



Instituto de Matemática
Departamento de Ciência da Computação

Arquitetura de Computadores

Unidade de controle

Micro-operações, controle do processador e controle
microprogramado

Prof. Marcos E Barreto

Tópicos

- Micro-operações
- Controle do processador
- Implementação por hardware
- Controle microprogramado

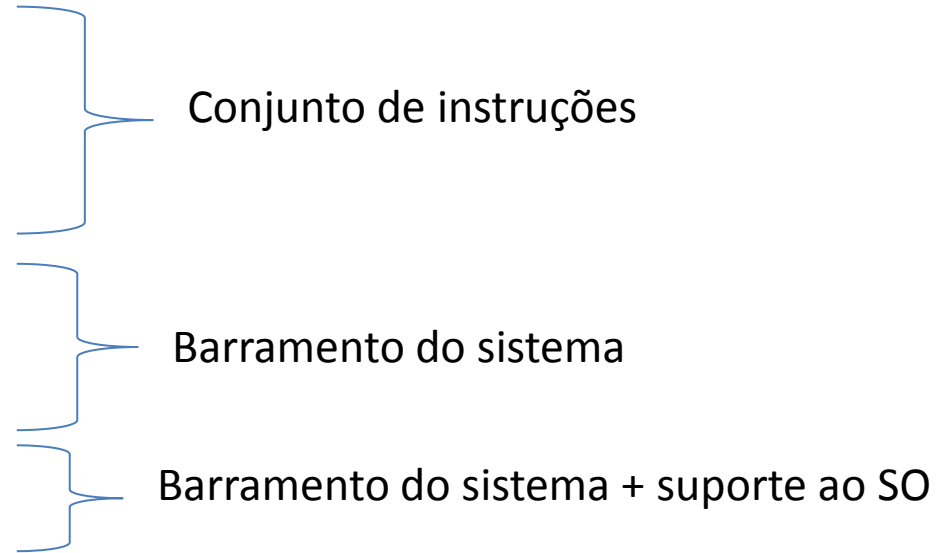
• Referências:

• William Stallings. Arquitetura e organização de computadores. 8 ed. Caps. 15 e 16.



Contextualização

- Funcionamento do processador é especificado em termos de:
 - Operações (*opcodes*)
 - Modos de endereçamento
 - Registradores
 - Interface com módulos de E/S
 - Interface com módulo de memória
 - Interrupções

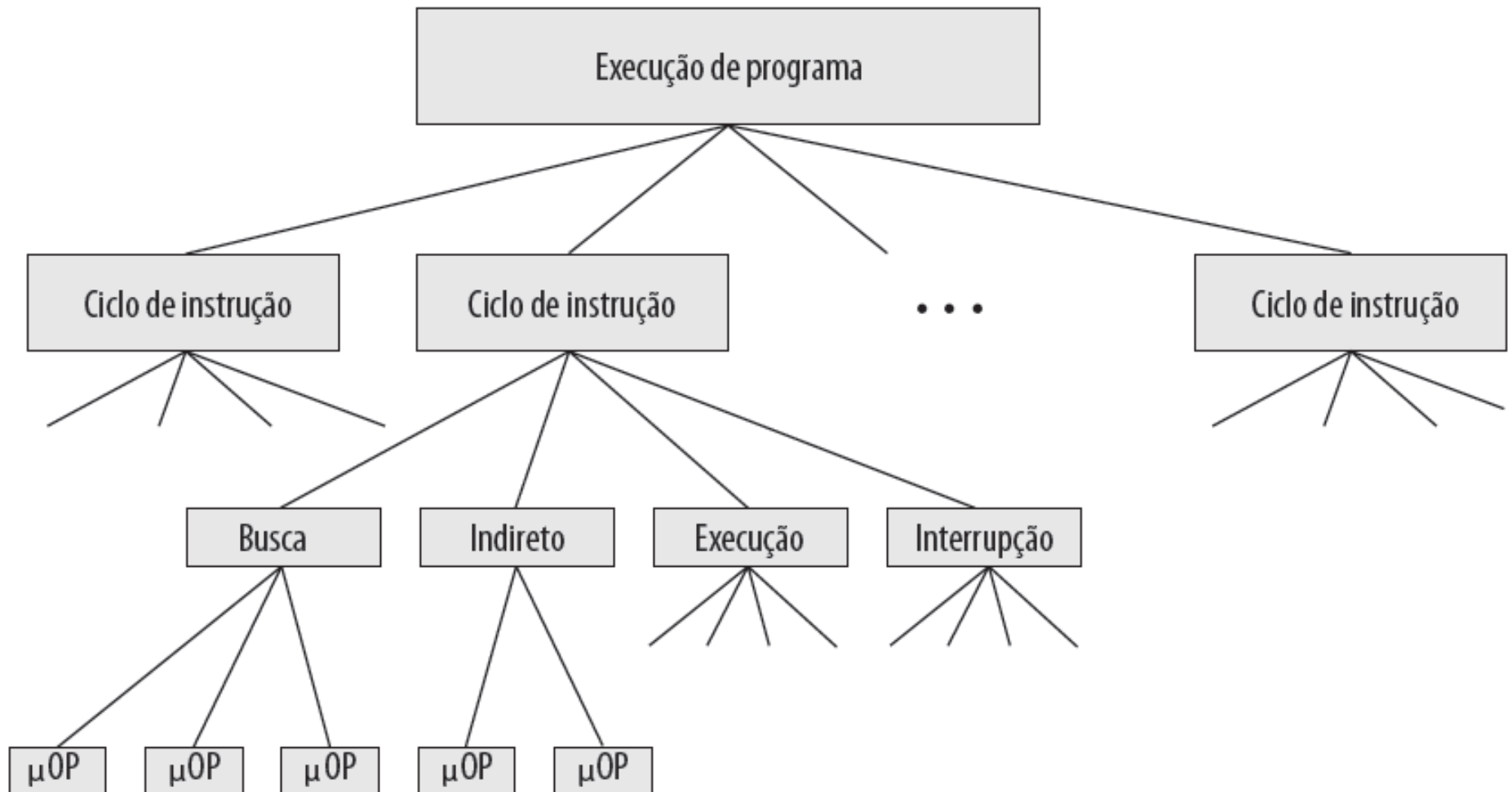


Requisitos funcionais (o que o processador tem que fazer)

Unidade de controle: controla como essas funções são efetuadas.

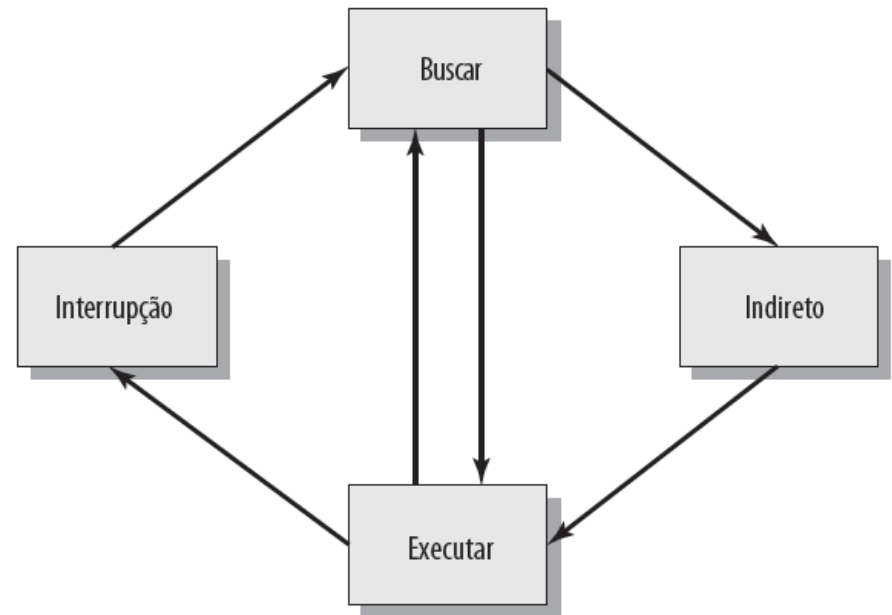
Contextualização

- Vários níveis de controle durante a execução de programas
 - Execução sequencial de instruções. Cada instrução é executada durante um ciclo de instruções, que é composto por vários subciclos. Cada subciclo, por sua vez, envolve operações mais curtas, denominadas micro-operações.



Micro-operações

- A execução de programas em computadores é feita com base no ciclo de instrução.
- Cada ciclo de instrução possui pequenas unidades, denominadas (sub)ciclos: busca, indireto, execução e interrupção.



- Cada um dos passos que compõem um determinado (sub)ciclo é denominado micro-operação, que corresponde à uma unidade funcional atômica.
 - Micro = simples, realiza pouco processamento.

Micro-operações

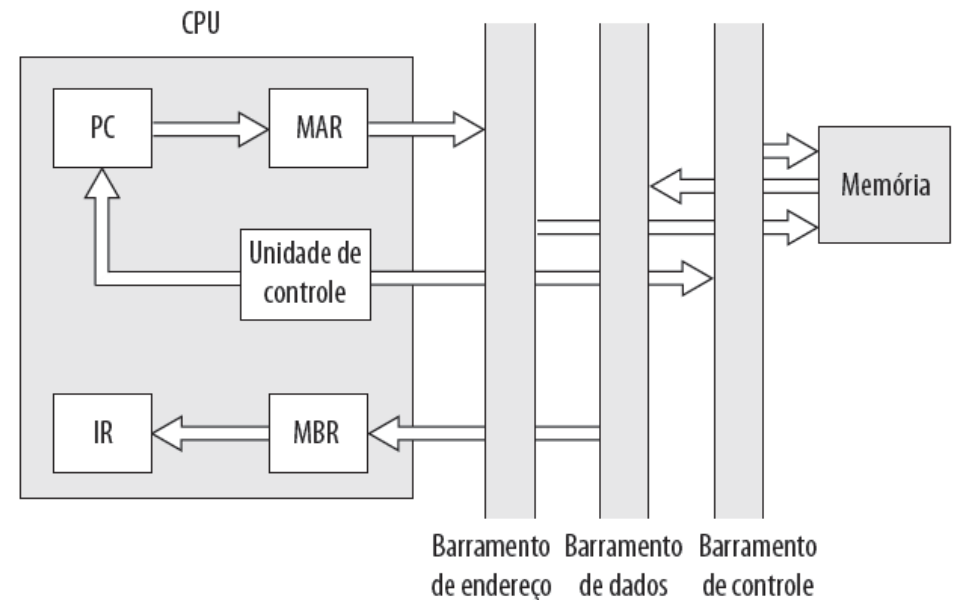
. Ciclo de busca

INÍCIO DO CICLO DE INSTRUÇÃO

1. PC = endereço da próxima instrução
(ex. 1100100)

CICLO DE BUSCA

2. MAR = PC
3. Barramento de endereços = MAR
Barramento de controle = comando READ
Barramento de dados = <VALOR>
MBR = <VALOR>
PC = PC + 1 (próxima instrução)
4. IR = MBR (libera MBR para possível ciclo indireto)



MBR = registrador de buffer de memória
MAR = registrador de endereço de memória
IR = registrador da instrução
PC = contador de programa

MAR	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
MBR	0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
PC	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1
IR	
AC	

3

(c) Depois do segundo passo

MAR	
MBR	
PC	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
IR	
AC	

(a) Início (antes de t₁)

1

MAR	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
MBR	
PC	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
IR	
AC	

(b) Depois do primeiro passo

2

MAR	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
MBR	0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
PC	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1
IR	0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
AC	

(d) Depois do terceiro passo

4

Micro-operações

- **Ciclo de busca**

- Na prática: 3 passos e 4 micro-operações

t1: MAR \leftarrow PC

t2: MBR \leftarrow memória(MAR)

PC \leftarrow (PC) + 1

t3: IR \leftarrow (MBR)

t1: MAR \leftarrow PC

t2: MBR \leftarrow memória(MAR)

t3: PC \leftarrow (PC) + 1

IR \leftarrow (MBR)

Dois agrupamentos possíveis para as micro-operações, obedecidas as dependências de controle e de dados (lógica).

