ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

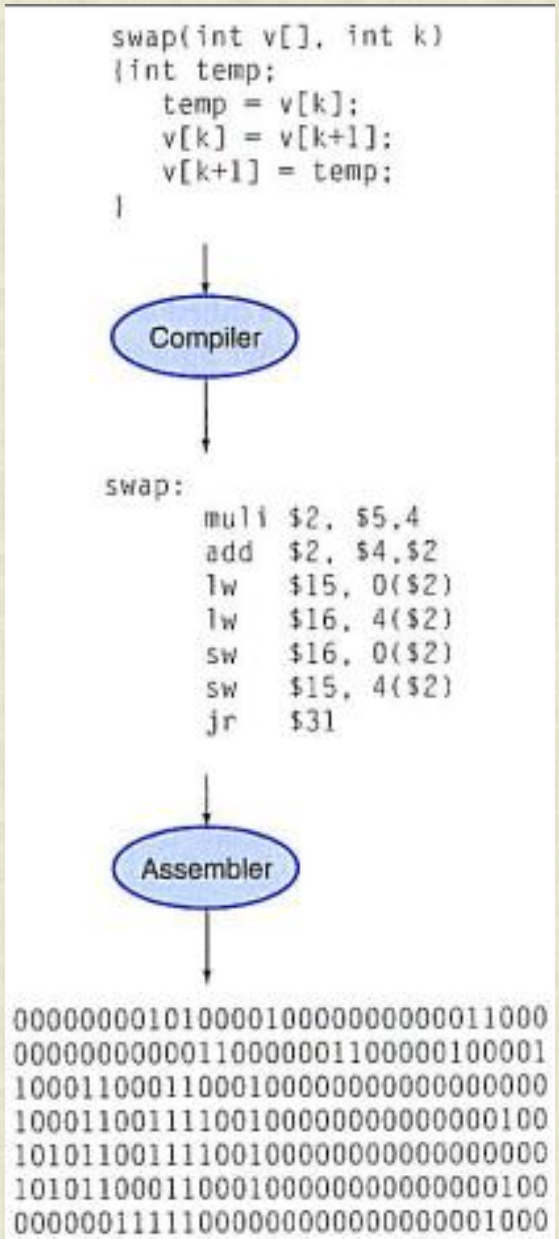
Computador Cleópatra

Introdução a Arquitetura e Programação

Alexandre Amory
Edson Moreno

Na Aula Anterior …

- **Abstração em Software**



Programa em linguagem C

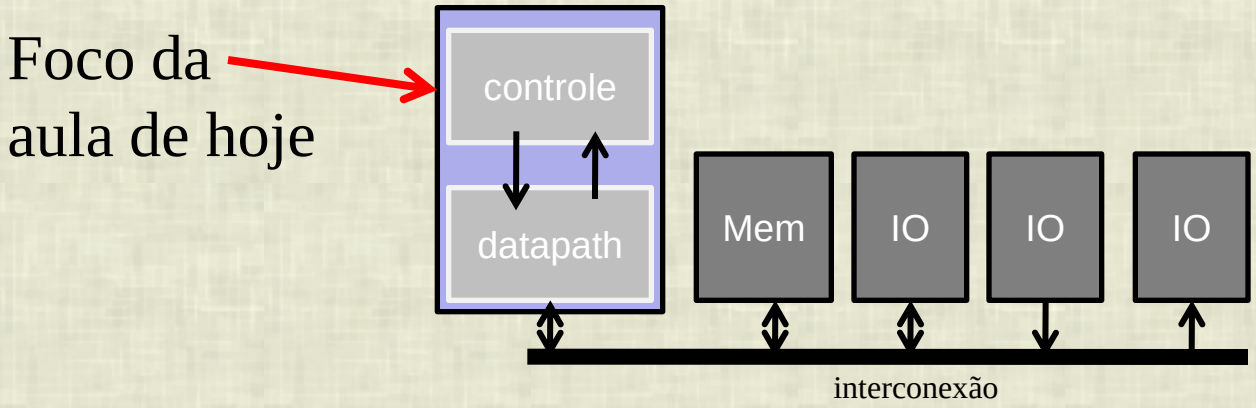
Programa em linguagem Assembly

