Arquitetura e Organização de Sistemas Computadorizados - Processador

Osmar de Oliveira Braz Junior

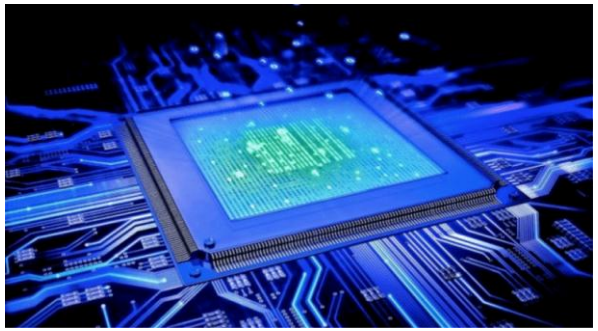
Márcia Cargnin Martins Giraldi

Objetivos

- Apresentar os conceitos e tipos de processadores de computadores

Processador

- Peça fundamental dos computadores
- Responsável direto pela movimentação e manipulação de dados
- Componente mais complexo e mais importante
- Circuito Integrado que realiza as funções de cálculo e tomada de decisão do computador
- Frequência de processamento, L1-Cache, Core



Arquitetura do Processador

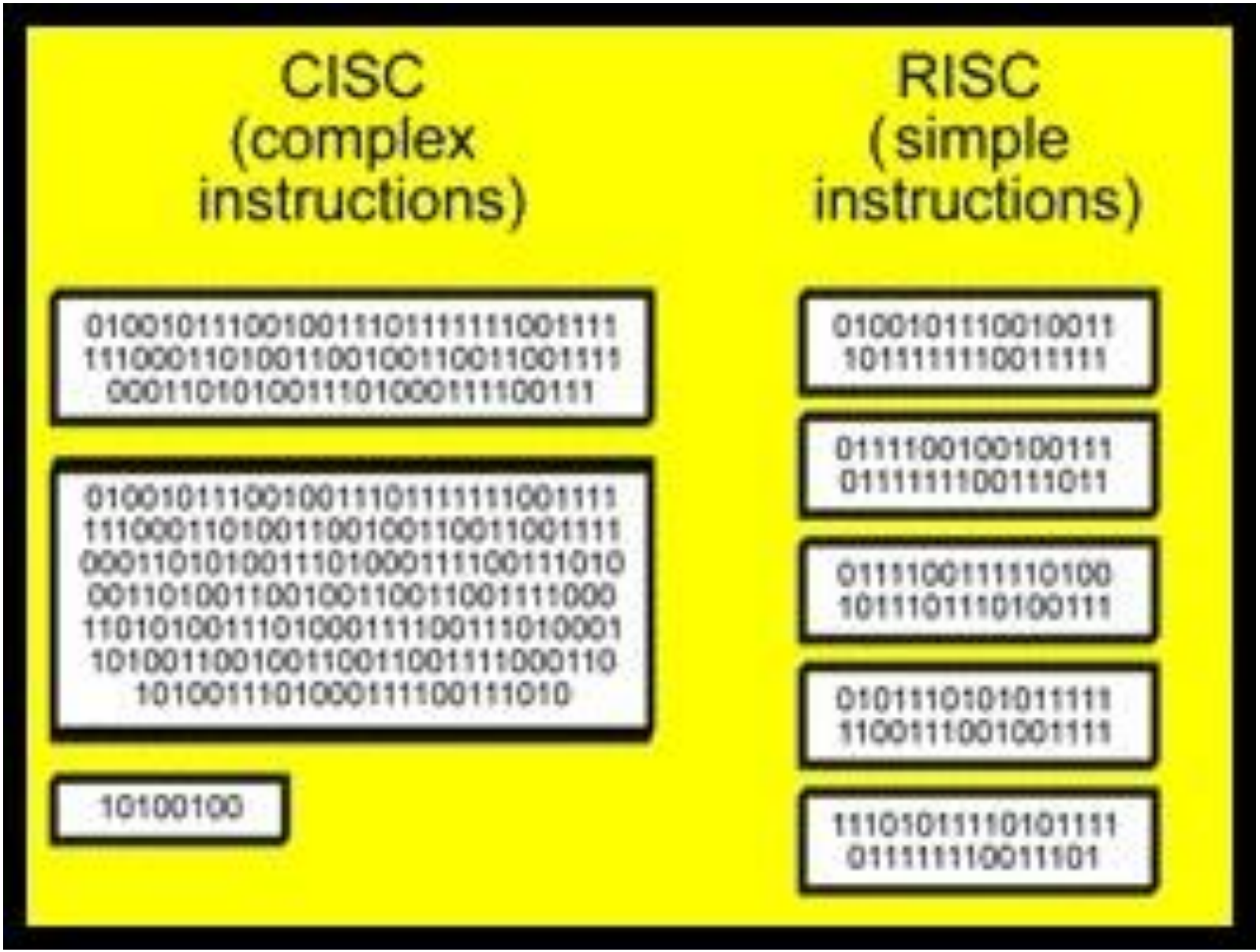
- Descreve o processador que está sendo usado no computador
- Grande parte dos computadores vem com identificação descrevendo o processador que contém dentro de si, arquitetura **RISC** ou **CISC.**

