INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE - IFRN
GRADUAÇÃO - CST EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS (ADS)

AULA - 07: GERENCIAMENTO DE MEMÓRIA

Prof. Dr: José de Arimateia Pinto Magno

Sumário

A memória

Importância e Crescimento da Capacidade de Memória:

"A memória principal (RAM) é um recurso importante que deve ser cuidadosamente gerenciado."

"O computador pessoal médio hoje em dia ter 10.000 vezes mais memória do que o IBM 7094, o maior computador no mundo no início da década de 1960."

Expansão dos Programas em Relação à Memória Disponível:

Parafraseando a Lei de Parkinson: “programas tendem a expandir-se a fim de preencher a memória disponível para contê-los”.

Idealização de Memória por Programadores:

"O que todo programador gostaria é de uma memória privada, infinitamente grande e rápida, que fosse não volátil também, isto é, não perdesse seus conteúdos quando faltasse energia elétrica."

Realidade Tecnológica Atual versus Idealização:

"Infelizmente, a tecnologia ainda não produz essas memórias no momento."

A memória

Hierarquia de Memórias em Computadores:

Descrição da hierarquia: "os computadores têm alguns megabytes de memória cache volátil, cara e muito rápida, alguns gigabytes de memória principal volátil de velocidade e custo médios, e alguns terabytes de armazenamento em disco em estado sólido ou magnético não volátil, barato e lento."

Função do Sistema Operacional na Gestão de Memória:

"É função do sistema operacional abstrair essa hierarquia em um modelo útil e então gerenciar a abstração."

O Papel do Gerenciador de Memória:

"A parte do sistema operacional que gerencia (parte da) hierarquia de memórias é chamada de gerenciador de memória."

Sua função: "gerenciar eficientemente a memória: controlar quais partes estão sendo usadas, alocar memória para processos quando eles precisam dela e liberá-la quando tiverem terminado."

Abordagem do Capítulo sobre Modelos de Gerenciamento de Memória:

"vários modelos diferentes de gerenciamento de memória, desde os muito simples aos altamente sofisticados."

A memória

Foco :

"O desse estar no modelo de memória principal do programador e como ela pode ser gerenciada.“

Progressão no Estudo dos Esquemas de Gerenciamento:

"Examinaremos primeiro os esquemas mais simples possíveis e então gradualmente avançaremos para os esquemas cada vez mais elaborados."

Sem abstração de memória

Conceito de Memória Física sem Abstração:

"Os primeiros computadores [...] não tinham abstração de memória. Cada programa apenas via a memória física."

Ilustração com a instrução: "MOV REGISTER1,1000 o computador apenas movia o conteúdo da memória física da posição 1000 para REGISTER1."

Modelos de Memória Física:

Diversas organizações de memória são possíveis, como "O sistema operacional pode estar na parte inferior da memória em RAM" ou "em ROM no topo da memória".

