Organização de Computadores

Aula 08

Bloco de controle multi-ciclo
Projeto com microprogramação

Aula passada

- **Foi visto na aula passada a implementação de um Bloco de Controle Multi-Ciclo através de uma máquina de estados finitos, FSM.**

- **Este controle poderia ser implementado por um conjunto de portas lógicas , uma ROM ou uma PLA.**

- **Entretanto, na medida que o caminho de dados torna-se complexo e portanto grande (Ex: Instruções de um x86), mais difícil é sua implementação, tornando difícil a especificação de funções de controle complexas.**

- **Portanto, uma solução é empregar algumas idéias da programação para ajudar na criação de um método de especificação do controle que torne fácil seu entendimento.**

- **Desta necessidade surge a proposta de empregar microinstruções**

Microprograma

- **Estabelece o conjunto de sinais de controle que deve estar ativo num determinado estado, como uma instrução a ser executada**

- **Para evitar confusão com as instruções do MIPS, estas instruções de controle serão chamadas de microinstruções.**

- **Cada microinstrução define o conjunto de sinais de controle do caminho de dados que precisa estar ativo num determinado estado.**

- **Portanto, executar uma microinstrução significa ativar os sinais de controle especificados para um determinado estado.**

Bloco de controle
projeto com microprogramação

1. Introdução

2. Formato das micro-instruções

3. Microprogramas

4. Implementação do bloco de controle

1. Introdução

- **Uma FSM, Máquina de Estados Finitos, é muito complexa para blocos de controle de processadores com conjuntos complexos de instruções**
  - **Formatos variados de instruções**
  - **Muitos modos de endereçamento**
  - **Instruções com número variável de ciclos**

- **Microprogramação: maneira estruturada de desenvolver um bloco de controle muito complexo**
  - **Cada micro-instrução define os sinais de controle necessários para execução de um passo da instrução**
  - **Microprograma é um conjunto de micro-instruções para a execução de uma ou mais instruções**
  - **Microprograma é armazenado numa "memória de controle" (ROM ou PLA)**

- **Microprograma pode ser desenvolvido simbolicamente**
  - **Micro-assembler gerando as micro-instruções em formato binário**

2. Formato das micro-instruções

- **Formatos possíveis para micro-instruções:**
  - **1 bit para cada sinal de controle – micro-instrução pode ficar muito larga**
  - **Codificação por campos – cada campo deve especificar sinais de controle que nunca precisam ser gerados simultaneamente**

- **É comum o emprego de um campo para:**
  - **Controlar a ULA**
  - **Fonte de operando A**
  - **Fonte de operando B**
  - **Destino do operando**

- **Importante que o formato:**
  - **Simplifique a representação, tornando o controle mais fácil de entender.**
  - **Impeça a escrita de microinstruções inconsistentes (cada campo represente um conjunto de sinais que não se sobreponham).**

Formato das micro-instruções

- **Sinais que nunca estão ativos ao mesmo tempo devem compartilhar o mesmo campo.**

- **Micro-instruções para o MIPS têm 7 campos:**
  - ***Controle da ALU*** **– Especifica a operação a ser executada na ALU**
  - ***SRC1*** **– Seleciona a fonte do 1º operando para ALU;**
  - ***SRC2*** **– Seleciona a fonte do 2º operando para ALU;**
  - **Controle do Registrador – Registrador a ser escrito com o resultado da ALU;**
  - ***Memória*** **– Escrita ou Leitura na memória e fonte de endereço;**
    - **Especifica o registrador destino, em instrução *load***
    - **Ou registrador fonte, em instrução *store***
  - ***Controle do PC*** **– escrita no PC**
  - ***Sequenciamento*** **– Seleção da próxima micro-instrução**

- **6 Campos controlam o caminho de dados,**
- **1 Campo de seqüenciamento especifica a próxima micro-instrução.**

Formato das micro-instruções

ALU control	Add	ALU executa uma soma
	Func code	ALU executa operação especificada no campo de função
	Subt	ALU executa subtração
SRC1	PC	PC é primeira entrada da ALU
	A	Registrador A é primeira entrada da ALU
SRC2	B	Registrador B é segunda entrada da ALU
	4	Constante 4 é segunda entrada para ALU
	Extend	Saída da unidade de extensão de sinal é 2ª entrada para ALU
	Extshift	Saída da unidade de deslocamento de 2 bits é 2ª entrada para ALU
Register Control	Read	Lê 2 registradores indicados em rs e rt e coloca em A e B
	Write ALU	Escreve conteúdo de ALUout no registrador indicado no campo rd
	Write MDR	Escreve conteúdo de MDR no registrador indicado no campo rt
Memory	Read PC	PC: endereço de leitura em memória; Resultado escrito em IR
	Read ALU	ALUout: endereço de leitura em memória; Resultado escrito em MDR
	Write ALU	ALUout: endereço de escrita em memória; Dado p/ escrita: registrador B