Arquitetura do Processador

- CISC (*Complex Instruction Set Computing*) – computador com um conjunto complexo de instruções: usado em processadores Intel e AMD; suporta mais instruções porém mais lenta a execução delas

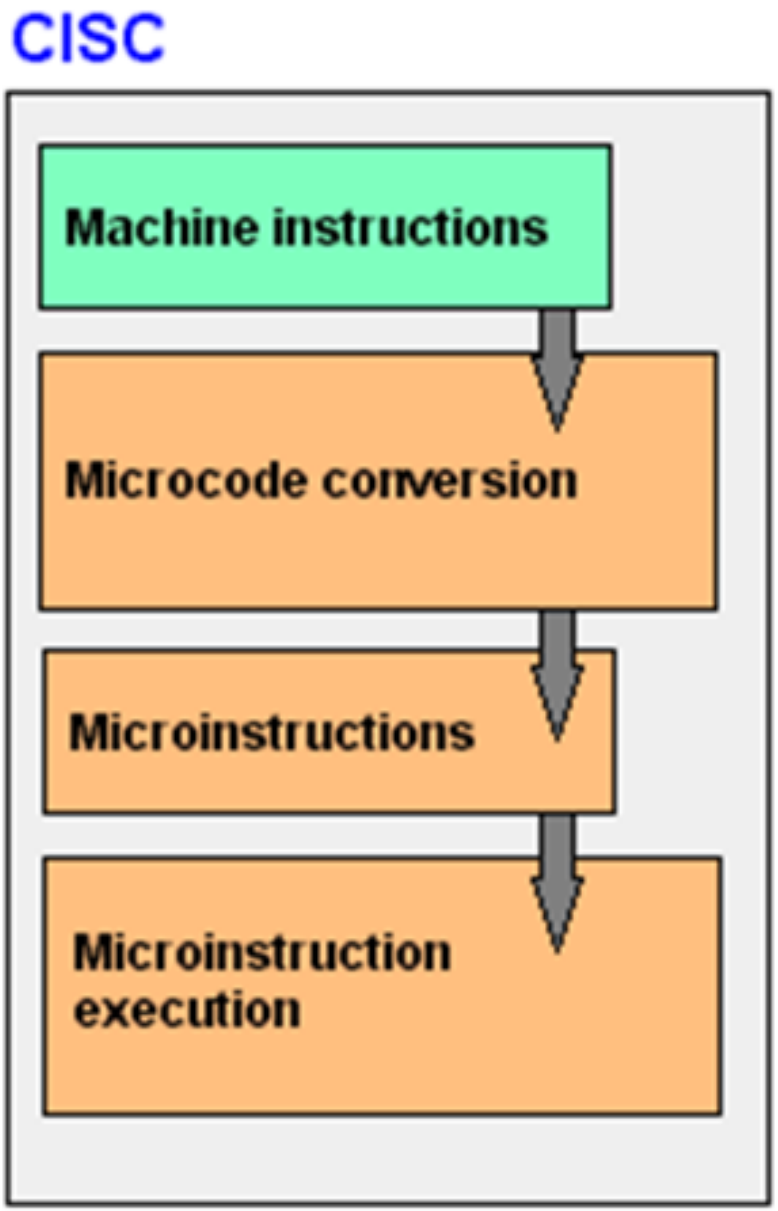
Arquitetura do Processador

- RISC (*Reduced Instruction Set Computing*) – usada em processadores PowerPC (da Apple, Motorola e IBM) e SPARC (SUN). Suporta menos instruções executando com maior rapidez o conjunto de instruções que são combinadas.


CISC
(complex
instructions)
RISC
(simple
instructions)
01001011100100111011111111001111
11100011010011001001100110011111
00011010100111010001111001110100
01001011100100111011111111001111
11100011010011001001100110011111
00011010100111010001111001110100
00110100110010011001100111100011
01001011100100111011111111001111
11100011010011001001100110011111
00011010100111010001111001110100
00110100110010011001100111100011
11010100111010001111001110100011
01001011100100111011111111001111
11100011010011001001100110011111
00011010100111010001111001110100
00110100110010011001100111100011
11010100111010001111001110100011
10100100
01001011100100111
01111001001001111
01111001001001111
01111111001110111
01111001111110100
10111011101001111
01011101010111111
11001110010011111
11101011110101111
01111111001110111

CISC

- Este processadores contêm uma microprogramação, ou seja, um conjunto de códigos de instruções que são gravados no processador
- permite receber as instruções dos programas e executá-las, utilizando as instruções contidas na sua microprogramação
- reduz o tamanho do código executável por já possuírem muito do código comum em vários programas, em forma de uma única instrução



