

Conceitos de hardware e software

Sumário

- 2.1 Introdução
- 2.2 Evolução de dispositivos de hardware
- 2.3 Componentes de hardware
 - 2.3.1 Placas principais
 - 2.3.2 Processadores
 - 2.3.3 Relógios
 - 2.3.4 Hierarquia da memória
 - 2.3.5 Memória principal
 - 2.3.6 Armazenamento secundário
 - 2.3.7 Barramentos
 - 2.3.8 Acesso direto à memória (Direct Memory Access — DMA)
 - 2.3.9 Dispositivos periféricos
- 2.4 Suporte de hardware para sistemas operacionais
 - 2.4.1 Processador
 - 2.4.2 Temporizadores e relógios

Conceitos de hardware e software

Sumário *(continuação)*

- 2.4.3 Autocarregamento (Bootstrapping)
- 2.4.4 Plug and play
- 2.5 Caching e buffer
- 2.6 Visão geral do software
 - 2.6.1 Linguagem de máquina e linguagem de montagem
 - 2.6.2 Interpretadores e compiladores
 - 2.6.3 Linguagens de alto nível
 - 2.6.4 Programação estruturada
 - 2.6.5 Programação orientada a objeto
- 2.7 Interfaces de programação de aplicação (APIs)
- 2.8 Compilação, ligação e carregamento
 - 2.8.1 Compilação
 - 2.8.2 Ligação
 - 2.8.3 Carregamento
- 2.9 Firmware
- 2.10 Middleware

Objetivos

■ Este capítulo apresenta:

Os componentes de hardware que devem ser gerenciados por um sistema operacional.

Como o hardware evoluiu para suportar funções de sistemas operacionais.

Como otimizar o desempenho de vários dispositivos de hardware.

O conceito de interface de programação de aplicação (API).

O processo de compilação, ligação e carregamento.

2.1 Introdução

■ Um sistema operacional é primariamente um gerenciador de recursos

Seu projeto deve estar intimamente ligado aos recursos de software e hardware que precisa gerenciar.

processadores

memória

armazenamento secundário (como os discos rígidos)

outros dispositivos de E/S

processos

threads

arquivos

bancos de dados

2.2 Evolução de dispositivos de hardware

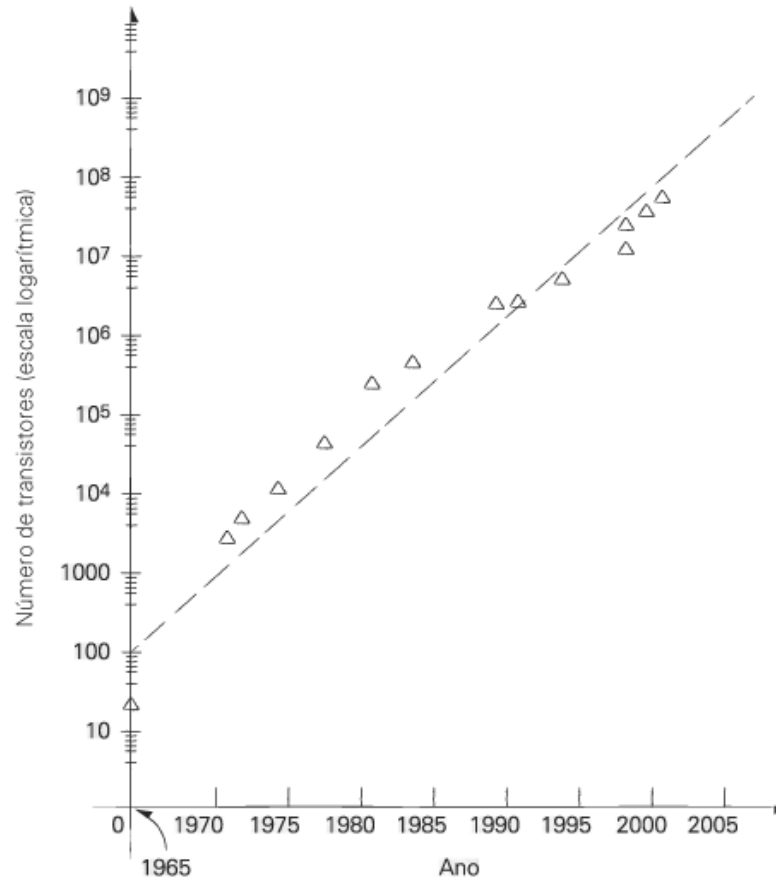
- **Os sistemas operacionais, em sua maioria, não dependem de configurações de hardware**

Os sistemas operacionais usam drivers de dispositivo para executar operações de E/S específicas ao dispositivo.

Por exemplo, os dispositivos plug-and-play, ao serem conectados, instruem o sistema operacional sobre o driver que deve usar, sem solicitar para isso nenhuma intervenção do usuário.