Micro-operações

- **Ciclo indireto**

- Uma vez lida a instrução, o próximo passo é buscar os operandos.
- Ex. formato de instrução com um endereço (direto ou indireto).

t1: MAR ← (IR(endereço))

t2: MBR ← memória(MAR)

t3: IR(endereço) ← (MBR(endereço))

1. Campo de endereço da instrução é transferido para o MAR.
2. Endereço do operando é obtido em memória(MAR).
3. Campo de endereço de IR é atualizado a partir do MBR, passando a ter um endereço de operando direto em vez de um indireto.
4. IR está pronto para o ciclo de execução.

Micro-operações

• Ciclo de interrupção

- Ao completar o ciclo de execução, um teste é feito para verificar se alguma interrupção foi habilitada.

```
t1: MBR <- (PC)
t2: MAR <- Endereço_salvar
    PC <- Endereço_rotina
t3: Memória <- (MBR)
```

1. Conteúdo de PC é movido para MBR, para que ele possa ser salvo para o retorno da interrupção.
2. MAR é carregado com o endereço onde o conteúdo do PC deve ser salvo. PC recebe o endereço da rotina de tratamento da interrupção.
3. Armazenar na memória o valor do MBR (valor antigo do PC).
4. Processador está pronto para iniciar novo ciclo de instrução (na rotina de tratamento da interrupção).

Micro-operações

- **Ciclo de execução**

- Série de diferentes micro-operações que podem ocorrer, devido à diversidade de opcodes existentes.
- Ex.: instrução de soma => **ADD R1, X**

t1: MAR <- (IR(endereço))

t2: MBR <- Memória(MAR)

t3: R1 <- (R1) + (MBR)

1. A parte de endereço do IR é carregado no MAR.
2. Posição de memória referenciada em MAR é lida e armazenada em MBR.
3. Os conteúdos de R1 e MBR são somados e armazenados em R1.

Micro-operações

- **Ciclo de execução (2)**

- Ex.: instrução incrementar e pular se zero => **ISZ X**

- O conteúdo da posição X é incrementado em 1. Se for igual a 0, a próxima instrução é pulada.

t1: MAR ← (IR(endereço))

t2: MBR ← Memória(MAR)

t3: MBR ← (MBR) + 1

t4: Memória ← (MBR)

se ((MBR) = 0) então (PC ← (PC) + 1)

Novo recurso => **ação condicional** => PC é incrementado se MBR = 0.

=> Uma única micro-operação, que pode ser feita em paralelo com a escrita do valor de MBR na memória.

Micro-operações

. Ciclo de execução (3)

- Ex.: instrução desvio e salvar endereço => **BSA X**
 - O endereço da instrução que segue a instrução BSA é salvo na posição X e a execução continua na posição X + I. O endereço salvo será usado posteriormente para retorno.
 - Técnica simples para implementar chamada de sub-rotinas.

```
t1: MAR <- (IR(endereço))  
    MBR <- (PC)  
t2: PC <- (IR(endereço))  
    Memória <- (MBR)  
t3: PC <- (PC) + 1
```

1. Endereço em PC no início da instrução é o endereço da próxima instrução na sequência.
2. Isso é salvo no endereço designado em IR.
3. Endereço posterior também é incrementado para fornecer o endereço da instrução para o próximo ciclo de instrução.

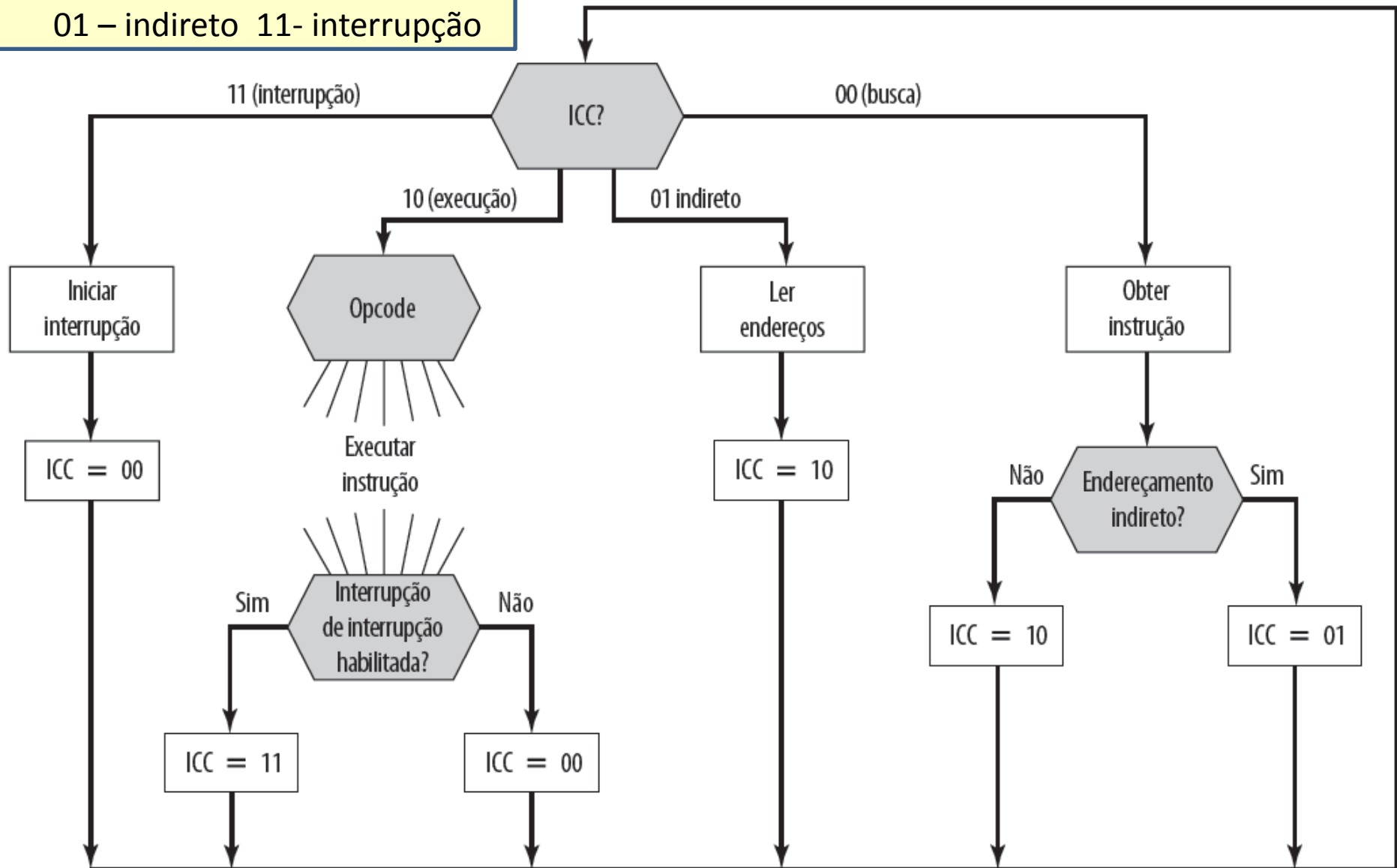
• Sequência completa de micro-operações

Um ciclo de micro-operações específico para cada ciclo de leitura, indireto e interrupção.
Ciclos de micro-operações dependentes do opcode para cada ciclo de execução.

Código de ciclo de instrução (ICC)

00 – busca 10 - execução

01 – indireto 11- interrupção



Controle do processador

- **Requisitos funcionais**

- Correspondem às funções que a unidade de controle tem que executar.
- Caracterização da unidade de controle em três passos:
 1. Definir **elementos básicos do processador**.
 2. Descrever as micro-operações que o processador executa.
 3. Definir as funções que a unidade de controle deve realizar para fazer com que as micro-operações sejam executadas.

Elementos básicos do processador: ULA, registradores, caminhos de dados internos (registradores \leftrightarrow ULA), caminhos de dados externos (registradores \leftrightarrow memória/módulos de E/S), unidade de controle.

Execução de programa \Rightarrow sequência de micro-operações que se enquadram em:

- Transferência de dados entre registradores
- Transferência de dados de um registrador para uma interface externa
- Transferência de dados de uma interface externa para um registrador
- Efetuar uma operação aritmética ou lógica, usando registradores de E/S.

Controle do processador

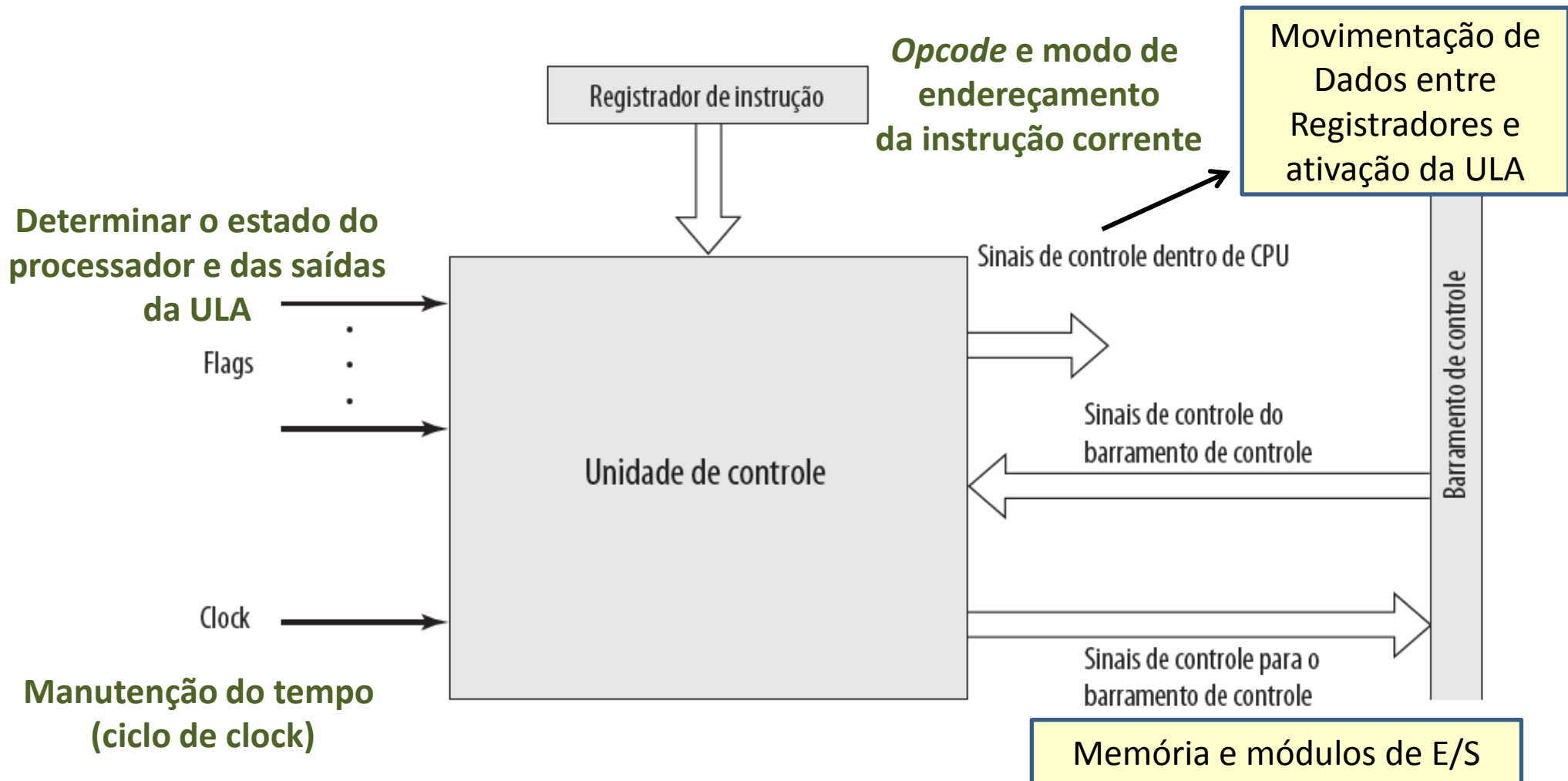
- Tarefas básicas da unidade de controle
- Sequenciamento
 - UC faz o processador executar uma série de micro-operações na sequência correta, com base no programa que está sendo executado.
- Execução
 - UC faz cada micro-operação ser executada.

