Aula 02: Conceitos e linguagens

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2019/1
Departamento de Computação – UFOP

Aula Anterior

- Apresentação da disciplina
  - Objetivos
  - Ementa
  - Avaliações
  - Bibliografia

Aula de Hoje

1. O que é um computador?
2. Organização de Computadores
3. Programação e Hierarquia de Dados
4. Linguagens de Programação
5. Primeiro programa

Aula de Hoje

O que é um computador?

- Um computador é um dispositivo capaz de realizar computações e tomar decisões lógicas milhões (e até bilhões) de vezes mais rapidamente que o homem. Muitos computadores realizam um bilhão de adições por segundo!
- Uma pessoa operando uma calculadora de mesa pode gastar a vida toda para fazer cálculos e ainda assim não concluir a mesma quantidade de cálculos que um poderoso computador pessoal (PC) pode realizar em um segundo.

O que é um computador?

Questões para ponderar:

- Como saber se a pessoa somou os números corretamente?
- Como saber se o computador somou os números corretamente?

O que é um computador?

- Os computadores processam **dados** sob o controle de conjuntos de instruções chamados **programas de computador**.
- Esses programas orientam o computador por meio de conjuntos ordenados de **ações** especificadas por pessoas chamadas **programadores de computador**.

O que é um computador?

- Um computador consiste de vários dispositivos referidos como **hardware**:
  - unidades de processamento
  - teclado
  - mouse
  - monitor
  - memória (RAM / SSD)
  - etc
- Os programas executados em um computador são referidos como **software**.

Aula de Hoje

1. O que é um computador?
2. Organização de Computadores
3. Programação e Hierarquia de Dados
4. Linguagens de Programação
5. Primeiro programa

Organização do Computador

Praticamente, todos os computadores podem ser considerados como divididos em seis unidades lógicas:

- Entrada
- Saída
- Armazenamento Principal
- Armazenamento Secundário
- Unidade de Aritmética e Lógica
- Unidade Central de Processamento

Organização do Computador

- **Unidade de Entrada**
  - Esta é a seção “receptora” do computador.
  - Ela obtém informações (dados e programas de computador) de **dispositivos de entrada** e coloca essas informações à disposição das outras unidades para o processamento.
  - A maioria das informações é inserida em computadores por meio de dispositivos de entrada, como teclados, *touch screens* e mouse.

Organização do Computador

- **Unidade de Entrada**
  Outras formas de entrada:
  - falar com seu computador, digitalizar imagens e códigos de barra;
  - leitura de dispositivos de memória secundária (HD, DVD, *Blu-ray Disc*, USB *Flash drives*);
  - receber um vídeo de uma *webcam* ou receber informações de uma rede, como a Internet;
  - obter dados de posicionamento a partir de um dispositivo de GPS;
  - coletar informações de movimento e orientação a partir de um acelerômetro em um *smartphone*;
  - etc

Organização do Computador

- **Unidade de Saída**
  - Esta é a seção de “envio” do computador.
  - A maioria das informações enviadas para a saída de computadores é exibida em telas, impressas em papel ou utilizadas para controlar outros dispositivos.
  - Os computadores também podem gerar saída de suas informações para redes, como a Internet, entre outros.

Organização do Computador

- **Unidade de Armazenamento Principal**
  - Esta é a seção de armazenamento de relativamente **baixa capacidade** e **rápido acesso** do computador.
  - Armazena os programas de computador enquanto estão sendo executados. Retém informações que foram inseridas pela unidade de entrada, para se tornarem imediatamente disponíveis para o processamento quando for necessário.
  - As informações na unidade de memória são **voláteis**, são perdidas quando o computador é desligado. A unidade de memória costuma ser chamada de memória ou memória principal.

Organização do Computador

- **Unidade de Armazenamento Secundária**
  - Esta é a seção de armazenamento de alta capacidade e longo prazo do computador (**acesso lento**).
  - As informações no armazenamento secundário são **persistentes**; preservadas quando o computador é desligado. Em geral exigem muito mais tempo para serem acessadas do que as informações na memória principal, mas o custo por unidade de armazenamento secundário é muito menor.
  - Exemplo: unidades de disco, CDs, DVDs, HDs, etc.

Organização do Computador

- **Unidade de Aritmética e Lógica**
  (ALU – *Arithmetic and Logic Unit*)
  - Esta é a seção de “produção” do computador.
  - Ela é responsável pela realização de cálculos, como adição, subtração, multiplicação e divisão.
  - Contém os mecanismos de decisão que permitem ao computador, por exemplo, comparar dois itens da unidade de memória para determinar se são iguais ou não.
  - Nos sistemas atuais, a ALU é usualmente implementada como uma parte da CPU.

