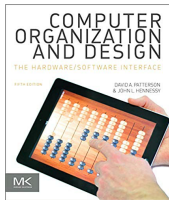
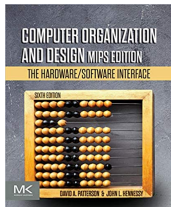


Aula 07 –Implementação Multiciclo

Prof. Dr. Clodoaldo A. de Moraes Lima

Material baseado no livro “Patterson, David A., Hennessy, J. L. - Computer Organization And Design: The Hardware/Software Interface”



Por que Multiciclo?

- Na implementação de ciclo único toda instrução opera em 1 clock de uma duração fixa.
- O ciclo de clock precisa ter a mesma duração para cada instrução. O ciclo de clock é determinado pelo caminho mais longo da máquina.
- Nesse caso, consideramos que o ciclo de clock é igual ao atraso do pior caso para todas as instruções;
- O desempenho geral de uma implementação de ciclo único provavelmente não será muito bom, já que várias classes de instrução poderiam ficar em um ciclo de clock mais curto.
- Portanto, a implementação de ciclo único é ineficiente tanto em seu desempenho quanto em seu custo de hardware (unidades funcionais duplicadas);

Uma Implementação Multiciclo

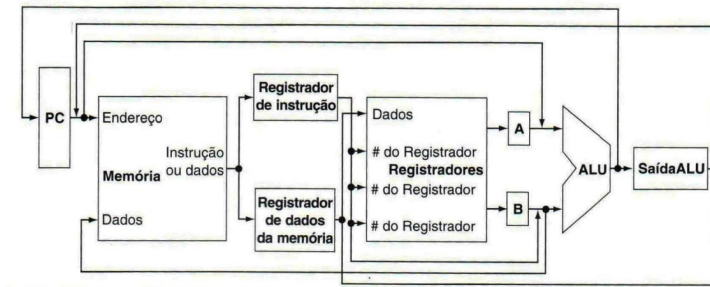
- Se dividirmos cada instrução em **uma série de etapas** correspondentes às operações das unidades funcionais necessárias, podemos usar essas etapas para criar uma implementação multiciclo.
- Em uma implementação multiciclo, cada etapa da execução levará 1 ciclo de clock.
- A implementação multiciclo permite que uma unidade funcional seja usada mais de uma vez por instrução, desde que seja usada em diferentes ciclos de clock.

Tempo para cada instrução

Classe de instrução	Memória de instrução	Leitura de registrador	Operação ALU	Memória de dados	Leitura de registrador	Total
Tipo R	200	50	100	0	50	400ps
Load word	200	50	100	200	50	600ps
Store word	200	50	100	200		550ps
Branch	200	50	100	0		350ps
Jump	200					200ps

Versão abstrata de uma implementação Multiciclo

Vejamos a visão alto nível de um caminho de dados multiciclo:



Uma implementação Multiciclo

Considerações:

- Uma única unidade de memória é usada para instruções e para dados;
- Existe uma única ULA, em vez de uma ULA e dois somadores;
- Um ou mais registradores são adicionados após cada unidade funcional para conter a saída dessa unidade até o valor a ser usado em um ciclo de clock subsequente;
- No final de um ciclo de clock, todos os dados usados nos ciclos de clock subsequentes precisam ser armazenados em um elemento de estado visível ao programador: banco de registradores, o PC ou a memória;

Considerações:

- Por outro lado, os dados usados pela mesma instrução em um ciclo de clock posterior precisam ser armazenados em um desses registradores adicionais;
- O registrador IR e o registrador de dados da memória (MDR) são incluídos para salvar a saída da memória para uma leitura de instrução e uma leitura de dados respectivamente;
- Os registradores A e B são usados para conter os valores dos registradores operandos lidos do banco de registradores;
- O registrador SaídaALU contém a saída da ULA;