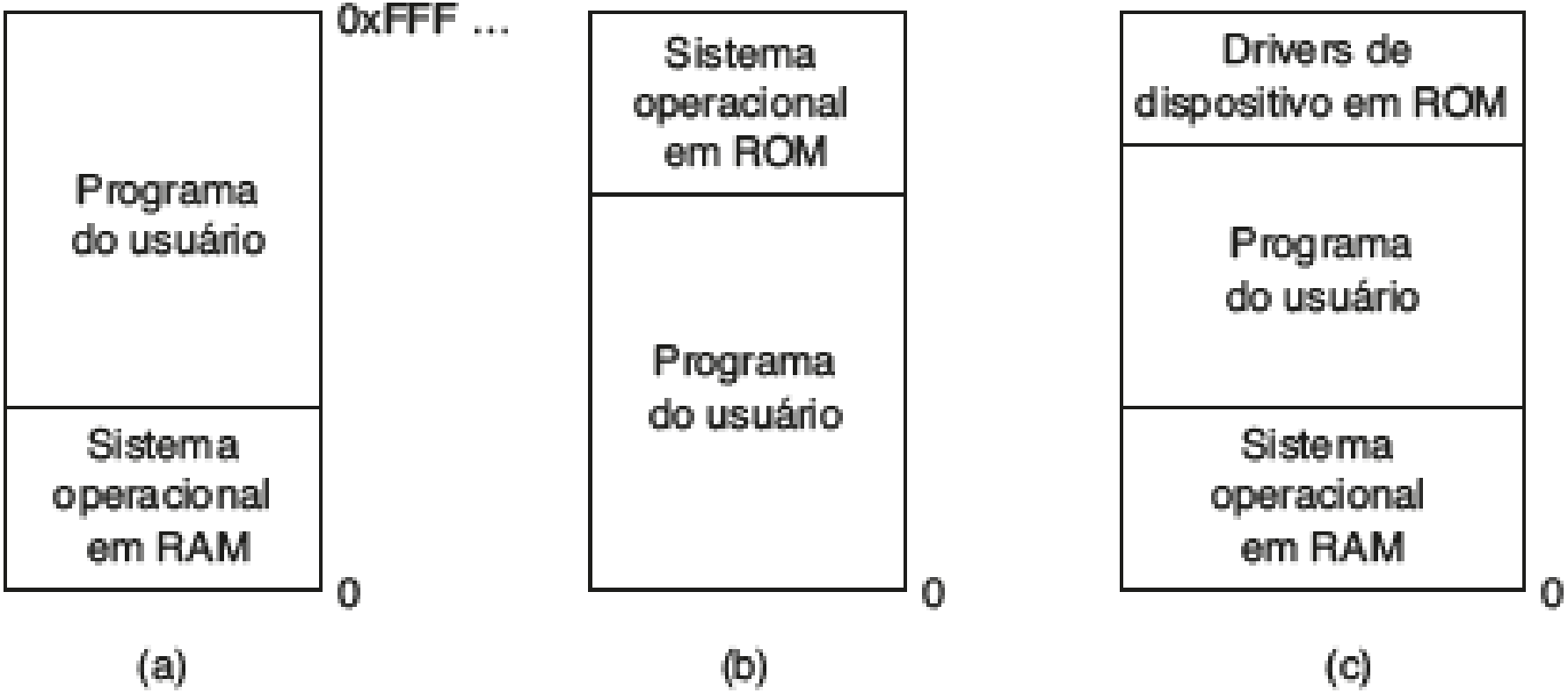
Limitações e Riscos:

"Os modelos (a) e (c) têm a desvantagem de que um erro no programa do usuário pode apagar por completo o sistema operacional."

"Geralmente apenas um processo de cada vez pode estar executando."

Sem abstração de memória

Conceito de Memória Física sem Abstração:



Sem abstração de memória

Executando Múltiplos Programas sem Abstração de Memória:

Descreve a possibilidade de executar vários programas: "O que um sistema operacional precisa fazer é salvar o conteúdo inteiro da memória em um arquivo de disco, então introduzir e executar o programa seguinte."

Desafios e Soluções Iniciais da IBM:

"Os primeiros modelos da IBM 360 solucionaram o problema" com uma abordagem de proteção de memória, mas enfrentaram limitações, como "ambos os programas referenciam a memória física absoluta".

Abordagem de Realocação Estática:

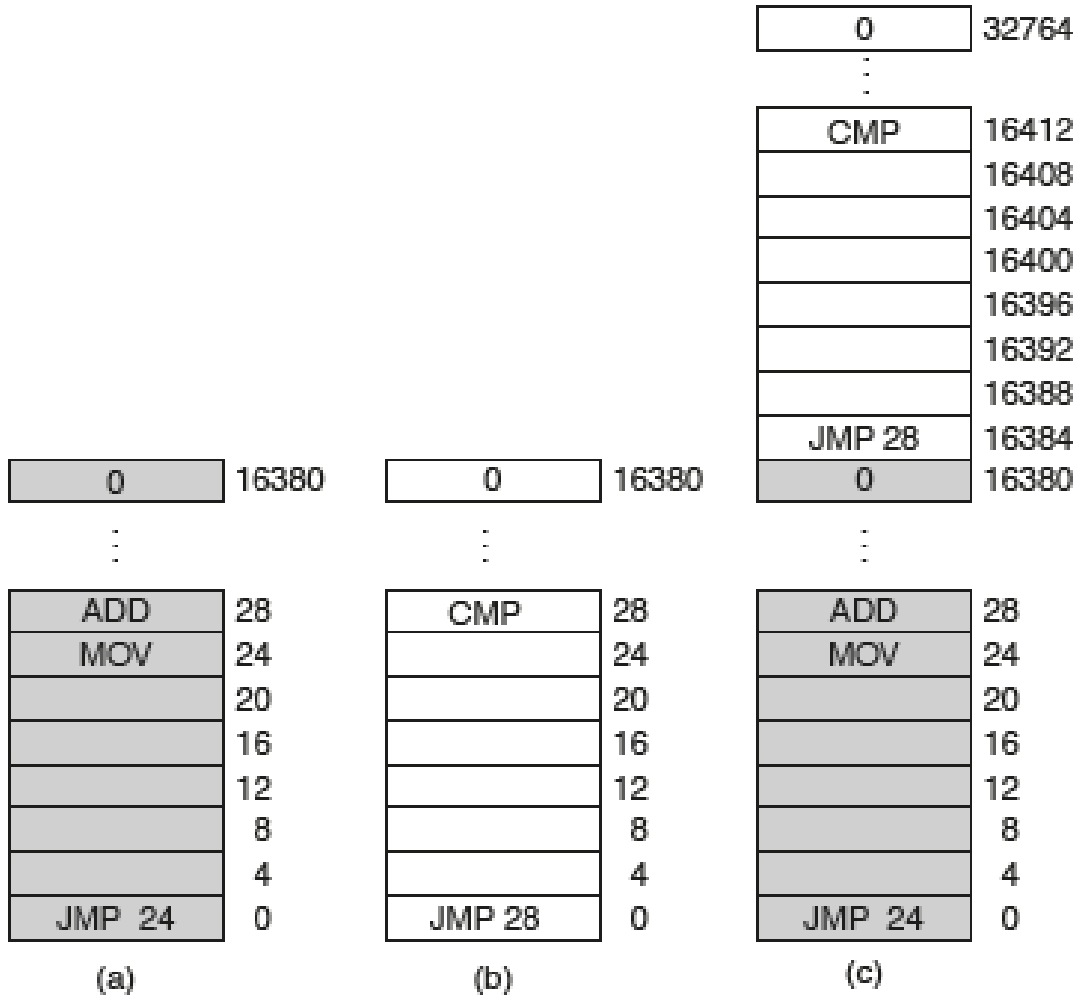
"O que o IBM 360 utilizou como solução temporária foi modificar o segundo programa dinamicamente enquanto o carregava na memória, usando uma técnica conhecida como realocação estática."

Abstração de Memória em Sistemas Embarcados:

"Embora o endereçamento direto de memória física seja apenas uma memória distante [...] a falta de uma abstração de memória ainda é comum em sistemas embarcados e de cartões inteligentes."

Sem abstração de memória

Executando Múltiplos Programas sem Abstração de Memória:



Processos e Swapping no Gerenciamento de Memória

Desvantagens da Exposição Direta da Memória Física:

"Expor a memória física a processos tem várias desvantagens importantes."
Riscos citados: "eles podem facilmente derrubar o sistema operacional, intencionalmente ou por acidente".

Desafio de Múltiplos Programas Executando Simultaneamente:

"É difícil ter múltiplos programas executando ao mesmo tempo" no modelo de memória física direta.

Espaço de Endereçamento como Solução:

Introdução do conceito: "Uma solução melhor é inventar uma nova abstração para a memória: o espaço de endereçamento."
Define o espaço de endereçamento: "Um espaço de endereçamento é o conjunto de endereços que um processo pode usar para endereçar a memória."