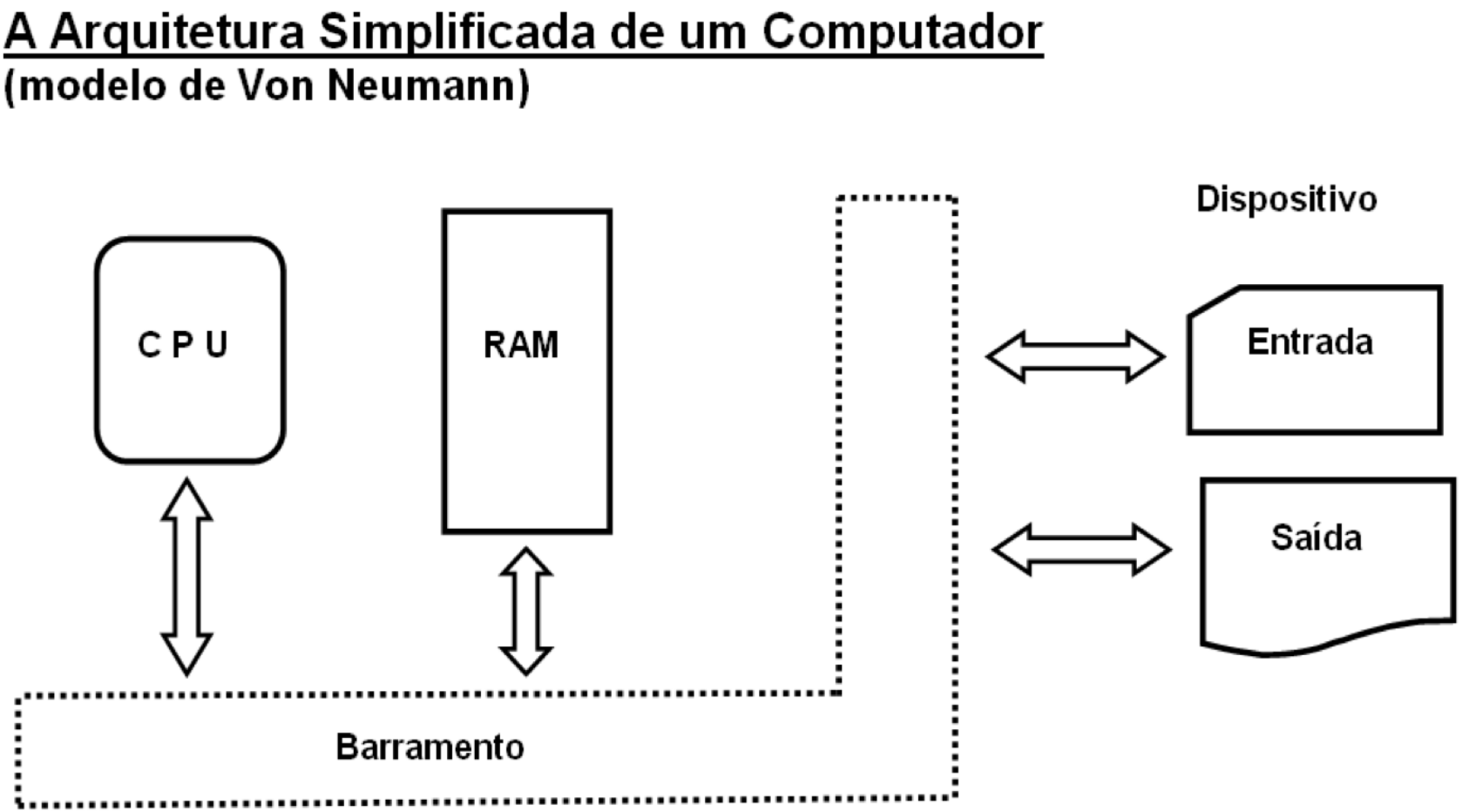
Organização do Computador

- **Unidade Central de Processamento**
  (CPU – *Central Processing Unit*)
  - Esta é a seção “administrativa” do computador. Ela coordena e supervisiona a operação das outras seções.
  - A CPU diz à unidade de entrada quando as informações devem se lidas e transferidas para a unidade de memória, informa à ALU quando as informações da unidade de memória devem ser utilizadas em cálculos e instrui a unidade de saída sobre quando enviar as informações da unidade de memória para certos dispositivos de saída.

