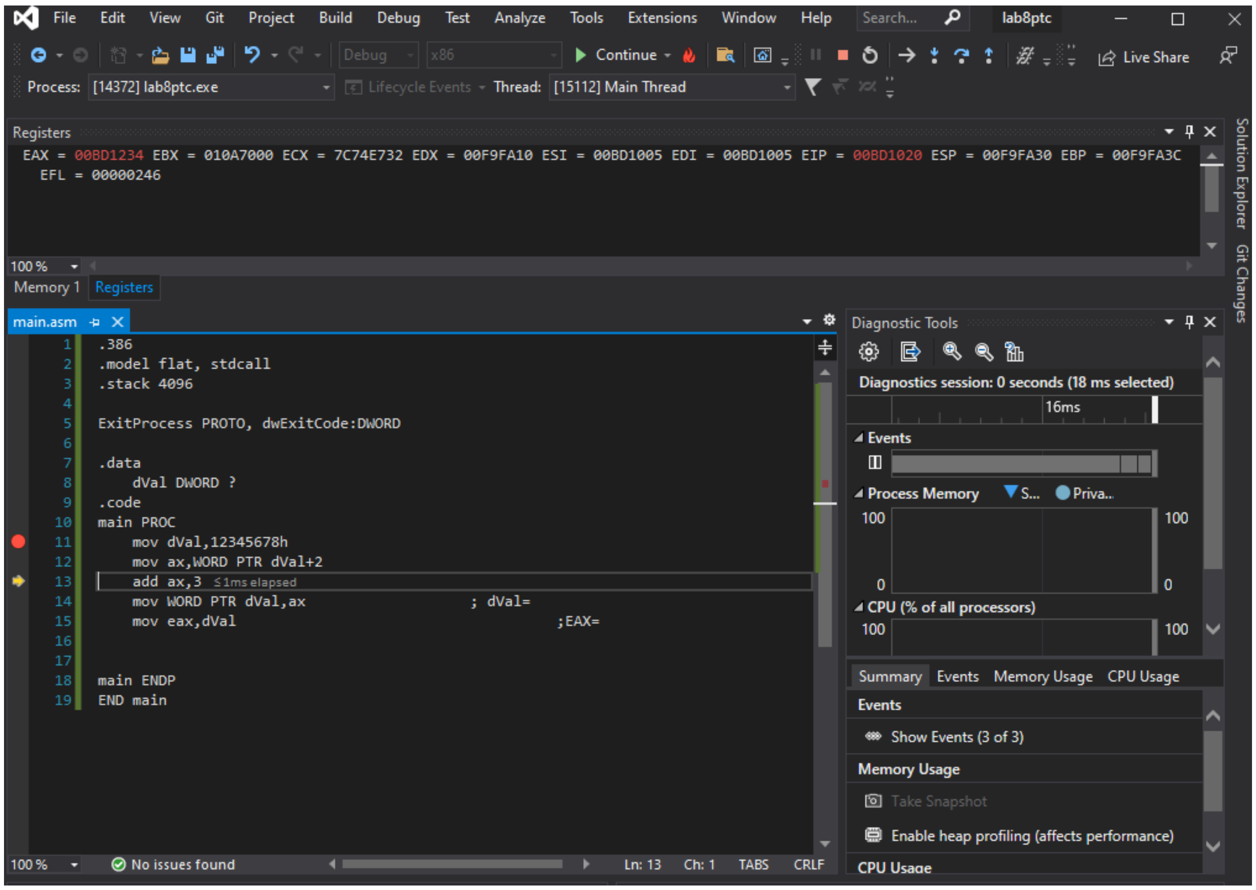
Explanation: here we are moving the value 1234568h, into the dVal variable

Line: 12

Instruction: mov ax, WORD PTR dVal+2

Register value: EAX: 00BD1234

Screenshot:

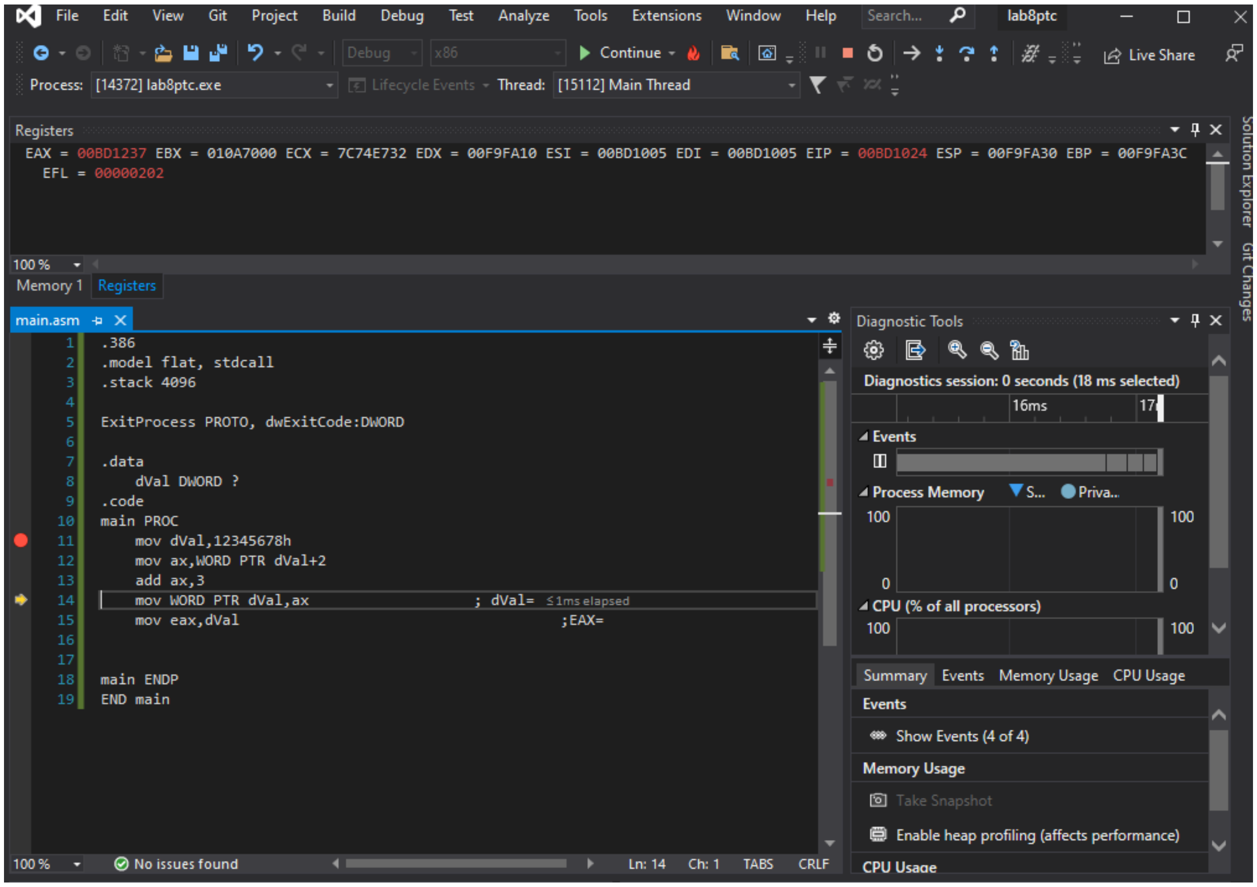

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
lab8ptc
Continue
Live Share
Process: [14372] lab8ptc.exe
Lifecycle Events
Thread: [15112] Main Thread
Registers
Memory 1
main.asm
Diagnostic Tools
Diagnostics session: 0 seconds (18 ms selected)
16ms
Events
Process Memory
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Show Events (3 of 3)
Take Snapshot
Enable heap profiling (affects performance)
CPU Usage
No issues found
Ln: 13 Ch: 1 TABS CRLF
Solution Explorer Git Changes

```
EAX = 00BD1234 EBX = 010A7000 ECX = 7C74E732 EDX = 00F9FA10 ESI = 00BD1005 EDI = 00BD1005 EIP = 00BD1020 ESP = 00F9FA30 EBP = 00F9FA3C
  EFL = 00000246
```

```
1   .386
2   .model flat, stdcall
3   .stack 4096
4
5   ExitProcess PROTO, dwExitCode:DWORD
6
7   .data
8       dVal DWORD ?
9   .code
10  main PROC
11      mov dVal,12345678h
12      mov ax,WORD PTR dVal+2
13      add ax,3  ≤1ms elapsed
14      mov WORD PTR dVal,ax                    ; dVal=
15      mov eax,dVal                                        ;EAX=
16
17
18  main ENDP
19  END main
```

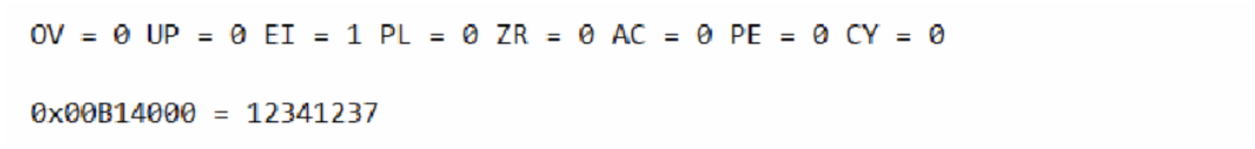
Explanation: Here we are moving the second part of the dVal and then storing that value in the ax register in the little endian order.

Line:13
Instruction: add ax, 3
Register value: EAX: 00BD1237
Screenshot:



```
EAX = 00BD1237 EBX = 010A7000 ECX = 7C74E732 EDX = 00F9FA10 ESI = 00BD1005 EDI = 00BD1005 EIP = 00BD1024 ESP = 00F9FA30 EBP = 00F9FA3C
  EFL = 00000202
```

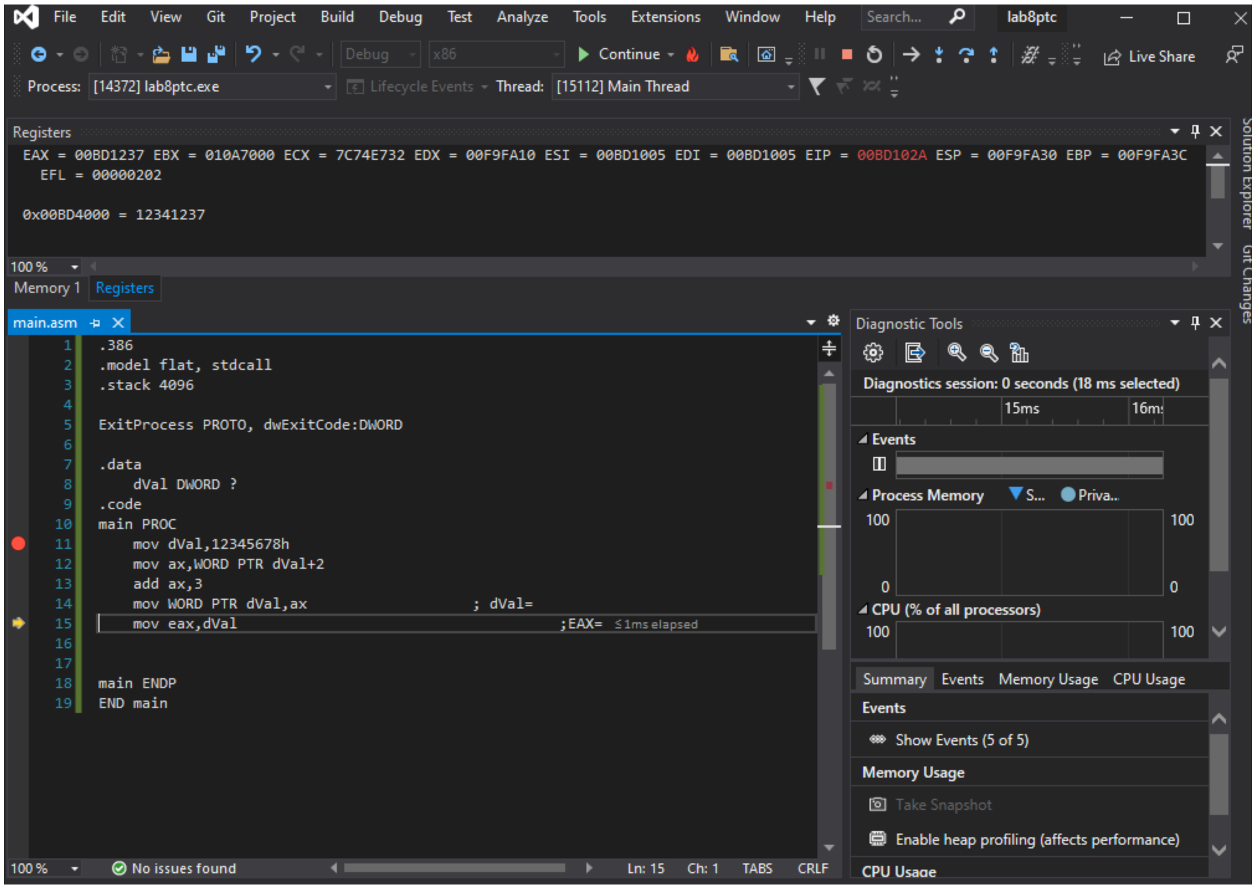
```
 1  .386
 2  .model flat, stdcall
 3  .stack 4096
 4
 5  ExitProcess PROTO, dwExitCode:DWORD
 6
 7  .data
 8      dVal DWORD ?
 9  .code
10  main PROC
11      mov dVal,12345678h
12      mov ax,WORD PTR dVal+2
13      add ax,3
14      mov WORD PTR dVal,ax                ; dVal=
15      mov eax,dVal                                    ;EAX=
16
17
18  main ENDP
19  END main
```



