Hardware e Manutenção de Computadores

Jariedson Dantas Maia

Jariedson Dantas Maia - Mini currículo

Graduado em redes de computadores IFRN - 2014

Pós-graduado em redes de computadores ESAB - 2016

Analista de telecomunicações Protele engenharia - Embratel - 2014 a 2016

Analista de TI - redes e infraestrutura IFPE - desde 2016

Coordenador de redes e infraestrutura da reitoria do IFPE - desde 2022

Bibliografia sugerida

- TORRES, Gabriel. Redes de Computadores - Versão Revisada e Atualizada. Novaterra, 1ª / 2009.
- MORIMOTO, Carlos E. Hardware II - O Guia Definitivo. Sul Editores, 1ª / 2010.

Hardware e Manutenção de Computadores

Jariedson Dantas Maia

Breve histórico

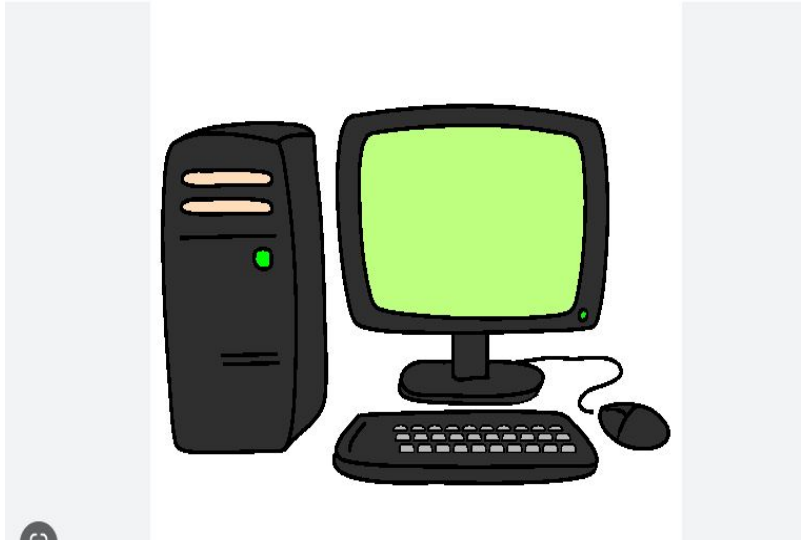
- **1847** - George Boole cria o sistema binário
- **1937** - Alan Turing cria a máquina universal, a máquina de Turing
- **1939** - é criada a máquina de Von Neuman
- **Anos 40** - ENIAC e EDIVAC
- **1954** - Texas Instruments lança o transistor de silício

Introdução

- **Computador digital**

Máquina que pode resolver problemas executando uma série de instruções que lhe são fornecidas

• Some dois números.
• Impr “positivo” se o resultado for > 0.
• Impr “negativo” se o resultado for < 0



- **Circuitos de um computador**

Reconhecem e executam um conjunto limitado e simples de instruções (linguagem de máquina-binária)

– Soma, comparação, transferência de dados de uma parte da memória para outra parte

Introdução

- **Instruções em linguagem de máquina**
  - Ser simples, compatíveis com o uso da máquina, compatíveis com o desempenho requerido, ter custo e complexidade da eletrônica reduzidos

Problema

- A linguagem de máquina está muito distante de uma linguagem natural.
- Complexidade do que precisa ser feito versus a simplicidade do conjunto de instruções do computador
- Ex: Calcular trajetória de um foguete à lua

Arquiteturas

Von Neuman

– Apresenta um barramento externo compartilhado entre dados e endereços
– Embora apresente baixo custo, esta arquitetura apresenta desempenho limitado pelo gargalo do barramento
– Modelo Refinado: UC, ULA, memória, e/s, registradores