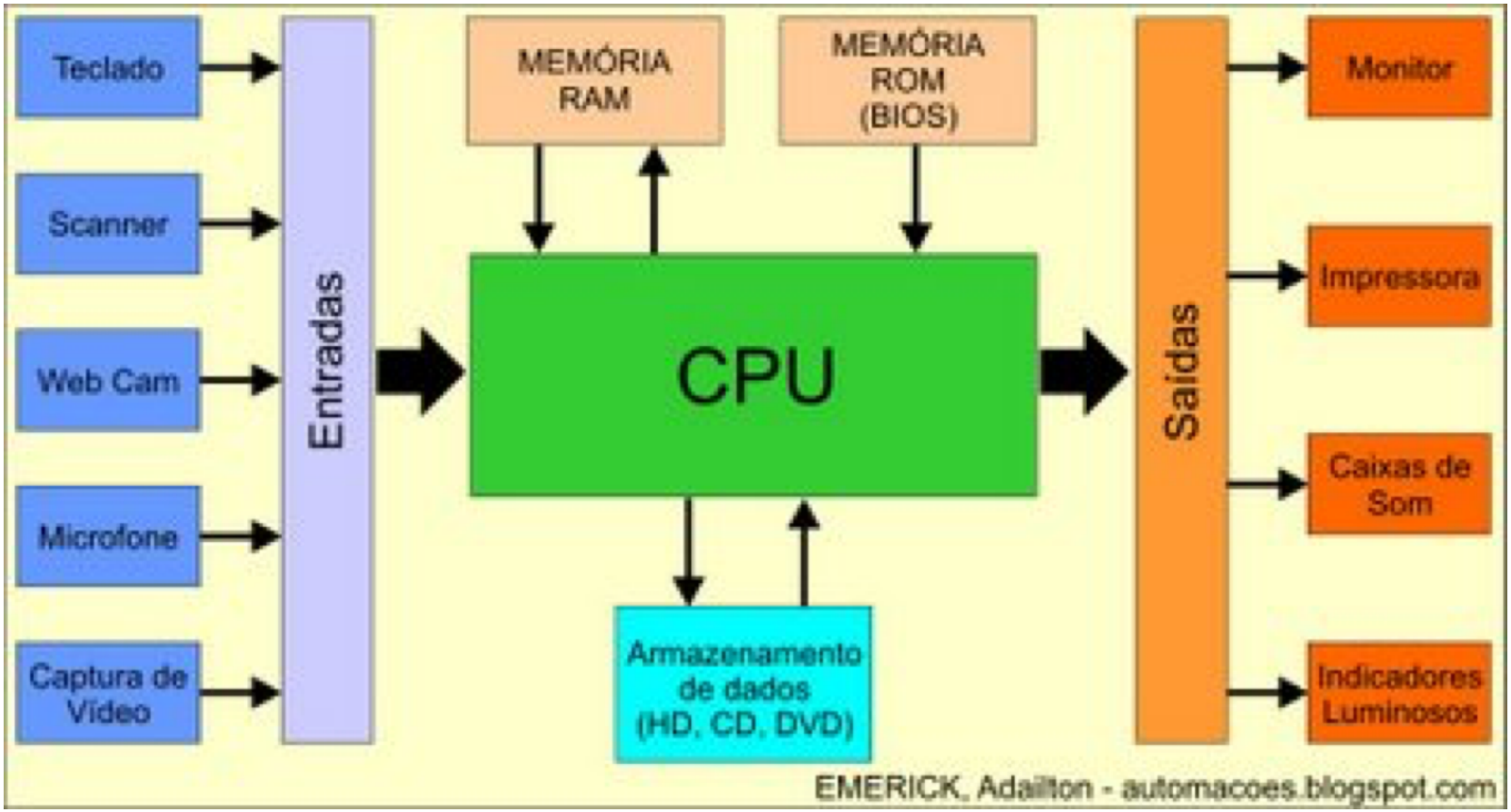
Organização do Computador

- **Unidade Central de Processamento**
  (CPU – *Central Processing Unit*)
  - Muitos computadores de hoje têm múltiplas CPUs e, portanto, podem realizar muitas operações simultaneamente - esses computadores são chamados de multiprocessados.
  - um processador *multi-core* implementa múltiplos processadores em um único chip de circuito integrado; um processador *dual-core* possui duas CPUs e um *quad-core* possui quatro CPUs.
  - Um computador *desktop* atual possui processadores que podem executar biliões de instruções por segundo.

Organização do Computador



Organização do Computador



Aula de Hoje

Programa e Programação

- **Programa**: geralmente referido como *software* (instruções escritas para que o computador realize ações e tome decisões).
- **Programação Estruturada**: metodologia de programação constituída por sequências, desvios e repetições de instruções de uma linguagem de programação.

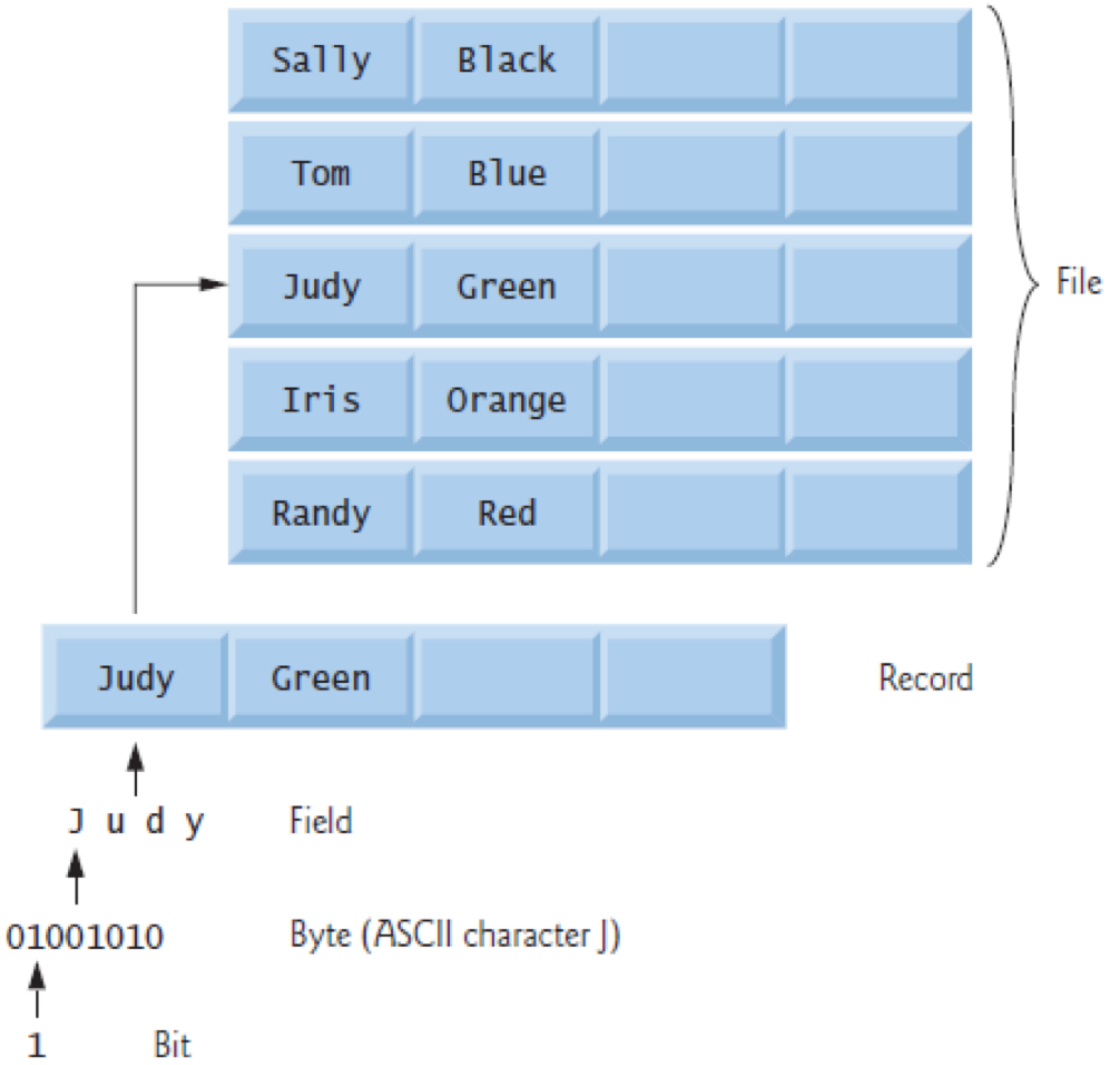
Programa e Programação

- C e C++ são duas das linguagens mais populares para o desenvolvimento de software.
- Deitel: usa a versão do C/C++ padronizado nos EUA através do ANSI - *American National Standards Institute* e mundialmente por meio dos esforços da ISO - *International Organization for Standardization*

Hierarquia de Dados

- Lembre-se que um computador digital utiliza bits (dados binário: 0 ou 1).
- Assim, os dados são processados por computadores a partir de uma hierarquia de dados que se torna maior e mais complexa em estrutura, a medida que avançamos de bits para caracteres, depois para campos, e assim por diante.

Hierarquia de Dados



Hierarquia de Dados

- **Bit**
  - É o menor item de dado em um computador, o qual pode assumir o valor **0** ou **1** **B**inary Dig**it**.
  - É notável como funções importantes de hardware realizam computações por meio da simples mudança do bit 0 para o bit 1, e vice-versa.
- **Byte**
  - Conjunto de 8 bits.