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

0x00B14000 = 12341237
```

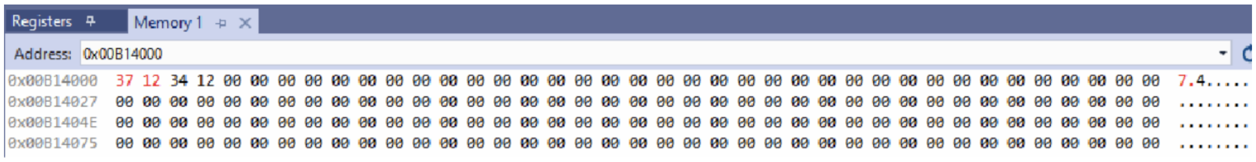
Explanation: In this instruction, we are adding 3 into the value that is already in the ax register.

Line: 14
Instruction: mov WORD PTR dVal, ax
Register value: EIP: 00BD102A, EAX: 00BD1237
Screenshot:


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search... lab8ptc
Debug x86 Continue Live Share
Process: [14372] lab8ptc.exe Lifecycle Events Thread: [15112] Main Thread
Diagnostic Tools
Diagnostics session: 0 seconds (18 ms selected)
15ms 16ms
Events
Process Memory S... Priva...
100 0 100 0
CPU (% of all processors)
100 100
Summary Events Memory Usage CPU Usage
Events
Show Events (5 of 5)
Memory Usage
Take Snapshot
Enable heap profiling (affects performance)
CPU Usage
100 % No issues found Ln: 15 Ch: 1 TABS CRLF
Memory 1 Registers
main.asm

```
Registers
 EAX = 00BD1237 EBX = 010A7000 ECX = 7C74E732 EDX = 00F9FA10 ESI = 00BD1005 EDI = 00BD1005 EIP = 00BD102A ESP = 00F9FA30 EBP = 00F9FA3C
   EFL = 00000202

 0x00BD4000 = 12341237
```

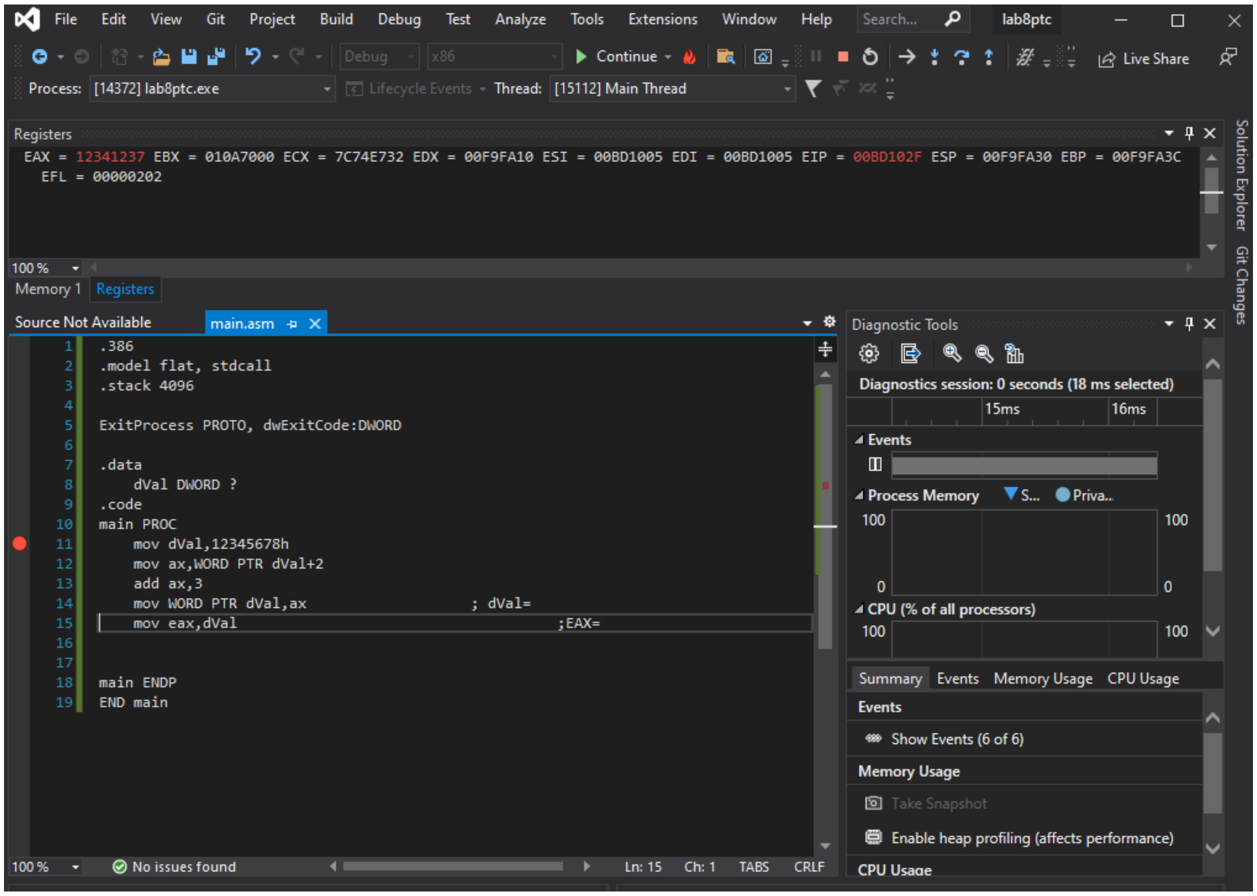
```
1   .386
2   .model flat, stdcall
3   .stack 4096
4
5   ExitProcess PROTO, dwExitCode:DWORD
6
7   .data
8       dVal DWORD ?
9   .code
10  main PROC
11      mov dVal,12345678h
12      mov ax,WORD PTR dVal+2
13      add ax,3
14      mov WORD PTR dVal,ax                   ; dVal=
15      mov eax,dVal                                     ;EAX=  ≤1ms elapsed
16
17
18  main ENDP
19  END main
```



