
UNIR

# Sistemas Operacionais

## Aula 04 – Conceitos de Hardware e Software – Continuação
## SCC5854

Prof. Dr. Jonathan Ramos
jonathan@unir.br

**Departamento Acadêmico de Ciências de Computação – DACC**

**Núcleo de Tecnologia – NT**

11/10/2022

# Sumário

# Dispositivos de Entrada e Saída – E/S

UNIR

**Permite a comunicação entre o sistema computacional e o mundo externo**

Podem ser **divididos em duas categorias**:

1. Os que são utilizados como **memória secundária**;
2. E os que servem para a **interface usuário-máquina**;

# Dispositivos de Entrada e Saída – E/S

UNIR

**Permite a comunicação entre o sistema computacional e o mundo externo**

Podem ser **divididos em duas categorias**:

1. Os que são utilizados como **memória secundária**;
2. E os que servem para a **interface usuário-máquina**;

## Usados como memória secundária:

- Caracterizam-se por ter **capacidade de armazenamento bastante superior ao da memória principal**;
- **Custo** relativamente **baixo**;
- **Tempo de acesso inferior** ao da memória principal;

# Dispositivos de Entrada e Saída – E/S

UNIR

**Permite a comunicação entre o sistema computacional e o mundo externo**

Podem ser **divididos em duas categorias**:

1. Os que são utilizados como **memória secundária**;
2. E os que servem para a **interface usuário-máquina**;

**Usados como memória secundária:**

- Caracterizam-se por ter **capacidade de armazenamento bastante superior ao da memória principal**;
- **Custo** relativamente **baixo**;
- **Tempo de acesso inferior** ao da memória principal;

**Usados como interface usuário-máquina:**

- Teclado, mouse, monitor, etc
- Interfaces amigáveis permitem que usuário necessitem de menos experiência para usar;
- Quanto mais intuitivo melhor;
- Dispositivos sensíveis a voz humana: tato, gesto etc

# Barramento

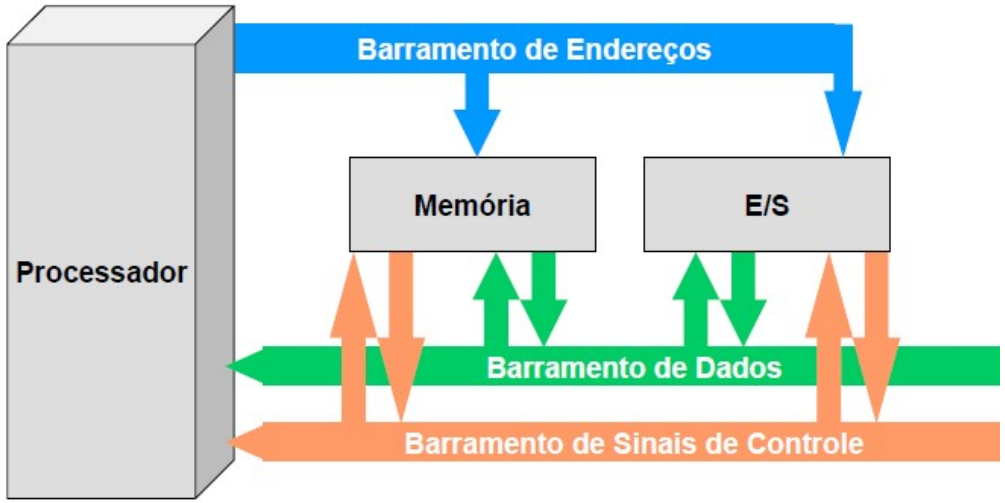



Figura: Barramentos: exemplo.

