



ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

ARQUITETURAS RISC E CISC

Prof. Dr. Daniel Caetano

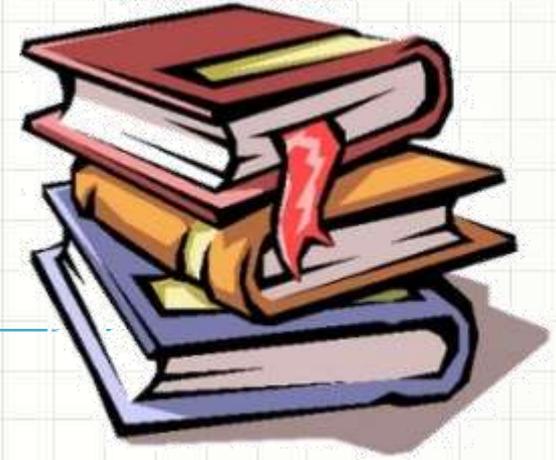
2012 - 2

Objetivos

- Conhecer as premissas da Arquitetura CISC
- Conhecer as premissas da Arquitetura RISC
- Compreender as diferenças entre RISC e CISC
- **Lembretes:**
 - AV2!



Material de Estudo



Material

Acesso ao Material

Notas de Aula

<http://www.caetano.eng.br/>
(Aula 14)

Apresentação

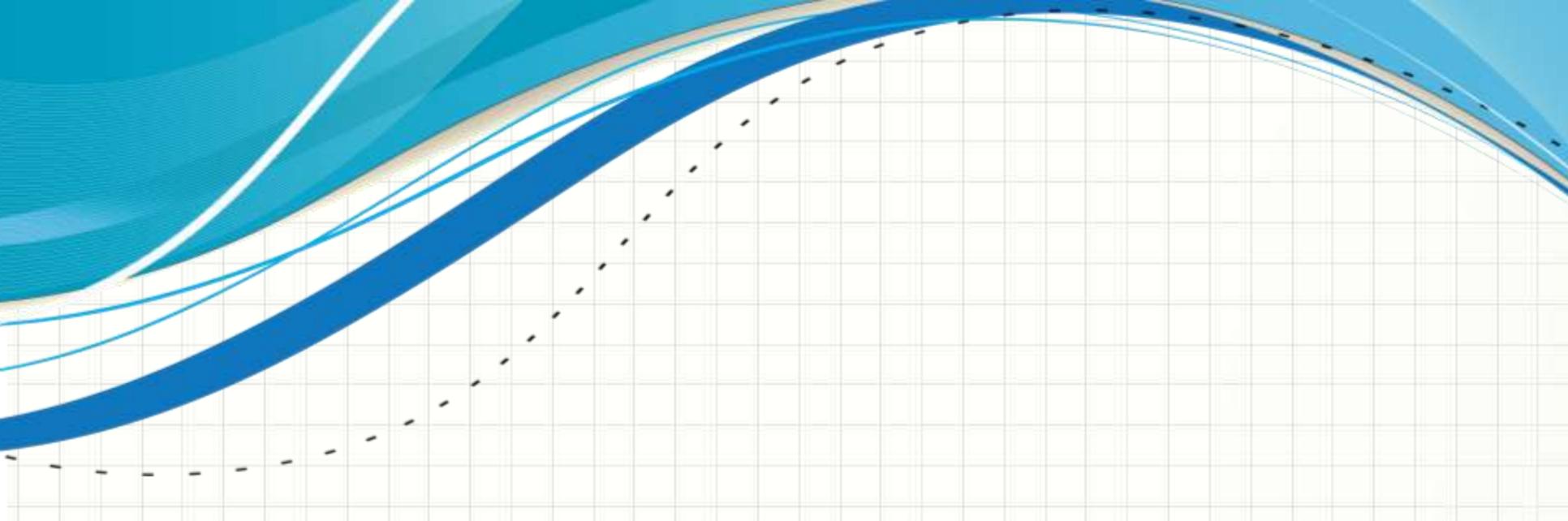
<http://www.caetano.eng.br/>
(Aula 14)