2.2 Evolução de dispositivos de hardware

Figura 2.1 Número de transistores plotado contra o tempo para processadores Intel.



2.3 Componentes de hardware

- **O hardware de um computador consiste em:**
 - processador(es)
 - memória principal
 - dispositivos de entrada/saída

2.3.1 Placas principais

■ Placa de circuito impresso

Componente de hardware que fornece conexões elétricas entre dispositivos.

A placa principal (placa-mãe) é o PCB central em um sistema.

Dispositivos como processadores e memória principal são encaixados nessa placa.

Incluem chips para realizar operações de baixo nível (por exemplo, BIOS).

2.3.2 Processadores

- **O processador é um hardware que executa linguagem de máquina.**

A CPU executa as instruções de um programa.

O co-processador executa instruções específicas.

Ex.: co-processador gráfico ou de áudio

Os registradores são memórias de alta velocidade localizadas em um processador.

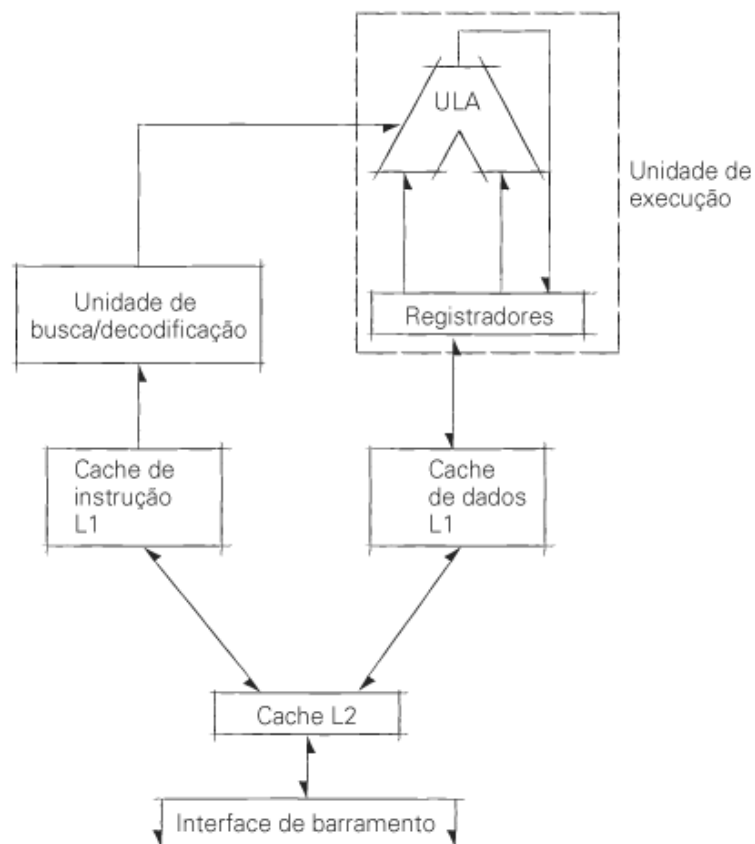
Os dados precisam estar nos registradores para que os processadores possam trabalhar com eles.

O comprimento da instrução é o tamanho de uma instrução em linguagem de máquina.

Alguns processadores suportam vários comprimentos de instrução.

2.3.2 Processadores

Figura 2.2 Componentes do processador.



2.3.3 Relógios

- **O tempo do computador é medido em ciclos.**

Oscilação completa de um sinal elétrico.

Fornecido por um gerador de relógio do sistema.

A velocidade do processador é medida em GHz (bilhões de ciclos por segundo).

A velocidade dos computadores de mesa modernos chega a centenas de megahertz ou a vários gigahertz.

2.3.4 Hierarquia da memória

■ A hierarquia da memória é um esquema de categorização da memória.

A memória mais rápida e cara está no topo da hierarquia, enquanto a mais lenta e barata está na base.