```
Address: 0x00B14000
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: In this instruction we are taking the last four digits and storing it in the little endian order in the WORD variable.

Line: 15 mov eax, dVal
Instruction:
Register value: WAX: 12341237
Screenshot:



```
EAX = 12341237 EBX = 010A7000 ECX = 7C74E732 EDX = 00F9FA10 ESI = 00BD1005 EDI = 00BD1005 EIP = 00BD102F ESP = 00F9FA30 EBP = 00F9FA3C
  EFL = 00000202
```

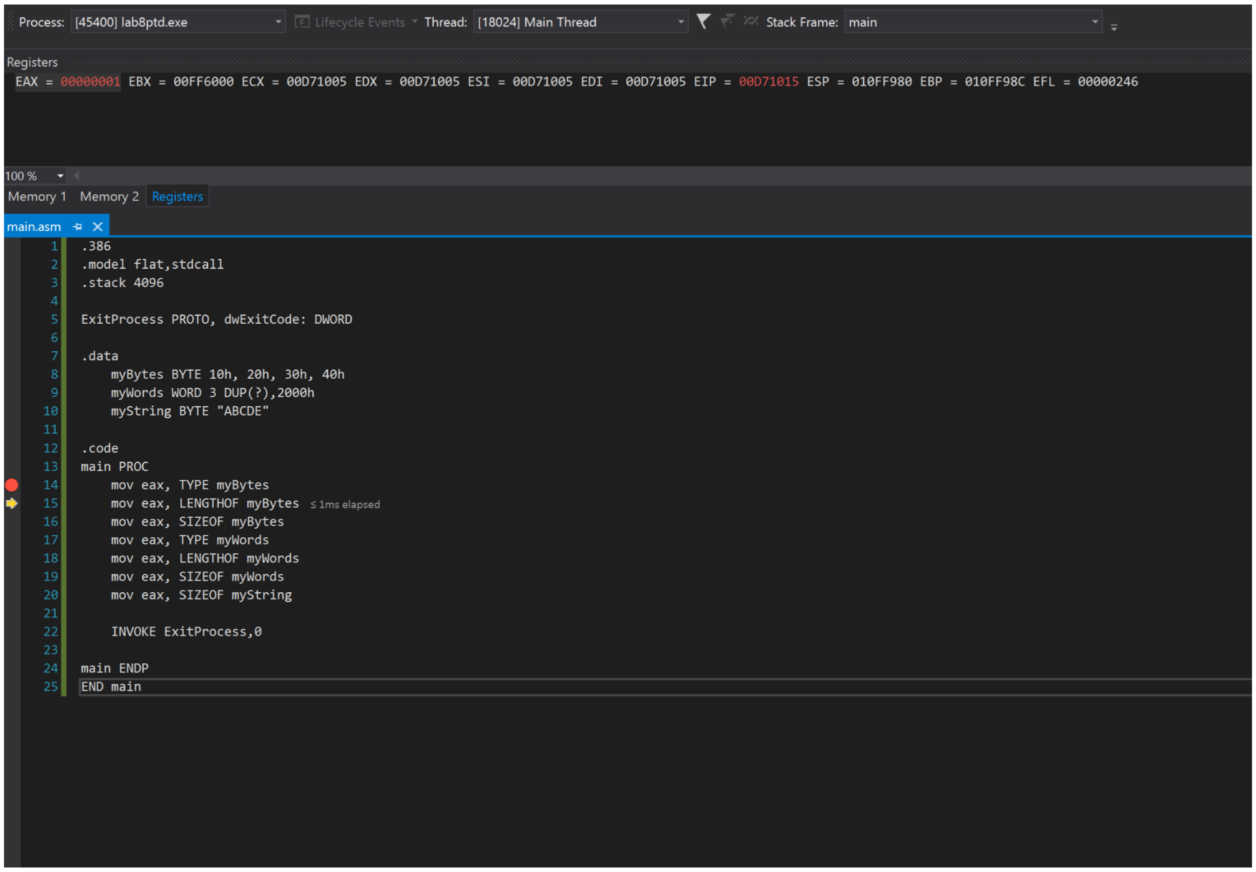
```
.386
.model flat, stdcall
.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.data
    dVal DWORD ?
.code
main PROC
    mov dVal,12345678h
    mov ax,WORD PTR dVal+2
    add ax,3
    mov WORD PTR dVal,ax                    ; dVal=
    mov eax,dVal                                      ;EAX=


main ENDP
END main
```

Explanation: in this instruction we are moving the value in the dVal into the eax register.

