Sistemas Operacionais I

Aula passada

- Permuta entre Processos (Swapping);
  - Permuta-Padrão;

Agenda

- Alocação de Memória Contígua;
  - Proteção da Memória;
  - Alocação de Memória;
  - Fragmentação;

Alocação de Memória Contígua

A **memória principal** deve acomodar tanto o **sistema operacional** quanto os diversos **processos de usuário**. Portanto, precisamos alocar a memória principal da maneira **mais eficiente possível**.

A memória é usualmente dividida em **duas partições**:

- Uma para o sistema operacional;
- E outra para os processos de usuário.

Podemos alocar o sistema operacional tanto na **memória baixa** quanto na **memória alta**.

O **principal fator** que afeta essa decisão é a localização do **vetor de interrupções**. Já que o vetor de interrupções costumam estar na **memória baixa**, os programadores também costumam alocar o **sistema operacional** na **memória baixa**.

Discutimos somente a situação em que o sistema operacional reside na memória baixa.

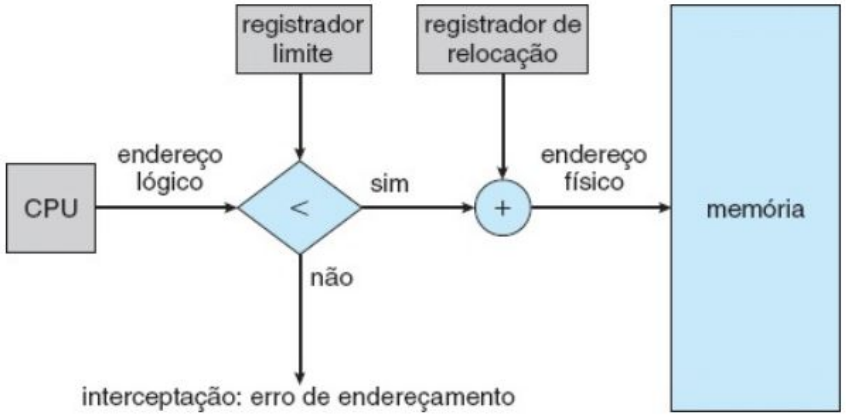
Alocação de Memória Contígua

Proteção da Memória

Proteção da Memória

Precisamos **impedir** que um **processo** acesse memória que ele **não possui**.

O esquema do **registrador de relocação** fornece um modo eficaz para permitir que o **tamanho do sistema operacional mude dinamicamente**.



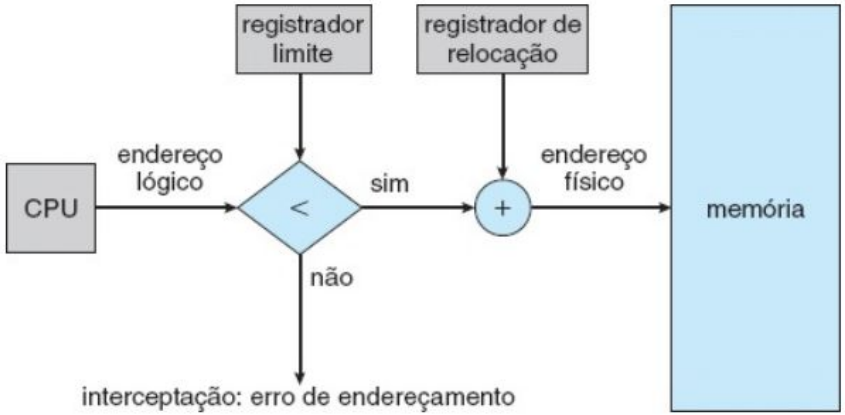
Suporte de hardware para registradores de relocação e registradores limite.

Proteção da Memória

Essa **flexibilidade** é desejável em muitas situações.

Por exemplo: o sistema operacional contém **código** e **espaço em buffer** de drivers de dispositivos.