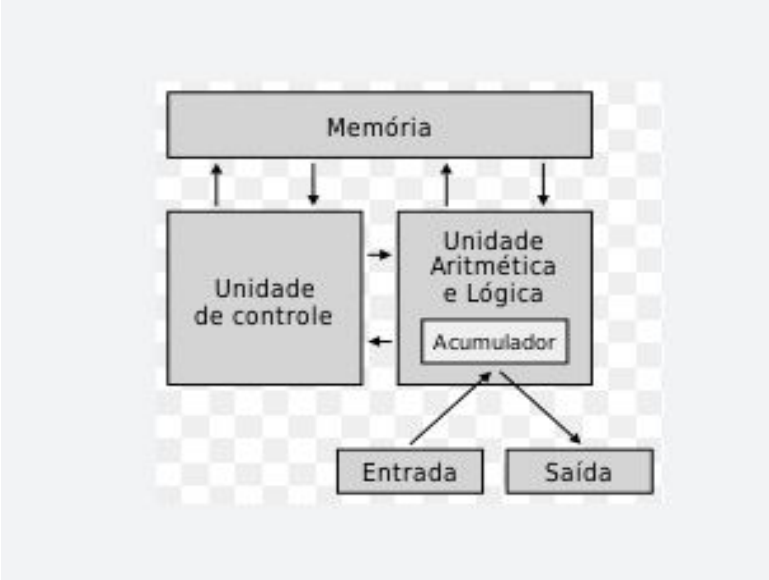
Harvard

– Existem dois barramentos externos independentes (e normalmente também memórias independentes) para dados e endereços
– Reduz de forma sensível o gargalo de barramento, que é uma das principais barreiras de desempenho, em detrimento do encarecimento do sistema como um todo

Von Neuman

Computadores são organizados em componentes ligados pelo barramento

- Processador (UC + ULA);
- Memória;
- Dispositivos de entrada e saída
- Registradores

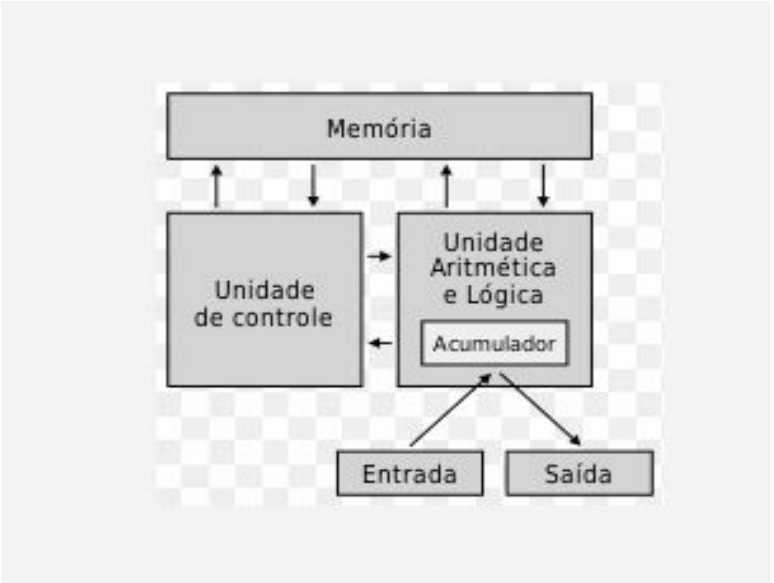


Von Neuman

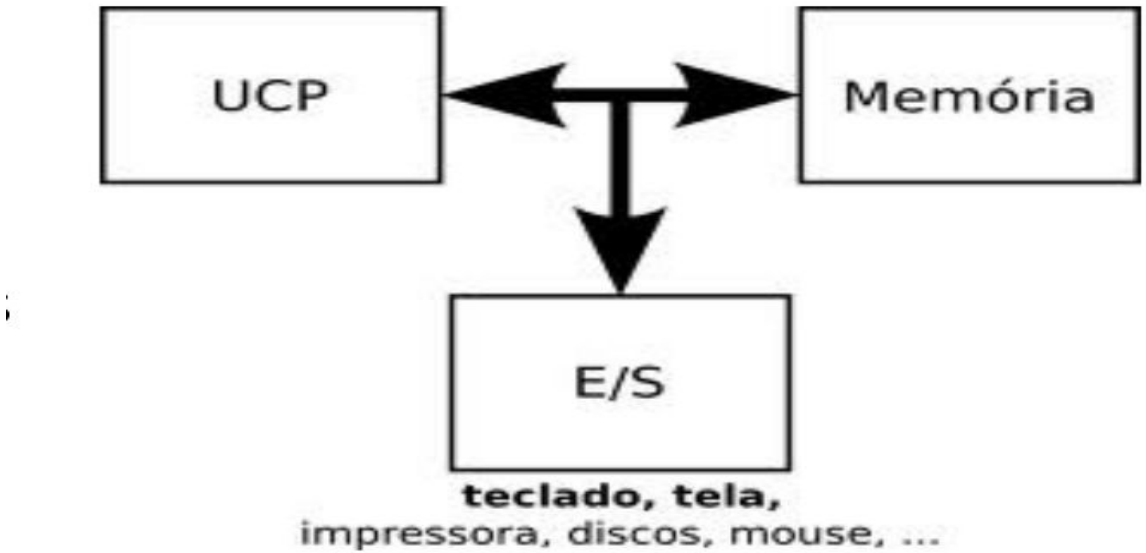
A memória armazena dados e instruções de programas.

