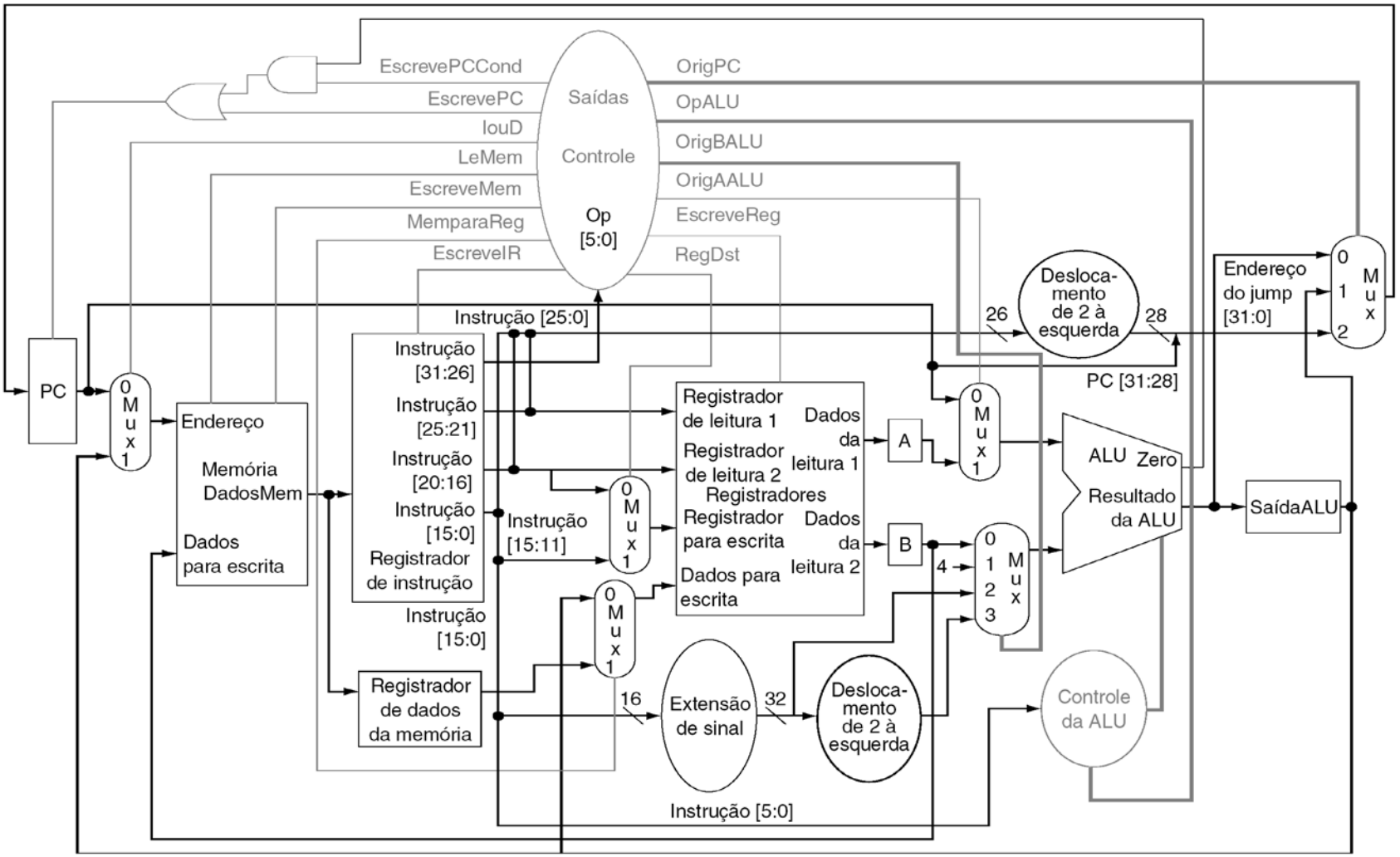
Universidade de Brasília
Departamento de Ciência da Computação

Aula 15
Implementação MIPS
MIPS Multiciclo – Unidade de Controle

Caminho de Dados Multiciclo

- Implementação do controle + controle do PC (PC+4, beq, jump)



