## Project 1 Description

The first project of this course is to write an assembler for MIPS assembly language. Basically, your program takes an input of MIPS assembly language file, assembles it, then generates a simple output file which is "executable". In other words, the output file should contain machine code for each instruction. The input format and output format would be specified later in this instruction. The file should be able to run on the MIPS Simulator, which is the second project of this course. You may write your code in **ONLY C or C++**.

The output should be a .txt file which contains the machine code of the corresponding input .asm file. I will provide you with a few input files and their corresponding output, so that you can test your program. Your code would be tested on a **MacOS system** or **Ubuntu 16.04** system, whichever you specify. The compiler you need to use are **gcc or g++**, for C and C++ respectively.

## Environment:

### For Mac Users:

Check whether you have C/C++ Language installed on your mac. Open up your terminal, then type the following command: "clang --version"

```
Kefans-MacBook-Pro-2:~ kefanshuai$ clang --version
Apple clang version 11.0.0 (clang-1100.0.33.17)
Target: x86_64-apple-darwin19.2.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

If your Mac shows the version of C Language, that means you already have the environment to write this program. Otherwise, you will have to install Xcode on your Mac. Go to App Store on your Mac and search for Xcode.

After doing that, check again to see if you have the environment for C/C++. If you do, you can now begin to write your code.

### For Windows Users:

There are multiple ways to obtain a Linux system on Windows. I would recommend using VMWare Workstation Pro. You can download a Ubuntu 16.04 image via http://releases.ubuntu.com/16.04/. You can simply deploy it in VMWare Workstation.

If you would like to write your program in Visual Studio or other IDE, you may. However, make sure you test your code on the required environment before you turn in your code.

## Logic of the Project

There are many ways to achieve the goal of this project, I will provide one for you to think about in the rest of this section.

The basic idea is to break this project into two parts: the first part is to scan through the file, and find out where are the labels, such as “main”, or “loop”. For each label you find, you will need to save the data of what is the name of the label, and what’s its corresponding memory address. Then you will need to define a proper data structure to store them.

Then, there is the second phase, where you will need to use the information generated in the first part (stored in the data structure you defined) to output the final answer. More specifically, you can write a loop in which each iteration will process one line of the input file. For each line you read, you will need to identify which instruction it is, what kind of instruction it is (R, I or J), then generate the machine code corresponding to that information.

## List of MIPS Instructions Your Code Needs to Support:

We will follow the appendix A of the text book, you need to support all the instructions from A51 to A71 that are **NOT PSEUDO-INSTRUCTION and NOT CO-PROCESSOR INSTRUCTION**. Also, I would recommend you group them according to their R, I, J format, for the ease of translation.

Also, the register names and their corresponding register numbers are attached to the end of this document.

## How to Properly Write Your Code:

You can use any IDE you like. However, make sure your code is executable in the required environment. Make sure all your files related to the program are under the same directory, including the testing input and output. Make sure your code is compilable using command line in terminal. For example: “gcc file1.c file2.c file3.c -o assembler”. This would produce an executable named assembler. Then, the executable should be able to be executed by the command “./assembler inputfile.asm outputfile.txt”, which user can specify which input file is used and what output file name is used. Your program should then produce a file with the output file name specified containing the machine code corresponded to the input file.

After you obtained an output text file, you may compare the result with the given expected output text file. One way of doing this is by using the command “diff file1.txt file2.txt”.

## Project Report

You will need to write a report to tell me:

1. Your understandings of this project (what this project is doing)
2. How did you do it. More specifically, when you explain your way of doing it, make sure you start from the big picture idea. Explain the high level logic first, then go into details such as what kind of data structure is defined, etc. An example of explaining the big picture idea can be found in "Logic of the Project" part of this file. I did not explain why I did it this way, which you should in your report.

Your report should be typed, Times New Roman, 12pt, with your name on the front page. Your report should be in **pdf** file, I would not accept any other format.

## Discussion Board

I will set up a discussion board on Blackboard after this is released. If you have any questions regarding this project, please post them on discussion board. I would check the discussion board daily after this project is released and answer questions. If you know the answers to some questions you see, you are more than welcomed to answer them for your fellow students. In most cases, I would not answer questions in the WeChat group, as last year students do not go through the chat history before asking, thus, repeating questions are asked a lot. This is not efficient, so I decided to use the discussion board for Q&As.

## How to Turn in The Project

Submit all of your source code along with the report to Project 1 on Blackboard. Before you turn in, put them all in a folder with your student ID only, and compress the folder. The project is due on 3/2/2020 at 11:59 pm. No late submissions are accepted.

## Academical Dishonesty

According to the University's policies, plagiarism is prohibited. Anyone who was caught cheating will obtain an automatic zero on this project. Please do not copy your fellow students' codes, or share your code with other students. Automatic zero goes to both sides.

## Grading

Project body - 80%

each wrongly translated instruction will result in 3% deduction

Coding style - 10%

clean and neat coding style and comments - 5%

write header files nicely and correctly - 5%

Project report -10%

| Register name | Number | Usage |
|---|---|---|
| $zero | 0 | constant 0 |
| $at | 1 | reserved for assembler |
| $v0 | 2 | expression evaluation and results of a function |
| $v1 | 3 | expression evaluation and results of a function |
| $a0 | 4 | argument 1 |
| $a1 | 5 | argument 2 |
| $a2 | 6 | argument 3 |
| $a3 | 7 | argument 4 |
| $t0 | 8 | temporary (not preserved across call) |
| $t1 | 9 | temporary (not preserved across call) |
| $t2 | 10 | temporary (not preserved across call) |
| $t3 | 11 | temporary (not preserved across call) |
| $t4 | 12 | temporary (not preserved across call) |
| $t5 | 13 | temporary (not preserved across call) |
| $t6 | 14 | temporary (not preserved across call) |
| $t7 | 15 | temporary (not preserved across call) |
| $s0 | 16 | saved temporary (preserved across call) |
| $s1 | 17 | saved temporary (preserved across call) |
| $s2 | 18 | saved temporary (preserved across call) |
| $s3 | 19 | saved temporary (preserved across call) |
| $s4 | 20 | saved temporary (preserved across call) |
| $s5 | 21 | saved temporary (preserved across call) |
| $s6 | 22 | saved temporary (preserved across call) |
| $s7 | 23 | saved temporary (preserved across call) |
| $t8 | 24 | temporary (not preserved across call) |
| $t9 | 25 | temporary (not preserved across call) |
| $k0 | 26 | reserved for OS kernel |
| $k1 | 27 | reserved for OS kernel |
| $gp | 28 | pointer to global area |
| $sp | 29 | stack pointer |
| $fp | 30 | frame pointer |
| $ra | 31 | return address (used by function call) |