Material Didático

-

Biblioteca Virtual

Arquitetura e Organização de Computadores, páginas
477 a 524



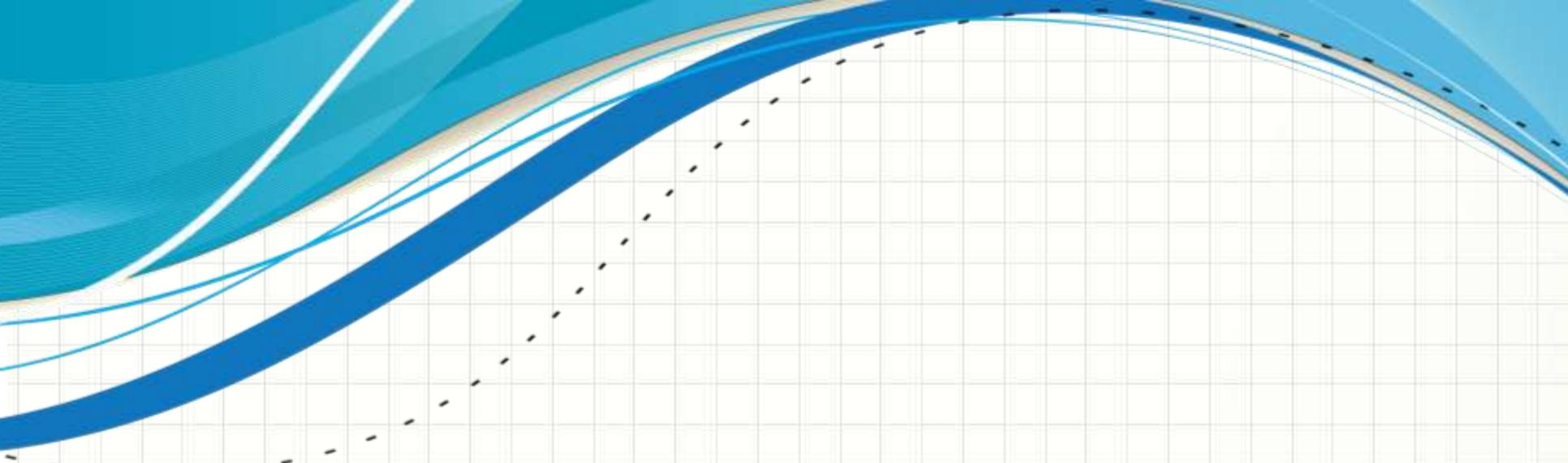
INTRODUÇÃO

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 perdeu
- 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
```

Arco

Funciona, mas...

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

- Cópia
- a p

000 para
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
```

Arco

Que tal uma instrução que faz cópia?