Registradores

Cache L1

Cache L2

Memória principal

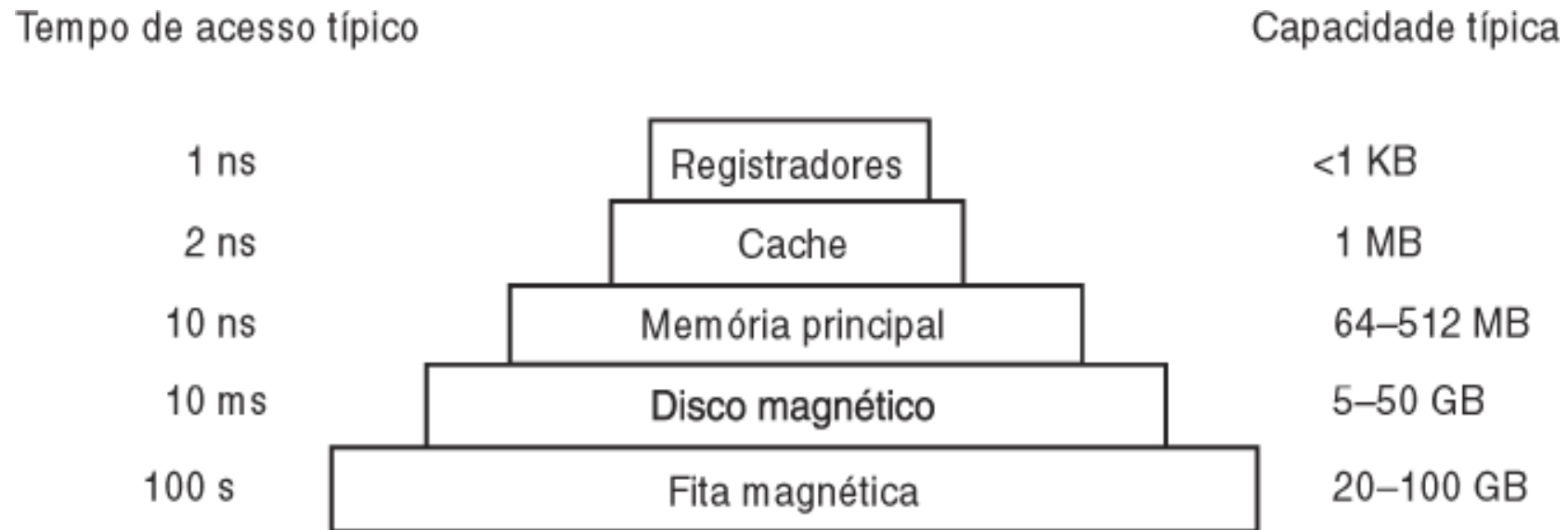
Memória secundária e terciária (CDs, DVDs e unidades de disco flexível)

A memória principal de um sistema é o armazenamento de dados de nível mais baixo da hierarquia da memória à que o processador pode se referir diretamente.

Volátil – perde o conteúdo quando o fornecimento de energia ao sistema é interrompido.

2.3.4 Hierarquia da memória

Figura 2.3 Hierarquia da memória.



- Típica hierarquia de memória
 - números mostrados são apenas de aproximações

2.3.5 Memória principal

- **A memória principal consiste na memória volátil de acesso aleatório (RAM).**

Os processos podem acessar localizações de dados em qualquer seqüência.

Dentre as formas comuns de RAM, incluem-se:

RAM dinâmica (DRAM) – requer circuito de renovação;

RAM estática (SRAM) – não requer circuito de renovação.

A largura de banda é a quantidade de dados que pode ser transferida por unidade de tempo.

2.3.6 Armazenamento secundário

■ O armazenamento secundário armazena grande quantidade de dados permanentes a um baixo custo.

Acessar dados em um disco rígido é mais vagaroso que na memória principal.

- Movimento mecânico do cabeçote de leitura/gravação.

- Latência rotacional.

- Tempo de transferência.

O armazenamento secundário removível facilita o backup e a transferência de dados.

- CDs (CD-R, CD-RW)

- DVDs (DVD-R, DVD+R)

- Discos Zip

- Discos flexíveis

- Cartões de memória Flash

- Fitas

2.3.7 Barramentos

■ Um barramento é um conjunto de pistas.

As pistas são conexões elétricas que transportam informações entre dispositivos de hardware.

Uma porta é um barramento que conecta exatamente dois dispositivos.

Um canal de E/S é um barramento compartilhado por vários dispositivos para executar operações de E/S.

Tratamento de E/S independentemente dos processadores principais do sistema.

Exemplo, o barramento frontal (FSB) conecta processadores à memória principal.

2.3.8 Acesso direto à memória (DMA)

- **O DMA melhora a transferência de dados entre a memória e os dispositivos de E/S.**

Dispositivos e controladores transferem dados para e da memória principal diretamente.

O processador fica livre para executar instruções de instructions.

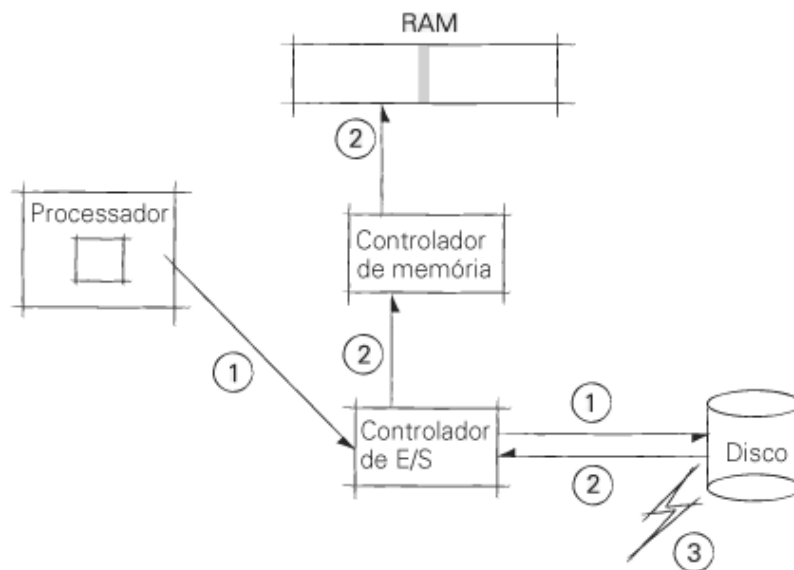
O canal DMA usa um controlador de E/S para gerenciar a transferência de dados.

