Assignmen 1: ASCII and Hexadecimal, C calling convention

Submissions which deviate from these 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 some credit for this.

Assignment Description

This assignment has two parts:

Part 1 - Converting number in hexadecimal base to ASCII characters

We provide you a simple program in C that inputs a string (as a line) from the user, and then calls a function my_func that receives a pointer to the beginning of a null terminated string.
You need to implement my_func in assembly language.Assume that the input string contains STRICTLY hexadecimal digits (assume input correctness), and is of even length, not including the null terminator.
Your function should read each two digits (string characters) at a time, and insert into an output string a character corresponding to their ASCII value (in hexadecimal representation). The output string should also be null terminated. my_func should print the resulting string.

You may assume the domain of characters (in hexadecimal asciitable values) is: 20 (space) to 7E (~).

Note that we are providing a skeleton to use in writing the assembly language code below.
You are not allowed make any changes to the main1.c file.

Examples:

- given "33343536" as the input string (the quotes are not part of the string), your program should output the string: "3456"
  ```
  > task1.bin
  33343536
  3456
  ```
- given "4c656d6f6e" as the input string (the quotes are not part of the string), your program should output the string: "Lemon"
  ```
  > task1.bin
  ```

```
4c656d6f6e
Lemon
```

- given "373E393D46616C7365" as the input string (the quotes are not part of the string), your program should output the string: "7>9=False"

  ```
  > task1.bin
  373E393D46616C7365
  7>9=False
  ```

Character conversion will **NOT** be in place this time. You are to write the output into another buffer (provided in the code) before printing.

Part 2 - C calling convention

In this part you need to implement entirely on your own. You are to create a C file and an assembly language file as specified in the submission instructions below.

You are to write a C program that:

1. Reads two integer (32 bits) numbers x and k (in decimal) from the user.
2. Call a function ‘calc_div(int x, int k)’ written in ASSEMBLY language with the above integers as arguments.

The assembly language function ‘calc_div(int x, int k)' performs the following steps:

1. Calls a C function 'check' (as defined below) to check if the numbers are legal.
2. If x and k are legal, computes z by calculating x div 2^k. Division does not consider the reminder and the result is integer (e.g. 5 div 2 = 2).
3. Calls printf to print the result z in decimal.
4. Either x or k, or both are not legal, calls printf to print "x or k, or both are off range".

The C function 'check(int x, int k)' performs:

1. Gets 2 arguments: integers x and k
2. Returns false (0) if x is negative
3. Returns false (0) if k is non-positive or greater than 31
4. Returns true (1) otherwise

Example:

For x equals 5 and k equals 1, the result is 5 div 2^1 = 2.

```
> task2.bin
5
1
2
```

What We Provide

The attached files:

- Makefile
- main1_ass1.c
- task1_ass1.s

For Part 1, you only need to edit task1.s in the location indicated in the file, as in Assignment 0. For Part 2, you are required to create 2 new files: main2.c and task2.s. These files will contain the code for Part 2 as described above. You are also required to modify the supplied Makefile so that main2.c and task2.s will be compiled in a manner similar to Part 1.

In order to compile the files for the first part, you only need to use the 'make' utility. In order to compile the files for the second part, you will have to modify the 'Makefile'. You can take example from the way Part 1 is compiled. Make sure the resulting files are in the appropriate directories as described for the first part. Also, make sure the name of the executable for the second part is 'task2.bin'.

Submission Instructions

You are to submit a single zip file, **ID1_ID2.zip** , includes 3 files: **task1.s**, **main2.c** and **task2.s** . Do not add new directory structure to the zip file! Make sure you follow the coding and submission instructions correctly (print exactly as requested).

Good Luck!