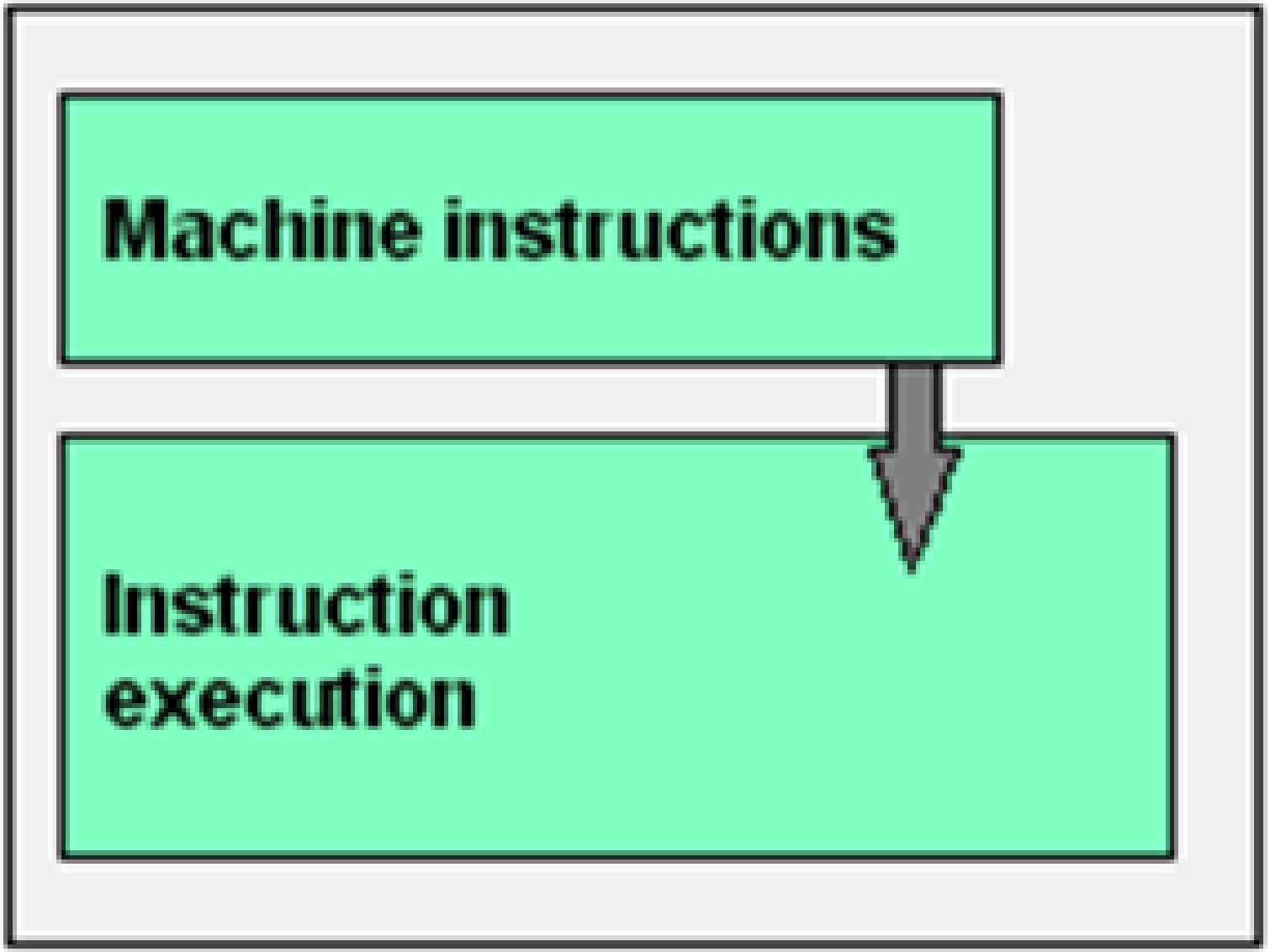
CISC

- Operação com o “a = a + b” descrita como “add a,b”
- podem simplesmente utilizar dois operandos para uma única instrução, sendo um deles fonte e destino (acumulador) e permite um ou mais operadores em memória para a realização das instruções

RISC

- No começo da década de 80, a tendência era construir chips com conjuntos de instruções cada vez mais complexos. Alguns fabricantes porém, resolveram seguir o caminho oposto, criando o padrão RISC
- Suporta menos instruções, e com isso executa com mais rapidez o conjunto de instruções que são combinadas
- Os chips são mais simples e muito mais baratos por terem um menor número de circuitos internos, podem trabalhar a frequências mais altas

RISC



RISC

- Operação com o “**a = b + c**” descrita como “**add a,b,c**”
- Podem especificar três operandos para uma única instrução, mas exclusivamente se estes forem registradores.
- Cada célula de memória contém apenas 8 bits, um inteiro (32 bits) ocuparia mais de uma célula

RISC	CISC
Conjunto de instruções semanticamente simples de tamanho fixo reduzido	Conjunto de instruções semanticamente complexas de tamanho variável extenso
Decodificação simplificada (tabela)	Decodificação complexa (microcódigo)
Execução regular, interpretada pelo próprio programa	Cada instrução executa à sua maneira, pelo hardware
Instruções requerem o mesmo número de ciclos de clock para executar	Grande variação no número de ciclos de clock por instrução
Possibilita o uso de pipeline (intenso)	Extremamente difícil/impossível o uso de pipeline
Apenas algumas operações (load/store) em memória	Qualquer instrução pode referenciar a memória
Poucas instruções e modos de endereçamento	Várias instruções e modos de endereçamento
Múltiplos conjuntos de registradores	Conjunto de registradores únicos (operadores aritméticos, apontadores para a memória e memória segmentada)
não possuem instruções para multiplicação ou divisão	possuem instruções para multiplicação ou divisão