Notifica o processador quando uma operação de E/S é concluída.

Melhora o desempenho em sistemas que realizam grande quantidade de operações de E/S (por exemplo, computadores de grande porte e servidores).

2.3.8 Acesso direto à memória (DMA)

Figura 2.4 Acesso direto à memória (DMA).



- ① Um processador envia uma solicitação de E/S ao controlador de E/S, que envia a solicitação ao disco. O processador continua executando instruções.
- ② O disco envia dados ao controlador de E/S; os dados são colocados no endereço de memória especificado pelo comando do DMA.
- ③ O disco envia uma interrupção ao processador indicando que a E/S foi concluída.

2.3.9 Dispositivos periféricos

Figura 2.5 Dispositivos periféricos.

<i>Dispositivo</i>	<i>Descrição</i>
Unidade de CD-RW	Lê e grava dados de e para discos óticos.
Unidade de Zip	Transfere dados de e para um disco magnético durável removível.
Unidade de disco flexível	Lê e grava dados de e para discos magnéticos removíveis.
Mouse	Transmite a mudança de localização de um ponteiro ou cursor em uma interface gráfica com o usuário (GUI).
Teclado	Transmite caracteres ou comandos digitados por um usuário.
Impressora multifuncional	Pode imprimir, copiar, enviar fax e escanear documentos.
Placa de som	Converte sinais digitais em sinais de áudio para alto-falantes. Também pode receber sinais de áudio via microfone e produzir um sinal digital.
Acelerador de vídeo	Exibe gráficos na tela; acelera gráficos bi e tridimensionais.
Placa de rede	Envia e recebe dados de e para outros computadores.
Câmera digital	Grava e muitas vezes exibe imagens digitais.
Dispositivo biométrico	Executa varredura (scan) de características humanas como impressões digitais e retinas, normalmente para finalidades de identificação e autenticação.
Dispositivo de infravermelho	Comunica dados entre dispositivos via conexão sem fio em linha de visada.
Dispositivo sem fio	Comunica dados entre dispositivos via conexão sem fio onidirecional.

2.3.9 Dispositivos periféricos

■ Dispositivos periféricos

Qualquer dispositivo de hardware não requerido por um computador para executar instruções de software.

Os dispositivos internos são referidos como dispositivos periféricos integrados.

Placas de interface de rede, modems, placas de som.
Unidades de disco rígido, CD e DVD.

Dispositivos de caracteres que transferem dados: um caractere por vez.

Teclados e mouses

Podem ser conectados a um computador por meio de portas e outros barramentos.

Portas seriais, portas paralelas, USB, portas IEEE 1394 e SCSI



2.4 Suporte de hardware para sistemas operacionais

■ As arquiteturas de computador contêm:

Recursos que executam funções de sistemas operacionais rapidamente em hardware para melhorar o desempenho.

Recursos que habilitam o sistema operacional a impor rígida proteção.

2.4.1 Processador

■ O processador implementa mecanismos de proteção do sistema operacional.

Evita que os processos acessem instruções privilegiadas ou memória.

Os sistemas de computador geralmente dispõem de diferentes modos de execução:

Modo usuário (estado usuário ou estado-problema)

O usuário pode executar apenas um subconjunto de instruções.

Modo núcleo (estado supervisor)

O processador pode acessar instruções privilegiadas e recursos em nome dos processos.



2.4.1 Processador

■ Proteção e gerenciamento da memória

Impede que processos acessem memória que não lhes foi designada.

É implementada por meio de registradores de processador que somente podem ser modificados com instruções privilegiadas.

■ Interrupções e exceções

Quando ocorre um evento, a maioria dos dispositivos envia ao processador um sinal denominado interrupção.

As exceções são interrupções geradas em resposta a erros.

O sistema operacional pode responder a uma interrupção notificando os processos que estão à espera desses eventos.

2.4.2 Temporizadores e relógios

■ Temporizadores

O temporizador de intervalo gera periodicamente uma interrupção.

Os sistemas operacionais usam temporizadores de intervalo para impedir que processos monopolizem o processador.

■ Relógios

Oferecem uma medida de continuidade.

Um relógio de 24 horas habilita o sistema operacional a determinar a hora e a data atuais.

