**Adder.v**

Input: carry in value, 2 32bit values (A and B)

Output: carry out value, 32bit value (sum or difference of A and B)

The Adder.v module was used to either do 32 bit addition or subtraction.

**ALU.v**

Input: clock, 6 bit op code, 6 bit function code, 2 32 bit inputs

Output: 32bit output

The ALU.v module takes output from Splitter and executes the instruction using the pieces from Splitter.v using thirtytwobitadder, thirtytwobitsubtractor, prod, and orop.

**Clock.v**

Input: none

Output: clock

Generates the clock value that most of the modules use

**DataMem.v**

Input: 6 bit opcode, 32 bit instruction, 32bit address, clock

Output: 32 bit output

DataMem.v contains a 256 element array of 2bit values and is used to simulate memory. If the opcode calls for the storage of a value, it will be put into address specified in the input.

**Gates.v**

Input: 2 32bit values

Output: A 32 bit value

Gates.v is an enumeration of the gates used in the Adder.v file

**InstMem**

Input: program counter, clock

Output: 32 bit instruction

InstMem.v stores and outputs the instructions used in the testing of this project. The instructions are stored as an array of 128 32bit elements. Depending on the program counter supplied in the input, a different instruction is loaded.

**PC.v**

Input: 32 bit input which needs to be offset to add to PC is a branch or jump is needed, clock

Output: Program counter

PC.v sets the new program counter. PC will be PC+1 if no branch or jump.

**RegisterFile.v**

Input: readwrite toggle bit, 5bit rs, 5 bit rt, 5 bit rd, 32 bit input value,

Output: 2 32 bit output values

RegisterFile.v is the representation of the 32 32bit registers. If input is 1, the instruction writes to a register. The input value goes into rd.

**SignExtendShift.v**

Input: 16bit value that that needs to be shifted

Output: 32bit value that is shifted

Simple bit shifting of an address to match 32bit format

**Splitter.v**

Input: 32 bit instruction

Output: 6bit opcode, 6 bit function code, 5 bit rs, 5 bit rt, 5bit rd, 16 bit address

Splitter.v breaks up a 32bit instruction into the proper parts depending on opcode.

**tb.v**

Test bench for the entire program.