A chave para o funcionamento da unidade de controle é o uso de **sinais de controle**.

Controle do processador

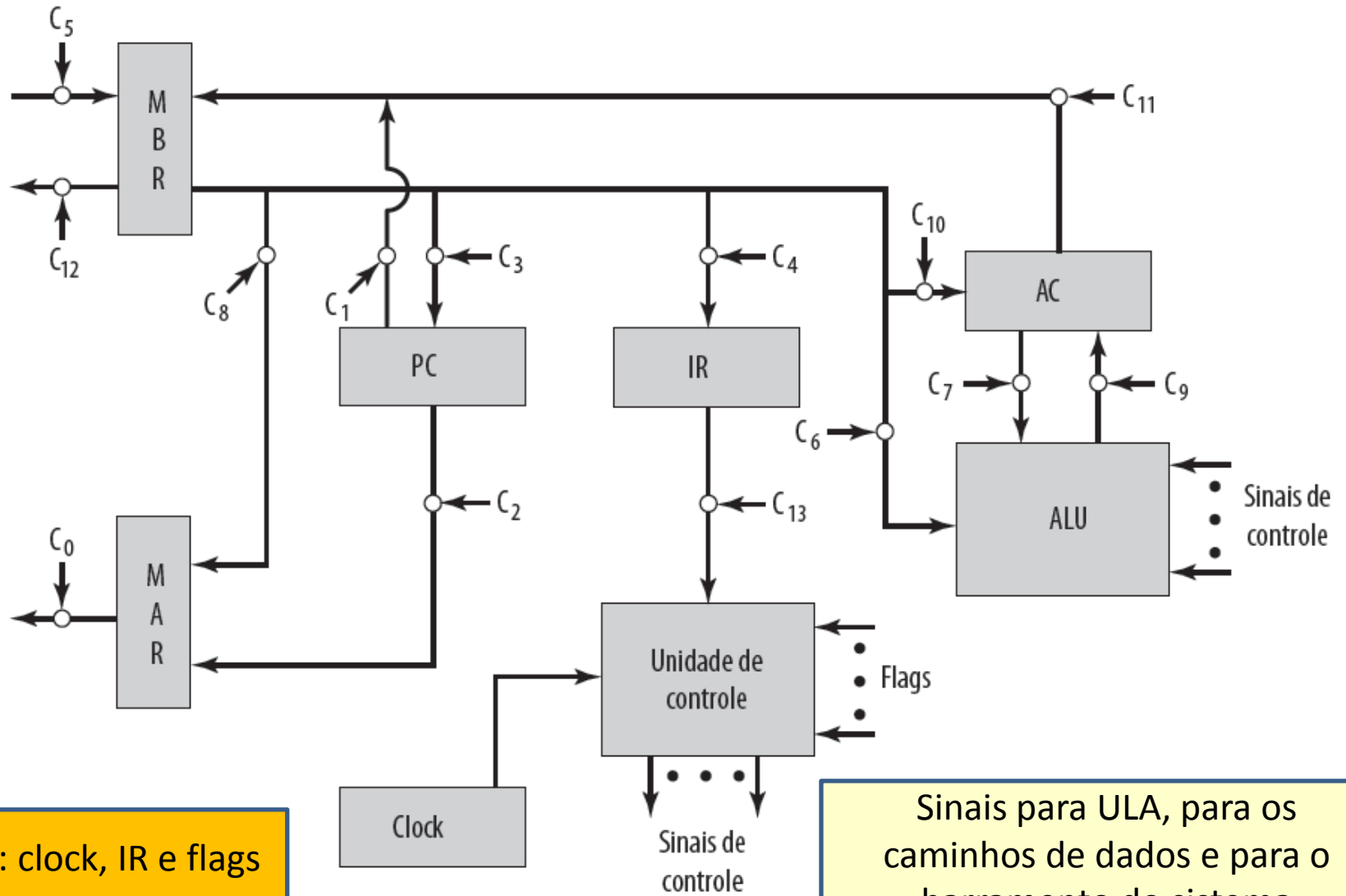
- Sinais de controle

- UC precisa de **entradas** que lhe permitam determinar o estado do sistema e de **saídas** que lhe permitam controlar o comportamento do sistema => **interfaces externas**.



• Exemplo de sinais de controle

- Ex.: processador simples com um acumulador (AC).
 - Caminhos de dados entre os elementos.
 - C_i indica terminação de controle para sinais oriundos da UC.



Controle do processador

- Relação entre micro-operações e sinais de controle

	Micro-operações	Sinais de controle ativos
Busca:	$t_1: \text{MAR} \leftarrow (\text{PC})$	C_2
	$t_2: \text{MBR} \leftarrow \text{Memória}$	C_5, C_R
	$\text{PC} \leftarrow (\text{PC}) + 1$	
	$t_3: \text{IR} \leftarrow \text{MBR}$	C_4
Indireto:	$t_1: \text{MAR} \leftarrow (\text{IR}(\text{Endereço}))$	C_8
	$t_2: \text{MBR} \leftarrow \text{Memória}$	C_5, C_R
	$t_3: \text{IR}(\text{Endereço}) \leftarrow (\text{MBR}(\text{Endereço}))$	C_4
Interrupção	$t_1: \text{MBR} \leftarrow (\text{PC})$	C_1
	$t_2: \text{MAR} \leftarrow \text{Endereço-salvar}$	
	$\text{PC} \leftarrow \text{Endereço-rotina}$	
	$t_3: \text{Memória (MBR)}$	C_{12}, C_W

C_R = Sinal de controle de leitura para o barramento de sistema.

C_W = Sinal de controle de escrita para o barramento de sistema.

Controle do processador

• Organização interna do processador

Existem várias opções de organização interna de barramento. Projeto objetiva simplicidade de layout e menor espaço físico.

Ex.: barramento único ligando ULA aos registradores.
Portas e sinais de controle para movimentação de dados do barramento para cada registrador e vice-versa.
Sinais de controle adicionais para a transferência de dados de/para barramento do sistema (externo) e operações na ULA.
Dois registradores temporários (Y e Z) para armazenamento de operandos e de resultados de operações na ULA.

Ex.: adicionar valor da memória para o registrador AC

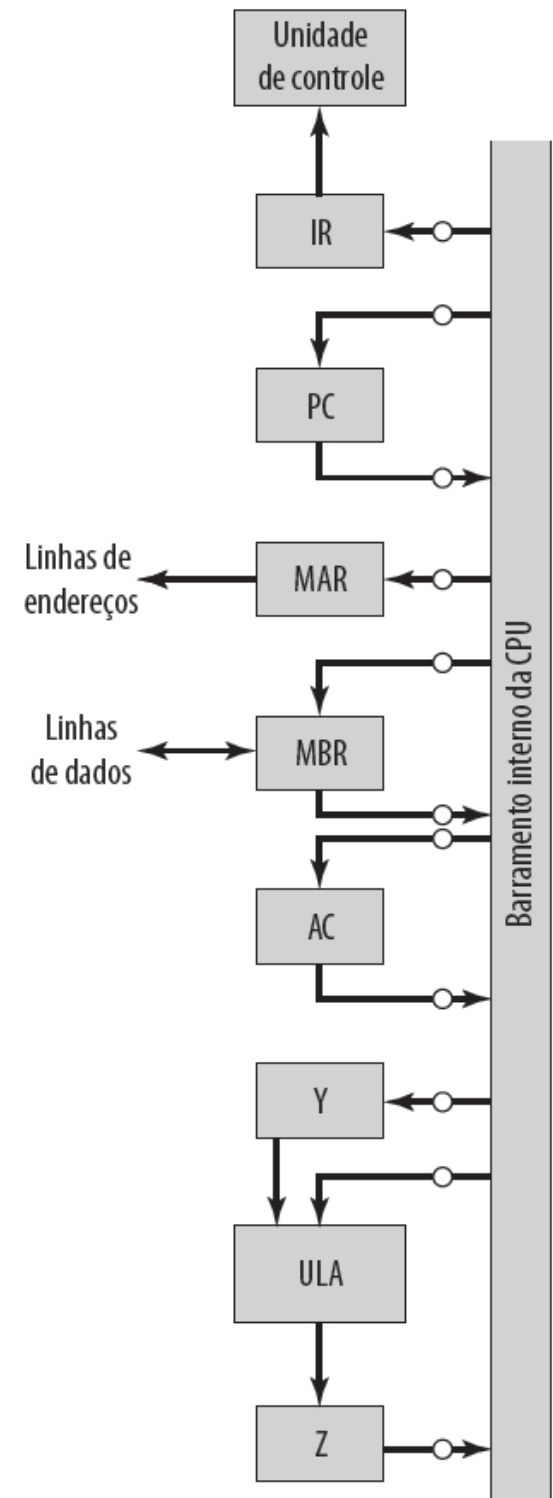
t1: **MAR** \leftarrow (**IR**(endereço))

T2: **MBR** \leftarrow **Memória**(**MAR**)

t3: **Y** \leftarrow (**MBR**)

t4: **Z** \leftarrow (**AC**) + (**Y**)

t5: **AC** \leftarrow (**Z**)



Controle do processador

- Implementação da unidade de controle

- Duas categorias de técnicas:

- Implementação por hardware

- UC é basicamente um circuito que implementa uma máquina de estados. Seus sinais lógicos de entrada se transformam em um conjunto de sinais lógicos de saída, que são os sinais de controle.

