Program 2

# Problem Statement

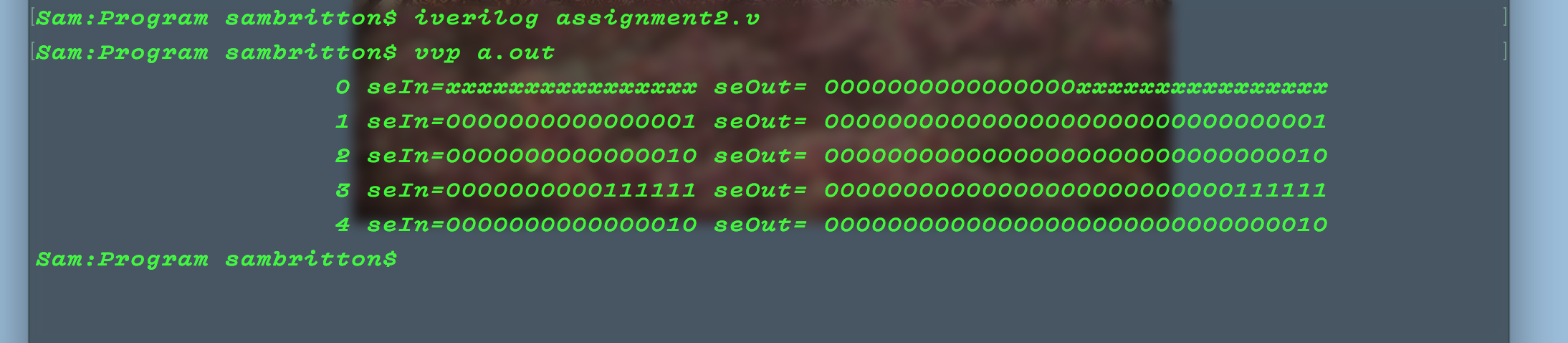
In this section of the overall project, we were tasked with design and implementation of a Verilog program that would emulate the registers, ALU, data memory, and sign extender modules of a MIPS processor. This portion of the project was the second segment in the overall completion of our MIPS simulation.

# Approach to solution

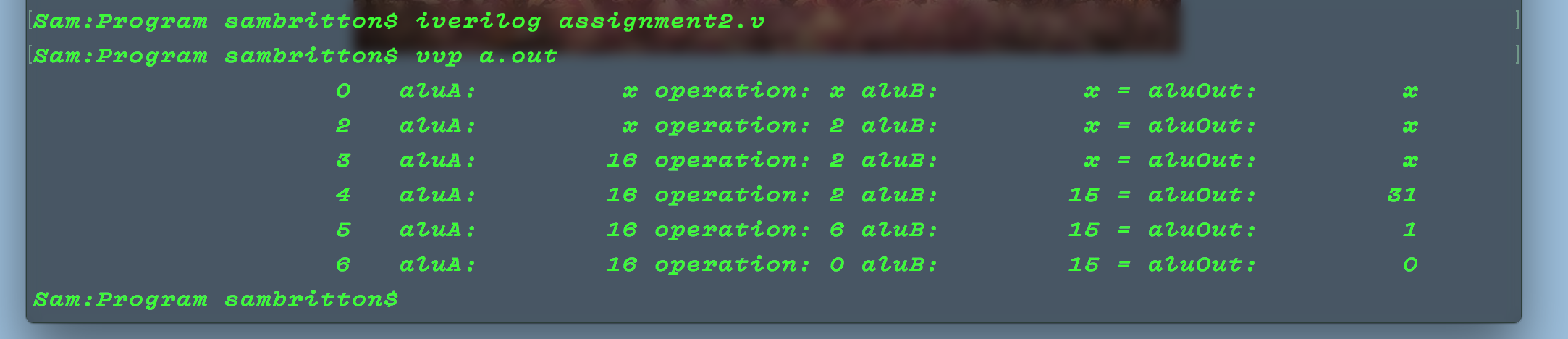
The project was very easily split up into five individual segments. Four of those parts were the actual modules in the design given to us, and the fifth testbench module was needed test each of the four components. Through the lectures and notes given to us through the slides, we were also given samples of modules for an already implemented MIPSALU, RegisterFile, and DataMemory, all written by Dr. Goodrum, and so I was able to utilize this contribution for the final project. Some of these modules were only given as an outline, so there needed to be some adjustments so that it was all syntactically accurate. As each of these modules were added to the file, a new block of code was added to the testbench to check functionality.