## Barramentos

- Meio de comunicação compartilhado;
- Permite a comunicação entre as unidades funcionais de um sistema computacional;
- Por meio deste que o processador se comunica com as demais partes do sistema computacional e vice-versa, como na Figura:
  - Memória RAM, HD, teclado, mouse, etc;

# Barramento

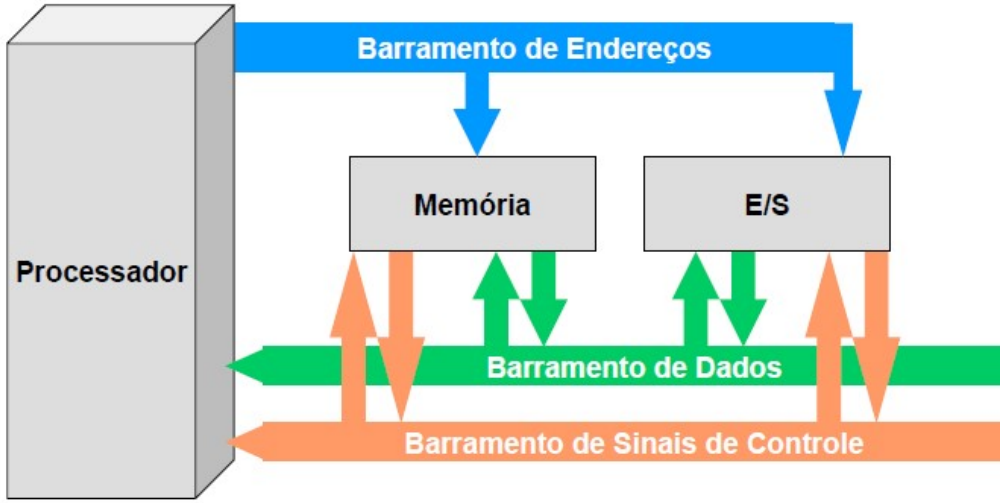



Figura: Barramentos: exemplo.

**Em geral possui:**

## Linha de controle

trafegam informações de sinalização como, por exemplo, o tipo de operação que está sendo realizada.

- Leitura, gravação, soma, divisão, etc

## Linha de dados

Nela são transferidos, entre as unidades funcionais, informações de instruções, operandos e endereços de memória.

# Barramento

## São divididos em três tipos principais:

1. Barramento Processador-memória: curta extensão e alta velocidade;
2. Barramentos de E/S: maior extensão, são mais lentos e permitem a conexão de diferentes dispositivos;
3. Adaptadores: Permite compatibilizar as diferentes velocidades dos barramentos.;

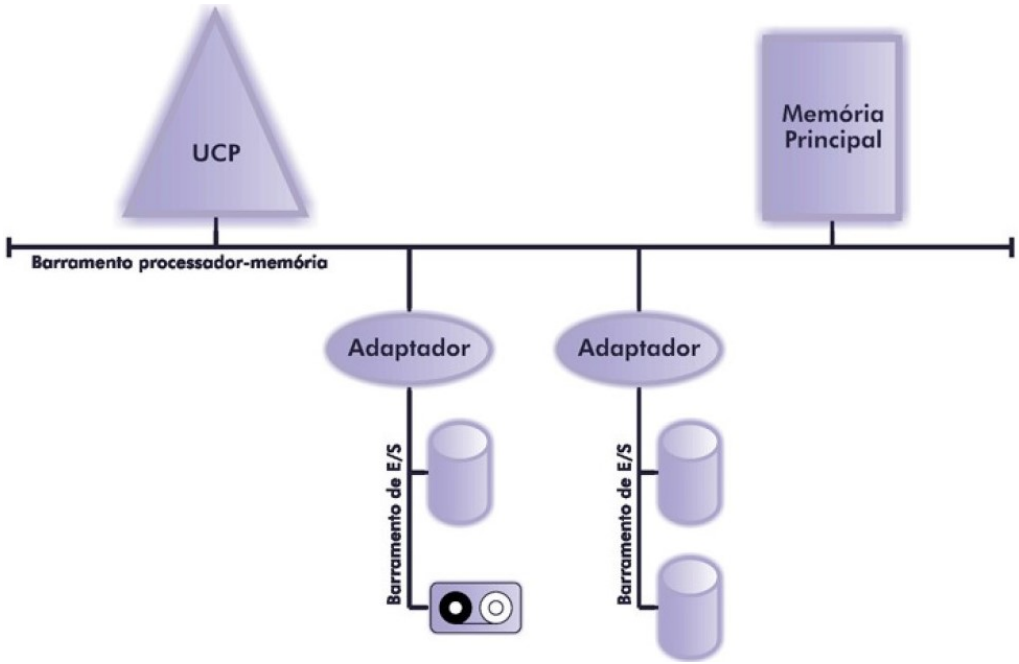



Figura: Barramentos processador-memória e de E/S.