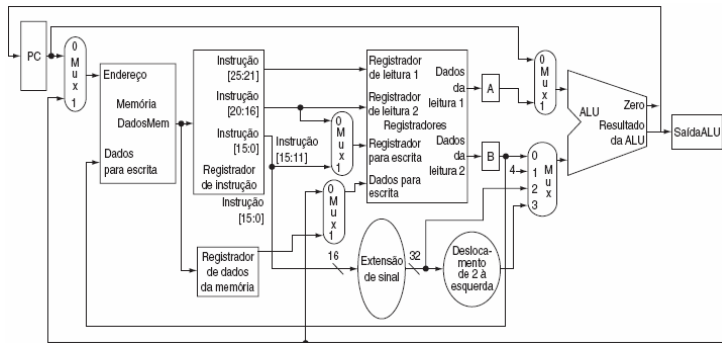
Uma implementação Multiciclo

Considerações:

- Todos os registradores exceto o IR contém dados apenas entre um par de ciclos de clock adjacente, e portanto, não precisarão de um sinal de controle de escrita;
- Como várias unidades funcionais são compartilhadas para diferentes finalidades, precisamos de ambos: incluir multiplexadores e expandir os multiplexadores existentes;
- Vejamos a figura que mostra os detalhes do caminho de dados com os multiplexadores adicionais:

Uma implementação Multiciclo

Caminho de dados multiciclo para o MIPS manipular as instruções básicas

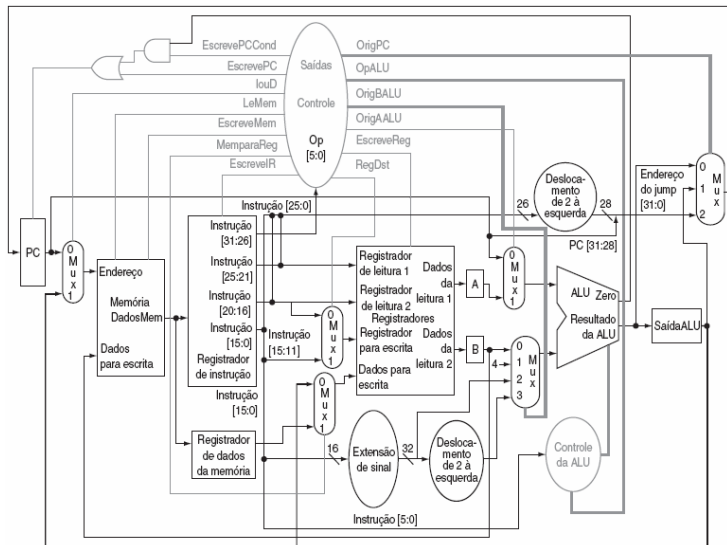


Uma implementação Multiciclo

O caminho de dados mostrado na figura anterior exigirá diferentes sinais de controle;

- As unidades de estado visíveis aos programador (o PC, a memória e os registradores), bem como o IR, precisarão de sinais de controle de escrita;
- A memória também precisará de um sinal de leitura;
- Cada multiplexador de quatro entradas exige duas linhas de controle;
- Vejamos o caminho de dados completo para a implementação multiciclo:

Uma implementação Multiciclo com sinal de controle



Dado o caminho de dados, vejamos o que deve acontecer em cada ciclo de clock da execução multiciclo:

- Busca da Instrução;
- Decodificação da instrução e busca dos registradores;
- Execução, cálculo do endereço de memória ou conclusão do desvio;
- Acesso à memória ou conclusão da instrução tipo R;
- Etapa de escrita adiada (write-back)

Etapa 1: Busca da Instrução

- Use o PC para obter a instrução e colocá-la no registrador de Instrução;
- Incremente o PC em 4 e coloque o resultado novamente no PC;
- Descrição na RTL (Register-Transfer Language), vejamos:

$IR \leftarrow \text{Memory}[PC];$

$PC \leftarrow PC + 4;$

Etapa 2: Decodificação da Instrução e Busca dos registradores

- Leia os registradores rs e rt no caso de instruções que utilizem eles;
- Calcule o endereço de desvio no caso da instrução ser um branch;
- Vejamos em RTL:

$A \leftarrow \text{Reg}[\text{IR}[25:11]];$

$B \leftarrow \text{Reg}[\text{IR}[20:16]];$

$\text{ALUOut} \leftarrow \text{PC} + (\text{sign-extend}(\text{IR}[15:0] < 2));$

Etapa 3: Dependente da instrução

- A ULA executará uma das três funções a seguir com base no tipo de instrução, vejamos em RTL:
- Tipo R:
 $ALUOut \leftarrow A \text{ op } B$
- Referência à memória
 $ALUOut \leftarrow A + \text{sign-extend}(IR[15:0])$
- Branch
 $\text{if}(A==B) \text{ PC} \leftarrow ALUOut$

Etapas de Execução

Etapa 4: Tipo R ou acesso à memória

- Loads e stores acessam a memória:

$\text{MDR} \leftarrow \text{Memory}[\text{ALUOut}];$
ou
 $\text{Memory}[\text{ALUOut}] \leftarrow B;$

- Instruções do tipo R finalizam

$\text{Reg}[\text{IR}[15:11]] \leftarrow \text{ALUOut}$

Etapa 5: Conclusão da leitura da memória

- Load

$\text{Reg}[\text{IR}[20:16]] \leftarrow \text{MDR};$

Etapas de Execução

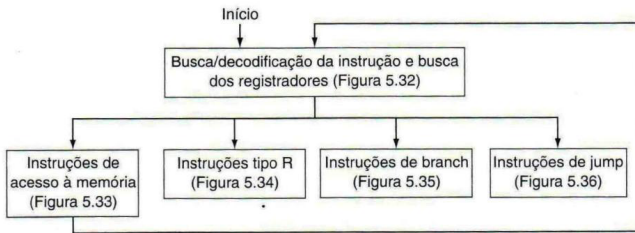
Resumo dos Passos Necessários para cada instrução

Nome do passo	Instrução tipo R	Instruções de referência à memória	Instrução de desvio condicional	Instrução de desvio incondicional
Busca da instrução	$RI = Mem[PC]$ $PC = PC + 4$		0	
Decodificação da instrução & leitura dos registradores Rs e Rt & cálculo do endereço de desvio (cond.)	$A = Reg [RI[25-21]]$ $B = Reg [RI[20-16]]$ $ULASaída = PC + (extensão de sinal(RI[15-0]) << 2)$		1	
Execução, cálculo do endereço de acesso à memória, término de uma instrução branch/jump	$ULASaída = A op B$	$ULASaída = A + extensão de sinal(RI[15-0])$	Se $(A == B)$ então $PC = ULASaída$	$PC = PC[31-28] + (RI[25-0] << 2)$
Término de uma instrução store word ou de tipo R	$Reg [RI[15-11]]$ $ULASaída$	$lw: RDM = Mem [ULASaída]$ $sw: Mem [ULASaída] = B$		
Término de uma instrução load word		$lw: Reg[RI[20-16]] = RDM$		
Número de passos	4	$lw: 5$ $sw: 4$	3	3

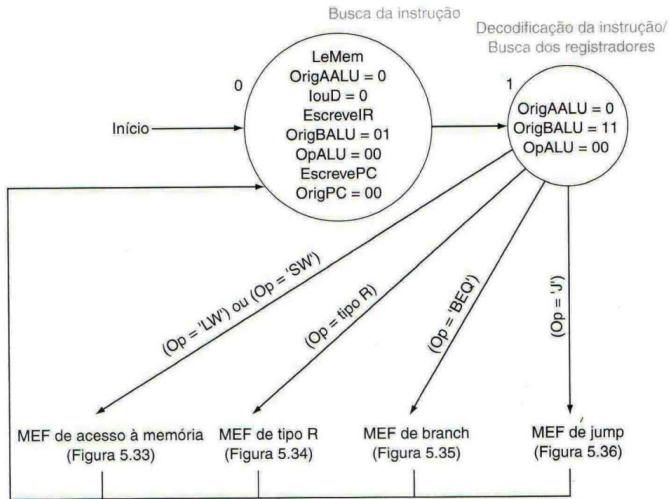
Implementação do controle

- No monociclo utilizamos um conjunto de tabelas-verdade;
- Na implementação de controle no projeto multiciclo veremos 2 abordagens:
 - Máquina de estados finitos;
 - Microprogramação