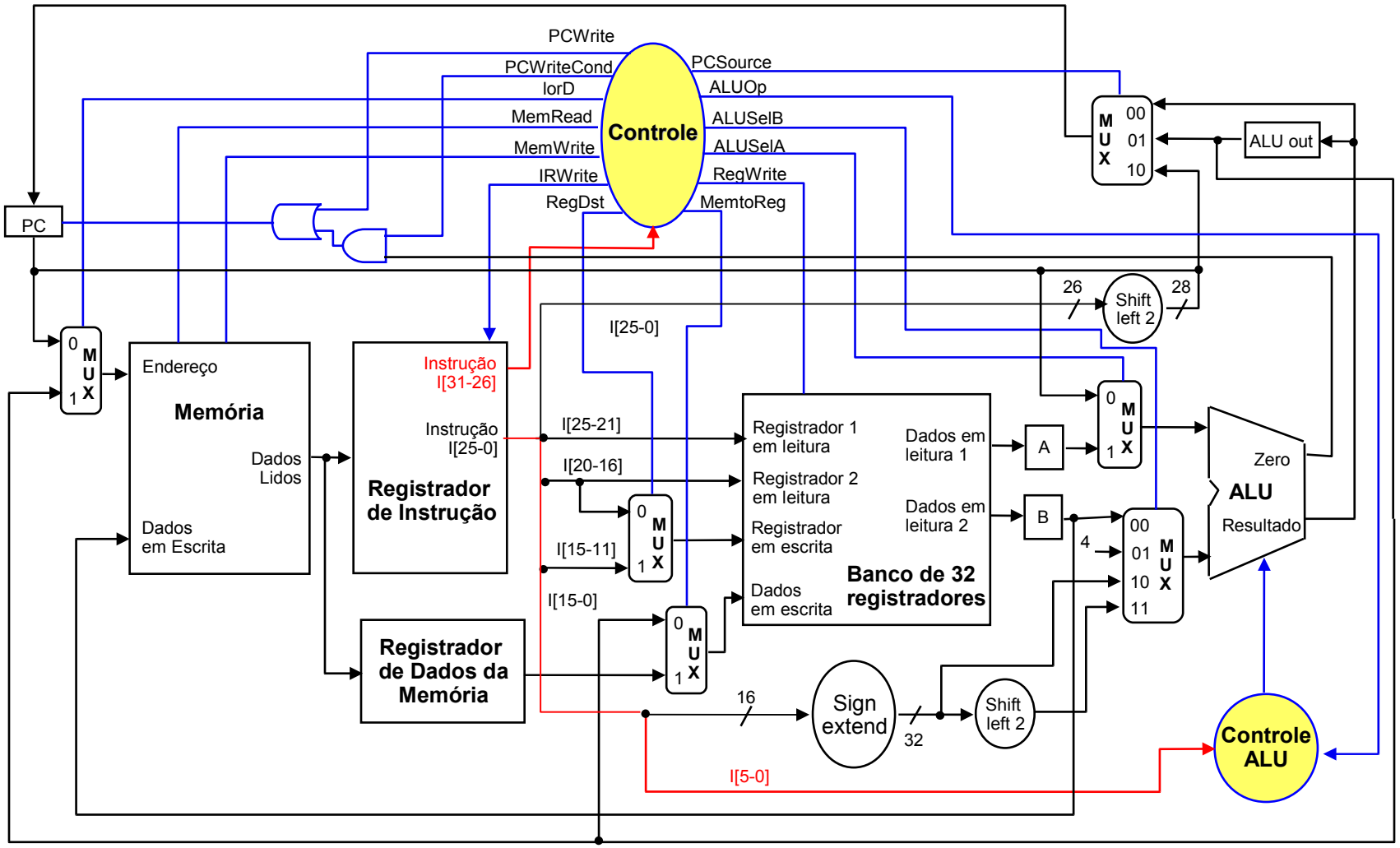
Formato das micro-instruções

PCWrite control	ALU	Escreve saída da ALU no PC
	ALUout-cond	Se saída Zero = 1, atualiza PC c/ conteúdo do registrador ALUout
	Jump address	Escreve endereço de desvio no PC
Sequencing	Seq	Escolhe a próxima microinstrução seqüencialmente
	Fetch (0)	Volta para 1ª microinstrução para iniciar uma nova instrução
	Dispatch i	Dispatch usando a ROM especificada por "i" (1 ou 2 para o mips)