Registradores Base e Limite para Realocação Dinâmica:

Descrição da solução: "equipar cada CPU com dois registradores de hardware especiais, normalmente chamados de registradores base e registradores limite."
Funcionamento: "cada endereço de memória gerado automaticamente tem o conteúdo do registrador base adicionado a ele antes de ser enviado para a memória."

Processos e Swapping no Gerenciamento de Memória

Se a memória física do computador for grande o suficiente para armazenar todos os processos, os esquemas descritos até aqui bastarão de certa forma. Mas na prática, o montante total de RAM demandado por todos os processos é muitas vezes bem maior do que pode ser colocado na memória. Em sistemas típicos Windows, OS X ou Linux, algo como 50-100 processos ou mais podem ser iniciados tão logo o computador for ligado. Por exemplo, quando uma aplicação do Windows é instalada, ela muitas vezes emite comandos de tal forma que em inicializações subsequentes do sistema, um processo será iniciado somente para conferir se existem atualizações para as aplicações.

Um processo desses pode facilmente ocupar 5-10 MB de memória. Outros processos de segundo plano conferem se há e-mails, conexões de rede chegando e muitas outras coisas. E tudo isso antes de o primeiro programa do usuário ter sido iniciado. Programas sérios de aplicação do usuário, como o Photoshop, podem facilmente exigir 500 MB apenas para serem inicializados e muitos gigabytes assim que começam a processar dados. Em consequência, manter todos os processos na memória o tempo inteiro exige um montante enorme de memória e é algo que não pode ser feito se ela for insuficiente.

Processos e Swapping no Gerenciamento de Memória

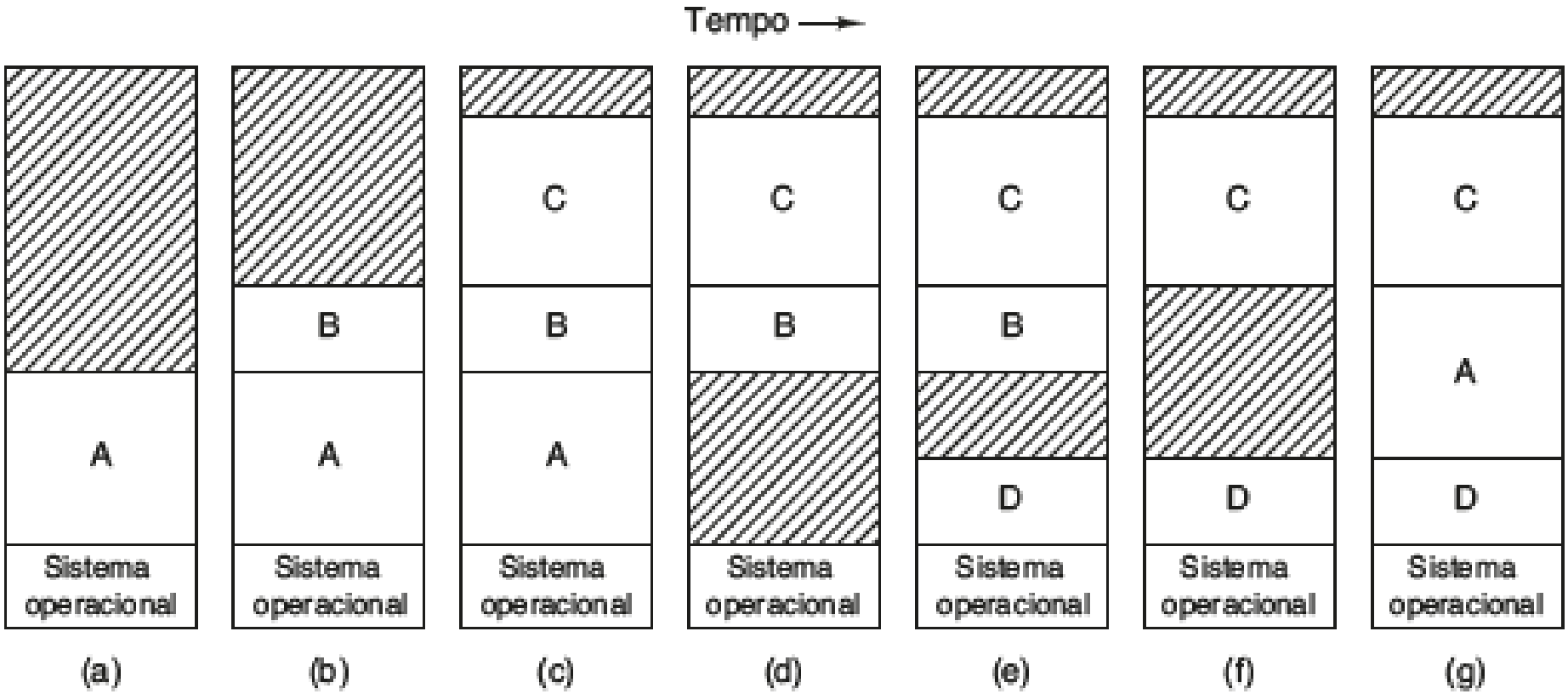
Duas abordagens gerais para lidar com a sobrecarga de memória foram desenvolvidas ao longo dos anos. A estratégia mais simples, chamada de swapping (troca de processos), consiste em trazer cada processo em sua totalidade, executá-lo por um tempo e então colocá-lo de volta no disco. Processos ociosos estão armazenados em disco em sua maior parte, portanto não ocupam qualquer memória quando não estão sendo executados (embora alguns "despertem" periodicamente para fazer seu trabalho, e então voltam a "dormir").