Hierarquia de Dados

- **Caractere**
  - É tedioso para o ser humano trabalhar com dados no formato de baixo nível (bit).
  - Preferência: dígitos decimais (0-9), letras (A-Z e a-z), símbolos especiais ($, @, %, #, &, *, (,), +, etc).
  - Caracteres: dígitos, letras e símbolos especiais.

Hierarquia de Dados

- **Caractere**
  - Conjunto de Caracteres do Computador: é o conjunto de todos os caracteres usados para escrever os programas e os itens de dados.
  - Os computadores processam somente 1s e 0s, assim o conjunto de caracteres do computador representa todo caractere com um padrão de 1s e 0s.
  - C/C++ usa a codificação ASCII – *American Standard Code for Information Interchange*.

Hierarquia de Dados

- **Campo – *Field***
  - Da mesma forma que os caracteres são formados por bits, os campos são compostos por caracteres.
  - Um campo é um grupo de caracteres, ou bytes, que transmite um significado.
  - Por exemplo, um campo constituído por letras maiúsculas e minúsculas pode ser usado para representar o nome de uma pessoa, e um campo constituído por dígitos decimais poderia apresentar a idade de uma pessoa.

Hierarquia de Dados

- **Registro – *Record***
  - Um conjunto de campos inter-relacionados compõe um registro.
  - Por exemplo, um registro de um empregado poderia conter:
    - Identificação (um número decimal)
    - Nome (uma cadeia de caracteres ? campo)
    - Ano de admissão (cadeia de caracteres numéricos)

Hierarquia de Dados

- **Arquivo – *File***
  - Um arquivo é um grupo de registros.
  - Genericamente, um arquivo contém dados arbitrários em formatos arbitrários.
  - Em alguns sistemas operacionais, um arquivo é simplesmente visto com uma sequência de bytes.
  - Uma organização de bytes em um arquivo, tal como uma lista de registros, é uma visão criada pela aplicação do programador.

Unidades de Armazenamento

Prefixo	Abreviação	Valor Geral	Valor em Ciência da Computação	Exemplo de Uso
Kilo	k	1.000	1.024	256 kilobytes
Mega	M	1.000.000	1.048.576	512 megabytes
Giga	G	1.000.000.000	1.073.741.824	1 gigabytes
Tera	T	1.000.000.000.000	1.099.511.627.776	

- 1 Megabyte = 1.024 * 1 kilobyte
- 1 Gigabyte = 1.024 * 1 Megabyte
- 1 Terabyte = 1.024 * 1 Gigabyte

Aula de Hoje

Linguagens de Programação – LP

- Os programadores escrevem seus programas em várias LPs, algumas entendidas diretamente pelos computadores, outras requerendo passos intermediários de tradução.
- As LPs são divididas em três tipos gerais:
  1. Linguagens de Máquina
  2. Linguagens Assembly
  3. Linguagens de Alto-nível

Linguagens de Programação

- Qualquer computador entende diretamente sua própria **linguagem de máquina**, definida por seu **projeto de *hardware***.
- São geralmente são constituídas por *strings* de números (1s e 0s), as quais instruem ao computador como realizar as operações mais elementares, uma de cada vez.
- São **dependentes de máquina**: uma linguagem de máquina particular somente pode ser usada em um único tipo de computador.

Linguagens de Máquina

- São incômodas para o ser humano.
- Por exemplo, seja uma seção de um programa que soma o pagamento da hora extra ao salário base, e a seguir, armazena o resultado no salário bruto:

  11000101 10010001
  10100101 10111010
  11100111 10011110

- São **dependentes de máquina**: uma linguagem de máquina particular somente pode ser usada em um único tipo de computador.

Linguagens de Assembly

- Substituir as ***strings* de números** (entendidas diretamente pelo computador) por **abreviações similares ao inglês** que representavam as operações elementares (mnemônicos).
- Esses mnemônicos formaram as bases das Linguagens Assembly.
- **Assembler**: é um programa tradutor para converter os programas em **linguagem assembly** para **programas em linguagem** de máquina (na velocidade do processamento de um computador).

Linguagens de Assembly

- Traduzindo o programa:

Linguagem de Máquina		Linguagem Assembly
11000101 10010001		`load salarioBase`
10100101 10111010	$\Longleftarrow$	`add horaExtra`
11100111 10011110	tradução	`store salarioBruto`

- Embora esse código seja mais claro para o ser humano, ele é incompreensível para os computadores (até que seja traduzido para linguagem de máquina.

Linguagens de Alto-nível

- A utilização dos softwares aumentou com o surgimento das linguagens assembly.
- Porém, os programadores escreviam muitas linhas de código, mesmo para tarefas simples!
- Linguagem de Alto-nível: desenvolvidas para acelerar o processo de criação de softwares, onde as instruções para o computador realizavam tarefas mais complexas.
- Os programas tradutores, chamados **compiladores**, traduziam programas em linguagem de alto-nível para programas em linguagem de máquina.

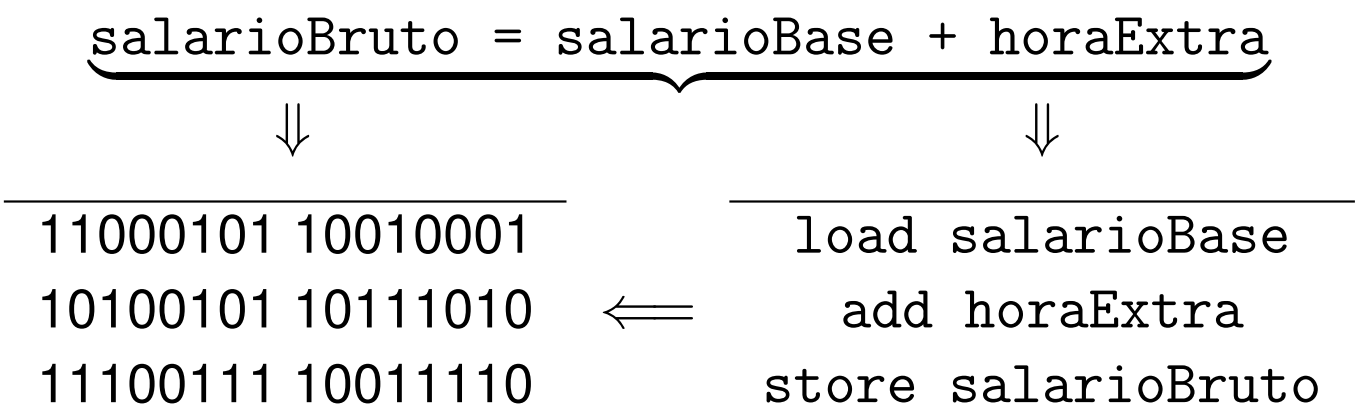
Linguagens de Alto-nível

- As instruções em alto-nível parecem com palavras cotidianas do inglês e contém comumente notações matemáticas.
- A instrução em alto-nivel representa o programa descrito anteriormente:

```
salarioBruto = salarioBase + horaExtra
```

