

| | |
|--|----|
| -- **** STUDENT: 64000225..... | 3 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 3 |
| -- **** STUDENT: 64190088..... | 6 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 6 |
| -- **** STUDENT: 64200100..... | 9 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 9 |
| -- **** STUDENT: 64200112..... | 12 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 12 |
| -- **** STUDENT: 64200163..... | 15 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 15 |
| -- **** STUDENT: 64200238..... | 18 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 18 |
| -- **** STUDENT: 64200288..... | 21 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 21 |
| -- **** STUDENT: 64200296..... | 24 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 24 |
| -- **** STUDENT: 64200385..... | 27 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 27 |
| -- **** STUDENT: 64210113..... | 30 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 30 |
| -- **** STUDENT: 64210290..... | 33 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 33 |
| -- **** STUDENT: 64210382..... | 36 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 36 |
| -- **** STUDENT: 64210384..... | 39 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 39 |
| -- **** STUDENT: 64210386..... | 42 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 42 |
| -- **** STUDENT: 64210445..... | 45 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 45 |
| -- **** STUDENT: 64210455..... | 48 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 48 |
| -- **** STUDENT: 64210457..... | 51 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 51 |

| | |
|--|----|
| -- **** STUDENT: 64240430..... | 54 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 54 |
| -- **** PREDLOGA VAJE | 57 |
| -- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb..... | 57 |

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1 : Vzporedno nalaganje x => Q
    -- 1 0 : Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1 : Pomikanje desno  ( v smeri od MSB do LSB )
    -- 0 0 : Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );

```

```

        PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );
              w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
              f      : OUT  STD_LOGIC
            );
    END COMPONENT;

    COMPONENT muxnto1_bus IS
        generic( n_addr      : INTEGER := 2;
              bus_width     : INTEGER := 8 );
        PORT ( s : IN      std_logic_vector( n_addr - 1 downto 0 );
              w : IN      muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
              f : OUT     std_logic_vector( bus_width - 1 downto 0 )
            );
    END COMPONENT;

    component reg_file is
        generic( nr_regs      : natural := 4;
              reg_width     : natural := 8 );
        PORT ( clk, -- clock input
              LE      : in std_logic; -- Load enable input ( active '1' )
              nRST    : in std_logic; -- reset input ( active '0' )
              dest_select, -- register number destination select input
              A_select, -- A, B bus destination select input
              B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
              D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
              A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
            );
    end component;

    END reg_file_functions;

    PACKAGE BODY reg_file_functions IS

        -- @Function name: sizeof
        -- @Parameters:
        -- a: input number
        -- @Return:
        -- Number of bits required to encode a binary input number a
        FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
            VARIABLE aggregate : NATURAL := a;

```

```

        VARIABLE return_val : NATURAL := 0;
BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64190088
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select   : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```



```

-- *****
-- **** STUDENT: 64200100
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select   : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64200112
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in      std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64200163
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s      : in  std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```



```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
  compute_sizeof:
  FOR i IN a DOWNTO 0 LOOP
    IF aggregate > 0 THEN
      return_val := return_val + 1;    -- increment number of encoding bits
    END IF;
    aggregate := aggregate / 2;      -- divide by base of 2
  END LOOP;
  RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
  variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
  for j in test_array'range loop
    test_array( j ) := j;
  end loop;
  return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic;    -- Load enable input      ( active '1' )
           nRST    : in std_logic;    -- reset input          ( active '0' )
           dest_select, -- register number destination select input
           A_select,  -- A, B bus destination select input
           B_select   : in  std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D           : in  std_logic_vector( reg_width - 1 downto 0 );    -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 )    -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```



```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select   : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D           : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B        : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

END reg_file_functions;

PACKAGE BODY reg_file_functions IS

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210113
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select   : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D           : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B        : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```



```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
  compute_sizeof:
  FOR i IN a DOWNTO 0 LOOP
    IF aggregate > 0 THEN
      return_val := return_val + 1;    -- increment number of encoding bits
    END IF;
    aggregate := aggregate / 2;      -- divide by base of 2
  END LOOP;
  RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
  variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
  for j in test_array'range loop
    test_array( j ) := j;
  end loop;
  return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```



```

BEGIN
  compute_sizeof:
  FOR i IN a DOWNTO 0 LOOP
    IF aggregate > 0 THEN
      return_val := return_val + 1;    -- increment number of encoding bits
    END IF;
    aggregate := aggregate / 2;      -- divide by base of 2
  END LOOP;
  RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
  variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
  for j in test_array'range loop
    test_array( j ) := j;
  end loop;
  return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210386
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN      std_logic_vector( n_addr - 1 downto 0 );
        w : IN      muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT     std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in  std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in  std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

END reg_file_functions;

PACKAGE BODY reg_file_functions IS

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
  compute_sizeof:
  FOR i IN a DOWNTO 0 LOOP
    IF aggregate > 0 THEN
      return_val := return_val + 1;    -- increment number of encoding bits
    END IF;
    aggregate := aggregate / 2;      -- divide by base of 2
  END LOOP;
  RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
  variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
  for j in test_array'range loop
    test_array( j ) := j;
  end loop;
  return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in      std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo   ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in      std_logic_vector( n_addr - 1 downto 0 );