Programa em linguagem de máquina em binário

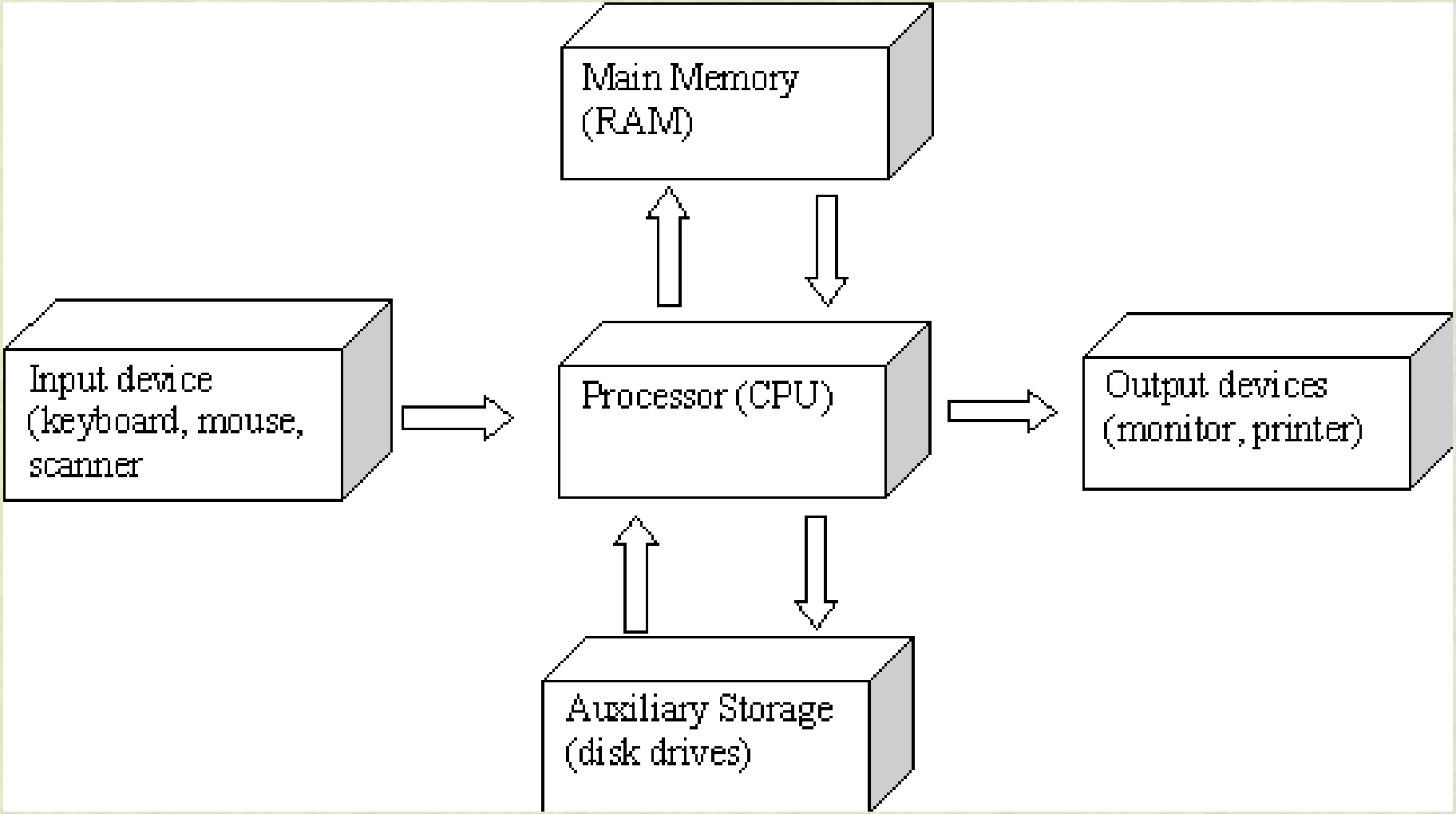
* Computer Organization and Design: The Hardware and Software Interface

Na Aula de Hoje …

- **Introdução a um sistema microprocessado**
- **O processo de transformação de código alto nível (Java, C) em linguagem de máquina binária**
- **Linguagem Assembly**

