Assignmen 1: C calling convention, Decimal to Hexadecimal

Submissions which deviate from the instructions will not be graded!

Please make your output exactly as in the examples below.
Otherwise automatic scripts would fail on your assignment and you will lose your points for this.

Assignment Description

This assignment contains two tasks:

Task 1 - C calling convention

You will implement entirely on your own (unlike task 2, we provide no files for this task). You may use C library functions fgets, printf, sscanf. Your program should be composed of two files, one written in C and one written in assembly language, according to the instructions below.

Write a C program that contains all your C code: function 'main(…)' performs the following steps:

1. Prompts for and reads two integer (32 bits) numbers x and y in decimal base from the user (you may use fgets() and sscanf()).
2. You may assume that the input is always valid.
3. Calls 'void assFunc(int x, int y)' written in assembly language with the above integers as arguments

'void assFunc(int x, int y)' performs the following steps:

1. Calls 'char c_checkValidity(int x, int y)' written in C to check the relation between x and y
2. If c_checkValidity(int x, int y) returns 1, calculate `z=x-y`, and print z in decimal base, using printf with '%d' format
3. Otherwise, calculate `z=x+y`, and prints z in decimal base, using printf with '%d' format

'char c_checkValidity(int x, int y)' performs the following:

1. Returns 1 if `x>=y`
2. Otherwise, returns 0.

Examples:

```
> task1Assignment1.out
5
1
4


> task1Assignment1.out
1
5
6
```

Task 2 - Converting unsigned numbers in decimal base to unsigned number in hexadecimal base

(Note: this task requires multiplication, and optionally division, which will be covered in a practical session after Passover.)

We provide you a simple program in C that inputs a string (as a line) from the user in a loop, and calls the convertor(…) function **that you need to implement in assembly language**. The main loop is terminated when user enters the "q" input string.

The input string contains ASCII digits, representing an unsigned decimal number in the range 0 to 2^32-1, i.e. the input number is to be seen as an ASCII string encoding an unsigned integer in decimal format.

- convertor(…) receives a pointer to the beginning of a null terminated string.
- You may assume that the input string contains only decimal digits (i.e. '0' , '1' , … , '9'), or only 'q' in a case that a user wants to quit the program
- The output string should also be null terminated.
- Note that the input string may contain a newline ('\n') character, make sure you are aware of it and treat it accordingly.
- convertor(…) should print the resulting hexadecimal representation as a **string**. That is, in your assembly code you should call the printf function with the '%s' format.

Examples:

```
> task2Assignment1.out
479
1DF
2545
9F1
q
```

Character conversion will **not** be in place this time.
You should write the output string into another buffer (provided in the code skeleton) before printing.

Note: you **may not** use any available function that automatically does the conversion, such as printf, to do the conversion. You need to compute the output digits yourself.

What We Provide

The attached files:

- makefile
- main_task2Assignment1.c
- asm_task2Assignment1.s

For task 2, you need to edit both assembly and C provided files.
Note that the provided C file does not contain the needed loop. Ensure you change the code appropriately.

For task 1, you are required to create two new files: main_task1Assignment1.c and asm_task1Assignment1.s. These files will contain the code for task 2 as described above. You are also required to modify the supplied Makefile so that main_task1Assignment1.c and asm_task1Assignment1.s will be compiled in a manner similar to task 2.

In order to compile the files for the first task , you only need to use the 'make' utility. In order to compile the files for the second task , you will have to modify the 'Makefile'. You can take example from the way task 2 is compiled.

Submission Instructions

You have to submit a single zip file, **ID1_ID2.zip** , includes 4 files: **asm_task1Assignment1.s**, **asm_task2Assignment1.s**, **main_task1Assignment1.c** and **main_task2Assignment1.c** . Do not add directory structure to the zip file! Make sure you follow the coding and submission instructions correctly (print exactly as requested).

Good Luck!