EGRE 426
Phase 2
Julion Rowland and Dustin Trimmer
10-30-2025

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Top_Level is
    Port (
        clk      : in  STD_LOGIC;
        Reset    : in STD_LOGIC;
        RegWr    : in  STD_LOGIC;
        Rd, Rs, Rt : in  unsigned(2 downto 0);
        ALUctr   : in  unsigned(3 downto 0);
--        Zero, Overflow, Carryout : out STD_LOGIC; -- i don't think we need these
        Result   : out unsigned(15 downto 0)
    );
end Top_Level;

architecture Behavioral of Top_Level is
-- signals and components are arranged from left to right and top to bottom
-- using the CPU block diagram
    signal PCIn, PCOut : unsigned(15 downto 0);
    signal adderOut : unsigned(15 downto 0);
    signal instrOut : unsigned(15 downto 0);
    signal Branch, MemtoReg, memWrite, memRead, ALUSrc, RegWrite : std_logic;
    signal RegDst, ALUOp : unsigned(15 downto 0);
    signal readData1, readData2 : unsigned(15 downto 0);
    signal signExtendOut : unsigned(15 downto 0);
    signal mux1Out : unsigned(15 downto 0);
    signal shiftOut : unsigned(15 downto 0);
    signal Zero : std_logic;
    signal ALU0Out : unsigned(15 downto 0);
    signal ALU1Out : unsigned(15 downto 0);
    signal andOut : std_logic;
    signal dataMemOut : unsigned(15 downto 0);
```

```
signal mux3Out : unsigned(15 downto 0);-- := RegWr;

component ProgramCounter -- Component: Program Counter
   generic ( N : integer := 16 );
   port (
      clk    : in  std_logic;
      rst    : in  std_logic;
      pc_in   : in  unsigned(N-1 downto 0);
      pc_out  : out unsigned(N-1 downto 0)
   );
end component;

component PCAdder -- Component: Program Adder
   generic ( N : integer := 16 );
      port (
   pc_in  : in  unsigned(N-1 downto 0);
   pc_out : out unsigned(N-1 downto 0)
);
end component;

component InstructionMemory -- Component: Instruction Memory
   generic ( N : integer := 16 );
      port (
   clk   : in  std_logic;
   addr  : in  unsigned(N-1 downto 0);
   instr : out unsigned(N-1 downto 0)
);
end component;

component MUX2to1 -- Component: MUX
   generic ( N : integer := 16 );
      port (
   A   : in  unsigned(N-1 downto 0);
   B   : in  unsigned(N-1 downto 0);
   Sel : in  std_logic;
   Y   : out unsigned(N-1 downto 0)
);
end component;
```

```
component Registers -- Component: Registers
   Port (
      clk    : in  STD_LOGIC;
      RegWr  : in  STD_LOGIC;
      Ra, Rb, Rw : in  unsigned(2 downto 0);
      busW   : in  unsigned(15 downto 0);
      busA, busB : out unsigned(15 downto 0)
   );
end component;

component SignExtend -- Component: Sign Extender
   generic ( N : integer := 16 );
      port (
   In6  : in  unsigned(5 downto 0);
   Out16 : out unsigned(N-1 downto 0)
);
end component;

component ShiftLeft2 -- Component: Left shift by 2
   generic ( N : integer := 16 );
      port (
   data_in  : in  unsigned(N-1 downto 0);
   data_out : out unsigned(N-1 downto 0)
);
end component;

component ALU -- Component: ALU
   generic ( N : integer := 16 );
   Port (
      A, B    : in  unsigned(N-1 downto 0);
      ALUctr  : in  unsigned(3 downto 0);
      Result  : out unsigned(N-1 downto 0);
      Zero, Overflow, Carryout : out STD_LOGIC
   );
end component;

component DataMemory -- Component: Data Memory
```

```
    generic ( N : integer := 16 );
       port (
    clk     : in  std_logic;
    MemRead  : in  std_logic;
    MemWrite : in  std_logic;
    Address  : in  unsigned(N-1 downto 0);
    WriteData: in  unsigned(N-1 downto 0);
    ReadData : out unsigned(N-1 downto 0)
  );
  end component;

begin
  PC: ProgramCounter
    port map(
      clk => clk,
      rst => Reset,
      pc_in => PCIn,
      pc_out => PCOut
    );

  PCADDER0 : PCAdder
    port map(
      pc_in => PCOut,
      pc_out => adderOut
    );

  INSTRUCTION_MEMORY : InstructionMemory
    port map(
      clk => clk,
      addr => PCOut,
      instr => instrOut
    );

  RF: Registers --My Register File
    port map (
      clk   => clk,
      RegWr => RegWr,
      Ra    => instrOut(11 downto 9),
```

```
      Rb    => instrOut(8 downto 6),
      Rw    => instrOut(5 downto 3),
      busW  => mux3Out,
      busA  => readData1,
      busB  => readData2
   );

SIGN_XTDR : SignExtend
   port map(
      In6 => instrOut(5 downto 0),
      out16 => signExtendOut
   );

MUX1 : Mux2to1-- MUX0 RESERVED FOR register input
   port map(
      A => readData2,
      B => signExtendOut,
      Sel => ALUSrc,
      Y => mux1Out
   );

LSHIFT : ShiftLeft2
   port map(
      data_in => signExtendOut,
      data_out => shiftOut
   );

ALU0: ALU --My Alu
   generic map ( N => 16 )
   port map (
      A       => readData1,
      B       => mux1Out,
      ALUctr   => ALUctr,
      Result   => ALU0Out,
      Zero     => Zero,
      Overflow => open,
      Carryout => open
   );
```

```
ALU1: ALU --My Alu
   generic map ( N => 16 )
   port map (
      A       => adderOut,
      B       => shiftOut,
      ALUctr   => "0000",
      Result   => ALU1Out,
      Zero     => open,
      Overflow => open,
      Carryout => open
   );

andOut <= Branch and Zero;

DATA_MEMORY : dataMemory
   port map(
      clk => clk,
      memRead => memRead,
      memWrite => memWrite,
      Address => ALU0Out,
      WriteData => readData2,
      ReadData => dataMemOut
   );

 MUX2 : Mux2to1
   port map(
      A => adderOut,
      B => ALU1Out,
      Sel => andOut,
      Y => PCIn
   );

MUX3 : Mux2to1
   port map(
      A => dataMemOut,
      B => ALU0Out,
      Sel => MemtoReg,
```

```
            Y => mux3Out
        );

    Result <= mux3Out; --Final Result

end Behavioral;
```