CSC 3210

Computer Organization and Programming

Lab 9

Answer Sheet

Student Name: Aparna Mandapaka

Section:

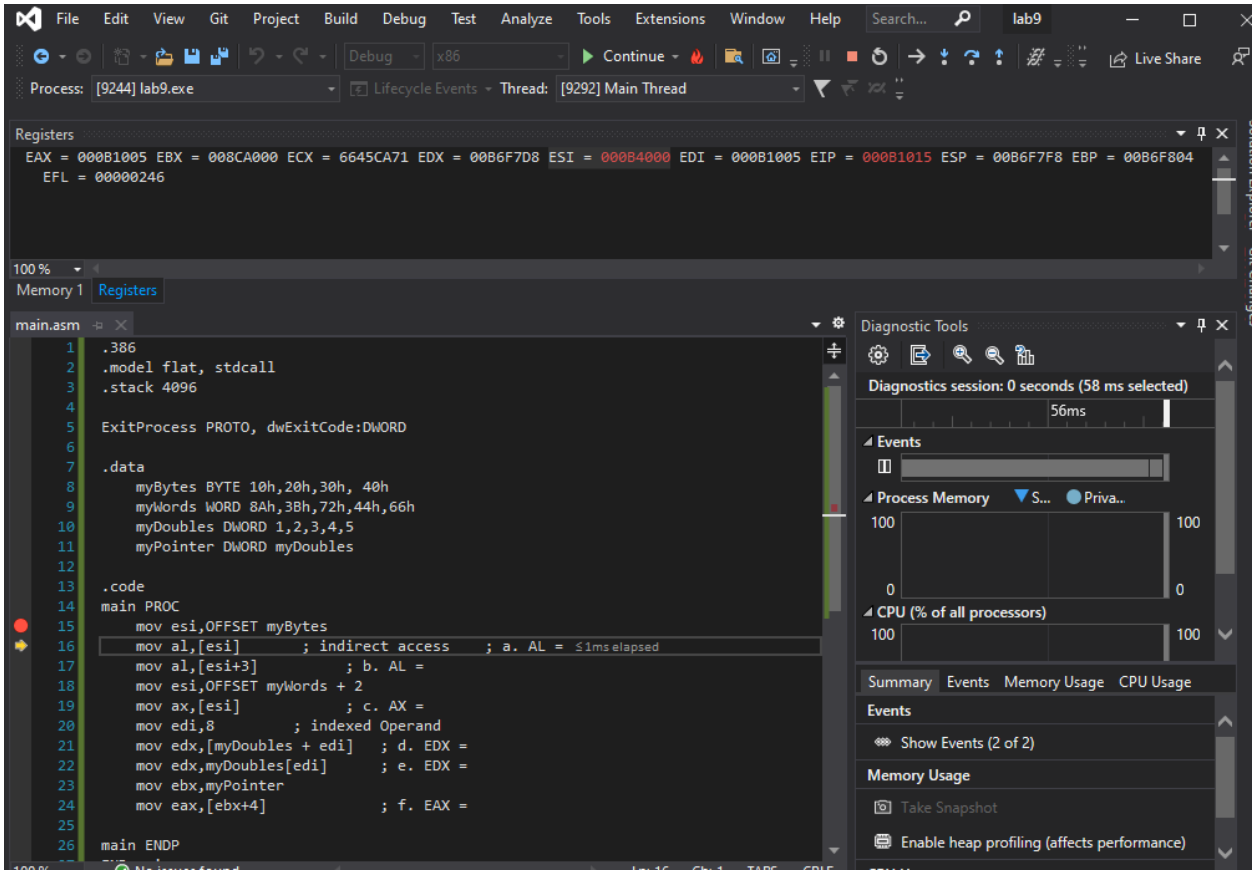
Lab 9 A

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

Line: 15

Instruction: mov esi,OFFSET myBytes

Register value: ESI = 000B4000



Screenshot:

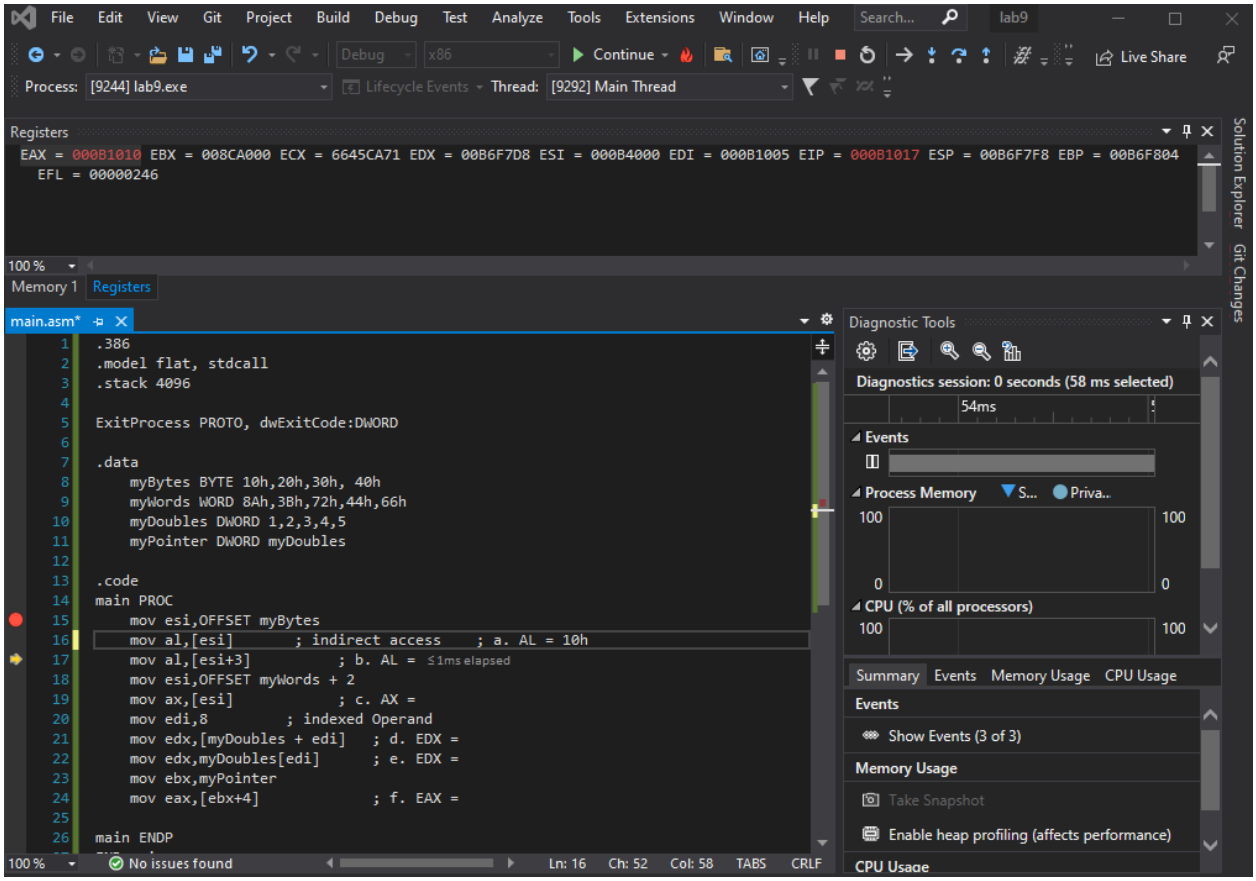
Explanation: The address of myBytes variable goes into the ESI register.

Line: 16

Instruction: mov al, [esi]

Register value: EAX = 000B1010

Screenshot:



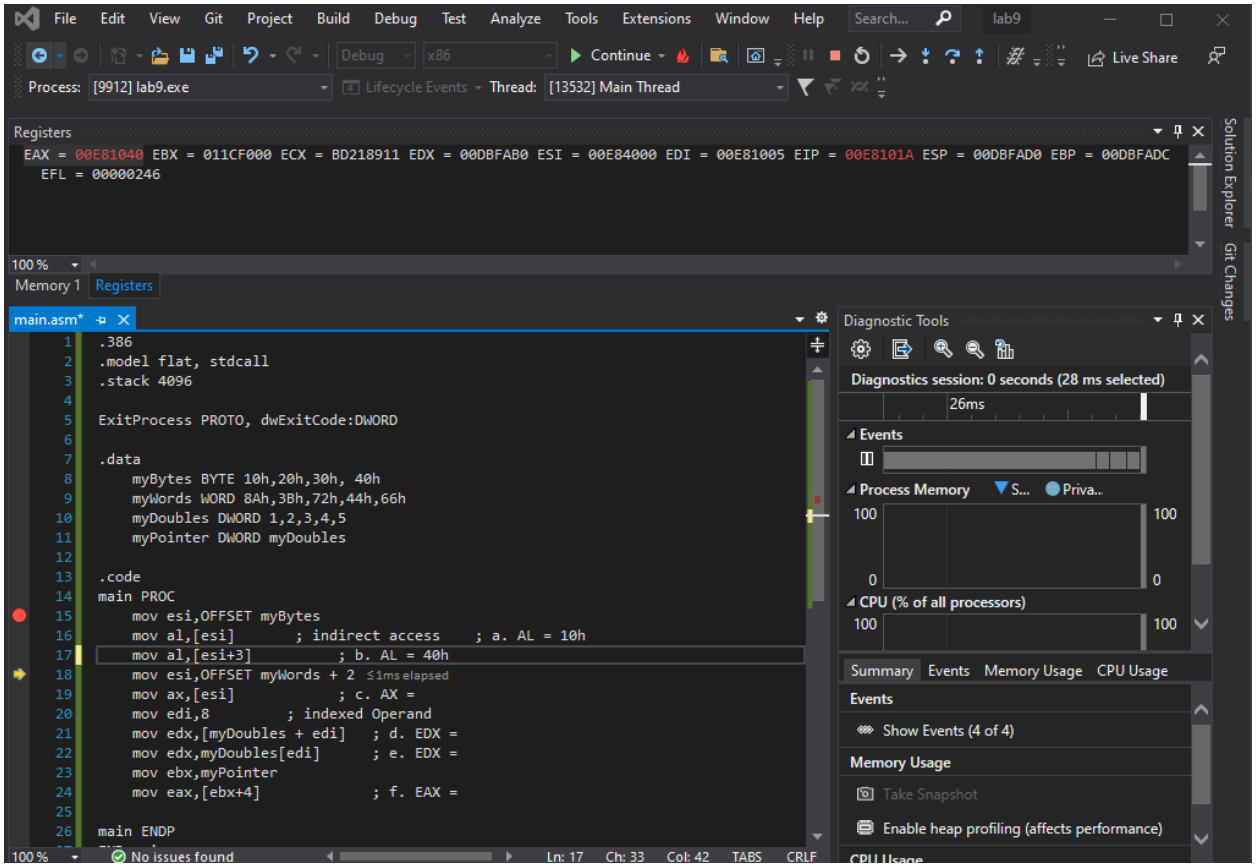
Explanation: after this instruction execution we are moving the first element of the myBytes variable into the al register, so in the eax register we can see that the last two digits is 10, and the last two places is 8 bit register and 10 is 8 bit too so it stores that value in the last two places