# Barramento: *backplane*

## Tem a função de integrar os dois barramentos

- Reduz o número de adaptadores existentes no barramento processador-memória, otimizando o desempenho

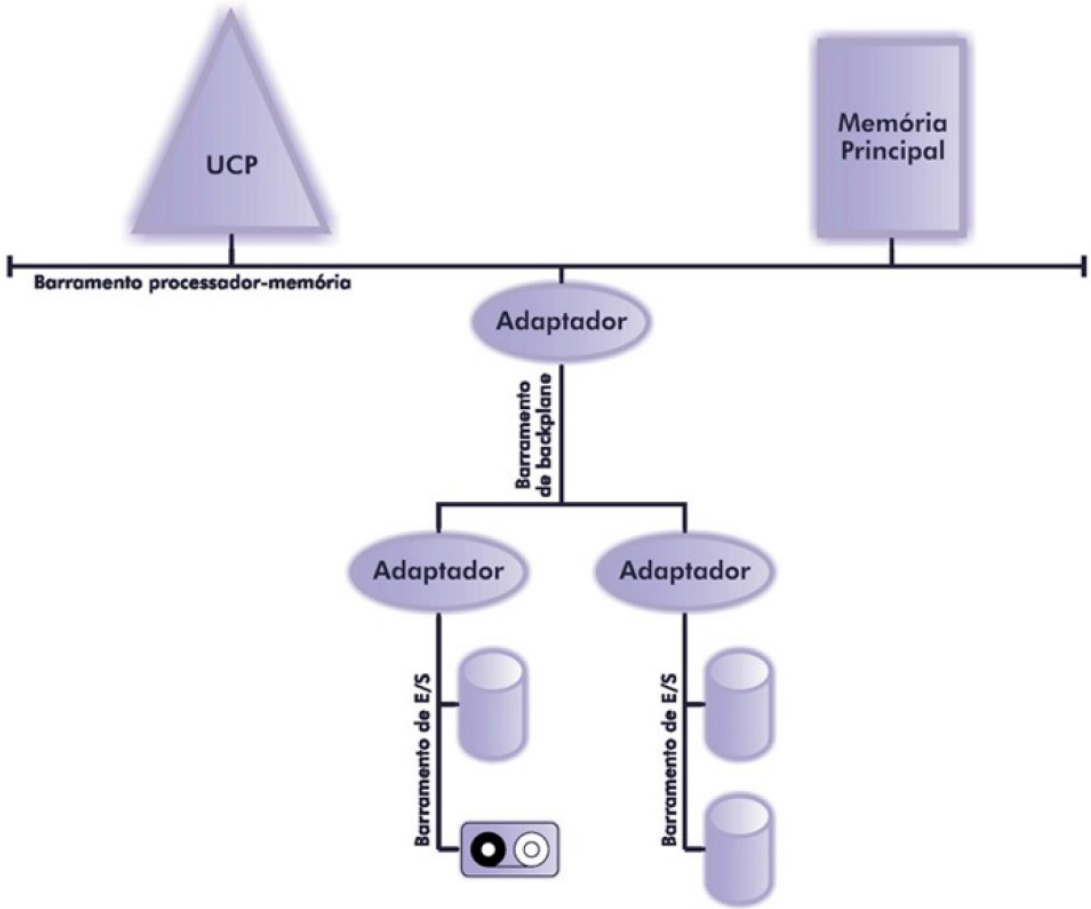
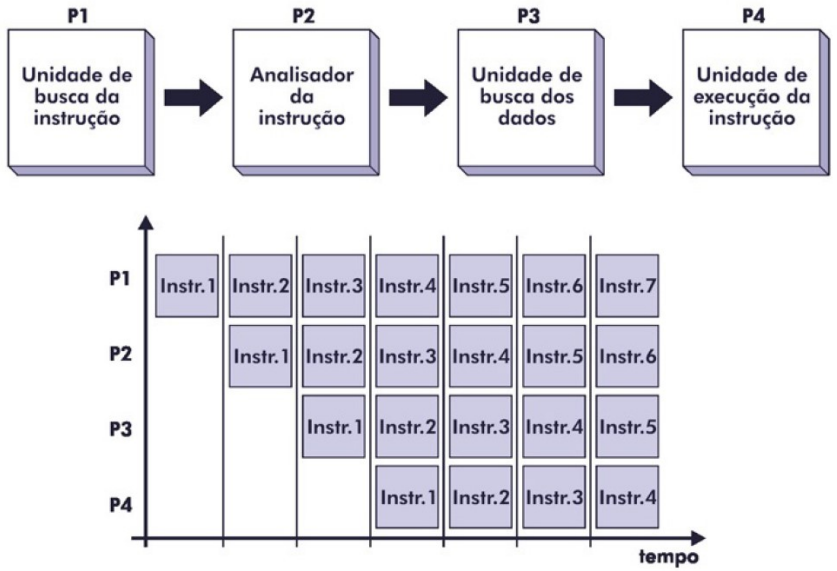



