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

entity register_file is
    Port ( load : in  STD_LOGIC;
                        Clk : in STD_LOGIC;
           dest_sel : in  STD_LOGIC_VECTOR (2 downto 0);
           a_sel : in  STD_LOGIC_VECTOR (2 downto 0);
           b_sel : in  STD_LOGIC_VECTOR (2 downto 0);
           data : in  STD_LOGIC_VECTOR (15 downto 0);
           a_out : out  STD_LOGIC_VECTOR (15 downto 0);
           b_out : out  STD_LOGIC_VECTOR (15 downto 0));
end register_file;

architecture Behavioral of register_file is

--register module
COMPONENT reg_16 is
        PORT(
                        D : in std_logic_vector(15 downto 0);
                        load, Clk : in std_logic;
                        Q : out std_logic_vector(15 downto 0)
                        );
END COMPONENT;

--3 to 8 decoder for destination select
COMPONENT decoder_3_8 is
        PORT(
                        A0 : in  STD_LOGIC;
                        A1 : in  STD_LOGIC;
                        A2 : in  STD_LOGIC;
                        Q0 : out  STD_LOGIC;
                        Q1 : out  STD_LOGIC;
                        Q2 : out  STD_LOGIC;
                        Q3 : out  STD_LOGIC;
                        Q4 : out  STD_LOGIC;
                        Q5 : out  STD_LOGIC;
                        Q6 : out  STD_LOGIC;
                        Q7 : out  STD_LOGIC
                        );
END COMPONENT;

--8 to 1 multiplexer for output select
COMPONENT mux_8_1_16bit is
        PORT(
                        In0 : in STD_LOGIC_VECTOR(15 downto 0);
                        In1 : in STD_LOGIC_VECTOR(15 downto 0);
                        In2 : in STD_LOGIC_VECTOR(15 downto 0);
                        In3 : in STD_LOGIC_VECTOR(15 downto 0);
                        In4 : in STD_LOGIC_VECTOR(15 downto 0);
                        In5 : in STD_LOGIC_VECTOR(15 downto 0);
                        In6 : in STD_LOGIC_VECTOR(15 downto 0);
                        In7 : in STD_LOGIC_VECTOR(15 downto 0);
                        S0 : in STD_LOGIC;
                        S1 : in STD_LOGIC;
                        S2 : in STD_LOGIC;
                        Z : out STD_LOGIC_VECTOR(15 downto 0)
                        );
END COMPONENT;

--signals
signal load_reg0, load_reg1, load_reg2, load_reg3, load_reg4, load_reg5, load_reg6, load_reg7 : STD_LOGIC;
signal reg0_q, reg1_q, reg2_q, reg3_q, reg4_q, reg5_q, reg6_q, reg7_q, data_src_mux_out, src_reg :
STD_LOGIC_VECTOR(15 downto 0);
signal sel_reg0, sel_reg1, sel_reg2, sel_reg3, sel_reg4, sel_reg5, sel_reg6, sel_reg7 : STD_LOGIC;

begin
--port maps
        --registers
        --register 0
        reg00: reg_16 PORT MAP(
                                        D => data,
                                        load =>load_reg0,
                                        Clk => Clk,
                                        Q => reg0_q
        );
        --register 1
```

```
reg01: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg1,
                                Clk => Clk,
                                Q => reg1_q
);
--register 2
reg02: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg2,
                                Clk => Clk,
                                Q => reg2_q
);
--register 3
reg03: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg3,
                                Clk => Clk,
                                Q => reg3_q
);
--register 4
reg04: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg4,
                                Clk => Clk,
                                Q => reg4_q
);
--register 5
reg05: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg5,
                                Clk => Clk,
                                Q => reg5_q
);
--register 6
reg06: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg6,
                                Clk => Clk,
                                Q => reg6_q
);
--register 7
reg07: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg7,
                                Clk => Clk,
                                Q => reg7_q
);

--multiplexers for output selection
--mux A
muxA: mux_8_1_16bit PORT MAP(
                                In0 => reg0_q,
                                In1 => reg1_q,
                                In2 => reg2_q,
                                In3 => reg3_q,
                                In4 => reg4_q,
                                In5 => reg5_q,
                                In6 => reg6_q,
                                In7 => reg7_q,
                                S0 => a_sel(2), --need to flip these because definition of MSB/LSB is vague
                                S1 => a_sel(1),
                                S2 => a_sel(0),
                                Z => a_out
);
--mux B
muxB: mux_8_1_16bit PORT MAP(
                                In0 => reg0_q,
                                In1 => reg1_q,
                                In2 => reg2_q,
                                In3 => reg3_q,
                                In4 => reg4_q,
                                In5 => reg5_q,
                                In6 => reg6_q,
                                In7 => reg7_q,
                                S0 => b_sel(2), --same as above
                                S1 => b_sel(1),
```

```
                                    S2 => b_sel(0),
                                    Z => b_out
        );

        --decoder for load selection
        --decoder
        decoder: decoder_3_8 PORT MAP(
                                    A0 => dest_sel(2),
                                    A1 => dest_sel(1),
                                    A2 => dest_sel(0),
                                    Q0 => sel_reg0,
                                    Q1 => sel_reg1,
                                    Q2 => sel_reg2,
                                    Q3 => sel_reg3,
                                    Q4 => sel_reg4,
                                    Q5 => sel_reg5,
                                    Q6 => sel_reg6,
                                    Q7 => sel_reg7
        );

        --AND gates for load thingys
        load_reg0 <= sel_reg0 and load;
        load_reg1 <= sel_reg1 and load;
        load_reg2 <= sel_reg2 and load;
        load_reg3 <= sel_reg3 and load;
        load_reg4 <= sel_reg4 and load;
        load_reg5 <= sel_reg5 and load;
        load_reg6 <= sel_reg6 and load;
        load_reg7 <= sel_reg7 and load;

end Behavioral;
```