A outra estratégia, chamada de memória virtual, permite que os programas possam ser executados mesmo quando estão apenas parcialmente na memória principal. A seguir estudaremos a troca de processos; na Seção 3.3 examinaremos a memória virtual. A operação de um sistema de troca de processos está ilustrada na Figura 3.4. De início, somente o processo A está na memória. Então os processos B e C são criados ou trazidos do disco. Na Figura 3.4(d) o processo A é devolvido ao disco. Então o processo D é inserido e o processo B tirado. Por fim, o processo A volta novamente. Como A está agora em uma posição diferente, os endereços contidos nele devem ser realocados, seja pelo software quando ele é trazido ou (mais provável) pelo hardware durante a execução do programa.

Processos e Swapping no Gerenciamento de Memória

Por exemplo, registradores base e limite funcionariam bem aqui. Quando as trocas de processos criam múltiplos espaços na memória, é possível combiná-los em um grande espaço movendo todos os processos para baixo, o máximo possível. Essa técnica é conhecida como compactação de memória. Em geral ela não é feita porque exige muito tempo da CPU. Por exemplo, em uma máquina de 16 GB que pode copiar 8 bytes em 8 ns, ela levaria em torno de 16 s para compactar toda a memória. Um ponto que vale a pena considerar diz respeito a quanta memória deve ser alocada para um processo quando ele é criado ou trocado. Se os processos são criados com um tamanho fixo que nunca muda, então a alocação é simples: o sistema operacional aloca exatamente o que é necessário, nem mais nem menos. Se, no entanto, os segmentos de dados dos processos podem crescer, alocando dinamicamente memória

Processos e Swapping no Gerenciamento de Memória



Gerenciamento de Memória livre

Abordagens para Rastrear Uso de Memória:

"O sistema operacional deve gerenciá-la [a memória]."

Métodos mencionados: "há duas maneiras de se rastrear o uso de memória: mapas de bits e listas livres."

Gerenciamento de Memória com Mapas de Bits:

Definição: "Com um mapa de bits, a memória é dividida em unidades de alocação tão pequenas quanto umas poucas palavras e tão grandes quanto vários quilobytes."

Funcionamento do mapa: "Correspondendo a cada unidade de alocação há um bit no mapa de bits, que é 0 se a unidade estiver livre e 1 se ela estiver ocupada (ou vice-versa)."