Line: 17

Instruction: mov al, [esi+3]

Register value: EAX = 00E81040

Screenshot:



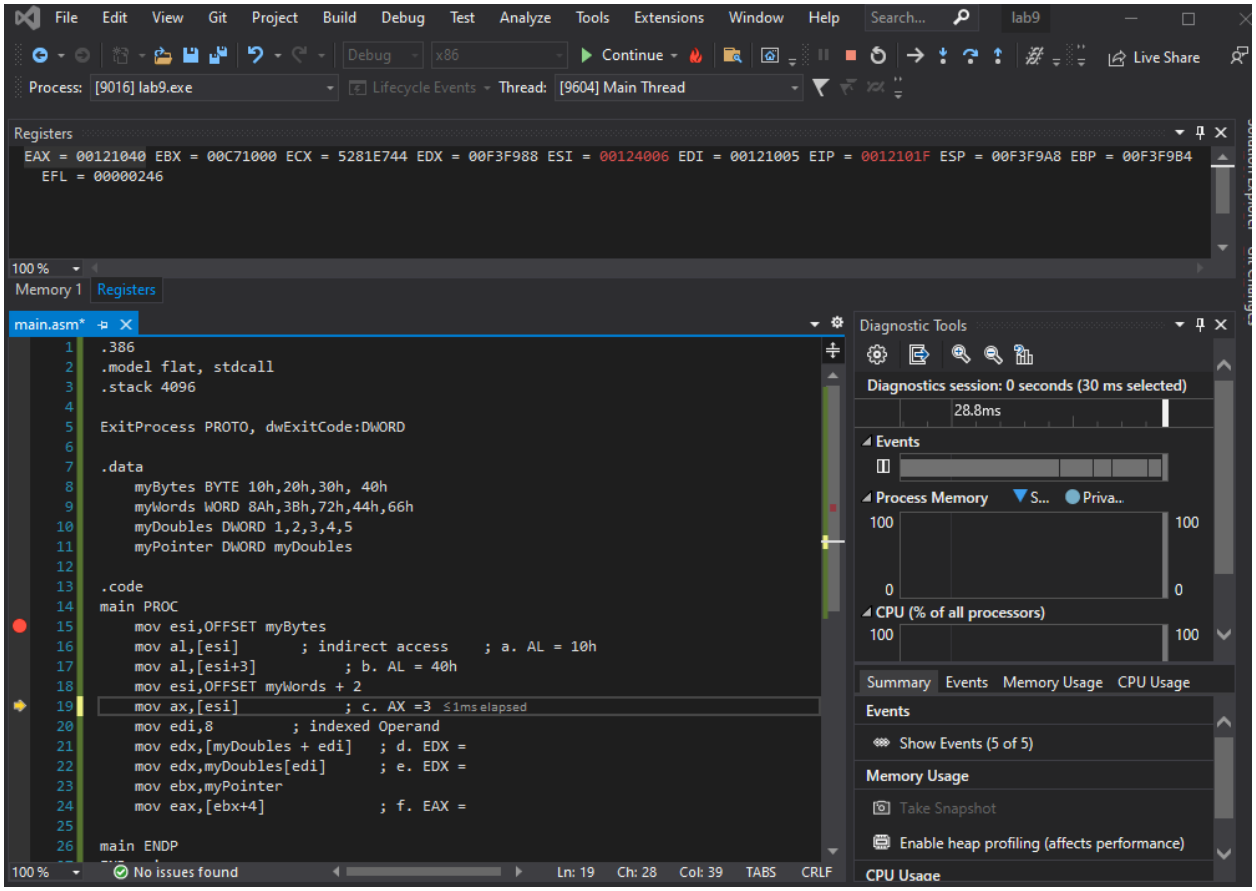
Explanation: now we are adding +3 to the esi register which is basically going to move 4 places to the right in the myBytes variable because the start is counted or started as 0,1,2,3, so we take the third value which is 40h and then store that or move that into the al register.

Line: 18

Instruction: mov esi, OFFSET myWords +2

Register value: ESI = 00124006, EAX = 00121040

Screenshot:



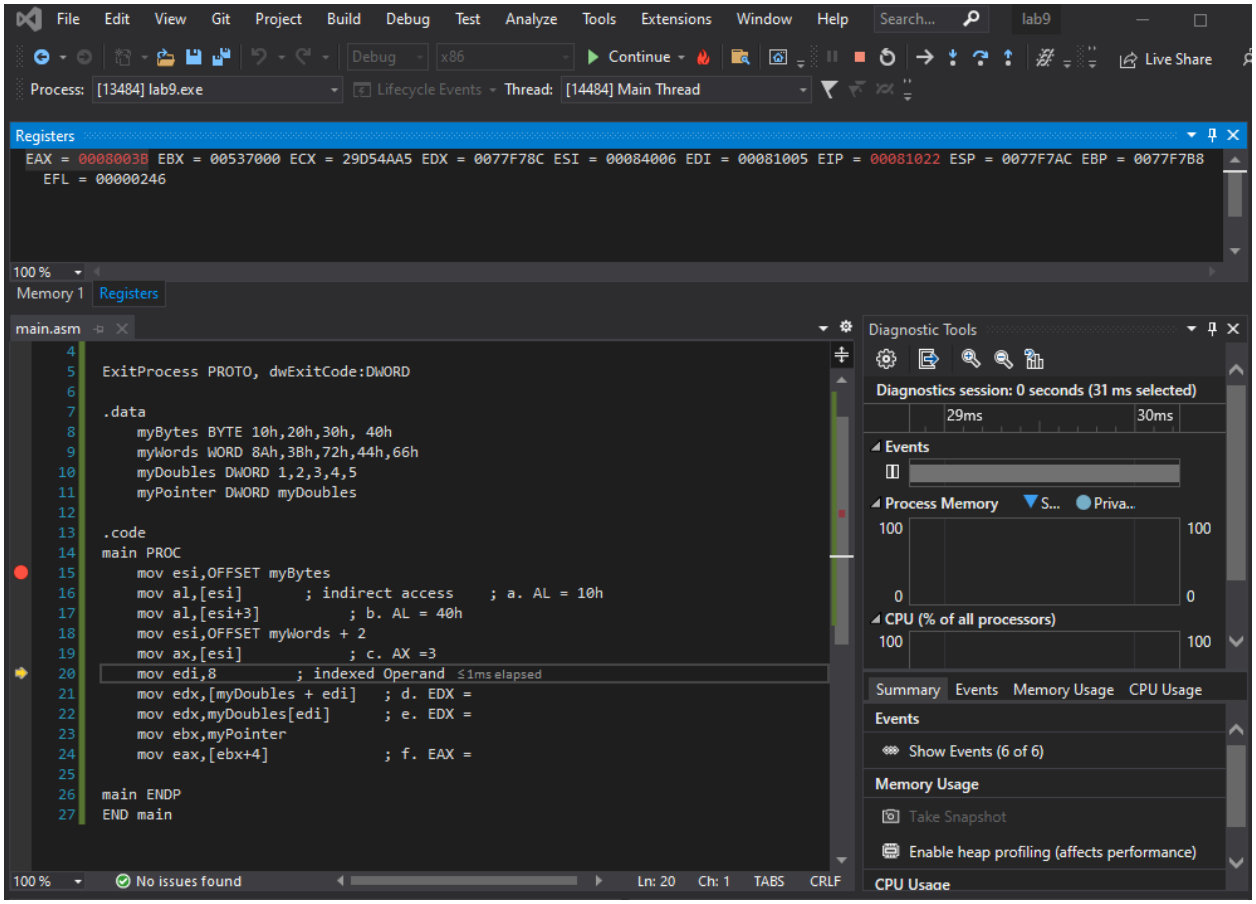
Explanation: Here we are moving the address of the second element of the myWords variable into the al register.