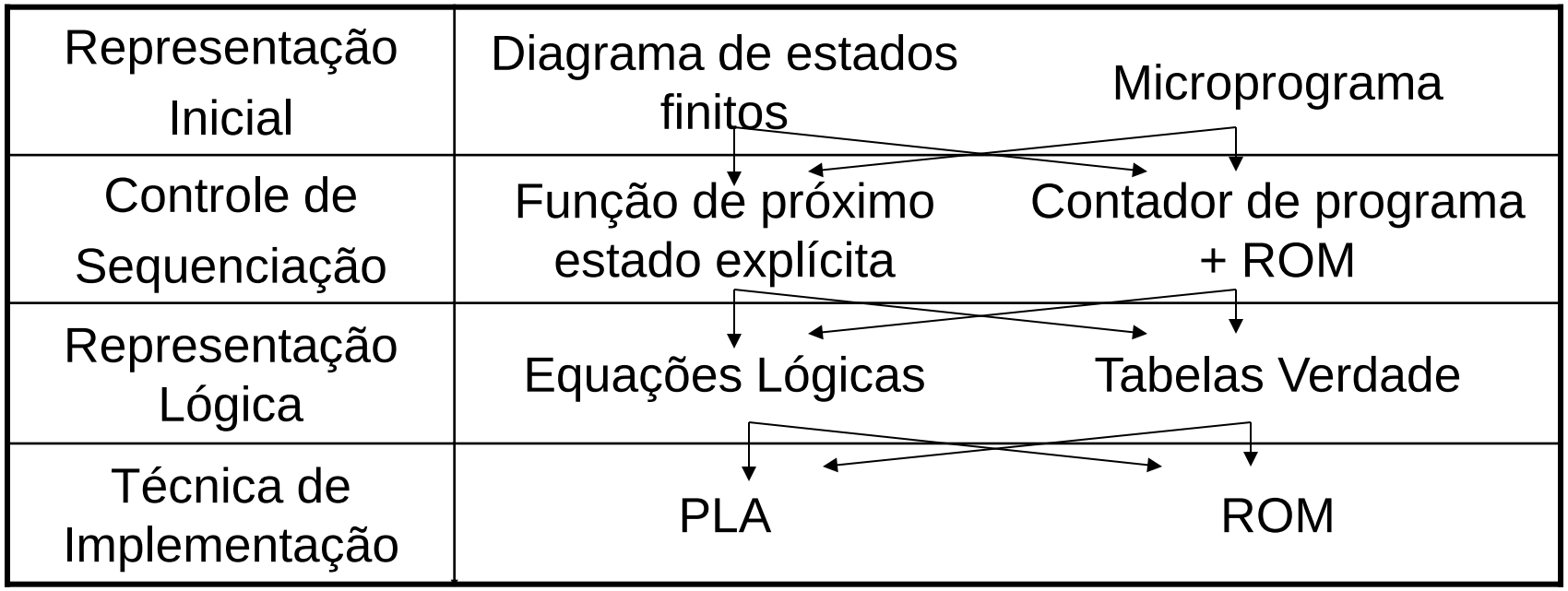
Resumo Controle Multiciclo

Etapa	Ação para instruções tipo R	Ação para instruções de acesso à memória	Ação para desvios	Ação para jumps
Busca da instrução	IR <= Memória[PC] PC <= PC + 4			
Decodificação da instrução e busca dos registradores	A <= Reg[IR[25:21]] B <= Reg[IR[20:16]] SaídaALU <= PC + (estende-sinal (IR[15:0]) << 2)			
Execução, cálculo do endereço, conclusão do desvio/jump	SaídaALU <= A op B	SaídaALU <= A + estende-sinal (IR[15:0])	if (A == B) PC <= SaídaALU	PC <= {PC [31:28], (IR[25:0]],2'b00)}
Acesso à memória ou conclusão de instrução tipo R	Reg[IR[15:11]]<= SaídaALU	Load: MDR <= Memória[SaídaALU] ou Store: Memória [SaídaALU] <= B		
Conclusão da leitura da memória		Load: Reg[IR[20:16]] <= MDR		

FIGURA 5.30 Resumo das etapas realizadas para executar qualquer classe de instrução. As instruções levam de três a cinco etapas de execu-

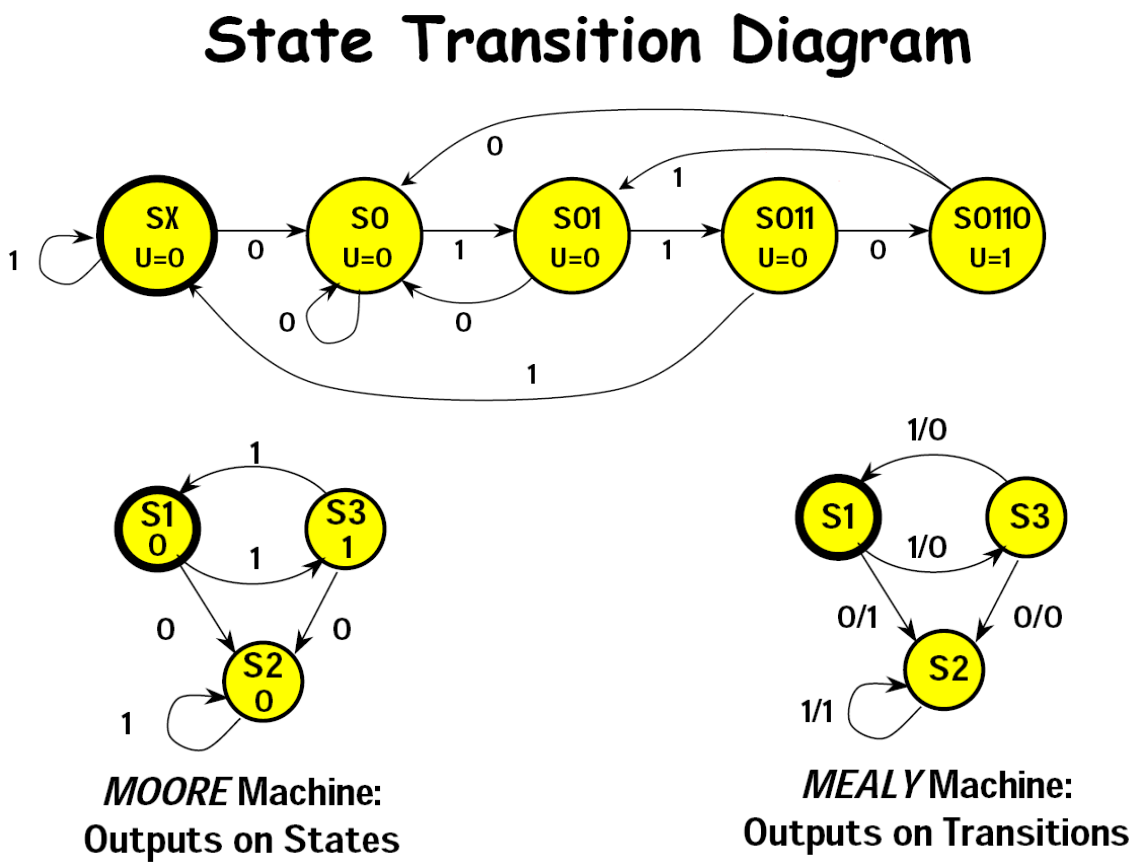
Projeto do Controle Multiciclo

- Controle feito em uma série de etapas
- Técnicas de Implementação:
  - Máquinas de Estado Finito
  - Microprogramação