- Implementação microprogramada

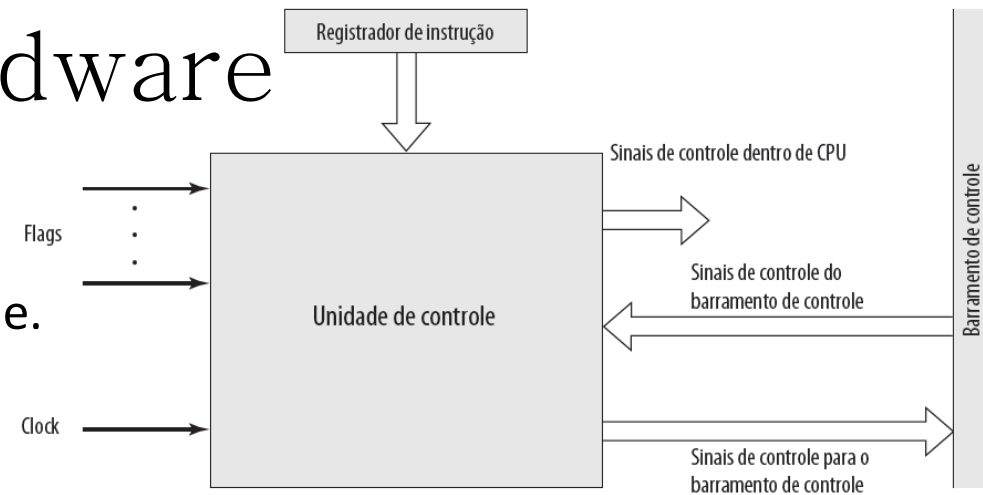
- A lógica da UC é especificada por um microprograma, o qual consiste de uma sequência de instruções em uma linguagem de microprogramação.

Implementação por hardware

UC – implementação por hardware

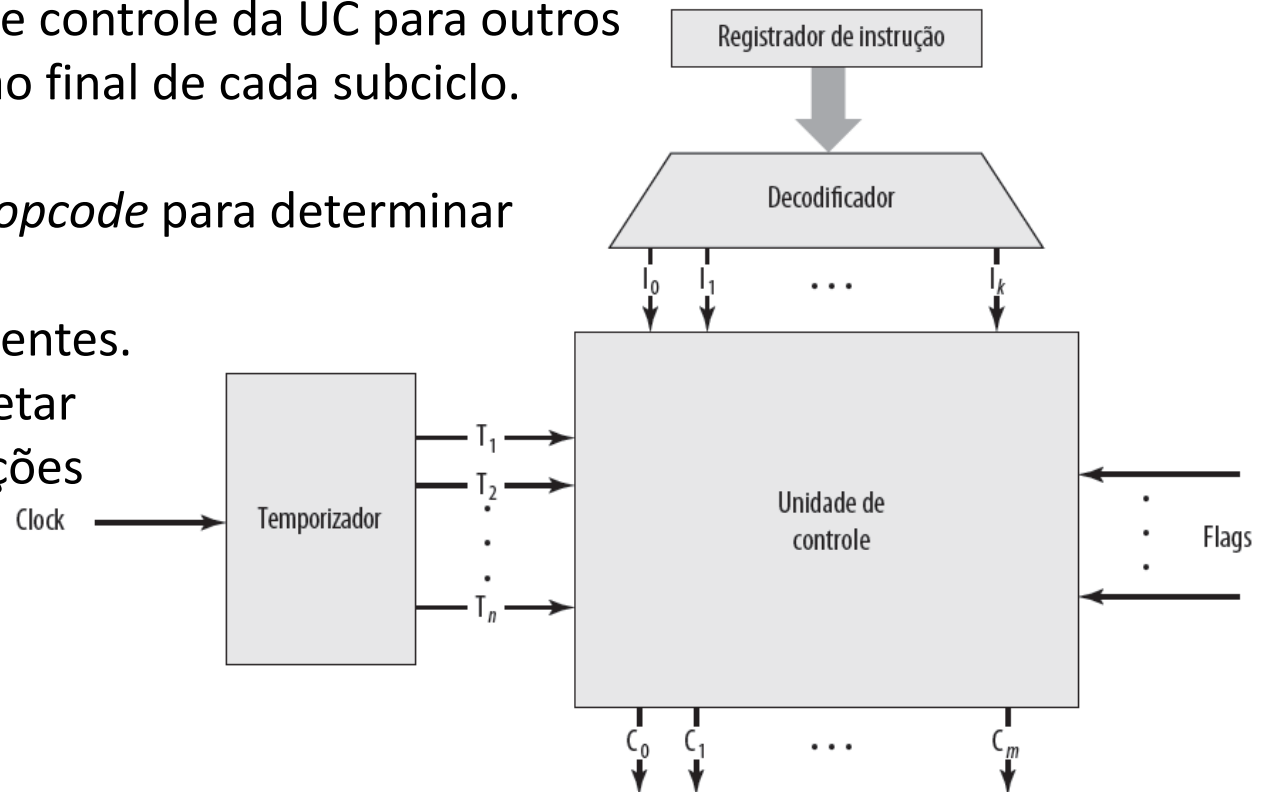
Entradas: RI, clock, flags e barramento de controle.

- **Flags e barramento de controle**: cada bit com significado específico.



- **Clock** envia sequência repetitiva de pulsos, usada para medir a duração das micro-operações. O período dos pulsos de clock deve ser longo o suficiente para permitir a propagação de sinais pelos caminhos de dados e pelos circuitos do processador. Precisa de um temporizador para sincronizar o envio de sinais de controle da UC para outros componentes, o qual é reiniciado ao final de cada subciclo.

- **Registrador de instrução**: UC usa *opcode* para determinar sequência de sinais de controle para cada uma das instruções diferentes. Usa um decodificador para interpretar os *opcodes* e gerar as micro-operações adequadas.



Lógica da unidade de controle

- Ex.: decodificador 2 X 4
- Expressão lógica para cada combinação de entrada.

E1	E2	S1	S2	S3	S4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Dois sinais de controle:

PQ = 00 ciclo de busca

PQ = 01 ciclo indireto

PQ = 10 ciclo de execução

PQ = 11 ciclo de interrupção

Ex.: sinal $C_5 \Rightarrow \text{MBR} \leftarrow \text{Memória}$
 Definido em t2 nos ciclos de busca
 e de indireto.

	Micro-operações	Sinais de controle ativos
Busca	$t_1: \text{MAR} \leftarrow (\text{PC})$	C_2
	$t_2: \text{MBR} \leftarrow \text{Memória}$	C_5, C_R
	$\text{PC} \leftarrow (\text{PC}) + 1$	
Indireto	$t_3: \text{IR} \leftarrow \text{MBR}$	C_4
	$t_1: \text{MAR} \leftarrow (\text{IR}(\text{Endereço}))$	C_8
	$t_2: \text{MBR} \leftarrow \text{Memória}$	C_5, C_R
	$t_3: \text{IR}(\text{Endereço}) \leftarrow (\text{MBR}(\text{Endereço}))$	C_4

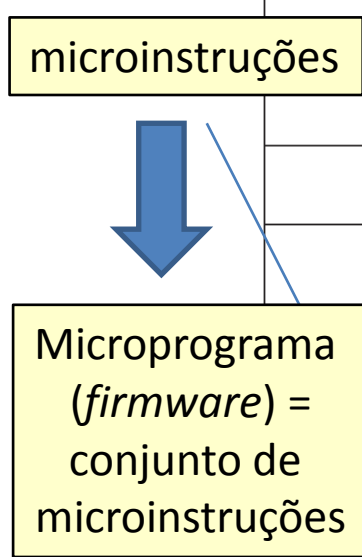
$$C_5 = \overline{P} \cdot \overline{Q} \cdot T2 + \overline{P} \cdot Q \cdot T2$$

C_5 também é necessário no ciclo de execução. Supondo que as operações LDA, ADD e AND leem dados da memória, a expressão ficaria:

$$C_5 = \overline{P} \cdot \overline{Q} \cdot T2 + \overline{P} \cdot Q \cdot T2 + P \cdot \overline{Q} \cdot (\text{LDA} + \text{ADD} + \text{AND}) \cdot T2$$

Controle microprogramado

- Mais flexível do que um projeto combinacional envolvendo lógica para sequenciamento de micro-operações, execução de instruções, interpretação de *opcodes* e de sinais de entrada da ULA.

Ordem	Efeito da ordem
 microinstruções	$C(Acc) + C(n)$ para Acc_1
$S n$	$C(Acc) - C(n)$ para Acc_1
$H n$	$C(n)$ para Acc_2
$V n$	$C(Acc_2) \times C(n)$ para Acc , onde $C(n) \geq 0$
$T n$	$C(Acc_1)$ para n , 0 para Acc
$U n$	$C(Acc_1)$ para n
$R n$	$C(Acc) \times 2^{-(n+1)}$ para Acc
$L n$	$C(Acc) \times 2^{n+1}$ para Acc
$G n$	IF $C(Acc) < 0$, transferir controle para n ; se $C(Acc) \geq 0$, ignorar (isto é, proceder serialmente)
$I n$	Ler próximo caractere do mecanismo de entrada para n
$O n$	Enviar $C(n)$ para mecanismo de saída

Acc = acumulador

Acc_1 = metade mais significativo do acumulador

Acc_2 = metade menos significativo do acumulador

n = localização de armazenamento n

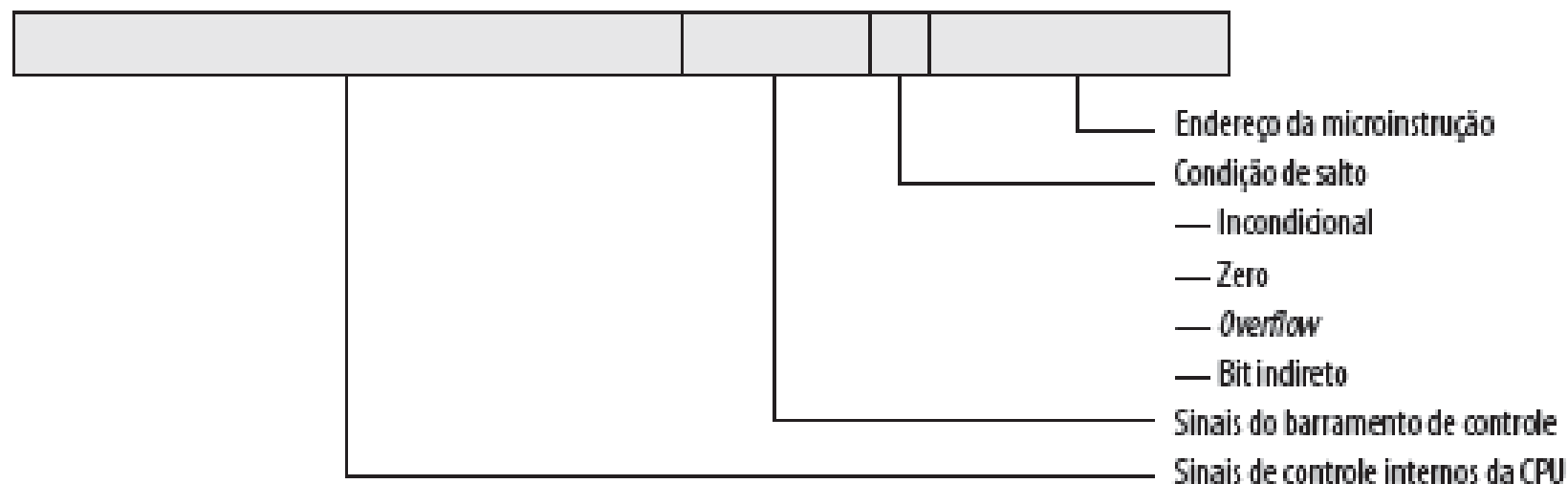
$C(X)$ = conteúdo de X (X = registrador da localização de armazenamento)

Controle microprogramado

- UC – implementação (controle) microprogramada
 - Usa sequências de instruções para controlar operações complexas.
 - Tudo o que a UC faz é gerar sinais de controle.
 - Cada sinal de controle está ligado ou desligado (dígito binário).
 - Como implementar?
 - Represente cada sinal de controle por um bit.
 - Tenha uma palavra de controle para cada micro-operação.
 - Tenha uma sequência de palavras de controle para cada instrução em código de máquina.
 - Acrescente um endereço para especificar a próxima microinstrução, dependendo das condições.