Line: 19

Instruction: mov ax, [esi]

Register value: EAX = 0008003B

Screenshot:



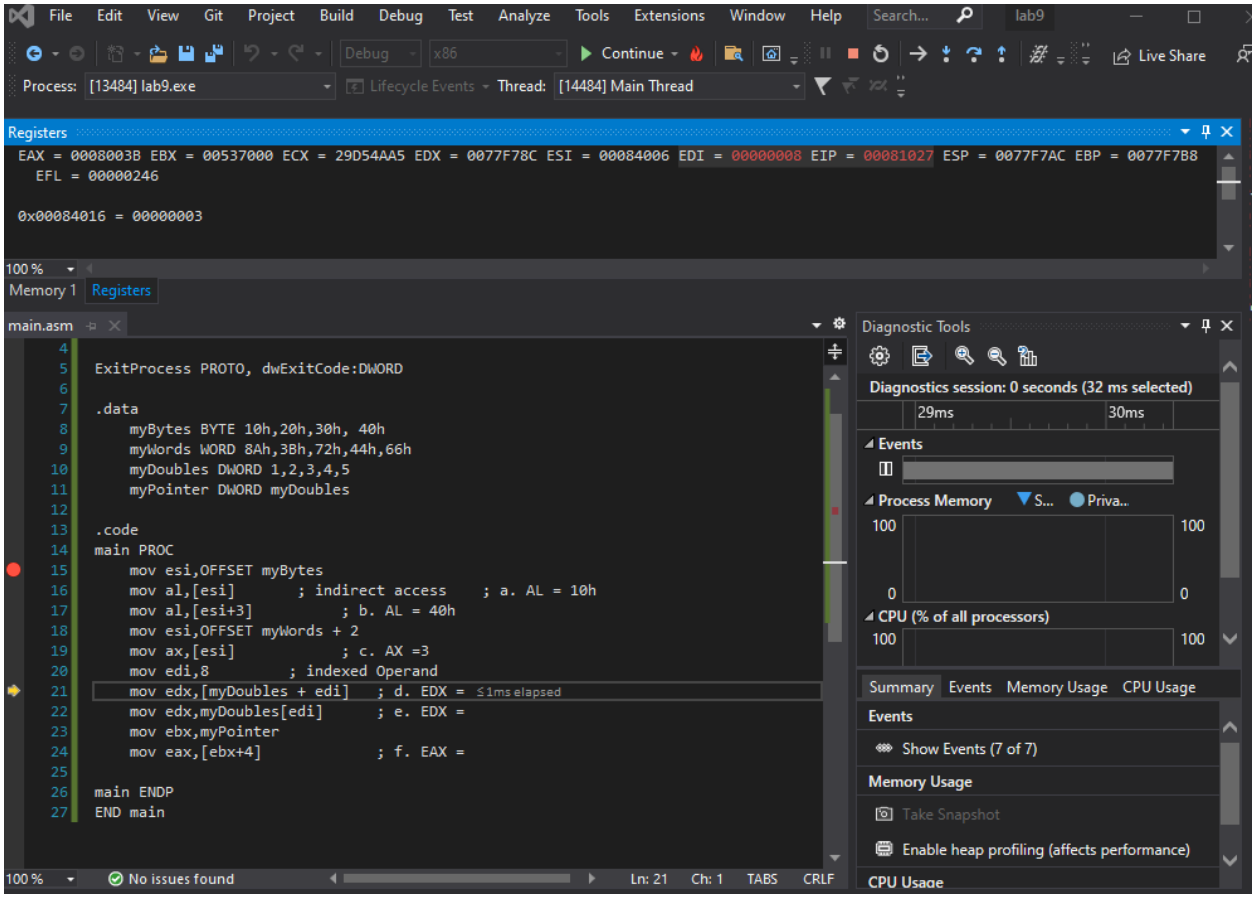
Explanation:In this instruction we are moving the second value of the myWords variable into the eax register.

Line: 20

Instruction: mov edi, 8

Register value: EDI = 00000008 EIP = 00081027

Screenshot:



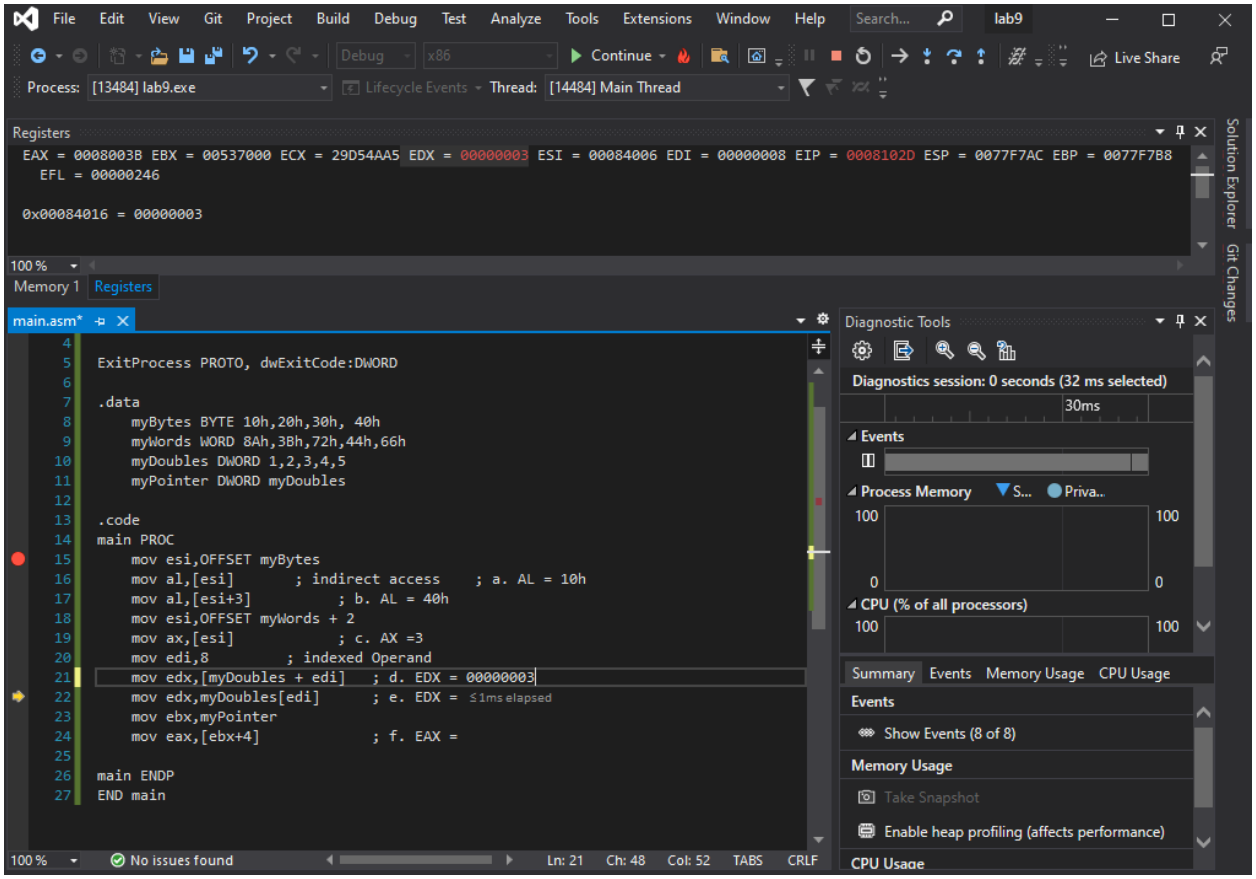
Explanation: In this instruction we moving the value 8 into the edi register

Line: 21

Instruction: mov edx, [myDoubles + edi]

Register value: EDX = 00000003

Screenshot:



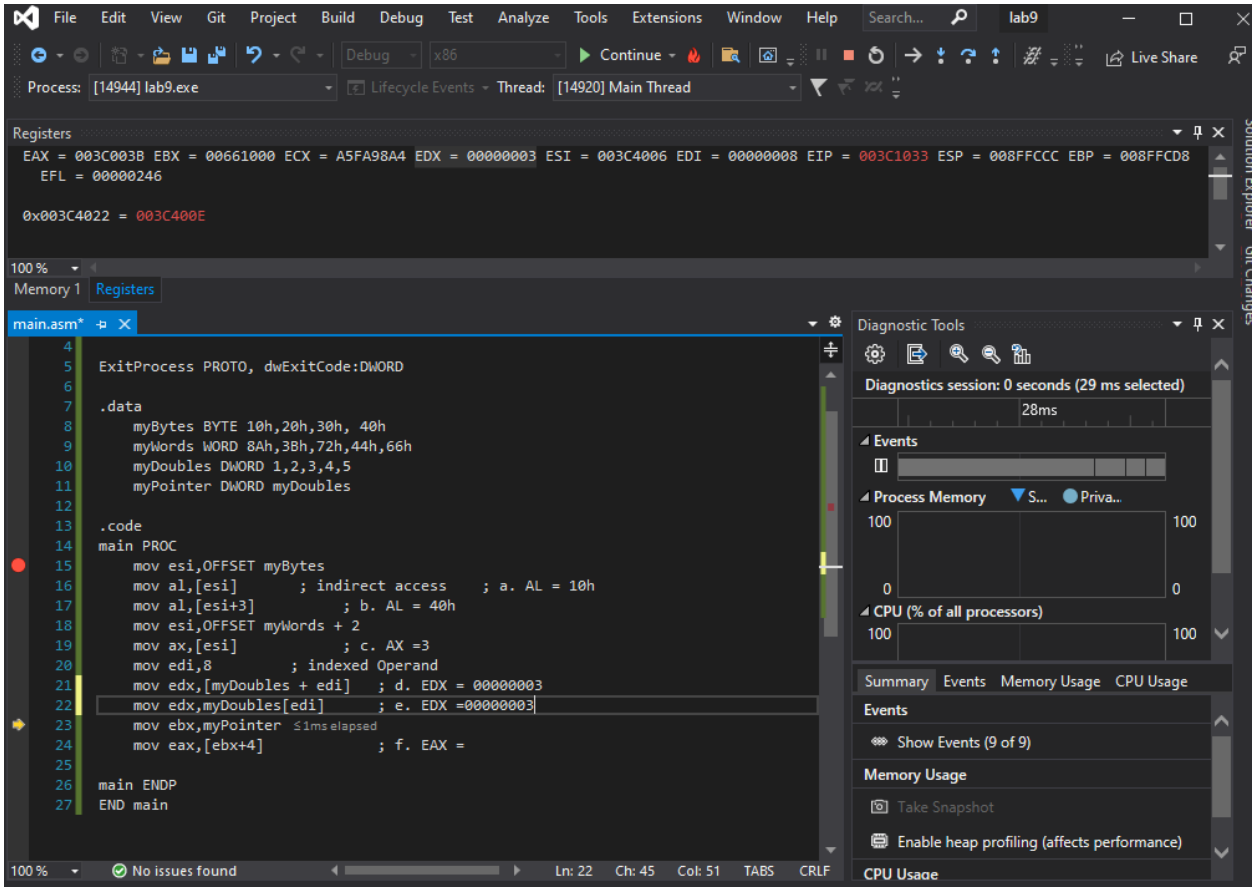
Explanation: In this instruction we are moving the third element of myDoubles variable into the the edx register.

