Problem

a) Design and implement an 8-bit register with 2-bit select switches, sel.
If sel = “00” no operation, else if sel = “01” assign the content of the register to “11110000”, else if sel = “10” assign the content to “10101010” and else if sel = “11” assign the content to “00110011”. Use the single step mechanism with buttons b0 and b1.

```
entity reg_sel is
port (ck, b0, b1: in   std_logic;
                sel: in   std_logic_vector(1 downto 0);
                  z: out std_logic_vector(7 downto 0)
      );
end reg_sel;
```

b) Implement an addition function to the register in Part a. When the input sel = '00' and if the switch inv = '1' then the new content is the bit-wise not of the current content.

```
entity reg_sel_reg is
port (ck,b0,b1,inv: in   std_logic;
                  sel: in   std_logic_vector(1 downto 0);
                    z: out std_logic_vector(7 downto 0)
      );
end reg_sel_inv;
```

Solution Part b

```
----------------------------------------------------
-- Company: Drexel ECE
-- Engineer: Prawat
-- Design and implement an 8-bit register with
-- 2-bit select switches, sel.
-- If sel = "00" then
--   if inv = '1' assign bit-wise not
-- else if sel = "01" assign "11110000"
-- else if sel = "10" assign "10101010" and
-- else if sel = "11" assign "00110011".
----------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg_sel_inv is
port (ck, inv, btn0, btn1: in  std_logic;
      sel: in  std_logic_vector(1 downto 0);
        z: out std_logic_vector(7 downto 0)
        );
end reg_sel_inv;

architecture Behavioral of reg_sel_inv is
signal temp : std_logic_vector(7 downto 0);
signal en : std_logic;
begin
process(en)
begin
if en = '1' and en'event then
  case sel is
   when "00" =>
      if inv = '1' then temp <= not temp;
      else null;
      end if;
   when "01" => temp <= "11110000";
   when "10" => temp <= "10101010";
   when "11" => temp <= "00110011";
   when others => null;
  end case;
end if;
end process;
 z <= temp;
-- single step, debounce (db)
-- btn0 to enter and btn1 to reset
process(ck)
type db_state is (not_rdy, rdy, pulse);
variable db_ns: db_state;
begin
if ck='1' and ck'event then
  case db_ns is
   when not_rdy => en <= '0';
   if btn1 = '1' then db_ns := rdy; end if;
   when rdy => en <= '0';
   if btn0 = '1' then db_ns := pulse; end if;
   when pulse => en <= '1';
   db_ns := not_rdy;
   when others => null;
  end case;
end if;
end process;
```