```



```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT STD_LOGIC
    );
END COMPONENT;

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

END reg_file_functions;

PACKAGE BODY reg_file_functions IS

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select   : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D           : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B        : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
    compute_sizeof:
    FOR i IN a DOWNTO 0 LOOP
        IF aggregate > 0 THEN
            return_val := return_val + 1;    -- increment number of encoding bits
        END IF;
        aggregate := aggregate / 2;        -- divide by base of 2
    END LOOP;
    RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
    variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
    for j in test_array'range loop
        test_array( j ) := j;
    end loop;
    return test_array;
end function;

END reg_file_functions;

```

```

-- *****
-- **** STUDENT: 64240430
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select   : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D           : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B        : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
  compute_sizeof:
  FOR i IN a DOWNTO 0 LOOP
    IF aggregate > 0 THEN
      return_val := return_val + 1;    -- increment number of encoding bits
    END IF;
    aggregate := aggregate / 2;      -- divide by base of 2
  END LOOP;
  RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
  variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
  for j in test_array'range loop
    test_array( j ) := j;
  end loop;
  return test_array;
end function;

END reg_file_functions;

```



```

-- *****
-- **** PREDLOGA VAJE
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
LIBRARY ieee;
USE ieee.std_logic_1164.all;

PACKAGE reg_file_functions IS
    Type muxnto1_bus_type is array ( natural range <>, natural range <> ) of STD_LOGIC;
    Type t_int_array_of_regs is array ( natural range <> ) of integer;
    FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL;
    function initRegConstants ( dim : integer ) return t_int_array_of_regs;

    COMPONENT dmuxnto1 IS
        generic( n_addr: natural := 2 );
        PORT (
            s : in      std_logic_vector( n_addr - 1 downto 0 );
            w : in      STD_LOGIC;
            f : out     std_logic_vector( 2**n_addr - 1 downto 0 )
        );
    END COMPONENT;

    -- S0 S1
    -- 1 1: Vzporedno nalaganje x => Q
    -- 1 0: Pomikanje levo ( v smeri od LSB do MSB )
    -- 0 1: Pomikanje desno ( v smeri od MSB do LSB )
    -- 0 0: Držanje vsebine
    COMPONENT shift_reg is
        generic( reg_size: natural := 4 );
        PORT ( clk, nCLR, sr_in, sl_in : IN std_logic;
            s : in std_logic_vector( 1 downto 0 );
            x : in std_logic_vector( reg_size - 1 downto 0 );
            Q : out std_logic_vector( reg_size - 1 downto 0 )
        );
    end COMPONENT;

    COMPONENT muxnto1 IS
        generic( n_addr: natural := 2 );
        PORT ( s : in std_logic_vector( n_addr - 1 downto 0 );

```

```

        w      : in  std_logic_vector( 2**n_addr - 1 downto 0 );
        f      : OUT  STD_LOGIC
    );
END COMPONENT;

```

```

COMPONENT muxnto1_bus IS
    generic( n_addr      : INTEGER := 2;
             bus_width   : INTEGER := 8 );
    PORT (
        s : IN  std_logic_vector( n_addr - 1 downto 0 );
        w : IN  muxnto1_bus_type( 2**n_addr - 1 DOWNT0 0, bus_width - 1 DOWNT0 0 );
        f : OUT std_logic_vector( bus_width - 1 downto 0 )
    );
END COMPONENT;

```

```

component reg_file is
    generic( nr_regs      : natural := 4;
             reg_width    : natural := 8 );
    PORT ( clk, -- clock input
           LE      : in std_logic; -- Load enable input ( active '1' )
           nRST    : in std_logic; -- reset input ( active '0' )
           dest_select, -- register number destination select input
           A_select, -- A, B bus destination select input
           B_select  : in std_logic_vector( sizeof( nr_regs - 1 ) - 1 downto 0 );
           D          : in std_logic_vector( reg_width - 1 downto 0 ); -- data input bus input
           A, B       : out std_logic_vector( reg_width - 1 downto 0 ) -- A, B bus output
    );
end component;

```

```

END reg_file_functions;

```

```

PACKAGE BODY reg_file_functions IS

```

```

--    @Function name: sizeof
--    @Parameters:
--    a: input number
--    @Return:
--    Number of bits required to encode a binary input number a
FUNCTION sizeof ( a: NATURAL ) RETURN NATURAL IS
    VARIABLE aggregate : NATURAL := a;
    VARIABLE return_val : NATURAL := 0;

```

```

BEGIN
  compute_sizeof:
  FOR i IN a DOWNTO 0 LOOP
    IF aggregate > 0 THEN
      return_val := return_val + 1;    -- increment number of encoding bits
    END IF;
    aggregate := aggregate / 2;      -- divide by base of 2
  END LOOP;
  RETURN return_val;
END sizeof;

function initRegConstants ( dim : integer ) return t_int_array_of_regs is
  variable test_array : t_int_array_of_regs( 0 to dim - 1 );
begin
  for j in test_array'range loop
    test_array( j ) := j;
  end loop;
  return test_array;
end function;

END reg_file_functions;

```