• UC – implementação (controle) microprogramada

– Microinstrução horizontal



Existe um bit para cada linha de controle interna do processador e um bit para cada linha de controle do barramento externo.

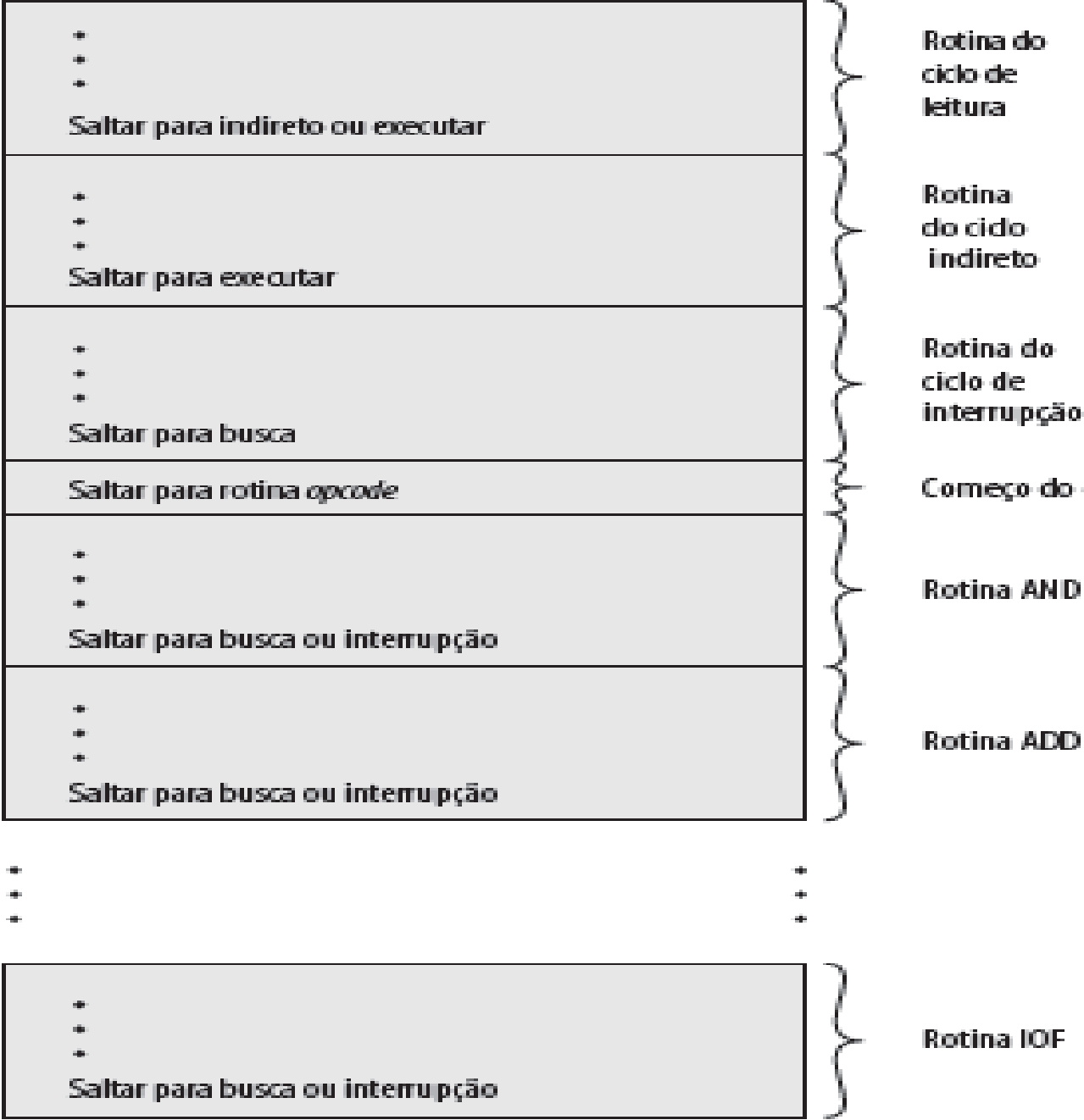
Há um campo de condição, indicando a condição em que deve haver um desvio.

Há um campo com o endereço da microinstrução caso o desvio seja tomado.

INTERPRETAÇÃO DA MICROINSTRUÇÃO

1. Ligar todas as linhas de controle indicados por bit 1; desligar as linhas indicadas por bit 0;
2. Se a condição indicada pelos bit de condição for falsa, executar a próxima instrução na sequência; do contrário, a próxima microinstrução a ser executada é indicada no campo de endereço.

– Memória de controle



Microinstruções de cada rotina são executadas sequencialmente.

Cada rotina termina com uma instrução de salto ou desvio.

Existe um ciclo de execução especial que sinaliza que uma das rotinas de instrução de máquina (AND, ADD etc) está para ser executada, dependendo do *opcode* corrente.

• UC – implementação (controle) microprogramada

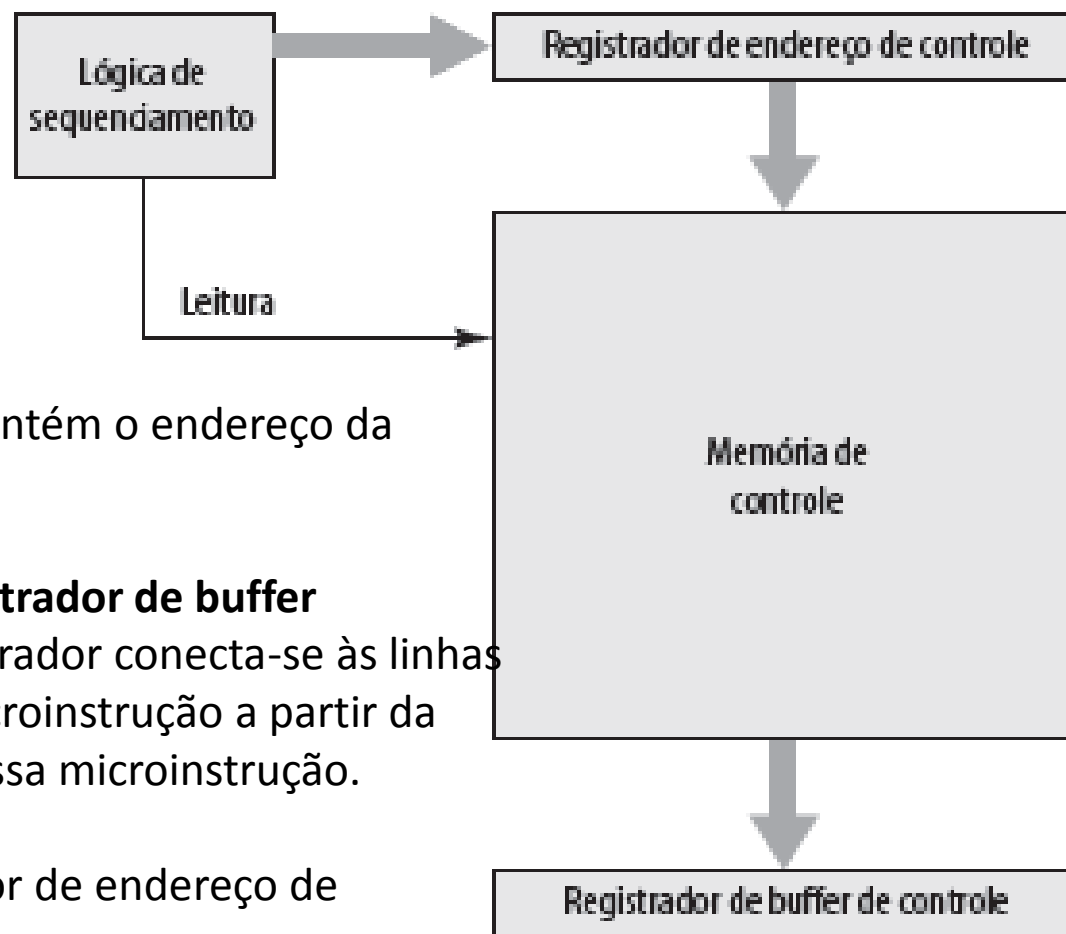
– Micro-arquitetura da unidade de controle

O conjunto de microinstruções é armazenado na **memória de controle**.

O **registrador de endereço de controle (REC)** contém o endereço da próxima microinstrução a ser lida.

A microinstrução lida é transferida para um **registrador de buffer de controle (RBC)**. A parte esquerda desse registrador conecta-se às linhas de controle que saem da UC. Assim, *ler* uma microinstrução a partir da memória de controle é o mesmo que *executar* essa microinstrução.

A **lógica de sequenciamento** carrega o registrador de endereço de controle e emite um comando de leitura.



FUNCIONAMENTO DESSA MICRO-ARQUITETURA NO PRÓXIMO SLIDE!!

• UC – implementação (controle) microprogramada

– Micro-arquitetura da unidade de controle

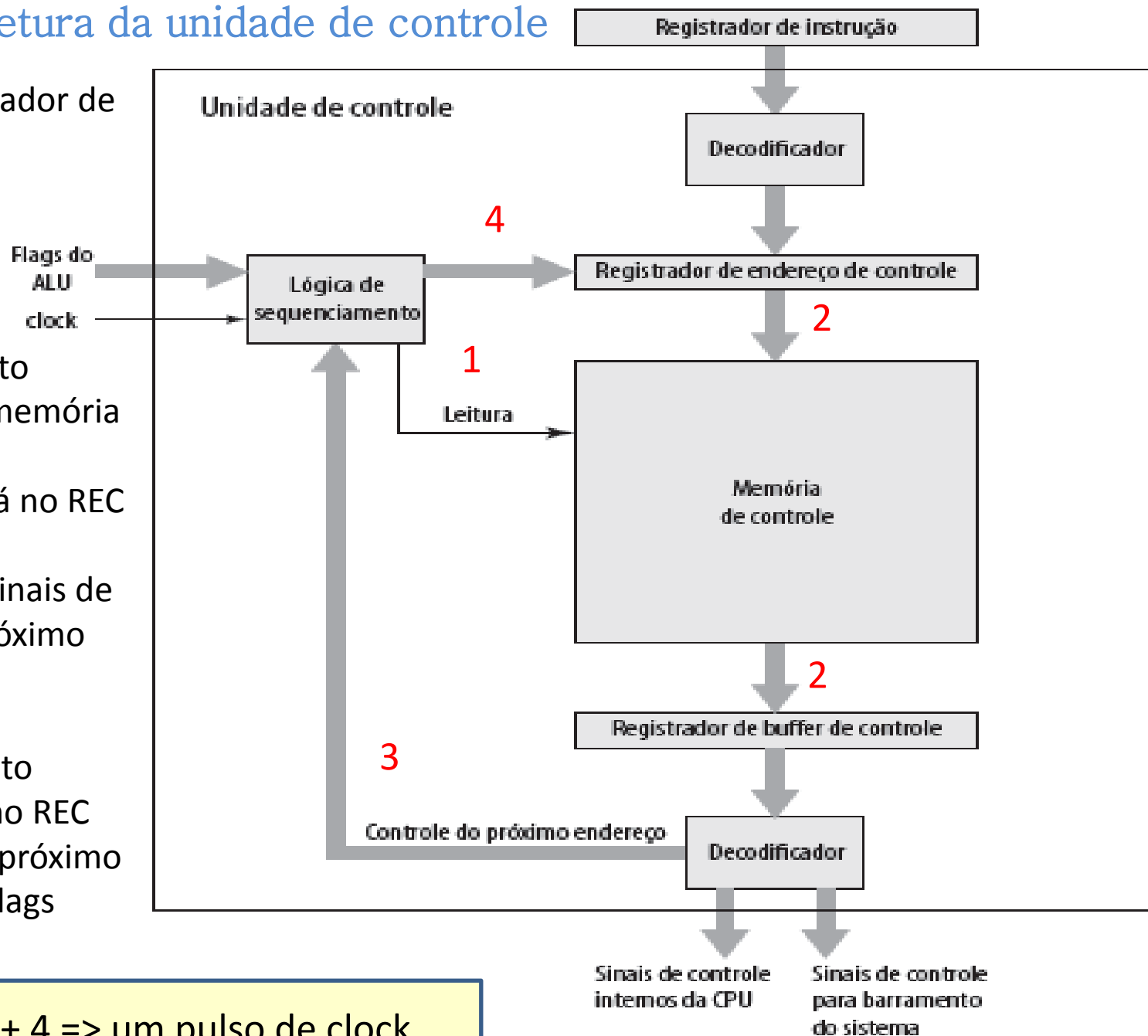
Entradas: flags, clock, registrador de instrução e ULA).