Linguagens de Alto-nível

- Tradução:



As linguagens C/C++

- C++ foi uma evolução a partir de C, linguagem desenvolvida por Dennis Ritchie na Bell Laboratories.
- C é disponível para a maioria dos computadores e é independente de *hardware*.
- Mediante um projeto criterioso, é possível escrever programas C **portáveis** para a maioria dos computadores.
- Problema: o uso difundido de C em vários tipos de computadores (as vezes chamados de plataforma de hardware) levou a muitas variações da linguagem.
- Necessidade: uma versão padronizada de C.

As linguagens C/C++

- ANSI – *American National Standards Institute*, em coorperação com a ISO – *International Organization for Standardization*, criaram o documento de padronização da linguagem C, publicado em 1990, referido como ANSI/ISO 9899:1990.
- C99 é o último padrão ANSI para a linguagem C.
- C++, uma extensão de C, foi desenvolvida por Bjarne Stroustrup no início dos anos 802, no Bell Laboratories.
- C11, é o padrão atual da linguagem C++ (já existe o C14, porém ele é, basicamente, uma pequena extensão do padrão C11).

As linguagens C/C++

- C++ provê várias características que “aprimoraram e renovaram” a linguagem C, mas o mais importante, foi o provimento de capacidades para programação orientada por objetos.
- Os programas C++ consistem de “pedaços” chamados classes e funções.
- Podemos escrever nossas próprias classes ou funções. Entretanto podemos reutilizar as que já estão prontas; contidas na *C++ Standard Library*.
- Duas partes no aprendizado de C++: 1) a própria linguagem C++; e 2) como usar as classes e funções da biblioteca padrão.

Um ambiente típico de desenvolvimento C/C++

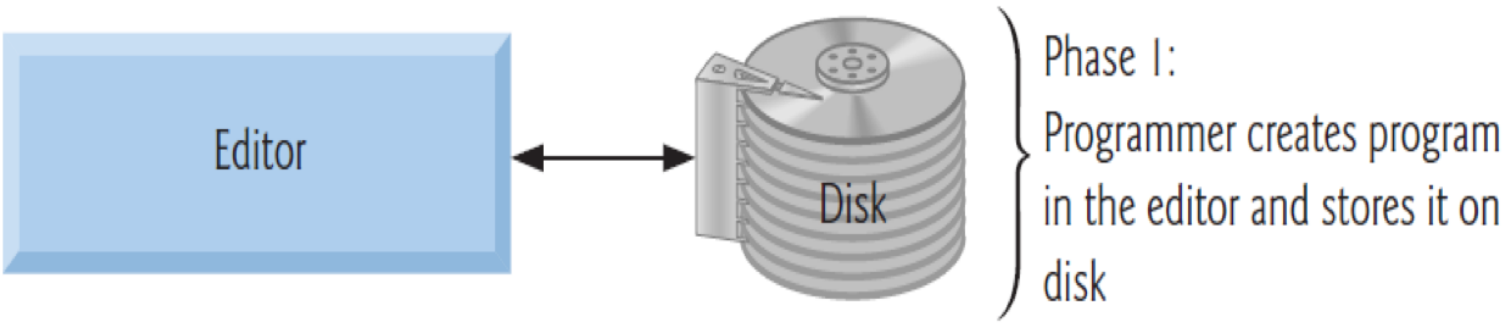
- Passos comuns utilizados na criação e execução de uma aplicação C/C++.

Um ambiente típico de desenvolvimento C/C++

- **Fase 1: Criando um programa**
  - Esta fase consiste da edição de um arquivo com um **programa editor** (normalmente conhecido como um editor).
  - Você digita um programa C/C++ (tipicamente conhecido como **programa fonte**) usando o editor, faz as correções necessárias e salva o programa em um dispositivo de memória secundária, por exemplo, o HD.
  - Frequentemente, os nomes de arquivos dos programas fonte C terminam com a extenção **.c** e de C++ com as extensões **.cpp**, **.cxx** ou **.cc**.