Figura: Barramento de *backplane*.

# *Pipelining*

**Permite ao processador executar múltiplas instruções paralelamente em estágios diferentes.**

- Linha de montagem: uma tarefa é dividida em uma sequência de subtarefas.
- A execução de uma instrução pode ser dividida em subtarefas:
  - fases de busca da instrução e dos operandos, execução e armazenamento dos resultados
  - Enquanto uma instrução se encontra na fase de execução, uma outra instrução possa estar na fase de busca simultaneamente



# *Pipelining*

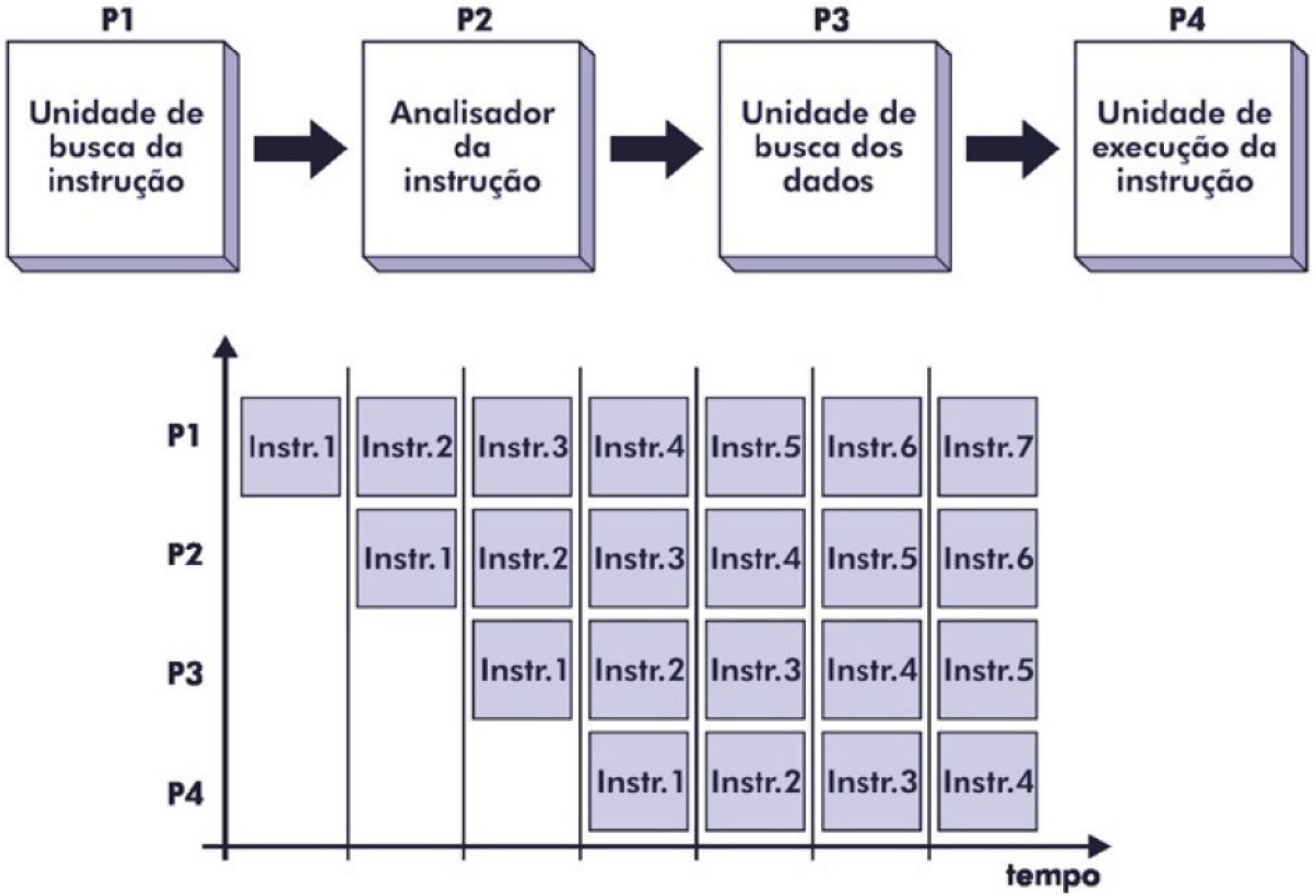



Figura: Arquitetura pipeline com quatro instruções em estágios diferentes.

# Arquiteturas RISC e CISC: *linguagem de máquina*

UNIR

**Linguagem realmente entendida pelo computador, zeros e uns:**

- Cada processador possui **um conjunto definido de instruções de máquina**;
- As instruções de máquina fazem referências a detalhes, como **registradores**, modos de **endereçamento** e **tipos de dados**;
- RISC: *Reduced Instruction Set Computer*;
- CISC: *Complex Instruction Set Computers*.

**Arquitetura RISC × Arquitetura CISC**:

## Arquitetura RISC

1. **Poucas** instruções;
2. Instruções executadas pelo **hardware**;
3. Instruções com **formato fixo**;
4. Instruções utilizam **poucos ciclos de máquina**;
5. Instruções com **poucos modos de endereçamento**;
6. Arquitetura com **muitos registradores**;
7. **Arquitetura** *pipelining*.

## Arquitetura CISC

1. **Muitas** instruções;
2. Instruções executadas por **microcódigo**;
3. Instruções com **diversos formatos**;
4. Instruções **utilizam múltiplos ciclos**;
5. Instruções com **diversos modos de endereçamento**;
6. Arquitetura com **poucos registradores**;
7. **Pouco uso** da técnica de *pipelining*.