Saídas: sinais de controle.

FUNCIONAMENTO

1. Lógica de sequenciamento emite comando READ para memória de controle.
2. Palavra cujo endereço está no REC é lida para dentro do RBC.
3. O conteúdo de RBC gera sinais de controle e informação do próximo endereço para a lógica de sequenciamento.
4. A lógica de sequenciamento carrega um novo endereço no REC com base na informação do próximo endereço a partir do RBC e flags da ULA.

1 + 2 + 3 + 4 => um pulso de clock



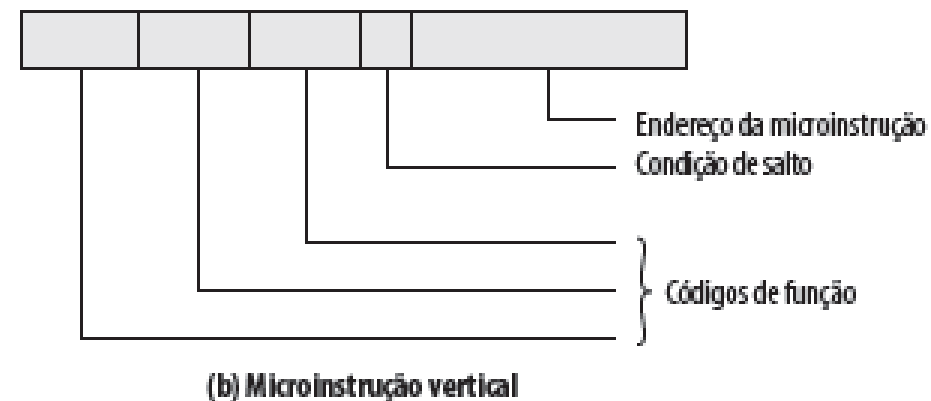
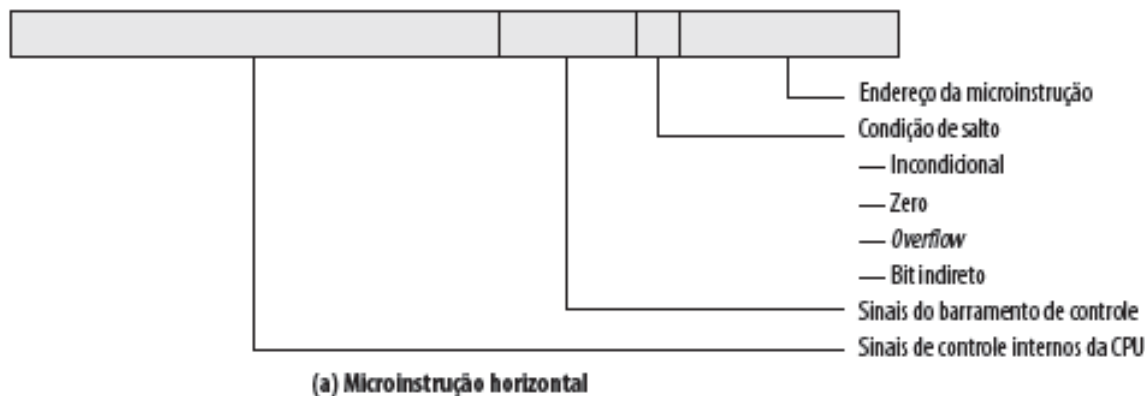
• UC – implementação microprogramada

– Microinstruções horizontais

- Palavra de memória larga
- Alto grau de operações paralelas possíveis
- Pouca codificação de informações de controle

– Microinstruções verticais

- Palavra de memória mais estreita (menos bits).
- N sinais de controle codificados em $\log_2 N$ bits.
- Capacidade limitada para expressar paralelismo.
- Requer decodificador externo para identificar a linha de controle exata sendo manipulada.



Controle microprogramado

- Sequenciamento de microinstruções

- Obter a próxima microinstrução da memória de controle.
- Endereço da próxima microinstrução:
 - Determinado pelo registrador de instrução.
 - Próximo endereço sequencial.
 - Desvio.
- O projeto deve considerar:
 - Tamanho da microinstrução.
 - Minimizar o tamanho da memória de controle.
 - Tempo de geração de endereço.
 - Executar as microinstruções o mais rápido possível.

Controle microprogramado

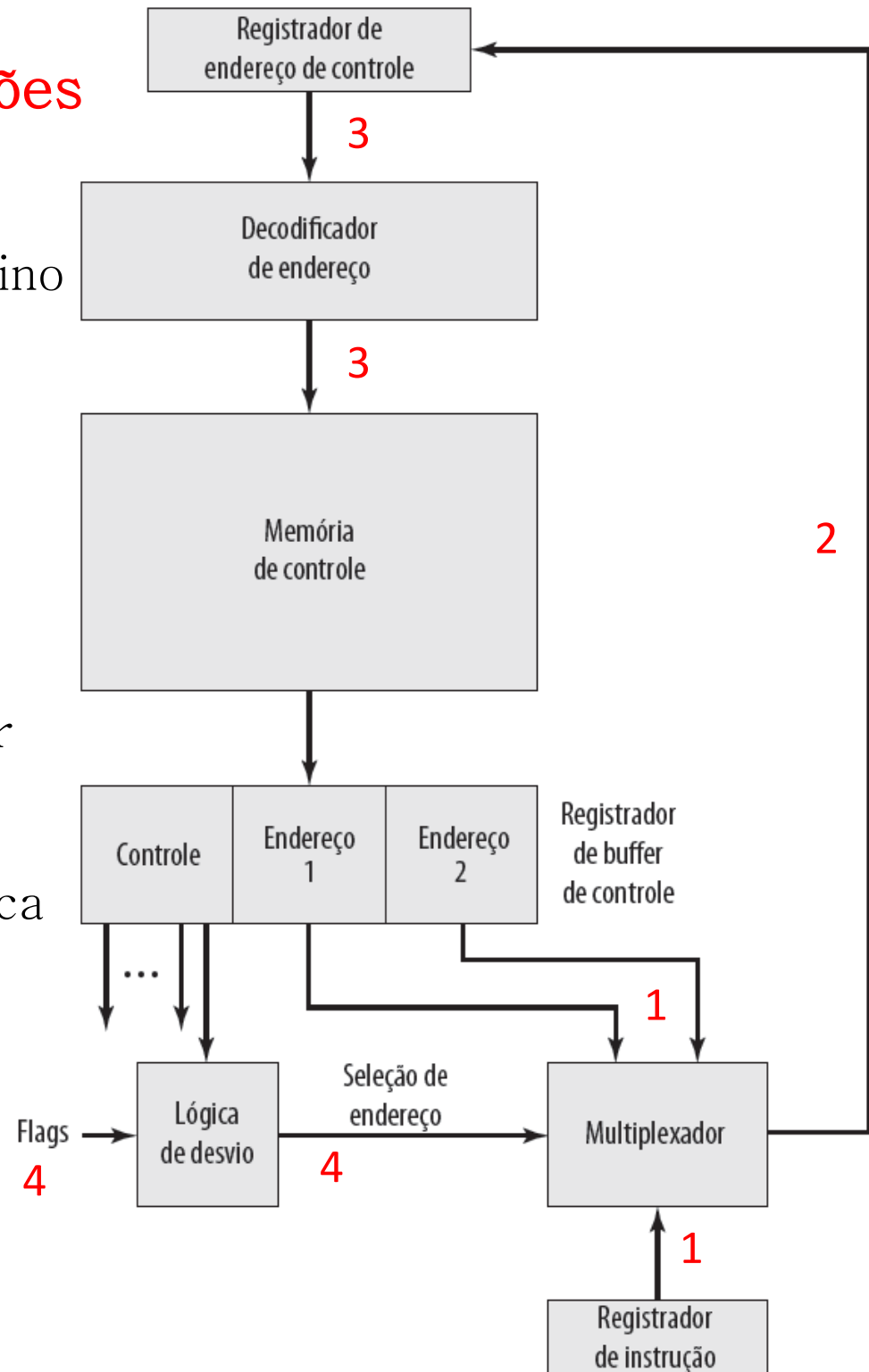
- Sequenciamento de microinstruções

- Técnicas de sequenciamento
- Com base na microinstrução corrente, nos flags de condição e no conteúdo do registrador de instrução, um endereço de memória de controle deve ser gerado para a próxima microinstrução.
- Diferentes técnicas, baseadas no formato da informação de endereço na microinstrução:
 - Dois campos de endereço
 - Campo de endereço único
 - Formato variável

• Sequenciamento de microinstruções

– Dois campos de endereço

1. Um multiplexador serve como destino para os campos de endereço e para o registrador de instrução.
2. Com base numa entrada de seleção de endereço, o mux transmite o *opcode* ou um dos dois endereços para o registrador de endereço de controle (REC).
3. O REC é decodificado para produzir o endereço da próxima instrução.
4. Os sinais de seleção de endereço são fornecidos por um módulo de lógica de desvio cuja entrada consiste de flags da UC mais os bits da parte de controle da microinstrução.