2.4.3 Autocarregamento (Bootstrapping)

■ Autocarregamento: carregamento inicial dos componentes do sistema operacional na memória

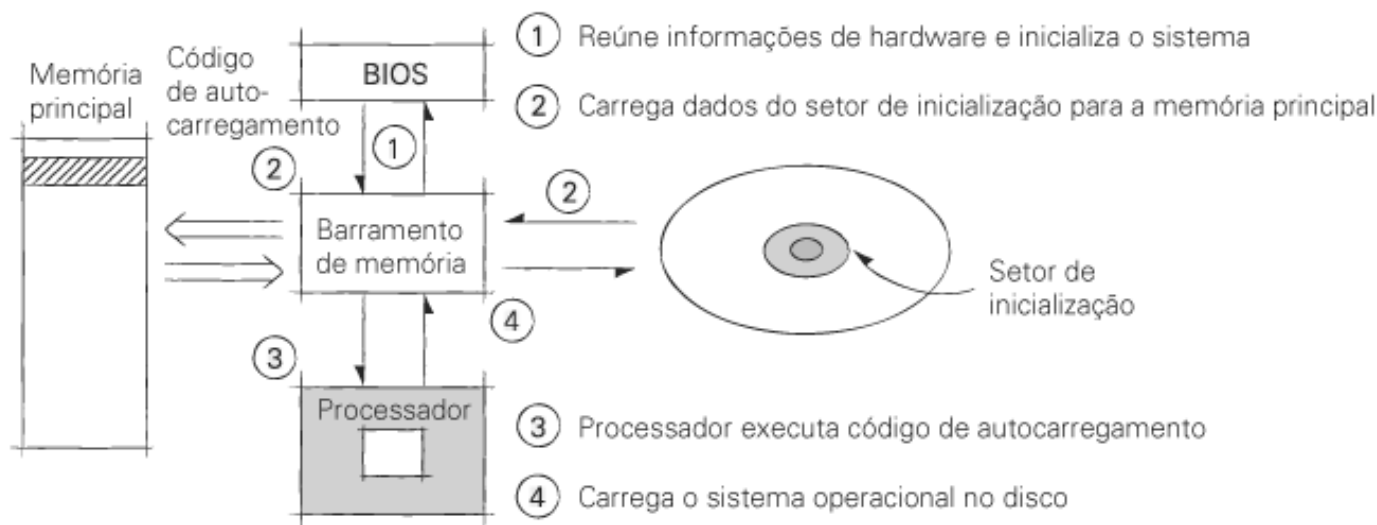
É executado pelo BIOS (Basic Input/Output System) do computador.

Inicializa o hardware do sistema.

Carrega instruções na memória principal provenientes de uma área do armazenamento secundário denominada setor de inicialização (*boot sector*).

Se o sistema não for carregado, o usuário não poderá acessar nenhum hardware do computador.

Figura 2.6 Autocarregamento.



2.4.4 Plug and Play

■ Tecnologia Plug-and-Play

Permite que os sistemas operacionais configurem e usem um hardware recém-instalado, sem precisar interagir com o usuário.

Para suportar a tecnologia plug-and-play, um dispositivo de hardware deve:

- Identificar-se exclusivamente no sistema operacional.

- Comunicar-se com o sistema operacional para indicar os recursos e serviços que requer para funcionar adequadamente.

- Identificar o driver correspondente que permite que o dispositivo seja configurado (por exemplo, que se atribua o dispositivo a um canal DMA).

2.5 Caching

■ Caches

São memórias relativamente rápidas.

Mantêm uma cópia dos dados que serão acessados logo em seguida.

Aumentam a velocidade de execução do programa.

Exemplos incluem:

- Caches L1 e L2 do processador.

- A memória principal pode ser considerada um cache para unidades de disco rígido e outros dispositivos de armazenamento secundário.

2.5.1 Buffer e Spooling

■ Buffers

Área de armazenamento temporário que guarda dados durante transferências de E/S.

São usados principalmente para:

- Coordenar comunicações entre dispositivos que funcionam em diferentes velocidades.

- Armazenar dados para processamento assíncrono.

- Permitir que alguns sinais sejam emitidos assincronamente.

■ Spooling

Técnica de buffer por meio da qual um dispositivo intermediário, como um disco, é interposto entre um processo e um dispositivo de E/S de baixa velocidade.

Permite que os processos solicitem operações a um dispositivo periférico sem que esse dispositivo esteja preparado para atender a essa solicitação.

Exemplo: um spooler de impressão ou em streaming de vídeos online

2.6 Visão geral do software

■ Linguagens de programação

Algumas são compreendidas diretamente pelos computadores, outras exigem tradução.

Em geral, são classificadas como:

- Linguagem de máquina

- Linguagem de montagem ou

- Linguagem de alto nível

2.6.1 Linguagem de máquina e linguagem de montagem

■ Linguagem de máquina

É definida pelo projeto de hardware do computador.

Em geral consiste em cadeias de números (reduzidos a 1s e 0s) que instruem os computadores a executar suas operações mais elementares.