Máquina de Estados Finitos - MEF

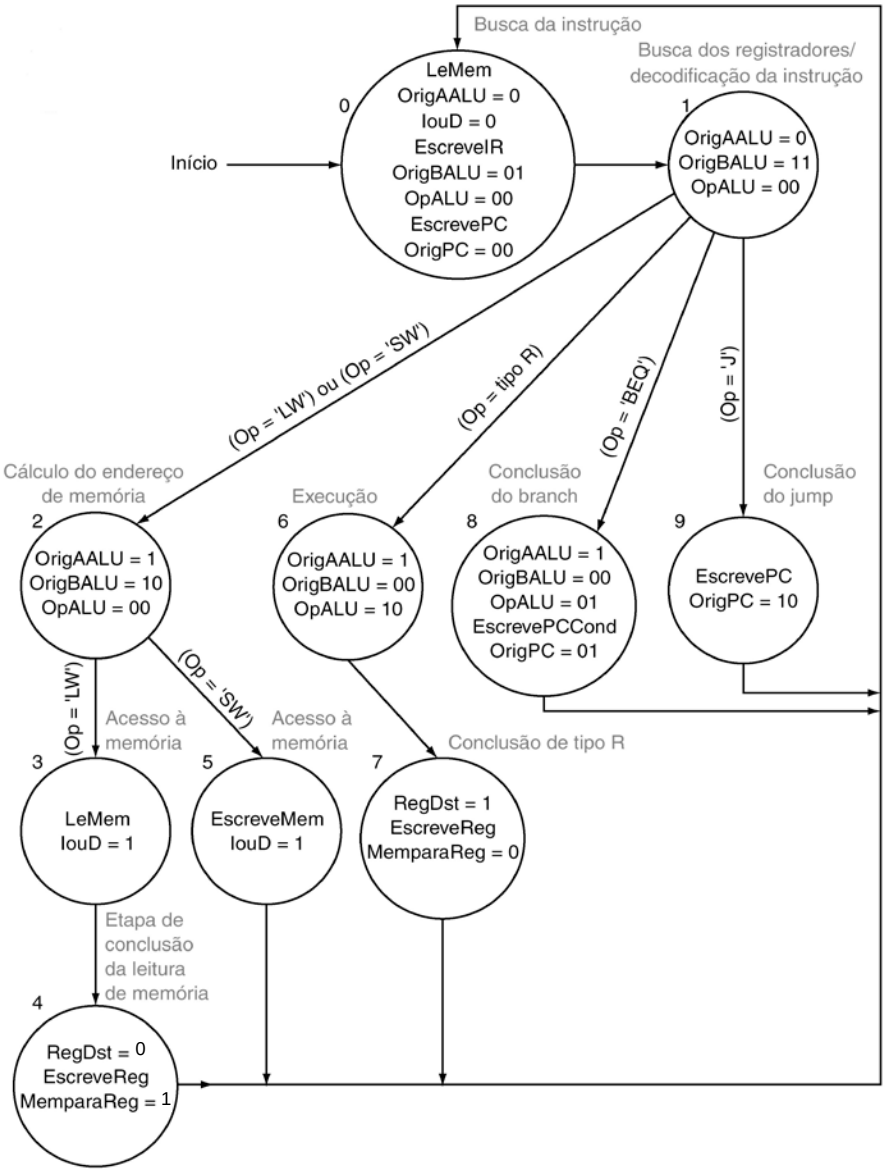
- **Diagrama de Estados**
- Cada nó do diagrama representa um estado
- A transição entre estados é indicada por arcos
- As condições de disparo de uma transição são associadas aos arcos
- Cada estado corresponde a um ciclo de relógio

