Arquitetura de Computadores

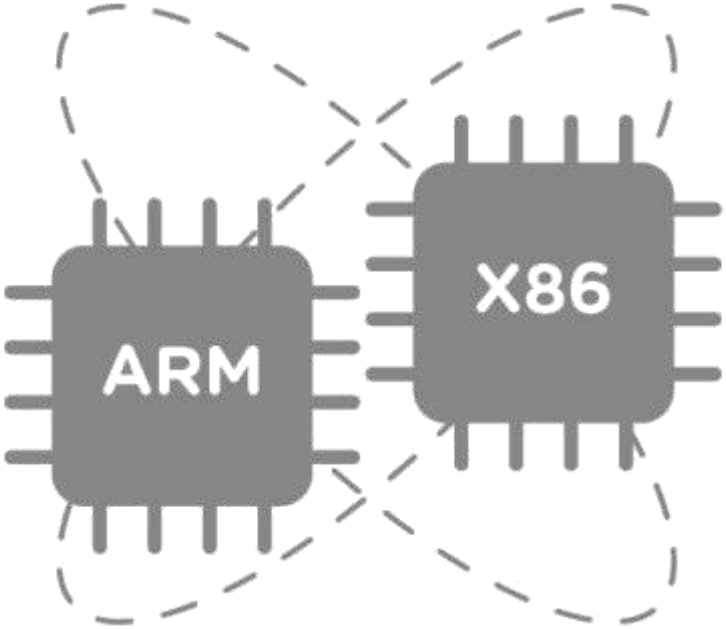
Arquiteturas RISC e CISC

Prof. Dr. Daniel Caetano

2022 - 1

Compreendendo o problema

- **Situação:** Existem duas siglas que permeiam boa parte da discussão tecnológica nas últimas décadas...

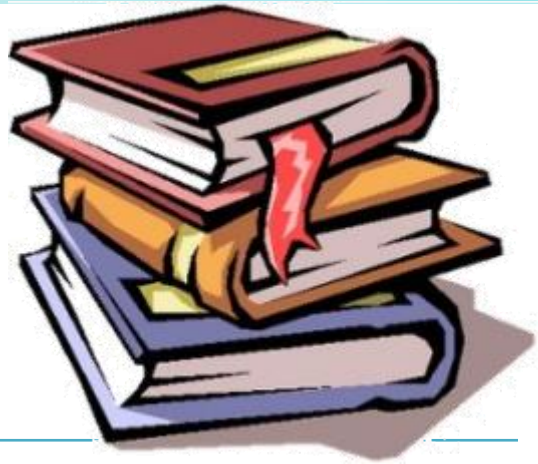


Qual a diferença?

Objetivos

- Conhecer a Arquitetura CISC
- Conhecer a Arquitetura RISC
- Compreender as diferenças entre RISC e CISC
- Compreender o conceito de microprogramação

Material de Estudo



Material	Acesso ao Material
Notas de Aula e Apresentação	https://www.caetano.eng.br/aulas/2022a/ara0039.php (Arquitetura de Computadores – Aula 11)
Biblioteca Virtual	• Arquitetura e Organização de Computadores (Stallings)
Material	• Conteúdo digital ambiente Aura

Introdução Conceitual

Introdução

- Até fim da década de 1970...
  - Memórias muito lentas
  - Ler instruções era lento
  - Criar instruções que faziam várias coisas ao mesmo tempo
  - Operações complexas com menos bytes
- Mais para a frente...
  - Instruções complexas são pouco usadas
  - Processador ficava mais lento, em geral...
  - ...porque é complexo implementar pipeline

Introdução

- Com o aumento da velocidade das memórias...
  - O ganho com as instruções complexas se reduziu
- CISC – Complex Instruction Set Computer
  - Instruções Complexas
- RISC – Reduced Instruction Set Computer
  - Instruções Simples

ARQUITETURA CISC

Arquitetura CISC

- Nos primórdios, gargalo era acesso a memória
- Não há como evitar: busca de instrução
- Solução: reduzir a busca de instruções
  - Realizando muitas tarefas com uma única instrução
  - Exemplo... cópia de dados de uma região para a outra da memória

Arquitetura CISC

- Cópia de 0x500 bytes da posição 0x1000 para a posição 0x2000, com instruções simples:

```
        LD    BC, 0500h          ; Número de bytes
        LD    HL, 01000h         ; Origem
        LD    DE, 02000h         ; Destino
        ; Aqui começa a rotina de cópia
COPIA:  LD    A, (HL)            ; Carrega byte da origem
        INC   HL                 ; Aponta próximo byte de  origem em HL
        LD    (DE), A            ; Coloca byte no destino
        INC   DE                 ; Aponta próximo byte de destino em HL
        DEC   BC                 ; Decrementa 1 de BC
        LD    A,B                ; Coloca valor de B em A
        OR    C                  ; Soma os bits ligados de C em A
        ; Neste ponto A só vai conter zero se B e C eram zero
        JP    NZ,COPIA           ; Continua cópia enquanto A != zero
```

Arc

Funciona, mas...