Part 8d

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

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



Process: [45400] lab8ptd.exe   Lifecycle Events   Thread: [18024] Main Thread   Stack Frame: main

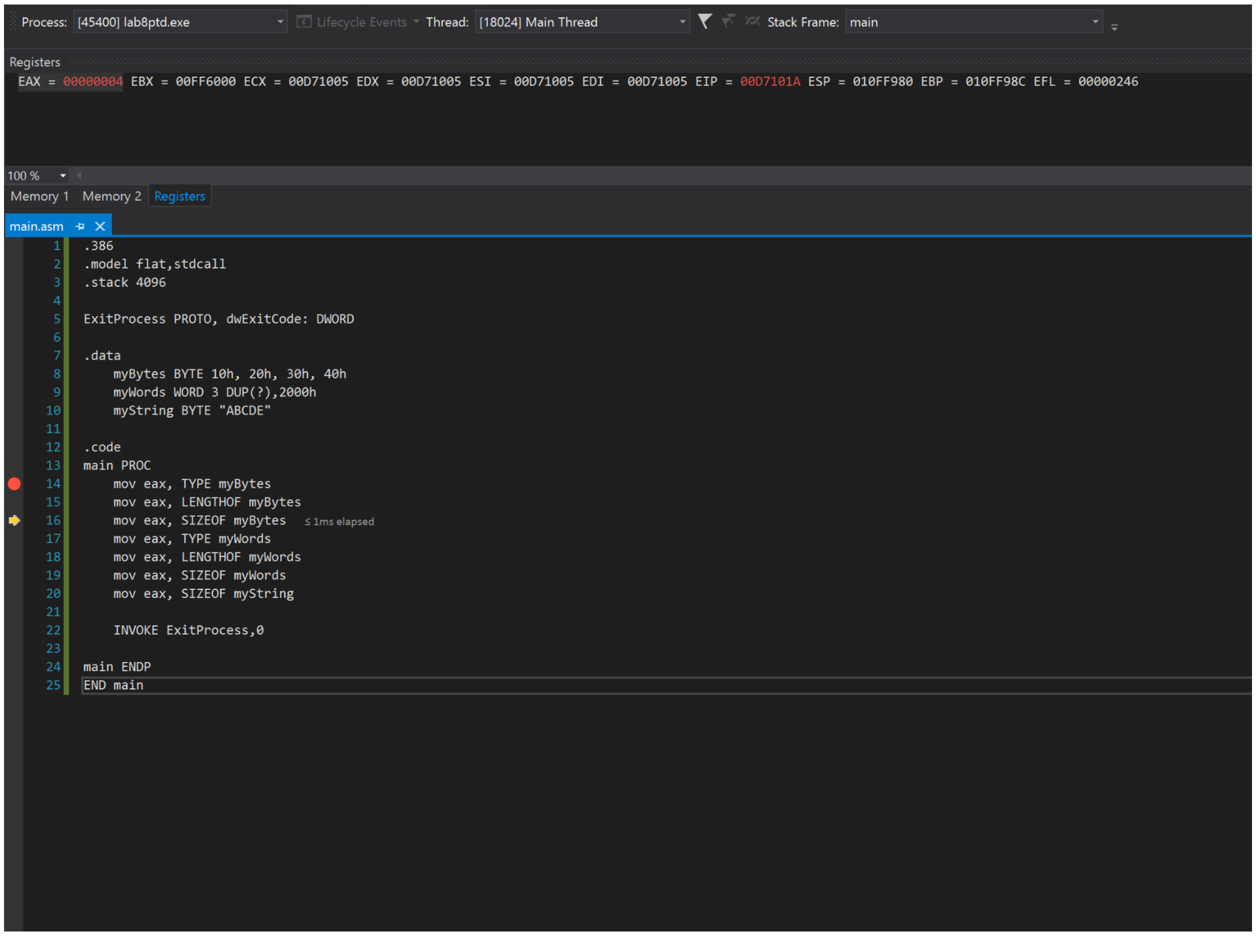
Registers
```
EAX = 00000001 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D71015 ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

```
 1  .386
 2  .model flat,stdcall
 3  .stack 4096
 4
 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
15      mov eax, LENGTHOF myBytes  ≤ 1ms elapsed
16      mov eax, SIZEOF myBytes
17      mov eax, TYPE myWords
18      mov eax, LENGTHOF myWords
19      mov eax, SIZEOF myWords
20      mov eax, SIZEOF myString
21
22      INVOKE ExitProcess,0
23
24  main ENDP
25  END main
```

Explanation: we are moving the value from TYPE myByte to the eax register. We known that the value of the BYTE is 1 bytes and the value of word is 2 bytes so in the eax register we will store 1 for myBytes

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

Screenshot:



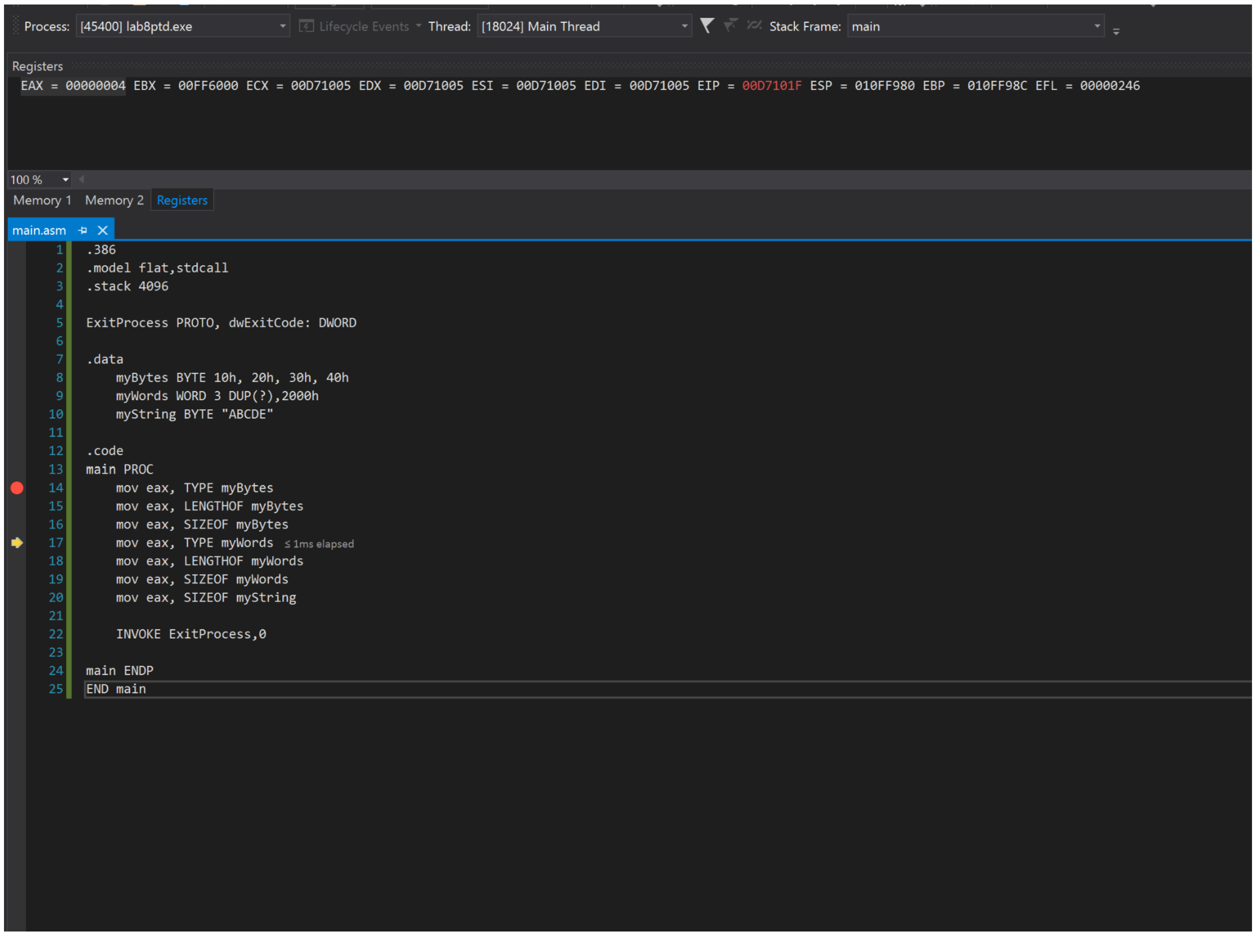
```
EAX = 00000004 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D7101A ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

```
.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
main PROC
    mov eax, TYPE myBytes
    mov eax, LENGTHOF myBytes
    mov eax, SIZEOF myBytes
    mov eax, TYPE myWords
    mov eax, LENGTHOF myWords
    mov eax, SIZEOF myWords
    mov eax, SIZEOF myString

    INVOKE ExitProcess,0

main ENDP
END main
```

