PCS3225 - Sistemas Digitais II
Trabalho 5 - Processador Monociclo PoliLEG

Edson S. Gomi

16/11/2019

Nos trabalhos anteriores, você projetou a memória de instruções (`rom`), a memória de dados (`ram`), o banco de registradores (`regfile`), a Unidade Lógico-Aritmética (`alu`), a unidade funcional `signExtend` e a Unidade de Controle (`controlunit`) do processador monociclo PoliLEG. O objetivo deste trabalho é completar o PoliLEG, construindo co fluxo de dados (`datapath`) e interligá-lo com a unidade de controle. O teste do processador será feito através da execução de um programa instalado na `rom`, junto com os dados colocados na `ram`.

Introdução

Conforme foi explicado no trabalho anterior, o processador PoliLEG usa o paradigma de Unidade de Controle e Fluxo de Dados. Na Figura 1 podemos observar o diagrama de blocos, com as interligações entre a unidade de controle e o fluxo de dados. É importante lembrar que as memórias de instruções e de dados são componentes externos ao processador.

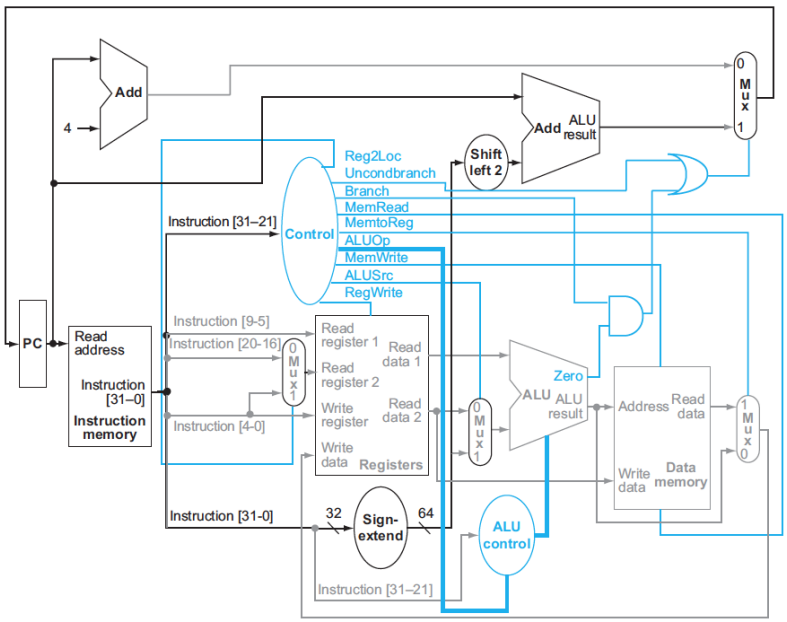

Figura 1: Diagrama do PoliLEG

Para facilitar o projeto dos circuitos deste trabalho, será bom rever o funcionamento e o formato das instruções do PoliLEG. A leitura recomendada são os Capítulos 2 (Seções 2.1 a 2.10), 4 (Seções 4.1 a 4.4) e o Apêndice A5 do livro texto da disciplina (*Computer Organization and Design - The Hardware/Software Interface, ARM Edition*, de David Patterson & John Hennessy).

Atividades

Trabalho 5, Atividade 1

T4A1 (10 pontos) Para implementar o PoliLEG, siga os passos explicados a seguir.

(a) Inicialmente implemente um componente em VHDL correspondente ao fluxo de dados (`datapath`). O fluxo de dados é composto pelo banco de registradores (`regfile`), a Unidade Lógico-Aritmética (`alu`), a unidade funcional `signExtend` e o `Shiftleft2`. Uma sugestão para a entidade `datapath` é mostrada na Listagem 2.

```
entity datapath is
  port(
    -- Common
    clock : in bit;
    reset : in bit;
    -- From Control Unit
    reg2loc : in bit;
    pcsrc: in bit;
    memToReg: in bit;
    aluCtrl: in bit_vector(3 downto 0);
    aluSrc: in bit;
    regWrite: in bit;
    -- To Control Unit
    opcode: out bit_vector(10 downto 0);
    zero:   out bit;
    -- IM interface
    imAddr: out bit_vector(63 downto 0);
    imOut:  in  bit_vector(31 downto 0);
    -- DM interface
    dmAddr: out bit_vector(63 downto 0);
    dmIn:   out bit_vector(63 downto 0);
    dmOut:  in  bit_vector(63 downto 0)
  );
end entity datapath;
```

Figura 2: Entidade para Sign-extend

(b) No próximo passo, implemente o componente do PoliLEG, cuja entidade é denominada `polilegsc` e cuja descrição está na Figura 3. O `polilegsc` essencialmente contém a interligação do fluxo de dados `datapath` com a unidade de controle `controlunit`. Observe que as interligações entre o fluxo de dados e a unidade de controle correspondem aos sinais com cor azul na Figura 1. Também observe que as memórias `ROM` e `RAM` não fazerm parte do processador e, portanto, são interligadas externamente ao `polilegsc`.

