```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.numeric_std.all;

entity rf32 is
port (
 rst_n : in std_logic;
 write_en: in std_logic;  -- write control
 write_addr: in std_logic_vector(4 downto 0);  -- write address
 write_data: in std_logic_vector(31 downto 0);
 read_addr_1: in std_logic_vector(4 downto 0);
 read_data_1: out std_logic_vector(31 downto 0);
 read_addr_2: in std_logic_vector(4 downto 0);
 read_data_2: out std_logic_vector(31 downto 0)
);
end rf32;

architecture Behavioral of rf32 is
type reg_type is array (0 to 31 ) of std_logic_vector (31 downto
0);
signal reg_array: reg_type;
begin
 process(rst_n,write_en,write_addr,write_data)
 begin
 if(rst_n='1') then
      reg_array<=(others => (others => '0'));
      reg_array(0)<= x"00000000";
      reg_array(1)<= x"00000001";
      reg_array(2)<= x"00000002";
      reg_array(3)<= x"00000003";
      reg_array(4)<= x"00000004";
      reg_array(5)<= x"00000005";
      reg_array(6)<= x"00000006";
      reg_array(7)<= x"00000007";
      reg_array(8)<= x"00000008";
      reg_array(9)<= x"00000009";
 else
   if(write_en='1') then
    reg_array(to_integer(unsigned(write_addr(4 downto 0)))) <=
write_data;
   end if;
 end if;
 end process;

 read_data_1 <= x"00000000" when read_addr_1(4 downto 0) =
"00000" else reg_array(to_integer(unsigned(read_addr_1(4 downto
0))));
 read_data_2 <= x"00000000" when read_addr_2(4 downto 0) =
"00000" else reg_array(to_integer(unsigned(read_addr_2(4 downto
0))));
```

```
end Behavioral;
```