# Solution Description

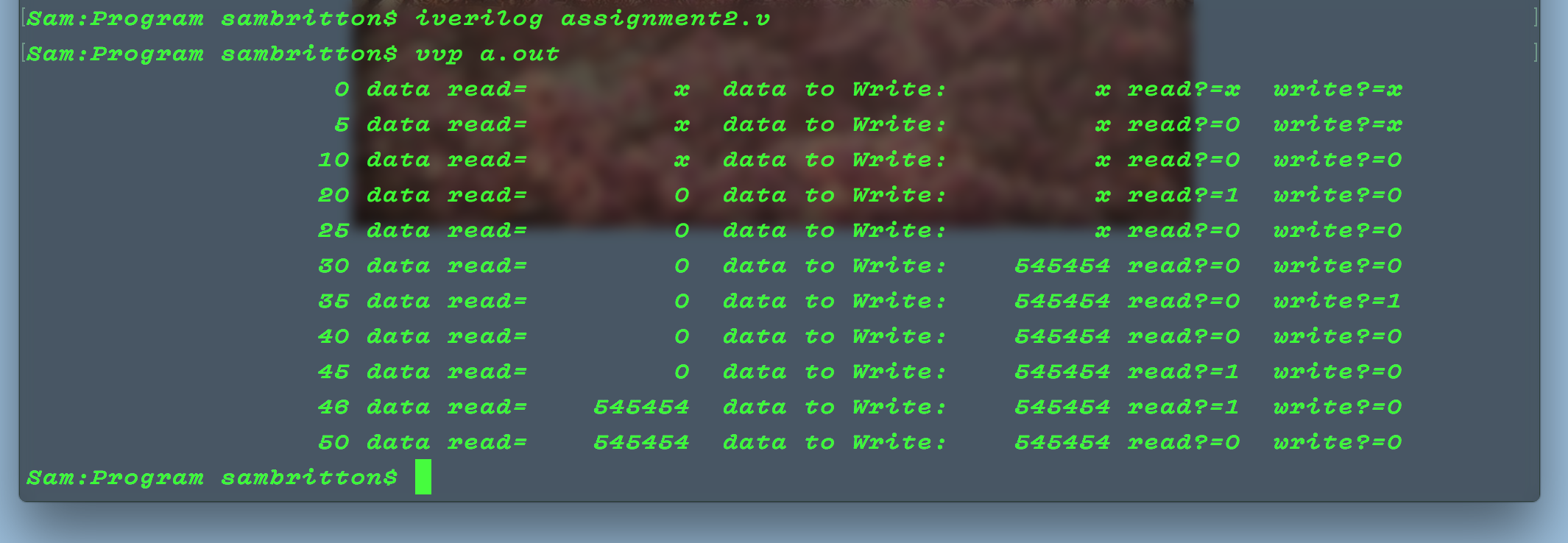
Much like the initial approach, the final solution was split up in to unique sections. Each module was added and then immediately tested to ensure that all the internal functions of the module were working properly.

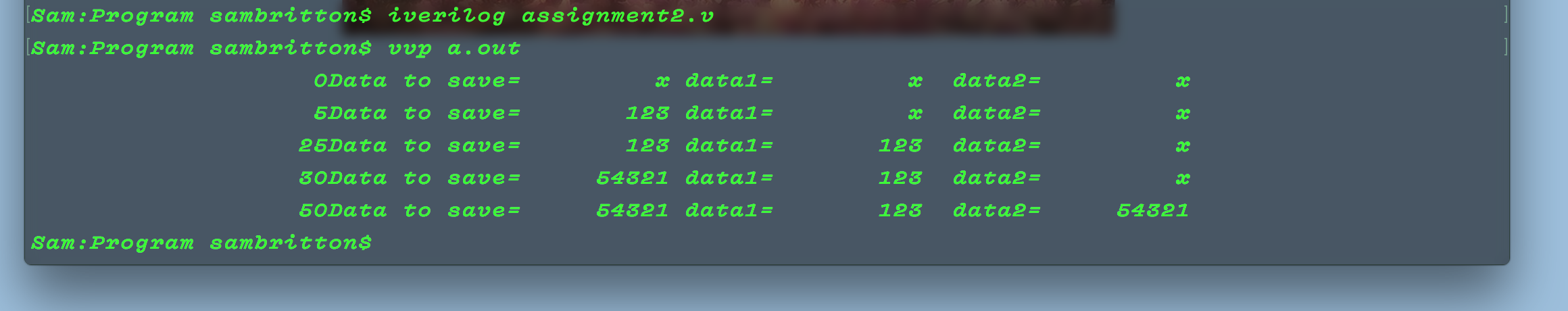


Above is the example from the SignExtender module testing showing that the module takes a 16-bit value and makes it into a 32-bit value.



Next is an example of testing the MIPSALU. This module used the values from aluA, aluB and preforms the operations 2,6, and 0 on them (Add, subtract, bitwise AND).

 Third is the example of testing DataMemory. In here we first read 0 from the memory (All memory values are by default 0). Then we write 545454, and then read that same value accurately.



The last segment was the testing of the Register module. Testing of this module was fairly straightforward. For this, a value was placed to be saved, it was then saved (written), and last read back into the registers by the module. Both register ports were tested and were both working well.