RISC x CISC

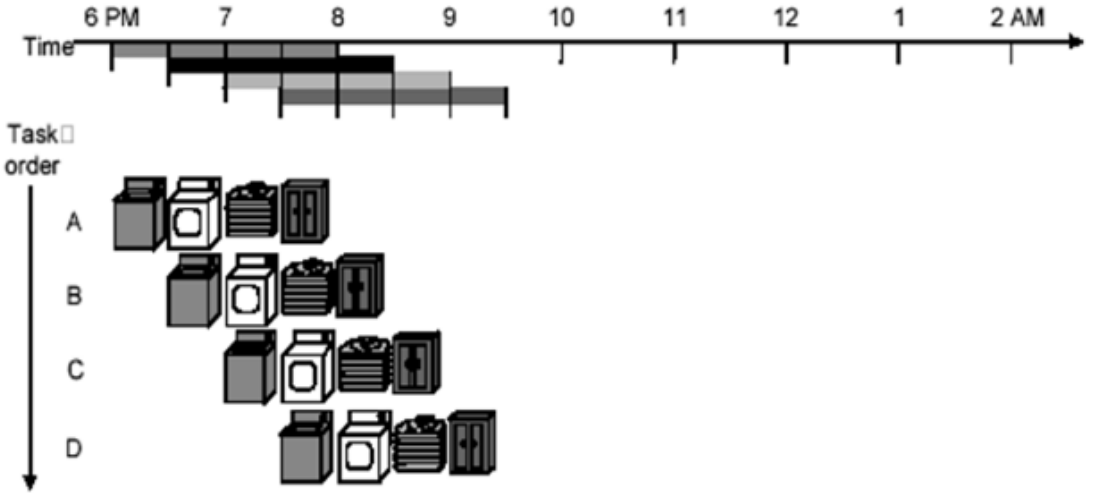
- Atualmente vemos processadores híbridos, que são essencialmente processadores CISC, mas incorporam muitos recursos encontrados nos processadores RISC (ou vice-versa);
- Apesar de por questões de Marketing, muitos fabricantes ainda venderem seus chips, como sendo "Processadores RISC", não existe praticamente nenhum processador atualmente que siga estritamente uma das duas filosofias

Pipeline

- técnica de hardware que permite que a CPU realize a busca de uma ou mais instruções além da próxima a ser executada
- Estas instruções são colocadas em uma fila de memória dentro do processador (CPU) onde aguardam o momento de serem executadas: assim que uma instrução termina o primeiro estágio e parte para o segundo, a próxima instrução já ocupa o primeiro estágio
- Em resumo, é o processo pelo qual uma instrução de processamento é subdividido em etapas, uma vez que cada uma destas etapas é executada por uma porção especializada da CPU, podendo colocar mais de uma instrução em execução simultânea.

Exemplo sem Pipeline

Exemplo com Pipeline



Pipeline

	Estágio do pipeline onde a instrução se encontra							
Ciclo de Clock	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Instrução 1	BI	DI	EX	AM	GR			
Instrução 2		BI	DI	EX	AM	GR		
Instrução 3			BI	DI	EX	AM	GR	
Instrução 4				BI	DI	EX	AM	GR