8 bytes de instruções e
2 bytes de dados
no processo da cópia

- Có 1000 para a p mples:

```
        LD    BC, 0500h          ; Número de bytes
        LD    HL, 01000h         ; Origem
        LD    DE, 02000h         ; Destino
        ; Aqui começa a rotina de cópia
COPIA:  LD    A, (HL)            ; Carrega byte da origem
        INC   HL                 ; Aponta próximo byte de  origem em HL
        LD    (DE), A            ; Coloca byte no destino
        INC   DE                 ; Aponta próximo byte de destino em HL
        DEC   BC                 ; Decrementa 1 de BC
        LD    A,B                ; Coloca valor de B em A
        OR    C                  ; Soma os bits ligados de C em A
        ; Neste ponto A só vai conter zero se B e C eram zero
        JP    NZ,COPIA           ; Continua cópia enquanto A != zero
```

Arc

Que tal uma instrução que faz cópia?

- Có 1000 para a p mples:

```
        LD    BC, 0500h          ; Número de bytes
        LD    HL, 01000h         ; Origem
        LD    DE, 02000h         ; Destino
        ; Aqui começa a rotina de cópia
COPIA:  LD    A, (HL)            ; Carrega byte da origem
        INC   HL                 ; Aponta próximo byte de  origem em HL
        LD    (DE), A            ; Coloca byte no destino
        INC   DE                 ; Aponta próximo byte de destino em HL
        DEC   BC                 ; Decrementa 1 de BC
        LD    A,B                ; Coloca valor de B em A
        OR    C                  ; Soma os bits ligados de C em A
        ; Neste ponto A só vai conter zero se B e C eram zero
        JP    NZ,COPIA           ; Continua cópia enquanto A != zero
```

Arquitetura CISC

- Cópia de 0x500 bytes da posição 0x1000 para a posição 0x2000, com instrução complexa

```
LD      BC, 0500h           ; Número de bytes
LD      HL, 01000h          ; Origem
LD      DE, 02000h          ; Destino
; Aqui começa a rotina de cópia
LDIR                        ; Realiza cópia
```

Arc

Uma única instrução

2 bytes de instrução

Redução de 80%!

- Có 1000 para a p mplexa

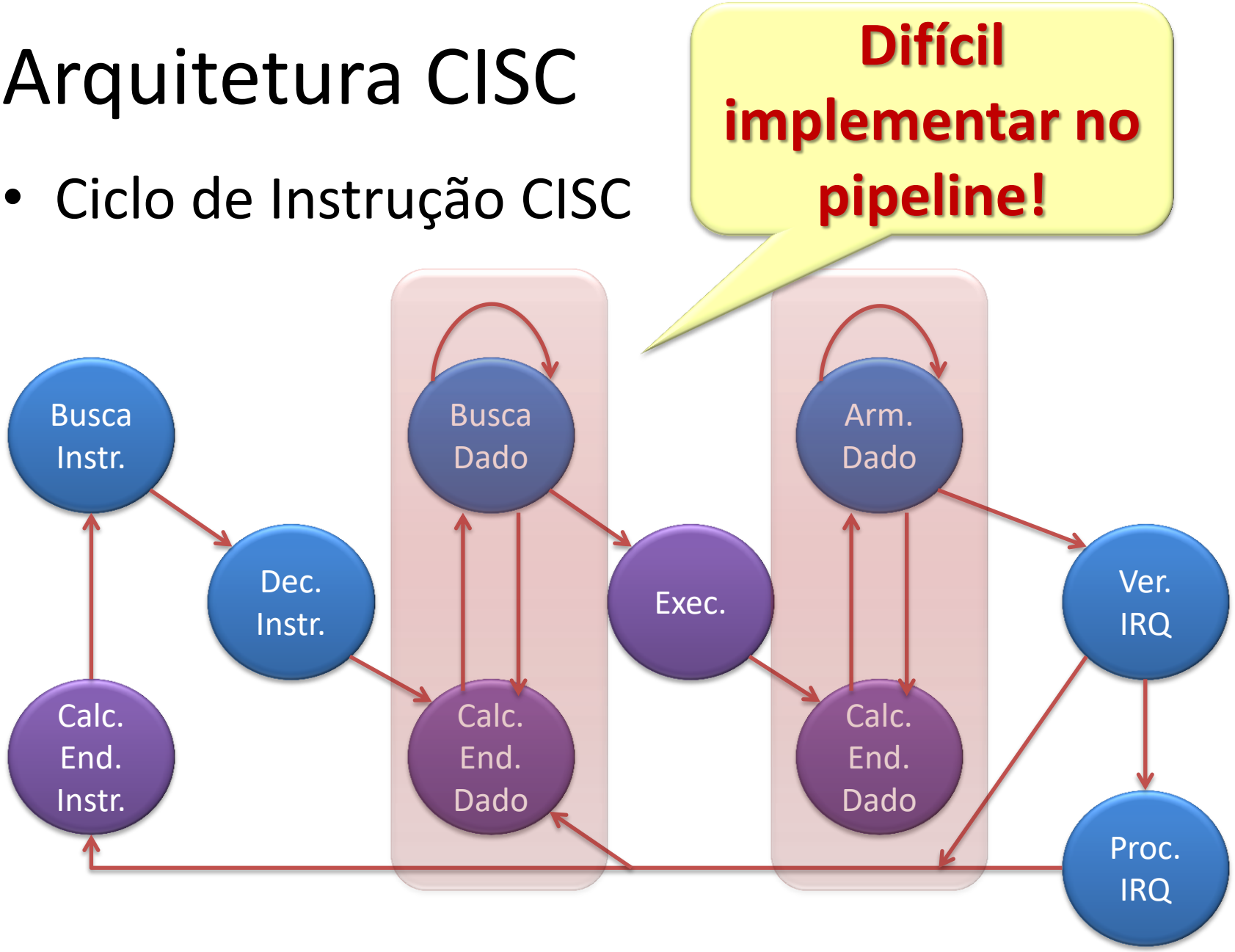
```
LD      BC, 0500h           ; Número de bytes
LD      HL, 01000h          ; Origem
LD      DE, 02000h          ; Destino
; Aqui começa a rotina de cópia
LDIR                        ; Realiza cópia
```