Um ambiente típico de desenvolvimento C/C++

- **Fase 1: Criando um programa**

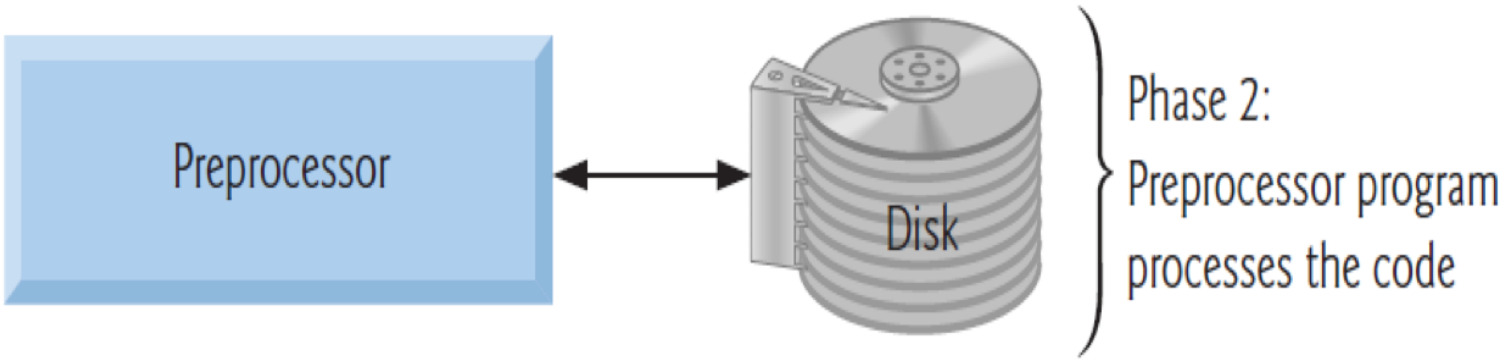


Um ambiente típico de desenvolvimento C/C++

- **Fase 2: Pré-processando um Programa C/C++**
  - Na fase 2, você fornece o comando para **compilar** o programa.
  - Em um sistema C/C++, um programa **pré-processador** executado automaticamente antes que a fase de tradução do compilador inicie (então, chamaremos a fase 2 de pré-processamento e a fase 3 de compilação).
  - O pré-processador obedece a comandos chamados **diretivas do pré-processador**, que indicam que certas manipulações são realizadas no programa antes da compilação.
  - Estas manipulações usualmente incluem outros arquivos de texto para serem compilados, e realizam várias substituições de texto.

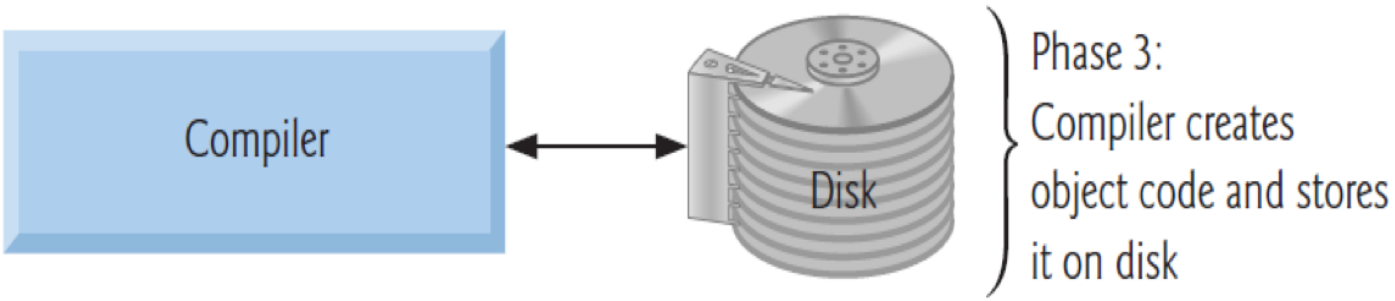
Um ambiente típico de desenvolvimento C/C++

- **Fase 2: Pré-processando um Programa C/C++**



Um ambiente típico de desenvolvimento C/C++

- **Fase 3: Compilando um Programa C/C++**
  - Na fase 3, o compilador **traduz** o programa C/C++ (código fonte, em **alto nível**) em um código de linguagem de máquina (código objeto, em **baixo nível**).

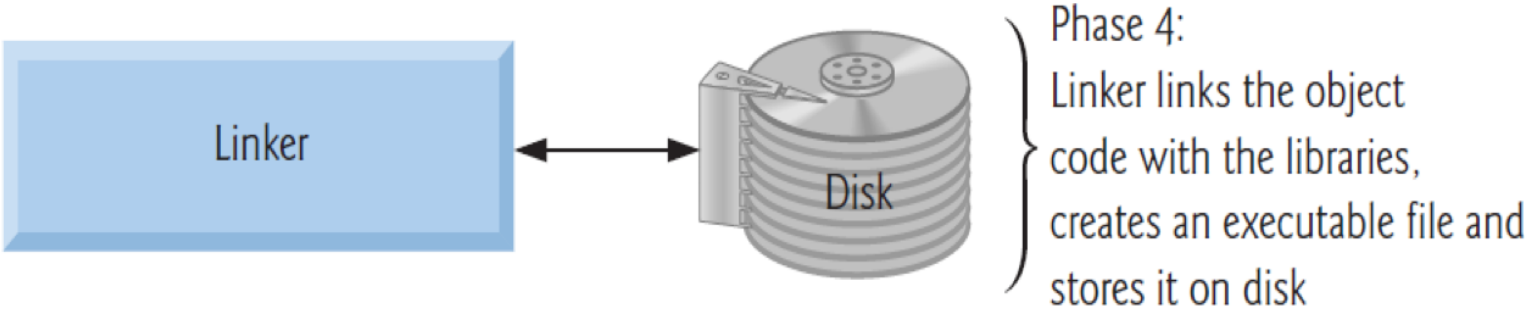


Um ambiente típico de desenvolvimento C/C++

- **Fase 4: Ligação (*linking*)**
  - Tipicamente, um programa C/C++ contém referências para funções e dados definidos em outros lugares, tais como nas bibliotecas padrão ou nas bibliotecas privadas de um grupo de programadores trabalhando em um projeto particular.
  - O código objeto produzido pelo compilador C ou C++ contém, tipicamente, “buracos” por causa dessas partes ausentes. Um **ligador (*linker*)** liga o código objeto com o código das funções ausentes para produzir um programa executável (sem partes ausentes).
  - Se um programa é compilado e ligado corretamente, é produzida uma **imagem executável**.