Line: 22

Instruction: mov edx, myDoubles[edi]

Register value:EDX = 00000003

Screenshot:



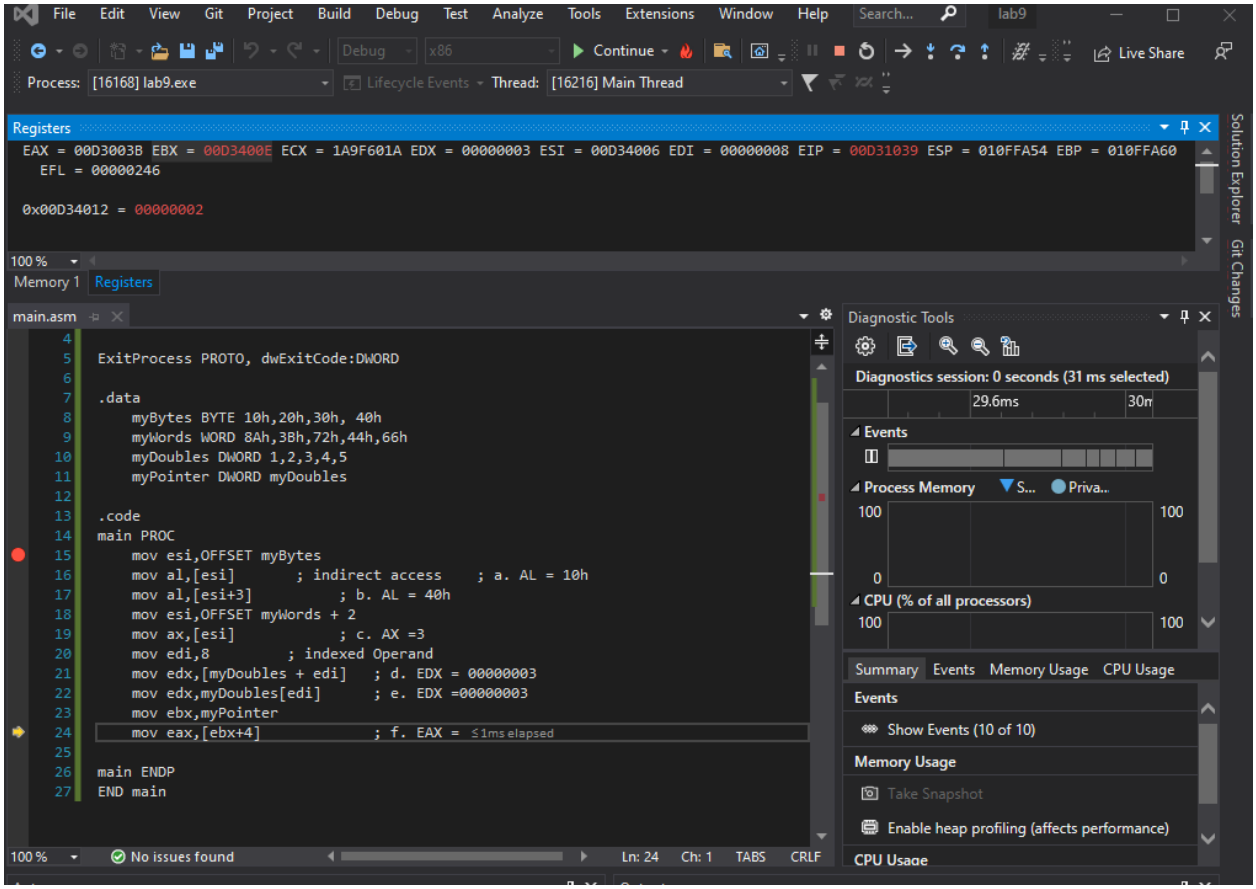
Explanation: The values of the edx register does not change because we are basically moving the same elements into the edx register.

Line: 23

Instruction: mov ebx, myPointer

Register value: EBX = 00D3400E

Screenshot:



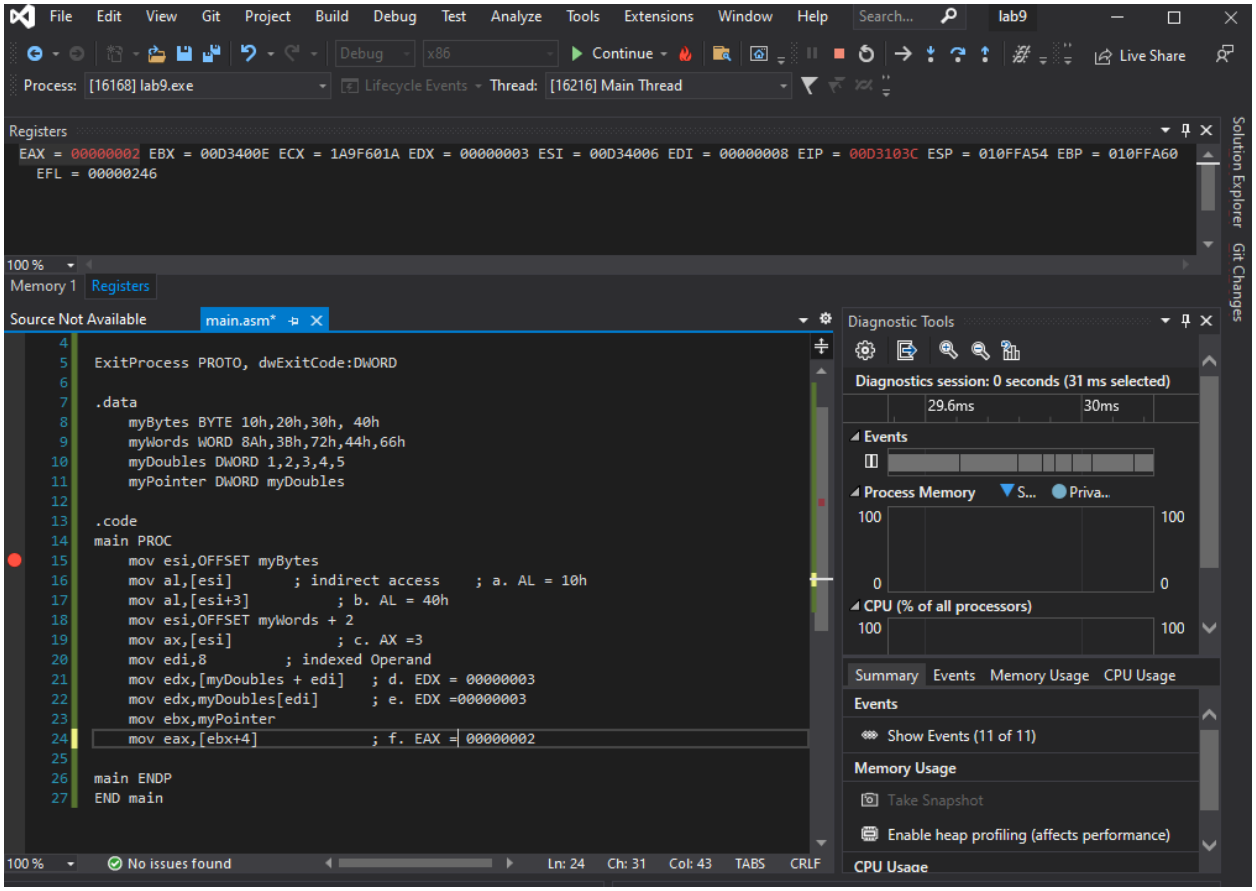
Explanation: In this instruction we are moving the memory address of myDoubles array into the ebx register.

