16.317: Microprocessor Systems Design I

Spring 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. 01110110_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.

$01110110_2 = 64 + 32 + 16 + 4 + 2 =$ **118**

b. 10011001_2

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

$10011001_2 = 128 + 16 + 8 + 1 =$ **153**

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

$-10011001_2 = 01100110_2 + 1 = 01100111_2 = 64 + 32 + 4 + 2 + 1 = 103$

Therefore, $10011001_2 =$ **-103** when treated as a signed integer.

c. BCh (or 0xBC—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, BCh = 10111100_2. However, we can also just convert directly to decimal:

$AE_{16} = (11 \times 16) + (12 \times 1) =$ **188**

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:

$-10111100_2 = 01000011_2 + 1 = 01000100_2 = 64 + 4 = 68$

Therefore, as a signed integer, BCh = **-68.**

d. 61DFh

Since the most significant bit of this number is 0 (61DFh = 0110 0001 1101 1111_2), it has the same value whether it is treated as a signed or unsigned integer. That value is:

$(6 \times 16^3) + (1 \times 16^2) + (13 \times 16^1) + (15 \times 16^0) =$
$(6 \times 4096) + (1 \times 256) + (13 \times 16) + (15 \times 1) = 24576 + 256 + 208 + 15 =$ **25055**

e. ACEDh

This number has different values when treated as signed or unsigned, since the MSB is 1 (ACEDh = 1010 1100 1110 1101_2). As an unsigned integer:

$(10 \times 16^3) + (12 \times 16^2) + (14 \times 16^1) + (13 \times 16^0) =$
$(10 \times 4096) + (12 \times 256) + (14 \times 16) + (13 \times 1) =$
40960 + 3072 + 224 + 13 = **44269**

As a signed integer, the magnitude is:

-ACEDh = -1010 1100 1110 1101_2 = 0101 0011 0001 0011_2 = 5313h

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:

$(5 \times 16^3) + (3 \times 16^2) + (1 \times 16^1) + (3 \times 16^0) =$
$(5 \times 4096) + (3 \times 256) + (1 \times 16) + (3 \times 1) = 20480 + 768 + 16 + 3 = 21267$

Therefore, ACEDh = **-21267** 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 10430h is 01h, address 10431h is 30h, address 10432h is 20h, and address 10433h is 15h.

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

Address	*Lo*			*Hi*
10430h	*01*	*30*	*20*	*15*
10434h	*10*	*47*	*FE*	*AA*
10438h	*18*	*1F*	*15*	*2B*
1043Ch	*BE*	*60*	*10*	*99*
10440h	*78*	*D6*	*32*	*33*
10444h	*34*	*35*	*12*	*16*
10448h	*93*	*03*	*7C*	*EF*

Furthermore, assume the following initial register values:

- *EBX = 00010430h*
- *ECX = FFFFFFF8h*
- *ESI = 00010440h*
- *EDI = 00000003h*

a. *(10 points) The table above shows the address range 10430h-1044Bh. List the starting addresses for every aligned word in this range.*

For aligned data, the address must be divisible by the number of bytes being accessed. Since a word is 2 bytes, aligned word addresses must be divisible by 2:

10430h, 10432h, 10434h, 10436h, 10438h, 1043Ah, 1043Ch, 1043Eh, 10440h, 10442h, 10444h, 10446h, 10448h, 1044Ah

b. *(10 points) What is the result of the instruction MOV AL, [1043Fh]? What is the decimal value of the data transferred in this instruction?*

The byte at address 1043Fh is the last byte in the line starting with 1043Ch, which is **99h**.

The question does not clearly indicate if these data should be treated as signed or unsigned integers, so either interpretation is acceptable. As an unsigned value, this byte has the decimal value (9 x 16) + (9 x 1) = 144 + 9 = **153**. As a signed value, this byte is negative (-99h = 67h) and has the decimal value –[(6 x 16) + (7 x 1)] = -[96 + 7] = **-103.**

c. *(10 points) What is the result of the instruction MOV DX, [EBX+**0013**h]? Is this memory access aligned? (Originally, constant offset was written as 0043h.)*

This instruction loads a word into DX starting at address EBX+0013h = 10430h + 0013h = 10443h. Since x86 data are little-endian, that word is **3433h**.

This access is **not aligned**, since the starting address of the word (10443h) is not divisible by 2, the number of bytes being accessed.

d. *(10 points) What is the result of the instruction MOV EDX, [ESI+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 ESI+ECX = 10440h + FFFFFFF8h. Note that, as a signed two's complement integer, FFFFFFF8h = -8, so the address being accessed is 10440h – 8 = 10438h. The double word at that address is **2B151F18h.**

This access **is aligned**, because the starting address of the double word (10438h) is divisible by 4, the number of bytes being accessed.

e. *(10 points) What is the result of the instruction MOV [EBX+4*EDI], AX, if EAX = 5599AA11h? 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 EBX+4*EDI = 10430h + 4*00000003h = 10430h + 0000000Ch = 1043Ch. Those bytes are the lowest two bytes of register EAX (AA11h); since the least significant byte is stored first, the result is as shown below, with the changed bytes underlined:

Address	Lo			Hi
1043Ch	<u>**11**</u>	<u>**AA**</u>	10	99

This access is aligned, since 1043Ch is divisible by 2, the number of bytes being accessed.