EEEE-521-621-Lab4

1) From mycourses download on your desktop this pdf file.
2) The goal of this lab is to create an assembler for your instruction set.
3) The output of your assembler should be the ***.mif** initialization file containing the user program.
4) You can use any high level programming level you like. The following instructions pertain to those who choose to use C. Obviously, the given code could also be compiled with a C++ compiler.
5) You can use any compiler you have access to, or download the DevC++ installation file from mycourses and install it on your own computer. By downloading it from mycourses you agree to the terms of the license, as if you had downloaded it from the original source!
6) I am providing as a DevC++ project with all input, intermediate, output, code, and header files.
7) This assembler works for a slightly different instruction set and instruction word format than yours. However, you should be able to modify it to fit your instruction set architecture.
8) The code is commented and uses "textbook" string processing functions.
9) The project can be opened by opening the ***.dev** file.
10) Open the project. Then "compile & run". In the console window that opens automatically, enter for the name of the source file **dxp.txt**. You can use any other name or extension for yours later. Once you enter this, every processing step is echoed back to the terminal. At the present time the code is intentionally very verbose.
11) Use a text editor, such as Notepad++, to open the **dir.txt, const.txt, and code.txt** files.
12) To prove that your assembler is working, you have to be able to compile the code used in labs 2 and 3. **For this lab your assembler should be able to assemble at least all manipulation instructions.**
13) You can come up with your own assembly syntax conventions or follow the ones provided in **dxp.txt**. Remember: the assembly code should be human friendly.
14) If you use the DevC++ development environment, archive your project folder, including any intermediately generated output files.
15) If you are using another compiler/IDE, provide information about these in your report. Include your source code files and meaningful screen shots to proof that your assembler is working.
16) Ultimately, compile a version for release. This should be a standalone executable file. The TA will use this to verify your assembler.
17) If time permits, show your working assembler to your TA. Note: only the TA will assign a grade.
18) Archive your project.
19) Write your report and upload it along with your archived project(s) in the dropbox on mycourses, as described in the lab policy.
20) This concludes this week's lab.
21) **Grading 521 AND 621**:
    a. 2 points for each correct and complete assembled instruction ➔ 2 x 15 lines of code = 30 points.