Explanation: Here we are moving the value or the length of the myBytes into the eax register, which is 4, because the length of the myBytes array is 4.

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

Screenshot:



Process: [45400] lab8ptd.exe Lifecycle Events Thread: [18024] Main Thread Stack Frame: main

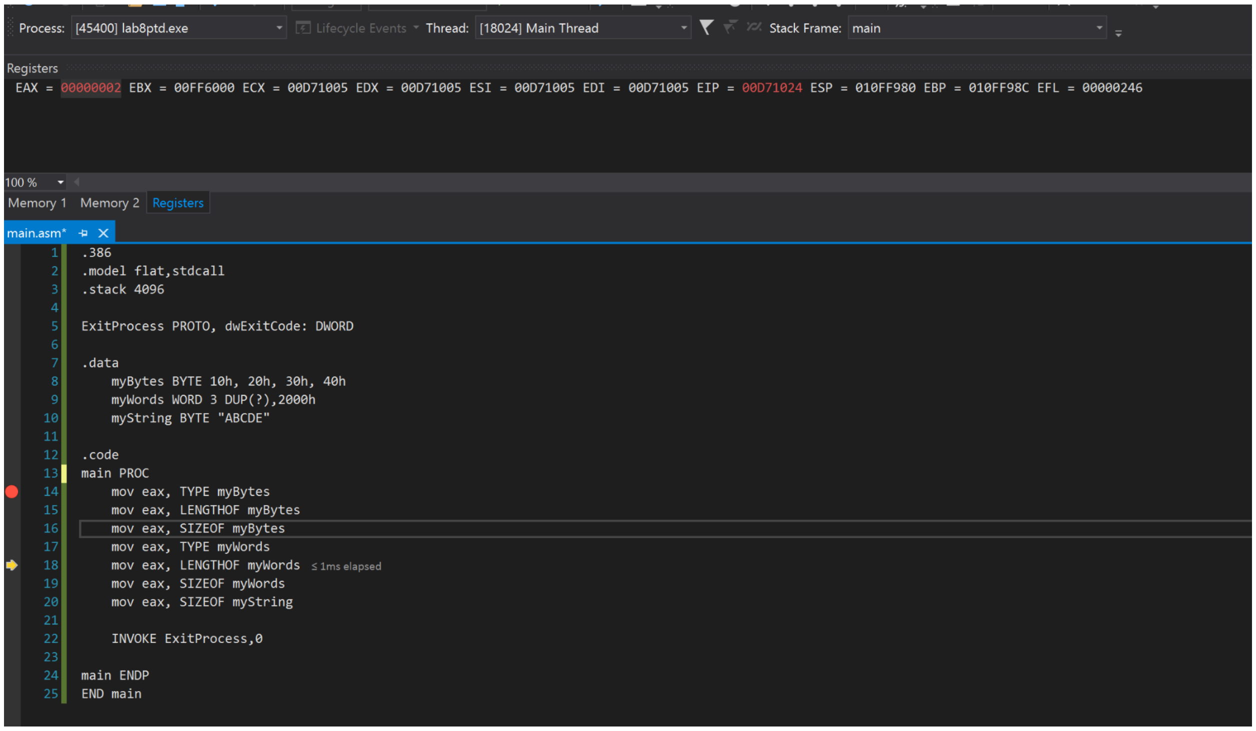
Registers

```
EAX = 00000004 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D7101F ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

```
.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
main PROC
    mov eax, TYPE myBytes
    mov eax, LENGTHOF myBytes
    mov eax, SIZEOF myBytes
    mov eax, TYPE myWords  ≤ 1ms elapsed
    mov eax, LENGTHOF myWords
    mov eax, SIZEOF myWords
    mov eax, SIZEOF myString