São fornecidos os arquivos `rom.dat` e `ram.dat` que contém as instruções e os dados do programa que calcuda o Máximo Divisor Comum (MDC). Recomendamos usar o conteúdo desses arquivos `dat` para testar a sua implementação do `polilegsc`. Este programa implementa o algoritmo de Euclides, que está descrito na Figura 4.

```
entity polilegsc is
    port (
      clock, reset: in bit;
      -- Data Memory
      dmem_addr:        out     bit_vector(63 downto 0);
      dmem_dati:        out     bit_vector(63 downto 0);
      dmem_dato:        in      bit_vector(63 downto 0);
      dmem_we:  out     bit;
      -- Instruction Memory
      imem_addr:        out     bit_vector(63 downto 0);
      imem_data:        in      bit_vector(31 downto 0)
    );
end entity;
```

Figura 3: Entidade para Sign-extend

```
int MDC(int a, int b){
    while(a!=b){
        if(a>b) a = a-b;
        else b = b-a;
    }
    return a;
}
```

Figura 4: Entidade para Sign-extend

O conteúdo da `rom` corresponde ao conteúdo binário da implementação do algoritmo de Euclides em Assembly do PolLEG, conforme é mostrado na Figura 5. Esta implementação assume que a memória de dados contém os valores 80000000_{16}, a e b, respectivamente nas posições 0, 8 e 4 da memória de dados. Os registradores X0, X1, X2, X3, X4 e X5 contém os valores zero, 80000000_{16}, a, b, temp1 e temp2

```
    LDUR X1,[X0,0]      -- X1=80000000
    LDUR X2,[X0,4]      -- X2 = a
    LDUR X3,[X0,8]      -- X3 = b
L1: SUB X4,X2,X3        -- temp1 = a - b
    CBZ X4, L0          -- if (temp1 == 0) goto L0
    SUB X4,X2,X3        -- temp1 = a - b
    AND X5,X4,X1        -- temp2 = temp1 AND 80000000
    CBZ X5,L3           -- if (temp2 == 0) goto L3
    SUB X3,X3,X2        -- b = b - a
    B L1                -- goto L1
L3: SUB X2,X2,X3        -- a = a - b
    B L1                -- goto L1
L0: STUR X2,[X0,4]      -- a = X2
L2: B L2                -- loop eterno
```

Figura 5: Entidade para Sign-extend

Você deverá submeter ao Juiz o arquivo VHDL da descrição (entidade e arquitetura) do `polilegsc`. O arquivo VHDL deverá conter todos os componentes que utilizou para implementar o `polilegsc`. O Juiz executará outro programa, diferente do MDC, para avaliar a sua implementação.

Como aconteceu nos trabalhos anteriores, recomendamos criar um testbench para testar a sua implementação do `polilegsc` antes de submeter a sua solução ao Juiz.

Instruções para Entrega

Você deve acessar `https://pelicano.pcs.usp.br`, logar com o email cadastrado no Júpiter e enviar os seus dois arquivos, um de cada vez. Durante o envio do arquivo, será solicitada a *tag* do problema a ser resolvido, que está abaixo:

Note que os *tags* são diferentes.

Tarefa	Problema	*tag*
T4A1	PoliLEG	`POLEG`

O juiz corrigirá imediatamente sua submissão e retornará com a nota. Caso não esteja satisfeito, você pode enviar novamente e somente a última nota para aquele problema será válida. A nota para este trabalho é composta pela soma das notas dadas pelo juiz para cada atividade. Como sugestão, faça seu *testbench* e utilize um simulador de VHDL para validar sua solução antes de postá-la para o juiz.

A quantidade de submissões para estes atividades foi limitada a 10 por Problema

Atenção: não atualize a página de submissões e não envie a partir de conexões instáveis (e.g. móveis).