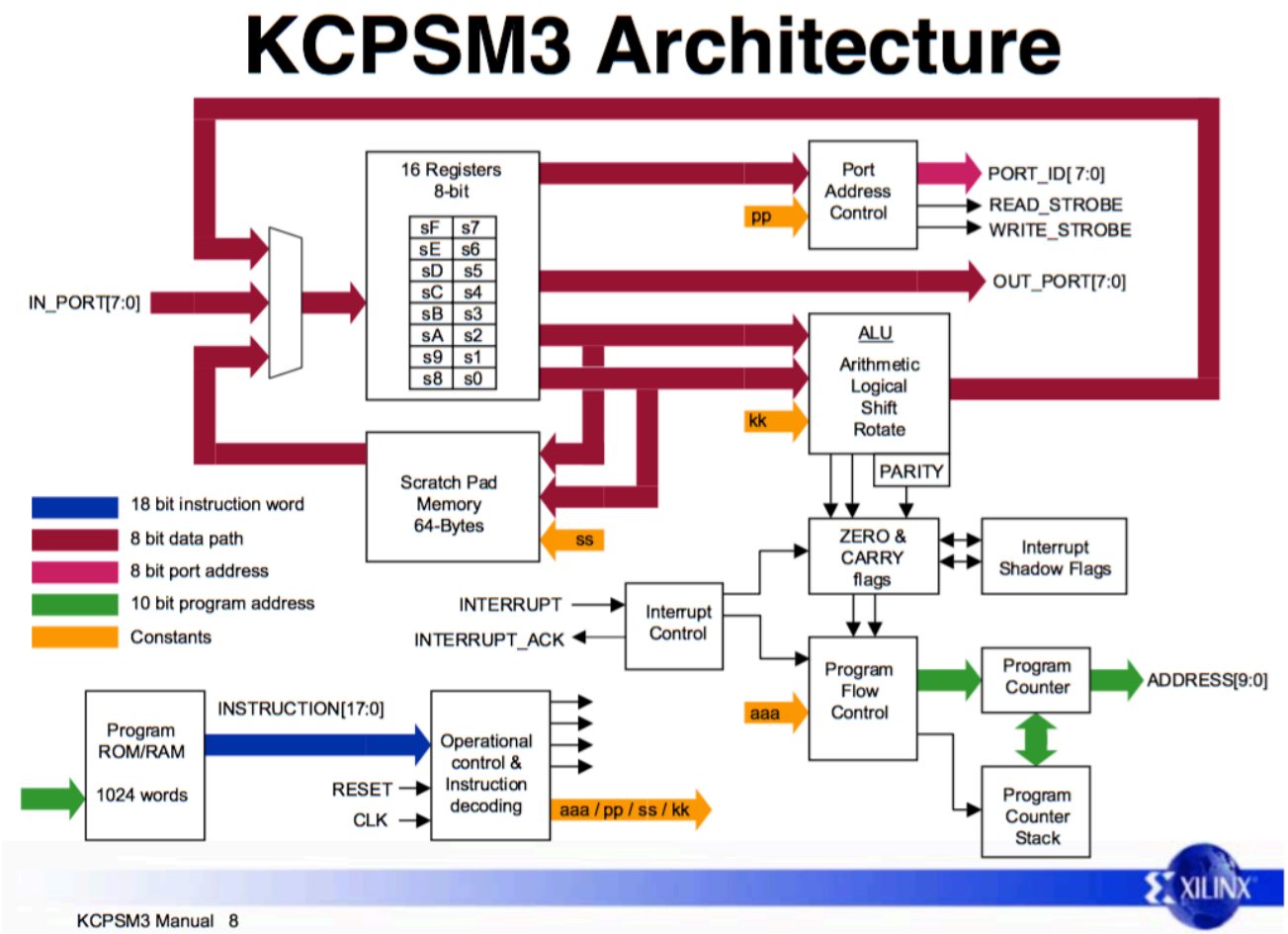
Lab. de Sistemas Digitais 2

Aula 6 – Microcontrolador PicoBlaze

Arquitetura



Instruções

KCPSM3 Instruction Set

'X' and 'Y' refer to the definition of the storage registers 's' in the range 0 to F.
'kk' represents a constant value in the range 00 to FF.
'aaa' represents an address in the range 000 to 3FF.
'pp' represents a port address in the range 00 to FF.
'ss' represents an internal storage address in the range 00 to 3F.

