

# Componentes do Computador

## Princípios Básicos e máquinas de 1º geração

Disciplina: Introdução à Arquitetura de Computadores

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: [lucianobrum18@gmail.com](mailto:lucianobrum18@gmail.com)

# Tópicos



- **Princípios Básicos;**
- **Elementos funcionais básicos;**
- **Computador de primeira geração: EDVAC;**
- **Modelo de Von Neumann: o computador IAS;**
- **Arquiteturas de 4,3,2,1 e 0 endereços;**
- **Resumo;**

## ➤ Arquitetura x Organização

- Arquitetura de computador se refere aos atributos de um sistema que são visíveis para o programador (**Conjunto de instruções, número de bits usados para representação de dados, mecanismos de E/S, técnicas de endereçamento, etc** ).

## ➤ Arquitetura x Organização

- Arquitetura de computador se refere aos atributos de um sistema que são visíveis para o programador (**Conjunto de instruções, número de bits usados para representação de dados, mecanismos de E/S, técnicas de endereçamento, etc** ).
- A organização refere-se as unidades operacionais e suas interconexões que implementam as especificações de sua arquitetura (**Sinais de controle, interfaces entre o computador e os periféricos, tecnologia de memória, tamanho da memória física, frequência de clock, etc** ).

➤ O computador realiza tarefas através de um conjunto de operações;

➤ Definição:

➤ **Instrução** =

**OPERAÇÃO**

**OPERANDOS**

- O computador realiza tarefas através de um conjunto de operações;
- Definição:
  - **Instrução** = 

OPERAÇÃO	OPERANDOS
----------	-----------
  - **Programa**: sequência pré-determinada de instruções, que deve ser seguida para atingir o objetivo computacional;



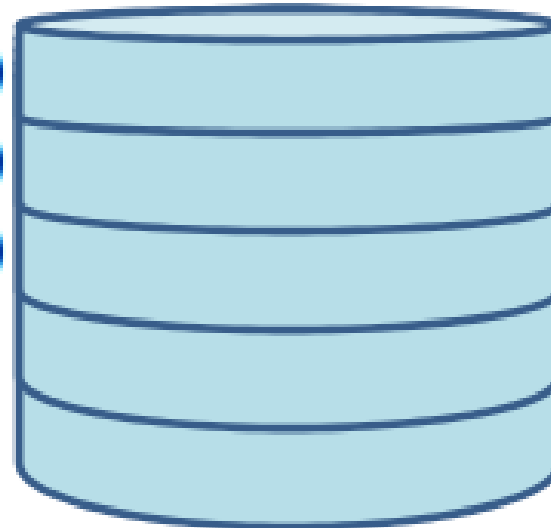
- O computador realiza tarefas através de um conjunto de operações;
- Definição:
  - **Instrução** = 

OPERAÇÃO	OPERANDOS
----------	-----------
  - **Programa**: sequência pré-determinada de instruções, que deve ser seguida para atingir o objetivo computacional;
  - **Memória**: tem a função de armazenar dados e instruções, é organizada em posições que podem ser vistas como elementos em uma matriz;

# Memória

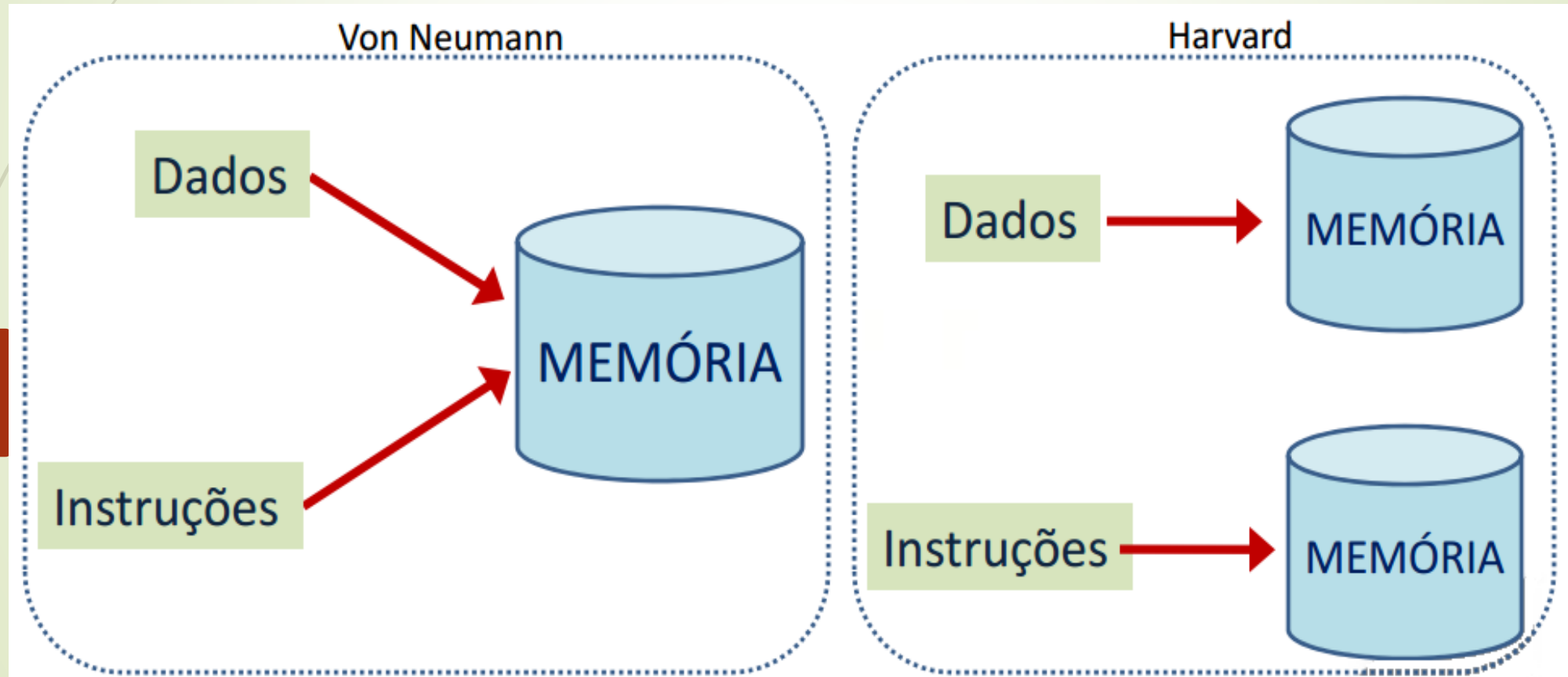
- Cada elemento na memória tem um **endereço** que representa uma **posição particular** da memória e pode ser formado de várias maneiras;

endereço  
00 (00000000)  
01 (00000001)  
02 (00000010)  
...  
FF (11111111)





