ALU.v

* Inputs
  + 2, 32 bit values from two files?
  + 3-bit ALUControl register that has the output from the ALU control unit.
    - Dictates what operation will happen
* Case module
  + Takes account for the ALUControl input
  + Conducts 3 different actions
    - Add
    - Subtract
    - Logical: AND, OR, set less than
* Outputs
  + ALUControl returns the ALUResult
  + Zero returns set on

ALU\_Control\_Unit.v

* Inputs
  + Funct field 6-bits: dictates what operation is returned as ALUControl
  + ALUOp 2-bit: dictates lw,sw, or beq
* Case module
  + Takes account for the ALUOp input
  + Operation lw -> add, subtract, AND, OR, set on less than
  + Operation sw -> add
  + Operation beq -> subtract
* Outputs
  + ALUControl is a 3-bit output that is sent to ALU.v to make operation happen.

Control\_unit.v

* Inputs
  + clk or clock – updates all state elements on the same clock edge
  + rst or reset – initializes hardware
  + Opcode 6-bit: dictates instruction
* Case module
  + Takes account for the Opcode input
  + Contains 6 instruction
    - R-type, beq, sw, lw, addi, j
  + Each instruction triggers 1 of 8 signals
    - RegWrite, RegDst, ALUSrc, Branch, MemWrite, MemtoReg, ALUOp, Jump
* Outputs
  + The outputs are the 8 different signals from the case module.
    - Not all will be outputted but each signal has its own register to output

Data Memory

* Inputs
  + clk or clock – updates all state elements on the same clock edge
  + rst or reset – initializes hardware
  + A or addres
  + RD or read data
* First always module
  + Assigns 32-bit data memory to RD input
* Initial module
  + Opens memory data text file and assigns write to fd, the closed.
* Second module

Imm\_SignExtend

* Inputs
  + Immediate 16-bit: number to extend
* Outputs
  + SignImm 32-bit: sign extended number of Immediate
* Assign SignImm
  + contains the result of the sign extending immediate.

Instr\_Memory

* Inputs
  + A: 8 instructions of 5 bits, width of 32-bits
* Initial module
  + Reads instruction from memory file
* Outputs
  + RD: 32 bit instructions

MIPS\_Single\_Cycle

* Inputs
  + clk or clock – updates all state elements on the same clock edge
  + rst or reset – initializes hardware
* Wires
  + Connecting all other Verilog files
* Instantiations
  + All other Verilog files.

PC\_Counter

* Inputs
  + clk or clock – updates all state elements on the same clock edge
  + rst or reset – initializes hardware
  + PRSrc – Asserted :adder computed by branch target/Deasserted:adder that computes the value of PC + 4
  + Jump
  + SignImm
  + Jump\_low\_26Bit
* Outputs
  + Register PC
* Wires
  + PC\_Next
  + PLCPus4
  + PCBranch
  + PCJump
* Assigns
  + PCPlus4
  + PCBranch
  + PCJump
  + PC\_Next
* Always module
  + If the hardware has not been initialized, will initialize with 32 bits of 0s

Reg\_File.v

* Inputs
  + clk or clock – updates all state elements on the same clock edge
  + A1, A2, A3 5-bit: addresses to be read
  + RegWrite: register that is written?
  + WD3 32-bit: write data
* register ROM
  + used to load initial values
* Outputs
  + RD1, RD2 32-bit: Holds the 32 bit addresses that will come from memory
* Initial module
  + Loading word instruction of 32 bit
* Always module
  + RegWrite is set to 1 when lw Instruction is executed

Tb\_MIPS\_Single\_Cycle

* Registers
  + Clk, rst\_n
  + Cnt 32-bit
* First initial module
  + Holds the vcd file to test
* Second initial module
  + Begins aa sequence of assigning bits to clk, rst\_n, cnt
  + Time delays between each assignment block
* Always
  + Executes when clk to 0 from 1
  + Adds 1 bit pf 1 to clk
* Instantation of MIPS\_Single\_Cycle
  + Assigns the values clk and rst\_n from MIPS\_Single\_Cycle to the tb