• Sequenciamento de microinstruções

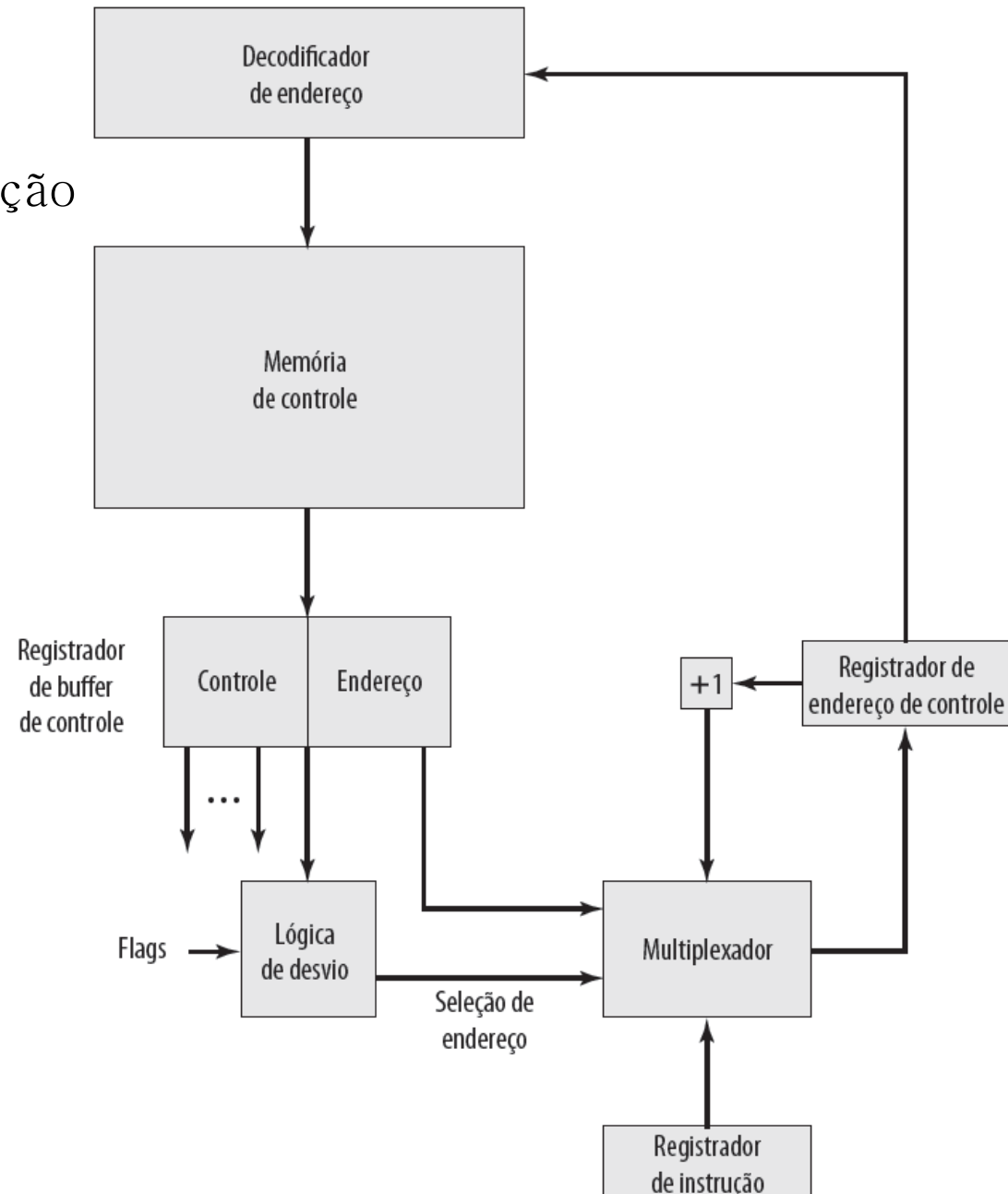
- Campo de endereço único

Opções para o próximo endereço:

- Campo de endereço
- Código do registrador de instrução
- Próximo endereço sequencial

Sinais de seleção de endereço determinar qual opção está selecionada.

Requer somente um campo de endereço.



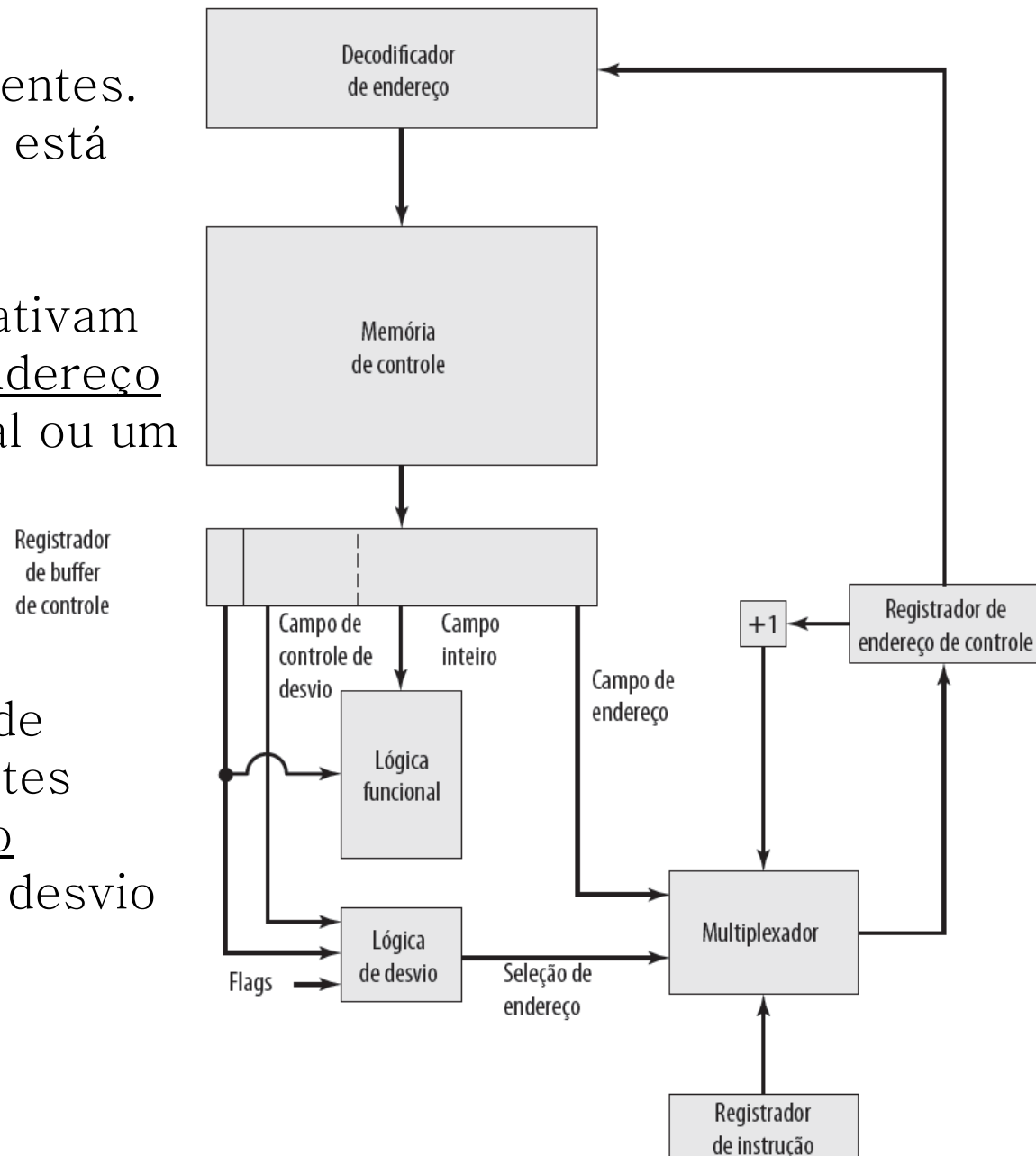
Sequenciamento de microinstruções

- Formato variável

Dois formatos de instrução diferentes.
Um bit para definir qual formato está sendo usado.

Num formato, os bits restantes ativam sinais de controle. O próximo endereço É o próximo endereço sequencial ou um endereço derivado a partir do registrador de instrução.

No outro formato, alguns bits conduzem o módulo de lógica de desvio e os bits restantes fornecem o endereço. O próximo endereço é especificado por um desvio condicional ou incondicional.



Controle microprogramado

- Sequenciamento de microinstruções

- Geração de endereços
- Outro ponto importante a considerar no sequenciamento de instruções é a geração do endereço da próxima instrução, que pode ser derivado ou calculado.
- As técnicas disponíveis podem ser explícitas ou implícitas.

- Explícitas

- Endereço está disponível explicitamente na microinstrução.

- Implícitas

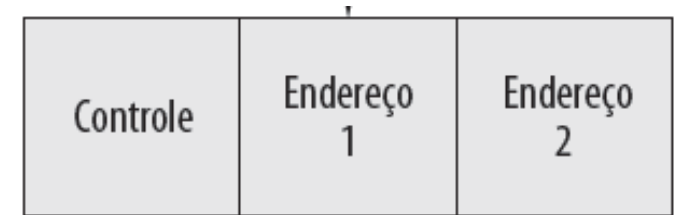
- Requerem lógica adicional para gerar o endereço.

Explícita	Implícita
Dois campos	Mapeamento
Desvio condicional	Adição
Desvio incondicional	Controle residual

Controle microprogramado

- Sequenciamento de microinstruções

- Abordagem explícita

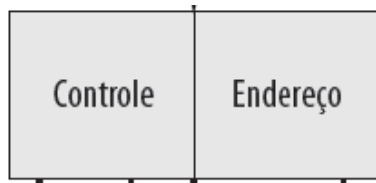


- Dois campos:

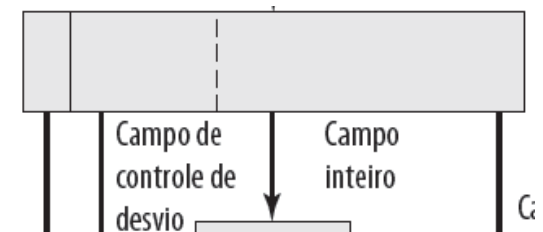
- Dois endereços alternativos estão disponíveis no formato da microinstrução.

- Campo de endereço único / formato variável:

- Várias instruções de desvio condicional e incondicional podem ser implementadas.
 - Desvio condicional depende das seguintes informações:



- Flags da ULA.
 - Parte do opcode ou campo de modo de endereço da instrução de máquina.
 - Partes do registrador selecionado, como o bit de sinal.
 - Bits de status dentro da UC.



Controle microprogramado

- Sequenciamento de microinstruções

- Abordagem implícita

- Mapeamento:

- Necessária em quase todos os projetos.
 - O opcode de uma instrução deve ser mapeado em um endereço de microinstrução. Isso ocorre apenas uma vez por ciclo de instrução.