• A CPU é encarregada de buscar as instruções e dados da memória, executar as instruções e então armazenar os valores resultantes de volta na memória.

• Os dispositivos de entrada e dispositivos de saída possibilitam a interação com o usuário



Von Neuman

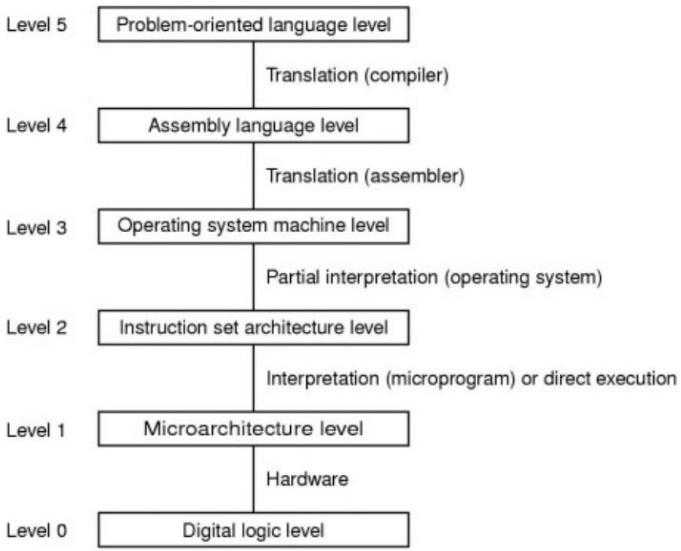


A máquina multinível

• Pode ser vista como tendo vários níveis, cada um capaz de executar um conjunto de instruções específicas

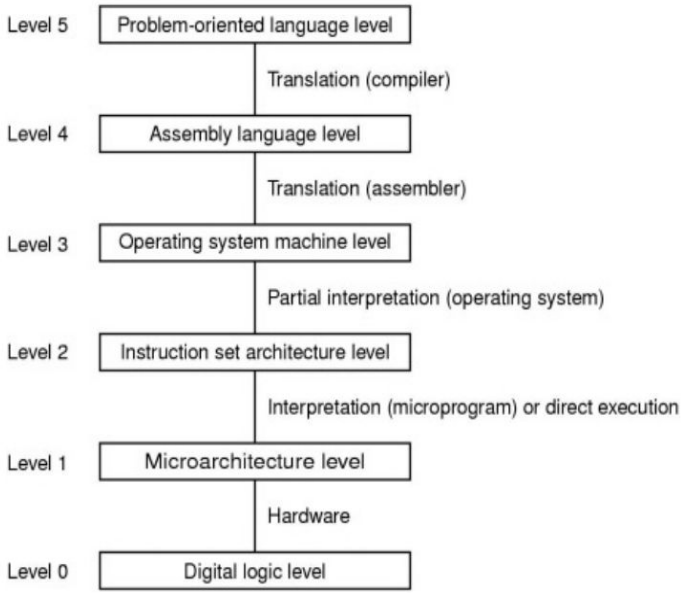
• Cada nível possui linguagens apropriadas para descrever as instruções que nele podem ser executadas

• A maioria dos computadores possui dois ou mais níveis



Modelo de 06 níveis

• **Nível 5:** Nível das linguagens orientadas para solução dos problemas
• **Nível 4:** Nível da linguagem do montador ou de montagem
(Assembly language)
• **Nível 3:** Nível do Sistema Operacional
• **Nível 2:** Nível da Arquitetura do Conjunto de Instruções
• **Nível 1:** Nível da Microarquitetura
• **Nível 0:** Nível da Lógica Digital



Conclusões

• Computadores são projetados como uma série de níveis, cada um deles construído em cima de seus precursores.

• Cada nível representa uma abstração distinta, com diferentes objetos e operações.

• A abstração permite ignorar detalhes irrelevantes de níveis mais baixos, reduzindo uma questão complexa a algo muito mais fácil de ser entendido

• Programador de um nível, em geral, não deve se preocupar com implementações de níveis inferiores

• Nos primeiros computadores a fronteira entre o hardware e o software era muito clara.