- Se um driver de dispositivos (ou outro serviço do sistema operacional) **não for comumente usado**, não queremos manter o **código** e os **dados** na memória;
- Podemos usar esse espaço para **outras finalidades**.

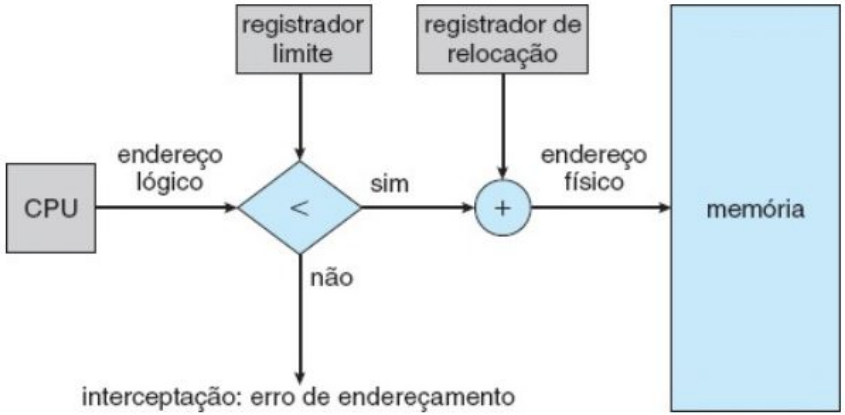


Suporte de hardware para registradores de relocação e registradores limite.

Proteção da Memória

Por exemplo: o sistema operacional contém **código** e **espaço em buffer** de drivers de dispositivos.

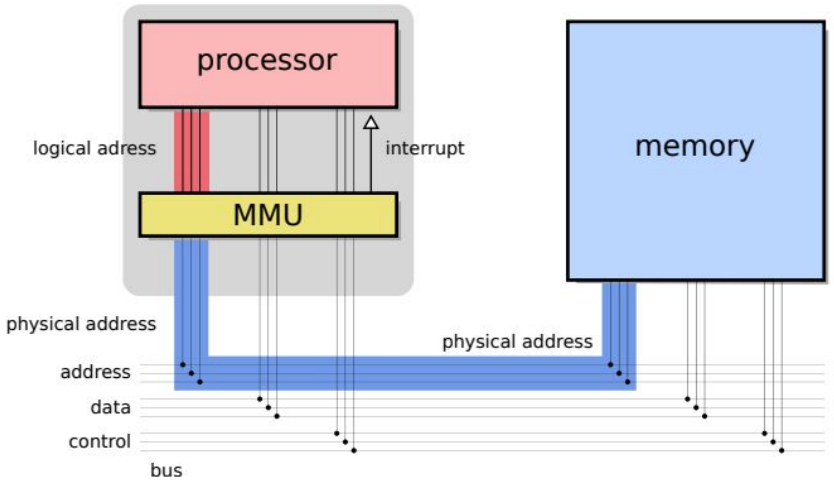
- Esse código é às vezes chamado de **código transiente** do sistema operacional;
- Ele vem e vai conforme necessário;
- Seu uso **altera** o **tamanho** do **sistema operacional** durante a **execução de programas**.



Suporte de hardware para registradores de relocação e registradores limite.

Proteção da Memória

A **MMU (*Memory Management Unit*)** mapeia o endereço lógico dinamicamente adicionando o valor ao registrador de relocação.



Funcionamento básico de uma **MMU**.

Alocação de Memória

Alocação de Memória

Agora estamos prontos para voltar à alocação da memória.

