

CSC3050

Project 1 Report

Bernaldy Jullian

119010520

## General Idea

The first thing that needs to be done is to tokenize each MIPS function. This will make each of the functions (which is in the .text part) be able to be translated into binary numbers (machine codes). To do this, the Label class is created, which can store the address and other things such as the name, data type, and the content of the functions. The Label class is also used to tokenize the .data part. However, I do not understand how to allocate the memory and how the program can execute the machine code.

## Functions

- `class Label`

The main use of the Label class is to store the address, name, data type of .data and .text

- `string makeR_type(string instruction, string rd, string rs, string r t, string shamt, string funct)`

Transforms the R-format into machine code.

- `string makeI_type(string instruction, string op, string rt, string r s, string immediate)`

Transforms the I-format into machine code.

- `string makeJ_type(string op, string address)`

Transforms the J-format into machine code

- `string trim(string line)`

Removes any spaces at the start or at the end of a line (used in string formatting)

- `bool operator==(Label label, string string)`
- `bool operator==(string string, Label label)`

Overloading operators for == for the label class.

- `Remove_comments`

Self-explanatory. Removes the comments from the input.

- Void firstParse(string iss)

Stores the labels.

- Stringstream secondParse(stringstream & is)

The function that assembles the input.

- Int assemble (string filename)

Open the file and uses the other functions to assemble.

- Int main()

Main function.

## Sample Output

```
00111100000000010000000001010000
00110100001100000000000000010000
10001110000100000000000000000000
00111100000000010000000001010000
00110100001100110000000000000000
00100000000000100000000000001001
00100000000001000000000000010000
00000000000000000000000000001100
00000000000000101000100000100001
0000000000000010001000000100001
0000000000000010001000000100001
00000000000100110010100000100001
00000000000100000011000000100001
0000110000010000000000000100110
0010000000000010000000000001111
0010000000000100000000000000001
00000000000100010010100000100001
00000000000100000011000000100001
00000000000000000000000000001100
0010000000000100000000000001010
00000000000000000000000000001100
001000001100111111111111111100
000110011110000000000000000110
1000110010101000000000000000000
1010110010001000000000000000000
001000001100011011111111111100
0010010010100101000000000000100
0010010010000100000000000000100
00001000000100000000000000010110
000100001100000000000000000110
1001000010101000000000000000000
1010000010001000000000000000000
001000001100011011111111111111
0010010010100101000000000000001
0010010010000100000000000000001
00001000000100000000000000011110
0000001111100000000000000001000
001000001100111111111111111100
0001100111100000111111111110110
0000000010000101110000000100110
```