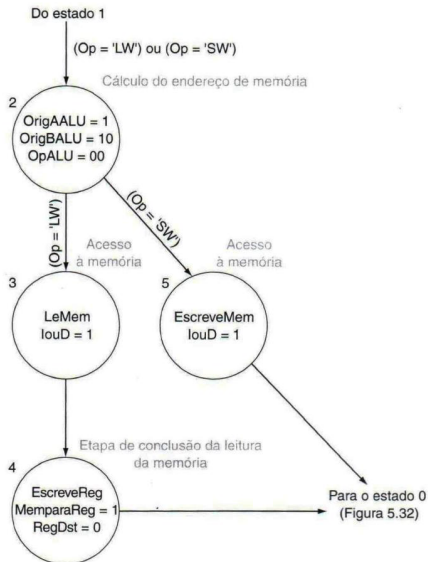
Controle Máquina de Estados Finitos



Controle Máquina de Estados Finitos - Busca da Instrução



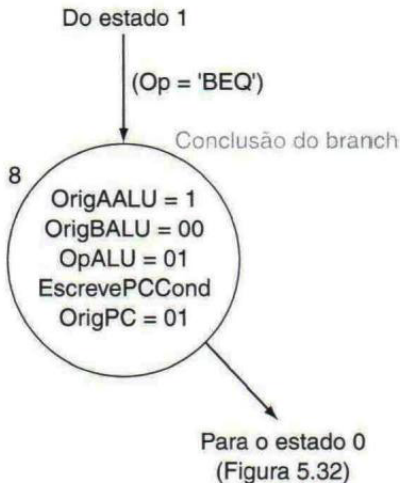
Controle Máquina de Estados Finitos - Instrução lw,sw



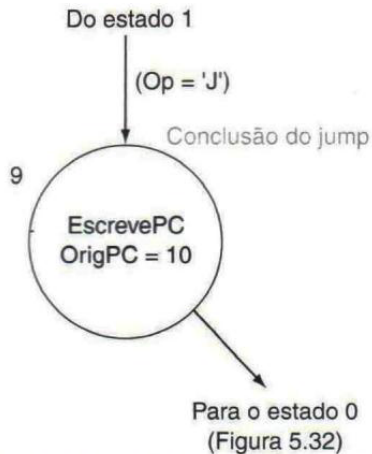
Controle Máquina de Estados Finitos - Instrução tipo R



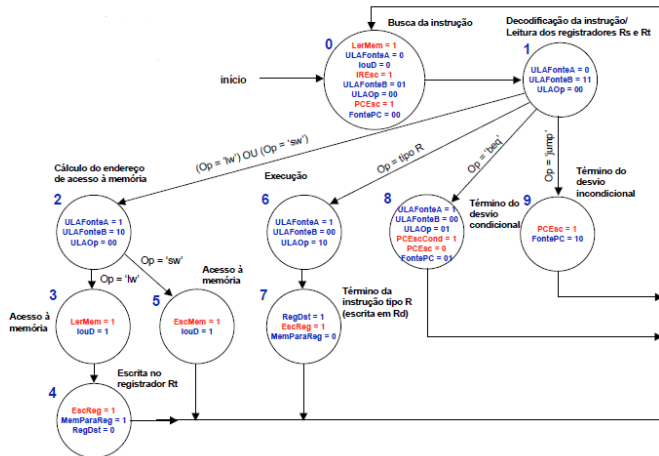
Controle Máquina de Estados Finitos - Instrução branch



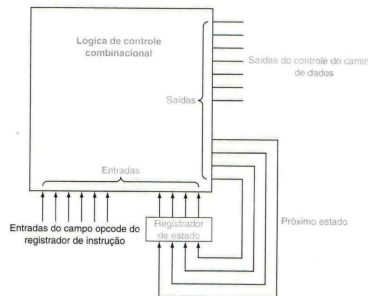
Controle Máquina de Estados Finitos - Instrução jump