Questões de Projeto Relacionadas ao Tamanho da Unidade de Alocação:

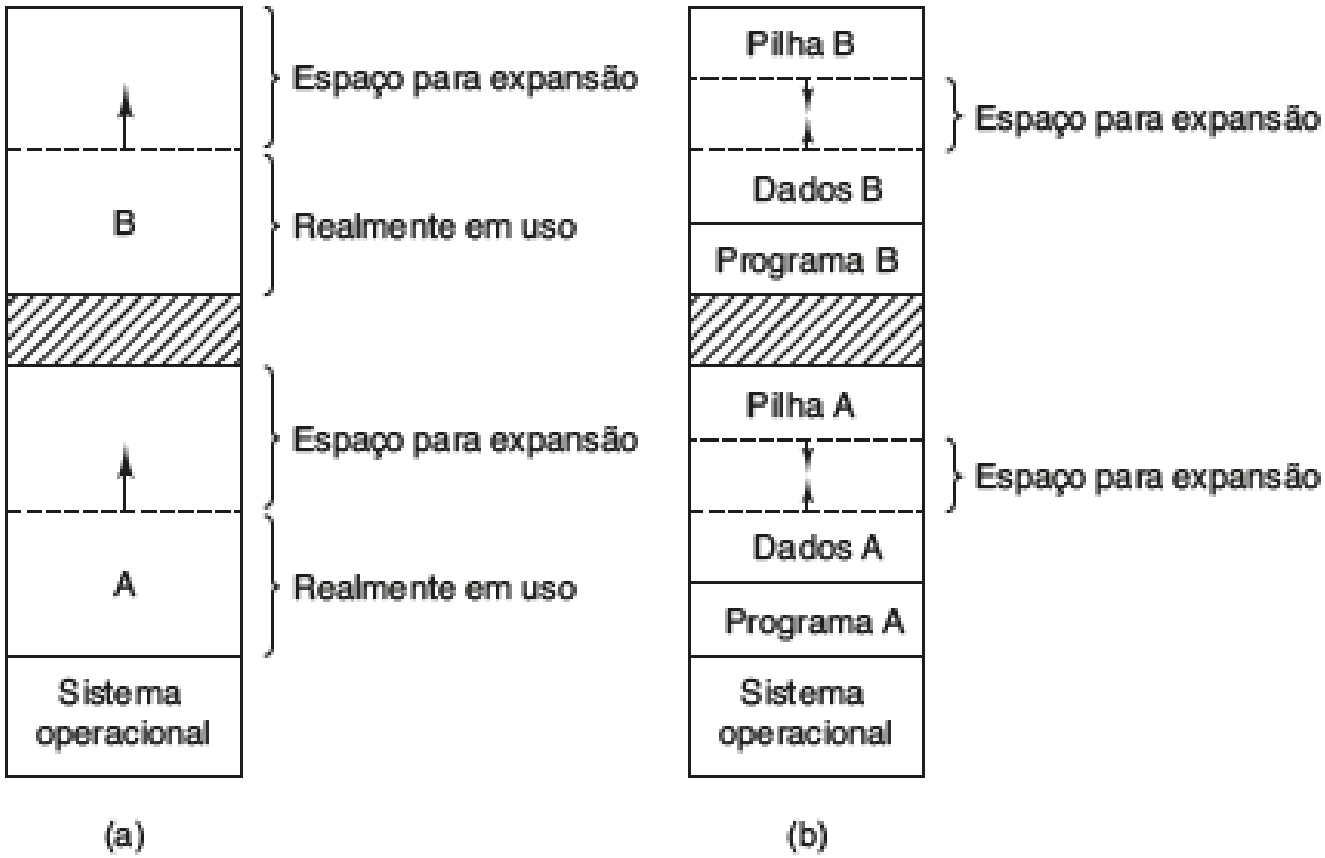
"O tamanho da unidade de alocação é uma importante questão de projeto."

Compromisso entre tamanho e eficiência: "Quanto menor a unidade de alocação, maior o mapa de bits."