Arquitetura CISC

- Mas essas instruções complexas são comuns?
  1. **Atribução:** 47%
  2. **Comparação/If:** 23%
  3. **Chamadas de Função:** 15%
  4. **Loops:** 6%
  5. **Saltos Simples:** 3%
  6. **Outras Operações:** 7%
- Ganhos com operações complexas...
  - Existem, mas são limitados!

Arquitetura CISC

- Ciclo de Instrução CISC



ARQUITETURA RISC

Arquitetura RISC

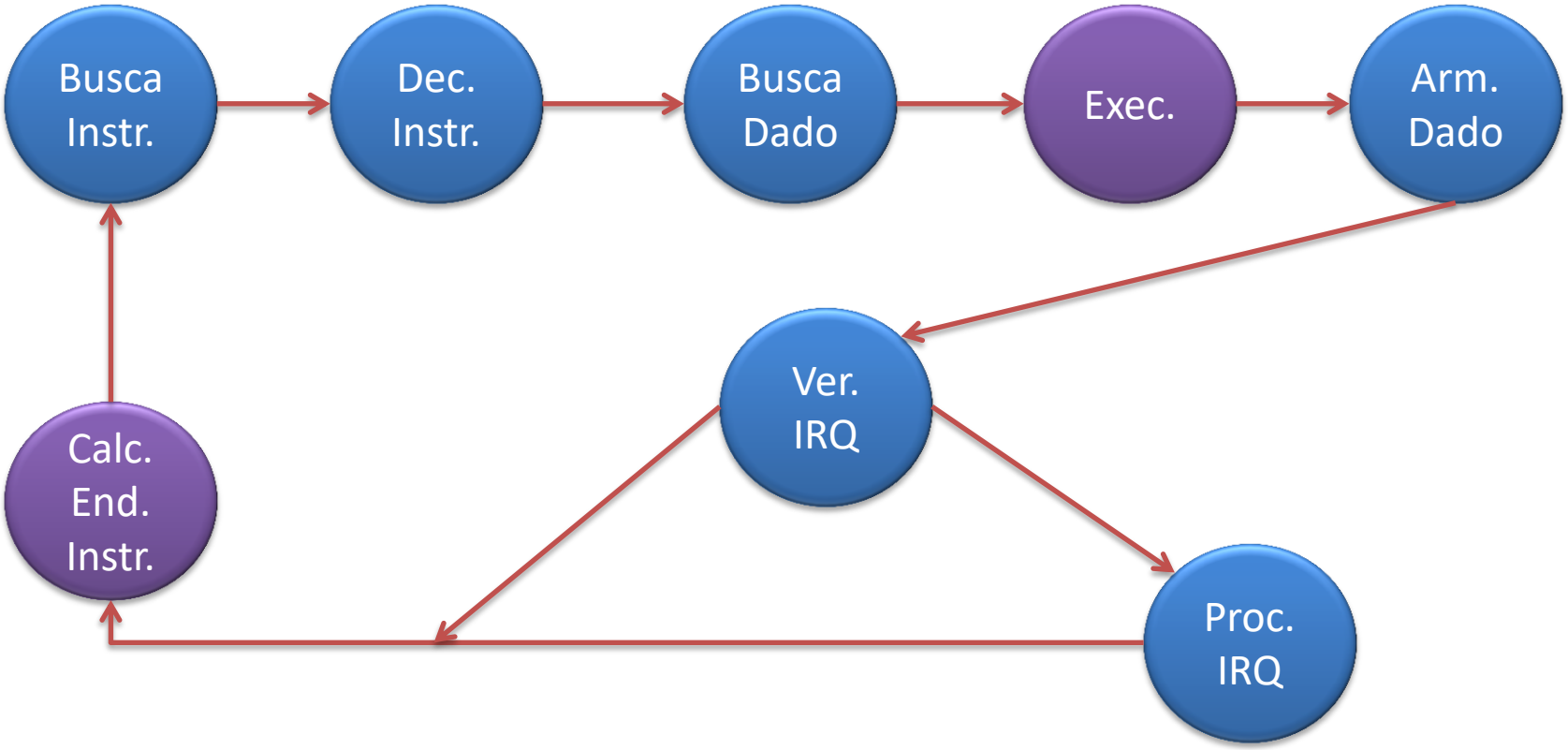
- Aumento do desempenho das memórias
  - Otimizar instruções mais comuns
  - Atribuições, comparações, saltos...
- CPUs mais simples e mais rápidas
  - Acesso à memória: apenas **load** e **store**
    - Reduz a interdependência das instruções
    - Facilita superpipeline