Máquina de Estados Finitos Completa



Máquina de Estados Finitos Completa



Projeto do Bloco de Controle com Máquina de Estados

- Adequado no caso de máquinas com um número de estados moderado
- É possível montar as tabelas-verdade e encontrar as equações lógicas sem introduzir erro

Projeto do Bloco de Controle com Microprogramação

- O conjunto completo de instruções do MIPS possui cerca de 100 instruções
- Instruções gastam de 1 a 20 ciclos de relógio para executar
- Controle será bastante complexo (sobretudo se comparado ao controle projetado na aula passada)
- Em alguns processadores reais o conjunto de instruções apresenta muitas classes (tipicamente, arquiteturas CISC):
 - O controle pode precisar de milhares de estados
 - Com centenas de diferentes seqüências
- Pode haver um grande número de opcodes válidos

Projeto do Bloco de Controle com Microprogramação

- Nos casos citados, é pouco prático especificar o controle de forma gráfica:
 - Milhares de estados
 - Milhares de arcos
- Um diagrama de estados que não cabe em uma página não é inteligível

- Cada microinstrução define os valores dos sinais do bloco operativo para um dado estado da máquina de estados
- Executar uma microinstrução significa ativar os sinais de controle especificados pela microinstrução
- O seqüenciamento das microinstruções :
 - Execução seqüencial é o default
 - Um desvio precisa ser indicado explicitamente
- O mecanismo default normalmente é implementado por um contador (seqüenciador)

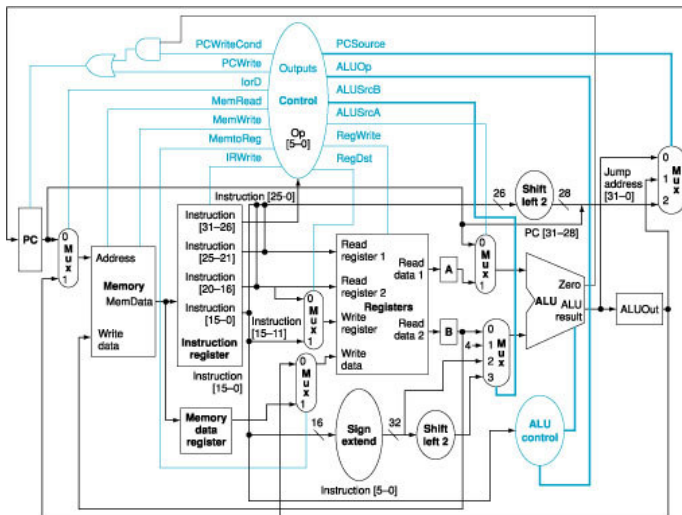
Definição

É o projeto do controle como um programa composto de estruturas mais simples denominadas microinstruções

Idéia Básica

Representar simbolicamente os valores ativos das linhas de controle, de modo que o microprograma seja uma representação das microinstruções

Arquitetura Multiciclo



Formato de uma Microinstrução

- Escolher quantos campos e quais sinais de controle são afetados por cada campo
- Simplificar a representação
- (Se possível,) dificultar a escrita de microinstruções inconsistentes

Formato de uma Microinstrução

- Cada Microinstrução Contém 7 Campos:

Nome do Campo	Função do Campo
Controle da ULA	Especifica a operação a ser realizada na ULA neste ciclo de clock. Resultado sempre escrito em ULASaida
SRC1	Especifica a fonte do primeiro operando da ULA
SRC2	Especifica a fonte do segundo operando da ULA
Controle do registrador	Especifica uma leitura ou uma escrita no banco de registradores, e a fonte do valor a ser escrito
Memória	Especifica uma leitura ou uma escrita na memória, e a fonte. No caso de uma leitura, especifica o registrador-destino.
Controle do PCEsc	Especifica escrita no PC
seqüenciamento	Especifica como escolher a próxima microinstrução a ser executada

