Sistemas Operacionais I

Aula passada

- Métodos para a Manipulação de Deadlocks;
- Prevenção/Impedimento de Deadlocks;
- Detecção e Recuperação de Deadlocks.

Agenda

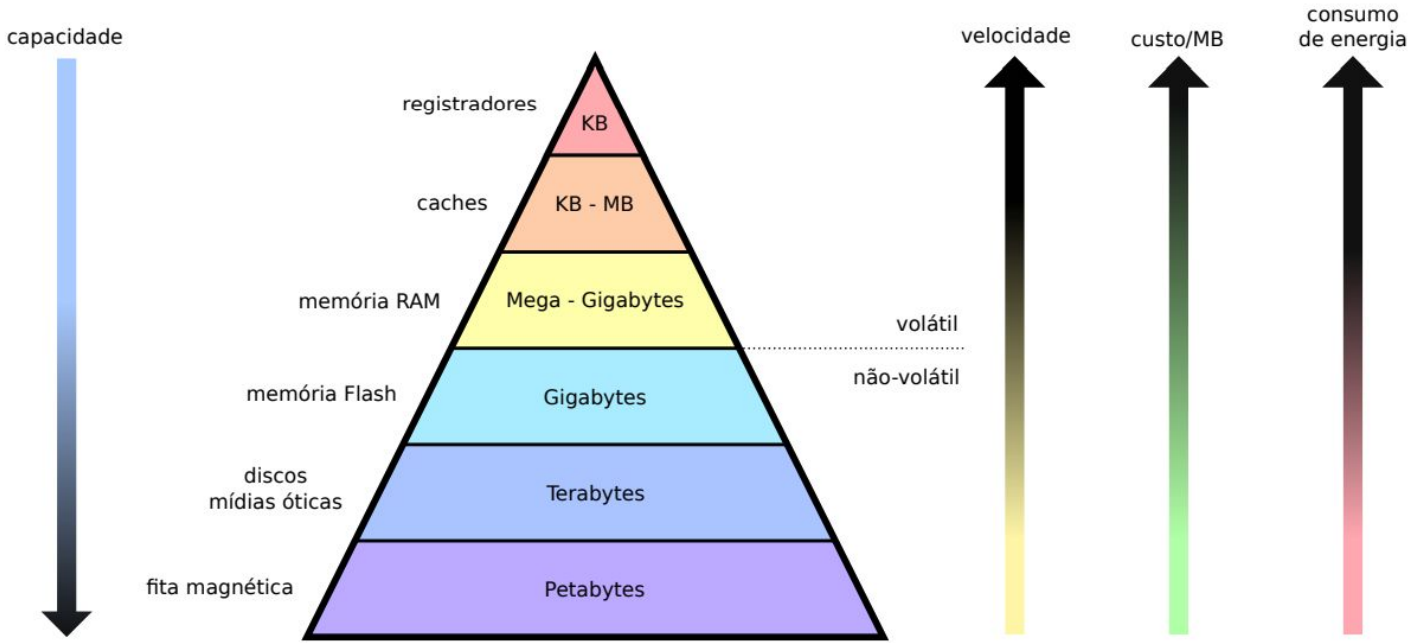
- Gerenciamento de Memória;
  - Hardware Básico;
  - Vinculação de Endereços;
  - Espaço de Endereçamento Lógico *Versus* Espaço de Endereçamento Físico

Hardware Básico

Hardware Básico

- **A memória principal** é um componente fundamental em qualquer sistema de computação;
- Ela constitui o “espaço de trabalho” do sistema, no qual são mantidos os **processos**, **threads** e **bibliotecas compartilhadas**, além do próprio **núcleo do sistema operacional**, com seu código e suas estruturas de dados;
- **O hardware de memória** pode ser bastante complexo, envolvendo diversas estruturas, como memórias **RAM** (*Random Access Memory*), **caches**, **unidade de gerência**, **barramentos**, etc, o que exige um esforço de gerência significativo por parte do sistema operacional.