- 2 Diferentes modelos de arquitetura: de Von Neumann e Harvard:



- **ULA** (unidade lógica e aritmética): realiza ações indicadas pelas instruções, executando operações aritméticas, lógicas e executando de desvios de programa;

# Princípios Básicos

- **ULA** (unidade lógica e aritmética): realiza ações indicadas pelas instruções, executando operações aritméticas, lógicas e executando de desvios de programa;
- **Unidade de Controle**: gerencia o fluxo interno de dados e o instante em que ocorrem as transferências entre os elementos da organização através de **sinais de controle**;

- **ULA** (unidade lógica e aritmética): realiza ações indicadas pelas instruções, executando operações aritméticas, lógicas e executando de desvios de programa;
- **Unidade de Controle**: gerencia o fluxo interno de dados e o instante em que ocorrem as transferências entre os elementos da organização através de **sinais de controle**;
- **Registradores**: elementos digitais capazes de armazenar dados. Têm associados a si um sinal de carga que indica quando o novo conteúdo deve ser armazenado;

# Princípios Básicos

- **ULA** (unidade lógica e aritmética): realiza ações indicadas pelas instruções, executando operações aritméticas, lógicas e executando de desvios de programa;
- **Unidade de Controle**: gerencia o fluxo interno de dados e o instante em que ocorrem as transferências entre os elementos da organização através de **sinais de controle**;
- **Registradores**: elementos digitais capazes de armazenar dados. Têm associados a si um sinal de carga que indica quando o novo conteúdo deve ser armazenado;
- A Unidade de Controle, a Unidade Lógica e Aritmética e os registradores internos formam a **Unidade Central de Processamento** (**UCP**, ou em inglês **CPU**);

- **E/S:** dispositivos que permitem a entrada e saída de dados do computador;



## Entrada e Saída

➤ **E/S:** dispositivos que permitem a entrada e saída de dados do computador;

➤ Exemplos:

➤ Mouse;



➤ Teclado;



➤ Disco;

➤ Pen-drive;

➤ Impressora;



# Barramentos

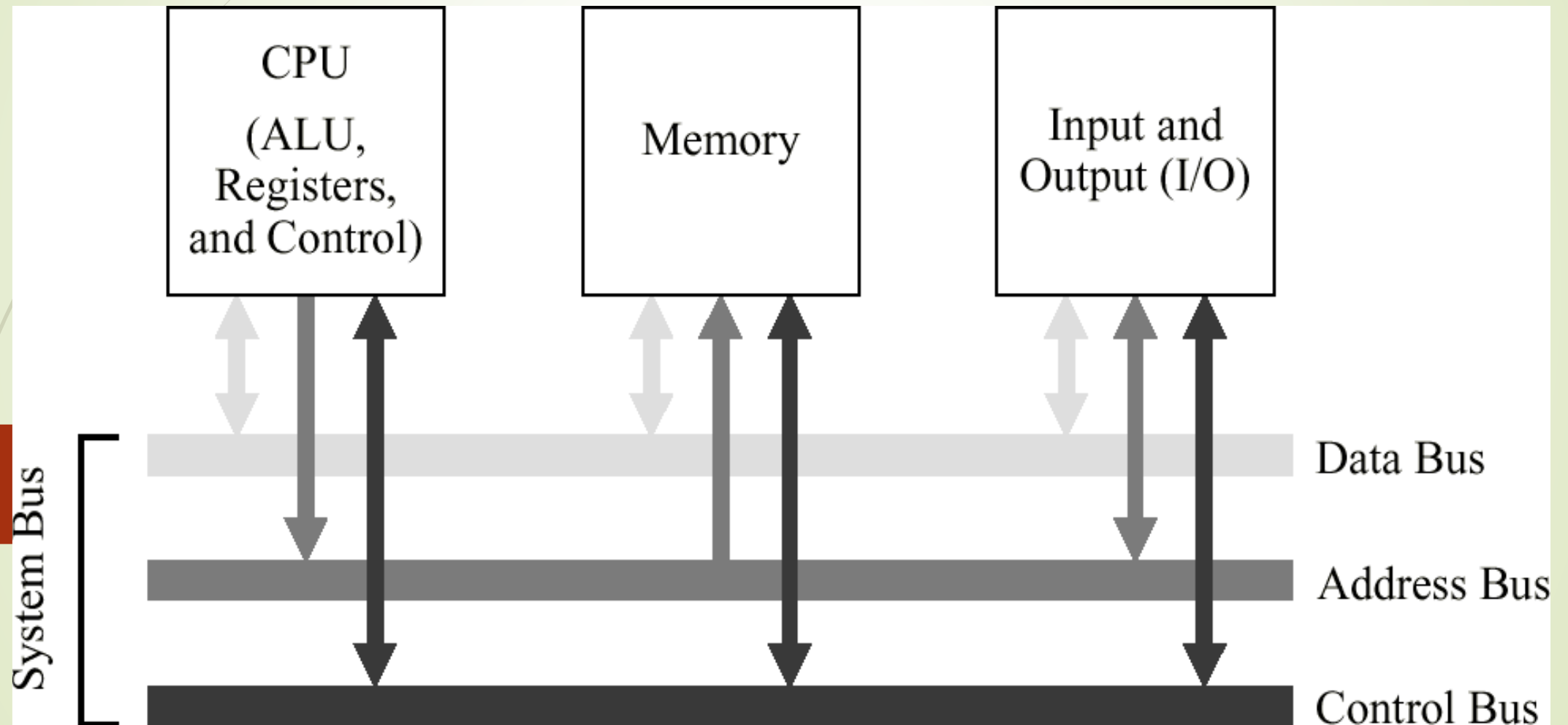
- São caminhos que permitem o transporte de dados entre os vários elementos da CPU, memória e dispositivos de E/S;