- Os sinais que nunca estão ativos ao mesmo tempo devem compartilhar um mesmo campo

Implementação Física

- As microinstruções são armazenadas em uma ROM ou uma PLA (Programmable Logic Array)
 - São configuráveis ao contrário da ROM
- Portanto, cada microinstrução tem um endereço
- Os endereços são sequenciais

Existem 3 possibilidades para escolha da próxima microinstrução a ser executada:

- 1. Incrementar o endereço da microinstrução corrente (indicado por “seq”)
- 2. Desviar para a microinstrução que começa a execução da próxima instrução (indicada por “Busca”, corresponde ao estado 0 da versão FSM)
- 3. Escolher a próxima microinstrução com base na entrada do bloco de controle (indicada por “despacho”)

Implementação do Microprograma

- Em geral, as operação de “despacho” são implementadas a partir de uma tabela que:
 - contém os endereços das microinstruções-alvo
 - é indexada pela entrada do bloco do controle
 - pode ser implementada em uma ROM ou uma PLA
- Podem existir diversas tabelas de “despacho”

Implementação do Microprograma

- Na implementação do controle microprogramado do MIPS iremos precisar de duas tabelas de “despacho”:
 - Uma para despachar a partir do estado 1
 - Outra para despachar a partir do estado 2
- Despacho i (onde i é o número da tabela a ser usada)

Formato de uma Microinstrução

Nome do Campo	Valores possíveis	Função do campo, com valor específico
identificação	Qualquer string	Usado para especificar nomes para controlar o seqüenciamento. Tais nomes devem terminar em "f" ou em "2" e são usados para despacho, usando uma tabela indexada pelos bits do opcode.
Controle da ULA	Soma	Faz com que a ULA some
	Subtração	Faz com que a ULA subtraia (usado para implementar a comparação para os desvios condicionais)
	Códigofunct	Usa o campo "funct" da instrução para definir o controle da ULA
SRC1	PC	Usa PC como entrada superior da ULA
	A	O Registrador A é a entrada superior da ULA
SRC2	B	O Registrador B é a entrada inferior da ULA
	4	Usa a constante 4 como entrada inferior da ULA
	Estendido	Usa a saída da unidade de extensão de sinal como entrada inferior da ULA
	Estendido_desloc	Usa a saída da unidade de deslocamento de 2 bits como entrada inferior da ULA

Formato de uma Microinstrução

Nome do Campo	Valores possíveis	Função do campo, com valor específico
Controle do registrador	Leitura	Lê dois registradores usando os campos Rs e Rt do RI, colocando seus valores nos registradores temporários A e B.
	Escrita a partir da ULA	Escreve no banco de registradores usando o campo Rd do RI; o dado a ser escrito se encontra no registrador temporário ULASaída.
	Escrita a partir do RDM	Escreve no banco de registradores usando o campo Rt do RI; o dado a ser escrito se encontra no RDM.
Memória	Leitura a partir do PC	Lê a memória usando o PC como endereço; o resultado é escrito no RI (e no RDM).
	Leitura a partir da ULA	Lê a memória usando o conteúdo de ULASaída como endereço; o resultado é escrito no RDM.
	Escrita a partir da ULA	Escreve na memória usando o conteúdo de ULASaída como endereço e o conteúdo de B como dado.
Controle do PCEsc	ULA	Escreve a saída da ULA no PC.
	ULASaída_Cond	Se o flag de zero da ULA estiver ativo, atualiza o PC com o conteúdo de ULASaída.
	Endereço de desvio incondicional	Atualiza o PC com o endereço de desvio incondicional.
Sequenciamento	Seq	Escolhe a próxima microinstrução sequencialmente.
	Busca	Desvia para a primeira microinstrução para iniciar nova instrução
	Despacho i	Despacha usando a ROM i(1 ou 2)

Criação de um Microprograma

- Situações em que um campo de microinstrução fica em branco:
 - Quando um campo que controla uma unidade funcional ou que faz com o estado seja escrito (como campo Memory ou campo ALU dest) esta em branco, nenhum sinal de controle deve ser ativado.
 - Quando um campo apenas especifica o controle de um multiplexador que determina a entrada para um unidade funcional, como campo SRC1, deixa-lo em branco significa que não nos importamos com a entrada para a unidade funcional (ou com a saída do multiplexador)

Criação de um Microprograma

- Busca da Instrução; PC+4
- Decodificação; cálculo do endereço-alvo (beq)

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Busca	Soma	PC	4		Leitura a partir do PC	ULA	Seq
	Soma	PC	Estendido_desloc	Leitura			despacho1

Analisando os campos da primeira microinstrução...