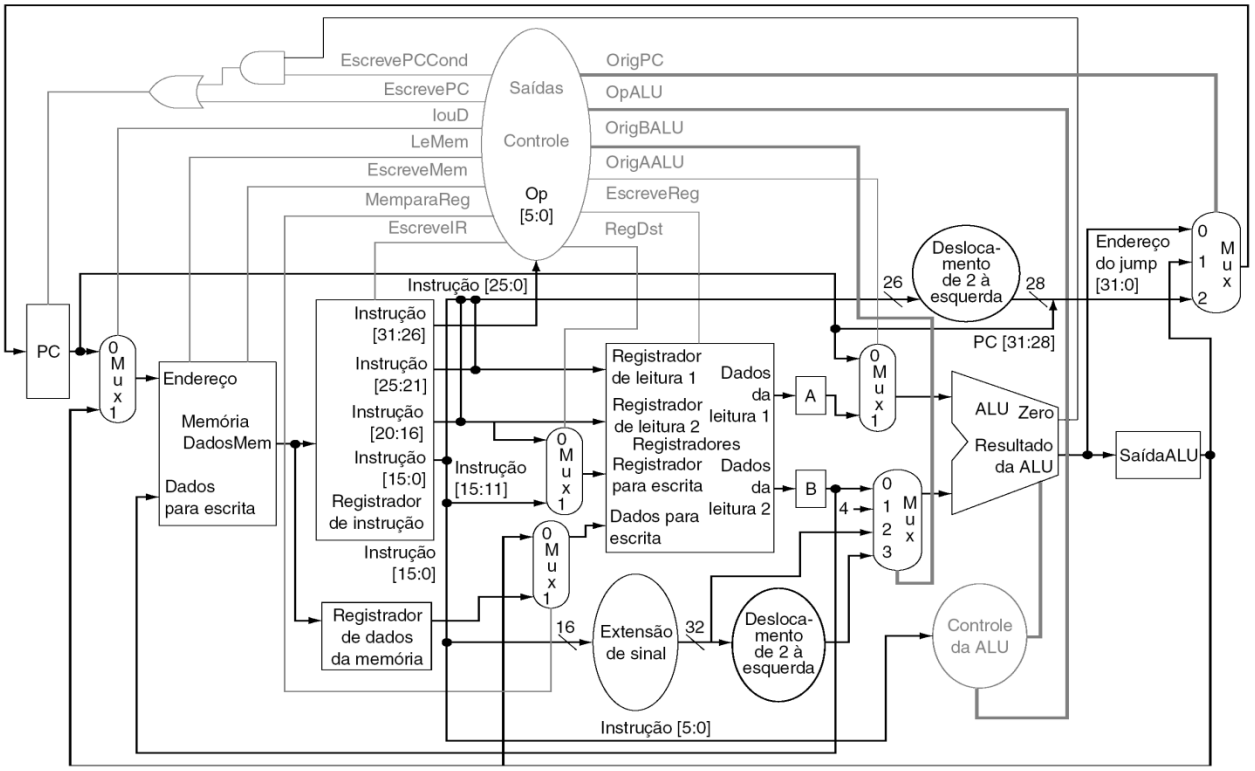


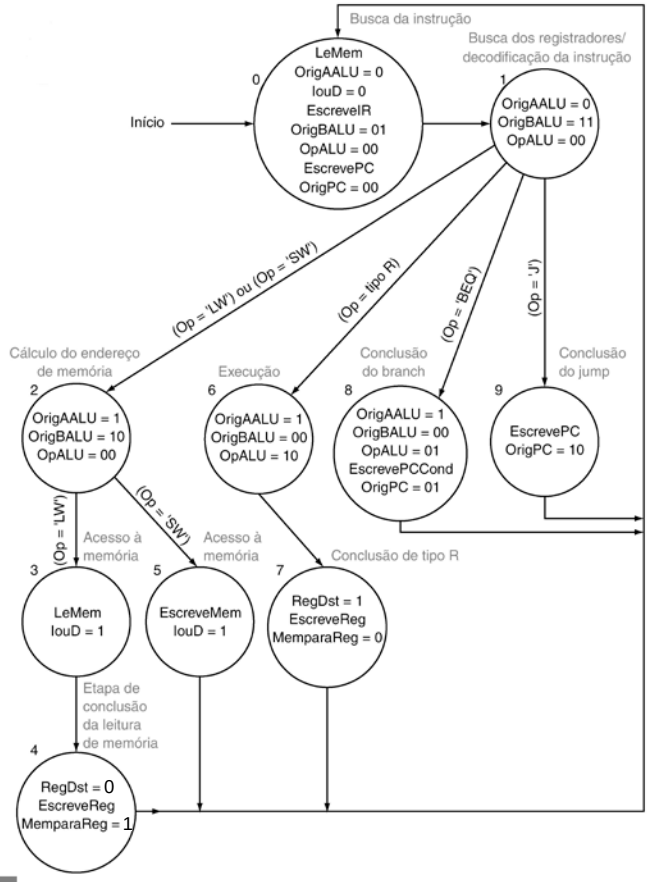
MEF do Controle do MIPS Multiciclo

Análise do controle para toda a ISA implementada:

1) Busca da Instrução
2) Decodificação
3) Execução
4) Acesso à Memória e Conclusão Tipo-R
5) Conclusão LW







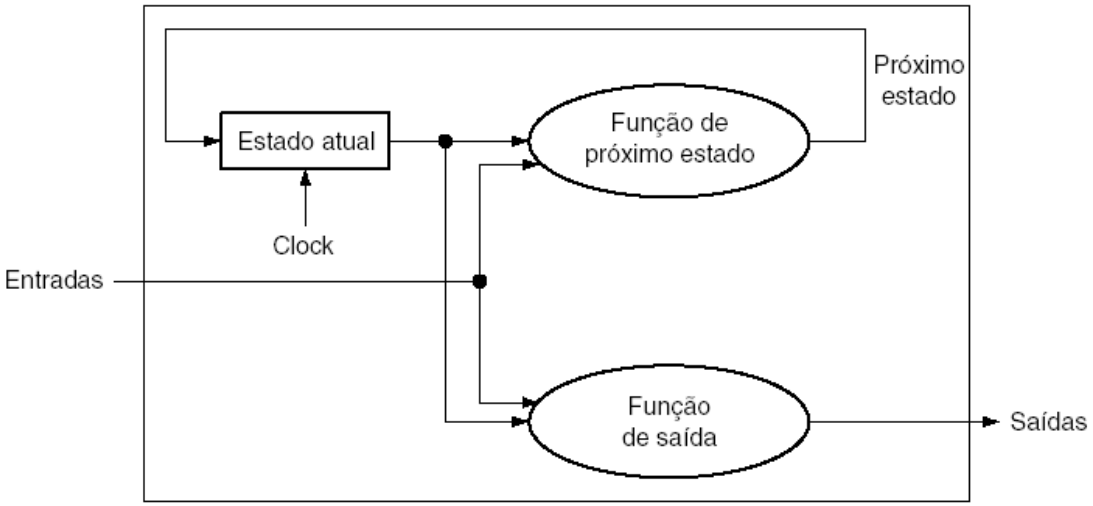
Etapa	Ação para instruções tipo R	Ação para instruções de acesso à memória	Ação para desvios	Ação para jumps
Busca da instrução	IR <= Memória[PC] PC <= PC + 4			
Decodificação da instrução e busca dos registradores	A <= Reg[IR[25:21]] B <= Reg[IR[20:16]] SaídaALU <= PC + (estende-sinal (IR[15:0]) << 2)			
Execução, cálculo do endereço, conclusão do desvio/jump	SaídaALU <= A op B	SaídaALU <= A + estende-sinal (IR[15:0])	if (A == B) PC <= SaídaALU	PC <= {PC [31:28], (IR[25:0]],2'b00)}
Acesso à memória ou conclusão de instrução tipo R	Reg[IR[15:11]]<= SaídaALU	Load: MDR <= Memória[SaídaALU] ou Store: Memória [SaídaALU] <= B		
Conclusão da leitura da memória		Load: Reg[IR[20:16]] <= MDR		

FIGURA 5.30 **Resumo das etapas realizadas para executar qualquer classe de instrução.** As instruções levam de três a cinco etapas de execu-

1

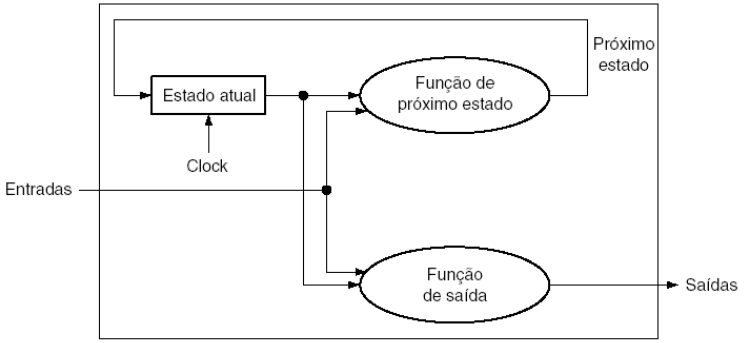
Máquinas de Estados Finitos - Implementação