• Atualmente é muito difícil separar o hardware do software, pois hardware e software são equivalentes logicamente

Exercícios

Quantos barramentos há na arquitetura de Von Neuman, e na de Harvard, respectivamente?

a: 2 e 1

b: 3 e 2

c: 2 e 2

d: 1 e 2

Exercícios

Quantos barramentos há na arquitetura de Von Neuman, e na de Harvard, respectivamente?

a: 2 e 1

b: 3 e 2

c: 2 e 2

d: 1 e 2

Exercícios

No modelo de 06 níveis, é importante que cada nível esteja ciente de toda a complexidade que a camada imediatamente inferior possui?

a: Verdadeiro

b: Falso

Exercícios

No modelo de 06 níveis, é importante que cada nível esteja ciente de toda a complexidade que a camada imediatamente inferior possui?

a: Verdadeiro

b: Falso

Exercícios

A arquitetura de Von Neuman não possui o seguinte componente:

a. Memória
b. Unidade de controle
c. Unidade lógica e aritmética
d. Dispositivos de entrada e saída
e. Barramento decimal

Exercícios

A arquitetura de Von Neuman não possui o seguinte componente:

a. Memória
b. Unidade de controle
c. Unidade lógica e aritmética
d. Dispositivos de entrada e saída
e. Barramento decimal

Mecanismos de Conversões de linguagens

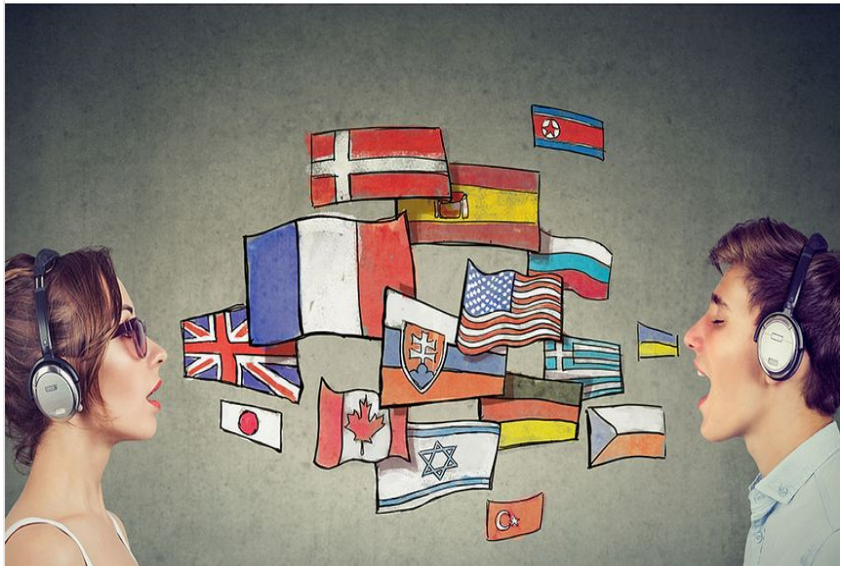
- Código fonte (macros)
– Pré-processador
- Código fonte (macros expandidas)
– Compilador
- Assembly
– Montador
- Objeto
– Linkeditor
- Executável

Pré processadores

• Tratamento pré-compilação
• Faz expansão de macros
• Dificuldade operacional - Manter o registro das linhas originais para que o compilador gere mensagens significativas
• Uso mais conhecido - Linguagem C
– Nem toda linguagem suporta pré-processamento

Compilação

• Cada instrução de L1 é substituída por um conjunto de instruções equivalentes de L0
• Processador executa programa em L0
• Todo programa em L0 é carregado em memória e é executado
• Programa pode ser traduzido uma única vez e executado várias vezes



Compiladores

- Recebem entrada em alto nível
- Traduzem todo código para execução posterior
- Conversão e a execução ocorrem em fases distintas
- Cada instrução do código fonte pode gerar várias instruções de máquina

Compiladores - fases

• Análise (divide o programa fonte nas partes constituintes e cria uma representação intermediária dos mesmos)

  – Léxica (tokens)

  – Sintática (árvore sintática)

  – Semântica (incoerências semânticas)

• Síntese (constrói o programa alvo desejado a partir da representação intermediária)