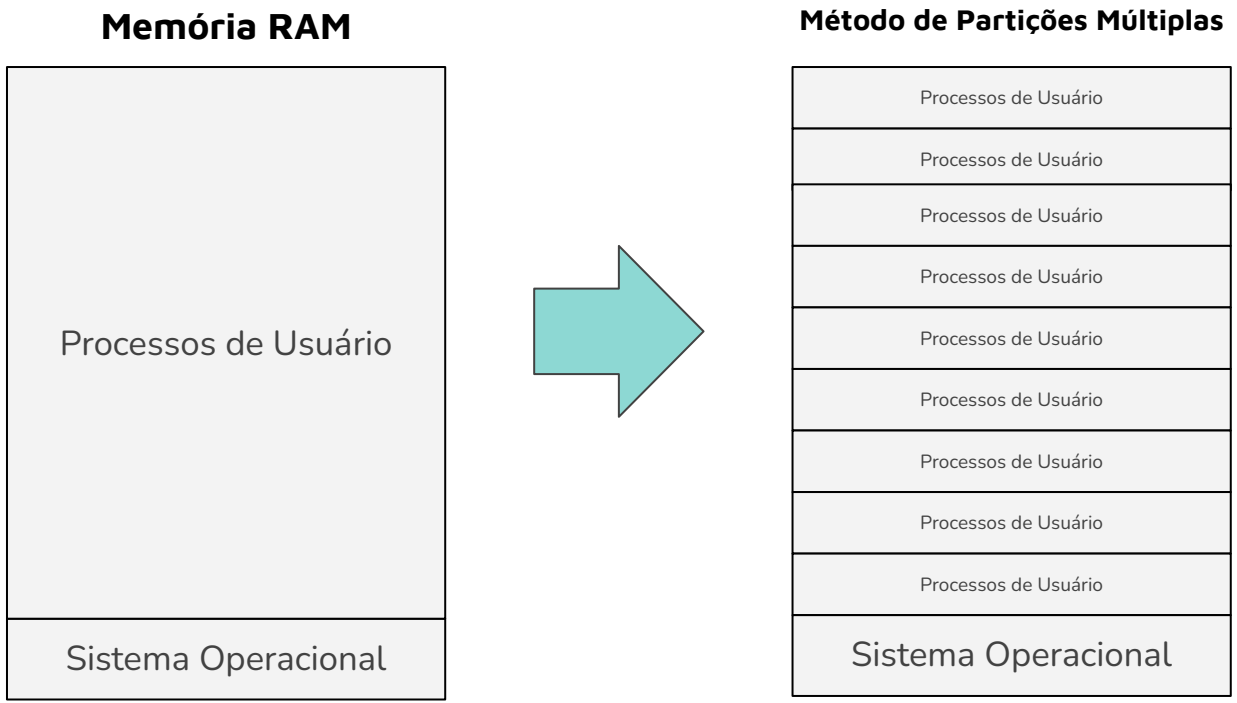
Um dos métodos **mais simples** para **alocação** da memória é **dividir** a memória em várias **partições de tamanho fixo**.

Cada **partição** pode conter **exatamente um processo**. Portanto, o grau de multiprogramação é limitado pelo número de partições.

Nesse **método de partições múltiplas**, quando uma partição está livre, um processo é selecionado da fila de entrada e carregado na partição disponível.

Esse método (**chamado MFT**) foi originalmente usado pelo sistema operacional IBM OS/360, **mas não está mais em uso**.

Alocação de Memória



Alocação de Memória

No esquema de **partições variáveis**, o sistema operacional mantém uma **tabela** indicando quais partes da memória estão **disponíveis** e quais estão **ocupadas**.

Inicialmente, toda a **memória está disponível** para processos de usuário e é considerada um **grande bloco** de memória disponível.

Conforme os processos entram no sistema, eles são colocados em uma **fila de entrada**.

O **sistema operacional** leva em consideração os **requisitos** de memória de **cada processo** e o montante de espaço disponível na memória para determinar que processos devem ter memória alocada.

Quando um **processo recebe espaço**, ele é **carregado** na memória e pode, então, competir por tempo da CPU.

Quando um processo **termina**, **libera** sua memória que o **sistema operacional** pode então **preencher** com outro processo da fila de entrada.

