(5 pts) CS 3844 Computer Organization - Lab #02 Name/abc123: Ariel Guerrero aes604

Due Fri, Sep 11th 2020 11:59 pm

This lab introduces you to a bit (well, 32 bits) of the x86 Intel architecture. The program adds a couple of one byte values and saves the result in a register. It also extends the one byte values to 4 bytes (32 bits). The goal of this lab is three-fold: #1 – Get the console project working in Visual Studio and use the debugger to examine register and memory values, #2 – See how C/C++ code and assembly code can interact, and #3 – see how registers are used to manipulate data of different sizes.

There is a 20 minute narration with a sample lab and a Lab2.cpp file. I HIGHLY recommend taking the time to watch the narration as it will save you more than 20 minutes in figuring this stuff out. First, create an empty console application and add Lab2.cpp as an existing file. Don't forget to set the multi-threaded debug option. There are no input parameters with this lab. Then run the program and observe the results.

To answer the following questions, set a breakpoint at the first assembly instruction, "xor eax,eax" and run the program – make sure the output window (all black) is visible. Open one memory window and the Registers window. Each question will guide you to the next. Turn in the answers to the questions below.

1. What is the value of EIP? Note that the "xor" instruction has not executed yet.

   Value: 33 C0 33 DB address: 0x 0043 4C2A

2. In the memory window, type &ssum32 and press enter. Now, hit the "step into" arrow at the top 3 times until the yellow arrow (EIP = 0x42E717) is pointing to "mov bl,uy." Note that EAX has turned red.

   a. What is the value of EAX in decimal?

      251

   b. What is the value of AX in decimal?

      251

   c. What is the value of AH in decimal?

      0

   d. What is the unsigned value of AL in decimal?

      251

   e. What is the signed value of AL in decimal?

      -5

3. Hit the "step into" arrow 1 time so the yellow arrow points at "add al,bl". Note that EBX is red now.

   a. What is the unsigned value of BL in decimal?

      240

   b. What is the signed value of BL in decimal?

      -16

EAX: 0000 00 EB
A1

4. Now, we are going to add BL to AL and store the result in AL. Hit the "step into" arrow 1 time.

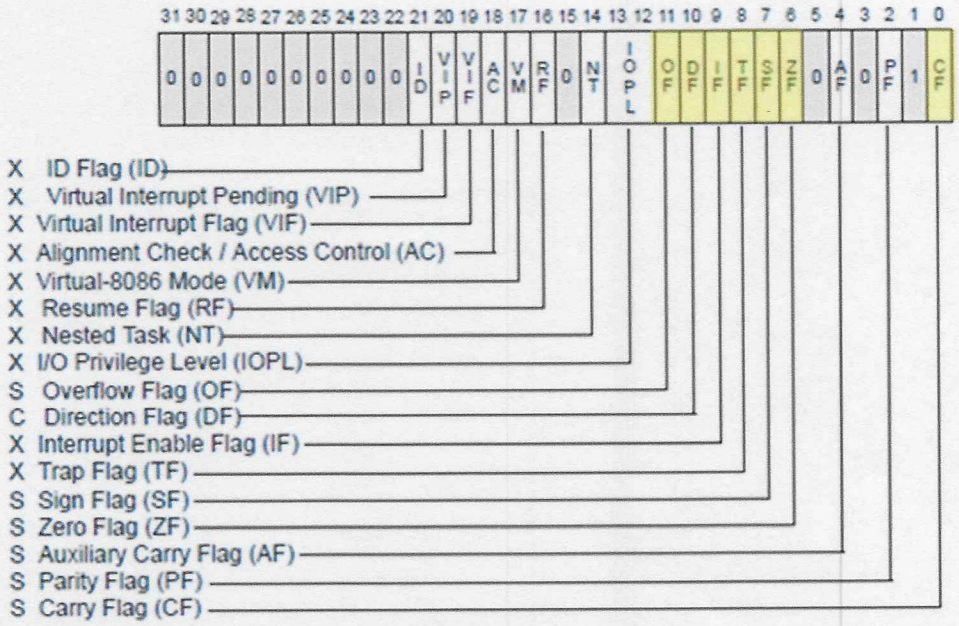
   a. What is the unsigned decimal value in AL? 1110 1011 — 235

   b. What is the signed value in AL? 1110 1011 → 0001 0100 → 0001 0101 → -21

   c. What is the hexadecimal value of the flags register? (EFL) 0x 0000 0287

   d. From the 8-bit math, what do you think the flags should be? Use N,V,Z,C for flag shorthand. (N=?, V=?, Z=?, C=?) N=1 V=0 Z=1 C=0