Hardware Básico

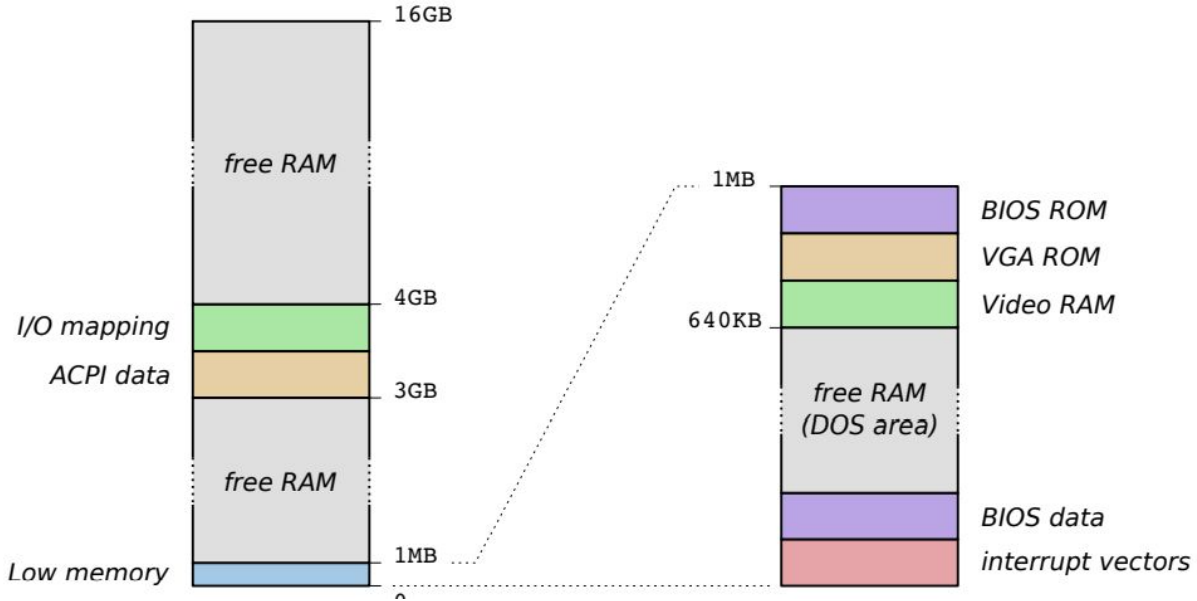


Hierarquia de memória.

Hardware Básico

- A **memória principal** do computador é uma área de **RAM** composta por uma grande **sequência de bytes**, que é a **menor unidade de memória usada pelo processador**;
- Cada **byte** da memória **RAM** possui um **endereço**, que é usado para acessá-lo;
- Um computador convencional atual possui alguns GBytes de memória RAM, usados para conter o sistema operacional e os processos em execução, além de algumas áreas para finalidades específicas, como *buffers* de dispositivos de entrada/saída.

Hardware Básico



Layout da **memória RAM** de um computador.

Hardware Básico

A memória principal e os **registradores embutidos** dentro do próprio processador são o **único** espaço de armazenamento de uso geral que a CPU pode acessar diretamente;

Há **instruções de máquina** que usam **endereços da memória** como argumentos, mas nenhuma que use **endereços de disco**;

Portanto, quaisquer instruções em execução e quaisquer dados que estiverem sendo usados pelas instruções devem estar em um desses dispositivos de armazenamento.

Se os dados não estiverem na memória, devem ser transferidos para lá antes que a CPU possa operar sobre eles.

Hardware Básico

- Os **registradores** que estão embutidos na CPU são geralmente acessíveis dentro de um **ciclo do relógio** (*clock*) da CPU;
- A maioria das CPUs pode decodificar instruções e executar operações simples sobre o conteúdo dos registradores à taxa de uma ou mais operações por tique do relógio.

O mesmo **não pode ser dito da memória principal** que é acessada por uma transação no bus da memória. Para completar um acesso à memória podem ser necessários **muitos ciclos do relógio da CPU**.

Nesses casos, o processador normalmente precisa ser **interrompido**, já que ele não tem os dados requeridos para completar a instrução que está executando. Essa situação é **intolerável** por causa da frequência de acessos à memória.

A solução é adicionar uma memória rápida entre a CPU e a memória principal, geralmente no chip da CPU para acesso rápido. A memória **cache**.

Hardware Básico

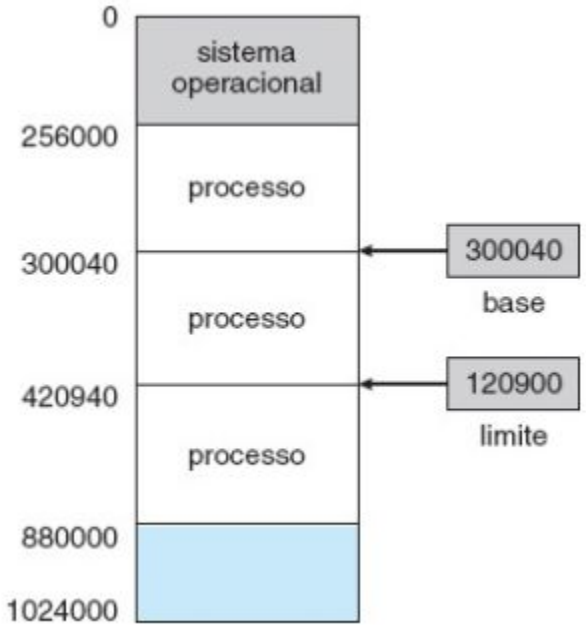
Além de estarmos preocupados com a **velocidade** relativa do acesso à memória física, também devemos assegurar a **operação correta**.

