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

entity datapath is
    Port ( control : in STD_LOGIC_VECTOR(16 downto 0);
                          TD : in STD_LOGIC;
                          TA : in STD_LOGIC;
                          TB : in STD_LOGIC;
                          clk : in STD_LOGIC;
           const_in : in  STD_LOGIC_VECTOR (2 downto 0);
           data_in : in  STD_LOGIC_VECTOR (15 downto 0);
                          PC_in : in STD_LOGIC_VECTOR (15 downto 0);
                          MM : in STD_LOGIC;
           V : out  STD_LOGIC;
           C : out  STD_LOGIC;
           N : out  STD_LOGIC;
           Z : out  STD_LOGIC;
           address_out : out  STD_LOGIC_VECTOR (15 downto 0);
           data_out : out  STD_LOGIC_VECTOR (15 downto 0));
end datapath;

architecture Behavioral of datapath is
--components
        --function unit
        COMPONENT function_unit
        PORT(
                        A : in  STD_LOGIC_VECTOR (15 downto 0);
                        B : in  STD_LOGIC_VECTOR (15 downto 0);
                        S : in  STD_LOGIC_VECTOR (4 downto 0);
                        F : out  STD_LOGIC_VECTOR (15 downto 0);
                        V : out  STD_LOGIC;
                        C : out  STD_LOGIC;
                        N : out  STD_LOGIC;
                        Z : out  STD_LOGIC
                        );
        END COMPONENT;

        --register file
        COMPONENT register_file
        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 COMPONENT;

        --2 to 1 multiplexer (16bit)
        COMPONENT mux_2_1_16bit
        PORT(
                        In0 : in std_logic_vector(15 downto 0);
                        In1 : in std_logic_vector(15 downto 0);
                        S : in std_logic;
                        Z : out std_logic_vector(15 downto 0)
                        );
        END COMPONENT;

        --zero fill
        COMPONENT zero_fill
        PORT(
                input : IN std_logic_vector(2 downto 0);
                output : OUT std_logic_vector(15 downto 0)
                );
        END COMPONENT;


--signals
signal zero_fill_out, data_mux_out, a_data_out, b_data_out, b_mux_out, function_unit_out : STD_LOGIC_VECTOR (15
downto 0);
signal load, MD_sel, MB_sel : STD_LOGIC;
signal FS_in : STD_LOGIC_VECTOR(4 downto 0);
signal A_sel, B_sel, dest_sel : STD_LOGIC_VECTOR(3 downto 0);
signal control_signal : STD_LOGIC_VECTOR(19 downto 0);
```

```vhdl
begin
--port maps
	--register file
	register_file0: register_file PORT MAP(
				load => load,
				Clk => clk,
				dest_sel => dest_sel,
				a_sel => A_sel,
				b_sel => B_sel,
				data => data_mux_out,
				a_out => a_data_out,
				b_out => b_data_out
	);

	--function unit
	function_unit0: function_unit PORT MAP(
				A => a_data_out,
				B => b_mux_out,
				S => FS_in,
				F => function_unit_out,
				V => V,
				C => C,
				N => N,
				Z => Z
	);

	--2 to 1 multiplexer for B/constant select (MB)
	mux0: mux_2_1_16bit PORT MAP(
				In0 => b_data_out,
				In1 => zero_fill_out,
				S => MB_sel,
				Z => b_mux_out
	);

	--2 to 1 multiplexer for data source select (MD)
	mux1: mux_2_1_16bit PORT MAP(
				In0 => function_unit_out,
				In1 => data_in,
				S => MD_sel,
				Z => data_mux_out
	);

	--2 to 1 multiplexer for address select (MM)
	mux2: mux_2_1_16bit PORT MAP(
				In0 => PC_in,
				In1 => a_data_out,
				S => MM,
				Z => address_out
	);

	--zero fill
	Inst_zero_fill: zero_fill PORT MAP(
		input => const_in,
		output => zero_fill_out
	);

	--control word mappings
	load <= control_signal(0);
	MD_sel <= control_signal(1);
	FS_in <= control_signal(6 downto 2);
	MB_sel <= control_signal(7);
	B_sel <= control_signal(11 downto 8);
	A_Sel <= control_signal(15 downto 12);
	dest_sel <= control_signal(19 downto 16);

	--address_out <= a_data_out;
	data_out <= b_data_out;
	control_signal(10 downto 0) <= control(10 downto 0);
	control_signal(11) <= TB;
	control_signal(14 downto 12) <= control(13 downto 11);
	control_signal(15) <= TA;
	control_signal(18 downto 16) <= control(16 downto 14);
	control_signal(19) <= TD;


end Behavioral;
```