Ex.: **Assembly (montador Assembler)**

■ Linguagem de montagem

Representa instruções em linguagem de máquina por meio de abreviaturas da língua inglesa.

Os montadores convertem a linguagem de montagem em linguagem de máquina.

2.6.2 Interpretadores e compiladores

■ Compilador

Programa de tradução que converte programas em linguagem de alto nível em linguagem de máquina.

■ Interpretador

Programa que executa diretamente o código-fonte ou um código reduzido a uma linguagem de baixo nível que não é o código de máquina.

2.6.3 Linguagens de alto nível

■ Linguagens de alto nível

As instruções se assemelham ao inglês do dia-a-dia.

Executam tarefas mais substanciais com uma quantidade menor de comandos.

Requerem compiladores e interpretadores.

■ Linguagens de alto nível mais conhecidas

Em geral são processuais e orientadas a objeto.

Fortran

Usada para aplicações científicas e de engenharia.

COBOL

Usada para aplicações de negócios que processam grande volume de dados.

C

Linguagem de desenvolvimento do sistema operacional UNIX.

C++/Java

Linguagens populares orientadas a objeto.

C#

Linguagem de desenvolvimento orientada a objeto para a plataforma .NET.

2.6.4 Programação estruturada

■ Abordagem disciplinada para a criação de programas

Programas que sejam claros, provavelmente corretos e fáceis de modificar.

Dentre as linguagens de programação estruturada incluem-se:

Pascal

Desenvolvida para ensinar programação estruturada.

Ada

Desenvolvida pelo Departamento de Defesa dos Estados Unidos.

Fortran

Usada para aplicações científicas e de engenharia.20

2.6.5 Programação orientada a objeto

■ **Objetos**

Um software reutilizável (qualquer substantivo pode ser representado).

Fáceis de modificar e compreender.

Têm propriedades (por exemplo, cor) e executam ações (por exemplo, mover).

■ **Classes**

São tipos de objetos relacionados.

Especificam o formato geral de um objeto e os atributos e ações disponíveis a esse objeto.

■ **Programação orientada a objeto**

Concentra-se em comportamentos e interações, não na implementação.

C++, Java e C# são linguagens orientadas a objeto muito conhecidas.

2.7 Interfaces de programação de aplicação (APIs)

■ Um conjunto de rotinas

Os programadores usam rotinas para solicitar serviços do sistema operacional.

Os programas solicitam funções da API, que então podem acessar o sistema operacional por meio de chamadas ao sistema (**System Call**).

Exemplos de API:

Win32 API para Windows

POSIX API para praticamente todas as versões de UNIX

Java API para a Java Virtual Machine (JVM).

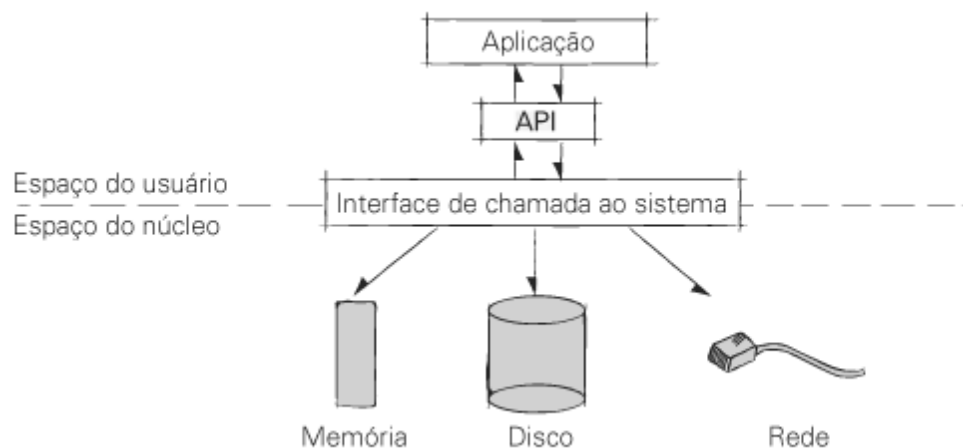


Figura 2.7 Interface de programação de aplicação (API).

2.8.1 Compilação

■ Tradução de código de alto nível para código de máquina

Aceita o código-fonte como entrada e retorna um código-objeto.

As fases da compilação incluem:

Analizador de léxico (lexer)

Separa os caracteres do fonte de um programa em símbolos (*tokens*).

Analizador sintático (parser)

Agrupa os símbolos em comandos sintaticamente corretos.

Gerador de código intermediário

Converte a estrutura sintática em uma cadeia de instruções simples.

Otimizador

Melhora a eficiência de execução do código e reduz os requisitos de memória do programa.

Gerador de código

Produz o arquivo-objeto contendo as instruções em linguagem de máquina.