Program Control Group

```
JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa

CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL C,aaa
CALL NC,aaa

RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC
```

Note that call and return supports up to a stack depth of 31.

Arithmetic Group

```
ADD sX,kk
ADDCY sX,kk
SUB sX,kk
SUBCY sX,kk
COMPARE sX,kk

ADD sX,sY
ADDCY sX,sY
SUB sX,sY
SUBCY sX,sY
COMPARE sX,sY
```

Interrupt Group

```
RETURNI ENABLE
RETURNI DISABLE

ENABLE INTERRUPT
DISABLE INTERRUPT
```

Logical Group

```
LOAD sX,kk
AND sX,kk
OR sX,kk
XOR sX,kk
TEST sX,kk

LOAD sX,sY
AND sX,sY
OR sX,sY
XOR sX,sY
TEST sX,sY
```

Storage Group

```
STORE sX,ss
STORE sX,(sY)
FETCH sX,ss
FETCH sX,(sY)
```

Shift and Rotate Group

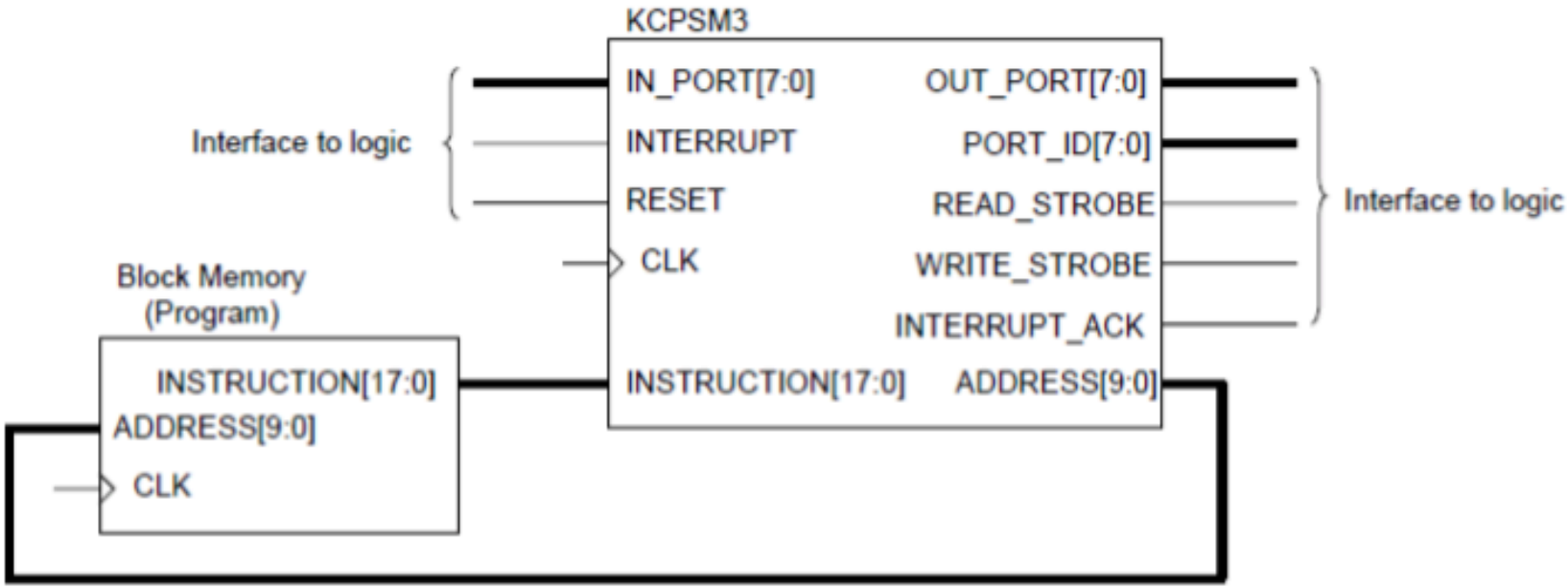
```
SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX

SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX
```

Input/Output Group

```
INPUT sX,pp
INPUT sX,(sY)
OUTPUT sX,pp
OUTPUT sX,(sY)
```