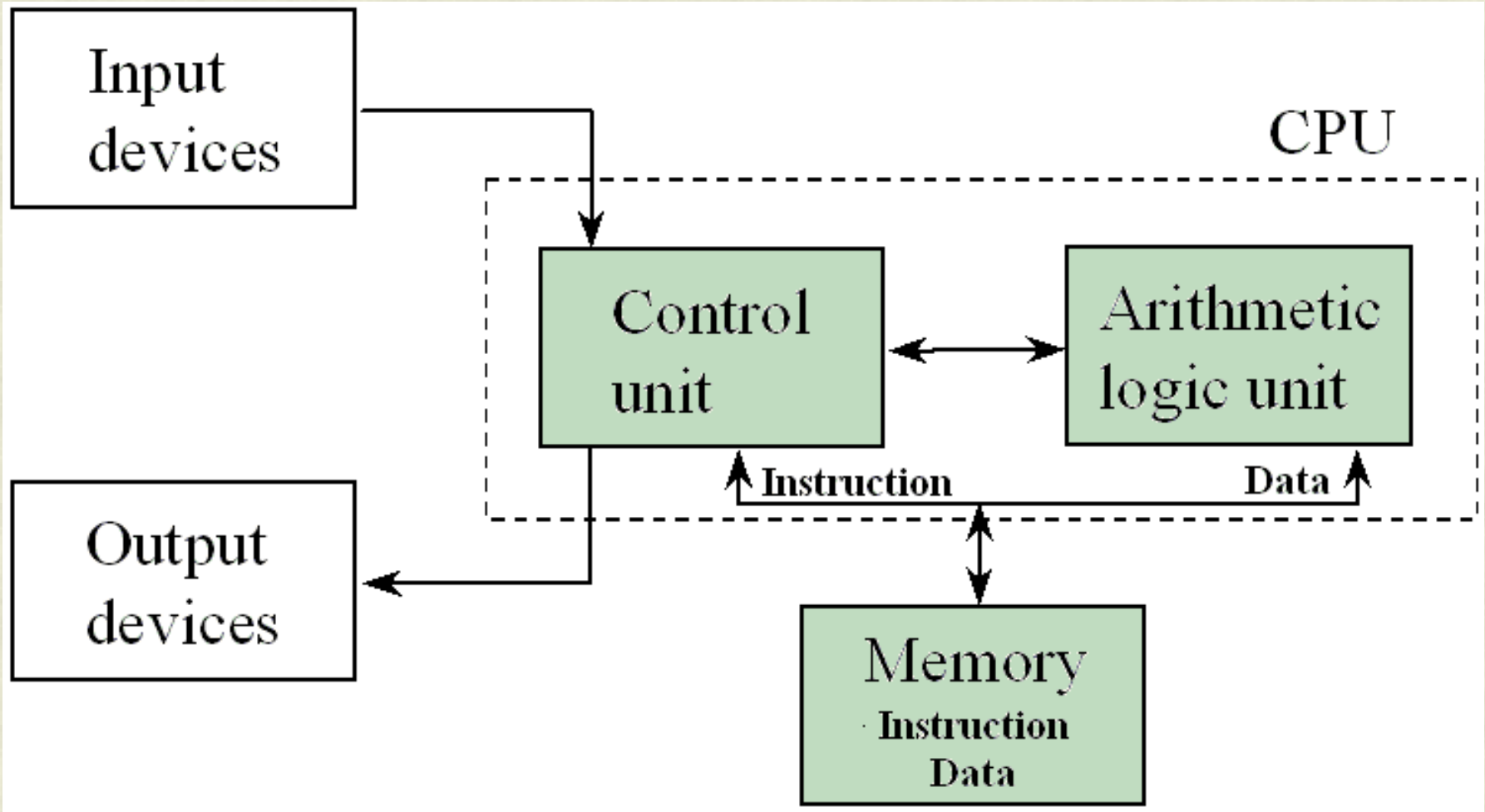


Arquiteturas de Computadores



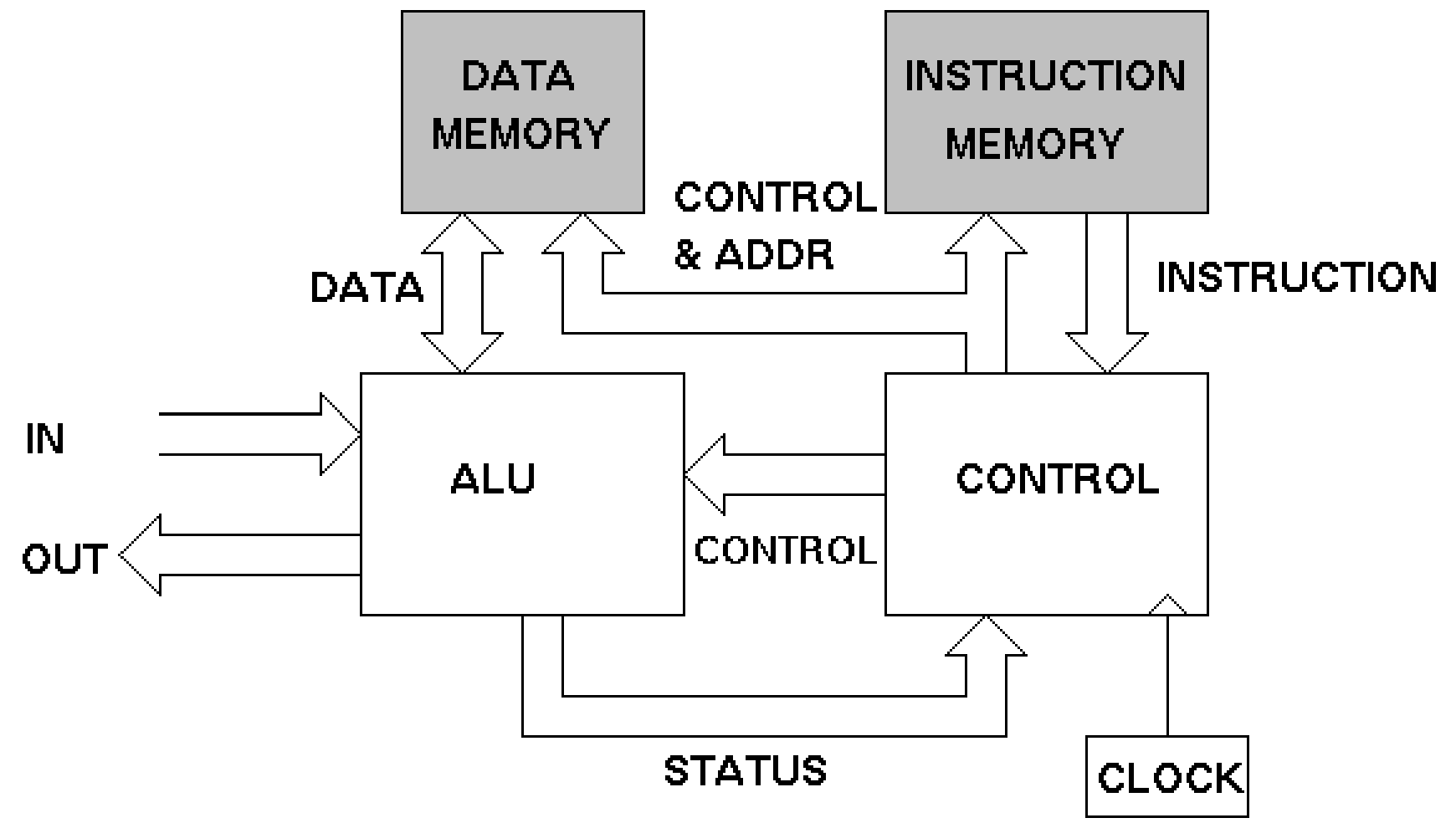
Arquitetura Von Neumann

- **Memória única de dados e instruções**



Arquitetura Harvard

- **Memória de dados e instruções são distintas**

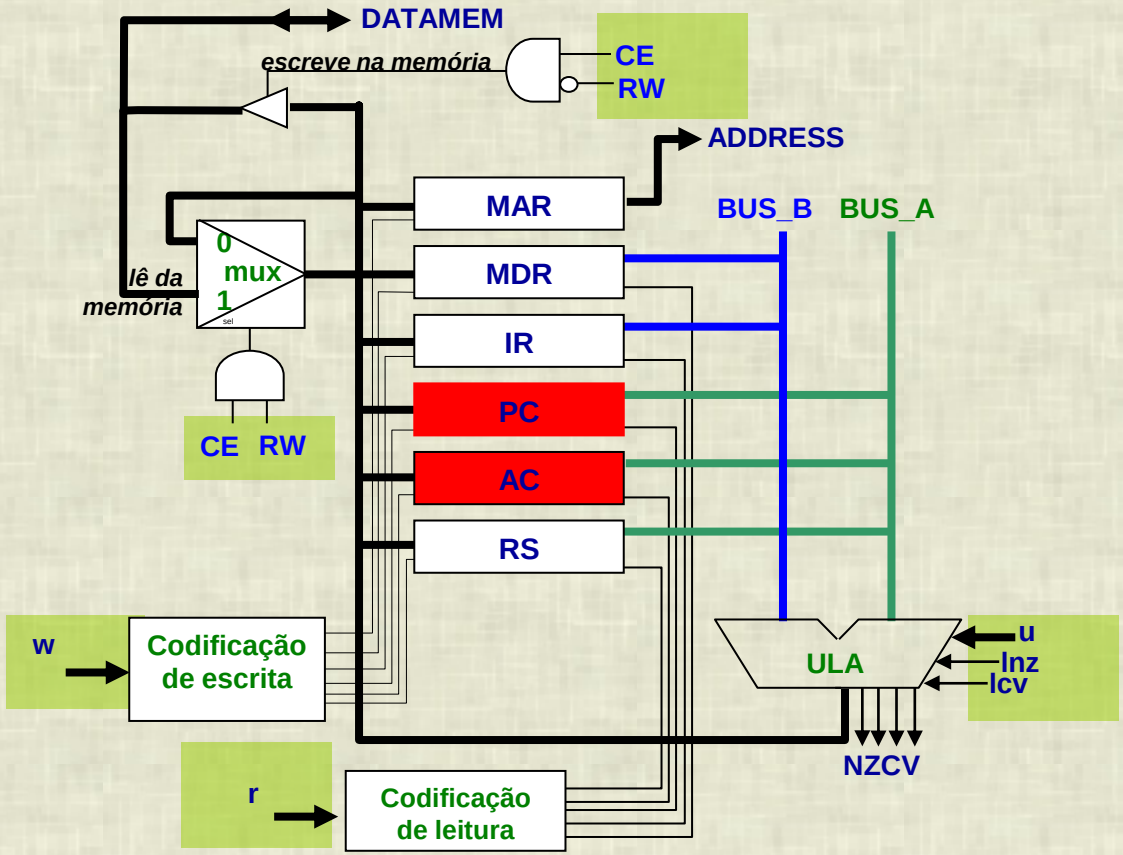


Arquitetura Cleópatra - Von Neumann

- **Cleópatra é um processador de 8 bits**
- **introduzir os conceitos gerais de organização de computadores**
- **barramento de endereços de apenas 8 bits**
  - mapa de memória de apenas 256 (2^8) posições disponíveis para armazenar programas e dados
  - Objetivo é facilitar a aprendizagem
- **Os conceitos apresentados na Cleo podem ser encontrados em processadores comerciais**
  - 8051 (Intel)
  - 68HC11 (Motorola)
- **Apesar de ter somente 14 instruções, a Cleo é uma arquitetura CISC**
  - CISC: complex instrustion set
  - RISC: reduced instruction set

Arquitetura Cleópatra: Principais Registradores

- **Registrador Primário de Dados:**
  - 1 Registrador Acumulador AC
- **Registradores de Controle:**
  - 1 Contador de Programa PC



Qualificadores

- **Existem mais qualificadores possíveis. Cleo usa apenas 4:**
