

-- **** STUDENT: 64000225.....	2
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	2
-- **** STUDENT: 64200100.....	5
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	5
-- **** STUDENT: 64200112.....	8
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	8
-- **** STUDENT: 64200238.....	11
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	11
-- **** STUDENT: 64200288.....	14
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	14
-- **** STUDENT: 64200296.....	17
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	17
-- **** STUDENT: 64200385.....	20
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	20
-- **** STUDENT: 64210113.....	23
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	23
-- **** STUDENT: 64210290.....	26
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	26
-- **** STUDENT: 64210382.....	29
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	29
-- **** STUDENT: 64210384.....	32
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	32
-- **** STUDENT: 64210386.....	35
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	35
-- **** STUDENT: 64210445.....	38
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	38
-- **** STUDENT: 64210455.....	41
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	41
-- **** STUDENT: 64210457.....	44
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	44
-- **** STUDENT: 64240430.....	47
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	47
-- **** PREDLOGA VAJE.....	50
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	50

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;

ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
END cpu_tb;

ARCHITECTURE ndv OF cpu_tb IS

    COMPONENT cpu
        generic( nr_regs          : natural := 8;
                 reg_width       : natural := 16;
                 ram_nr_addr     : natural := 4;
                 rom_nr_addr     : natural := 4
                );
        PORT ( clk, -- clock input
              nRST : in std_logic; -- reset input ( active '0' )
              IOBUS_WnR : out std_logic; -- io bus write input ( active '1' )
        );
    END COMPONENT;

```

```

        IOBUS_Data_in      : in  std_logic_vector( reg_width - 1 downto 0 );  -- io bus data input
        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 );  -- io address bus
        IOBUS_Data_out    : out std_logic_vector( reg_width - 1 downto 0 );  -- io bus data output
        ProgMem_Addr : out std_logic_vector( reg_width - 1 downto 0 );  -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0';  -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk;  -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,
            rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,
            IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in      => IOBUS_Data_out,  -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN
        WAIT FOR ( PERIOD/2 );
        nRST  <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64200100
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out,  -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```

-- *****
-- **** STUDENT: 64200112
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;

```

```

ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );

```

```

END cpu_tb;

```

```

ARCHITECTURE ndv OF cpu_tb IS

```

```

    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;   -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;   -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input

```



```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map ( nRST => nRST,                addr => ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
        data => IR
        );

    DATAMEM : sram
    generic map ( n_addr => ram_nr_addr, bus_width => reg_width )
    Port map ( clk => nclk, nRST => nRST, W_nR => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in => IOBUS_Data_out, -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
    clk <= '0';
    WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    clk <= '1';
    WAIT FOR ( PERIOD * DUTY_CYCLE );
    END LOOP CLOCK_LOOP;
    END PROCESS;

PROCESS
BEGIN
    WAIT FOR ( PERIOD/2 );
    nRST <= '1';
    WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64200238
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map ( nRST => nRST,                addr => ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
        data => IR
        );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map ( clk      => nclk,          nRST  => nRST,          W_nR  => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in      => IOBUS_Data_out,  -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

        WAIT FOR ( PERIOD/2 );
        nRST  <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64200288
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out, -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN
                    WAIT FOR ( PERIOD/2 );
                    nRST  <= '1';
                    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```



```
-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;   -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;   -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out, -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR             : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out,  -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210113
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```



```

        Port map ( nRST => nRST,                addr => ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
        data => IR
        );

    DATAMEM : sram
    generic map ( n_addr => ram_nr_addr, bus_width => reg_width )
    Port map ( clk => nclk, nRST => nRST, W_nR => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in => IOBUS_Data_out, -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN
        WAIT FOR ( PERIOD/2 );
        nRST <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out, -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out, -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```



```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map ( nRST => nRST,                addr => ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
        data => IR
        );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map ( clk      => nclk,          nRST  => nRST,          W_nR  => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in    => IOBUS_Data_out, -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN
        WAIT FOR ( PERIOD/2 );
        nRST  <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210386
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map ( nRST => nRST,          addr => ProgMem_Addr( rom_nr_addr - 1 downto 0 ),      -- use
only rom_nr_addr bits of PC
        data => IR
        );

    DATAMEM : sram
    generic map ( n_addr => ram_nr_addr, bus_width => reg_width )
    Port map ( clk => nclk,          nRST => nRST,          W_nR => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in => IOBUS_Data_out, -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN
        WAIT FOR ( PERIOD/2 );
        nRST <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out, -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```



```
-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out, -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

    WAIT FOR ( PERIOD/2 );
    nRST  <= '1';
    WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

        DATAMEM : sram
        generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
        Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
                    addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
                    data_in    => IOBUS_Data_out,  -- data input bus input
                    data_out => IOBUS_Data_in -- A, B bus output
                    );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