Arquitetura RISC

- 5 Características RISC marcantes
  - **Instruções de tamanho fixo**
    - **U**ma palavra
    - 16, 32, 64 bits...
  - **Execução em UM ciclo de clock**
  - **Processamento apenas em registradores**
    - Nada de ADD A,10
  - **Não há endereçamentos complexos**
    - Nada de registradores de índice como LD A,(IX+10)
  - **Grande número de registradores gerais**
    - Usualmente não há registradores específicos
    - Nomeados de R1 a Rnn

Arquitetura RISC

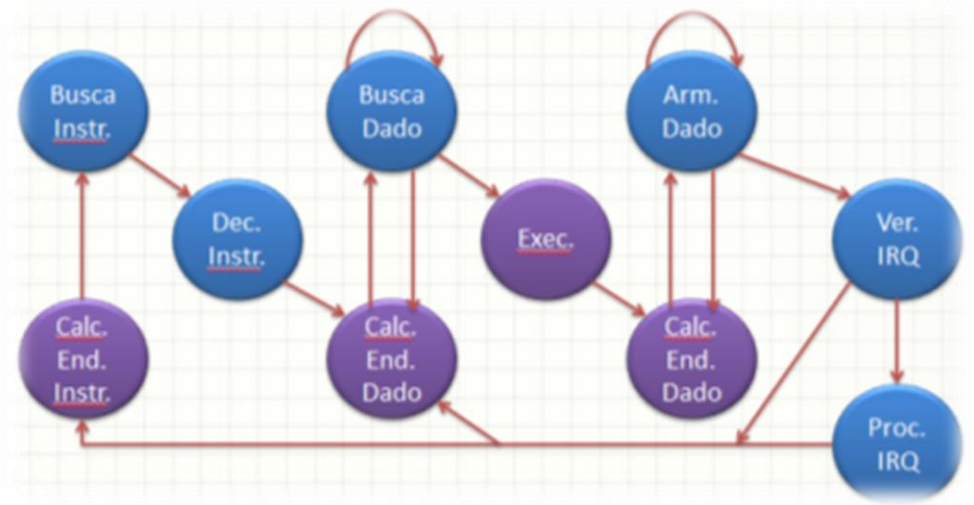
- Ciclo de Instrução RISC – Instruções Gerais



DESEMPENHO

CISC x RISC

- Complexidade



CISC

RISC

CISC x RISC

- Cálculo de Speedup

$$S = 100 * \frac{(Ts - Tc)}{Tc}$$

- S: **S**peedup
- Ts: **T**empo **S**em otimização (CISC)
- Tc: **T**empo **C**om otimização (RISC)

CISC x RISC

- Cálculo dos Tempos de Processamento

$$T = NI * CPI * P$$

- T: **T**empo de Processamento
- NI: **N**úmero de **I**nstruções
- CPI: **C**iclos de clock **P**or **I**nstrução
- P: **P**eríodo (tempo de cada ciclo)