Campos	Efeito
Controle da ULA, SRC1, SRC2	Calcula PC+4 (o valor é também escrito em ULASaída, apesar de nunca ser lido deste registrador).
Memória	Busca instrução e carrega no RI.
Controle de EscPC	Faz com que a saída da ULA seja escrita no PC.
Seqüenciamento	Vai para a próxima microinstrução (seqüencial).

Criação de um Microprograma

- Busca da Instrução; PC+4
- Decodificação; cálculo do endereço-alvo (beq)

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Busca	Soma	PC	4		Leitura a partir do PC	ULA	Seq
	Soma	PC	Estendido_desloc	Leitura			despacho1

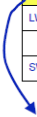
Analisando os campos da segunda microinstrução...

Campos	Efeito
Controle da ULA, SRC1, SRC2	Armazena PC+(extensão de sinal(RI[15:0] << 2) em ULASaida.
Controle dos registradores	Usa os campos Rs e Rt para ler os registradores, colocando os valores lidos em A e B.
Seqüenciamento	Usa a tabela de despacho 1 para encontrar o endereço da próxima microinstrução.

Criação de um Microprograma

• Execução de sw/lw

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Mem1	Add	A	Estendido				despacho2
LW2					Leitura a partir da ULA		Seq
				Escrita a partir do RDM			Busca
SW2					Escrita a partir da ULA		Busca




Campos	Efeito
Controle da ULA, SRC1, SRC2	Calcula o endereço de acesso à memória: Registrador(Rs) + extensão de sinal(RI[15-0]). Escreve o resultado em ULASaida.
Seqüenciamento	Usa a tabela de despachos 2 para para desviar para a microinstrução identificada por "LW2" ou "SW2".

Criação de um Microprograma

• Execução de sw/lw

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Mem1	Soma	A	Estendido				despacho2
LW2					Leitura a partir da ULA		Seq
				Escrita a partir do RDM			Busca
SW2					Escrita a partir da ULA		Busca




Campos	Efeito
Memória	Lê da memória usando ULASaída como endereço. Escreve o dado lido em RDM.
Seqüenciamento	Vai para a próxima microinstrução (seqüencial).

Criação de um Microprograma

• Execução de sw/lw

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Mem1	Soma	A	Estendido				despacho2
LW2					Leitura a partir da ULA		Seq
				Escrita a partir do RDM			Busca
SW2					Escrita a partir da ULA		Busca




Campos	Efeito
Controle dos registradores	Escreve o conteúdo do RDM na entrada do banco de registradores especificada por Rt.
Seqüenciamento	Desvia para a microinstrução identificada como "Busca".

Criação de um Microprograma

• Execução de sw/lw

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Mem1	Soma	A	Estendido				despacho2
LW2					Leitura a partir da ULA		Seq
				Escrita a partir do RDM			Busca
SW2					Escrita a partir da ULA		Busca




Campos	Efeito
Memória	Escreve na memória usando o conteúdo de ULASaida como endereço e o conteúdo de B como dado.
Seqüenciamento	Desvia para a microinstrução identificada como "Busca".

Criação de um Microprograma

• Execução formato R

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Rformat1	Código funct	A	B				Seq
				Escrita a partir da ULA			Busca




Campos	Efeito
Controle da ULA, SRC1, SRC2	ULA recebe como operandos os conteúdos dos registradores A e B. A operação a ser realizada é definida pelo campo "funct".
Seqüenciamento	Vai para a próxima microinstrução (seqüencial).