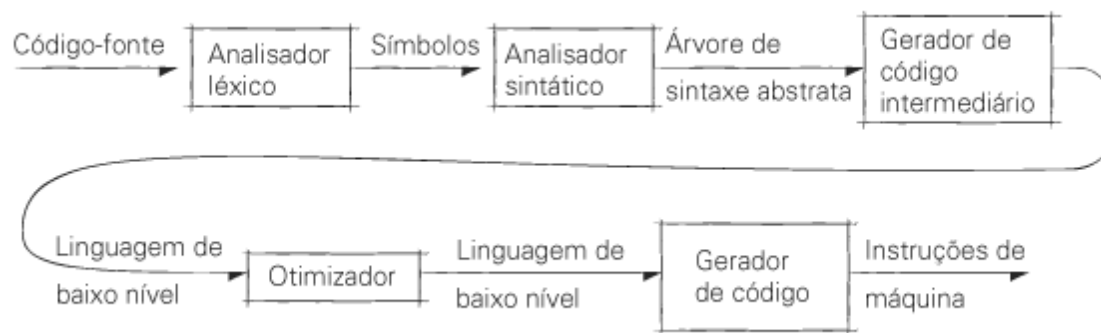


Figura 2.8 Fases da compilação.

2.8.2 Ligação

■ Ligadores

Criam uma única unidade executável.

Integram módulos previamente compilados, denominados bibliotecas, referidos por um programa.

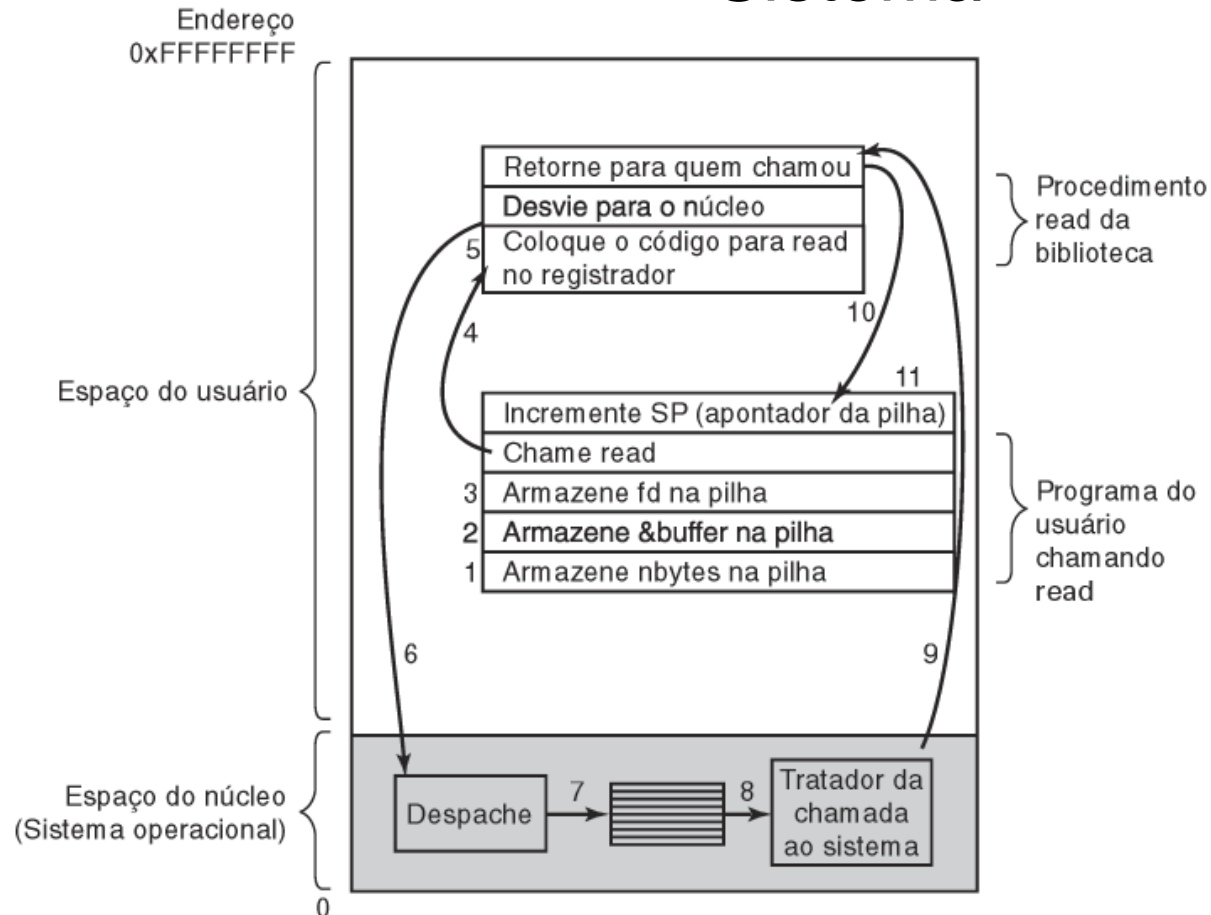
Atribuem endereços relativos a diferentes programas ou unidades de dados.