- Combinação ou adição:

- Endereço completo formado a partir da combinação ou adição de duas portas de um endereço.
 - Ex.: IBM S/360, IBM S/370

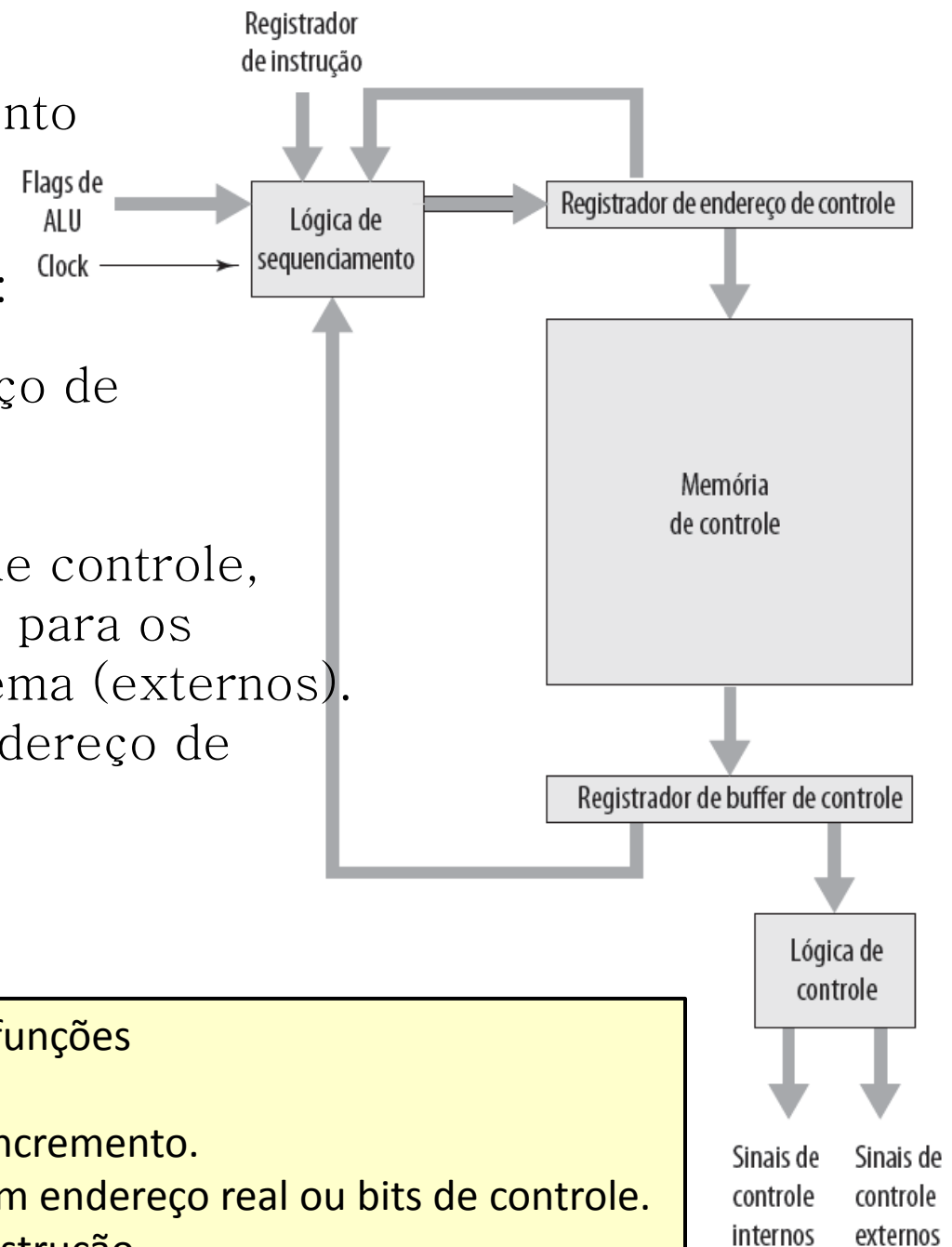
- Controle residual:

- Uso de um endereço de microinstrução que foi salvo previamente em armazenamento temporário dentro da UC (em uma pilha de registradores ou registrador interno).
 - Ex.: versão LSI-11 (do PDP-11).

• Execução de microinstruções

- O ciclo de microinstrução é o evento básico num processador microprogramado.
- Cada ciclo é feito de duas etapas:

- **Busca:** geração de um endereço de microinstrução.
- **Execução:** geração de sinais de controle, para o processador (internos) e para os barramentos/interfaces do sistema (externos). De forma secundária, gera o endereço de uma microinstrução.



Lógica de sequenciamento: lógica para efetuar as funções (gera o endereço da próxima microinstrução).

Registrador de endereço de controle (REC): para incremento.

Registrador de buffer de controle (RBC): fornece um endereço real ou bits de controle.

Clock: determina o tempo de ciclo de uma microinstrução.

Lógica de controle: gera sinais de controle em função de alguns bits de microinstrução.

• Taxonomia de microinstruções

- Vertical X horizontal
- Empacotada X não-empacotada
- Microprogramação soft X hard
- Codificação direta X indireta

Dependem do formato da instrução

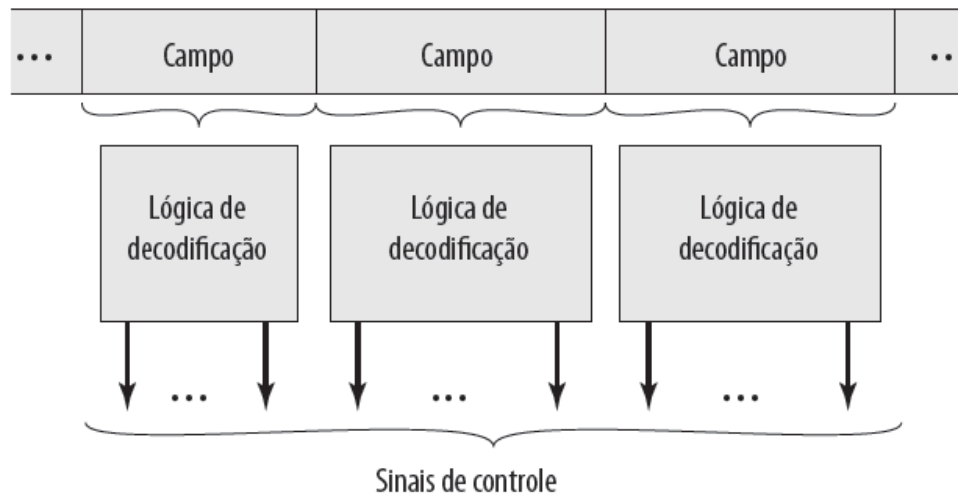
Espectro de microinstruções

Características	
Não codificado	Altamente codificado
Muitos bits	Poucos bits
Visão detalhada de hardware	Visão agregada de hardware
Dificuldade de programar	Facilidade de programar
Concorrência totalmente explorada	Concorrência não totalmente explorada
Pequena ou nenhuma lógica de controle	Lógica de controle complexa
Execução rápida	Execução lenta
Desempenho otimizado	Programação otimizada
Terminologia	
Não empacotada	Empacotada
Horizontal	Vertical
Hard	Soft

• Taxonomia de microinstruções

– Codificação direta X indireta

- UCs microprogramadas não são projetadas usando um formato de microinstrução puramente horizontal ou não codificado. Pelo menos algum grau de codificação é usado para reduzir o tamanho da memória de controle e para simplificar a tarefa de microprogramação

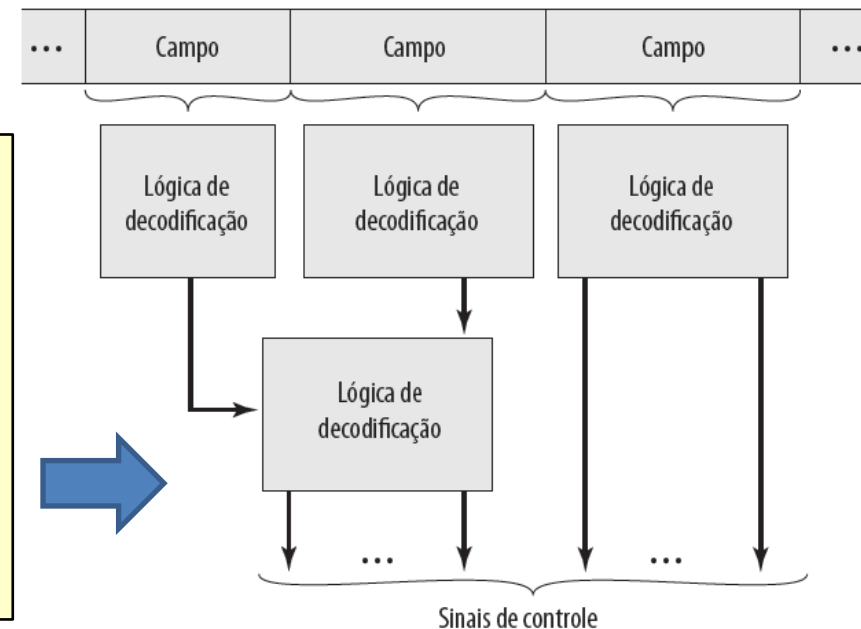


CODIFICAÇÃO DIRETA

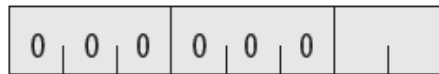
- Microinstrução composta por um conjunto de campos.
- Cada campo, após decodificado, gera um ou mais sinais de controle.
- N campos $\Rightarrow 2^N$ ações simultâneas
- Campo com L bits $\Rightarrow 2^L$ códigos diferentes

CODIFICAÇÃO INDIRETA

- Um campo é usado para determinar a interpretação de outro campo.
- Ex.: ULA que executa 8 operações aritméticas e 8 operações de deslocamento diferentes.
- Campo de 1 bit para indicar operação aritmética ou deslocamento + 3 bits para indicar qual das 8 operações.
- 2 níveis de decodificação \Rightarrow maior atraso de propagação.



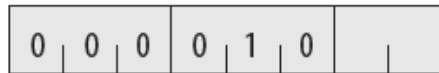
Transferências simples entre registradores



MDR ← Registrador



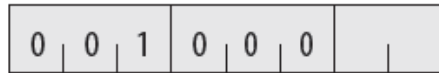
Registrador ← MDR



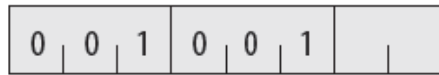
MAR ← Registrador

Seleção de registrador

Operações de memória



Leitura

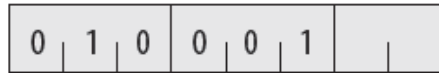


Escrita

Operações especiais de sequenciamento



CSAR ← MDR decodificado



CSAR ← Constante (no próximo byte)



Pular

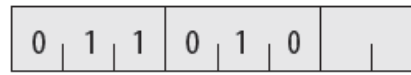
Operações de ALU



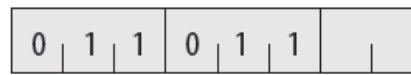
ACC ← ACC + Registrador



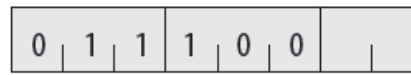
ACC ← ACC - Registrador



ACC ← Registrador



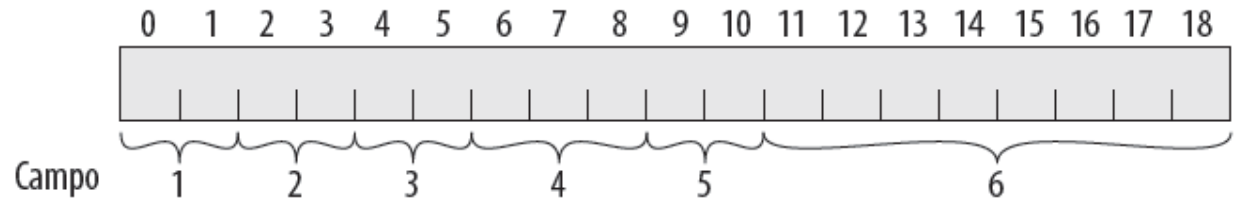
Registrador ← ACC



ACC ← Registrador + 1

Seleção de registrador

Formato vertical de microinstrução



Campo

Definição de campo

1 – transferência entre registradores

2 – operação de memória

3 – operação de sequenciamento

4 – operação da ALU

5 – seleção de registrador

6 – constante

Formato horizontal de microinstrução