    INVOKE ExitProcess,0

main ENDP
END main
```

Explanation: now we are updating the eax register with the value or the size of the myBytes array

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



Process: [45400] lab8ptd.exe Lifecycle Events Thread: [18024] Main Thread Stack Frame: main

Registers
```
 EAX = 00000002 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D71024 ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

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
main PROC
    mov eax, TYPE myBytes
    mov eax, LENGTHOF myBytes
    mov eax, SIZEOF myBytes
    mov eax, TYPE myWords
    mov eax, LENGTHOF myWords  ≤ 1ms elapsed
    mov eax, SIZEOF myWords
    mov eax, SIZEOF myString

    INVOKE ExitProcess,0

main ENDP
END main
```

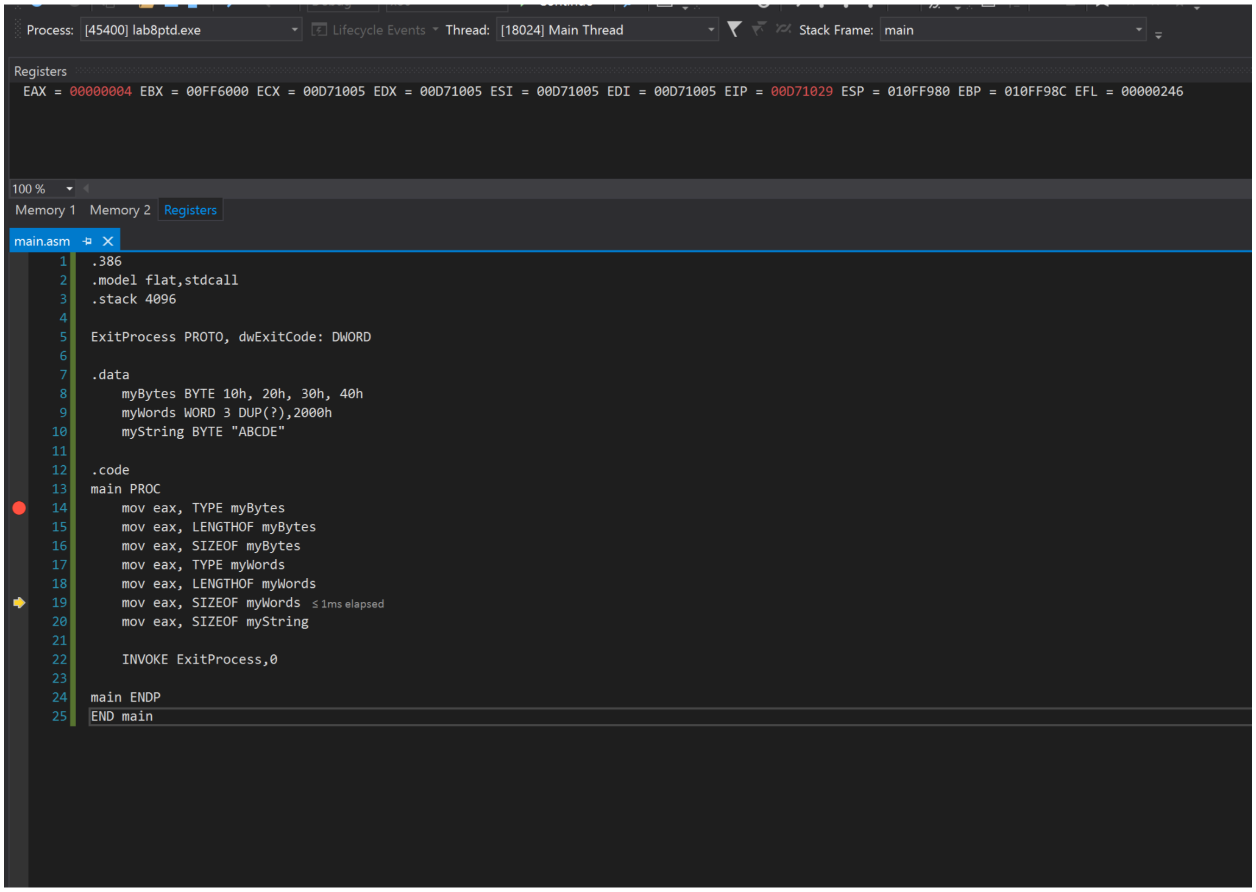
Explanation: here we are updating the eax register with the value of myWord with is 2, which is the size of each element

Line number: 18

Instruction: mov eax, LENGTHOF myWords

Register value: EAX = 00000004

Screenshot:



Process: [45400] lab8ptd.exe Lifecycle Events Thread: [18024] Main Thread Stack Frame: main

Registers

```
EAX = 00000004 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D71029 ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

