Dalhousie University Faculty of Computer Science
Introduction to Computer Organization with Assembly
Assignment 2 Due: ~~3 Feb 2021, 9:00am~~ 4 Feb 11:00pm

Total [**30**]

(1) (a) [**4**] In class, we examined an algorithm to generate the 2'sC of an integer (flip all bits; add 1). We also saw how any number in binary can be directly represented in hex. Work out an algorithm that takes a number in hex and directly obtains its 2'sC representation in hex (*e.g.* $0xBEEF \rightarrow 0x4111$). Justify your answer - *i.e.* give reason(s) why your algorithm works correctly.

(b) [**1+1+1**] The following bit pattern, $0xC0D$ was found in a 12-bit register. What is the decimal number represented if it is interpreted as: *(i)* unsigned binary; *(ii)* sign-magnitude; *(iii)* Two's complement?

(2) [**8**] This exercise is about the bit-wise operators in C. Complete each function skeleton using only straight-line code (i.e., no loops, conditionals, or function calls) and limited of C arithmetic and logical C operators. Specifically, you are only allowed to use the following eight operators: ! $\sim, \&, \hat{|}, +$ <<>>. A specific problem may restrict the list further: For example, write a function to compute the bitwise `xor` of x and y, *only* using `& |, ~`

```
int bitXor(int x, int y)
{ return ((x&~y) | (~x & y));}
```

(a)
```
/* bitAnd:  x&y  Ex:  bitAnd(6, 5) = 4  Legal ops:  ~ and | */

int bitAnd(int x, int y)
{return 2; }
```

(b)
```
/* minusOne:  return -1;  Legal ops:  !  ~, &, ^, + <<>> */

int minus1(void)
{return 2; }
```

(c)
```
/* isPositive:  return 1 if x >= 0, else return 0;  Ex:  isPositive(-1)
= 0  Legal ops:  !  ~, &, ^, + <<>> */

int isPositive(int x)
{ return 2;}
```

(d)
```
/* twoCmax:  return maximum two's complement integer
Legal ops:  !  ~, &, ^, + <<>> */
int twoCmax(void)
{ return 2; }
```

(3) **[15]** Given the code stub in Figure 1 complete `main.c` using bitwise operations to provide the appropriate mapping between the two sets of ASCII characters in Table 3.

- Compile the initial assignment code using:
  `gcc -o Assign01Code Assign01Code.c`
- Assuming files containing the test text of Table 3, deploying the code from Figure 1 will produce an 'output' text file '`Out.txt`':
  `./Assign01Code <Alphabet.txt >Out.txt`
  or, the following will post the output directly to the screen:
  `./Assign01Code <Alphabet.txt`
- A summary of the ASCII character set is available at:
  `https://en.wikipedia.org/wiki/ASCII`
- *Hint:* For the two suggested input texts, use the table of ASCII codes to identify the bitwise mapping between input and outputs.
- Your submitted code will be tested on a third text file to determine the validity of your solution.
- *Important:* Leave the current code stub unchanged. The additional code should appear in the region indicated by the comment field.
- Do not introduce other 'loop' structures

Input file (`Alphabet.txt`)	Output file (`Out.txt`)
abcdefghijklmnopqrstuvwxyz	dhlptx|aeimquy}bfjnrvz~cgk
ABCDEFGHIJKLMNOPQRSTUVWXYZ	DHLPTX\AEIMQUY]BFJNRVZ^NCGK

TABLE 1. Source (I/P) and Target (O/P) texts assuming ASCII encoded characters.

```c
#include <stdio.h>

int main(void)
{
  int inChar, outChar; // note that getchar and putchar use int type

  while ((inChar = getchar()) != EOF)
    {
        // Your bitwise code appears here...


        // uncomment the following to return the input character as the output
        // outChar = inChar;

        // last instruction in while loop
        putchar(outChar);

    }

  return 0;
}
```

FIGURE 1. Initial code snippet (Assign01Code.c). C library function `getchar(·)` returns an integer expressing the ASCII encoded character from the standard input (see note regarding redirection). The `putchar(·)` function converts the specified integer into the corresponding ASCII character and posts it to the standard output (again subject to any redirection). The only code that you need to add should fall between the two sets of comment characters.