Report:

The goal of this project was to build a MIPS ISA for the core instructions. The code is written in C++ and uses a combination of e data types to store information that was parsed through the input file

Data Types:

- Hash map: To store label names and corresponding locations they point to. (Line of label+1)
- String: stores the first element of each line which could be either an opcode or label
- Vector: stores the 3 elements that appear after the string opcode. It can be a combination of: rs, rt, rd / rt, imm, rs / rd, rt, sh.
  In certain instructions where 3 elements are not needed after the opcode is read (jr, lui etc.) empty elements are pushed into vector.

Parsing:

Parsing is done twice. Once for creating HashMap of labels and their locations and the second time for assembling the instructions. Using strtok, I pass each line of the input file as a char* array. To convert String to writable char* array, I use a vector. Strtok tokenizes input string according to delimiters.

Assembling Hex instruction:

To determine the type of instruction I first read the opcode string in an if statement. According to the opcode, I determine which element of vector goes into appropriate register bits.

I use a function to convert valid registers in the reference card to appropriate binary numbers. Immediates were converting from string to int, and then to sign extended binary.

Error Checking:

If an opcode string did not match with any of the if statements and also did not have a ":" to show that it's a label, then I show error at that line. Error checking is robust in that it checks for registers in wrong locations. That is if an instruction accepts two regs and an immediate, if an extra register is passed or an invalid immediate instruction is passed then it is flagged as false.

Output:

The output file is written to line by line. If any of the instructions are not assembled, then the output file is removed.