Line: 24

Instruction: mov eax,[ebx+4]

Register value: EAX = 00000002

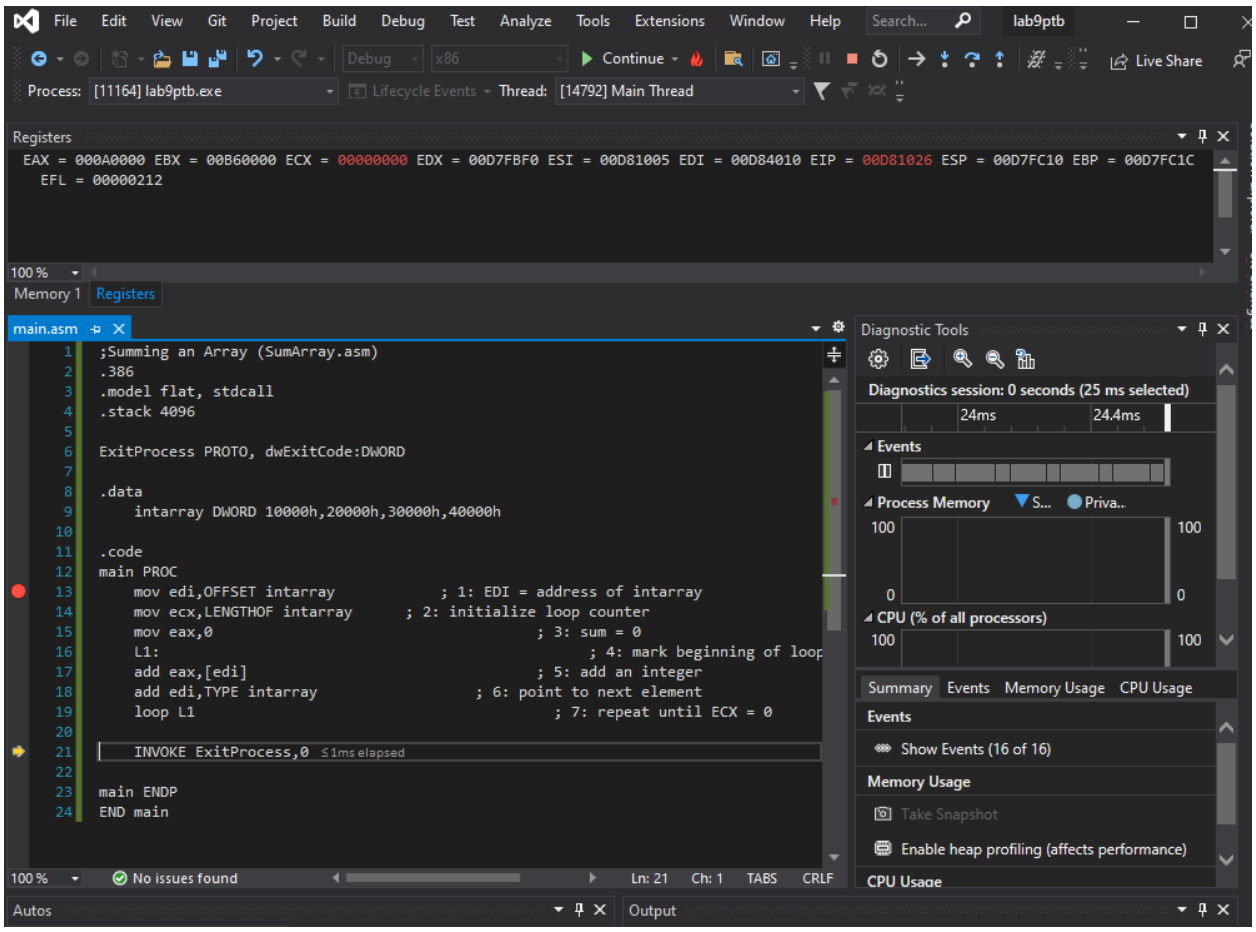
Screenshot:



Explanation: In this instruction we are moving the second element of myDoubles array into the eax register and then the address in the ebx register is accessed after we add 4 to it.

Lab 9B

Debug until you reach “INVOKE ExitProcess,0” . Take a screenshot of the code and register window at the end. Record the content of the EAX register. Then explain the register content.



Explanation: After performing the whole loop we can see that the whole contents in the EAX register is 000A0000. So basically what’s happening is that we are adding up all the values in the inarray variable are being added into one. Like below:

0 0 0 1 0 0 0 0 h

+0 0 0 2 0 0 0 0 h

+0 0 0 3 0 0 0 0 h

+0 0 0 4 0 0 0 0 h

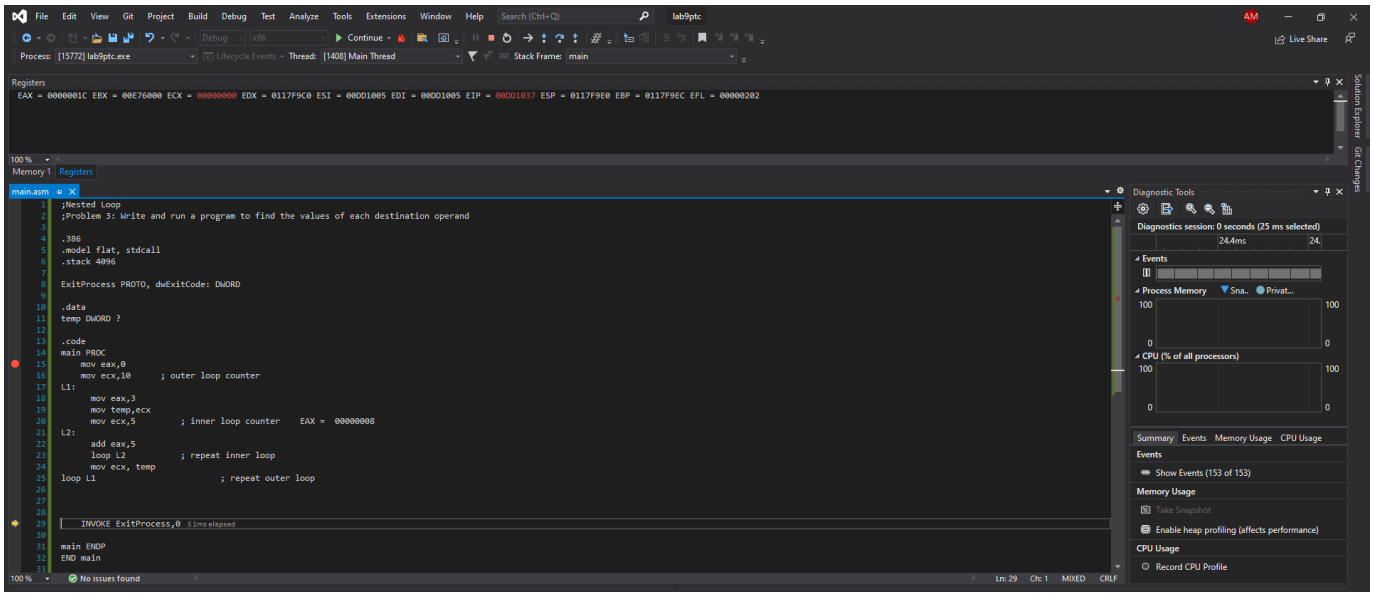
---

0 0 0 A 0 0 0 0 h

So if we add 1 + 2 + 3 + 4 = 10 and 10 in hexadecimal is A.

Lab 9 C

Debug until you reach “INVOKE ExitProcess,0”. Take a screenshot of the code and the register window at the end. Record the EAX register. Then explain the EAX register contents.



EAX = 0000001C

So in this whole code we are basically running the inside loop for 5 times of every iteration of 10 loops, which means it will run for 50 times. So for each iteration, which is 5 iterations, 5 is added to three. In each iteration, EAX is rest but at the end, we will have 28d, which is 1C

Lab 9 D

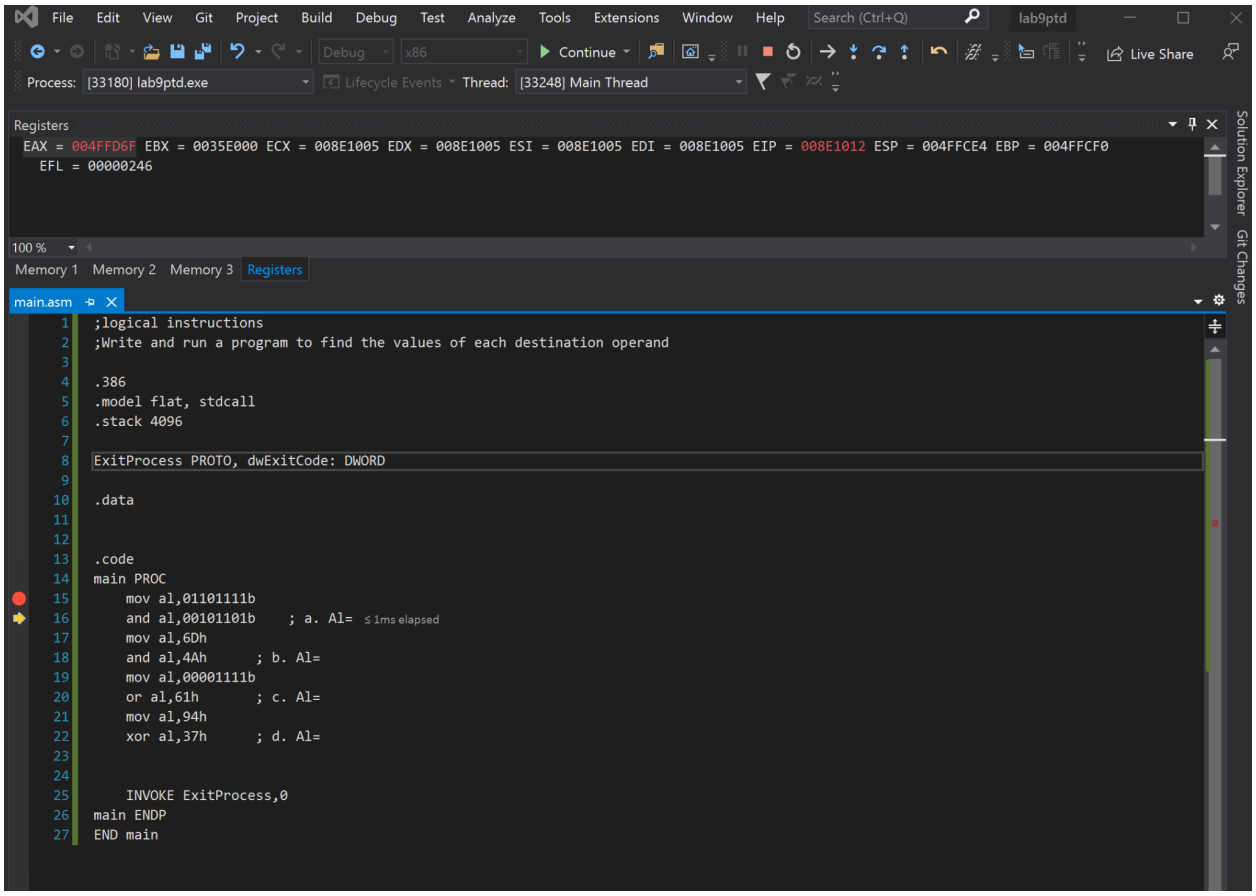
Debug through each line of code. Execute the instruction. Take a screenshot of the code and register the window. Record the line number, instruction, register values in the answer sheet. Then explain the register or memory contents.