  – Geração de código intermediário

  – Otimização do código

  – Geração do código

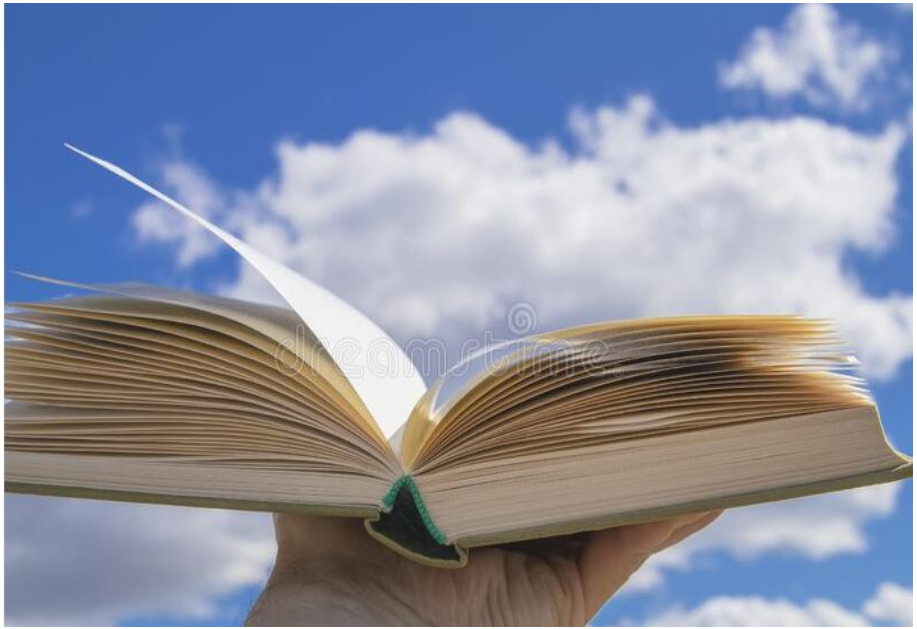
Interpretação

• Cada instrução de L1 é substituída por um conjunto de instruções equivalentes de L0

• Processador executa uma instrução de L1 (transformada para L0) antes de executar próxima instrução

• Cada instrução de L1, transformada para L0, é carregada na memória e executada

• Não é criado um programa em L0. Programa deve ser novamente interpretado para ser executado



Interpretadores

• Conversor on-line (incremental), onde a tradução e a execução das instruções ocorrem passo a passo, a cada instrução

• Execução simultânea à leitura, logo após a análise

• Recebem como entrada arquivos texto contendo programas em linguagem assembly, linguagem de alto nível, arquivos binários com instruções de máquina e os executam diretamente

• OBS: Processadores são interpretadores implementados em hardware

Interpretação Híbrida

- Mescla compilação com interpretação
- Programas fonte são traduzidos para uma linguagem intermediária que é interpretada
- Tem maior portabilidade que uma linguagem compilada
- São mais rápidas que uma linguagem interpretada
- Ex: Bytecode Java

Comparativo

Compilação

– Programas são traduzidos
para linguagem de máquina e são executados diretamente no computador
– Envolve dois processos distintos: Tradução (compilação) e Execução
– Não existe acesso ao programa fonte na execução

Interpretação

– O interpretador “executa” diretamente as instruções do programa fonte, sem traduzir para linguagem de máquina
– Execução mais lenta, devido ao passo de decodificação da instrução de alto nível
– tem acesso ao programa fonte, para depuração ou mesmo para alterar o código sendo executado

Montadores

- Tradução de uma linguagem de montagem (assembly) para código de máquina.
- Em geral, não pode ser executado diretamente pela máquina, por conter referências a sub-rotinas e dados especificados em outros arquivos
- Relação 1:1 com a linguagem de máquina
- É dependente da arquitetura da máquina
- Uso de comandos - Endereçamento simbólico
- Programação difícil - Esforço 5x maior Maior performance
    - 33% mais rápida
    - Solução crítica de sucesso

Ligadores

• Programa que liga objetos gerados por um compilador ou montador formando o executável

  – É ele quem gera o executável e não o compilador

• Recebem como entrada arquivos objetos e geram como saída o programa final em linguagem de máquina

• Gera um programa executável a partir de um ou mais arquivos objeto

• Resolução das chamadas de funções através da unificação dos objetos num único executável

Empacotadores

- Compacta um executável e gera outro executável auto extraível
- Packer muda a assinatura do executável
- Possibilita criação de vírus com assinaturas diferentes
- Comprimem, cifram e ofuscam o executável
- Dificultam a identificação do compilador usado por ferramentas adequadas
- Impossibilita a análise estática, pois o packer é quem se torna o ator principal
- Decifragem sob demanda

