

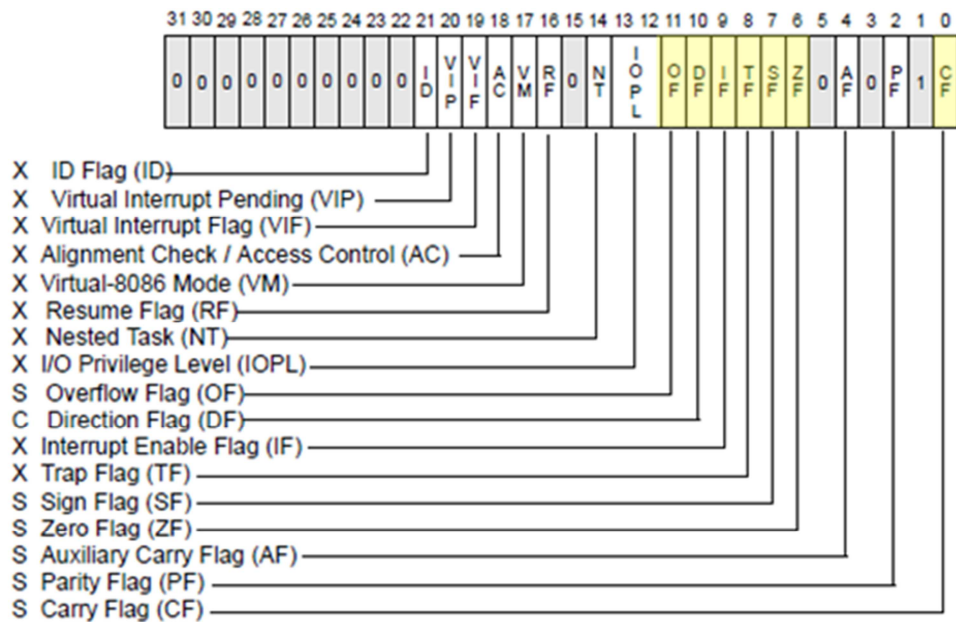
This lab introduces you to a bit (well, 32 bits) of the x86 Intel architecture. The program adds a couple of one byte values and saves the result in a register. It also extends the one byte values to 4 bytes (32 bits). The goal of this lab is three-fold: #1 – Get the console project working in Visual 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.
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?
 - b. What is the value of AX in decimal?
 - c. What is the value of AH in decimal?
 - d. What is the unsigned value of AL in decimal?
 - e. What is the signed value of AL in decimal?
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?
 - b. What is the signed value of BL in decimal?

4. Now, we are going to add BL to AL and store the result in AL. Hit the “step into” arrow 1 time.
- What is the unsigned decimal value in AL?
 - What is the signed value in AL?
 - What is the hexadecimal value of the flags register? (EFL)
 - 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=?)
 - 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.



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.)

Memory 2															
Address: 0x0018FED8															
Columns: 16															
0x0018FED8	00	00	00	00	cc	cc	cc	cc	cc	cc	cc	cc	00	00	00
0x0018FEE8	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0018FEF8	cc	cc	cc	cc	cc	cc	cc	f0	cc	cc	cc	cc	cc	cc	cc
0x0018FF08	cc	cc	cc	fb	cc	cc	cc	cc	cc	cc	cc	cc	cc	eb	cc
0x0018FF18	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	f0	cc	cc	cc
0x0018FF28	cc	cc	cc	cc	cc	cc	cc	fb	cc	cc	cc	cc	80	ff	18
0x0018FF38	e7	f0	42	00	01	00	00	00	f8	26	57	00	c0	27	57
0x0018FF48	32	90	2b	29	00	00	00	00	00	00	00	00	00	e0	fd
0x0018FF58	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0018FF68	48	ff	18	00	2f	00	00	00	c4	ff	18	00	7d	c5	42
0x0018FF78	d2	31	7a	29	00	00	00	00	88	ff	18	00	bf	ef	42
0x0018FF88	94	ff	18	00	4d	34	f4	76	00	e0	fd	7e	d4	ff	18

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?
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.>
8. Hit the “step into” arrow 1 time.
- What is the value of EAX in hexadecimal?
 - What is the value of AX in hexadecimal?
 - What is the value of AL in hexadecimal? (It did NOT change)
 - 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.)
 - What is the signed value of AH in decimal?
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.

10. Step again. The “movsx” is a move with sign extend. Check out EDX.

- a. What is the hexadecimal value of EDX?
- b. What is the signed decimal value of EDX?
- c. What is the hexadecimal value in DX?
- d. What is the signed decimal value in DX?
- e. What is the signed decimal value in DL?
- f. What is the signed decimal value in DH?

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:

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