Line: 15

Instruction: mov al, 01101111b

Register value: EAX = 004FFD6F

Screenshot:



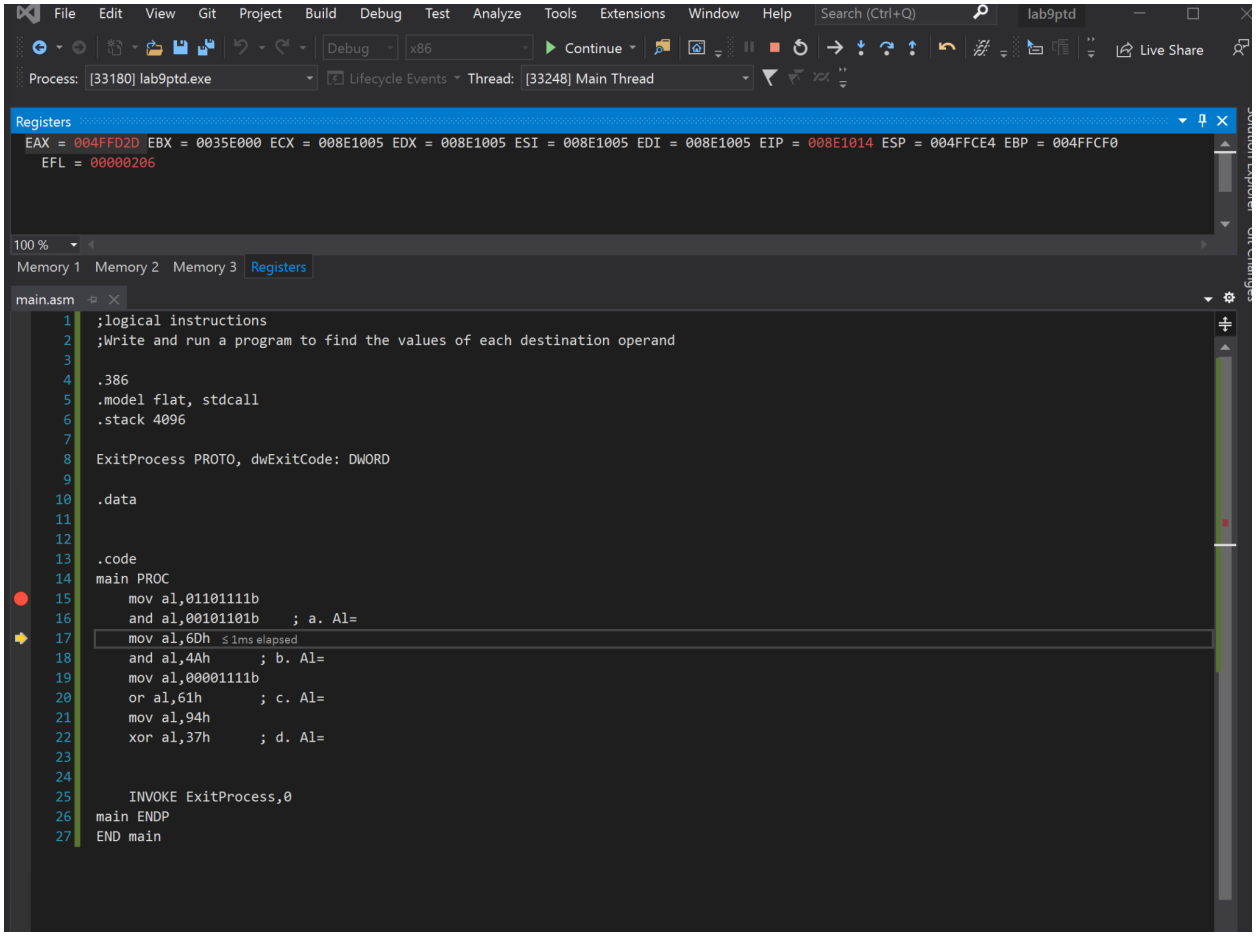
Explanation: In this instruction, we are moving the hexadecimal value 01101111b into the al register as the hexadecimal value.

Line: 16

Instruction: and al, 00101101b

Register value: EAX = 004FFD2D

Screenshot:



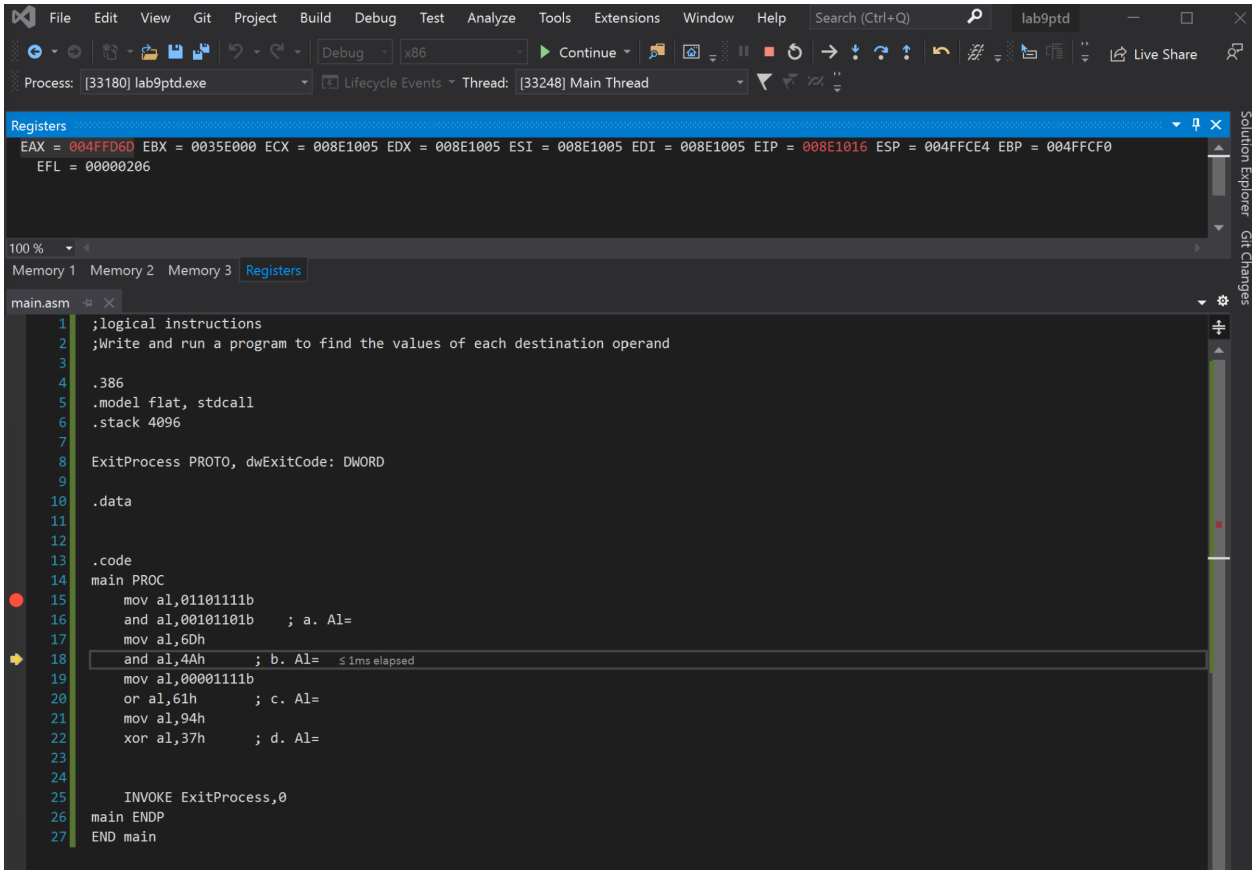
Explanation: In this instruction, we are using the and instruction, and instruction is known to clear the overflow and carry flags. It modifies the sign, zero, and parity flags. The bits that are the same in both areas are carried, and others are replaced with zeros.