Carregadores

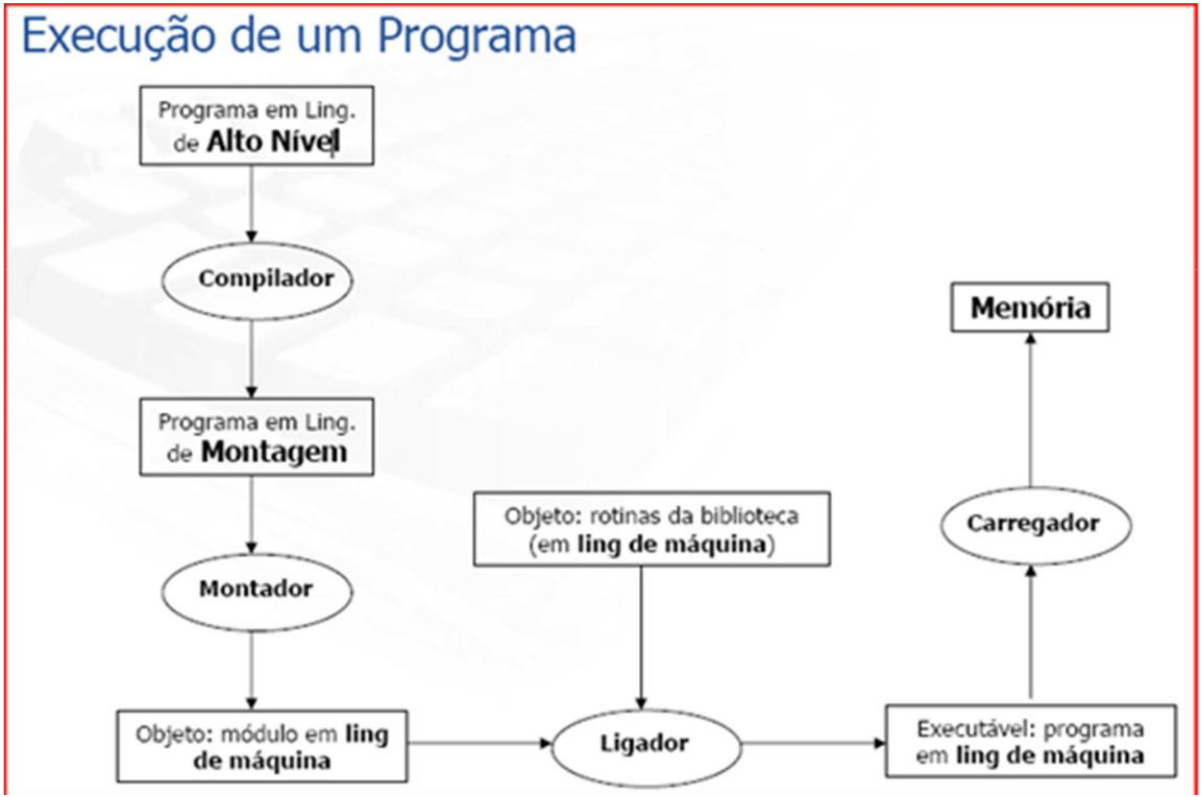
- Para executar um programa, um loader deve ser utilizado.
- O carregador é, em geral, parte do sistema operacional
- copia o arquivo em formato binário para a memória

Carregadores - Tipos

• **Binários / absolutos** - programa que usa carregadores absolutos é associado com localizações específicas de memória, e por isso deve sempre ser carregado na mesma área de memória (.COM)

• **Relocáveis** - O programa executável relocável é semelhante ao programa executável absoluto, exceto que os endereços são todos relativos a zero (não são absolutos) (.EXE)

Esquema de execução



Exercícios

Nos interpretadores, as instruções do código vão sendo executadas na medida em que são traduzidas?

a: verdadeiro

b: falso

Exercícios

Nos interpretadores, as instruções do código vão sendo executadas na medida em que são traduzidas?

a: verdadeiro

b: falso

Exercícios

Linguagens como python, c e c++ são exemplos de linguagens compiladas?

a: verdadeiro

b: falso

Exercícios

Linguagens como python, c e c++ são exemplos de linguagens compiladas?

a: verdadeiro

b: falso

FIM !!!