16.317: Microprocessor Systems Design I

Fall 2015

Homework 1 Solution

1. *(50 points) Given each of the binary or hexadecimal number below, determine what the decimal value is if the number is (i) an unsigned integer, and (ii) a signed integer. Note that, in some cases, your answers for both will be the same.*

a. *01011101_2*

Since MSB = 0, value is same whether unsigned or signed—figure out the significance of each position in which a bit = 1, and sum those values together.

$01011101_2 = 64 + 16 + 8 + 4 + 1 =$ **93**

b. *10100110_2*

For an unsigned integer, we use the same method as in part (a)

$10100110_2 = 128 + 32 + 4 + 2 =$ **166**

For a signed integer, recognize that this value is negative; to find its magnitude, take the two's complement:

$-10100110_2 = 01011001_2 + 1 = 01011010_2 = 64 + 16 + 8 + 2 = 90$

Therefore, $10100110_2 =$ **-90** when treated as a signed integer.

c. *9Ch (or 0x9C—recall that, in x86 assembly notation, the "h" at the end of a number signifies that the previous value is in hexadecimal)*

For an unsigned integer, we don't really need to convert to binary; if you want to do so, 9Ch = 10011100_2. However, we can also just convert directly to decimal:

$9C_{16} = (9 \times 16) + (12 \times 1) =$ **156**

As a signed integer, note that this value is negative, since its MSB = 1. To find the magnitude, once again take the two's complement:

$-10011100_2 = 01100011_2 + 1 = 01100100_2 = 64 + 32 + 4 = 100$

Therefore, as a signed integer, 9Ch = **-100.**

d. 4B93h

Since the most significant bit of this number is 0 (4B93h = 0100 1011 1001 0011_2), it has the same value whether it is treated as a signed or unsigned integer. That value is:

$(4 \times 16^3) + (11 \times 16^2) + (9 \times 16^1) + (3 \times 16^0) =$
$(4 \times 4096) + (11 \times 256) + (9 \times 16) + (3 \times 1) = 16384 + 2816 + 144 + 3 =$ **19347**

e. FEEDh

This number has different values when treated as signed or unsigned, since the MSB is 1 (FEEDh = 1111 1110 1110 1101_2). As an unsigned integer:

$(15 \times 16^3) + (14 \times 16^2) + (14 \times 16^1) + (13 \times 16^0) =$
$(15 \times 4096) + (14 \times 256) + (14 \times 16) + (13 \times 1) =$
$61440 + 3584 + 224 + 13 =$ **65261**

As a signed integer, the magnitude is:

-FEEDh = $-1111\ 1110\ 1110\ 1101_2 = 0000\ 0001\ 0001\ 0011_2 =$ 0113h

I've shown the conversion back into hexadecimal because it might be slightly easier to figure out the decimal value of the magnitude using what we already know about converting a 16-bit value from hex to decimal:

$(0 \times 16^3) + (1 \times 16^2) + (1 \times 16^1) + (3 \times 16^0) =$
$(0 \times 4096) + (1 \times 256) + (1 \times 16) + (3 \times 1) = 256 + 16 + 3 = 275$

Therefore, FEEDh = **-275** as a signed integer.

2. *(50 points) Assume the contents of memory are shown below. All values are in hexadecimal. The table shows four bytes per line; the given address is the starting address of each line.*

   Each block in the table contains a single byte, with the low and high bytes per line indicated as shown. Each byte has its own address, so the byte at address 20590h is 09h, address 20591h is 12h, address 20592h is 15h, and address 20593h is 20h.

   You should assume all multi-byte values are stored in little-endian format.

Address	Lo			Hi
20590h	09	12	15	20
20594h	62	AB	DD	EF
20598h	11	3C	77	91
2059Ch	CF	06	48	55
205A0h	78	6D	72	00
205A4h	B3	46	13	17
205A8h	39	30	C7	B9

Furthermore, assume the following initial register values:

- *EBX = 000205A0h*
- *ECX = FFFFFFF4h*
- *ESI = 00020590h*
- *EDI = 00000005h*

a. *(10 points) In class, we discussed how to determine if words or double-words (in an x86 processor) are aligned. Newer Intel processors support quad-words that contain 64 bits of data. Determine the starting addresses of all aligned quad-words shown in the diagram above, and explain why those addresses are aligned.*

For aligned data, the address must be divisible by the number of bytes being accessed. Since a quad-word is 8 bytes, aligned word addresses must be divisible by 8:

20590h, 20598h, 205A0h, and 205A8h

b. *(10 points) What is the result of the instruction MOV AL, [2059Ah]? What is the decimal value of the data transferred in this instruction?*

The byte at address 2059Ah is the third byte in the line starting with 20598h, which is **77h**.

This byte has the same decimal value whether treated as signed or unsigned, since the most significant bit is 0: (7 x 16) + (7 x 1) = 112 + 7 = **119**.

c. (10 points) What is the result of the instruction MOV DX, [ESI+001Ah]? Is this memory access aligned?

This instruction loads a word into DX starting at address ESI+001Ah = 20590h + 001Ah = 205AAh. Since x86 data are little-endian, that word is **B9C7h**.

This access is **aligned**, since the starting address of the word (205AAh) is divisible by 2, the number of bytes being accessed.

d. (10 points) What is the result of the instruction MOV EDX, [EBX+ECX]? (Hint: treat ECX as a signed, two's complement integer and check its decimal value.) Is this memory access aligned?

This instruction accesses four bytes starting at address EBX+ECX = 205A0h + FFFFFFF4h. Note that, as a signed two's complement integer, FFFFFFF4h = -12, so the address being accessed is 205A0h – 12 = 20594h. The double word at that address is **EFDDAB62h.**

This access **is aligned**, because the starting address of the double word (20594h) is divisible by 4, the number of bytes being accessed.

*e. (10 points) What is the result of the instruction MOV [ESI+2*EDI], AX, if EAX = 3399FFAAh? Is this memory access aligned?*

This instruction writes two bytes of data into memory (since the given address is the destination operand) starting at address ESI+2*EDI = 20590h + 2*00000005h = 20590h + 0000000Ah = 2059Ah. Those bytes are the lowest two bytes of register EAX (FFAAh); since the least significant byte is stored first, the result is as shown below, with the changed bytes underlined:

Address	Lo			Hi
20598h	11	3C	**AA**	**FF**

This access is aligned, since 2059Ah is divisible by 2, the number of bytes being accessed.