```
--------------------------------------------------------------------------------------------------------------------------------
-- rom.vhd
--------------------------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.ALL ;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;
library lib_nanoproc;
use lib_nanoproc.all;
USE lib_nanoproc.nano_pkg.all;

entity ROM is
port (  address : in std_logic_vector(len_addr_bus-2 downto 0) ;
    output  : out std_logic_vector(len_data_bus-1 downto 0) ) ;
end ROM;

--------------------------------------------------------------------
-- arch_prog_0
--------------------------------------------------------------------
architecture arch_prog_0 of ROM is

type tab_rom is array (0 to 2**(len_addr_bus-1)-1) of bit_vector(len_data_bus-1 downto 0) ;

constant MY_ROM : tab_rom :=
( 0 => x"0040", -- ldi r0,#AAAA
  1 => x"AAAA",
  2 => x"0058", -- ldi r3,#000F
  3 => x"000F",
  4 => x"0083", -- add r0,r3
  5 => x"027F", -- bra,-1 pour boucle (10 0111 1111)
  6 => x"0000",
  others=>x"0000");
begin

    output <= to_stdlogicvector(MY_ROM(conv_integer(address))) ;

end architecture;

--------------------------------------------------------------------------------------------------------------------------------
-- ram.vhd
--------------------------------------------------------------------------------------------------------------------------------
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

-- synthese d'une ram cf ALTERA AN 225

entity RAM is
generic (add_bits : integer;
     data_bits : integer);
port (
         add    : in std_logic_vector(add_bits-1 downto 0);
     data_in : in std_logic_vector(data_bits-1 downto 0);
     data_out: out std_logic_vector(data_bits-1 downto 0);
     clk,write : in std_logic);
end ram;
```

```
architecture A of ram is
  subtype word is std_logic_vector(data_bits-1 downto 0);
  constant nwords :integer :=2**add_bits;
  type mem_type is array (0 to nwords-1) of word;
  signal  mem : mem_type;

begin
  seq : process(clk,add)
  begin
      if (clk'event and clk = '1') then
    if (write = '1') then
          mem(conv_integer(add))<=data_in; -- positionne mem fin process
    end if;
      end if;
  end process seq;

  data_out<=mem(conv_integer(add));

end A;

--------------------------------------------------------------------------------------------------------------------------------------------------------
-- port_io.vhd
--------------------------------------------------------------------------------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.ALL ;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;
library lib_nanoproc;
use lib_nanoproc.all;
USE lib_nanoproc.nano_pkg.all;

-- Port I/O de 32 entrees sorties
-- entrees :
-- clk, reset_n
-- ecr : signal ecriture dans registre out, actif a 1
-- dat_in_io : signal du bus de donnees pour ecriture dans registre
-- dat_out_io : signal du bus de donnees pour lecture des entrees
-- adress : selection 0 : poids faible, 1 poids forts
-- in_io : 32 entrees exterieures
-- out_io : 32 sorties exterieures

entity PORT_IO is
port (  clk, reset_n, ecr : in std_logic;
    adress      : in std_logic;
    dat_in_io   : in std_logic_vector(len_data_bus-1 downto 0) ;
    in_io       : in std_logic_vector(31 downto 0);
    dat_out_io      : out std_logic_vector(len_data_bus-1 downto 0);
    out_io      : out std_logic_vector(31 downto 0)) ;
end PORT_IO;

architecture A of PORT_IO is

signal reg_d, reg_q : std_logic_vector(31 downto 0);

begin
```

```
synch : process(clk)
begin
    if (clk'event and clk='1') then
        if (reset_n='0') then
            reg_q<=(others=>'0');    -- RAZ registre
        else
            reg_q<=reg_d;        -- fonction registre
        end if;
    end if;
end process;

combi : process(ecr,adress,dat_in_io,reg_q,in_io)
begin
    reg_d<=reg_q;        -- fonction memo du registre
    if (ecr='1') then
        if (adress='0') then
            reg_d(15 downto 0)<=dat_in_io;
        else
            reg_d(31 downto 16)<=dat_in_io;
        end if;
    end if;
    out_io<=reg_q;
    if (adress='0') then
        dat_out_io<=in_io(15 downto 0);
    else
        dat_out_io<=in_io(31 downto 16);
    end if;

end process;


end A;
```