- Cópia
- a p

000 para
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
```


Arco

Uma única instrução

- Cópia
a p

2 bytes de instrução
Redução de 80%!

1000 para
complexa

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

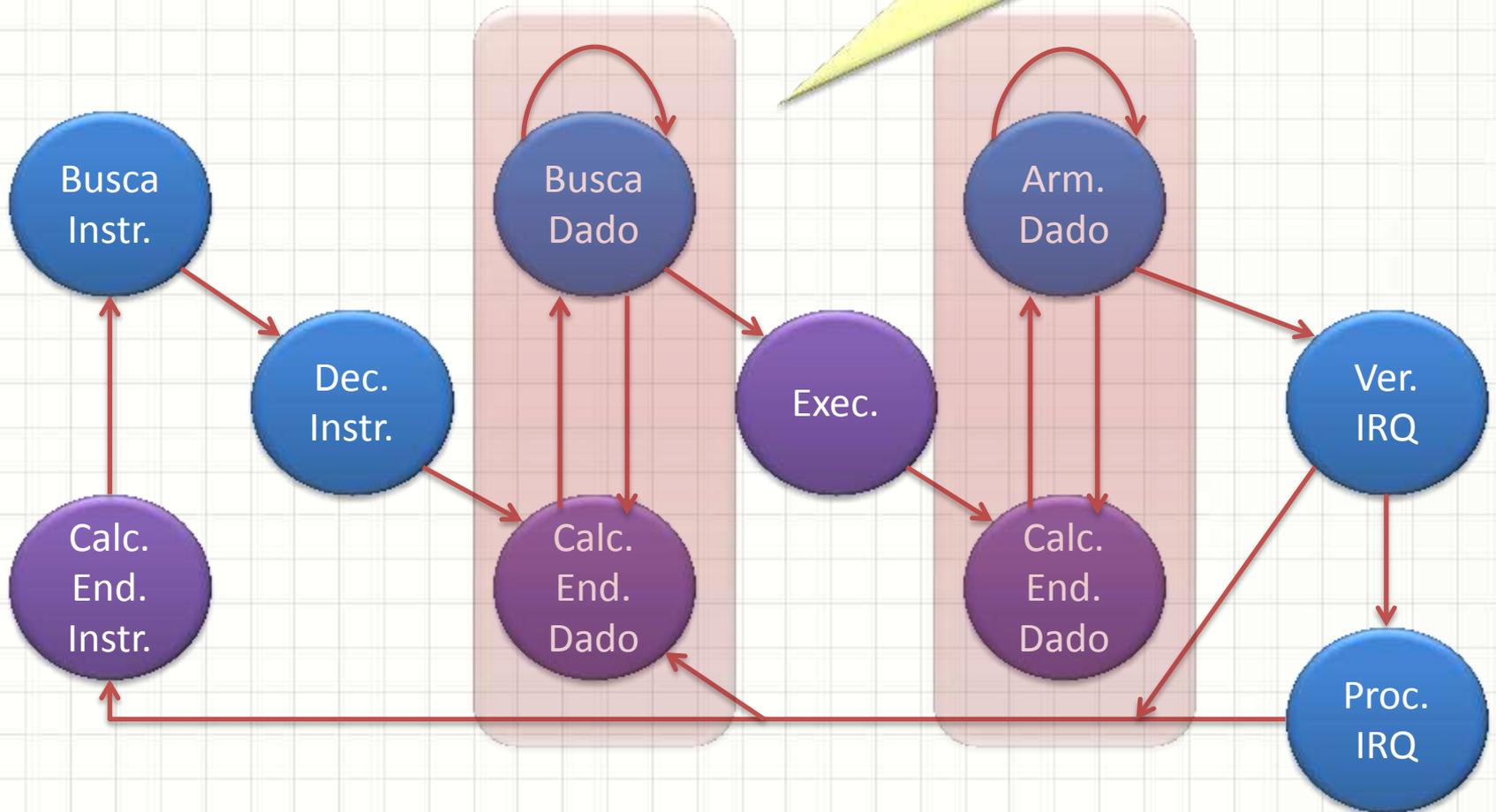
Arquitetura CISC

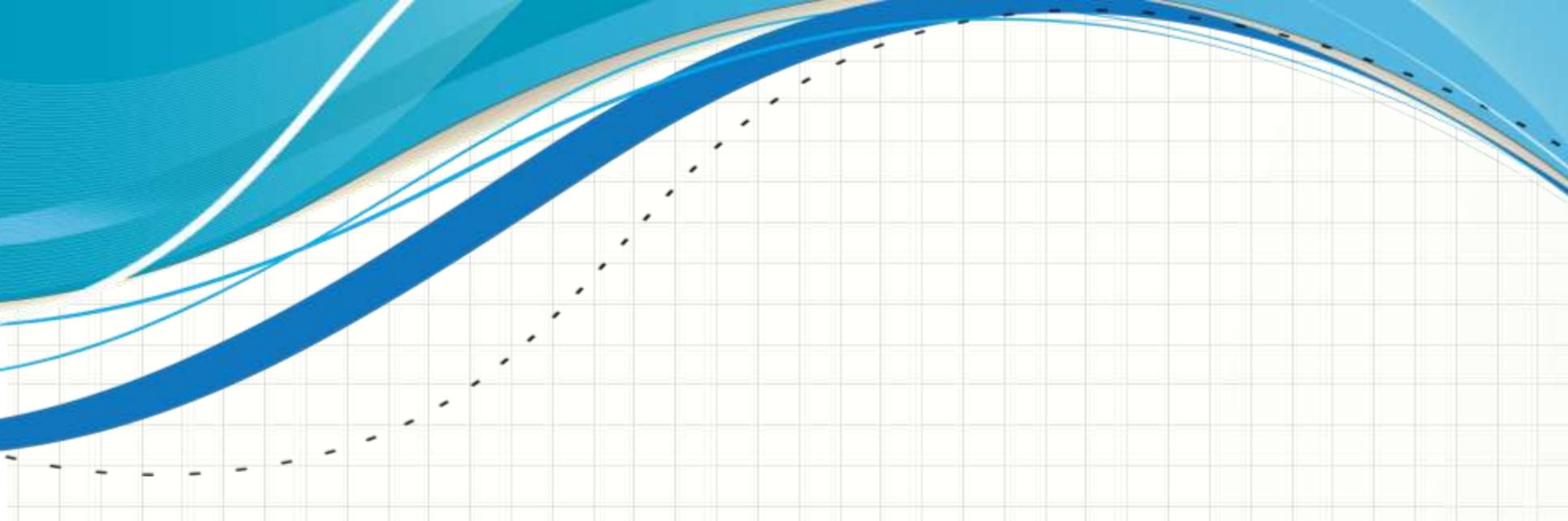
- Mas essas instruções complexas são comuns?
 1. **Atribuição:** 47%
 2. **Comparação/If:** 23%