  - Resultado da operação é zero **(Z)**
    - Qualificador **Z** recebe 1 quando todos os bits do resultado de uma operação forem 0, senão recebe 0
  - Resultado da operação é um número negativo **(N)**
    - Qualificador **N** recebe 1 quando o bit mais significativo do resultado de uma operação for 1, (representação de um número negativo em complemento de 2) senão recebe 0
  - Não foi possível representar o resultado da operação com números **sem** sinal **(carry**, ou apenas **C)**
    - Qualificador **C** recebe 1 quando o último estágio do somador gerar um “vai um”
  - Não foi possível representar o resultado da operação com números **com** sinal **(overflow**, ou apenas **Ov**, ou também **V)**
    - Qualificador **V** recebe 1 quando a soma de dois positivos der negativa, a soma de dois negativos der positiva, um positivo subtraído de um negativo der negativo, um negativo subtraído de um positivo der positivo, senão recebe 0
- **Conforme os tipos de dados que estamos trabalhando em alto nível, a informação dos qualificadores pode ser utilizada pelos compiladores para gerar linguagem de montagem adequada**
  - Qualificadores Z e C são usados em operações aritméticas de números sem sinal
  - Qualificadores Z e N são usados em operações lógicas
  - Qualificadores Z, N e V são usados em operações aritméticas de números com sinal

Instruções

Mnemônico	Operação
NOT	Complementa (inverte) todos os bits de AC.
STA oper	Armazena AC na memória dada por oper.
LDA oper	Carrega AC com conteúdos de memória da posição dada por oper.
ADD oper	Adiciona AC ao conteúdo da memória dada por oper.
OR oper	Realiza OU lógico do AC com conteúdo da memória dada por oper.
AND oper	Realiza E lógico do AC com conteúdo da memória dada por oper.
JMP oper	PC recebe dado especificado por oper (desvio incondicional).
JC oper	Se C=1, então PC recebe valor dado por oper (desvio condicional).
JV oper	Se V=1, então PC recebe valor dado por oper (desvio condicional).
JN oper	Se N=1 então PC recebe valor dado por oper (desvio condicional).
JZ oper	Se Z=1, então PC recebe valor dado por oper (desvio condicional).
JSR oper	RS recebe conteúdo de PC e PC recebe dado de oper (subrotina).
RTS	PC recebe conteúdos de RS (retorno de subrotina).
HLT	Suspende processo de busca e execução de instruções.