Line: 17

Instruction: mov al 6Dh

Register value: EAX = 004FFD6D

Screenshot:



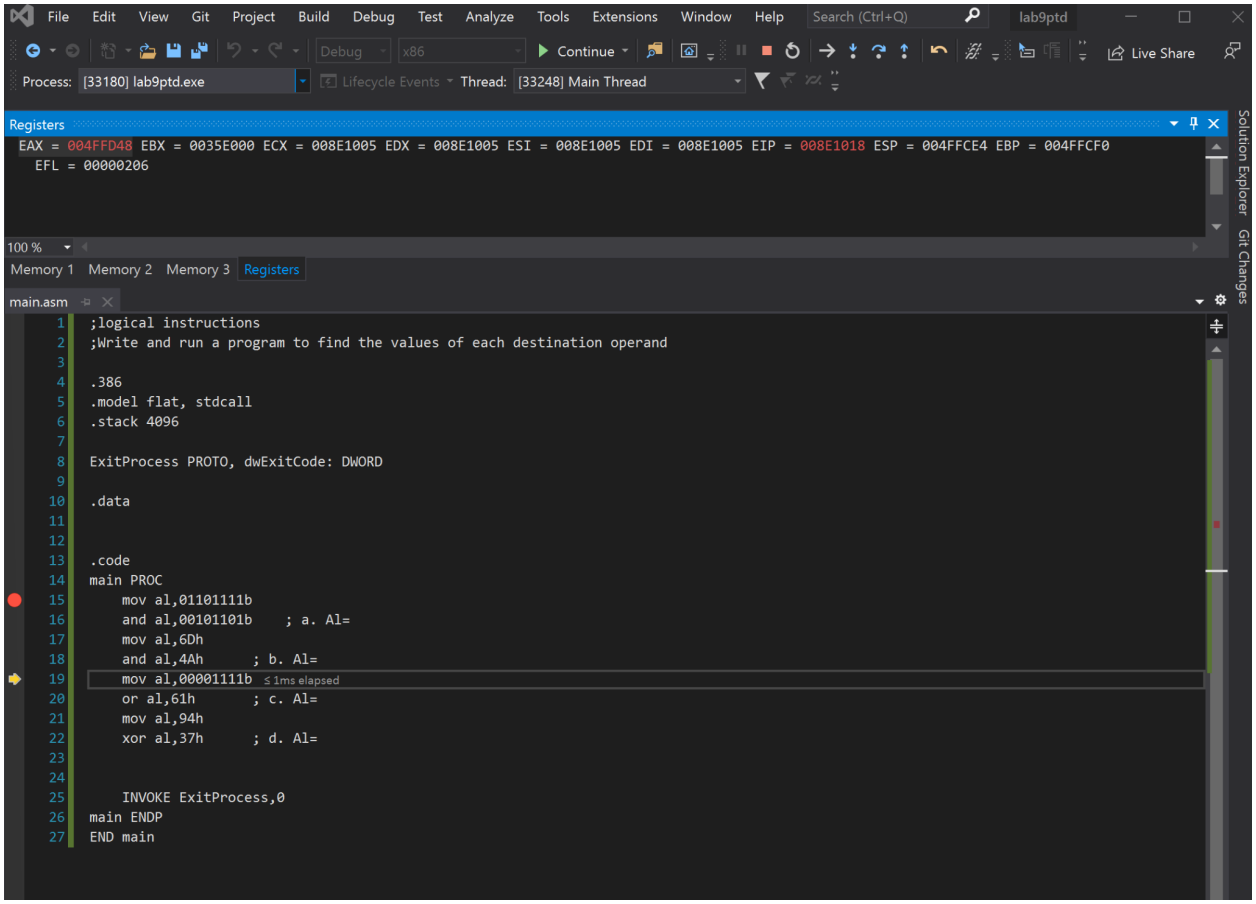
Explanation: In this instruction, we are moving the value 6Dh into the al register.

Line: 18

Instruction: and al, 4Ah

Register value: EAX = 004FFD48

Screenshot:



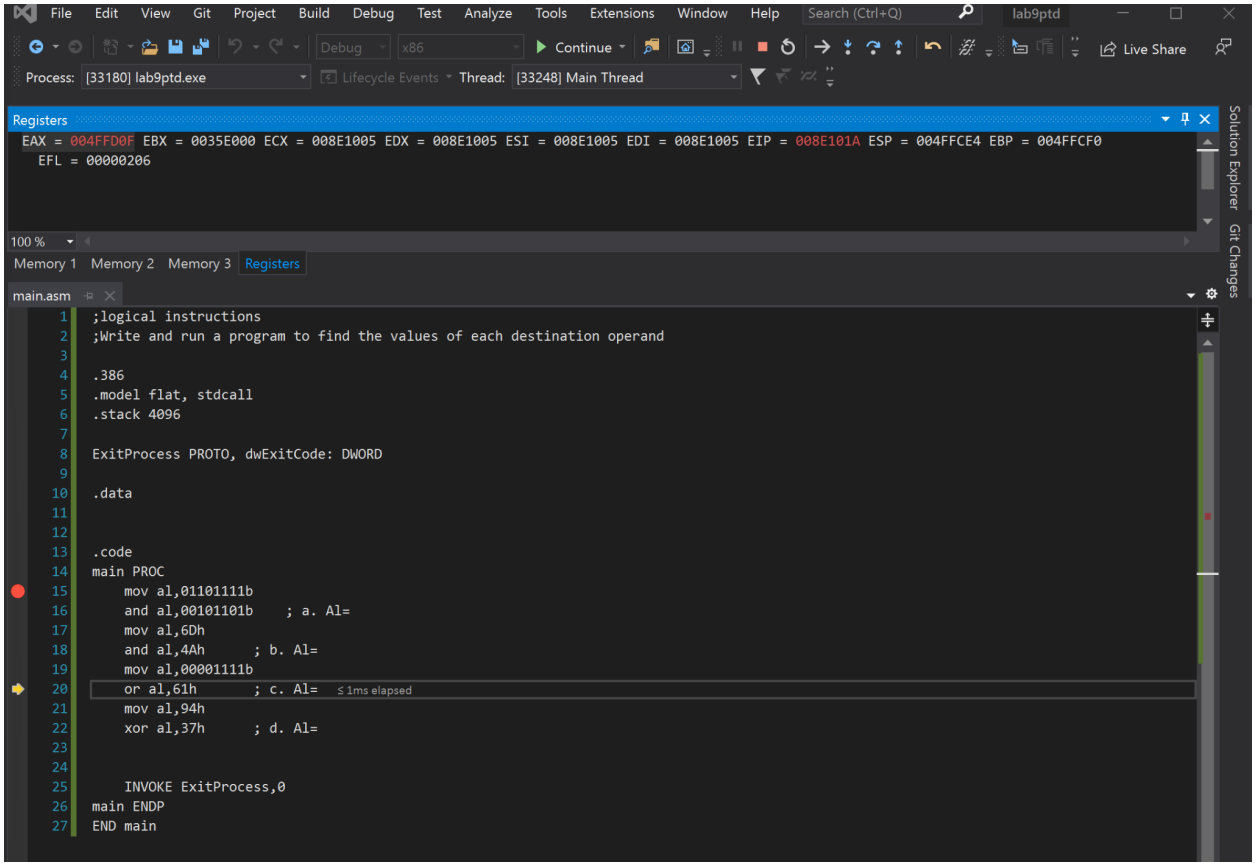
Explanation: In this instruction we are using the and operation again, the and operation has been performed between two numbers, the same digits are carried and others are 0.

Line: 19

Instruction: mov al, 00001111b

Register value: EAX = 004FFD0F

Screenshot:



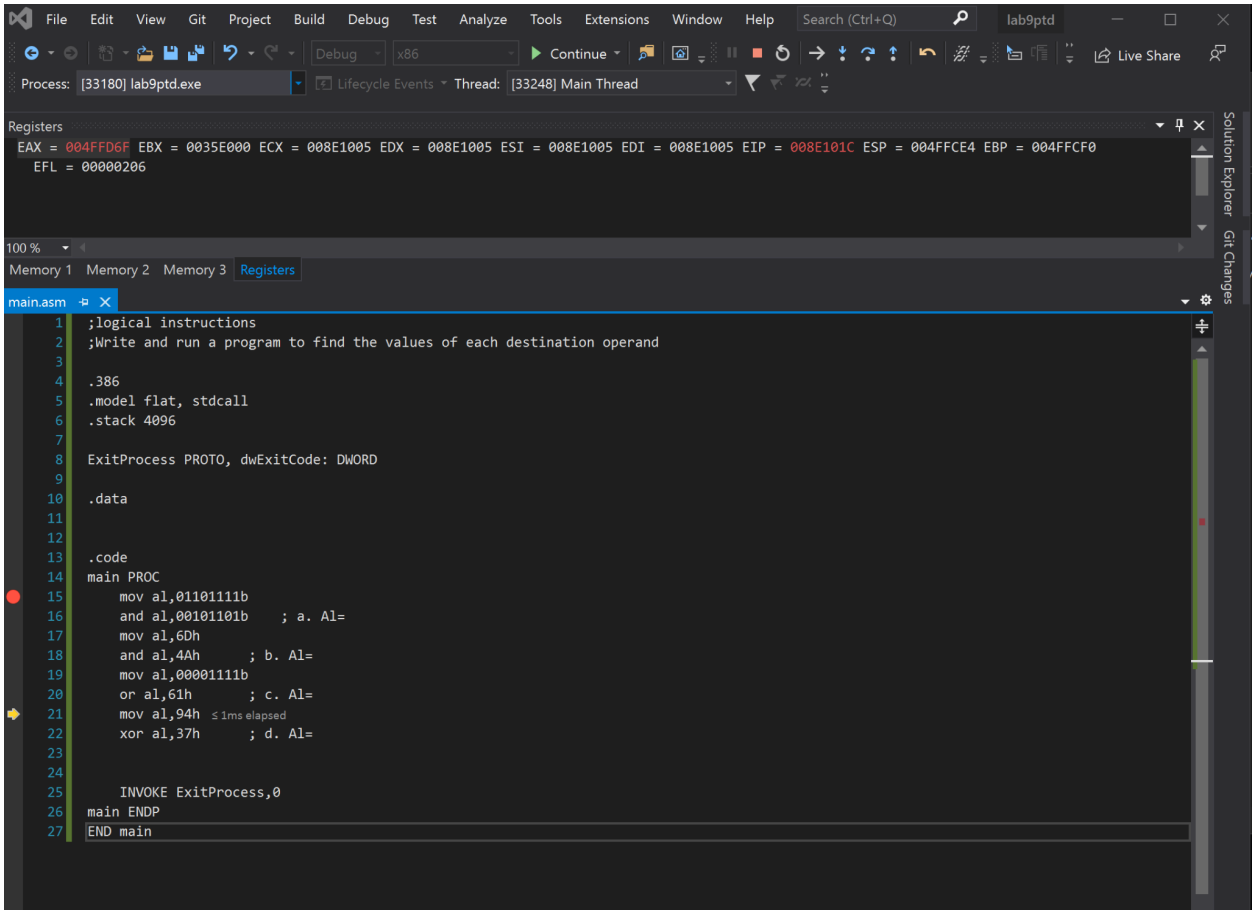
Explanation: In this instruction we are moving the value 00001111b into the al register, the value for 15 binary is moved to eax register.