Tamanho da micro-instrução: 13 bits, supondo campos codificados

ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
2	1	2	2	2	2	2

Sinais de controle



Sinais de controle e os campos

- **Cada campo da micro-instrução gera um ou mais sinais de controle**
  - **ALU control : ALUOp**
  - **Src1 : AluSelA**
  - **Src2 : AluSelB**
  - **RegisterControl : RegDst, MemtoReg, RegWrite**
  - **Memory : IorD, MemRead, MemWrite, IRWrite**
  - **PC Write control : PCWrite, PCWriteCond, PCSource**

- **Para cada valor no campo, diferentes valores devem ser atribuídos aos sinais de controle**
- **Exemplo: campo Memory**

	IorD	MemRead	MemWrite	IRWrite
ReadPC	0	1	0	1
ReadALU	1	1	0	0
WriteALU	1	0	1	0

Campo de seqüenciamento

- **A escolha da próxima microinstrução pode ter 3 métodos:**
  - **Incrementar o endereço da microinstrução corrente, para tal é colocado no campo "Seq",**
  - **Desviar a micro-instrução , para tal é colocado no campo "Busca",**
  - **Escolher a próxima microinstrução com base na entrada da unidade de controle, para tal é colocado "Despacho".**

Normalmente são implementadas várias tabelas de despacho. Na nossa implementação são necessárias duas tabelas:

- **Despacho 1**
- **Despacho 2**

Valor Dispatch do campo Sequencing

- **Seleciona endereço da próxima micro-instrução de acordo com entradas do bloco de controle**
- **Tabela de endereços, usualmente armazenada em ROM, é indexada pelos sinais de entrada do BC**
- **De acordo com a FSM, esta situação ocorre nas transições a partir dos estados 1 e 2**