- Para a operação apropriada do sistema, devemos **proteger** o sistema operacional contra o acesso de **processos de usuário**;
- Em sistemas multiusuários, devemos adicionalmente proteger os **processos de usuário uns dos outros**. Essa proteção deve ser fornecida pelo **hardware** porque **o sistema operacional não costuma intervir entre a CPU e seus acessos à memória**;
- O hardware implementa essa proteção de várias maneiras diferentes...

Hardware Básico

- Primeiro, precisamos nos certificar de que **cada processo** tenha um espaço de memória **separado**;
- O **espaço de memória separado** por processo **protege os processos uns contra os outros** e é fundamental para que existam múltiplos processos carregados na memória para execução concorrente;
- Para separar os espaços de memória, precisamos ser capazes de determinar o **intervalo de endereços legais** que o processo pode **acessar** e **assegurar** que ele possa acessar **somente** esses endereços legais;
- É possível fornecer essa proteção usando dois registradores, usualmente um **registrador base** e um **registrador limite**.

Hardware Básico

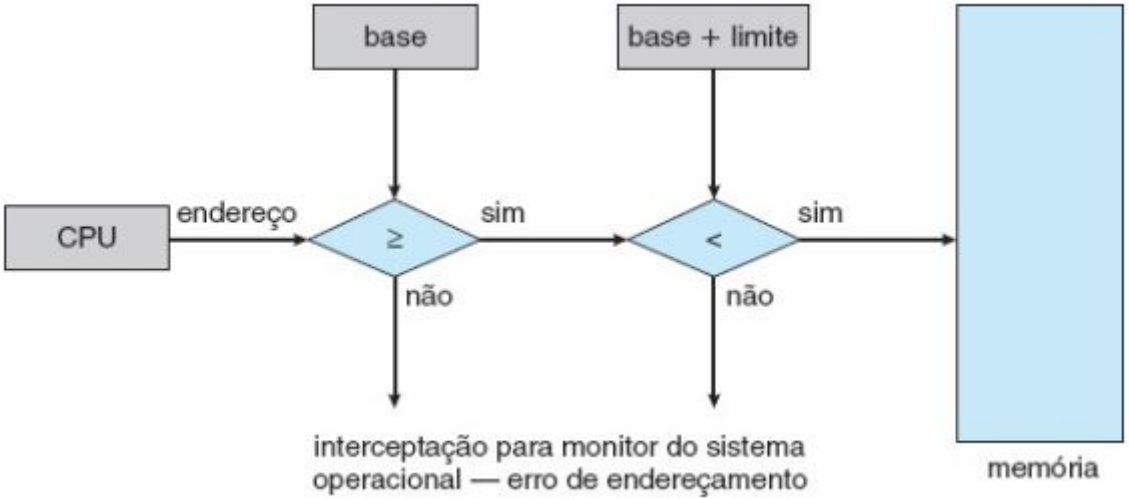


Um **registrador base** e um **registrador limite** definem um espaço de **endereçamento lógico**.

Hardware Básico

- A **proteção** do espaço da memória é fornecida pela **comparação** que o **hardware da CPU** faz entre cada endereço gerado na modalidade de usuário e os registradores;
- **Qualquer tentativa** de um **programa**, executando em modalidade de **usuário**, de **acessar** a **memória do sistema operacional ou a memória de outros usuários** resulta em uma interceptação para o sistema operacional que trata a tentativa como um **erro fatal**.

Hardware Básico



Proteção de endereço de hardware com registradores **base** e **limite**.