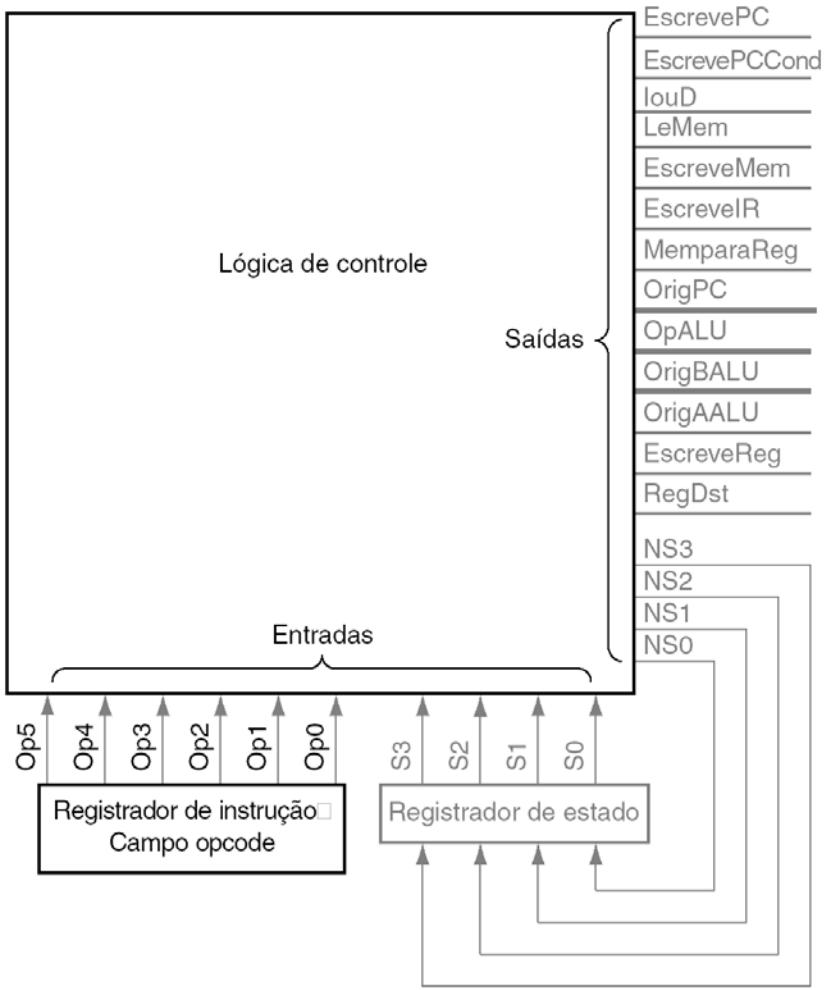
- Conjunto de estados
- Função de próximo estado: Determinada pelo estado atual e entrada
- Saída: Determinada pelo estado atual (Moore) e
  possivelmente pela entrada (Mealy)



Controle do MIPS com MEF

- estrutura da máquina de estados:
  - lógica de saída
  - lógica de transição
  - registrador de estado
  - entradas externas (código da instrução)



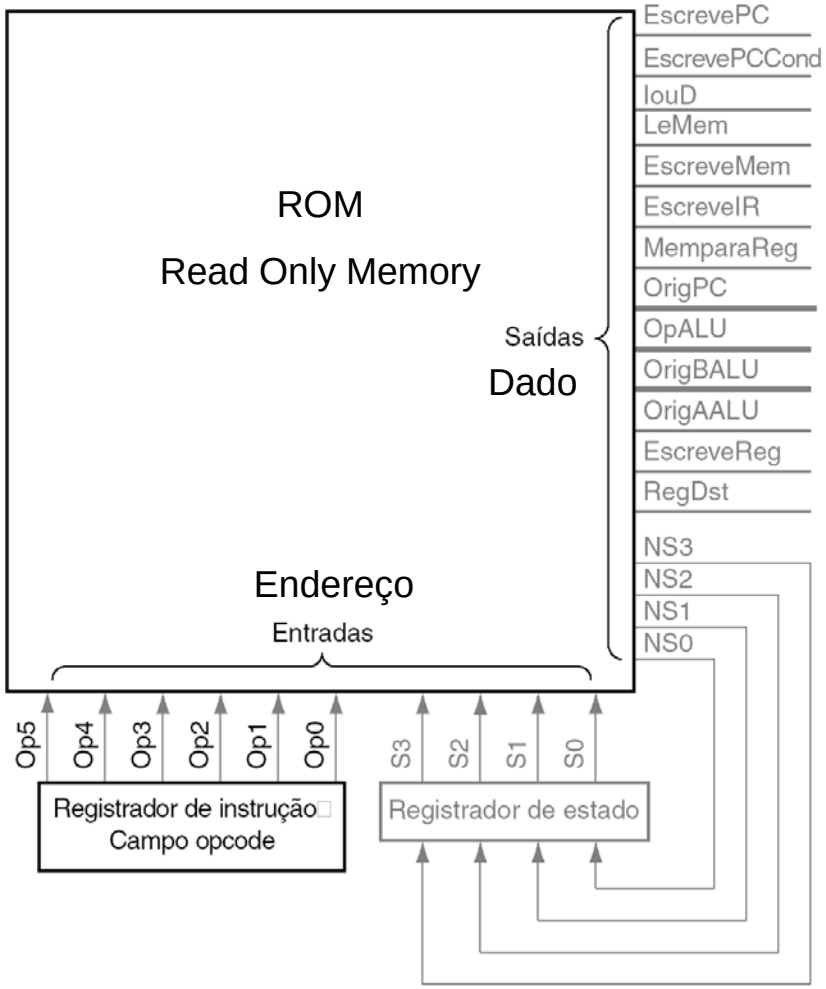


Controle com MEF

- Implementação com ROM
  - Simples!
- Tamanho da memória
  - 10 bits endereço:
    1024 posições de memória
  - 20 bits de dados