Superpipeline – acima de 5 estágios

Níveis de Paralelismo

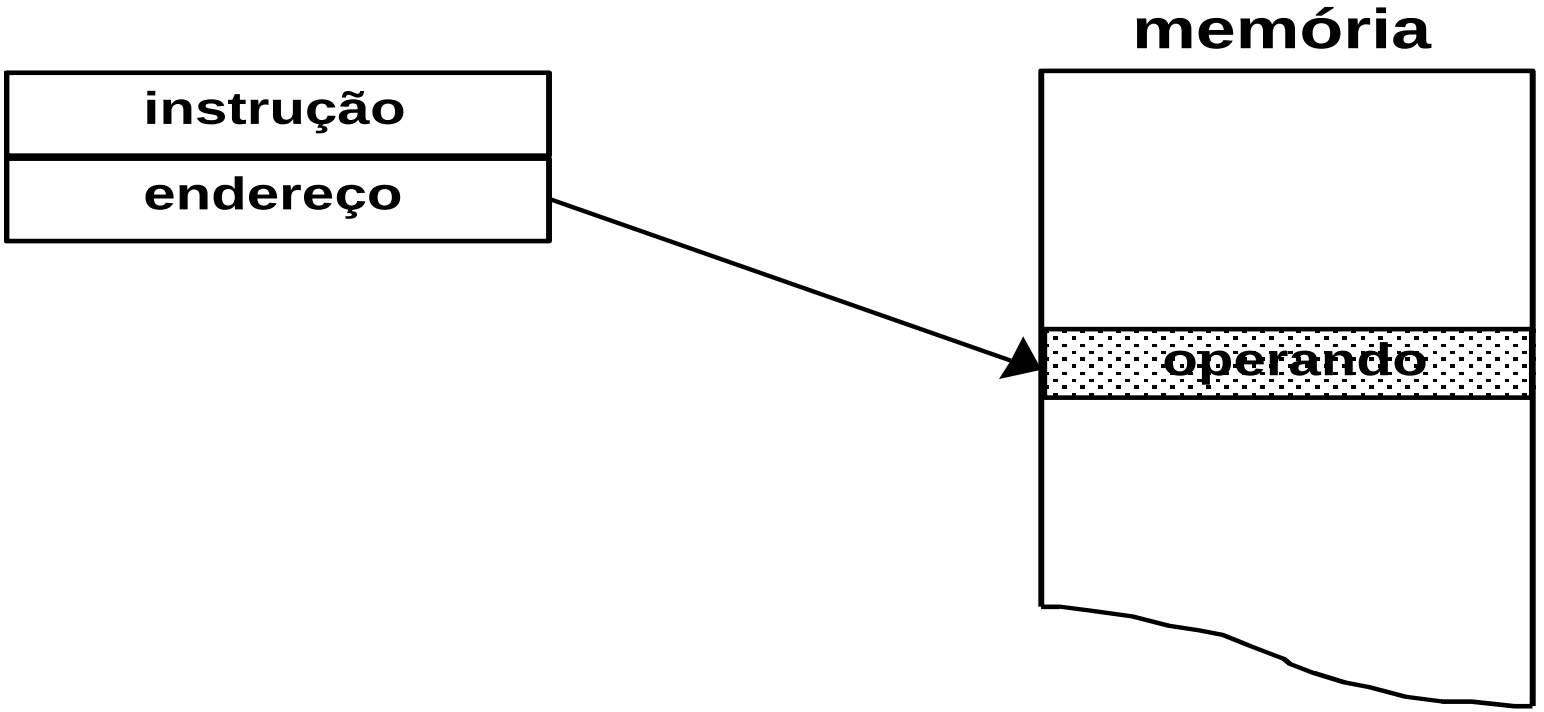
- INSTRUÇÃO/ *INSTRUCTION* (granulosidade fina)
  - Paralelismo entre as instruções
  - Arquiteturas Pipeline, Superescalar, VLIW
- TAREFAS/*THREADS* (granulosidade média)
  - Paralelismo entre as threads
  - Arquiteturas SMT (Simultaneous MultiThreading)
- PROCESSOS/*PROCESS* (granulosidade grossa)
  - Paralelismo entre os processos
  - Computação Paralela
  - Arquiteturas multiprocessadores e multicomputadores

Modos de endereçamento*

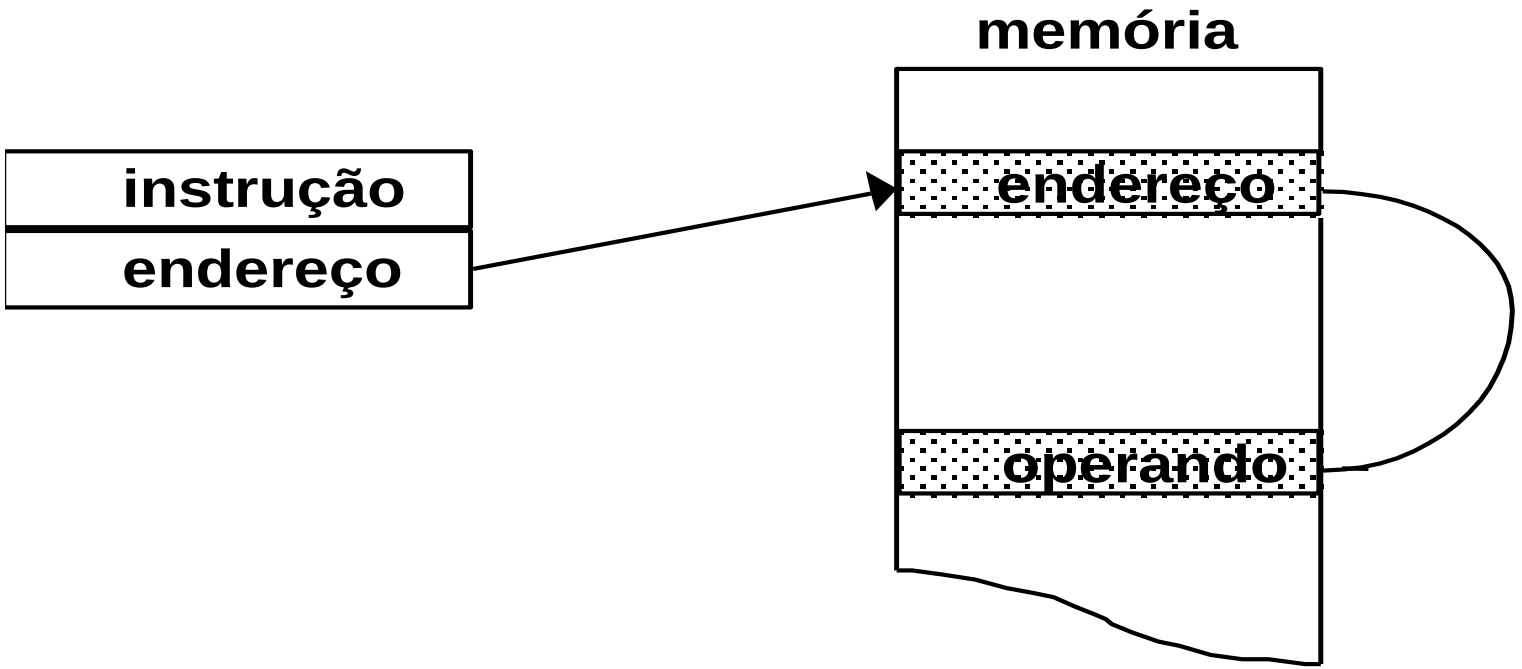
***Prof. Raul Weber, UFRGS**

(Mil e uma maneiras de especificar um operando)