P = 1 / frequencia

CISC x RISC

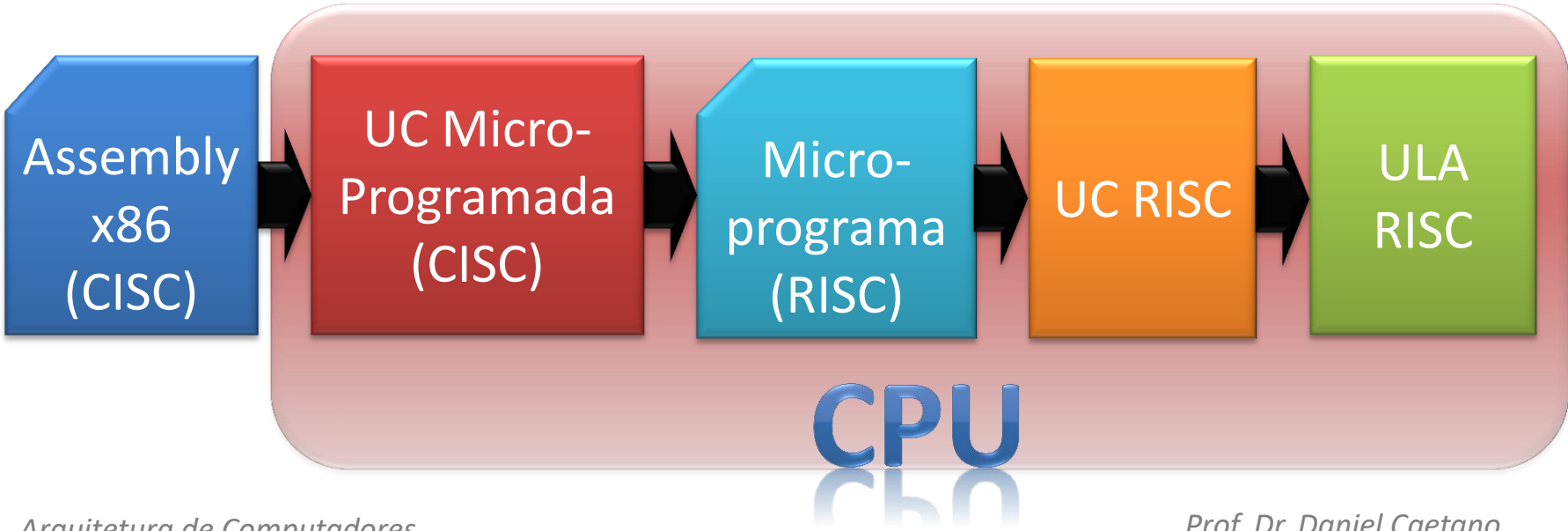
- Tomemos processadores de 3.57MHz
  - Período de aprox. 280ns
  - CPI do CISC: 10 ciclos por instrução
  - CPI do RISC: 1 ciclo por instrução
- Exemplo:
  - Código CISC com 1283 instruções
  - Código RISC equivalente com 10243 instruções
- Tcisc = 1.283 * 10 * 280ns = 3.592.400ns
- Trisc = 10.243 * 1 * 280ns = 2.868.040ns

CISC x RISC

- Tcisc = 1.283 * 10 * 280ns = 3.592.400ns
- Trisc = 10.243 * 1 * 280ns = 2.868.040ns
- Speedup
  - S = 100 * (Tcisc – Trisc) / Trisc
- S = 25,3%
- “Pouco”!
- Mas...
  - CPU RISC é menor, gasta menos e é mais barata
  - É possível acelerá-la mais, facilmente!

CISC x RISC

- RISC é mais eficiente: consenso
- 8086/88 eram CISC...
- Mudar para RISC e permanecer compatível?
  - Sim! Controle Microprogramado!



MICROPROGRAMAS

Microprogramas

- Cada estágio de instrução parece simples
  - Busca de Instrução
  - Decodificação de Instrução
  - ...
- Mas será que isso é fácil de implementar usando circuitos lógicos?
- Não, é muito difícil!
- Para solucionar esse problema...
  - **IBM criou a microprogramação**
  - Série 360 (década de 1960)

Microprogramas

- IBM percebeu o seguinte...
- Busca de instrução pode ser descrita como
  1. Configurar MAR com valor do PC
  2. Configurar bar. de controle para leitura de memória
  3. Aguardar intervalo para resposta da RAM
  4. Ler o valor do MBR para o IR
  5. Remover sinais do MAR e barramento de controle
- Isso parece um pequeno programa, não?
- Tarefas simples com circuitos lógicos

