library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity reg1 is

port( clk : in std\_logic;

reg\_alu\_in: in std\_logic\_vector(7 downto 0);

reg\_alu\_out: out std\_logic\_vector(7 downto 0);

deop\_alu\_in: in std\_logic\_vector(3 downto 0);

deop\_alu\_out: out std\_logic\_vector(3 downto 0);

offset\_sig\_in: in std\_logic\_vector(7 downto 0);

offset\_sig\_out: out std\_logic\_vector(7 downto 0);

pcjump\_sig\_in: in std\_logic\_vector(7 downto 0);

pcjump\_sig\_out: out std\_logic\_vector(7 downto 0);

instructions\_in: in std\_logic\_vector(15 downto 0);

instructions\_out: out std\_logic\_vector(15 downto 0)

);

end reg1;

architecture behv of reg1 is

begin

reg\_alu: process(clk, reg\_alu\_in)

begin

if (clk'event and clk = '1') then

reg\_alu\_out <= reg\_alu\_in;

end if;

end process;

deop\_alu: process(clk, deop\_alu\_in)

begin

if(clk'event and clk = '1') then

deop\_alu\_out <= deop\_alu\_in;

end if;

end process;

offset\_sig: process(clk, offset\_sig\_in)

begin

if(clk'event and clk = '1') then

offset\_sig\_out<= offset\_sig\_in;

end if;

end process;

pcjump\_sig: process(clk , pcjump\_sig\_in)

begin

if(clk'event and clk = '1') then

pcjump\_sig\_out <= pcjump\_sig\_in;

end if;

end process;

instructions: process(clk, instructions\_in)

begin

if(clk'event and clk = '1') then

instructions\_out <= instructions\_in;

end if;

end process;

end behv;