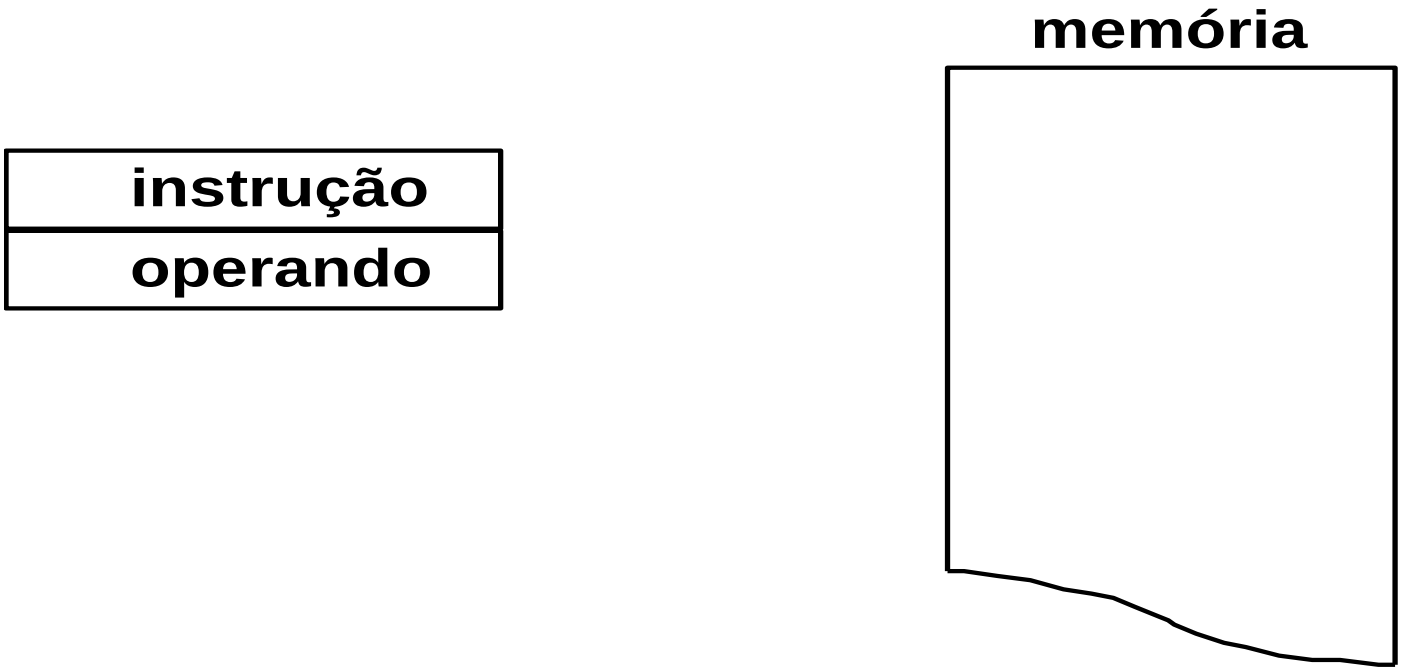
Endereçamento direto (absoluto)