Componentes



Fluxo de Projeto

- Escrever um programa em **Assembly** no editor de texto com extensão **.psm**
- Executar o **Assembler** no arquivo **.psm**. O componente **Block Memory** é gerado como arquivo .vhd
- **ISE**: Instanciar os componentes **KCPSM3** e **Block Memory**
- Mapear os pinos e programar a placa

Parte 1 - PicoBlaze

- Baixar o arquivo **KCPSM3.zip** (**Moodle**) e descompactar

Assembler	4/08/2005 4:51 PM	File folder	
DATA2MEM_assistance	4/08/2005 4:51 PM	File folder	
JTAG_loader	4/08/2005 4:51 PM	File folder	
Verilog	4/08/2005 4:51 PM	File folder	
VHDL	4/08/2005 4:50 PM	File folder	
kcpsm3.ngc	15/06/2004 1:59 PM	NGC File	50 KB
KCPSM3_Manual.pdf	10/10/2003 4:06 PM	Adobe Acrobat D...	609 KB
read_me.txt	4/08/2005 4:56 PM	Text Document	22 KB
UART_Manual.pdf	23/04/2003 10:46 ...	Adobe Acrobat D...	111 KB
UART_real_time_clock.pdf	7/10/2003 4:27 PM	Adobe Acrobat D...	316 KB

Parte 1 - PicoBlaze

- Criar um diretório de trabalho (ex: **tutorial**) e copiar nele os seguintes arquivos:
  - Origem: pasta **Assembler**
    - KCPSM3.EXE
    - ROM_form.coe
    - ROM_form.v
    - ROM_form.vhd
  - Origem: pasta **VHDL**
    - kcpsm3.vhd

Parte 1 - PicoBlaze

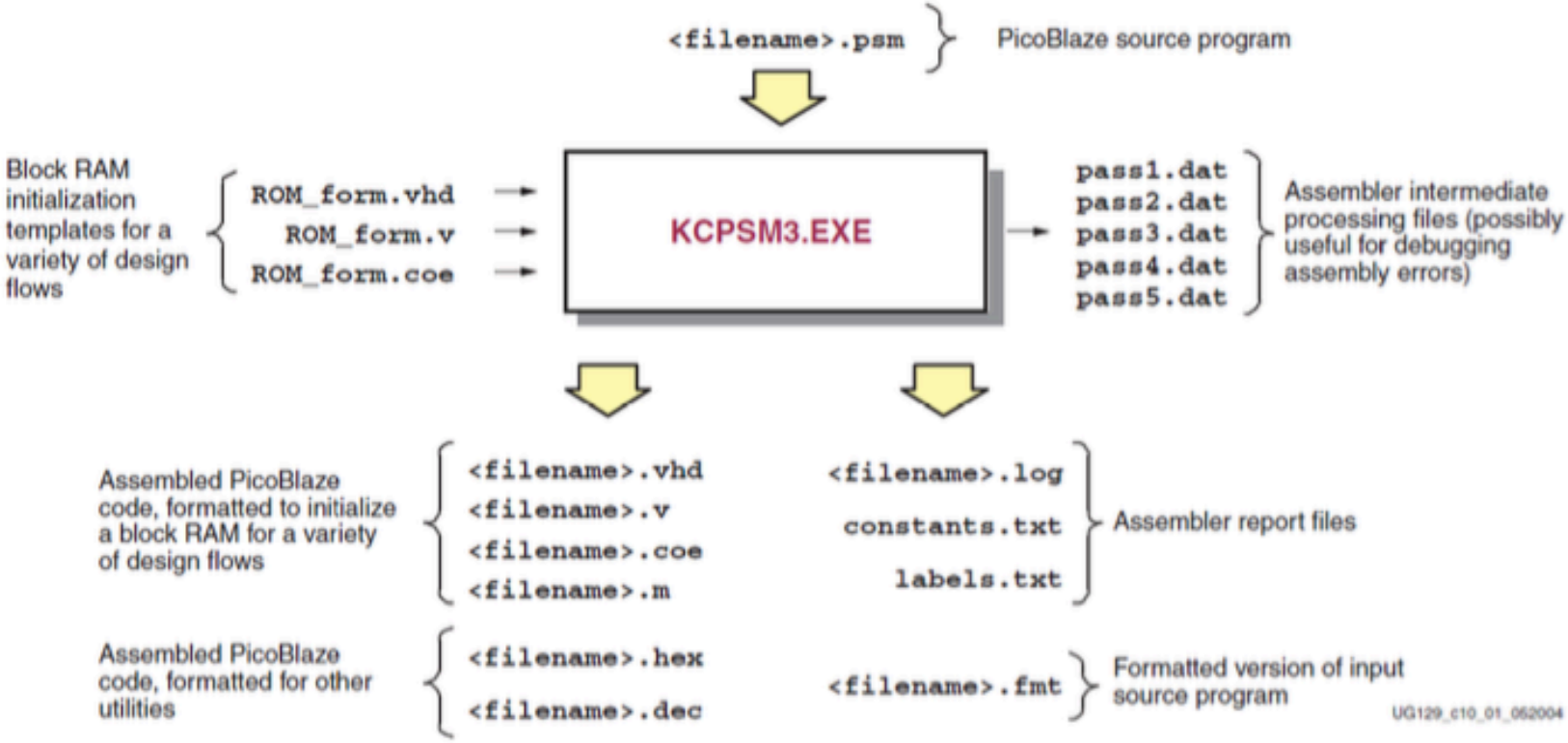
- Abrir o editor de texto, digitar o programa abaixo e salvar como **tutorial.psm** no diretório de trabalho:

```
; Simple loop that puts contents of input register
; into the output register
;
; switches DSIN $00
; LEDS DSOUT $80

start: INPUT s0, 00     ; read switches into register s0
       OUTPUT s0, 80    ; write contents of s0 to output port 80 - leds.
       JUMP start       ; loop back to start
```