Line: 20

Instruction: or al, 61h

Register value: EAX = 004FFD6F

Screenshot:



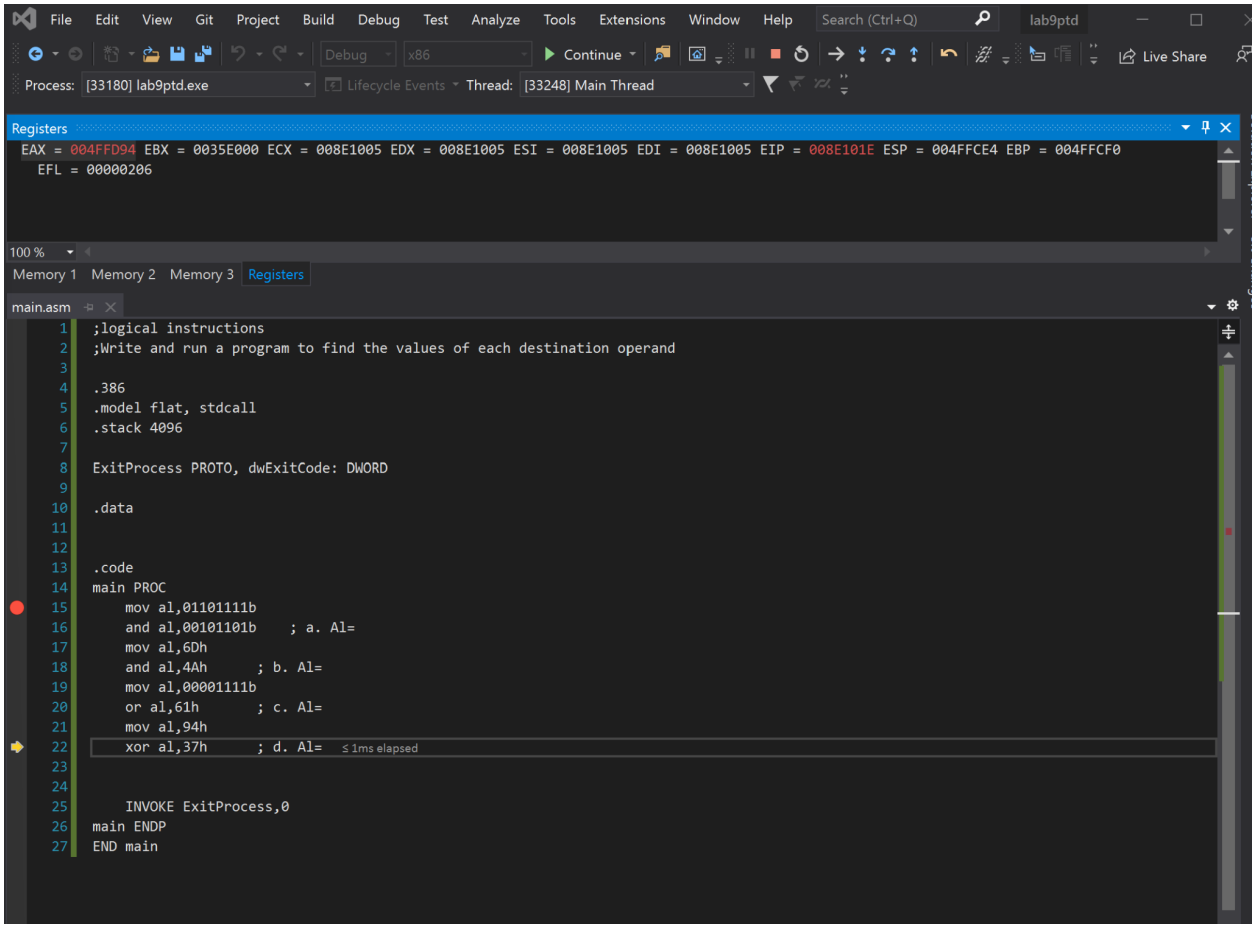
Explanation: In this instruction we are using the OR instruction, which basically performs a Boolean OR operation between each pair of matching bits in two operands. The OR instruction always clears the carry and overflow flags, it modifies the sign, zero and parity flags. The values with larger numbers of 1 bits are carried over due to being an OR instruction.

Line: 21

Instruction: mov al, 94h

Register value: EAX = 004FFD94

Screenshot:



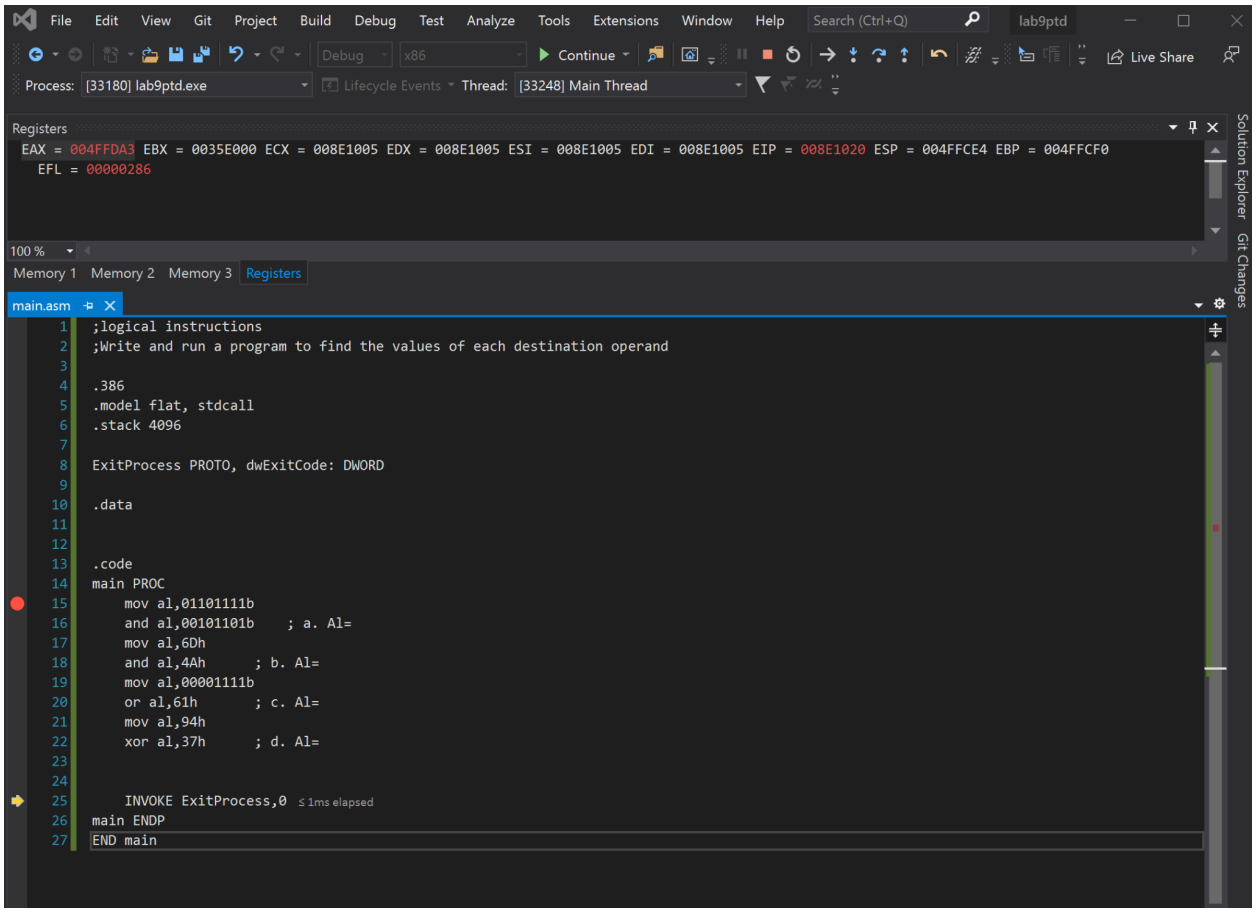
Explanation: in this instruction, we are moving the value 94h into the al register, which is then moved into eax register.

Line: 22

Instruction: xor al, 37h

Register value: EAX = 004FFDA3

Screenshot:



Explanation: In this instruction, we are using xor instruction, which basically performs exclusive OR operation between each pair of matching bits in two operands. The xor operation with 2 numbers, every bit with the different value replaced with 1 and same values are replaced with 0 bits

Complete for the all the lines inside main