# Barramentos

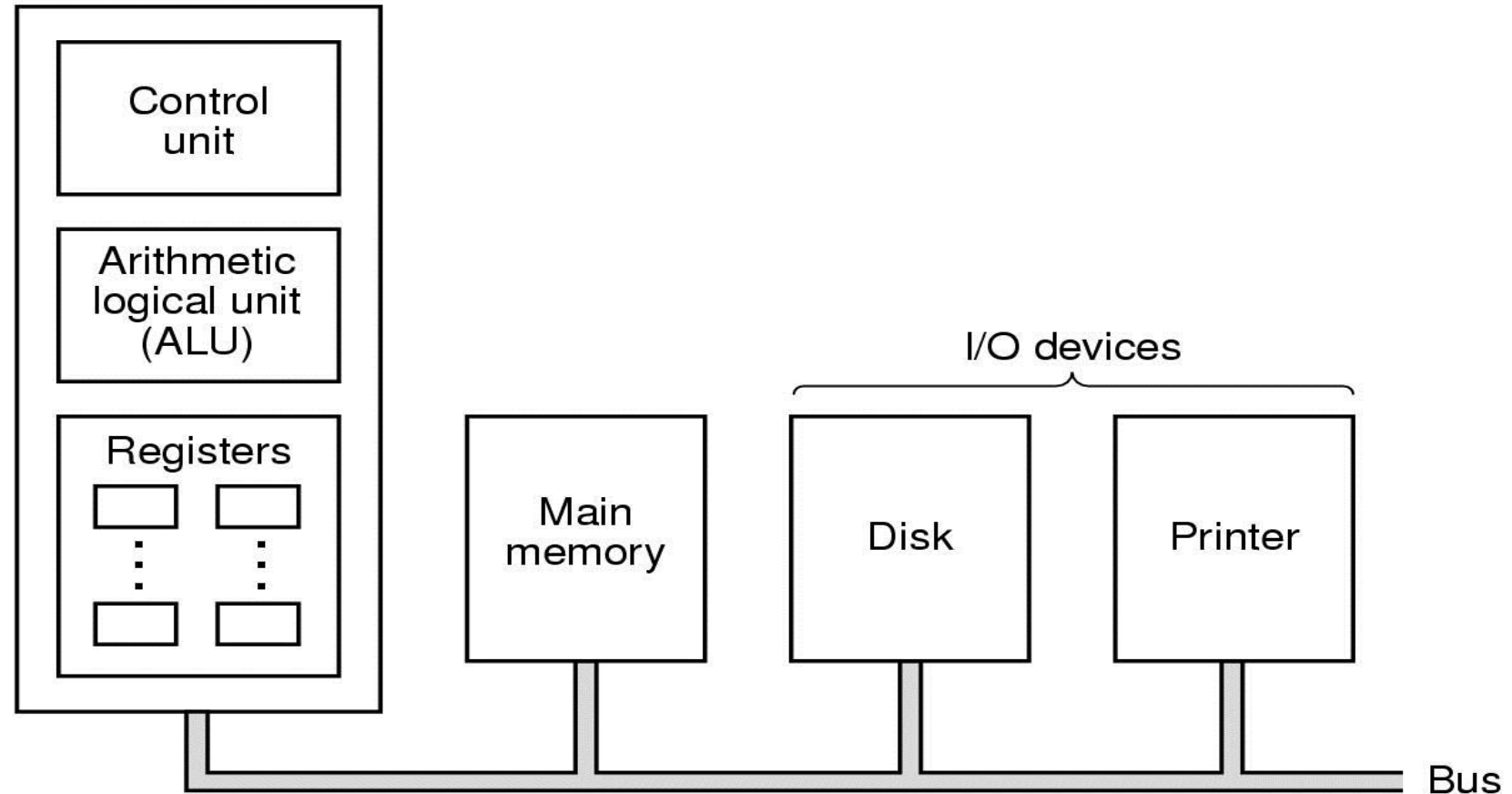
- São caminhos que permitem o transporte de dados entre os vários elementos da CPU, memória e dispositivos de E/S;
- O barramento só pode receber dados de **uma fonte por vez**;

# Barramentos

- São caminhos que permitem o transporte de dados entre os vários elementos da CPU, memória e dispositivos de E/S;
- O barramento só pode receber dados de **uma fonte por vez**;
- É caracterizado pela **largura em bits**, que deve ser correspondente ao tamanho dos dados e endereços transportados;



### Central processing unit (CPU)



**Figura 1: Organização de um computador simples. Fonte: Tanenbaum, 2010.**



- Ciclo de **Busca** - Decodificação – Execução de Instruções:
  - O elemento no processador chamado de **Contador de Programa (PC)** armazena o **endereço** da **próxima instrução** a ser executada;

## ➤ Ciclo de **Busca** - Decodificação – Execução de Instruções:

- O elemento no processador chamado de **Contador de Programa (PC)** armazena o **endereço** da **próxima instrução** a ser executada;
- Quando uma execução de instrução tem início, é buscada da **memória** a instrução contida no endereço apontado pelo PC;

## ➤ Ciclo de **Busca** - Decodificação – Execução de Instruções:

- O elemento no processador chamado de **Contador de Programa (PC)** armazena o **endereço** da **próxima instrução** a ser executada;
- Quando uma execução de instrução tem início, é buscada da **memória** a instrução contida no endereço apontado pelo PC;
- Essa instrução é armazenada em um **registrador de instruções**;

- **Ciclo de Busca - Decodificação – Execução de Instruções:**
  - A instrução é interpretada pelo **decodificador** que faz com que sinais eletrônicos sejam gerados no processador como resultado do valor do campo de operação;

## ➤ Ciclo de Busca - **Decodificação** – Execução de Instruções:

- A instrução é interpretada pelo **decodificador** que faz com que sinais eletrônicos sejam gerados no processador como resultado do valor do campo de operação;

➤ Instrução =

OPERAÇÃO	OPERANDOS
10101001	1111010011001101

## ➤ **Ciclo de Busca - Decodificação – Execução de Instruções:**

- A aplicação da função da operação nos operandos através dos sinais decodificados e de sinais de controle caracterizam a **execução** da instrução;



## ➤ Ciclo de Busca - Decodificação – Execução de Instruções:

- A aplicação da função da operação nos operandos através dos sinais decodificados e de sinais de controle caracterizam a **execução** da instrução;
- Após a execução, o contador de programa é **atualizado** com o endereço da próxima instrução a ser buscada;

- Uma instrução pode conter não só endereços de operandos, mas também de **outras instruções**;

- Uma instrução pode conter não só endereços de operandos, mas também de **outras instruções**;
- Essas instruções causam mudanças no **fluxo do programa** como resultado das condições dos dados;

- Uma instrução pode conter não só endereços de operandos, mas também de **outras instruções**;
- Essas instruções causam mudanças no **fluxo do programa** como resultado das condições dos dados;
- São instruções de **desvio condicional e incondicional**;
  - Ex: Se  $(a > b)$  Faça C, se não, Faça D.

- O processador somente executa código em linguagem de máquina;

- O processador somente executa código em linguagem de máquina;
- A linguagem de máquina é numérica, representada em forma binária;



- O processador somente executa código em linguagem de máquina;
- A linguagem de máquina é numérica, representada em forma binária;
- A linguagem de máquina representa o conjunto de instruções e dados no computador;

00110000 10000000 | 01100000 10000000

- O processador somente executa código em linguagem de máquina;
- A linguagem de máquina é numérica, representada em forma binária;
- A linguagem de máquina representa o conjunto de instruções e dados no computador;

0011000010000000 | 0110000010000000

**ADD**

**NOT**

- Para facilitar a programação de um processador:
  - Mnemônicos são associados as instruções.

