The code for our scanner and parser is in reader.c, it transforms the binary input of a .gb rom file using a structured array into an input that our c code will be able to run.

Included for an example is Tetris.gb, this file goes over all of the possible inputs that we will be trying to implement. Our final goal is to be able to run Tetris.

Also included is Tetris.asm, which contains the assembly of the Tetris rom as parsed by no$gmb, a gameboy emulator. The line numbers, the hex input and any code that was not a part of the actual rom were stripped out of the file. This allows for a side by side comparison with the correct output.

A few things to note: the header is not a part of our scanner, the input of that is more important for emulators that will run a variety of types of roms. As such, it is stripped out of both our output and the output of the no$gmb emulator. The jr commands rely on the value of the HL register, since we are only scanning it is impossible to know what that value is. The values of our jr commands will be different because of this. Some of the other call and jp commands also use signed values to get a positive or negative value, which will also result in some differences. Finally the logical operators (and, or, xor …) all default to A, however the no$gmb emulator specifically states A in the asm output.

The script to compile and run the reader on our test case of Tetris is found in test.bash

For our grammar, we gave a representation after the language was parsed into assembly so that it makes more sense. A grammar from the binary representation would consist of just the hex values.

We have included a parser that will scan both our output and the output of the no$gmb and display the differences.