Modo de Endereçamento

- **Imediato**
  - Operando é o próprio dado
  - Usado para representar constantes
    - Ex.: Número **45** do trecho de código C

```
int valor;
int *p = &valor;
int entrada = 8;

*p = entrada + 45;
```

- **Direto**
  - Operando é endereço do dado
  - Usado para representar variáveis
    - Ex. Variável **entrada** do trecho de código C

- **Indireto**
  - Operando é endereço do endereço do dado
  - Usado para representar ponteiros
    - Ex. Ponteiro ***p** (que aponta para valor) do trecho de código C

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C = A+B
- **Assembly da Cleo**
  - LDA A
  - ADD B
  - STA C

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C = A+B
- **Assembly da Cleo**
  - LDA A
  - ADD B
  - STA C

Carrega a variável A no reg interno

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C = A+B
- **Assembly da Cleo**
  - LDA A
  - ADD B
  - STA C

Soma o conteúdo do reg interno (A) com a variável B. o resultado é salvo no reg interno. Utiliza Modo Direto.

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C = A+B
- **Assembly da Cleo**
  - LDA A
  - ADD B
  - STA C

Carrega conteúdo do reg interno (A+B) na variável C

- **Veremos mais exemplos até o fim deste módulo ...**

Formatos de Instrução da Cleópatra

- **Intruções sem operando: NOT, RST HLT**

INS							
I	I	I	I	-	-	-	-

- **Instruções com 1 operando:**

INS				ME			
I	I	I	I	M	M	-	-
D	D	D	D	D	D	D	D

- **INS: LDA, ADD, STA, etc**
- **ME: imediato, direto, indireto**

Formatos de Instrução: Observações

- Na codificação de instruções, uma posição marcada com "–" indica que o valor do bit em questão é irrelevante, podendo ser 0 ou 1;
- operando - operando que depende do modo de endereçamento para ter seu valor e semântica determinados;
- Instruções lógicas (NOT, AND e OR) e LDA - afetam os códigos de condição N e Z;
- Instrução aritmética (ADD) - Afeta N, Z, C e V;
- Instruções restantes (STA, JMP, JN, JZ, JC, JV, JSR, RTS HLT) - não alteram códigos de condição;
- Instruções sem operando (NOT, RTS e HLT) - estas instruções operam sobre dados implícitos ou não têm operandos explícitos;
- Instruções de desvio (JMP, JN, JZ, JC, JV, JSR) e STA - nestas, os modos de endereçamento direto e imediato são idênticos (possuem o mesmo efeito). Para uma definição do que é modo de endereçamento, ver Seção seguinte.

Código do Modo de Endereçamento

Denominação	Codificação	Descrição
Imediato	00	Operando é o próprio dado de 8 bits armazenado segunda palavra de memória da instrução.
Direto	01	Operando é endereço de memória de 8 bits onde se encontra o dado.
Indireto	10	Operando é endereço de memória de 8 bits onde se encontra o endereço do dado.

Código das Instruções

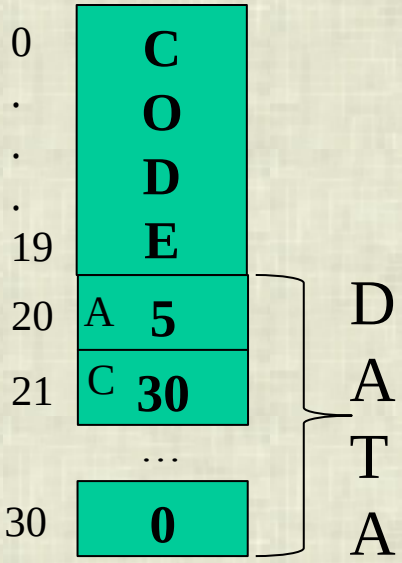
Descrição da Instrução	Mnemônico	Código	Descrição
Inversão do acumulador	NOT	000-	Complementa (ou seja, inverte) todos os bits de AC.
Escrita na memória	STA **operando**	001-	Armazena dado em AC na posição de memória especificada por **operando**.
Leitura da memória para o acumulador	LDA **operando**	0100	Carrega AC com conteúdos de memória da posição especificada por **operando**.
Adição de valor a valor do acumulador	ADD **operando**	0101	Adiciona dado em AC a conteúdo da memória dada por **operando**, colocando o resultado em AC.
Ou de valor com valor do acumulador	OR **operando**	0110	Faz OU bit a bit de AC com conteúdo da memória dada por **operando**, colocando o resultado em AC.
E de valor com valor do acumulador	AND **operando**	0111	Faz E bit a bit de AC com conteúdo da memória dada por **operando**, colocando o resultado em AC.
Salto incondicional	JMP **operando**	1000	PC recebe valor especificado por **operando**.
Salto condicionado ao valor do qualificador de vai-um	JC **operando**	1001	Se C=1, PC recebe valor dado por **operando**. Senão, PC← PC+2.
Salto condicionado ao valor do qualificador de transbordo aritmético	JV **operando**	1110	Se V=1, PC recebe valor dado por **operando**. Senão, PC← PC+2.
Salto condicionado ao valor do qualificador de sinal	JN **operando**	1010	Se N=1, PC recebe valor dado por **operando**. Senão, PC← PC+2.
Salto condicionado ao valor do qualificador zero	JZ **operando**	1011	Se Z=1, PC recebe valor dado por **operando**. Senão, PC← PC+2.
Salto incondicional para subrotina	JSR **operando**	1100	RS recebe conteúdo de PC, enquanto que PC recebe valor especificado por **operando**.
Retorno de subrotina	RTS	1101	PC recebe conteúdos de RS.
Parada do Processador	HLT	1111	Suspende a realização dos ciclos de busca, decodificação e execução.