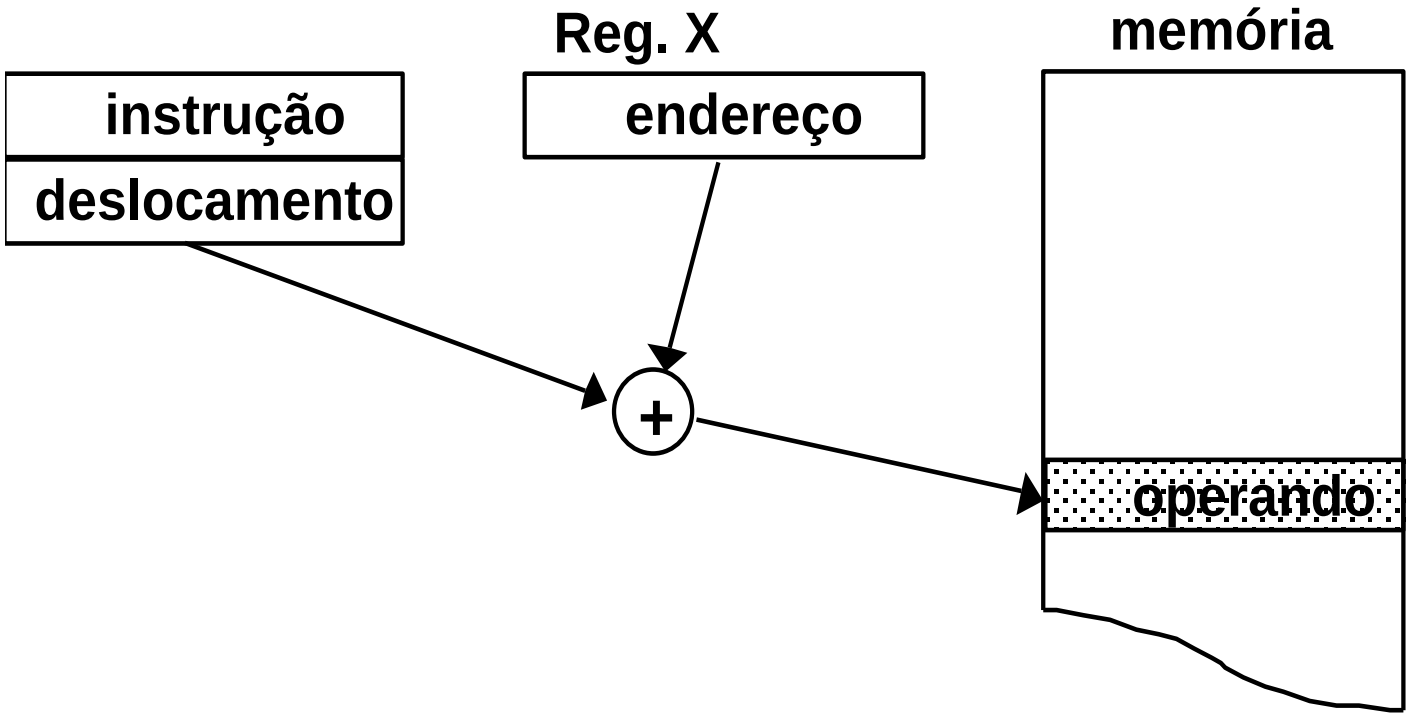
Endereçamento indireto



Endereçamento imediato



Endereçamento indexado



Modo direto

- Tratamento de variáveis
- Endereço conhecido (programador ou compilador)
- Variáveis escalares (uma palavra de memória)
- Exemplo:

  total := total + parcela

  ```
  LDR A total
  ADD A parcela
  STR A total
  ```

Modo imediato

- Tratamento de constantes
- Não necessita reservar posições de memória (de variáveis)
- Economiza um acesso à memória
- Exemplos:

  contador := contador + 1;

  total := 0;

```
LDR A contador
ADD A #1
STR A contador
```

```
LDR A #0
STR A total
```

Modo indireto

- Tratamento de endereços de variáveis
- Necessita de um acesso extra à memória
- Exemplos:
  variavel := ponteiro^;

LDR A ponteiro,I A ← mem(mem(ponteiro))
STR A variavel mem(variável) ← A

Modo indexado

- Tratamento de vetores (arrays, matrizes, etc)
- Endereço do operando calculado através da soma de dois valores
- Um valor codificado na instrução
- Outro valor armazenado no registrador X
- Significado de cada valor depende do programa

NEANDER x RAMSES*

*Prof. Raul Weber, UFRGS

(Ou porque da necessidade de upgrade :-)