# Arquiteturas RISC e CISC: *linguagem de máquina*

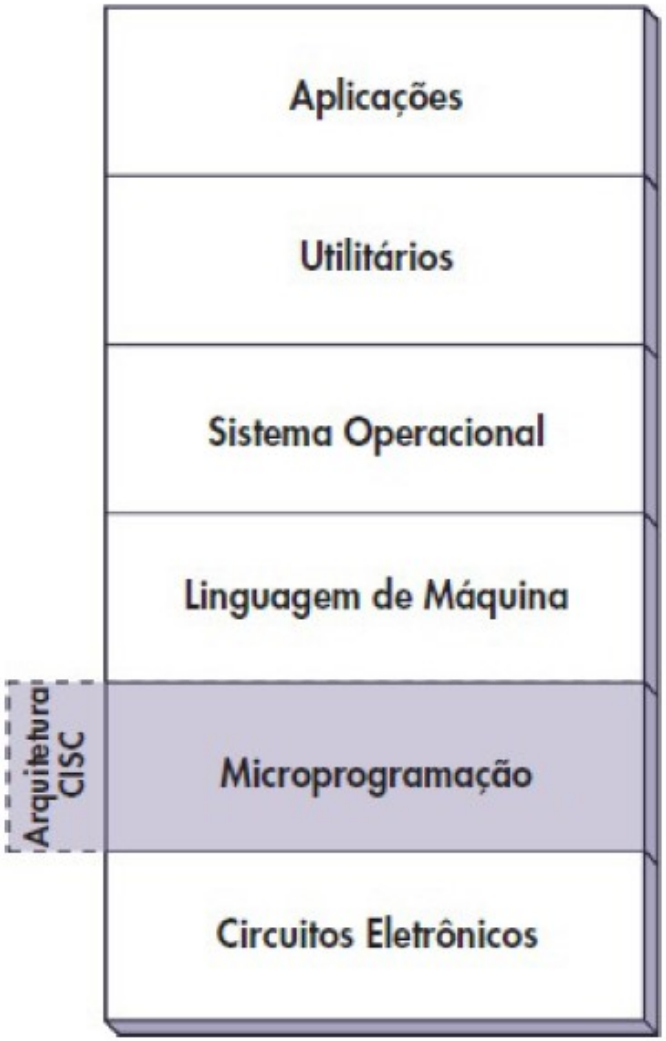



Figura: Máquina de níveis.

## Na RISC

- Um programa em linguagem de máquina é executado **diretamente pelo hardware**
- Não ocorre nos processadores CISC, como na Figura.
- Nível intermediário, **microprogramação** (CISC).
- **Microprogramas** definem a linguagem de máquina para CISC

# Software

**Interface entre as necessidade do usuário e as capacidades do hardware.**

## Torna o trabalho mais fácil...

Softwares são adequados para às **diversas tarefas e aplicações**, fazendo-os mais simples e eficientes.

# Software

**Interface entre as necessidade do usuário e as capacidades do hardware.**

## Torna o trabalho mais fácil...

Softwares são adequados para às **diversas tarefas e aplicações**, fazendo-os mais simples e eficientes.

**Existem dois tipos principais de software**

## Utilitários

São aqueles softwares relacionados mais diretamente com serviços complementares ao SO:

- Linkers;
- Depuradores.

# Software

**Interface entre as necessidade do usuário e as capacidades do hardware.**

## Torna o trabalho mais fácil...

Softwares são adequados para às **diversas tarefas e aplicações**, fazendo-os mais simples e eficientes.

**Existem dois tipos principais de software**

## Utilitários

São aqueles softwares relacionados mais diretamente com serviços complementares ao SO:

- Linkers;
- Depuradores.

## Software desenvolvidos pelo usuário

Chamados de aplicativos ou aplicações.

# Tradutor

**Apesar das inúmeras vantagens proporcionadas pelas linguagens de montagem e de alto nível, os programas escritos nessas linguagens não estão prontos para ser diretamente executados pelo processador (programas-fonte).**

## Etapa de conversão

Toda representação simbólica das instruções é traduzida para código de máquina. Esta conversão é realizada por um utilitário denominado tradutor.

## Programa objeto

Ainda não pode ser executado: dependência com sub-rotinas externas.