 3. **Chamadas de Função:** 15%
 4. **Loops:** 6%
 5. **Saltos Simples:** 3%
 6. **Outras Operações:** 7%
- Ganhos com operações complexas...
 - Limitados!

Arquitetura CISC

- Ciclo de Instrução CISC

Difícil implementar no pipeline!





ARQUITETURA RISC

Arquitetura RISC

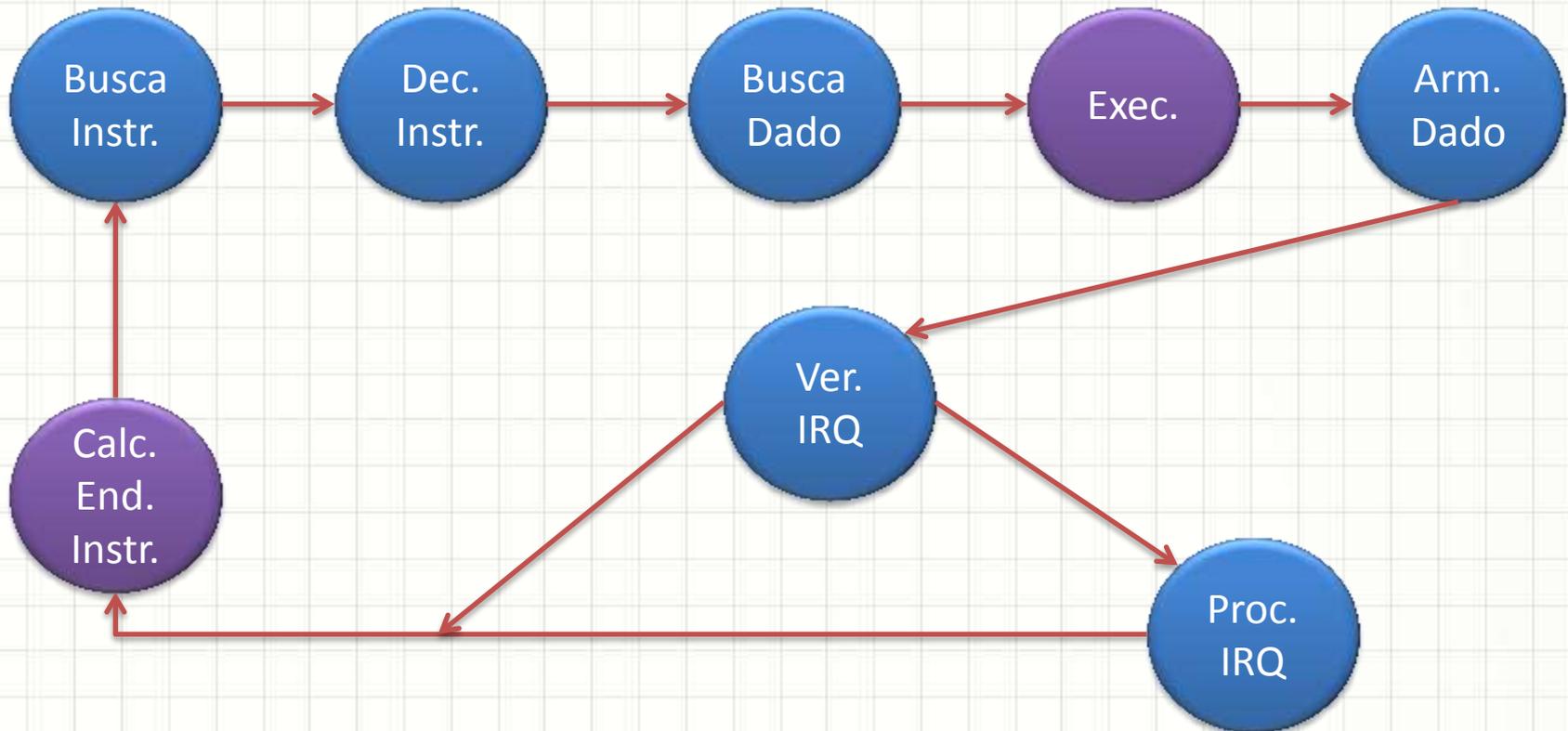
- Aumento do desempenho das memórias
 - Otimizar instruções mais comuns
 - Atribuições, comparações, saltos...