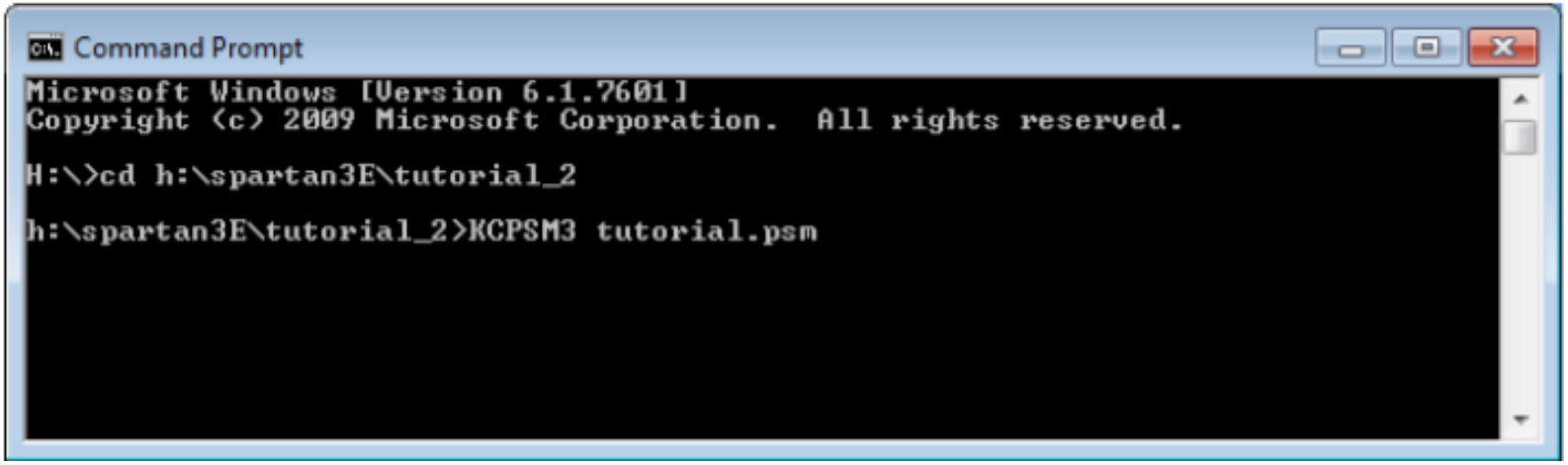
Parte 1 - PicoBlaze

- Executar o **Assembler**



Parte 1 - PicoBlaze

- O **Assembler** pode ser executado na janela *DOS Command Prompt*
- Escolher a versão do executável de acordo com o sistema operacional (**32** ou **64 bits**)