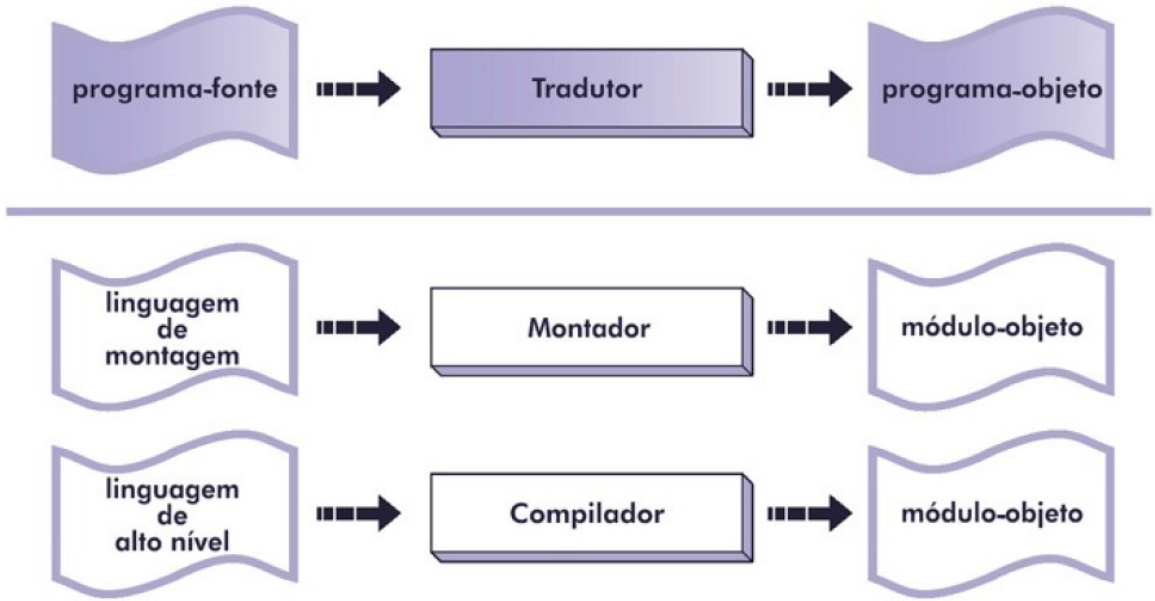



Figura: Tradutor.

# Interpretador

UNIR

## Interpretador

- Um tradutor que não gera módulo-objeto;
- Durante a execução do programa, traduz cada instrução e a executa imediatamente;
- Não gera código executável

## Desvantagem

- Tempo gasto na tradução das instruções.
- Feito toda vez que o programa é executado.

## Vantagem

- Flexibilidade: Tipos de dados dinâmicos.
- Mudam de tipo durante a execução do programa.

# Linker

**Gera um único código executável a partir de um ou mais códigos-objeto**

## Linker

- Resolve as referências simbólicas entre módulos e aloca memória.
- Pesquisa bibliotecas do sistema (diversos módulos-objeto)

## Relocação de memória

- Feita também pelo linker.
- Porém, mais complexa de ser feita em sistemas multi-programáveis (memória compartilhada)
- Código realocável: diferentes regiões de memória toda vez que for rodar.
- Solução: loader.

