LAB1, Bit manipulation and data representation

Goal

Learn about bit manipulations and data types in C. You will learn (refresh your knowledge) how to use and combine the bitwise operators `&`, `|`, `~`, `^`, `<<`, `>>` and how they behave on signed and unsigned types. You will also use the very common hexadecimal representation of numbers, and write/modify some simple C-functions and preprocessor macros.

Given files

Assuming you follow the instruction below you will have given files in ~/`TDDI11/lab1`. Your modifications goes to `main.c`. To solve the task, and verify your solution, you must run and compare to the reference solution. The reference solution exists as the precompiled program `lab1_ref` in the folder (you are allowed disassemble and reverse engineer back to C-code, but it's probably more work...).

Assignment

You may follow the instructions in next chapter if you want to, but this first lab do not require all setup. Open a terminal by right-clicking the desktop and find it in the menu. Then make the `lab1` directory (folder) the current working directory and open the given main program in `emacs`:

```
cp -r /home/TDDI11/lab/skel ~/TDDI11
cd ~/TDDI11/lab1
emacs main.c &
```

The final `&` is important, if you forget the `&` the terminal will wait for `emacs` to terminate, preventing you from entering further commands. Run the reference program and try with various input to get a feeling for how it shall work. Your program must generate `EXACTLY` the same output given the same input.
You are supposed to hit some "snags" and apply problem solving / manual reading to get past them. To compile you use the normal C-compiler from the terminal:

```
gcc -Wall -Wextra -std=c99 -pedantic -g main.c -o main
```

Deliverables

Show your code to the assistant and demonstrate your application. Be prepared to answer questions.

Reference

Use your favorite C-book or webpage to find out how bit manipulation works, how integers and character are represented, how hexadecimal representation works, and what BCD is. Quick reference that should be enough for most are given as comments in the given code.

The given code includes all `printf` formatting flags you need, and it should not be modified, but to know what it all means you can read the manual pages for `printf` in the terminal:

```
man -s3c printf
```

To find information about compiler flags you can use the manual page for gcc: `man gcc`