Um ambiente típico de desenvolvimento C/C++

- **Fase 4: Ligação (*linking*)**

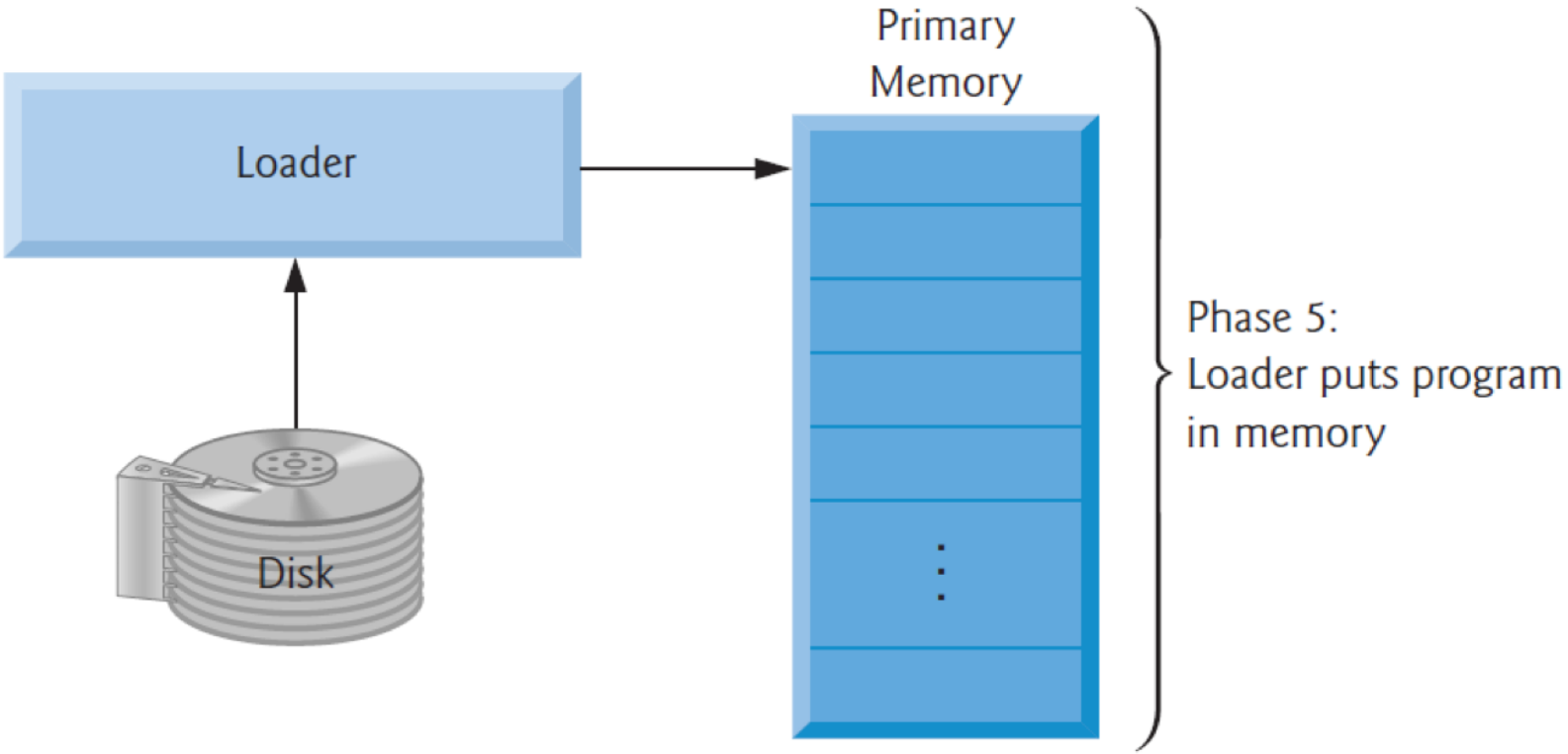


Um ambiente típico de desenvolvimento C/C++

- **Fase 5: Carga (*loading*)**
  - Antes de um programa ser executado, ele deve ser primeiramente colocado na memória (primária). Isto é feito pelo **carregador (*loader*)**, que toma a imagem executável do disco e a transfere para a memória.
  - Os componentes adicionais das bibliotecas compartilhadas, que proveem suporte ao programa, também são carregados.

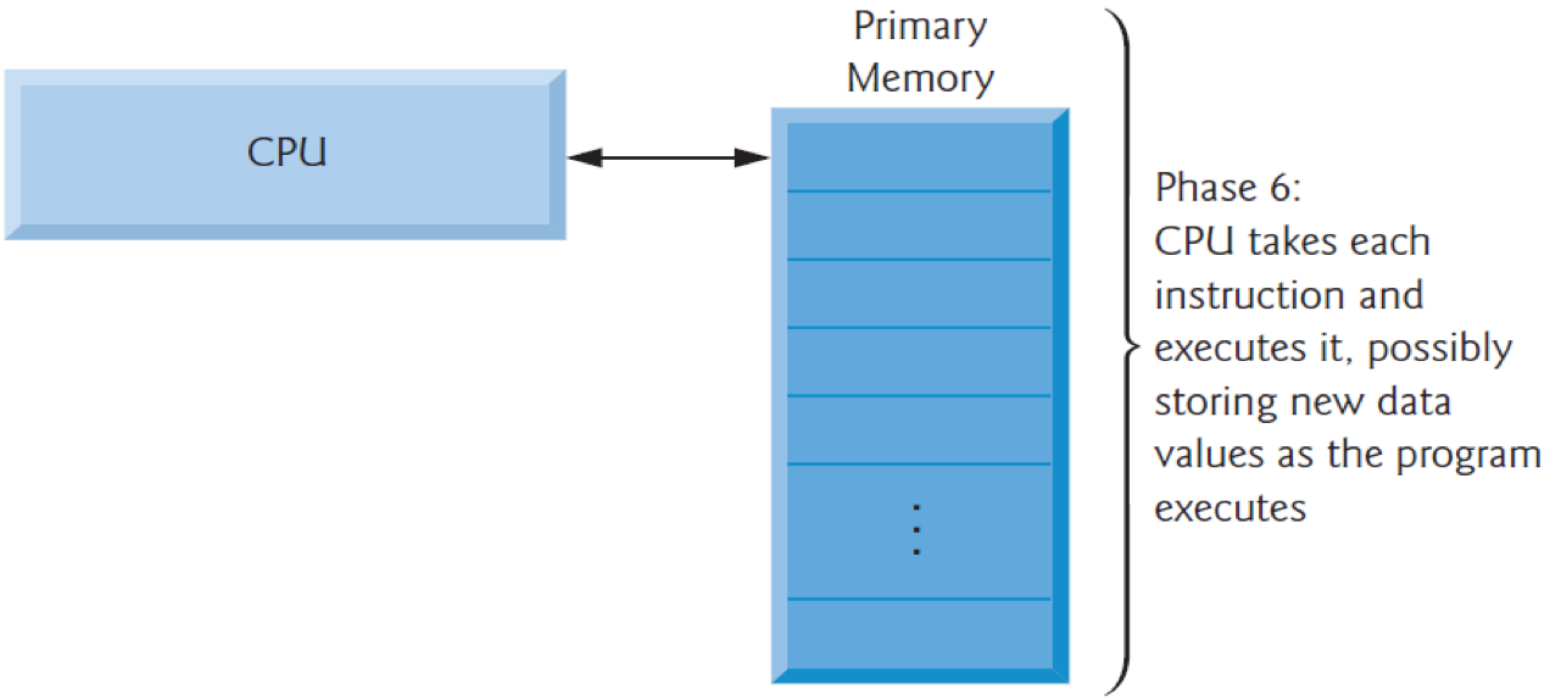
Um ambiente típico de desenvolvimento C/C++