  Logo ROM de 20kbits

  Quantas posições de memória são realmente utilizadas?
- Porém, ineficiente



Exemplo do projeto lógico para cada saída na forma soma de produtos

- EscrevePC: Acionado nos estados 0 ou 9

s3	s2	s1	s0
0	0	0	0
1	0	0	1

a. Tabela verdade para EscrevePC

$$EscrevePC = \overline{s_0}.\overline{s_1}.\overline{s_2}.\overline{s_3} + s_0.\overline{s_1}.\overline{s_2}.s_3$$

- NS_0

Op5	Op4	Op3	Op2	Op1	Op0	S3	S2	S1	S0
X	X	X	X	X	X	0	0	0	0
1	0	0	0	1	1	0	0	1	0
1	0	1	0	1	1	0	0	1	0
X	X	X	X	X	X	0	1	1	0
0	0	0	0	1	0	0	0	0	1

d. A tabela verdade para a saída NS0, que está ativa quando o próximo estado é 1, 3, 5, 7 ou 9. Isso ocorre apenas se o estado atual é 0, 1, 2 ou 6.

$$NS_0 = \overline{s_3}.\overline{s_2}.\overline{s_1}.\overline{s_0} + op_5.\overline{op_4}.\overline{op_3}.\overline{op_2}.op_1.op_0.\overline{s_3}.\overline{s_2}.s_1.\overline{s_0} +$$
$$+op_5.\overline{op_4}.op_3.\overline{op_2}.op_1.op_0.\overline{s_3}.\overline{s_2}.\overline{s_1}.\overline{s_0} + \overline{s_3}.s_2.s_1.\overline{s_0} + \overline{op_5}.\overline{op_4}.\overline{op_3}.\overline{op_2}.\overline{op_1}.\overline{op_0}.s_3.s_2.s_1.s_0$$

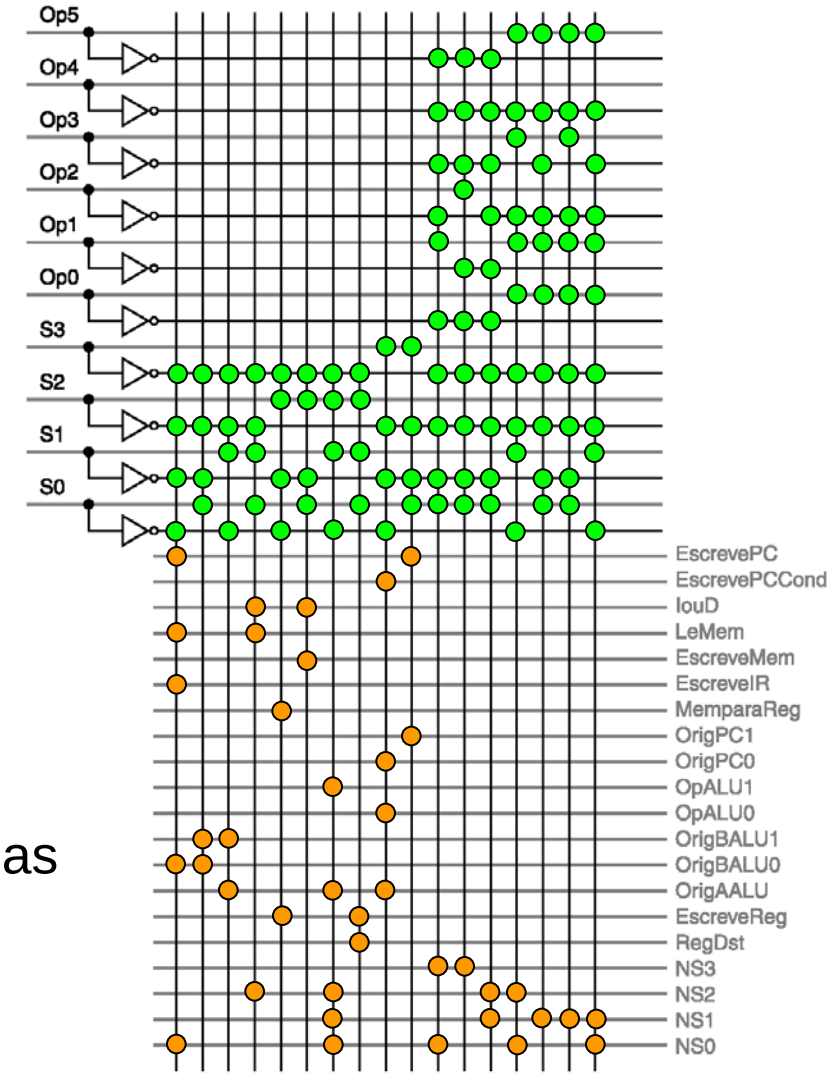
Controle com MEF

- Implementação com PLA

(Programmable Logic Array)

- Mais eficiente:
  - Pode compartilhar termos de produtos
  - Apenas entradas que possuem saídas ativas
  - Pode considerar *don't cares*

Tamanho: (EntradasxN.Prod.)+
(Saidas xN.Prod)
Tamanho: (10x17)+(20x17)=510 células



Microprogramação

Problemas da MEF:

- O projeto da parte de controle através de diagramas de transição de estados pode rapidamente se tornar inviável se o número de estados for muito grande
- MEF's de processadores complexos (x86!) podem ter milhares de estados