- CPUs mais simples e mais rápidas
 - Acesso à memória: apenas **load** e **store**
 - Reduz a interdependência das instruções
 - Facilita superpipeline

Arquitetura RISC

- 5 Características RISC marcantes
 - **Instruções de tamanho fixo**
 - Uma 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

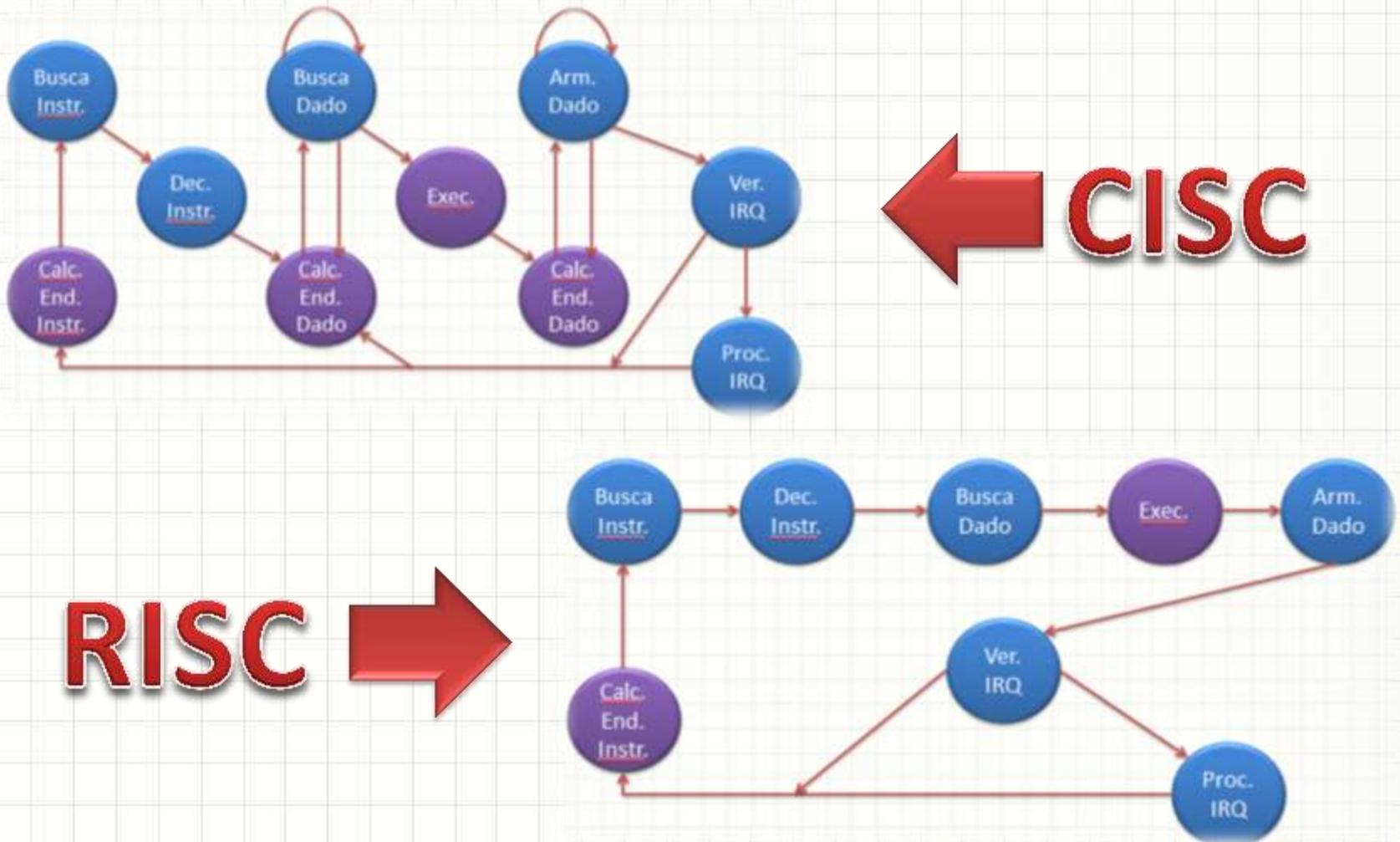




DESEMPENHO

CISC x RISC

- Complexidade



CISC x RISC

- Cálculo de Speedup

