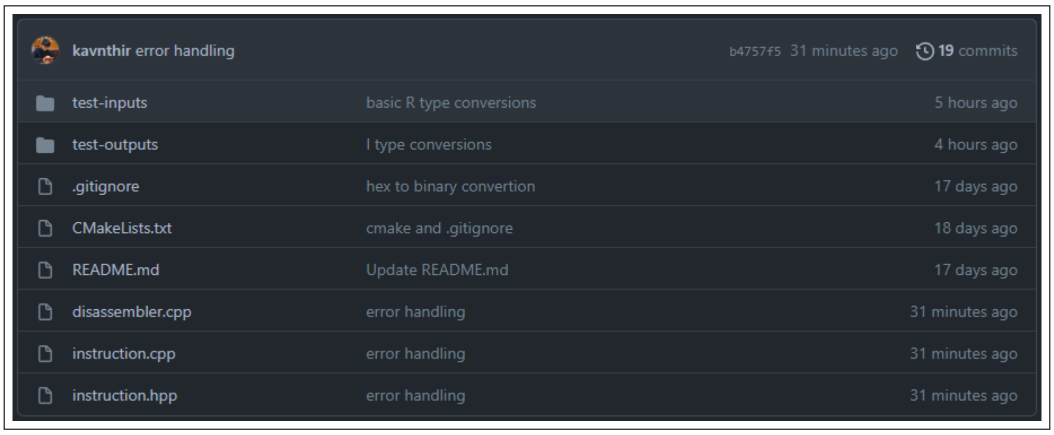
Project Approach

When looking at this project, I realized it would be alot of string manipulation, this is because storing values as hex and decimal is easiest done in strings. Because of this I decided to break my program up into a class that handles converting each instruction, and a main function that takes each converted instruction and compiles it into an output file. This easily allowed me to abstract that radix conversions into separate functions. For any number to string conversions, I used the `std::map` data structure, this reduces the amount of code I need to write to look for matching pairs since the data structure takes care of that internally, it also gives me an easy way to write the conversion pairs in an organized manner as I did in the private variables of my class.

To handle the issue of having to insert labels in after the running through one time, I stored disassembled instruction in a `std::vector<std::string>` such that later I could insert the labels into the correct position, and at the very end I would print the vector out to a file. To store the locations of the labels, I had a `std::map<int,std::string>` which mapped a line number to a Label, the information from which was used in a for loop later to insert the labels into the correct position in the file.



kavnthir error handling b4757f5 31 minutes ago 19 commits

test-inputs	basic R type conversions	5 hours ago
test-outputs	I type conversions	4 hours ago
.gitignore	hex to binary convertion	17 days ago
CMakeLists.txt	cmake and .gitignore	18 days ago
README.md	Update README.md	17 days ago
disassembler.cpp	error handling	31 minutes ago
instruction.cpp	error handling	31 minutes ago
instruction.hpp	error handling	31 minutes ago

https://github.com/kavnthir/mips-disassembler

Above is an image of the file structure of the program, a `.cpp` file containing the main function, and two other files holding the class which is used to convert instructions.

Documentation

I used `cmake v3.21.1` to create the executable for this project, and also provided a CMakeLists.txt in the project submission. If for some reason cmake doesn't feel like cooperating, it is just cpp code so you should be able to throw it into a visual studio project and run it.

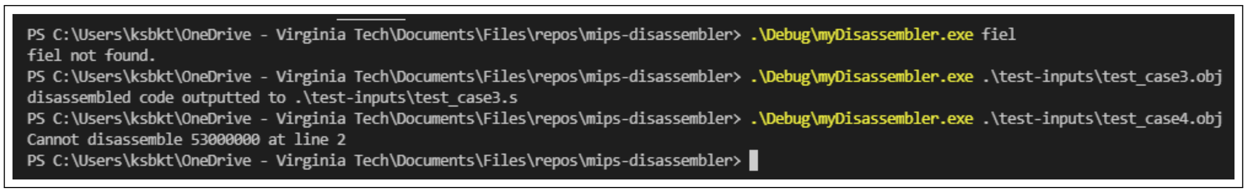
```
# Configuring Cmake
cmake .\CMakeLists.txt

# Building executable with cmake
cmake --build .

# Executing
.\Debug\myDisassembler.exe *.obj
```

To build and launch the program the above instructions can be used.

Running Example



```
PS C:\Users\ksbkt\OneDrive - Virginia Tech\Documents\Files\repos\mips-disassembler> .\Debug\myDisassembler.exe fiel
fiel not found.
PS C:\Users\ksbkt\OneDrive - Virginia Tech\Documents\Files\repos\mips-disassembler> .\Debug\myDisassembler.exe .\test-inputs\test_case3.obj
disassembled code outputted to .\test-inputs\test_case3.s
PS C:\Users\ksbkt\OneDrive - Virginia Tech\Documents\Files\repos\mips-disassembler> .\Debug\myDisassembler.exe .\test-inputs\test_case4.obj
Cannot disassemble 53000000 at line 2
PS C:\Users\ksbkt\OneDrive - Virginia Tech\Documents\Files\repos\mips-disassembler>
```