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C[0] = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I
- **Linguagem de máquina**

LDA DIR

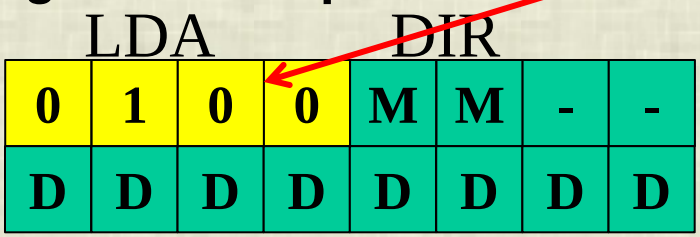
I	I	I	I	M	M	-	-
D	D	D	D	D	D	D	D

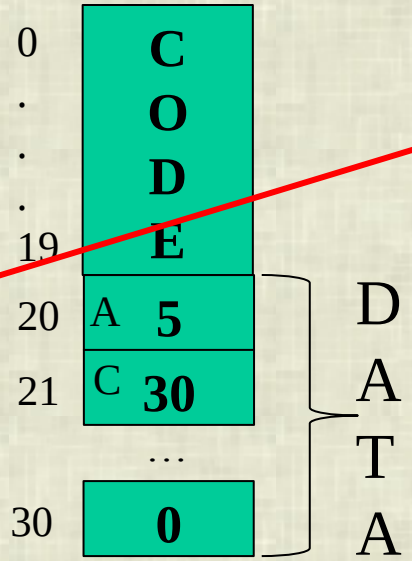


Mnemônico	Código
NOT	000-
STA operando	001-
LDA operando	0100
ADD operando	0101
OR operando	0110
AND operando	0111
JMP operando	1000
JC operando	1001
JV operando	1110
JN operando	1010
JZ operando	1011
JSR operando	1100
RTS	1101
HLT	1111

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C[0] = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I
- **Linguagem de máquina**



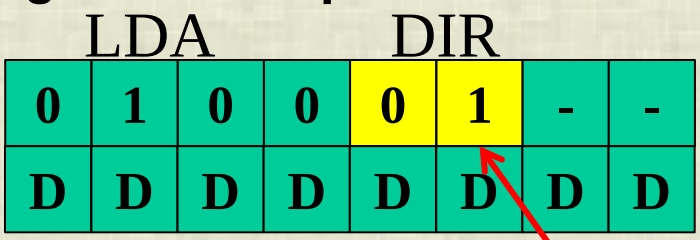


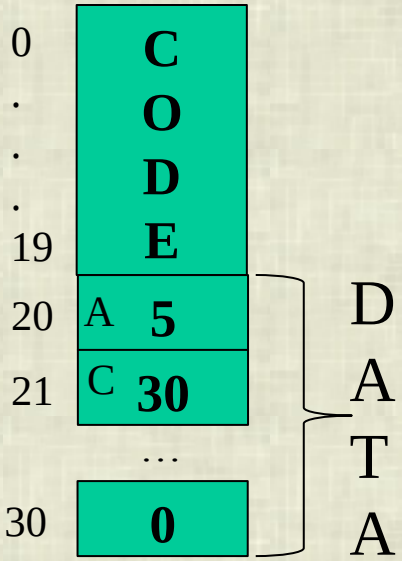
Denominação	Codificação
Imediato	00
Direto	01
Indireto	10