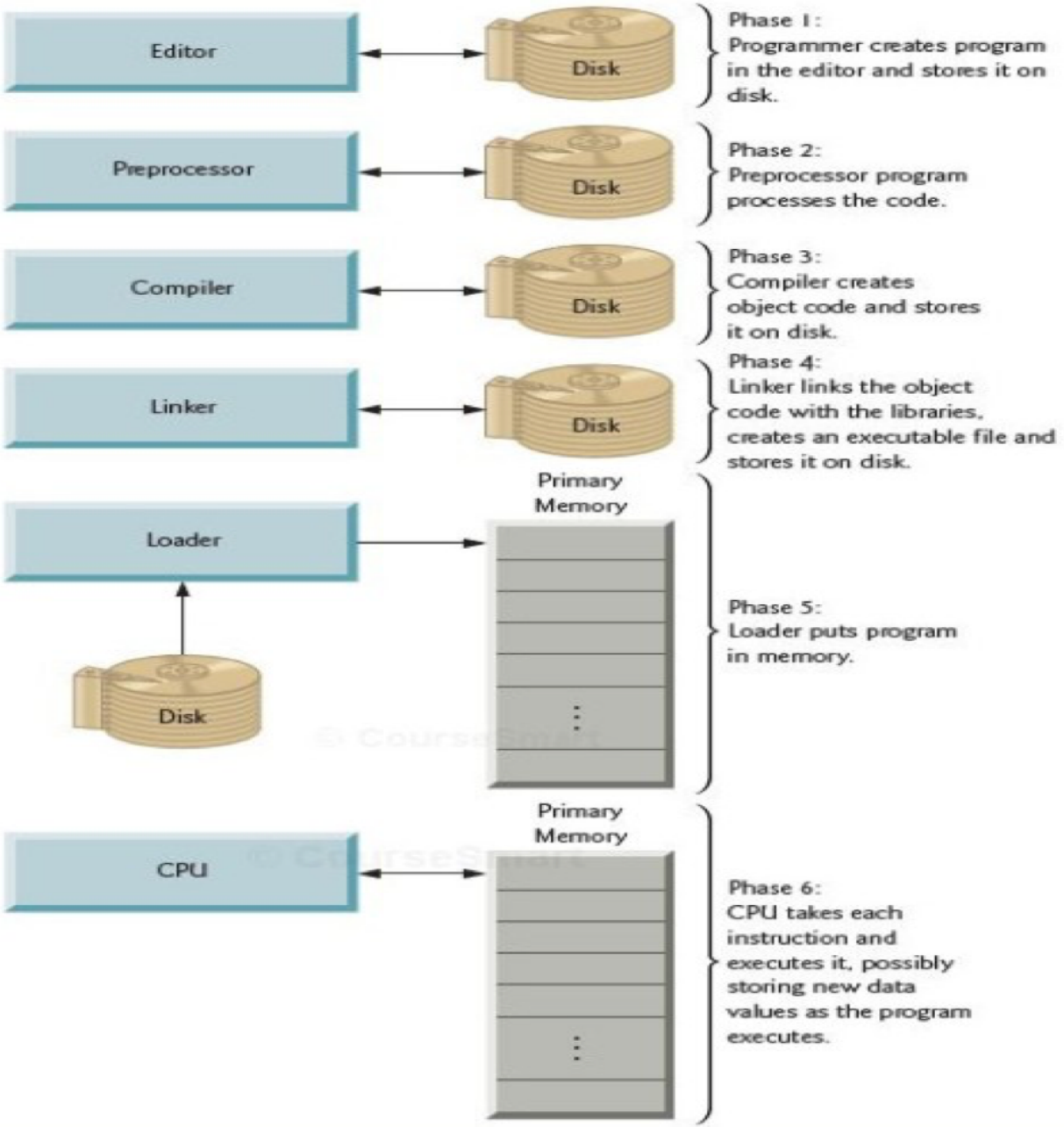
- **Fase 5: Carga (*loading*)**



Um ambiente típico de desenvolvimento C/C++

- **Fase 6: Execução**
  - Finalmente, o computador, sob o controle de sua CPU, executa o programa, uma instrução por vez.
  - A maioria das arquiteturas de computadores atuais podem executar várias instruções em paralelo.




Editor
Disk
Phase 1:
Programmer creates program in the editor and stores it on disk.
Preprocessor
Disk
Phase 2:
Preprocessor program processes the code.
Compiler
Disk
Phase 3:
Compiler creates object code and stores it on disk.
Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk.
Primary Memory
Loader
Disk
Phase 5:
Loader puts program in memory.
Primary Memory
CPU
Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Aula de Hoje

1. O que é um computador?
2. Organização de Computadores
3. Programação e Hierarquia de Dados
4. Linguagens de Programação
5. Primeiro programa

Hello world em C

```c
/* Meu primeiro programa: primeiro.c
 * Programa que imprime uma mensagem na tela
 */

#include <stdio.h>

// Função principal. Inicia a execução do programa
int main()
{
    printf("Hello world!\n");
    return 0; // indica que o programa terminou com sucesso
}
```

Saída do programa:

```
Hello world!
```

Hello world em C++

```cpp
/* Meu primeiro programa: primeiro.cpp
 * Programa que imprime uma mensagem na tela
 */

#include <iostream>

// Função principal. Inicia a execução do programa
int main()
{
    std::cout << "Hello world!" << std::endl;
    return 0; // indica que o programa terminou com sucesso
}
```

Saída do programa:

```
Hello world!
```

Perguntas?