- Para facilitar a programação de um processador:
  - Mnemônicos são associados as instruções.
  - Nomes são associados aos operandos.

- Para facilitar a programação de um processador:
  - Mnemônicos são associados as instruções.
  - Nomes são associados aos operandos.
  - Rótulos são associados as posições ocupadas pelo programa.

- Para facilitar a programação de um processador:
  - Mnemônicos são associados as instruções.
  - Nomes são associados aos operandos.
  - Rótulos são associados as posições ocupadas pelo programa.
- Outro problema: Como traduzir da linguagem simbólica (assembly) para linguagem de máquina?



- Para facilitar a programação de um processador:
  - Mnemônicos são associados as instruções.
  - Nomes são associados aos operandos.
  - Rótulos são associados as posições ocupadas pelo programa.
- Outro problema: Como traduzir da linguagem simbólica (assembly) para linguagem de máquina? R: o processo de tradução é chamado montagem e quando automatizado, é chamado de montador (assembler).

# Tópicos



- ~~Princípios Básicos;~~
- **Elementos funcionais básicos;**
- **Computador de primeira geração: EDVAC;**
- **Modelo de Von Neumann: o computador IAS;**
- **Arquiteturas de 4,3,2,1 e 0 endereços;**
- **Resumo;**

## Elementos funcionais básicos

- O processador, memória e os dispositivos de E/S são formados por elementos de **menor complexidade**;

## Elementos funcionais básicos

- O processador, memória e os dispositivos de E/S são formados por elementos de **menor complexidade**;
- Registradores, contadores, multiplexadores, seletores, decodificadores, somadores e portas lógicas são alguns desses elementos;

## Elementos funcionais básicos

- O processador, memória e os dispositivos de E/S são formados por elementos de **menor complexidade**;
- Registradores, contadores, multiplexadores, seletores, decodificadores, somadores e portas lógicas são alguns desses elementos;
- Contadores, multiplexadores, seletores, decodificadores, somadores e portas lógicas operam sobre os dados, alterando-os ou fornecendo um novo dado como resultado;

## Elementos funcionais básicos

- Elementos digitais precisam **ser ativados ou habilitados** para realizar suas funções;



## Elementos funcionais básicos

- Elementos digitais precisam **ser ativados ou habilitados** para realizar suas funções;
- Os sinais responsáveis pela ativação dos componentes digitais são chamados de **sinais de controle**;

# Memória

- Uma memória é dividida em **palavras** (**words**);

# Memória

- Uma memória é dividida em **palavras** (**words**);
- Uma palavra pode ter vários bytes de tamanho;

# Memória

- Uma memória é dividida em **palavras** (**words**);
- Uma palavra pode ter vários bytes de tamanho;
- Cada palavra da memória é identificada por um **endereço**;

# Memória

- Uma memória é dividida em **palavras** (**words**);
- Uma palavra pode ter vários bytes de tamanho;
- Cada palavra da memória é identificada por um **endereço**;
- O conteúdo da palavra de memória pode ser tanto um **dado** como uma **instrução**;

**Figura 2: Estrutura convencional de memória. Fonte: Adaptado de Weber, 2001.**



- Uma memória é caracterizada por:
  - Tamanho;
  - Velocidade;
  - Tecnologia;

➤ Uma memória é caracterizada por:

- Tamanho;
- Velocidade;
- Tecnologia;

➤ A nível de arquitetura:

- Tamanho da palavra em bits (afeta tamanho do RDM);
- Tamanho da memória em palavras (afeta tamanho do REM);



- Processadores de 16 bits endereçam até  $2^{16}$  bits de RAM = 64Kb
- Processadores de 32 bits endereçam até  $2^{32}$  bits de RAM = 4Gb
- Processadores de 64 bits endereçam até  $2^{64}$  bits de RAM = 128Eb
- O Sistema Operacional também precisa suportar esta mesma capacidade de endereçamento.

# Unidade Lógica e Aritmética

- Realiza as operações **lógicas** e **aritméticas** sobre **um ou mais operandos**;



# Unidade Lógica e Aritmética

- Realiza as operações **lógicas** e **aritméticas** sobre **um ou mais operandos**;
- **Aritméticas**: soma e subtração de dois operandos, negação de um operando, inversão de um operando, etc;



# Unidade Lógica e Aritmética

- Realiza as operações **lógicas** e **aritméticas** sobre **um ou mais operandos**;
- **Aritméticas**: soma e subtração de dois operandos, negação de um operando, inversão de um operando, etc;
- **Lógicas**: AND, OR, deslocamento do operando para esquerda ou direita, rotação de um operando para esquerda ou direita;

# Unidade Lógica e Aritmética

- Realiza as operações **lógicas** e **aritméticas** sobre **um ou mais operandos**;
- **Aritméticas**: soma e subtração de dois operandos, negação de um operando, inversão de um operando, etc;
- **Lógicas**: AND, OR, deslocamento do operando para esquerda ou direita, rotação de um operando para esquerda ou direita;
- Operações **complexas** podem ser realizadas pela ativação de **várias operações básicas**;



- Uma ULA é caracterizada por:
  - Comprimento em bits dos operandos;
  - Número e tipo de operações;
  - Códigos de condição gerados;

- Uma ULA é caracterizada por:
  - Comprimento em bits dos operandos;
  - Número e tipo de operações;
  - Códigos de condição gerados;
- **Acumulador:** é um registrador que tem por função armazenar um operando ou o resultado gerado na ULA e é ativado por um sinal de controle;

- Uma ULA é caracterizada por:
  - Comprimento em bits dos operandos;
  - Número e tipo de operações;
  - Códigos de condição gerados;
- **Acumulador:** é um registrador que tem por função armazenar um operando ou o resultado gerado na ULA e é ativado por um sinal de controle;
- É caracterizado pelo seu tamanho em bits;



# Unidade de Controle

- Cada sinal de controle comanda uma **microoperação**;
- Unidade de Controle são máquinas de estado finito realizada por **lógica sequencial**;
  - Lógica sequencial: o sinal de saída é função dos sinais de entrada e o estado anterior do sistema;
  - Lógica combinacional: o sinal de saída é função exclusiva dos sinais de entrada;

- Existem 2 organizações usuais para a UC:
  - **Organização convencional:** componentes digitais (flip-flops, contadores, decodificadores,...) que geram os SC nos tempos adequados para o sistema;
  - Organização microprogramada: sinais de controle são armazenados na **memória de controle**. Os sinais são agrupados em words chamadas **microinstruções** e o conjunto delas formam um **microprograma**;

**Figura 4: Esquema de uma unidade de controle. Fonte: Adaptado de Weber, 2001.**

## Registradores Especiais:

- A localização desses registradores depende da **arquitetura e organização** de cada máquina (UC, Unid. Operacional);
- **PC**: O Contador de Programa mantém atualizado o endereço de memória da próxima instrução que deve ser executada e é caracterizado pelo **tamanho em bits**;