Vinculação de Endereços

Vinculação de Endereços

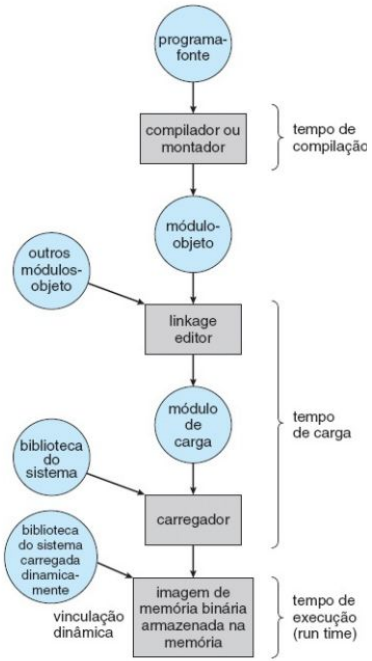
- Usualmente, um **programa** reside em um **disco como um arquivo binário executável**;
- Para ser executado, o programa deve ser **trazido para a memória** e **inserido dentro de um processo**;
- Dependendo do **esquema de gerenciamento da memória** em uso, o processo pode ser movimentado entre o disco e a memória durante sua execução;
- Os processos em disco que estão esperando para serem trazidos à memória para execução formam a **fila de entrada**.

Vinculação de Endereços

Vinculação de Endereços

- Na maioria dos casos, um **programa de usuário** percorre vários passos — alguns dos quais podem ser opcionais — antes de ser executado;
- Os endereços podem ser representados de diferentes maneiras durante esses passos;
- Os **endereços** do **programa-fonte** são, em geral, **simbólicos**;
- Normalmente, um **compilador** vincula esses **endereços simbólicos** a **endereços relocáveis;**
- **Por sua vez, o *linkage editor* ou carregador vincula os endereços relocáveis a endereços absolutos (como 74014). Cada vinculação é um mapeamento de um espaço de endereçamento para outro.**

Vinculação de Endereços



Processamento de um programa de usuário em vários passos.

Vinculação de Endereços

Classicamente, a vinculação de **instruções** e **dados** a **endereços da memória** pode ser feita em qualquer passo ao longo do percurso:

- **Tempo de compilação:**
  - Se você souber, em **tempo de compilação,** onde o processo residirá na memória, um código absoluto pode ser gerado;
  - Por exemplo, se você souber que um processo de usuário residirá a partir da locação *R*, então o código gerado pelo compilador começará nessa locação e se estenderá a partir daí;
  - Se, em algum momento mais tarde, a locação inicial mudar, então será necessário recompilar esse código;
  - Os programas no formato COM do MS-DOS são vinculados em tempo de compilação.

Vinculação de Endereços

Classicamente, a vinculação de **instruções** e **dados** a **endereços da memória** pode ser feita em qualquer passo ao longo do percurso:

- **Tempo de carga:**
  - Se não for conhecido, em tempo de compilação, onde o processo residirá na memória, então o compilador deve gerar **código relocável**;
  - Nesse caso, a vinculação final é adiada até o **tempo de carga**;
  - Se o **endereço inicial mudar**, precisaremos apenas **recarregar** o código do usuário para incorporar esse valor alterado.

Vinculação de Endereços

Classicamente, a vinculação de **instruções** e **dados** a **endereços da memória** pode ser feita em qualquer passo ao longo do percurso:

- **Tempo de execução:**
  - Se o **processo** puder ser **movido** de um segmento de **memória** para outro durante sua execução, então a vinculação deverá ser adiada até o **tempo de execução**;
  - Um **hardware especial** deve estar disponível para esse esquema funcionar;
  - A maioria dos sistemas operacionais de uso geral empregam esse método.

Espaço de Endereçamento Lógico *Versus* Espaço de Endereçamento Físico

Espaço de Endereçamento Lógico *Versus* Espaço de Endereçamento Físico

- Um endereço **gerado pela CPU** é comumente referenciado como **endereço lógico**, enquanto um endereço visto pela **unidade de memória** — aquele que é carregado no registrador de endereços da memória — costuma ser referenciado como **endereço físico**;
- Os métodos de **vinculação de endereços** em **tempo de compilação** e em **tempo de carga** geram **endereços lógicos** e **físicos idênticos**;
- O esquema de vinculação de endereços em **tempo de execução** resulta em **endereços lógicos** e **físicos diferentes**;
- Nesse caso, usualmente referenciamos o **endereço lógico** como um **endereço virtual**.