Criação de um Microprograma

• Execução formato R

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Rformat1	Código funct	A	B				Seq
				Escrita a partir da ULA			Busca



Campos	Efeito
Controle dos registradores	O Conteúdo de ULASaída é escrito no registrador cujo endereço está especificado pelo campo Rd.
Seqüenciamento	Desvia para a microinstrução identificada como "Busca".

Criação de um Microprograma

• Execução beq

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
BEQ1	Subtração	A	B			ULASaída_Cond	Busca



Campos	Efeito
Controle da ULA, SRC1, SRC2	ULA recebe como operandos os conteúdos dos registradores A e B. A operação a ser realizada é subtração.
Controle de escrita no PC	Faz com que o PC seja escrito usando o valor armazenado em ULASaída, se a saída zero da ULA for verdadeira.
Seqüenciamento	Desvia para a microinstrução identificada como "Busca".

Criação de um Microprograma

• Execução jump

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
JUMP1						Endereço de desvio incondicional	Busca



Campos	Efeito
Controle de escrita no PC	Faz com que o PC seja escrito com o valor do endereço-alvo de desvio incondicional.
Seqüenciamento	Desvia para a microinstrução identificada como "Busca".

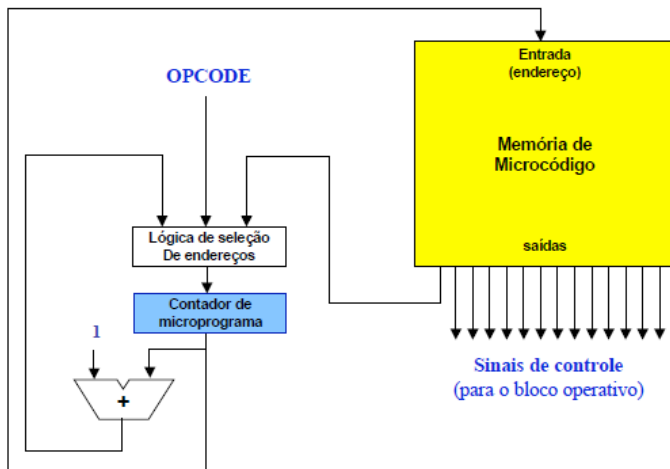
Microprograma Completo

identificação	Controle da ULA	SRC1	SRC2	Controle dos registradores	Memória	Controle de EscPC	seqüenciamento
Busca	Add	PC	4		Ler a partir do PC	ULA	Seq
	Add	PC	Estendido_desloc	Leitura			despacho1
Mem1	Add	A	Estendido				despacho2
LW2					Leitura a partir da ULA		Seq
				Escrita a partir do RDM			Busca
SW2					Leitura a partir da ULA		Busca
Rformat1	Código funct	A	B				Seq
				Escrita a partir da ULA			Busca
BEQ1	Subtração	A	B			ULASaida_Cond	Busca
JUMP1						Endereço de desvio incondicional	Busca

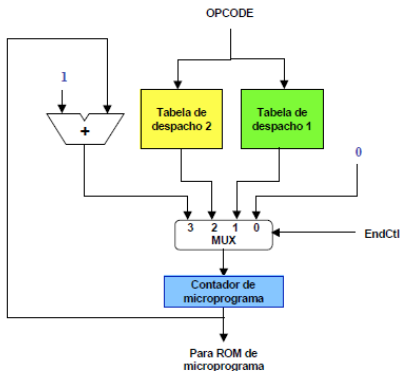
Implementação da Microprogramação

- O Mapeamento de um microprograma em hardware envolve dois aspectos:
 - A decisão sobre como implementar a função de seqüenciamento
 - A escolha do método para armazenar as funções do controle principal
- O microprograma pode ser imaginado como sendo uma representação textual de uma máquina de estados, com:

Implementação da Microprogramação



Lógica de Seleção de Endereços



Valor de EndCtl	ação
0	Vá para a busca (estado 0)
1	Tabela de despacho 1
2	Tabela de despacho 2
3	Use o estado incrementado