## Registradores Especiais:

- A localização desses registradores depende da **arquitetura e organização** de cada máquina (UC, Unid. Operacional);
- **PC**: O Contador de Programa mantém atualizado o endereço de memória da próxima instrução que deve ser executada e é caracterizado pelo **tamanho em bits**;
- **RI**: O Registrador de Instruções Armazena a instrução que está sendo executada e é caracterizado pelo **tamanho em bits**;



## Registradores Especiais:

- A localização desses registradores depende da **arquitetura e organização** de cada máquina (UC, Unid. Operacional);
- **PC**: O Contador de Programa mantém atualizado o endereço de memória da próxima instrução que deve ser executada e é caracterizado pelo **tamanho em bits**;
- **RI**: O Registrador de Instruções Armazena a instrução que está sendo executada e é caracterizado pelo **tamanho em bits**;
- **RST**: O Registrador de Estado armazena os códigos de condição gerados pela ULA e sinais de interrupção gerados por E/S, e é caracterizado pelo **tamanho em bits**;



- Instruções de **transferência** de dados;

- Instruções de **transferência** de dados;
- Instruções **aritméticas** e **lógicas**;

## Classificações comuns de instruções:

- Instruções de **transferência** de dados;
- Instruções **aritméticas** e **lógicas**;
- Instruções de teste e **desvio**;

## Classificações comuns de instruções:

- Instruções de **transferência** de dados;
- Instruções **aritméticas** e **lógicas**;
- Instruções de teste e **desvio**;
- Para a UC saber onde estão os operandos, é necessário que os **endereços dos operandos** apareçam junto a instrução;

- Em instruções de desvio, aparece o endereço de desvio em vez do endereço do operando;

➤ Ex:

➤ add \$s1,\$s2,\$s3

➤ jmp 128

OPERAÇÃO	ENDEREÇO
JMP	1000

ENDEREÇO	DADO
0000	00101110
0001	10101010
0010	11010100
...	10101010
1000	10100111

## Modos de endereçamento do computador

- **Modos de endereçamento do computador** são as diversas formas em que o endereço de um operando ou um endereço de desvio pode aparecer;

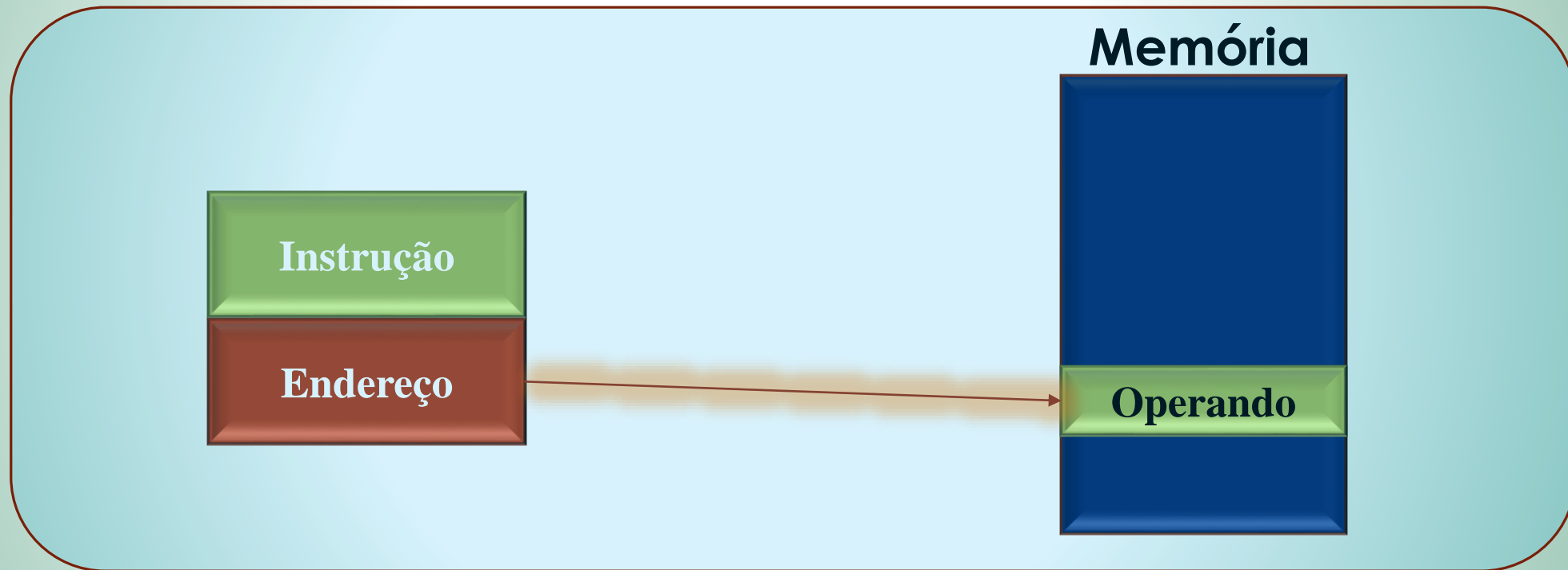


## Modos de endereçamento do computador

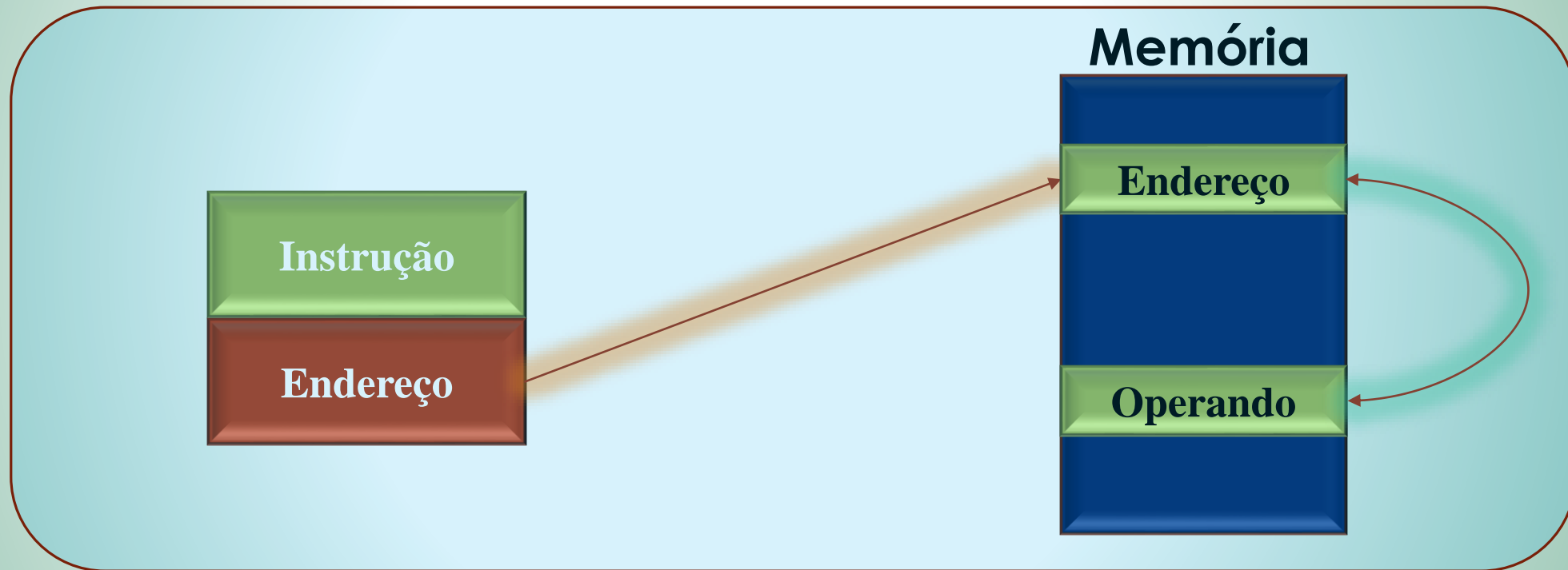
- **Modos de endereçamento do computador** são as diversas formas em que o endereço de um operando ou um endereço de desvio pode aparecer;
- Ex:
  - Endereçamento direto: o endereço do operando segue o código da instrução;