Uma alternativa para projeto é seguir um processo semelhante a programação

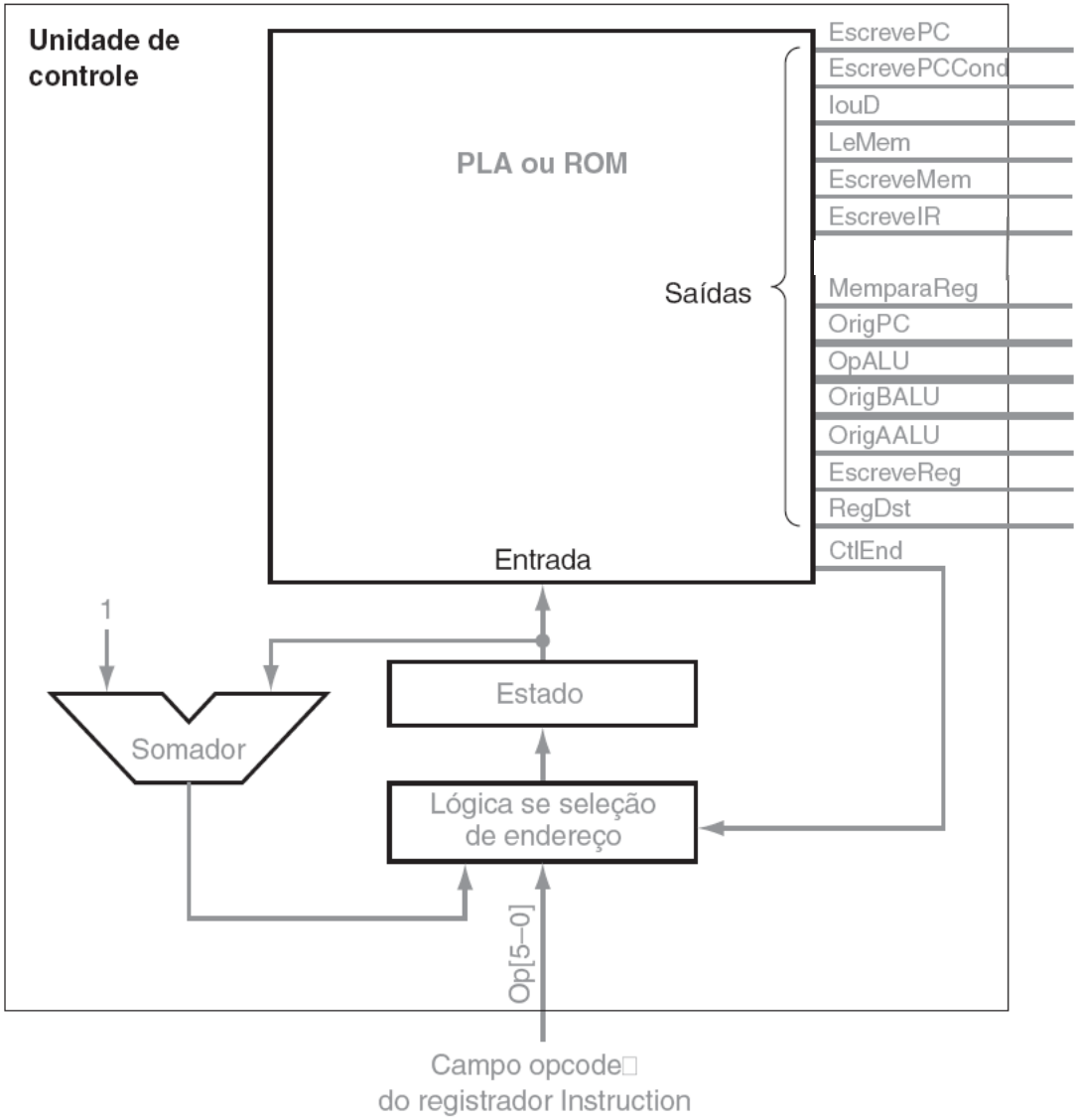
Microprogramação

- Uma microinstrução é definida pelos valores dos sinais de controle que atuam na unidade operativa durante um estado da MEF (ESTADO)

- A execução de uma instrução do processador pode então ser realizada através de uma sequência de microinstruções (TRANSIÇÕES)

- O conjunto de microinstruções que implementa o controle de um processador é chamado de microprograma (DIAGRAMA DE ESTADOS)

Estrutura do Sequenciador



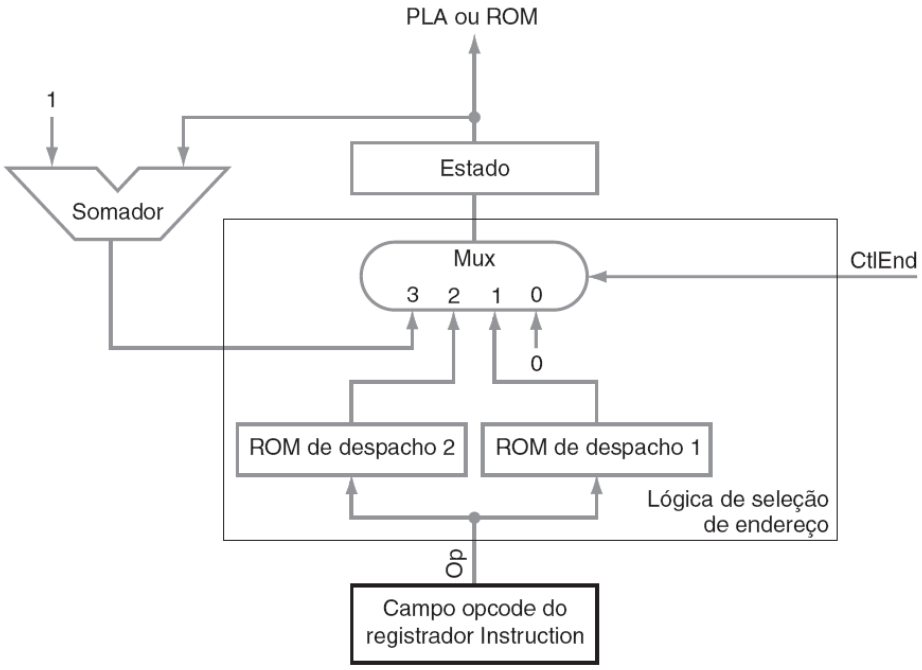
Microprograma

- O sequenciamento das microinstruções é realizado de forma similar a de um programa normal
  - microinstruções são usualmente executadas em sequência -> correspondem aos caminhos no diagrama de estados.
  - em alguns casos, a sequência a ser seguida depende de informações externas (código da instrução, por exemplo). Nestes casos, são necessários mecanismos de desvio.