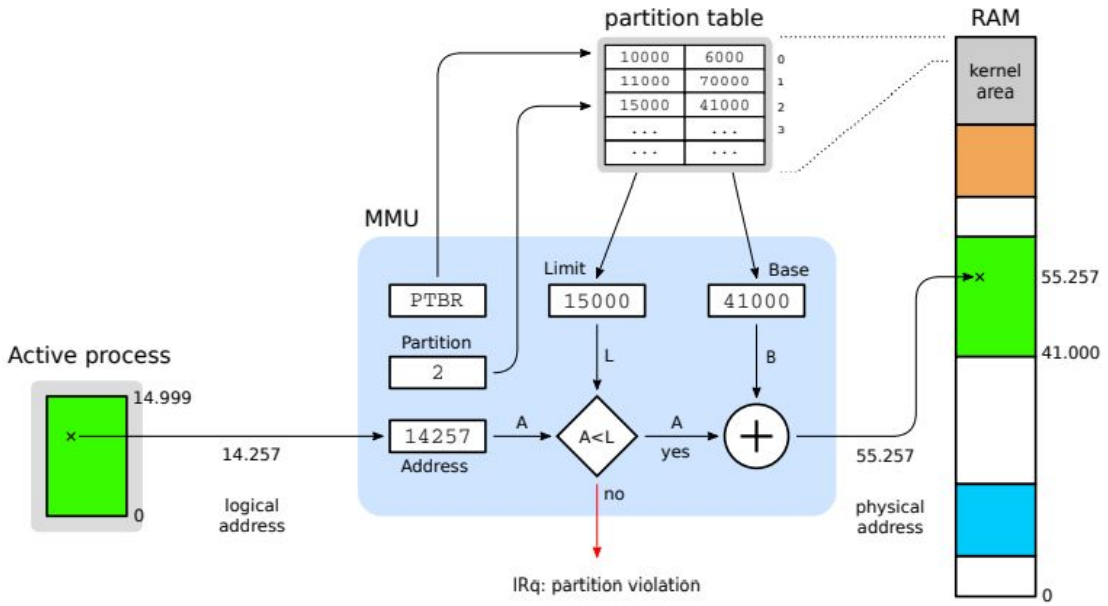
Alocação de Memória

Os blocos de memória disponíveis compõem um **conjunto** de **brechas** de **vários tamanhos** espalhadas pela memória.

Quando um **processo** chega e **precisa** de **memória**, o sistema **procura** no conjunto por uma **brecha** que seja suficientemente grande para esse processo.

- Se a **brecha** for grande demais, ela será dividida em duas partes.
  - Uma parte é alocada ao processo que chegou; a outra é devolvida ao conjunto de brechas.
- Quando um **processo** é **encerrado**, ele **libera** seu bloco de memória que é, então, colocado novamente no **conjunto** de **brechas**.
- Se a nova **brecha** for adjacente a outras **brechas**, essas **brechas** adjacentes serão mescladas para formar uma **brecha** maior.
  - Nesse momento, o sistema pode precisar verificar se existem processos esperando por memória e se essa memória recém liberada e recombinada poderia atender às demandas de algum desses processos em espera.

Alocação de Memória



Esquema de **partições variáveis**.

Alocação de Memória

Esse procedimento é uma **instância específica** do **problema de alocação de memória dinâmica** geral que diz respeito a como satisfazer uma solicitação de tamanho n a partir de uma lista de brechas livres.

Há muitas soluções para esse problema. As estratégias do:

- **Primeiro-apto** (*first-fit*);
- **Mais-apto** (*best-fit*);
- **Menos-apto** (*worst-fit*).

São as mais usadas para selecionar uma **brecha** livre no conjunto de **brechas** disponíveis.

Alocação de Memória

Primeiro-apto (*first-fit*):

- Aloca a primeira **brecha** que seja suficientemente grande;
- A busca pode começar tanto no início do conjunto de **brechas** quanto na locação na qual a busca anterior pelo primeiro-apto terminou;
- Podemos encerrar a busca, assim que encontrarmos uma **brecha** livre suficientemente grande.