- **Modos de endereçamento do computador** são as diversas formas em que o endereço de um operando ou um endereço de desvio pode aparecer;
- Ex:
  - Endereçamento direto: o endereço do operando segue o código da instrução;
  - Endereçamento indireto: o endereço do endereço do operando segue o código da instrução;

- **Modos de endereçamento do computador** são as diversas formas em que o endereço de um operando ou um endereço de desvio pode aparecer;
- Ex:
  - Endereçamento direto: o endereço do operando segue o código da instrução;
  - Endereçamento indireto: o endereço do endereço do operando segue o código da instrução;
  - Endereçamento imediato: o operando segue o código da instrução;



**Figura 5: Endereçamento Direto. Fonte: Adaptado de Weber, 2001.**



**Figura 6: Endereçamento Indireto. Fonte: Adaptado de Weber, 2001.**



**Figura 7: Endereçamento Imediato. Fonte: Adaptado de Weber, 2001.**



# Ciclo BUSCA – DECODIFICAÇÃO – EXECUÇÃO de instruções

- Fase de Busca envolve:
  - Copiar o conteúdo do contador de programa (PC) para o registrador de endereços de memória (REM);

# Ciclo BUSCA – DECODIFICAÇÃO – EXECUÇÃO de instruções

- Fase de Busca envolve:
  - Copiar o conteúdo do contador de programa (PC) para o registrador de endereços de memória (REM);
  - Ler a instrução da memória e armazenar no registrador de dados da memória (RDM);

# Ciclo BUSCA – DECODIFICAÇÃO – EXECUÇÃO de instruções

## ➤ Fase de Busca envolve:

- Copiar o conteúdo do contador de programa (PC) para o registrador de endereços de memória (REM);
- Ler a instrução da memória e armazenar no registrador de dados da memória (RDM);
- Copiar o conteúdo do registrador de dados da memória (RDM) para o registrador de instruções (RI);

# Ciclo BUSCA – DECODIFICAÇÃO – EXECUÇÃO de instruções

## ➤ Fase de Busca envolve:

- Copiar o conteúdo do contador de programa (PC) para o registrador de endereços de memória (REM);
- Ler a instrução da memória e armazenar no registrador de dados da memória (RDM);
- Copiar o conteúdo do registrador de dados da memória (RDM) para o registrador de instruções (RI);
- Atualizar o contador de programa (PC) para a próxima instrução a ser buscada;

➤ Fase de Decodificação envolve:

- Determinar qual instrução deve ser executada através da decodificação do campo de operação da instrução;
- A decodificação é realizada por lógica combinacional;

- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;



- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;

- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;
  - Seleção da operação da ULA através dos sinais de controle gerados na decodificação;

- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;
  - Seleção da operação da ULA através dos sinais de controle gerados na decodificação;
  - Carga de registradores de acordo com os sinais de controle gerados na decodificação;

- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;
  - Seleção da operação da ULA através dos sinais de controle gerados na decodificação;
  - Carga de registradores de acordo com os sinais de controle gerados na decodificação;
  - Escrita de operandos na memória;

- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;
  - Seleção da operação da ULA através dos sinais de controle gerados na decodificação;
  - Carga de registradores de acordo com os sinais de controle gerados na decodificação;
  - Escrita de operandos na memória;
  - Atualização do endereço do contador de programa (PC) para desvios tomados;

- Fase de Execução, dependendo do tipo de instrução, envolve:
  - Cálculo do endereço de operandos;
  - Busca de operandos na memória;
  - Seleção da operação da ULA através dos sinais de controle gerados na decodificação;
  - Carga de registradores de acordo com os sinais de controle gerados na decodificação;
  - Escrita de operandos na memória;
  - Atualização do endereço do contador de programa (PC) para desvios tomados;
  
- O **controle** das operações do ciclo de busca-decod-execução é de responsabilidade da **unidade de controle (UC)**, o programador não precisa preocupar-se com isso;



# Programação de um Processador

- A linguagem que o processador entende e executa é a **linguagem de máquina**;

- A linguagem que o processador entende e executa é a **linguagem de máquina**;
- Linguagem de máquina é uma imagem **binária** que representa a codificação do **conjunto de instruções** de um computador;

# Programação de um Processador

- A linguagem que o processador entende e executa é a **linguagem de máquina**;
- Linguagem de máquina é uma imagem **binária** que representa a codificação do **conjunto de instruções** de um computador;
- Para facilitar na programação, foram criados **mnemônicos** para os **códigos das instruções**, nomes aos operandos e rótulos às posições ocupadas pelo programa;

# Programação de um Processador

- A linguagem que o processador entende e executa é a **linguagem de máquina**;
- Linguagem de máquina é uma imagem **binária** que representa a codificação do **conjunto de instruções** de um computador;
- Para facilitar na programação, foram criados **mnemônicos** para os **códigos das instruções**, nomes aos operandos e rótulos às posições ocupadas pelo programa;
- **Não é mais necessário trabalhar com códigos numéricos !**

- Os símbolos precisam ser traduzidos para linguagem de máquina para serem executados;

- Os símbolos precisam ser traduzidos para linguagem de máquina para serem executados;
- A tradução recebe o nome de montagem, e quando automatizado, o programa que o realiza recebe o nome de montador;



- Os símbolos precisam ser traduzidos para linguagem de máquina para serem executados;
- A tradução recebe o nome de montagem, e quando automatizado, o programa que o realiza recebe o nome de montador;
- Exemplo:
  - add \$s1,\$s2,\$s3; (Mips)
  - ADD 130; (Neander)

- Como a unidade de controle sabe o momento exato que deve gerar os sinais de controle?

