```
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 (3 downto 0);
           a_sel : in  STD_LOGIC_VECTOR (3 downto 0);
           b_sel : in  STD_LOGIC_VECTOR (3 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;

--4 to 16 decoder for destination select
COMPONENT decoder_4_to_16
PORT(
        A : IN std_logic_vector(3 downto 0);
        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;
        Q8 : OUT std_logic;
        Q9 : OUT std_logic;
        Q10 : OUT std_logic;
        Q11 : OUT std_logic;
        Q12 : OUT std_logic;
        Q13 : OUT std_logic;
        Q14 : OUT std_logic;
        Q15 : 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;

--16 to 1 multiplexer for output select
COMPONENT mux_16_1_16bit
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);
        In8 : IN std_logic_vector(15 downto 0);
        In9 : IN std_logic_vector(15 downto 0);
        In10 : IN std_logic_vector(15 downto 0);
        In11 : IN std_logic_vector(15 downto 0);
        In12 : IN std_logic_vector(15 downto 0);
        In13 : IN std_logic_vector(15 downto 0);
        In14 : IN std_logic_vector(15 downto 0);
        In15 : IN std_logic_vector(15 downto 0);
        S : IN std_logic_vector(3 downto 0);
        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, load_reg8 :
STD_LOGIC;
signal reg0_q, reg1_q, reg2_q, reg3_q, reg4_q, reg5_q, reg6_q, reg7_q, reg8_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, sel_reg8 : 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
);
--register 8
reg08: reg_16 PORT MAP(
                                D => data,
                                load =>load_reg8,
                                Clk => Clk,
                                Q => reg8_q
);

--multiplexers for output selection
--mux A
muxA: mux_16_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,
        In8 => reg8_q,
        In9 => "0000000000000000",
        In10 => "0000000000000000",
        In11 => "0000000000000000",
        In12 => "0000000000000000",
        In13 => "0000000000000000",
        In14 => "0000000000000000",
        In15 => "0000000000000000",
        S => a_sel,
        Z => a_out
);
--mux B
muxB: mux_16_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,
        In8 => reg8_q,
        In9 => "0000000000000000",
        In10 => "0000000000000000",
        In11 => "0000000000000000",
        In12 => "0000000000000000",
        In13 => "0000000000000000",
        In14 => "0000000000000000",
        In15 => "0000000000000000",
        S => b_sel,
        Z => b_out
);

--decoder for load selection
--decoder
decoder: decoder_4_to_16 PORT MAP(
        A => dest_sel,
        Q0 => sel_reg0,
```

```
                Q1 => sel_reg1,
                Q2 => sel_reg2,
                Q3 => sel_reg3,
                Q4 => sel_reg4,
                Q5 => sel_reg5,
                Q6 => sel_reg6,
                Q7 => sel_reg7,
                Q8 => sel_reg8
        );

        --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;
        load_reg8 <= sel_reg8 and load;

end Behavioral;
```