Parte 2 - ISE

- Criar um novo projeto no **ISE Project Navigator**
- Configurar para a placa **Spartan-3** ou **Nexys2**
- Em **Project -> Add Source**, selecionar os arquivos **tutorial.vhd** e **kcpsm3.vhd** gerados no diretório de trabalho anterior
- Abrir os arquivos e verificar as entidades de ambos

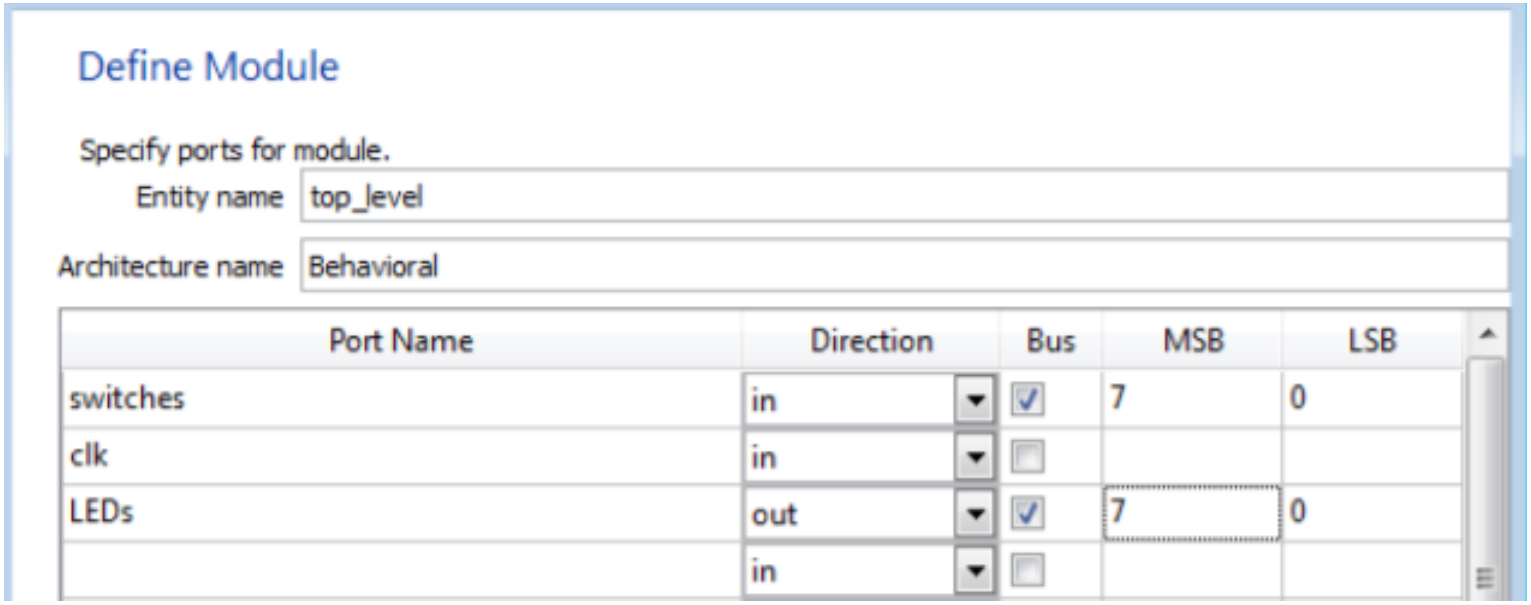
Parte 2 - ISE

```
20  entity tutorial is
21      Port (      address : in std_logic_vector(9 downto 0);
22              instruction : out std_logic_vector(17 downto 0);
23                      clk : in std_logic);
24      end tutorial;
```

```
77  entity kcpsm3 is
78      Port (      address : out std_logic_vector(9 downto 0);
79              instruction : in std_logic_vector(17 downto 0);
80                  port_id : out std_logic_vector(7 downto 0);
81             write_strobe : out std_logic;
82                 out_port : out std_logic_vector(7 downto 0);
83              read_strobe : out std_logic;
84                  in_port : in std_logic_vector(7 downto 0);
85               interrupt : in std_logic;
86           interrupt_ack : out std_logic;
87                   reset : in std_logic;
88                     clk : in std_logic);
89      end kcpsm3;
```

Parte 2 - ISE

- Criar um módulo VHDL **top level** e definir o **clk**, as 8 chaves e os 8 LEDs como entradas e saídas