$$S = 100 * \frac{(T_s - T_c)}{T_c}$$

- **S**: Speedup
- **T_s**: Tempo Sem otimização (CISC)
- **T_c**: Tempo Com otimização (RISC)

CISC x RISC

- Cálculo dos Tempos de Processamento

$$T = NI * CPI * P$$

- T: Tempo de Processamento
- NI: Número de Instruções
- CPI: Ciclos de clock Por Instrução
- P: Período (tempo de cada ciclo)

$$P = 1 / \text{frequencia}$$

CISC x RISC

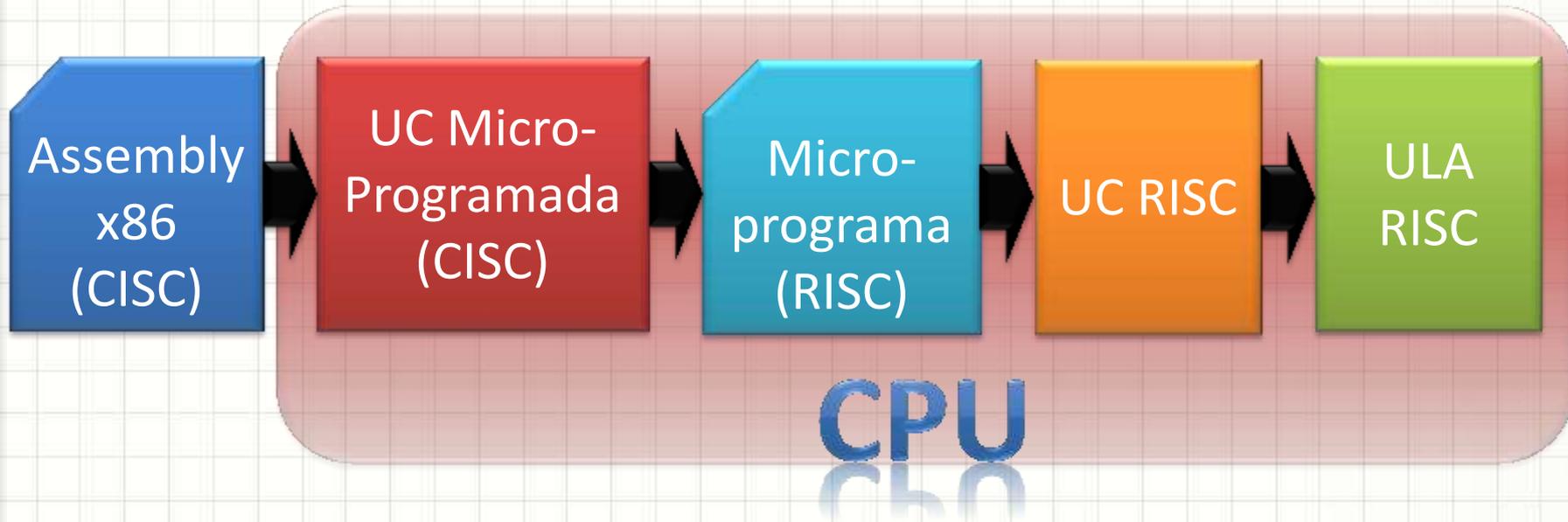
- Tomemos processadores de 3.57MHz