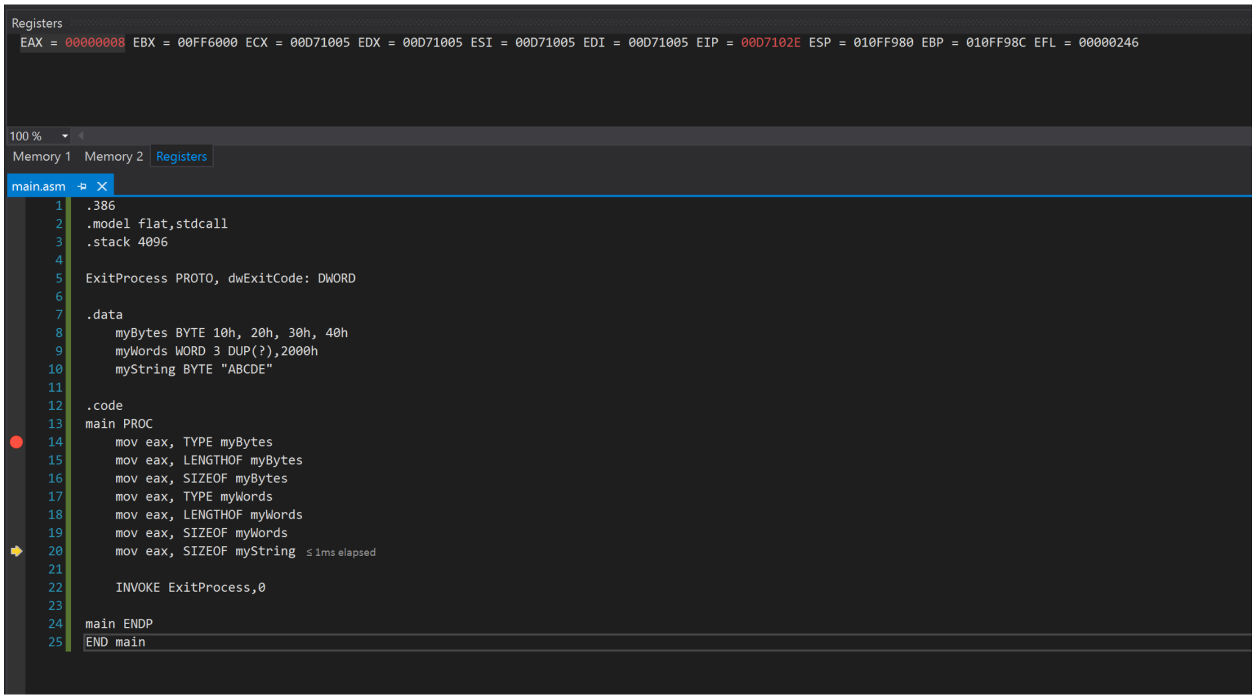
Memory 1 Memory 2 Registers

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
main PROC
    mov eax, TYPE myBytes
    mov eax, LENGTHOF myBytes
    mov eax, SIZEOF myBytes
    mov eax, TYPE myWords
    mov eax, LENGTHOF myWords
    mov eax, SIZEOF myWords  ≤ 1ms elapsed
    mov eax, SIZEOF myString

    INVOKE ExitProcess,0

main ENDP
END main
```

Explanation: here we are updating the eax register with the amount of or the length of the myWords which is 4.

Line number: 19
Instruction: mov eax, SIZEOF myWords
Register value:EAX = 00000008
Screenshot:



```
EAX = 00000008 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D7102E ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

```
.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
main PROC
    mov eax, TYPE myBytes
    mov eax, LENGTHOF myBytes
    mov eax, SIZEOF myBytes
    mov eax, TYPE myWords
    mov eax, LENGTHOF myWords
    mov eax, SIZEOF myWords
    mov eax, SIZEOF myString

    INVOKE ExitProcess,0

main ENDP
END main
```

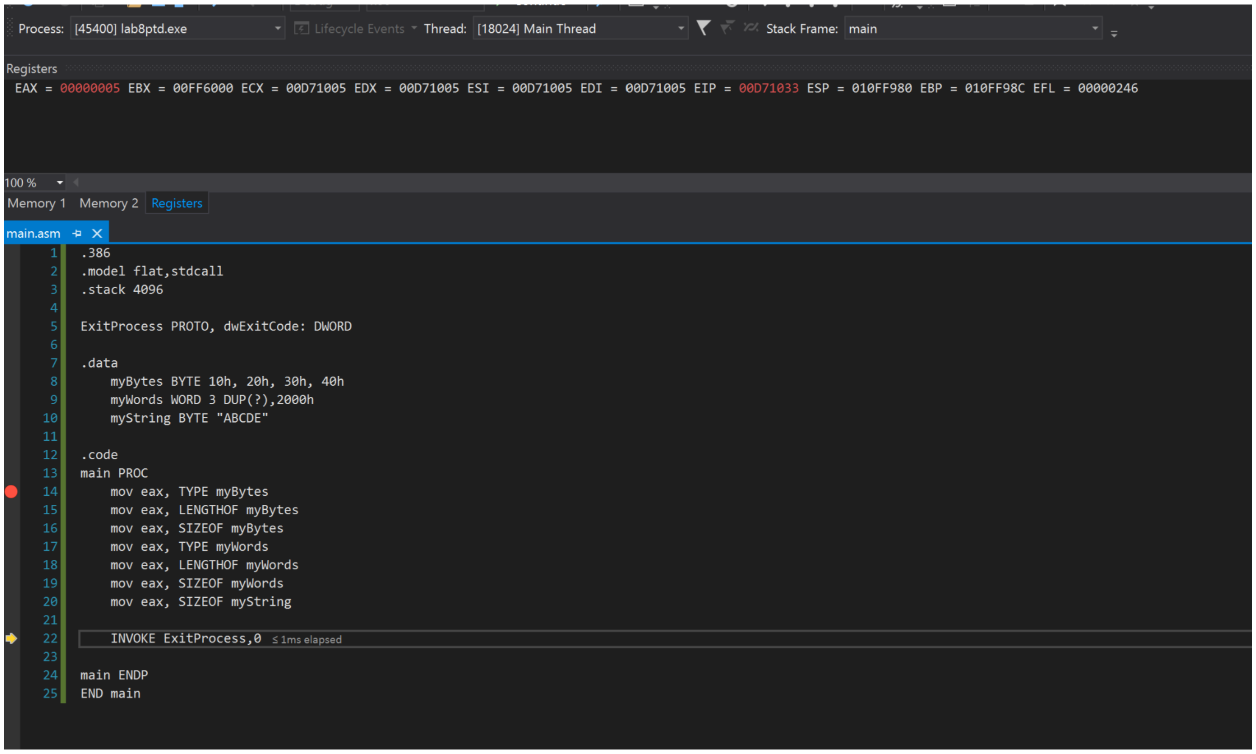
Explanation: the eax register is being updated by the size of myWords, which is 8.

Line number: 20

Instruction: mov eax, SIZEOF myString

Register value: EAX = 00000005

Screenshot:



```
EAX = 00000005 EBX = 00FF6000 ECX = 00D71005 EDX = 00D71005 ESI = 00D71005 EDI = 00D71005 EIP = 00D71033 ESP = 010FF980 EBP = 010FF98C EFL = 00000246
```

```
.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
main PROC
    mov eax, TYPE myBytes
    mov eax, LENGTHOF myBytes
    mov eax, SIZEOF myBytes
    mov eax, TYPE myWords
    mov eax, LENGTHOF myWords
    mov eax, SIZEOF myWords
    mov eax, SIZEOF myString

    INVOKE ExitProcess,0  ≤ 1ms elapsed

main ENDP
END main
```

Explanation: here we are updating the eax register with the size of the myString, which is 5

Complete for the all the lines inside main