Resolvem todas as referências externas entre subprogramas.

Produzem um módulo integrado denominado módulo de carga.

A ligação pode ser realizada em tempo de compilação, antes do carregamento, em tempo de carregamento ou em tempo de execução.

Os Passos de uma Chamada ao Sistema



Os 11 passos para fazer uma chamada ao sistema read (fd, buffer, nbytes)



Algumas Chamadas ao Sistema para Gerenciamento de

Gerenciamento de processos

Chamada	Descrição
<code>pid = fork()</code>	Crie um processo filho idêntico ao processo pai
<code>pid = waitpid(pid, &statloc, options)</code>	Aguarde um processo filho terminar
<code>s = execve(name, argv, environp)</code>	Substitua o espaço de endereçamento do processo
<code>exit(status)</code>	Termine a execução do processo e retorne o estado

Algumas Chamadas ao Sistema para Gerenciamento de Arquivos

Gerenciamento de arquivos

Chamada	Descrição
<code>fd = open(file, how, ...)</code>	Abra um arquivo para leitura, escrita ou ambas
<code>s = close(fd)</code>	Feche um arquivo aberto
<code>n = read(fd, buffer, nbytes)</code>	Leia dados de um arquivo para um buffer
<code>n = write(fd, buffer, nbytes)</code>	Escreva dados de um buffer para um arquivo
<code>position = lseek(fd, offset, whence)</code>	Mova o ponteiro de posição do arquivo
<code>s = stat(name, &buf)</code>	Obtenha a informação de estado do arquivo





Algumas Chamadas ao Sistema para Gerenciamento de Diretório

Gerenciamento do sistema de diretório e arquivo

Chamada	Descrição
s = mkdir(name, mode)	Crie um novo diretório
s = rmdir(name)	Remova um diretório vazio
s = link(name1, name2)	Crie uma nova entrada, name2, apontando para name1
s = unlink(name)	Remova uma entrada de diretório
s = mount(special, name, flag)	Monte um sistema de arquivo
s = umount(special)	Desmonte um sistema de arquivo

Algumas Chamadas ao Sistema para Tarefas Diversas

Diversas

Chamada	Descrição
s = chdir(dirname)	Altere o diretório de trabalho
s = chmod(name, mode)	Altere os bits de proteção do arquivo
s = kill(pid, signal)	Envie um sinal a um processo
seconds = time(&seconds)	Obtenha o tempo decorrido desde 1º de janeiro de 1970



Chamadas ao Sistema

- O interior de um shell:

```
#define TRUE 1

while (TRUE) {
    type_prompt( );
    read_command(command, parameters);

    if (fork( ) !=0) {
        /* Parent code. */
        waitpid(-1, *status, 0);
    } else {
        /* Child code. */
        execve(command, parameters, 0);
    }
}
```

/* repita para sempre */
/* mostra prompt na tela */
/* lê entrada do terminal */

/* cria processo filho */

/*aguarda o processo filho acabar */

/*executa o comando */

Interrupções x Traps

Um software pode interromper seu próprio processo(ao fazer uma chamada ao sistema):

- 1) Usando **TRAPs** (Interrupção de software ou Exceções)
- 2) Para isso, a aplicação tem que estar rodando.

Mas ocorrem interrupção que não são causadas por aplicações em execução:

- 1) Interrupções de hardware(eventos externos)
- 2) Um sinal elétrico no hardware;
- 3) Causa: é normalmente iniciada por dispositivos de E/S ou o Clock.

2.8.3 Carregamento

- **Carregadores**

Convertem endereços relativos em endereços físicos.
Colocam cada instrução e dado na memória principal.

- **Técnicas de carregamento de um programa na memória**

 - Carregamento absoluto**

Coloca o programa nos endereços especificados pelo programador ou compilador (supondo que esses endereços estejam disponíveis).

 - Carregamento realocável**

Realoca os endereços do programa de modo que correspondam à sua localização real na memória.

 - Carregamento dinâmico**

Carrega módulos de programa na primeira utilização.

2.9 Firmware

- **O firmware contém instruções executáveis armazenadas na memória persistente ligada a um dispositivo.**

É programado com microprogramação.

Uma camada de programação abaixo da linguagem de máquina de um computador.

Microcódigo

Instrução simples, fundamental, necessária para implementar todas as operações em linguagem de máquina.

2.10 Middleware

- **Middleware é um software para sistemas distribuídos**

Permite interações entre vários processos executados em um ou mais computadores interligados na rede.

Facilita a operação em sistemas heterogêneos distribuídos.

Simplifica o desenvolvimento de aplicações.

Exemplo, Conectividade aberta para banco de dados (Open DataBase Connectivity — ODBC).

Permite que as aplicações acessem um banco de dados por meio de um middleware denominado unidade de ODBC.