 - Período de aprox. 280ns
 - CPI do CISC: 10 ciclos por instrução
 - CPI do RISC: 1 ciclo por instrução
- Exemplo:
 - Código CISC com 1283 instruções
 - Código RISC equivalente com 10243 instruções
- $T_{cisc} = 1.283 * 10 * 280ns = 3.592.400ns$
- $T_{risc} = 10.243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

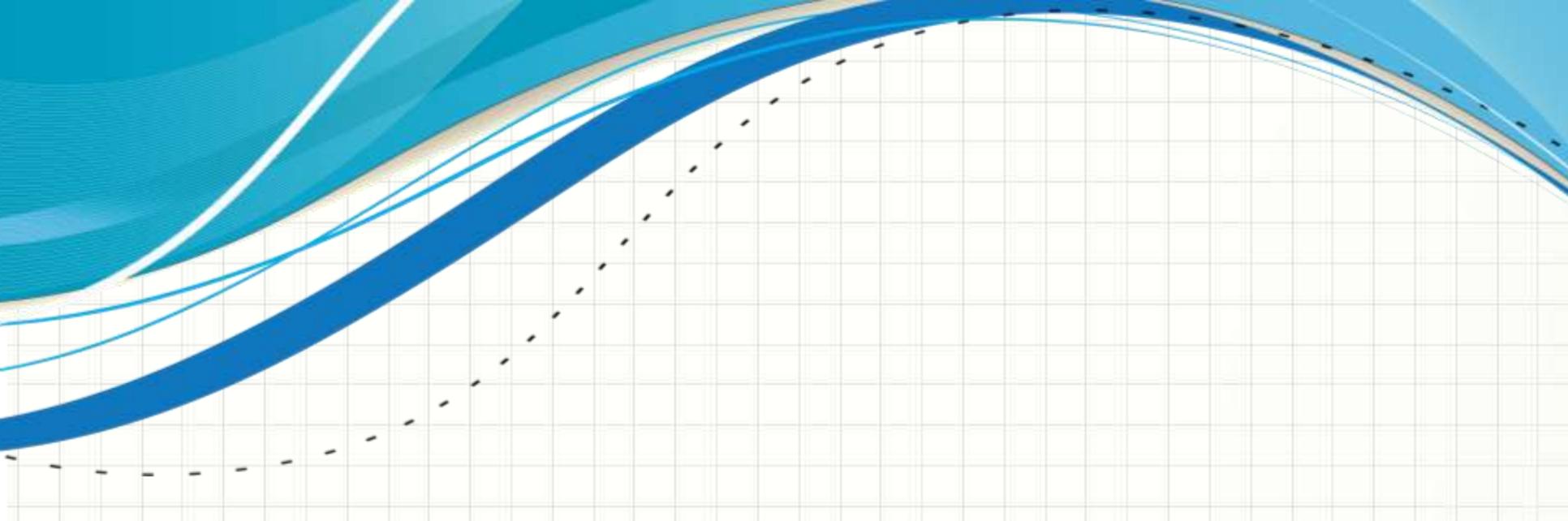
CISC x RISC

- $T_{cisc} = 1.283 * 10 * 280ns = 3.592.400ns$
