# Assignment 1

Name: Junchen Li

Student ID: 301385486

a. Convert each of the **unsigned** decimal values below into its corresponding binary value (w = 8), then convert the binary value into its corresponding hexadecimal value.

I. $157_{10}$

II. $248_{10}$

b. Convert each of the **signed** decimal values below into its corresponding **two's complement** binary value (w = 8), then convert the binary value into its corresponding hexadecimal value.

I. $123_{10}$

III. $-74_{10}$

c. Interpret each of the binary values below first as an **unsigned** decimal value, then as a **signed** decimal value (using the **two's complement** encoding scheme).

I. $11101001_2$

II. $10010110_2$

CMPT 295 – Spring 2020

d. Convert $\mathbf{247_{10}}$ into a **signed** value directly, without converting it first to its corresponding binary value (w = 8).

e. Convert $\mathbf{-152_{10}}$ into a **unsigned** value directly, without converting it first to a binary number (w = 8).

2. [6 marks] Unsigned and signed arithmetic operations and overflow

For **a.** below, convert each of the operands (**unsigned** decimal values) into its corresponding binary value (w = 8).

For **b.** below, convert each of the operands (**signed** decimal values) into its corresponding **two's complement** binary value (w = 8).

For **a.** and **b.** below, perform both the decimal addition and the binary addition and indicate the **true sum** and the **actual sum** and whether they are the same or different.

For the binary addition, clearly label all **carry in bits** (by using the label "carry in") and the **carry out bit** (by using the label "carry out").

Finally, indicate whether or not an overflow occurred (for **signed** values, specify whether the overflow is positive or negative). If an overflow occurred, explain how addition overflow can be detected ...

1. at the bit level, and

2. using the decimal operands.

**a. Unsigned addition:**

**I.** $\mathbf{74_{10} + 63_{10}}$

**II.** $\mathbf{123_{10} + 157_{10}}$

**b. Signed (two's complement) addition:**

**I.** $\mathbf{28}_{10} \mathbf{+ -74_{10}}$

**II.** $\mathbf{-117}_{10}\mathbf{+ 126}_{10}$

**III.** $\mathbf{74}_{10} \mathbf{+ 63}_{10}$

**IV.** $\mathbf{-119}_{10} \mathbf{+ -105}_{10}$

**1.**

a. I. $157_{10}$ $\underline{10011101_2}$ $\underline{0X9D}$

II. $248_{10}$ $\underline{11111000_2}$ $\underline{0XF8}$

b. I. $123_{10}$ $\underline{01111011_2}$ $\underline{0X7B}$

III. $-74_{10}$ $\underline{10110110_2}$ $\underline{0XB6}$

c. I. $11101001_2$ For the unsigned decimal values it is $\underline{233_{10}}$ and for the signed decimal it is $\underline{-23_{10}}$

II. $10010110_2$ For the unsigned decimal values it is $\underline{150_{10}}$ and for the signed decimal it is $\underline{-106_{10}}$

d. $247-2^8=-9_{10}$

e. $-112+2^8=144_{10}$

2.

a.

I. $74_{10}$ + $63_{10}$ = $137_{10}$ (true sum and actual sum are same)

1111111 → carry in (front six bits)

$01001010_2 + 00111111_2 = 10001001_2$ $\underline{\text{not an overflow occurred}}$

II. $123_{10}$ + $157_{10}$ = $280_{10}$ (true sum) ≠ 280-2^8=280-256=24(actual sum)

carry out 1111111 → carry in (front six bits)

$01111011_2 + 10011101_2 = (1)00011000_2$ $\underline{\text{overflow occurred,}}$ the overflow is positive. For the bit level the word size is eight however, we need use ten bits to show the whole decimal. The range for unsigned decimal is $0\sim2^w-1$ (when w=8 the biggest number it can get is 255) the true sum is 280>255 so it must have bit overflow occurred.

b. I. $28_{10}$ + $-74_{10}$ = $-46_{10}$ (true sum and actual sum are same)

1111 → carry in (from second to fifth)

$00011100_2 + 10110110_2 = 11010010_2$ not an overflow occurred

II. $-117_{10}$ + $126_{10}$ = $9_{10}$ (true sum) and actual sum -247)

carry out ← 111111 → carry in (from first to sixth)

$10001011_2 + 01111110_2 = (1)00001001_2$ an overflow occurred, the overflow is negative.

III. $74_{10}$ + $63_{10}$ = $137_{10}$ (true sum) and actual sum is -119

carry out ← 1111111 carry in (from first to sixth)

$01001010_2 + 00111111_2 = (0)10001001_2$ overflow occurred, the overflow is positive. For the bit level the word size is eight however, we need use nine bits to show the whole decimal. The range for signed decimal is $-2^{w-1}$~$2^{w-1}$-1 (when w=8 the biggest number it can get is 127) the true sum is 137>127 so it must have overflow occurred. Or it could be changed to negative decimal.

IV. $-119_{10}$ + $-105_{10}$ = $-224_{10}$ (true sum) ≠ -224+2^8=32(actual sum)

carry out ← 11111 → carry in (from third to seventh)

$10001001_2 + 10010111_2 = (1)00100000_2$ overflow occurred, the overflow is negative. For the bit level the word size is eight however, we need use ten bits to show the whole decimal. The range for signed decimal is $-2^{w-1}$~$2^{w-1}$-1 (when w=8 the biggest number it can get is -128) the true sum is -224<-128 so it must have bit overflow occurred.

For the Assn1_Q3, my CSIL computer is using a little endian. For example, when I run the code for the decimal 12345, it gives me the answer: 39 30 00 00. By the definition of little endian, the lower Address stores the lower value, higher address stores the higher value. The output is exactly what the definition describes.

Another example: 2550 f6 09 00 00

3450 7a 0d 00 00

22500 e4 57 00 00 ...