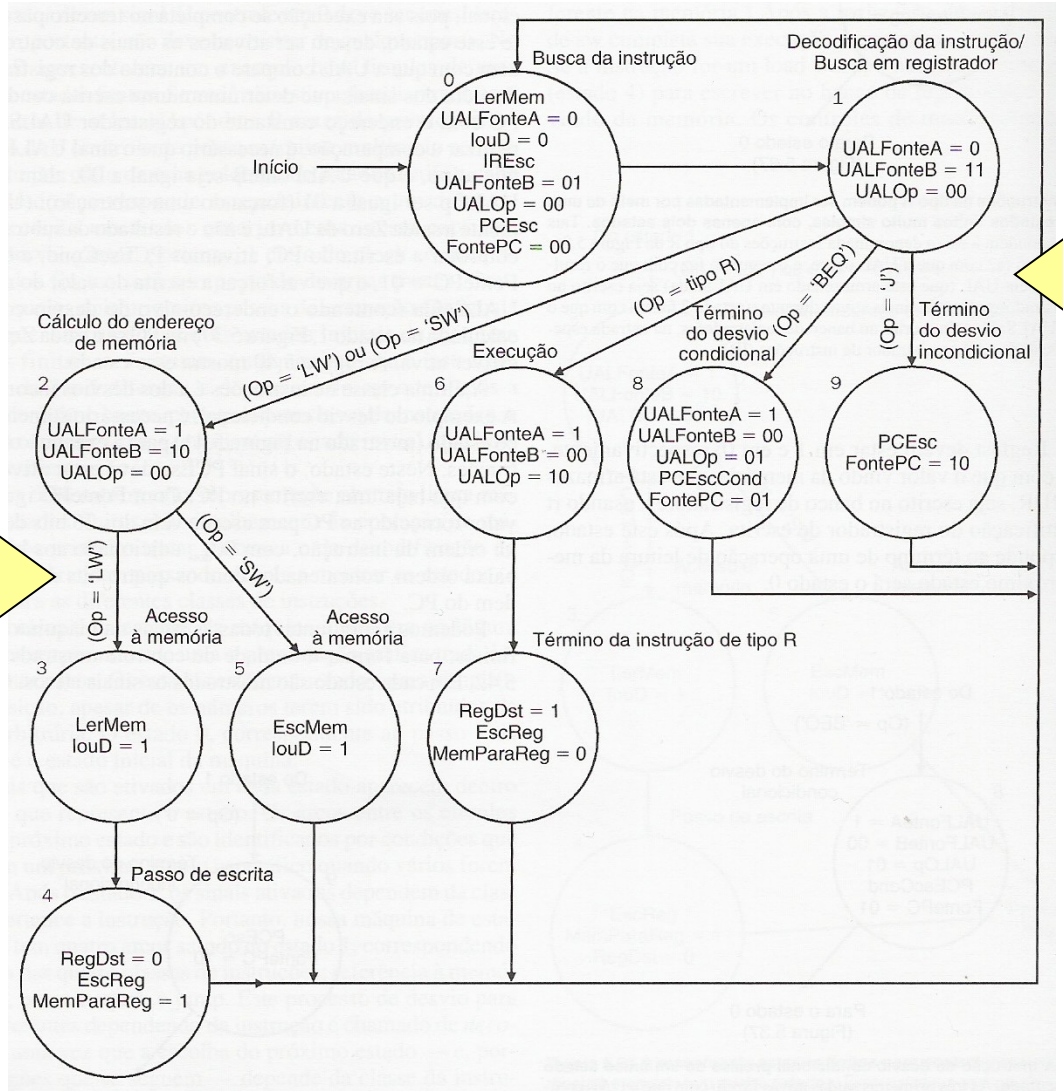
Tabela de Dispatch 1

op-code	próxima µinstrução
'lw' ou 'sw'	2
tipo R	6
'beq'	8
'jump'	9

Tabela de Dispatch 2

op-code	próxima µinstrução
'lw'	3
'sw'	5

Tabela de Despacho



3. Microprogramas
Busca da Instrução, Decodificação, PC+4, Target

Exemplo das 2 primeiras microinstruções:
Busca da instrução, decodificação, cálculo de PC+4, cálculo de Target

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Fetch	Add	PC	4		ReadPC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1

Primeira micro-instrução

ALU control, SRC1, SRC2	calcular PC + 4
Memory	busca da instrução e escrita em IR
PCWrite control	saída da ALU é escrita em PC
Sequencing	seguir para próxima microinstrução

Segunda micro-instrução

ALU control, SRC1, SRC2	calcular PC + deslocamento x 4, estendido para 32 bits
Register control	usa rs e rt p/ ler os registradores; resultado em A e B
Sequencing	usar tabela 1 para obter endereço da próxima μinstrução

Microprograma para instruções de referência a memória

Instruções de referência à memória

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
LWSW1	Add	A	Extend				Dispatch 2
LW2					ReadALU		Seq
				WriteMDR			Fetch
SW2					WriteALU		Fetch

ALU Control, SRC1, SRC2	Calcula endereço de memória, rs + extend >UALOut
Sequencing	Usa tabela de despacho 2 para desviar se lw ou sw

Memory	Lê memória usando UALOut, escrvendo em MDR
Sequencing	seguir para próxima microinstrução

Register control	Escreve conteúdo de MDR no banco de registradores
Sequencing	desvia para microinstrução cujo label é Fetch

Register control	Escr. memória usando UALOut como end e B como dado
Sequencing	desvia para microinstrução cujo label é Fetch

Microprograma para instruções de tipo R

Instruções de tipo R

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Rform1	Funct	A	B				Seq
				WriteALU			Fetch

ALU control, SRC1, SRC2	UAL opera sobre A e B, baseado em funct
Sequencing	seguir para próxima microinstrução

Register control	Valor da UALSaida é escrito no banco de registradores
Sequencing	seguir para próxima microinstrução

Microprograma para instruções de tipo Branch

Instrução de branch

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
BEQ1	Subt	A	B			ALUout-cond	Fetch

ALU Control, SRC1,SRC2	UAL subtrai valores de A e B para gerar saída Zero
PCWrite control	saída da ALU é escrita em PC caso Zero verdadeiro
Sequencing	desvia para microinstrução cujo label é Fetch

Microprograma para instruções de tipo Jump

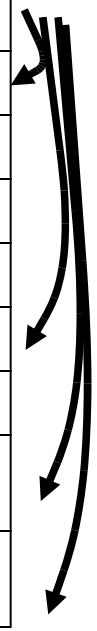
Instrução de jump

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Jump1						Jump address	Fetch