Microprogramas

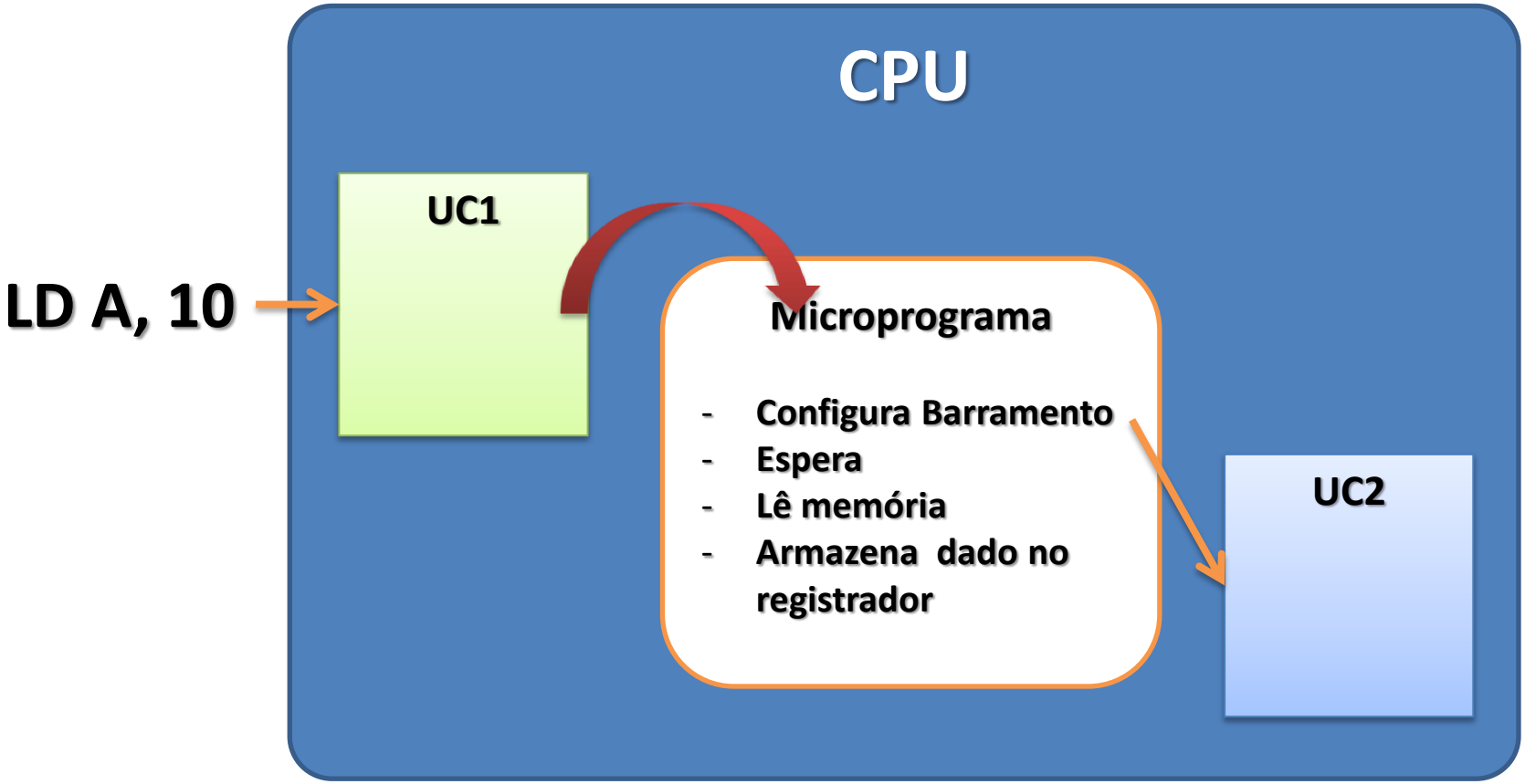
- Cada Instrução: pequeno programa
  - **firmware**