Parte 2 - ISE

- Editar a arquitetura do **top level**

```
architecture Behavioral of top_level is
-- declaration of KCPSM3 (always use this declaration to call
-- up PicoBlaze core)
component kcpsm3
  port (address        : out std_logic_vector(9 downto 0);
        instruction    : in std_logic_vector(17 downto 0);
        port_id        : out std_logic_vector(7 downto 0);
        write_strobe   : out std_logic;
        out_port       : out std_logic_vector(7 downto 0);
        read_strobe    : out std_logic;
        in_port        : in std_logic_vector(7 downto 0);
        interrupt      : in std_logic;
        interrupt_ack  : out std_logic;
        reset          : in std_logic;
        clk            : in std_logic);
end component;
-----------------------------------------------------------------------
```

Parte 2 - ISE

- Editar a arquitetura do **top level**

```
-- declaration of program memory (here you will specify the entity name
-- as your .psm prefix name)
component tutorial
  port (address     : in std_logic_vector(9 downto 0);
        instruction : out std_logic_vector(17 downto 0);
        clk         : in std_logic);
end component;
-----------------------------------------------------------------------
```

- Criar os sinais e, após o **begin**, instanciar os componentes e criar os processos de leitura/escrita das portas

Parte 2 - ISE

```
-- Signals used to connect PicoBlaze core to program memory and I/O logic
signal address : std_logic_vector(9 downto 0);
signal instruction : std_logic_vector(17 downto 0);
signal port_id : std_logic_vector(7 downto 0);
signal out_port : std_logic_vector(7 downto 0);
signal in_port : std_logic_vector(7 downto 0);
signal write_strobe : std_logic;
signal read_strobe : std_logic;
signal interrupt_ack : std_logic;
signal reset : std_logic;
-- the following input is assigned an inactive value since it is
-- unused in this example
signal interrupt : std_logic :='0';
-----------------------------------------------------------------------
```

Parte 2 - ISE

```
-- Instantiating the PicoBlaze core
processor: kcpsm3
  port map (address => address,
            instruction => instruction,
            port_id => port_id,
            write_strobe => write_strobe,
            out_port => out_port,
            read_strobe => read_strobe,
            in_port => in_port,
            interrupt => interrupt,
            interrupt_ack => interrupt_ack,
            reset => reset,
            clk => clk);

-- Instantiating the program memory
program: tutorial
  port map (address => address,
            instruction => instruction,
            clk => clk);
```

Parte 2 - ISE

```
------------------------------------------------------------------------
---- KCPSM3 Define input ports
------------------------------------------------------------------------
-- The inputs connect via a pipelined multiplexer
input_ports: process(clk)
begin
  if clk'event and clk='1' then
    case port_id(1 downto 0) is
         -- read simple toggle switches and buttons at address 00 hex
         when "00" =>
              in_port <= switches;
         -- Don't care used for all other addresses to ensure minimum
         -- logic implementation
         when others =>
              in_port <= "XXXXXXXX";
       end case;
  end if;
end process input_ports;
```

Parte 2 - ISE

```
-----------------------------------------------------------------------
-- KCPSM3 Define output ports
-----------------------------------------------------------------------
-- adding the output registers to the processor at address 80 hex
output_ports: process(clk)
begin
  if clk'event and clk='1' then
    if port_id(7)='1' then
         LEDS <= out_port;
        end if;
  end if;
end process output_ports;

end Behavioral;
```

Parte 2 - ISE

- Mapear os pinos de entrada e saída no arquivo **.ucf**, sintetizar e programar a placa
- Testar o funcionamento do programa acionando as chaves

Referência: Banks, Jasmine, ”**The Spartan-3E Tutorial 2: Introduction to using the PicoBlaze Microcontroller**”, Queensland University of Technology, 2012.

Sugestão

- Ver as **vídeo-aulas** e executar os exemplos
  - https://www.youtube.com/watch?v=-USZXC334b8&index=19&list=PLKIWpQ56tY7KeqdSf36lrdsvTm2TGvdFq
  - https://www.youtube.com/watch?v=G4w2BqStTaQ&index=21&list=PLKIWpQ56tY7KeqdSf36lrdsvTm2TGvdFq
  - https://www.youtube.com/watch?v=omDaYbzicN8&index=22&list=PLKIWpQ56tY7KeqdSf36lrdsvTm2TGvdFq