        WAIT FOR ( PERIOD/2 );
        nRST  <= '1';
        WAIT FOR PERIOD;
END PROCESS;

END ndv;

```

```
-- *****
-- **** STUDENT: 64240430
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
```

```
ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );
```

```
END cpu_tb;
```

```
ARCHITECTURE ndv OF cpu_tb IS
```

```
    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input    ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input
        );
    end component;
```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```



```

        Port map ( nRST => nRST,                addr => ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
        data => IR
        );

    DATAMEM : sram
    generic map ( n_addr => ram_nr_addr, bus_width => reg_width )
    Port map ( clk => nclk, nRST => nRST, W_nR => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in => IOBUS_Data_out, -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN
        WAIT FOR ( PERIOD/2 );
        nRST <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

```

-- *****
-- **** PREDLOGA VAJE
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library ieee;
use ieee.numeric_std.all;
use ieee.math_real.all;
library work;
use work.reg_file_functions.all;
use work.cpu_datapath_functions.all;
use work.cpu_functions.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;

```

```

ENTITY cpu_tb IS
    generic( nr_regs          : natural := 8;
             reg_width       : natural := 16;
             ram_nr_addr     : natural := 4;
             rom_nr_addr     : natural := 4
            );

```

```

END cpu_tb;

```

```

ARCHITECTURE ndv OF cpu_tb IS

```

```

    COMPONENT cpu
        generic(
            nr_regs          : natural := 8;
            reg_width       : natural := 16;
            ram_nr_addr     : natural := 4;
            rom_nr_addr     : natural := 4
        );
        PORT (
            clk,             -- clock input
            nRST             : in  std_logic;    -- reset input      ( active '0' )
            IOBUS_WnR        : out std_logic;    -- io bus write input   ( active '1' )
            IOBUS_Data_in    : in  std_logic_vector( reg_width - 1 downto 0 ); -- io bus data input

```

```

        IOBUS_Address: out std_logic_vector( reg_width - 1 downto 0 ); -- io address bus
        IOBUS_Data_out  : out std_logic_vector( reg_width - 1 downto 0 ); -- io bus data output
        ProgMem_Addr   : out std_logic_vector( reg_width - 1 downto 0 ); -- program memory address
        IR              : in  std_logic_vector( reg_width - 1 downto 0 )  -- program memory data
input
    );
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nclk : std_logic := '0'; -- inverted clock input
SIGNAL      nRST : std_logic := '0';
SIGNAL      IOBUS_WnR : std_logic := '0';
SIGNAL      IOBUS_Data_in : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Address : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IOBUS_Data_out : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      ProgMem_Addr : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );
SIGNAL      IR      : std_logic_vector ( reg_width - 1 DownTo 0 ) := ( others => '0' );

constant    PERIOD : time := 400 ns;
constant    DUTY_CYCLE : real := 0.5;
constant    OFFSET : time := 100 ns;

BEGIN

    nclk  <= not clk; -- invert main clock ( all memory writes are on negative edge of main clock )

    UUT : cpu
        generic MAP (
            nr_regs      => nr_regs,          reg_width      => reg_width,          ram_nr_addr  =>
ram_nr_addr,          rom_nr_addr  => rom_nr_addr
        )
        PORT MAP (
            clk => clk,          nRST => nRST,          IOBUS_WnR => IOBUS_WnR,          IOBUS_Data_in =>
IOBUS_Data_in,          IOBUS_Address => IOBUS_Address,          IOBUS_Data_out => IOBUS_Data_out,
            ProgMem_Addr => ProgMem_Addr,          IR => IR
        );

        PROGMEM : rom
        generic map (          n_addr => rom_nr_addr, bus_width => reg_width )

```

```

        Port map (    nRST => nRST,                addr  =>    ProgMem_Addr( rom_nr_addr - 1 downto 0 ),    -- use
only rom_nr_addr bits of PC
                    data => IR
                    );

    DATAMEM : sram
    generic map ( n_addr      => ram_nr_addr, bus_width => reg_width )
    Port map (    clk      => nclk,                nRST  => nRST,                W_nR   => IOBUS_WnR, -- write/read
control ( write = '1' )
        addr  => IOBUS_Address( ram_nr_addr - 1 downto 0 ), -- register number destination select input
        data_in    => IOBUS_Data_out, -- data input bus input
        data_out => IOBUS_Data_in -- A, B bus output
        );

PROCESS
BEGIN
WAIT for OFFSET;
CLOCK_LOOP : LOOP
clk  <= '0';
WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
clk  <= '1';
WAIT FOR ( PERIOD * DUTY_CYCLE );
END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
BEGIN

        WAIT FOR ( PERIOD/2 );
        nRST  <= '1';
        WAIT FOR PERIOD;
    END PROCESS;

END ndv;

```