- Result. da Instrução

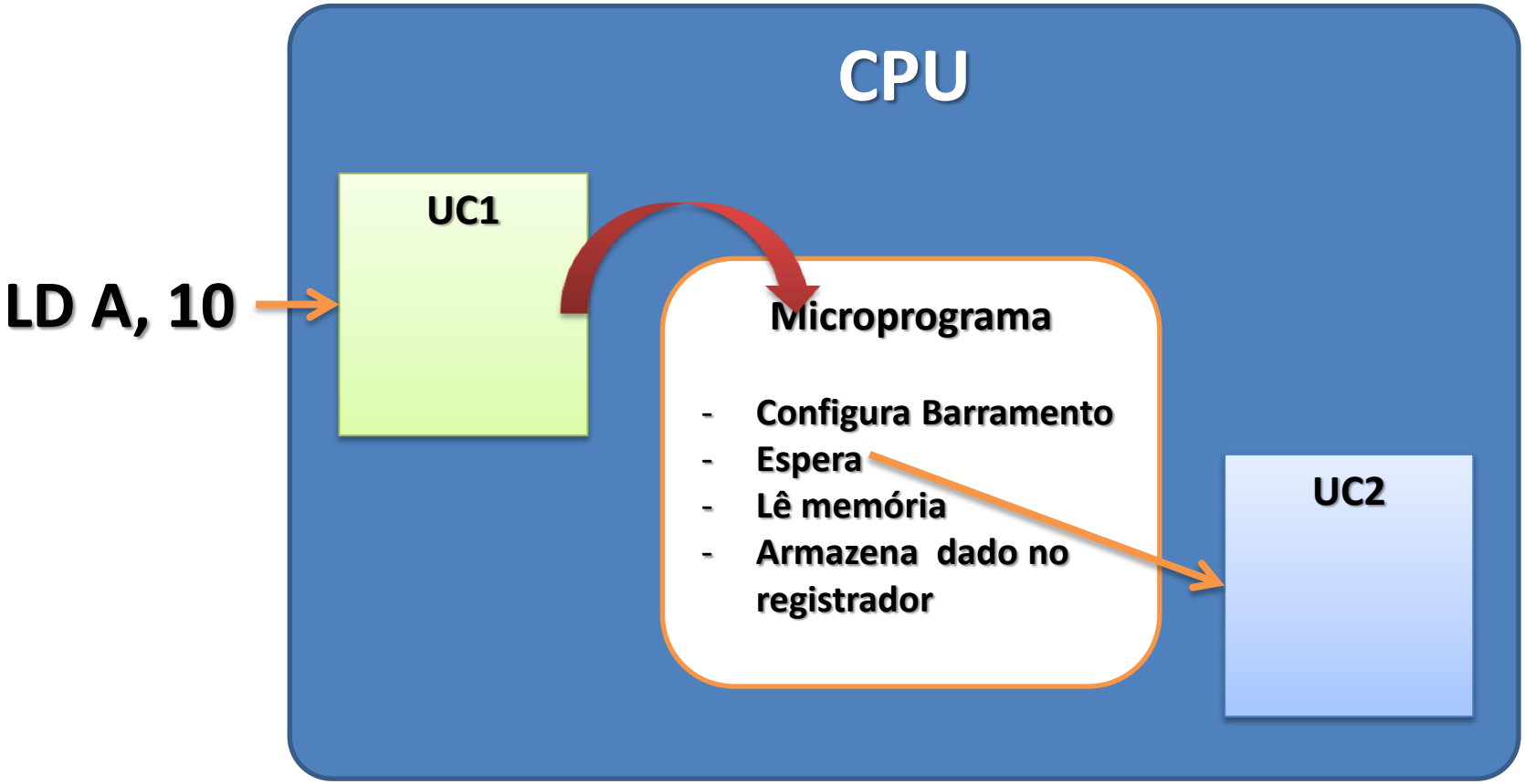
  =

  Result. do Microprograma

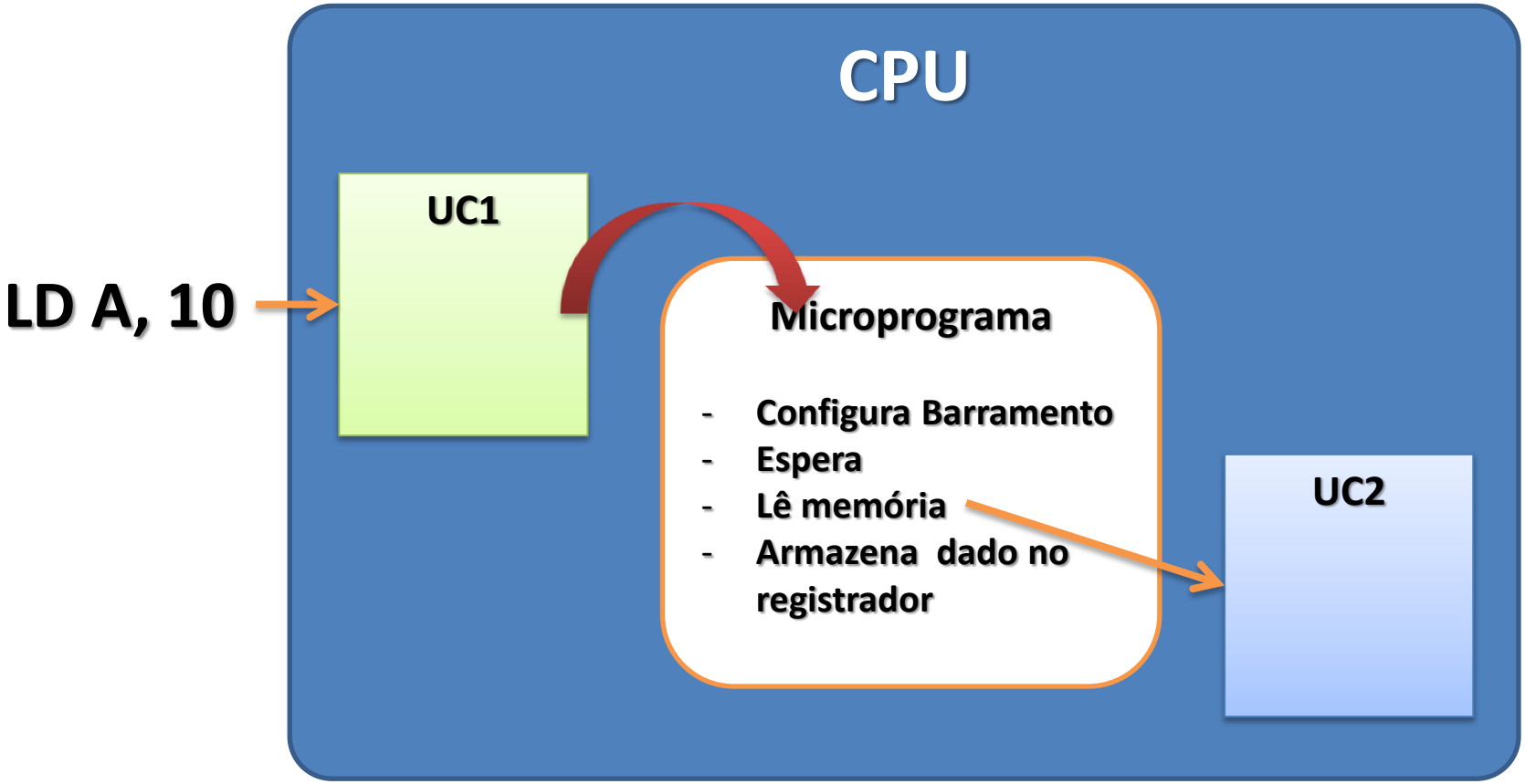
Microprogramas



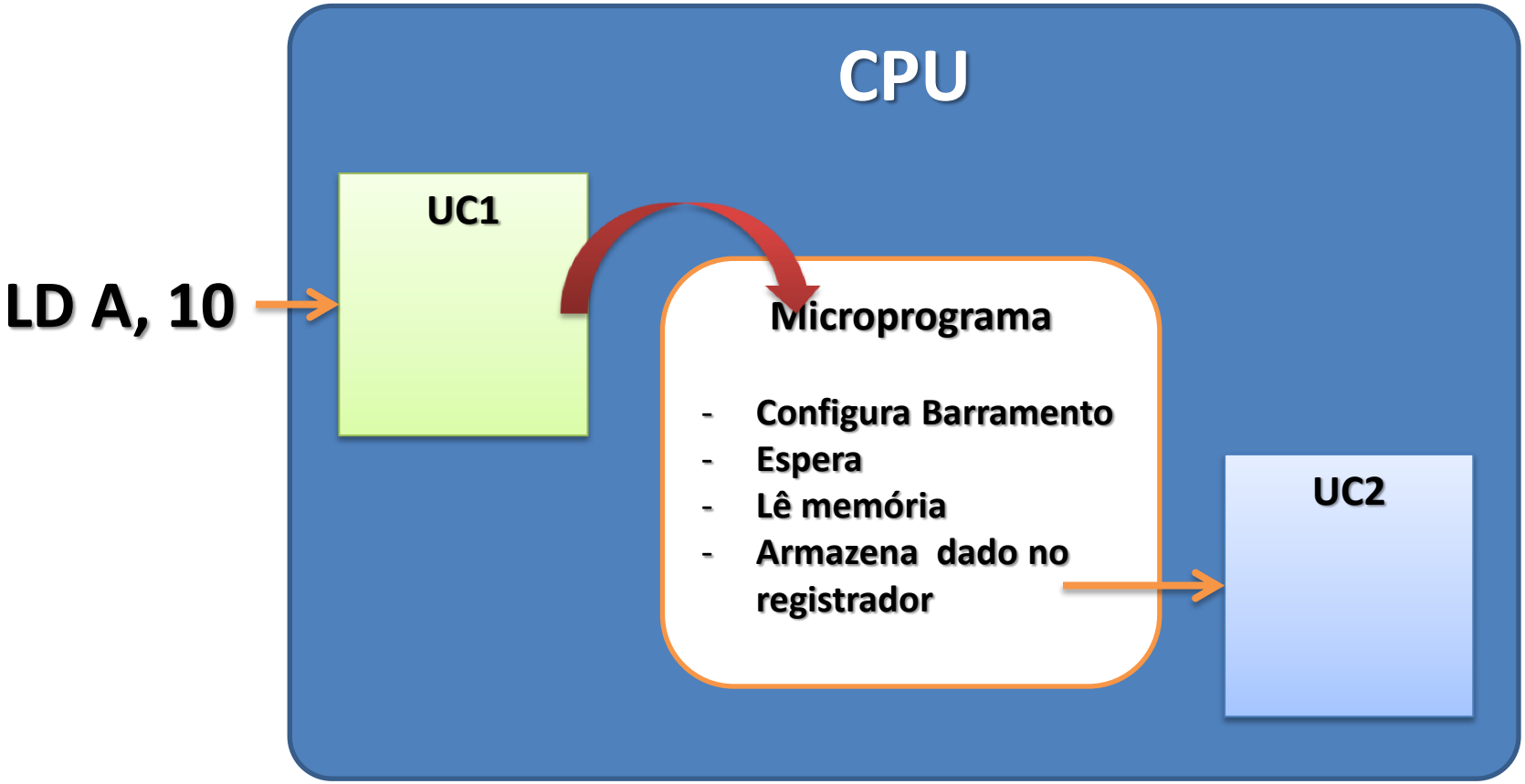
Microprogramas



Microprogramas



Microprogramas



ENCERRAMENTO

Resumo e Próximos Passos

- Arquitetura CISC: dominante inicialmente
- Melhoria das memórias + Compiladores
  - Vantagens da CISC se perderam
- Como RISC facilita a melhoria das CPUs...
  - ...ela é a base de todas as CPUs modernas
- **Pós Aula**: Conteúdo digital AURA!

- Prepare-se!
  - Avaliações finais!

PERGUNTAS?