Espaço de Endereçamento Lógico *Versus* Espaço de Endereçamento Físico

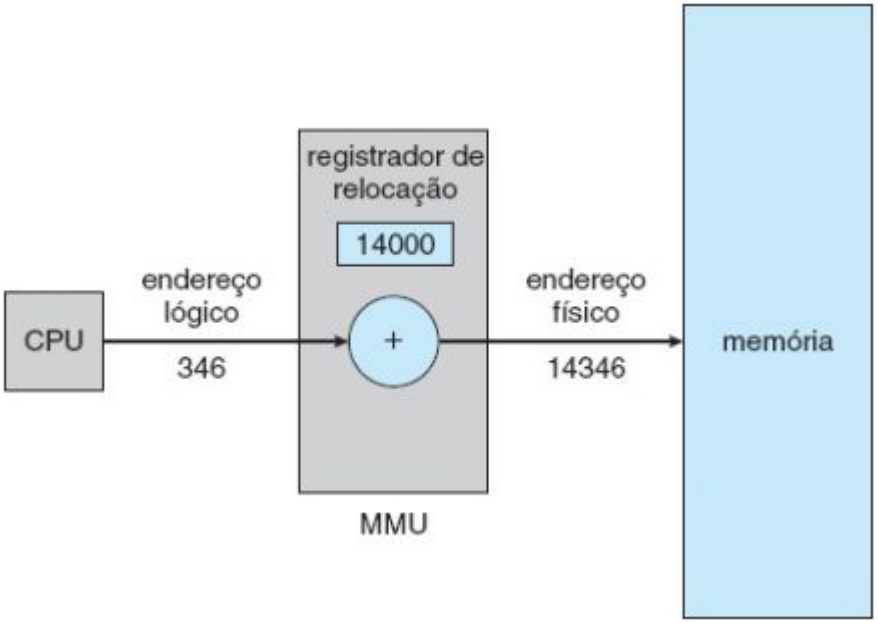
- O conjunto de todos os **endereços lógicos** gerados por um programa é um **espaço de endereçamento lógico**;
- O conjunto de todos os **endereços físicos** correspondentes a esses endereços lógicos é um **espaço de endereçamento físico**;
- Portanto, no esquema de vinculação de endereços em **tempo de execução**, os espaços de **endereçamento lógico e físico diferem**.

Espaço de Endereçamento Lógico *Versus* Espaço de Endereçamento Físico

O mapeamento de **endereços virtuais** para **endereços físicos** em **tempo de execução** é feito por um **dispositivo de hardware** chamado de **unidade de gerenciamento da memória** (MMU — *memory management unit*).

Podemos selecionar entre vários métodos diferentes para fazer esse mapeamento

Espaço de Endereçamento Lógico *Versus* Espaço de Endereçamento Físico



Relocação dinâmica usando um registrador de relocação.

Carga Dinâmica

Carga Dinâmica

- Em nossa discussão até o momento, o **programa inteiro** e **todos os dados** de um processo tinham de estar na **memória física** para o processo ser executado;
- Portanto, o tamanho de um processo ficava **limitado** ao **tamanho da memória física**;
- Para obter uma utilização melhor do espaço da memória, podemos usar a **carga dinâmica**.

Carga Dinâmica

- Com a **carga dinâmica**, uma rotina não é carregada até ser chamada;
- Todas as rotinas são mantidas em **disco** em um formato de carga relocável;
- O **programa principal** é carregado na **memória e executado**;
- Quando uma rotina precisa chamar outra rotina, a rotina chamada verifica, primeiro, se a outra rotina foi carregada;
- Se não o foi, o **carregador de vinculação relocável** é chamado para carregar a rotina desejada na memória e atualizar as tabelas de endereços do programa para que reflitam essa alteração.

Carga Dinâmica

- A vantagem da **carga dinâmica** é que uma rotina é carregada somente quando ela é necessária;
- Esse método é particularmente **útil** quando **grandes volumes de código** são necessários para manipular situações **pouco frequentes**, como as **rotinas de erro**;
- Nesse caso, embora **o tamanho total do programa possa ser grande**, a parte que é usada (e, portanto, **carregada**) **pode ser muito menor**;
- **A carga dinâmica não requer suporte especial do sistema operacional. Os usuários têm a responsabilidade de projetar seus programas de modo a tirarem vantagem desse método.**

Vinculação Dinâmica e Bibliotecas Compartilhadas

Vinculação Dinâmica e Bibliotecas Compartilhadas

- As **bibliotecas vinculadas dinamicamente** são bibliotecas do sistema que são **vinculadas** a programas de usuário quando os **programas** são **executados**;
- Alguns sistemas operacionais suportam apenas a **vinculação estática** em que as **bibliotecas do sistema** são tratadas como qualquer outro módulo objeto e incorporadas pelo carregador à imagem binária do programa;
- A vinculação dinâmica é semelhante à carga dinâmica.