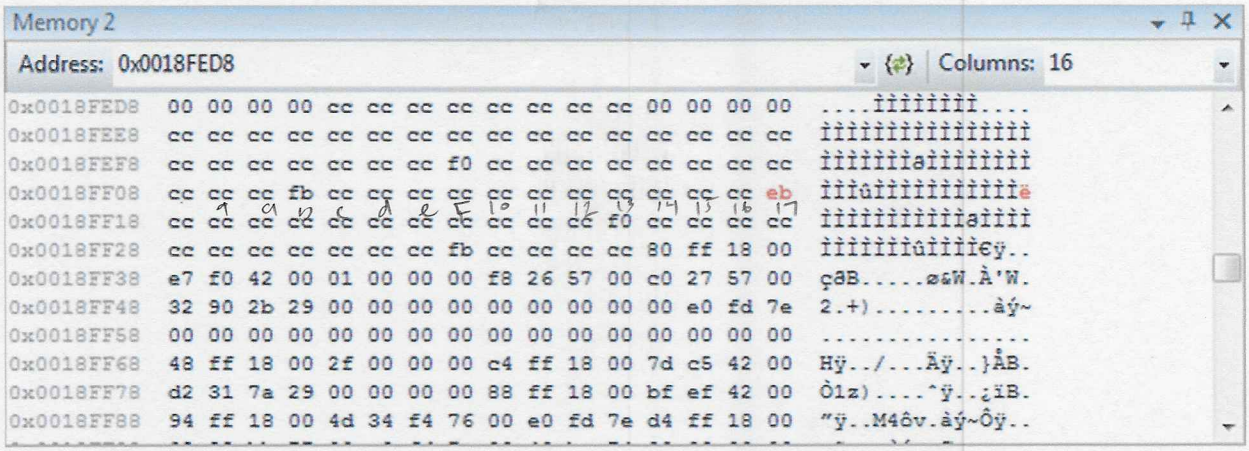
   e. Convert your answer for the EFL to binary. Check the image below and see if your answers match. Note: I use N, V, Z, C for flag shorthand, but here they use SF = SignFlag, OF = OverflowFlag, ZF=ZeroFlag, CF=CarryFlag.



0000 0010 1000 0111
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

CF=1
ZF=0
SF=1
OF=0

5. Hit the "step into" arrow 1 time. That saves the one byte in AL to stack memory. The address may be different, but in the memory window, you should see one byte turn red and have the value of 0xEB. Given the screenshot of MY run-through, at what address is 0xEB stored? (Yes, it's the usum variable's address.)



0x 0018FF17

6. Hit the "step into" arrow 1 time. Note the change in ecx. "movzx" is a move with zero-extend. That is used to take a smaller unsigned number and convert it to the size of the destination register. In this case, 8 bits to 16 bits. What is the decimal value in ecx?

US signed
eb = 235 = -21

ecx → eb 00

1110 1011 0000 0000

7. Hit the "step into" arrow 1 time. ECX was stored in memory. Find it and using the memory window above, at what address was ECX stored? (Note: only 1 byte turned red even though 4 were stored because usum32 was initialized to zero.) <It is possible that your compiler has it at a different relative location. If unsure, include a screenshot of your memory window.>

0x 0018FF17

8. Hit the "step into" arrow 1 time.

EAX= 0000 FBEB

   a. What is the value of EAX in hexadecimal?

0x0000 FBEB

   b. What is the value of AX in hexadecimal?

0xFBEB

   c. What is the value of AL in hexadecimal? (It did NOT change)

0xEB

   d. What is the unsigned value of AH in decimal? (Note, even though sx is signed, it makes no difference as to the hex value in the register.)

0xFB

1111 1011

   e. What is the signed value of AH in decimal?

-21

9. Hit the "step into" arrow 1 time. See EBX and BH change. Step again. See AH change to the same value as AL. Step again and the red 0xEB is the address of ssum. What is the current value of EDX? (May be different for everyone, but put it here anyway.) Include any leading zeros.

0x 0061 0008

10. Step again. The "movsx" is a move with sign extend. Check out EDX.
    a. What is the hexadecimal value of EDX? 0x FFFFFF EB
    b. What is the signed decimal value of EDX? -21
    c. What is the hexadecimal value in DX? 0x FFEB
    d. What is the signed decimal value in DX? -21
    e. What is the signed decimal value in DL? -21
    f. What is the signed decimal value in DH? -1

11. Place a breakpoint on the last printf. Hit the green "Continue" arrow at the top. Note the output in the black output window. Also note that ssum32 has changed – it should be the first set of bytes in your memory window. Show how ssum is represented in memory:
    -21 is in DL

12. Continue again. Should stop at system pause. Note the output. Congratulations, you have completed the lab.

usum32 = 0x 0000 00EB = 235

ssum32 = 0x FF FF FFEB = -21