- Como a unidade de controle sabe o momento exato que deve gerar os sinais de controle?
- R: Através de um sinal periódico extremamente preciso chamado **clock**.
- O clock é um pulso alternado de sinais de tensão, gerado pelos circuitos de relógio (composto de um cristal oscilador e circuitos auxiliares);
- A velocidade de clock é um dos fatores que determinam o desempenho do computador ! (assunto abordado em detalhes em AOC I);

# Tópicos



- ~~Princípios Básicos;~~
- ~~Elementos funcionais básicos;~~
- **Computador de primeira geração: EDVAC;**
- **Modelo de Von Neumann: o computador IAS;**
- **Arquiteturas de 4,3,2,1 e 0 endereços;**
- **Resumo;**

- EDVAC (1945) era um computador de **programa armazenado** (programa e dados na mesma memória), ideia atribuída ao matemático **John Von Neumann**;

- EDVAC (1945) era um computador de **programa armazenado** (programa e dados na mesma memória), ideia atribuída ao matemático **John Von Neumann**;
- Memórias de capacidade aumentada (primária de 1024 words de 44 bits e a secundária de 20K words);



- EDVAC (1945) era um computador de **programa armazenado** (programa e dados na mesma memória), ideia atribuída ao matemático **John Von Neumann**;
- Memórias de capacidade aumentada (primária de 1024 words de 44 bits e a secundária de 20K words);
- Representação **interna** em **binário**;

- EDVAC (1945) era um computador de **programa armazenado** (programa e dados na mesma memória), ideia atribuída ao matemático **John Von Neumann**;
- Memórias de capacidade aumentada (primária de 1024 words de 44 bits e a secundária de 20K words);
- Representação **interna** em **binário**;
- Circuitos aritméticos binários seriais, devido à entrada de dados serial;

**Figura 8: Arquitetura de um computador de 1º geração. Fonte: Adaptado de Weber, 2001.**

➤ Todas instruções e dados eram colocados na memória principal antes de executar programas;

➤ Formato das instruções aritméticas:

A1	A2	A3	A4	OP
10 bits	10 bits	10 bits	10 bits	4 bits

➤ A1: endereço do primeiro operando;

➤ A2: endereço do segundo operando;

➤ A3: endereço do resultado;

➤ A4: endereço da próxima instrução;

➤ OP: operação aritmética a ser realizada;

- Todas instruções e dados eram colocados na memória principal antes de executar programas;

- Formato das instruções condicionais:

A1	A2	A3	A4	C
10 bits	10 bits	10 bits	10 bits	4 bits

- Se  $A1 > A2$ , executar a instrução do endereço A3, senão, executar a instrução do endereço em A4;
- C indica qual é o tipo de desvio;



- Todas instruções/dados eram colocados na memória principal antes de executar programas;

- Formato de instruções de E/S entre memórias:

A1	m,n	A3	A4	W
10 bits	10 bits	10 bits	10 bits	4 bits

- A1 representa a primeira posição da memória principal;
- m = 1 indica transferência para memória secundária;
- m = 2 indica transferência para memória principal;
- n representa o endereço de destino na memória secundária;
- A3 representa a última posição na memória principal;
- A4 representa qual endereço da próxima instrução;



# Desvantagens

- Muitas **interações** com a **memória** que não podiam ser feitas paralelamente;

- Muitas **interações** com a **memória** que não podiam ser feitas paralelamente;
- Isso gerava uma queda significativa no desempenho, problema conhecido por “**Gargalo de Von Neumann**”;

- Muitas **interações** com a **memória** que não podiam ser feitas paralelamente;
- Isso gerava uma queda significativa no desempenho, problema conhecido por “**Gargalo de Von Neumann**”;
- Os inconvenientes foram causados devido as escolhas do formato de instrução, bem como arquitetura de organização implementadas;

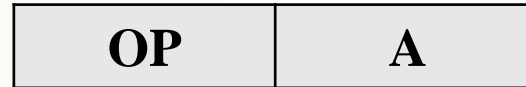
# Tópicos



- ~~Princípios Básicos;~~
- ~~Elementos funcionais básicos;~~
- ~~Computador de primeira geração: EDVAC;~~
- **Modelo de Von Neumann: o computador IAS;**
- Arquiteturas de 4,3,2,1 e 0 endereços;
- Resumo;

- Era possível o acesso a uma palavra inteira em uma operação;

- Era possível o acesso a uma palavra inteira em uma operação;
- Formato de instrução:





- Era possível o acesso a uma palavra inteira em uma operação;
- Formato de instrução: 

OP	A
----	---
- Soma de dois números: somar o conteúdo das posições 100 e 101 e armazenar o resultado na posição 102 de memória.

Instrução	Comentários
$AC \leftarrow M(100)$	Transfere conteúdo da posição 100 de MEM para o AC
$AC \leftarrow AC + M(101)$	Soma o conteúdo da posição 101 de MEM com o conteúdo do AC e armazena o resultado no AC
$M(102) \leftarrow AC$	Armazena o conteúdo do AC no endereço 102 de MEM

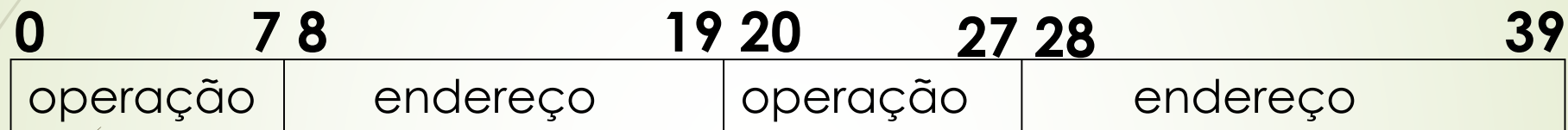
**Figura 9: Exemplo de programa no IAS. Fonte: Adaptado de Weber, 2001.**

**Figura 10: Estrutura do IAS. Fonte: Weber, 2001.**

- DR: Registrador de dados de 40 bits que recebe dados lidos da memória ou fornece dados que serão gravados na memória. Também usado para armazenamento temporário de dados;
- AR: Registrador de endereços de 12 bits que apontam para posições de memória;
- AC: registrador que armazena operandos temporariamente;
- MQ: Registrador multiplicador-quociente;
- IBR: instrução não-executada é armazenada no Reg. IBR;