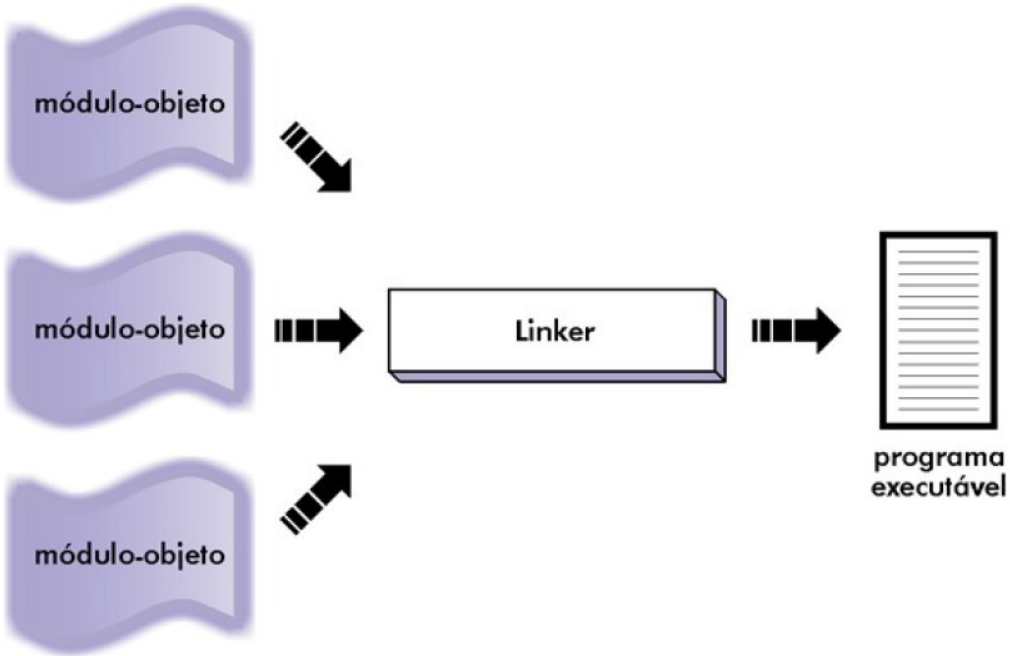


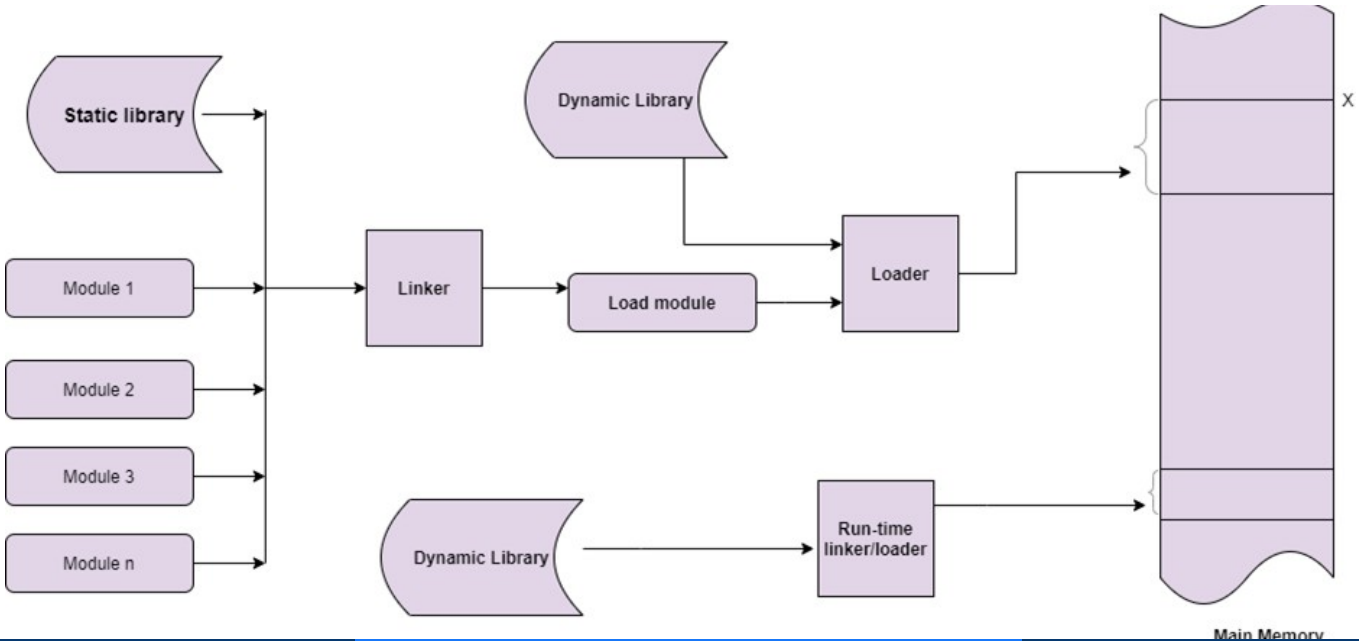

Figura: Linker.

# Loader

## Carrega na memória um programa para ser executado

### Depende do código gerado pelo linker

- Absoluto: o loader só necessita conhecer o endereço de memória inicial e o tamanho do módulo para realizar o carregamento:
  - o loader transfere o programa da memória secundária para a memória principal e inicia sua execução.
- Relocável: o programa pode ser carregado em qualquer posição de memória
  - o loader é responsável pela relocação no momento do carregamento=.



# Depurador (Debugger)

**Verifica possíveis erros lógicos no código:**

- Acompanha a execução de um programa instrução por instrução;
- Possibilita a alteração e a visualização do conteúdo de variáveis;
- Implementa pontos de parada dentro do programa (*breakpoint*), de forma que, durante a execução, o programa pare nesses pontos;
- Especificar que, toda vez que o conteúdo de uma variável for modificado, o programa envie uma mensagem (*watchpoint*).

<https://www.javatpoint.com/flow-of-c-program>

# Exercícios

1. Diferencie as funções básicas dos dispositivos de E/S.
2. Caracterize os barramentos processador-memória, E/S e *backplane*.
3. Como a técnica de *pipelining* melhora o desempenho dos sistemas computacionais?
4. Compare as arquiteturas de processadores RISC e CISC.
5. Por que o código-objeto gerado pelo tradutor ainda não pode ser executado?
6. Por que a execução de programas interpretados é mais lenta que a de programas compilados?
7. Quais as funções do *linker*?
8. Qual a principal função do *loader*?
9. Quais as facilidades oferecidas pelo depurador?
10. Na sua opinião, quais as vantagens que uma linguagem de alto nível permite ao programador?

UNIR

# FIM!

jonathan@unir.br