Arquitetura e Organização de Computadores
Turma C - 2016/02

Projeto do Controle do MIPS Multiciclo

Objetivo: projetar, simular e sintetizar a parte de controle do MIPS multiciclo, usando uma estrutura baseada na solução microprogramada.

Descrição: A estrutura vista em aula para implementação do controle microprogramado do MIPS multiciclo é ilustrada na figura 1. O sistema consiste basicamente em uma unidade de endereçamento e uma memória de microcódigo. A memória de microcódigo pode ser implementada na forma de um arranjo de constantes, onde cada entrada do arranjo contém uma microinstrução.

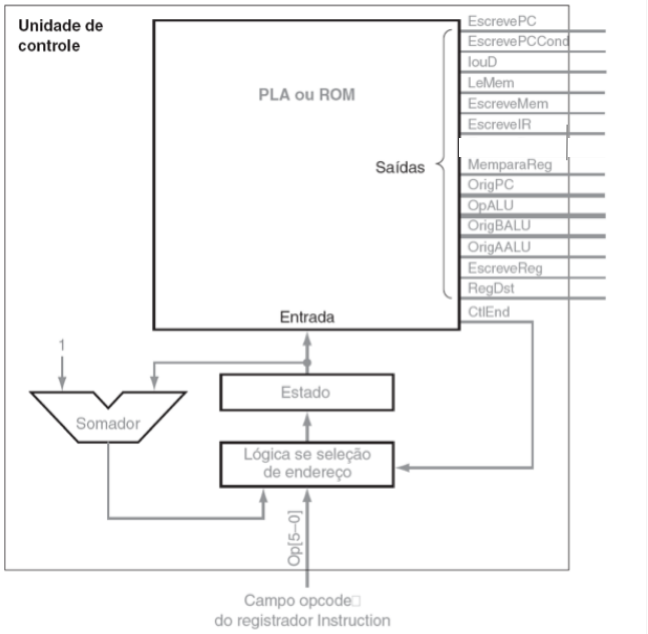


Figura 1. Estrutura de controle microprogramada.

A implementação da lógica de endereçamento é apresentada na figura 2. É composta por um registrador de microendereço ("Estado", na figura), um incrementador para gerar endeço seguinte, duas tabelas para geração de endereços e um multiplexador controlado pelo campo da microinstrução *CntrEnd*, que seleciona a próxima microinstrução da instrução que está sendo executada. A interface do controlador é apresentada abaixo, tendo como entrada o código da operação e como saída os campos da microinstrução (figura 3).

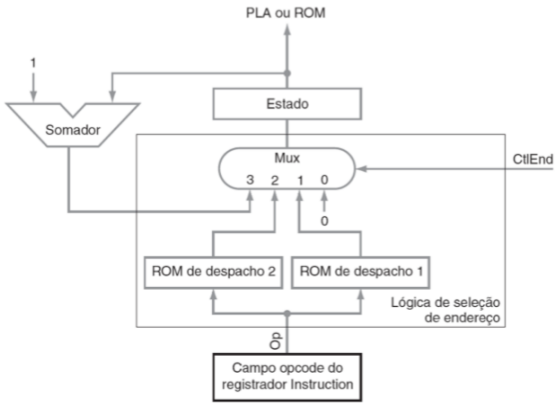
A implementação dever ser na forma estrutural, ou seja, cada componente da figura 1 é implementado em uma entidade à parte. Assim, somador, microPC, lógica de endereçamento e microcódigo são entidades instanciadas separadamente. O código a seguir descreve o unidade microprogramada na forma estrutural.

Interface:

```
entity cntrMIPS is
       port (
              clk                       : in std_logicç
              Op                        : in std_logic_vector(5 downto 0);
              OpALU, OrigBALU, OrigPC   : out std_logic_vector(1 downto 0);
              OrigAALU                  : out std_logic;
              EscreveReg, RegDst, MemparaReg, EscrevePC, EscrevePCCond, IouD,
              EscreveMem, EscreveIR     : out std_logic;
              CtlEnd                    : out std_logic_vector(1 downto 0));
end cntrMIPS;
```

ROM de despacho 1		
Op	Nome do opcode	Valor
000000	formato R	0110
000010	jmp	1001
000100	beq	1000
100011	lw	0010
101011	sw	0010

ROM de despacho 2		
Op	Nome do opcode	Valor
100011	lw	0011
101011	sw	0101



Número do estado	Ação do controle de endereço	Valor de CtlEnd
0	Usa o estado incrementado	3
1	Usa a ROM de despacho 1	1
2	Usa a ROM de despacho 2	2
3	Usa o estado incrementado	3
4	Substitui o número do estado por 0	0
5	Substitui o número do estado por 0	0
6	Usa o estado incrementado	3
7	Substitui o número do estado por 0	0
8	Substitui o número do estado por 0	0
9	Substitui o número do estado por 0	0

Figura 2. Lógica de endereçamento do microprograma.