Alocação de Memória

Mais-apto (*best-fit*):

- Aloca a menor **brecha** que seja suficientemente grande;
- Devemos **pesquisar** a lista inteira, a menos que ela seja **ordenada** por tamanho;
- Essa estratégia produz a **brecha** com menos espaço sobrando.

Alocação de Memória

Menos-apto (*worst-fit*):

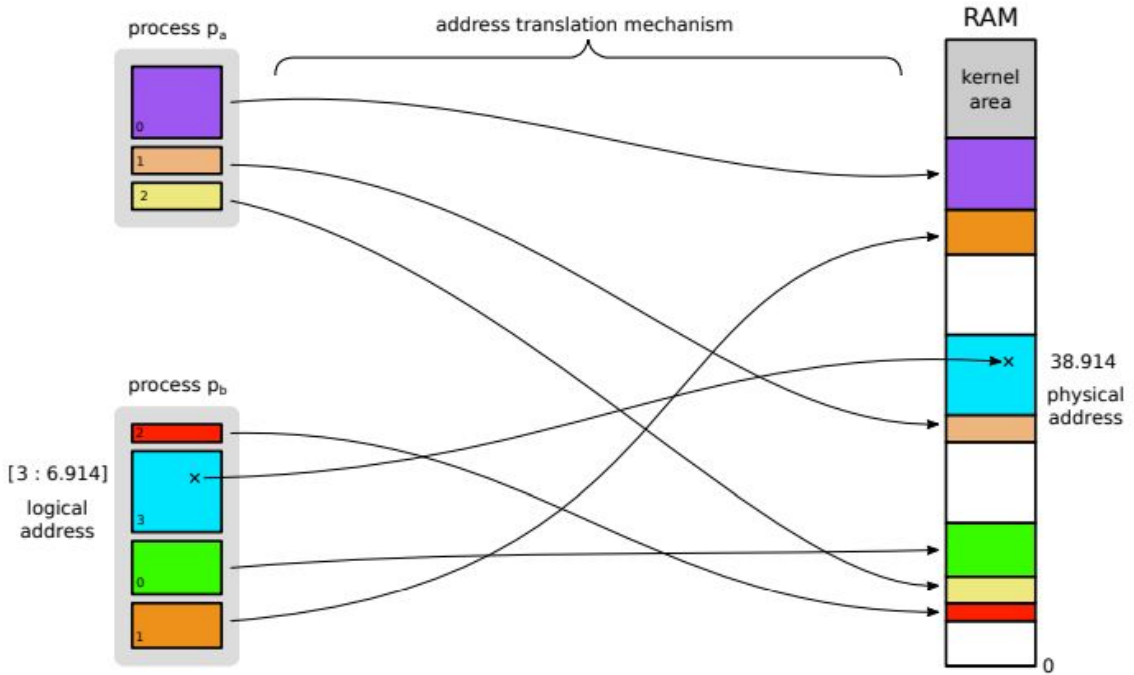
- Aloca a maior **brecha**;
- Novamente, devemos **pesquisar** a lista inteira, a menos que ela seja classificada por tamanho;
- Essa estratégia produz a **brecha** com mais espaço sobrando, que pode ser mais útil do que a **brecha** com menos espaço sobrando da abordagem do mais-apto.

Alocação de Memória

Simulações têm mostrado que tanto o **primeiro-apto** quanto o **mais-apto** são melhores do que o **menos-apto** em termos de redução de tempo e utilização de memória.

Nem o **primeiro-apto**, nem o **mais-apto** é claramente melhor do que o outro em termos de utilização de memória, mas o **primeiro-apto** geralmente é mais rápido.

Alocação de Memória



Esquema de **partições por segmentos**.

Fragmentação

Fragmentação