PCWrite control	escrita do PC com desvio condicional concatenado
Sequencing	seguir para microinstrução cujo label é Fetch

Memória de controle completa

Label	ALU control	SRC1	SRC2	Register control	Memory	PCWrite control	Sequencing
Fetch	Add	PC	4		ReadPC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1
LWSW1	Add	A	Extend				Dispatch 2
LW2					ReadALU		Seq
				WriteMDR			Fetch
SW2					WriteALU		Fetch
Rform1	Funct	A	B				Seq
				WriteALU			Fetch
BEQ1	Subt	A	B			ALUout-cond	Fetch
Jump1						Jump address	Fetch



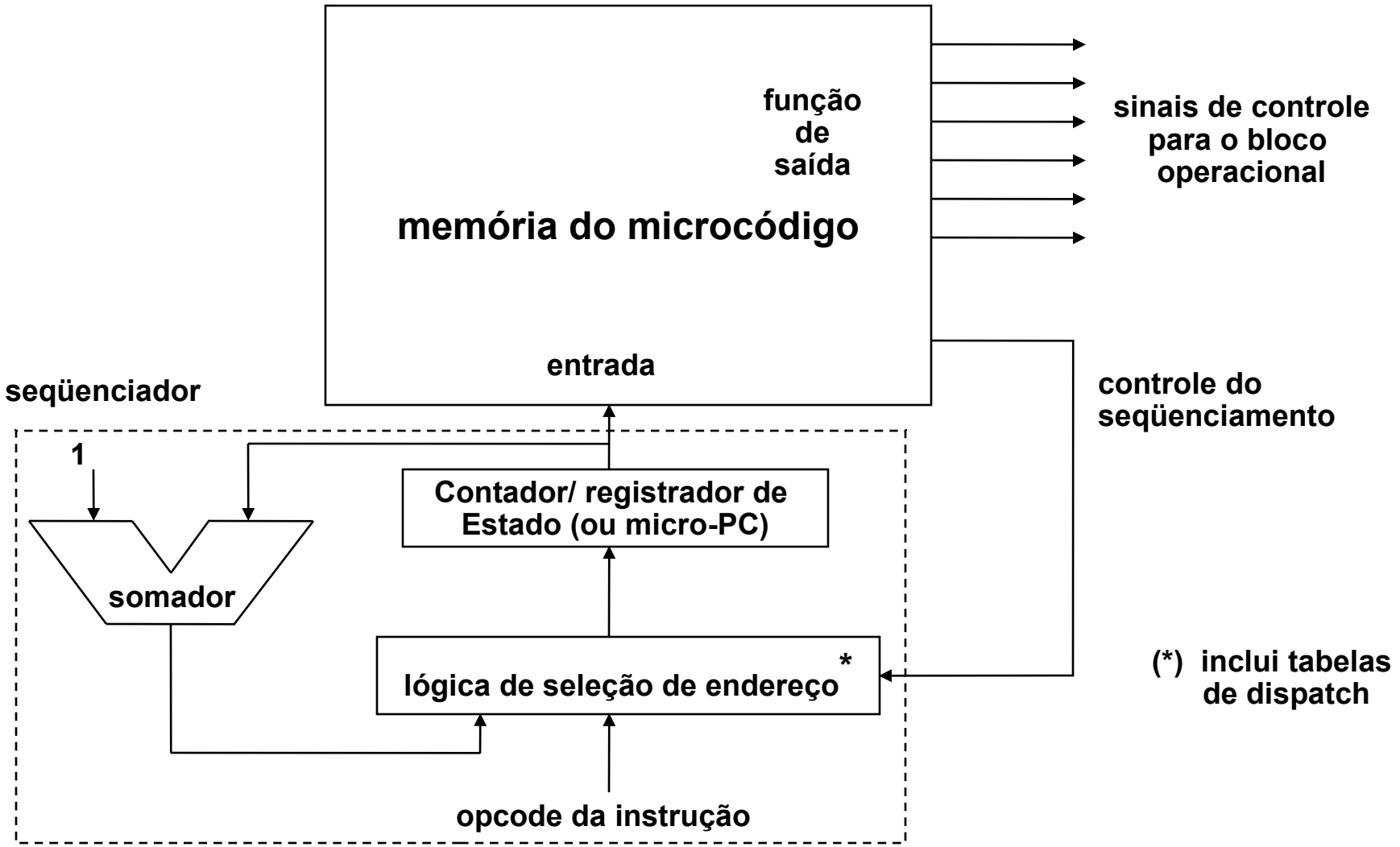
Implementação do Microprograma

- **O mapeamento de um microprograma em hardware envolve 2 aspectos:**
  - **Como implementar a função de seqüenciamento,**
  - **Escolha do método de armazenar as funções do controle principal.**