Mnemônico	Código
NOT	000-
STA operando	001-
LDA operando	0100
ADD operando	0101
OR operando	0110
AND operando	0111
JMP operando	1000
JC operando	1001
JV operando	1110
JN operando	1010
JZ operando	1011
JSR operando	1100
RTS	1101
HLT	1111

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C[0] = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I
- **Linguagem de máquina**



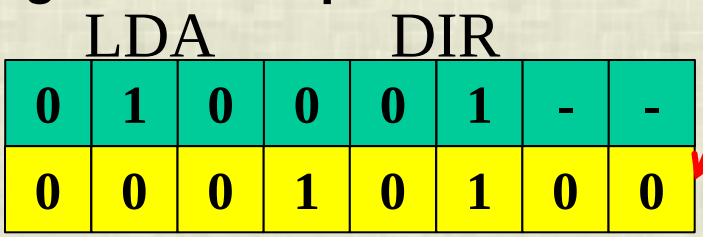


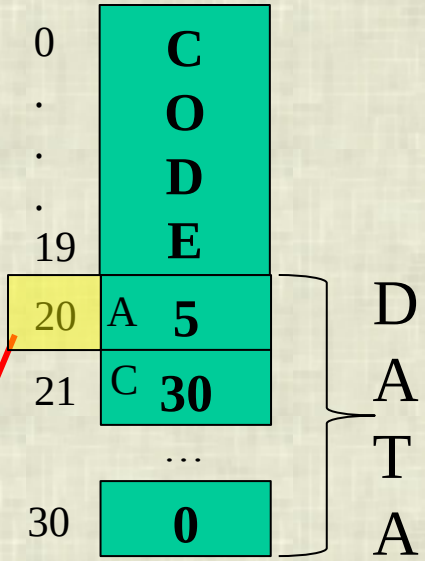
Denominação	Codificação
Imediato	00
Direto	01
Indireto	10

Mnemônico	Código
NOT	000-
STA operando	001-
LDA operando	0100
ADD operando	0101
OR operando	0110
AND operando	0111
JMP operando	1000
JC operando	1001
JV operando	1110
JN operando	1010
JZ operando	1011
JSR operando	1100
RTS	1101
HLT	1111

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C[0] = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I
- **Linguagem de máquina**





Denominação	Codificação
Imediato	00
Direto	01
Indireto	10

Mnemônico	Código
NOT	000-
STA operando	001-
LDA operando	0100
ADD operando	0101
OR operando	0110
AND operando	0111
JMP operando	1000
JC operando	1001
JV operando	1110
JN operando	1010
JZ operando	1011
JSR operando	1100
RTS	1101
HLT	1111

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - C[0] = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I
- **Linguagem de máquina**

LDA DIR

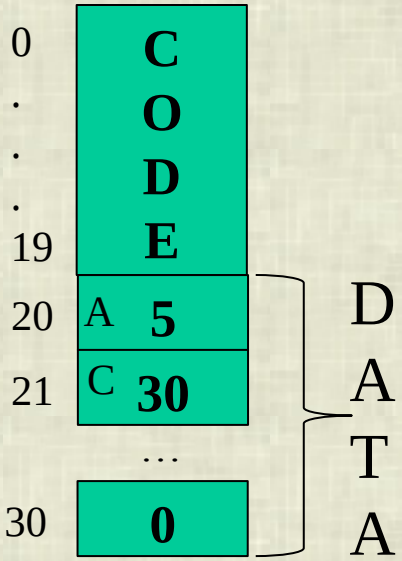
0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0

ADD IM

0	1	0	1	0	0	0	0
0	0	0	0	0	0	1	1

STA IN

0	0	1	0	1	0	0	0
0	0	0	1	0	1	0	1



Mnemônico	Código
NOT	000-
STA operando	001-
LDA operando	0100
ADD operando	0101
OR operando	0110
AND operando	0111
JMP operando	1000
JC operando	1001
JV operando	1110
JN operando	1010
JZ operando	1011
JSR operando	1100
RTS	1101
HLT	1111

Da Arquitetura Cleópatra para a Organização

- **Da arquitetura Cleópatra**
  - a visão do programador em linguagem de montagem
- **Para a Organização Cleópatra**
  - A visão do projetista de hardware
  - Somente uma forma de implementar a arquitetura

Resumo

- **Apresentação dos principais macro blocos de um processador**
  - Parte de controle
  - Parte operativa (de dados)
  - Entrada e saída
  - Memória

- **O restante desta disciplina detalha cada uma destas partes !!!!**