- Memória principal: tubos de raios catódicos e de acesso aleatório;
- Memória com 4096 words de 40 bits cada;
- Podiam ser transferidos entre a UCP e a memória 40 bits por vez;
- Arquitetura de um endereço;
- Formato dos dados: binários em ponto fixo e complemento de dois;

- Formato das instruções: Uma palavra de memória poderia conter **2 instruções**;



- Alterações em relação ao EDVAC que permitiram a redução do tamanho das instruções:
- Há registradores pré-definidos na UCP para **armazenagem** de operandos e resultados;
- Instruções estão armazenadas na memória sequencialmente (**próxima instrução = PC + 1**);



- Conjunto de instruções do IAS: Instruções de transferência de dados entre registradores, desvios condicionais e incondicionais, aritméticas e de modificação de endereço;

Instruções aritméticas		
opcode	mnemônico	significado
00000101	ADD M(X)	$AC \leftarrow AC + \text{mem}(X)$
00000111	ADD  M(X)	$AC \leftarrow AC +  \text{mem}(X) $
00000110	SUB M(X)	$AC \leftarrow AC - \text{mem}(X)$
00001000	SUB  M(X)	$AC \leftarrow AC -  \text{mem}(X) $
00001011	MUL M(X)	$AC:MQ \leftarrow MQ * \text{mem}(X)$
00001100	DIV M(X)	$MQ:AC \leftarrow AC / \text{mem}(X)$
00010100	LSH	$AC \leftarrow AC * 2$ (shift esq.)
00010101	RSH	$AC \leftarrow AC / 2$ (shift dir.)

Figura 11: Algumas Instruções do IAS. Fonte: <http://slideplayer.com.br/slide/288038/>




# Tópicos



- ~~Princípios Básicos;~~
- ~~Elementos funcionais básicos;~~
- ~~Computador de primeira geração: EDVAC;~~
- ~~Modelo de Von Neumann: o computador IAS;~~
- **Arquiteturas de 4,3,2,1 e 0 endereços;**
- **Resumo;**

- Serão demonstradas arquiteturas de diversos endereços, utilizando como exemplo a equação matemática abaixo:


$$A = ( ( B + C ) * D + E - F ) / ( G * H )$$

Endereço	Instrução	Comentário $A = ((B + C) * D + E - F) / (G * H)$
e1	ADD B C A e2	Soma B com C, resultado em A, vai para e2
e2	MUL A D A e3	Multiplica A por D, resultado em A, vai para e3
e3	ADD A E A e4	Soma A com E, resultado em A, vai para e4
e4	SUB A F A e5	Subtrai F de A, resultado em A, vai para e5
e5	DIV A G A e6	Divide A por G, resultado em A, vai para e6
e6	DIV A H A e7	Divide A por H, resultado final em A, vai para e7
e7	HALT	Fim do programa

Figura 12: Programa em uma máquina de 4 endereços. Fonte: Weber, 2001.

Endereço	Instrução	Comentário $A = ((B + C) * D + E - F) / (G * H)$
e1	ADD B C A	Soma B com C, resultado em A, incrementa PC
e1+1	MUL A D A	Multiplica A por D, resultado em A, incrementa PC
e1+2	ADD A E A	Soma A com E, resultado em A, incrementa PC
e1+3	SUB A F A	Subtrai F de A, resultado em A, incrementa PC
e1+4	DIV A G A	Divide A por G, resultado em A, incrementa PC
e1+5	DIV A H A	Divide A por H, resultado final em A, incrementa PC
e1+6	HALT	Fim do programa

Figura 13: Programa em uma máquina de 3 endereços. Fonte: Weber, 2001.

Endereço	Instrução	Comentário $A = ((B + C) * D + E - F) / (G * H)$
e1	MOV A B	Move B para A
e1+1	ADD A C	Soma A com C, resultado em A
e1+1	MUL A D	Multiplica A por D, resultado em A
e1+2	ADD A E	Soma A com E, resultado em A
e1+3	SUB A F	Subtrai F de A, resultado em A
e1+4	DIV A G	Divide A por G, resultado em A
e1+5	DIV A H	Divide A por H, resultado final em A
e1+7	HALT	Fim do programa

Figura 14: Programa em uma máquina de 2 endereços. Fonte: Weber, 2001.

Endereço	Instrução	Comentário $A = ((B + C) * D + E - F) / (G * H)$
e1	LDA B	Move B para o acumulador ( AC )
e1+1	ADD C	Soma AC com C, resultado no AC
e1+2	MUL D	Multiplica AC por D, resultado em AC
e1+3	ADD E	Soma AC com E, resultado em AC
e1+4	SUB F	Subtrai F de AC, resultado em AC
e1+5	DIV G	Divide AC por G, resultado em AC
e1+6	DIV H	Divide AC por H, resultado em AC
e1+7	STA A	Armazena AC no endereço de A
e1+8	HALT	Fim do programa

Figura 15: Programa em uma máquina de 1 endereço. Fonte: Weber, 2001.



Endereço	Instrução	Comentário $A = ((B + C) * D + E - F) / (G * H)$
e1	PUSH H	Coloca H no topo (atual) da pilha
e1+1	PUSH G	Coloca G no topo da pilha
e1+2	PUSH F	Coloca F no topo da pilha
e1+3	PUSH E	Coloca E no topo da pilha
e1+4	PUSH D	Coloca D no topo da pilha
e1+5	PUSH C	Coloca C no topo da pilha
e1+6	PUSH B	Coloca B no topo da pilha
e1+7	ADD	Topo da pilha recebe (B+C)
E1+8	MUL	Topo da pilha recebe (B+C)*D
E1+9	ADD	Topo da pilha recebe (B+C)*D + E
E1+10	SUB	Topo da pilha recebe (B+C)*D + E - F
E1+11	DIV	Topo da pilha recebe ((B+C)*D + E - F)/G
E1+12	DIV	Topo da pilha recebe ((B+C)*D + E - F)/(G*H)
E1+13	POP A	Topo da pilha é armazenado em A
E1+14	HALT	Fim do programa

Figura 16: Programa em uma máquina de 0 endereços. Fonte: Weber, 2001.

# Tópicos



- ~~Princípios Básicos;~~
- ~~Elementos funcionais básicos;~~
- ~~Computador de primeira geração: EDVAC;~~
- ~~Modelo de Von Neumann: o computador IAS;~~
- ~~Arquiteturas de 4,3,2,1 e 0 endereços;~~
- **Resumo;**

# Resumo

- **Princípios básicos dos componentes que formam um computador;**
- **Elementos funcionais básicos;**
- **Exemplos de conjuntos de instruções e modos de endereçamento;**
- **Ciclo de busca – decodificação – execução de instruções;**
- **Computadores EDVAC e IAS;**
- **Arquiteturas de quantidades variadas de endereços;**



Dúvidas ?