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

entity TwoPortMemory is
    Port ( clk : in STD_LOGIC;
           addressPort1 : in STD_LOGIC_VECTOR(7 downto 0);
           writeEnablePort1 : in STD_LOGIC;
           readEnablePort1 : in STD_LOGIC;
           dataInPort1 : in STD_LOGIC_VECTOR(7 downto 0);
           dataOutPort1 : out STD_LOGIC_VECTOR(7 downto 0);

           addressPort2 : in STD_LOGIC_VECTOR(7 downto 0);
           writeEnablePort2 : in STD_LOGIC;
           readEnablePort2 : in STD_LOGIC;
           dataInPort2 : in STD_LOGIC_VECTOR(7 downto 0);
           dataOutPort2 : out STD_LOGIC_VECTOR(7 downto 0)
    );
end TwoPortMemory;

architecture Behavioral of TwoPortMemory is
    type MemoryArray is array (0 to 255) of STD_LOGIC_VECTOR(7 downto 0);
    signal memory : MemoryArray := (others => (others => '0'));

    procedure WriteMemory(proc_address : in STD_LOGIC_VECTOR; proc_data : in
STD_LOGIC_VECTOR) is
    begin
        memory(to_integer(proc_address)) <= proc_data;
    end WriteMemory;

    function ReadMemory(func_address : in STD_LOGIC_VECTOR) return
STD_LOGIC_VECTOR is
    begin
        return memory(to_integer(func_address));
    end ReadMemory;

begin
    process(clk)
    begin
        if rising_edge(clk) then
            -- Port 1
            if writeEnablePort1 = '1' then
                WriteMemory(addressPort1, dataInPort1);
            elsif readEnablePort1 = '1' then
                dataOutPort1 <= ReadMemory(addressPort1);
```

```
            end if;

            -- Port 2
            if writeEnablePort2 = '1' then
                WriteMemory(addressPort2, dataInPort2);
            elsif readEnablePort2 = '1' then
                dataOutPort2 <= ReadMemory(addressPort2);
            end if;
        end if;
    end process;

end Behavioral;
```