**Register File Module**

The register module for this project was made according to the specifications given, which was a register file composed of sixteen 16-bit registers. Unlike other blocks in our data path, the register file specifications allowed us to create this block using behavioral Verilog. This module is made up of 6 inputs and 2 outputs. The six inputs are composed of: 3 4-bit addresses which point to the 2 registers to be read and the one register to be written to; one 16-bit input which contains the data that is to be written into the specified register; a 1-bit write signal; and finally, the clock. The only 2 outputs of the register file are the 16-bit fields which contain the data from the two specified registers to be read. When the write signal is asserted, the data is written into the specified register at the next positive clock edge.

**Instruction Memory**

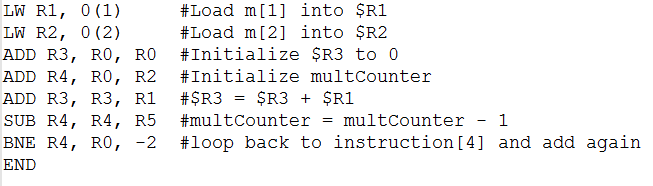
The instruction memory module for this system, like the register file module, was written in behavioral Verilog. Just like the register file, the instruction memory block is made up of sixteen 16-bit memory blocks. This module however, only takes in one input and only has one output. The input is the memory address which is the value of the program counter. The output of the instruction memory is a 16-bit instruction which is stored at a block pointed to by the input address. Each 16-bit instruction in the memory is split up into four 4-bit sections. The highest 4 bits are the opcode, the next two 4-bit sections are the register addresses of the Rs and Rt registers. The lowest 4 bits contain either the address of the Rd register or an offset. Whether it is an address or an offset depends solely on the type of instruction.

**Data Memory**

The data memory module for the processor is made up of 32 16-bit memory blocks. This module is similar to the register file, except that it only has one read output. The data memory module takes a 16-bit address as the input, and outputs a 16-bit value for the given address. This module also has two other inputs, a clock and a write enable signal. If a write signal is asserted, the input write value is written into the specified address at the next positive edge of the clock.

**Explanation of MIPS code (Instructions)**

The main goal of our project was to create a single-cycle RISC processor. In order to test that our instruction set is functional, our objective was to create a program to compute multiplication of two decimal numbers using the given instruction set. The following is the MIPS assembly code for out multiplication program:



The first two instructions are designed to load the two values to be multiplied into registers 1 and 2. We then initialize the result register to 0. In order to implement the multiplication, we will create a counter to have R1 add to R3 for R2 amount of iterations. After every addition, the counter is decremented and the program counter returns to the addition instruction. This loop is continued until the counter is equal to zero, this completes the multiplication and the result is stored in R3.