Microprograma:

Label	ALU Control	SRC1	SRC2	Register Control	Memory	PCWrite Control	Sequencing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1
Mem1	Add	A	Extend				Dispatch 2
LW2					Read ALU		Seq
				Write MDR			Fetch
SW2					Write ALU		Fetch
Rformat1	Func Code	A	B				Seq
				Write ALU			Fetch
BEQ1	Subt	A	B			ALUOut-cond	Fetch
JUMP1						Jump Address	Fetch

FIGURA 5.7.3 O microprograma para a unidade de controle. Lembre-se de que os rótulos são usados para determinar os destinos para as operações de despacho. `Dispatch 1` realiza um jump baseado no IR para um rótulo terminado em 1, enquanto `Dispatch 2` realiza um jump baseado no IR para um rótulo terminado em 2.

Codificação: definir a ROM como um *array* de microinstruções. Sugestão de tipos de dados: criar um tipo para o vetor de sinais de controle, outro para o campo de endereçamento. A microinstrução é um *record* que contém dois campos, um de cada tipo. Definir os valores do campo de sequenciamento de acordo com as tabelas. Criar através de constantes cada uma das microinstruções que compõe o microprograma. O módulo de sequenciamento deve ser criado na forma estrutural, com uma entidade para cada tabela de despacho, uma para o multiplexador e

```
SUBTYPE microComandos_T is std_logic_vector(0 to 14);
SUBTYPE nextAddress_T is std_logic_vector(0 to 1);

TYPE microInstrucao_T is RECORD
  microCmds   : microComandos_T;
  nextAddress : nextAddress_T;
end RECORD;

TYPE microPrograma_T is array (0 to 9) of microInstrucao_T;
```

```
-- valores para o campo de sequenciamento
constant SEQ              : nextAddress_T := "11";
constant FETCH            : nextAddress_T := "00";
constant DISPATCH_1       : nextAddress_T := "01";
constant DISPATCH_2       : nextAddress_T := "10";

-- micro programa: listar os sinais de saida na ordem da figura
-- microinstrucao | ALU Cntr | SRC 1    | SRC 2    | Regs | PC Write | Seq
constant mFETCH  : microInstrucao_T := ("000000000000000", SEQ );
```

Verificação: para verificar o funcionamento da parte de controle, deve-se prover o código das operações e verificar se os sinais de controle gerados seguem a sequência desejada.

Entrega: 30 de novembro.

Campos da Microinstrução:

Nome do campo	Valor	Sinais ativos	Comentário
Controle da ALU	Add	OpALU = 00	Faz com que a ALU realize uma soma.
	Subt	OpALU = 01	Faz com que a ALU realize uma subtração; isso implementa a comparação para desvios.
	Func code	OpALU = 10	Usa o código de função da instrução para determinar o controle da ALU.
SRC1	PC	OrigAALU = 0	Usa o PC como a primeira entrada da ALU.
	A	OrigAALU = 1	O registrador A é a primeira entrada da ALU.
SRC2	B	OrigBALU = 00	O registrador B é a segunda entrada da ALU.
	4	OrigBALU = 01	Usa 4 como a segunda entrada da ALU.
	Extend	OrigBALU = 10	Usa a saída da unidade de extensão de sinal como a segunda entrada da ALU.
	Extshft	OrigBALU = 11	Usa a saída da unidade de deslocamento em dois bits como a segunda entrada da ALU.
Controle dos Registradores	Read		Lê dois registradores usando os campos rs e rt do IR como os números de registrador e colocando os dados nos registradores A e B.
	Write ALU	EscreveReg, RegDst = 1, MemparaReg = 0	Escreve num registrador usando o campo rd do IR como o número de registrador e o conteúdo de SaídaALU como os dados.
	Write MDR	EscreveReg, RegDst = 0, MemparaReg = 1	Escreve num registrador usando o campo rt do IR como o número de registrador e o conteúdo de MDR como os dados.
Controle da Memória	Read PC	LeMem, IouD = 0, EscreveIR	Lê a memória usando o PC como o endereço; escreve o resultado no IR (e no MDR).
	Read ALU	LeMem, IouD = 1	Lê a memória usando SaídaALU como o endereço; escreve o resultado no MDR.
	Write ALU	EscreveMem, IouD = 1	Escreve na memória usando SaídaALU como o endereço e o conteúdo de B como os dados.
Controle de Escrita no PC	ALU	OrigPC = 00, EscrevePC	Escreve a saída da ALU no PC.
	ALUOut-cond	OrigPC = 01, EscrevePCCond	Se a saída Zero da ALU estiver ativa, escreve o PC com o conteúdo do registrador SaídaALU.
	jump address	OrigPC = 10, EscrevePC	Escreve o PC com o endereço de jump da instrução.
Seqüenciamento	Seq	CtlEnd = 11	Escolhe a próxima microinstrução seqüencialmente.
	Fetch	CtlEnd = 00	Vai para a primeira microinstrução para iniciar uma nova instrução.
	Dispatch 1	CtlEnd = 01	Despacha usando a ROM 1.
	Dispatch 2	CtlEnd = 10	Despacha usando a ROM 2.

Figura 3. Campos da Microintrução.