- $T_{risc} = 10.243 * 1 * 280ns = 2.868.040ns$
- Speedup
 - $S = 100 * (T_{cisc} - T_{risc}) / T_{risc}$
- $S = 25,3\%$
- “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!





CONCLUSÕES

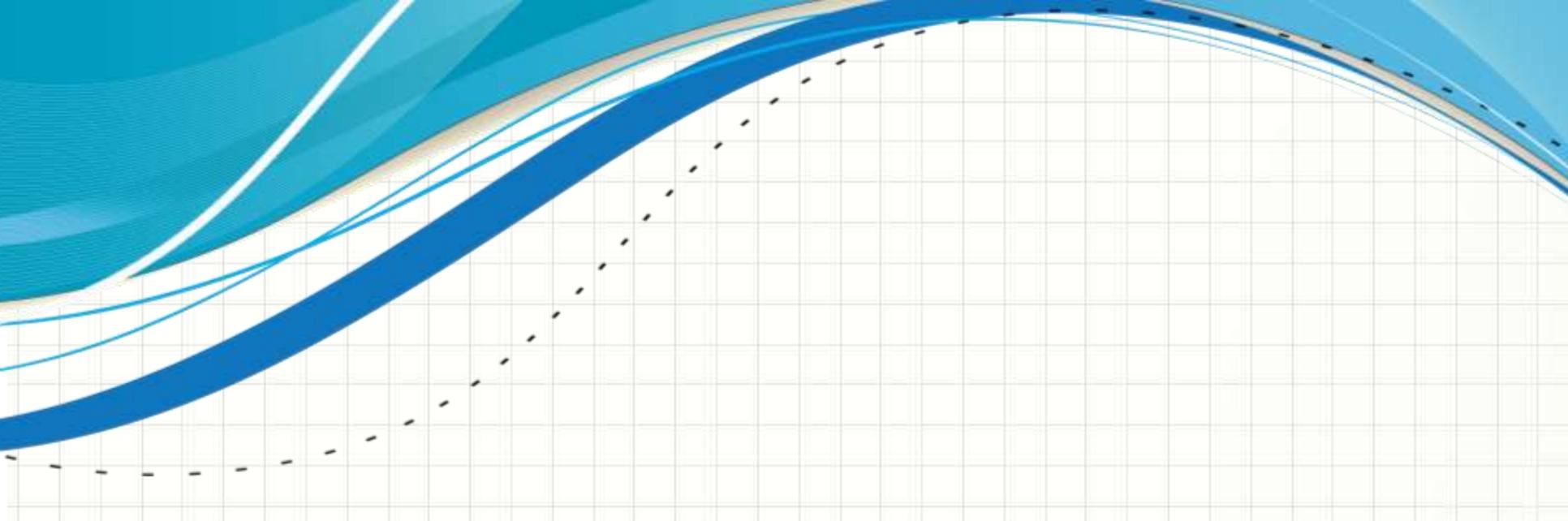
Resumo

- 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
- **TAREFA**
 - AV2

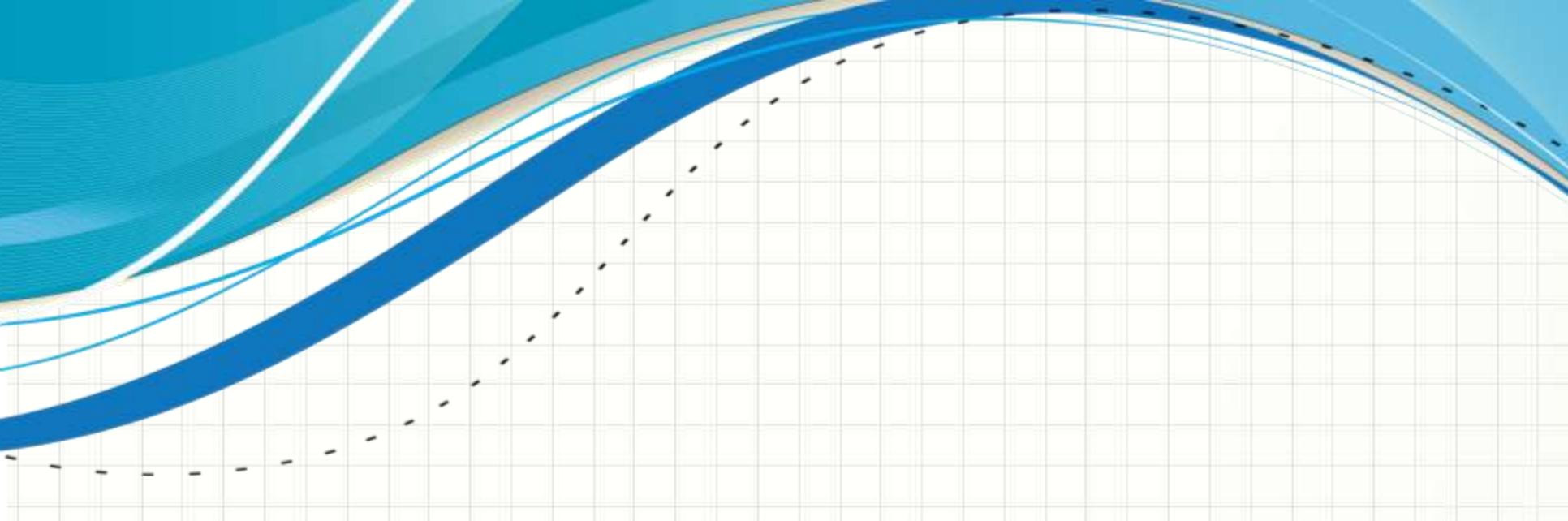
Próxima Aula



THE END!



PERGUNTAS?



**BOM DESCANSO
A TODOS!**