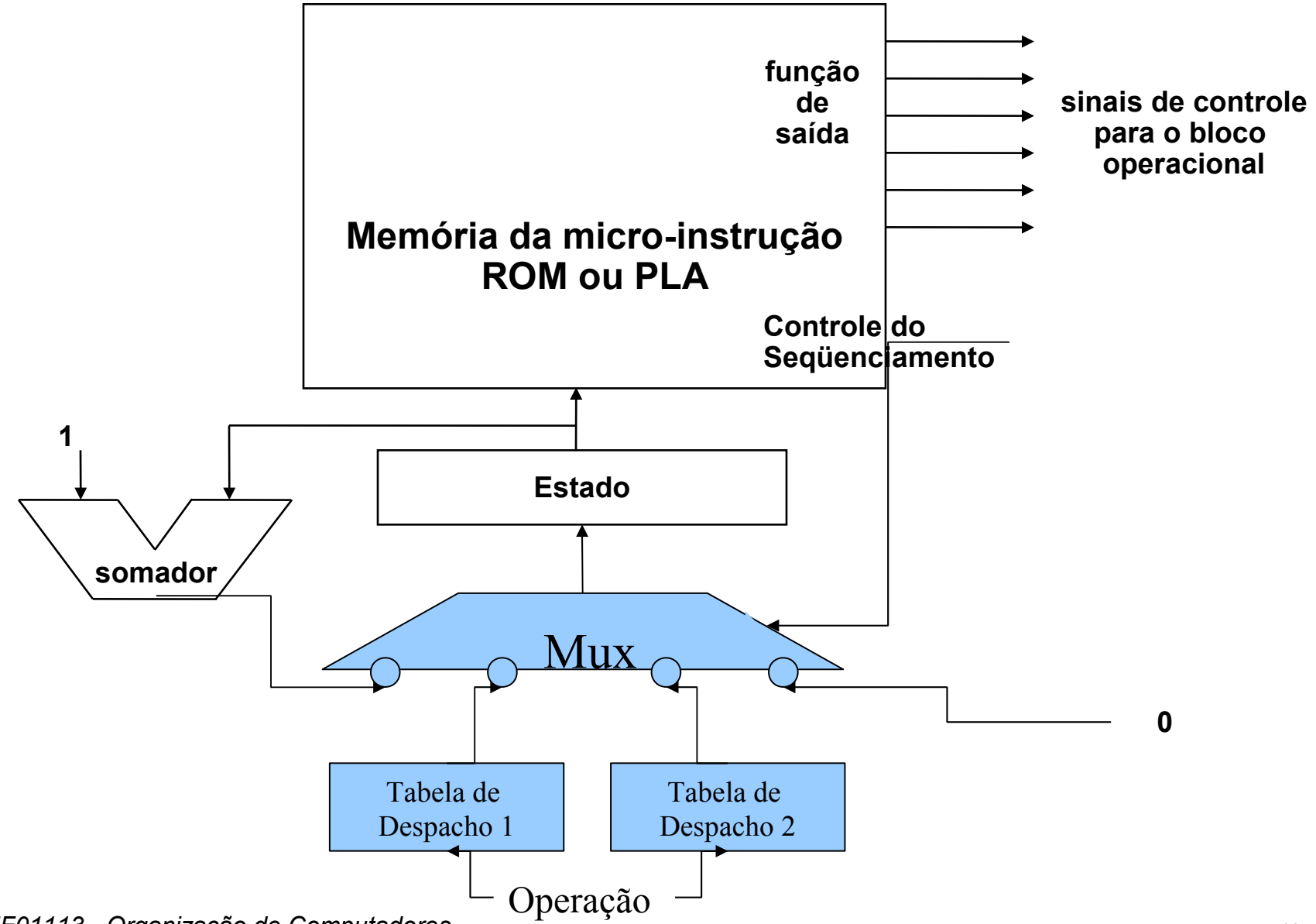
- **Normalmente o armazenamento das funções de controle é feito numa memória ROM**

- **A função seqüenciamento emprega um sistema incrementador para escolha da próxima instrução de controle.**

4. Implementação do BC



4. Implementação do BC



FIM