Sequenciamento para o subset do MIPS

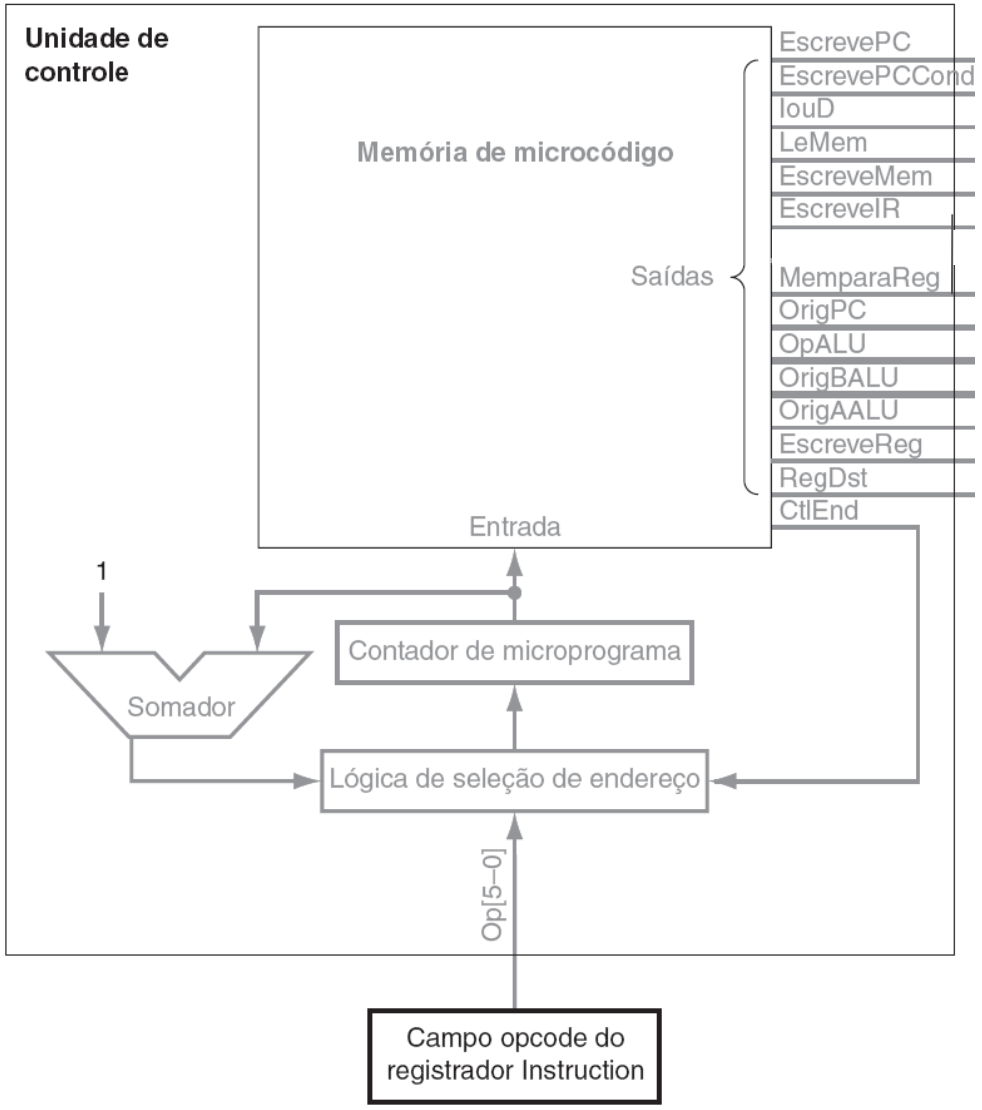
ROM de despacho 1		
Op	Nome do opcode	Valor
000000	formato R	0110
000010	jmp	1001
000100	beq	1000
100011	lw	0010
101011	sw	0010

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

Sequenciador Microcodificado



Formato da Microinstrução

- A microinstrução é dividida em campos que atuam sobre conjuntos de elementos da unidade operativa

- Os campos são escolhidos de acordo com sua finalidade. O controle da ULA, por exemplo, é associado a um campo

- O microprograma é usualmente implementado em ROM, EEPROM, FLASH, etc (ou PLA), onde cada microinstrução tem seu próprio endereço

Função dos campos das Microinstruções

Nome do campo	Função do campo
ALU Control	Especificar a operação sendo realizada pela ALU durante este clock; o resultado é sempre escrito em SaídaALU.
SRC1	Especificar a origem para o primeiro operando da ALU.
SRC2	Especificar a origem para o segundo operando da ALU.
Register Control	Especificar leitura ou escrita para o banco de registradores e a origem do valor para uma escrita.
Memory	Especificar leitura ou escrita e a origem para a memória. Para uma leitura, especifica o registrador destino.
PCWrite Control	Especificar a escrita do PC.
Sequencing	Especificar como escolher a próxima microinstrução a ser executada.

FIGURA 5.7.1 Cada microinstrução contém esses sete campos. Os valores para cada campo são mostrados na Figura 5.7.2

MicroISA

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 C.5.1 Cada campo de microcódigo é traduzido para um conjunto dos sinais de controle a serem definidos.

Microprograma para a Unidade de Controle MIPS

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.

Exercício

- Considerando o workload do compilador gcc, qual a CPI média do MIPS multiciclo implementado?
  - Load: 22% (5 ciclos)
  - Store: 11% (4 ciclos)
  - Operações aritméticas: 49% (4 ciclos)
  - Branches: 16% (3 ciclos)
  - Jumps: 2% (3 ciclos)

CPI = 0.22*5 + 0.11*4 + 0.49*4 + 0.16*3 + 0.02*3 =

1.1 + 0.44 + 1.96 + 0.48 + 0.06 = 4.04