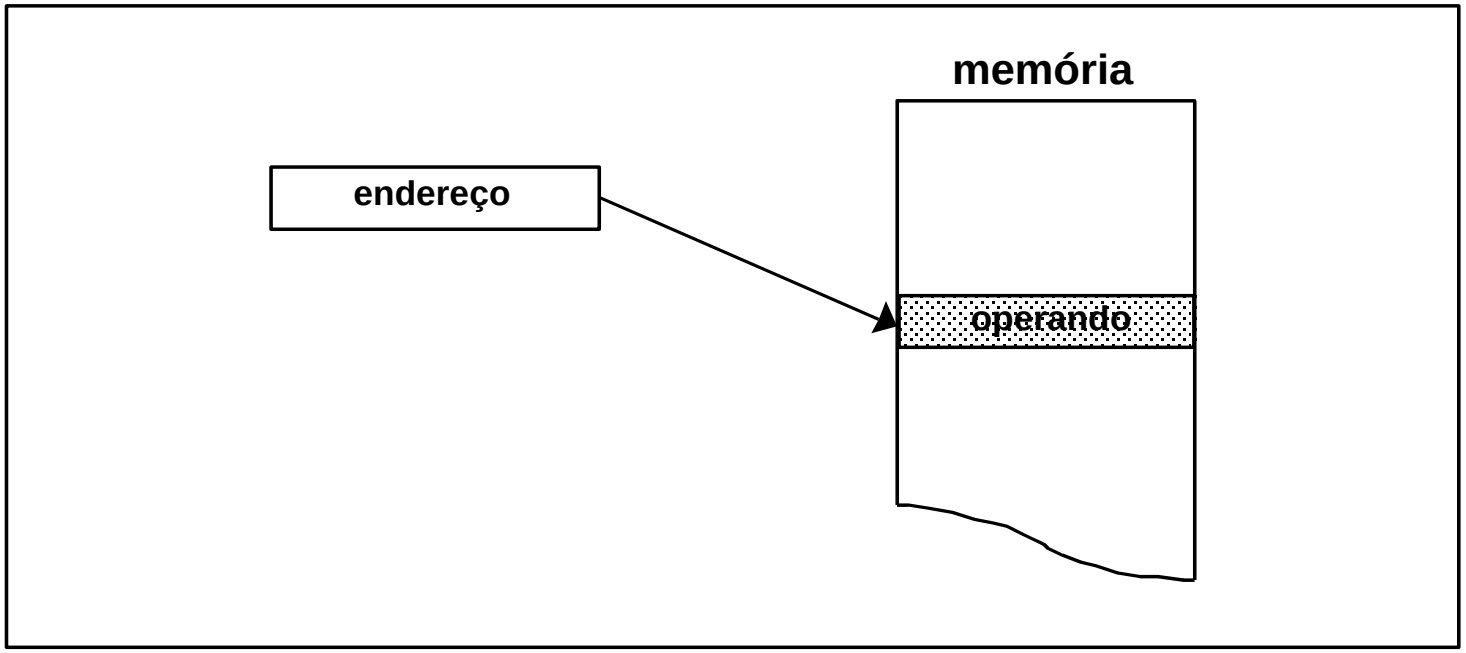
Neander - características

- Largura de dados e endereços de 8 bits
- Dados representados em complemento de dois
- 1 acumulador de 8 bits (AC)
- 1 apontador de programa de 8 bits (PC)
- 1 registrador de estado com 2 códigos de condição: negativo (N) e zero (Z)

Instruções do Neander

Código	Instrução	Comentário
0000	NOP	nenhuma operação
0001	STA end	armazena acumulador - (store)
0010	LDA end	carrega acumulador - (load)
0011	ADD end	soma
0100	OR end	“ou” lógico
0101	AND end	“e” lógico
0110	NOT	inverte (complementa) acumulador
1000	JMP end	desvio incondicional - (jump)
1001	JN end	desvio condicional - (jump on negative)
1010	JZ end	desvio condicional - (jump on zero)
1111	HLT	término de execução - (halt)

Modo de endereçamento



Programa Exemplo

Somar (totalizar) *n* posições consecutivas de memória, a partir do endereço inicial *e*. (Sem consistência sobre os valores de *n* e *e*).

Em alto nível, o programa seria:

```
        total:=0
        ponteiro := e
        contador := n
laço:   if contador = 0, termina
        total := total + mem(ponteiro)
        ponteiro := ponteiro + 1
        contador := contador – 1
        goto laço
```

Ramses - características

- quatro modos de endereçamento,
- dois registradores de uso geral,
- um registrador de índice,
- indicadores de carry, negativo e zero,
- instruções adicionais (chamada de subrotina, negação e deslocamento de bits, etc ...).

Instruções do Ramses

Código	***Instrução***	***Operação Executada***
0000	NOP	nenhuma operação
0001	STR r end	armazena registrador na memória - (store)
0010	LDR r end	carrega registrador da memória - (load)
0011	ADD r end	adição - soma memória ao registrador
0100	OR r end	"ou" (adição lógica) - resultado no registrador
0101	AND r end	"e" (multiplicação lógica) - resultado no registrador
0110	NOT r	inverte (complementa para 1) registrador
0111	SUB r end	subtração - subtrai memória do registrador
1000	JMP end	desvio incondicional - (jump)
1001	JN end	desvio condicional se < Ø - (jump on negative)
1010	JZ end	desvio condicional se = Ø - (jump on zero)
1011	JC end	desvio condicional se carry = 1 - (jump on carry)
1100	JSR end	desvio para subrotina - (jump subroutine)
1101	NEG r	troca de sinal - (negate)
1110	SHR r	deslocamento para a direita - (shift right)
1111	HLT	parada - (halt)

Registradores e Modos de endereçamento

- **00 = A (registrador RA)**
- **01 = B (registrador RB)**
- **10 = X (registrador de índice)**
- **11 = nenhum registrador**

- **00 = direto**
- **01 = indireto**
- **10 = imediato**
- **11 = indexado**

- 0000 (A)
- 0100(B)
- 1000 (X)

LDA			
	a	b	x
direto	32	36	40
indireto	33	37	41
imediato	34	38	42
indexado	35	39	43

Endereço	Instrução	
0	LDR A #0	; inicializa (zera) o total
2	LDR X 129	; inicializa ponteiro
4	LDR B 128	; inicializa contador
6	JZ 16	; testa se contador é zero
8	ADD A 0,X	; soma com posição de memória
10	ADD X #1	; incrementa ponteiro
12	SUB B #1	; decrementa contador
14	JMP 6	; retorna ao início do laço
16	STR A 130	; atualiza total
18	HLT	
128	n	número de posições
129	e	endereço inicial
130	total	

Conclusão

- Conhecemos um pouco sobre os tipos de processadores e sua arquitetura em computadores.
- Vimos as várias formas de acessar um operando – modo de endereçamento.
- A tecnologia continua a evoluir, portanto o estudo não para aqui.

Referências

WEBER, Raul Fernando. Fundamentos de arquitetura de computadores. 4. ed. Porto Alegre: Bookman, 2012. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788540701434

STALLINGS, William. Arquitetura e organização de computadores. 8.ed. São Paulo: Pearson, 2010. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/459/epub/0

HOGLUND, Greg. Como quebrar códigos: a arte de explorar (e proteger) software. São Paulo: Pearson, 2006. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/179934/epub/0

Fim