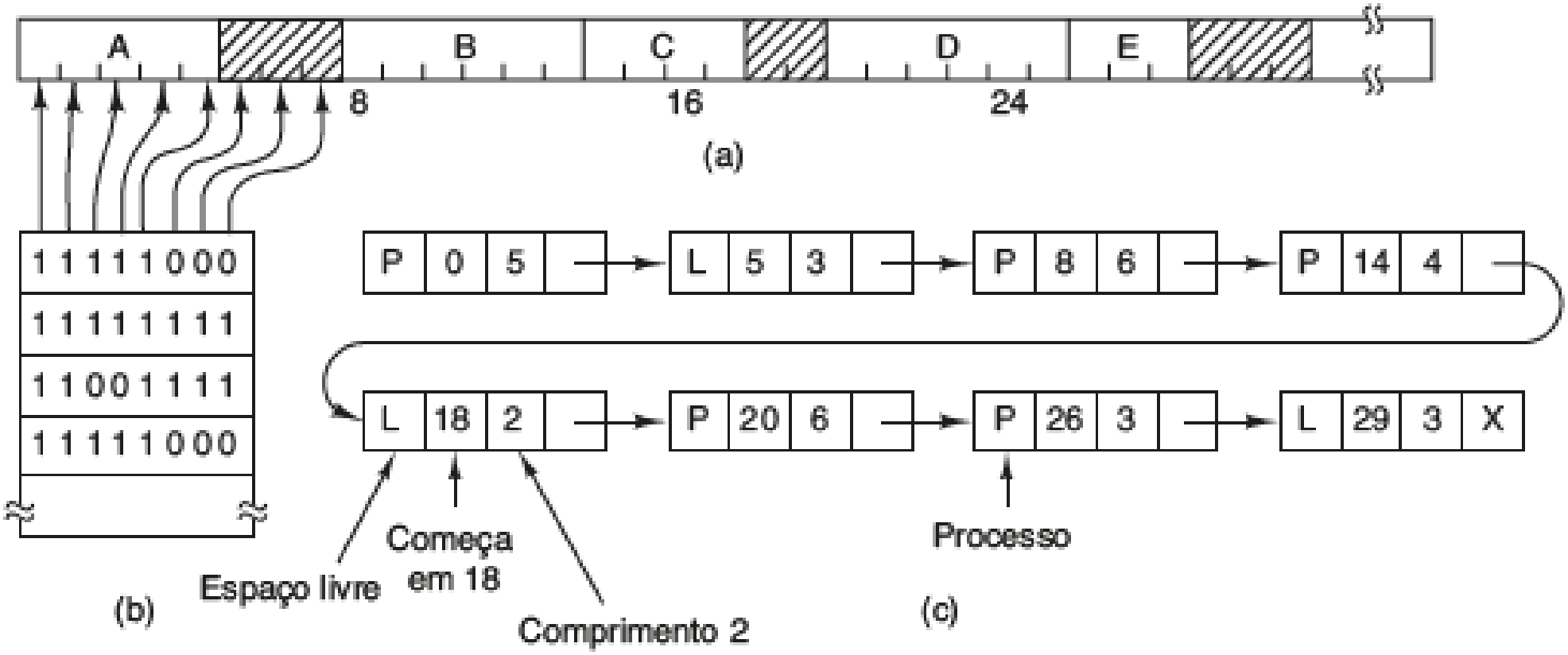
Desafios do Uso de Mapas de Bits:

"Procurar em um mapa de bits por uma sequência de um comprimento determinado é uma operação lenta."

Gerenciamento de Memória livre



Gerenciamento de Memória livre



Fundamentos de Memória Virtual e Paginação

Introdução à Memória Virtual:

"Embora os registradores base e os registradores limite possam ser usados para criar a abstração de espaços de endereçamento, há outro problema que precisa ser solucionado: gerenciar o bloatware."

Crescimento dos Tamanhos de Software:

"Os tamanhos dos softwares estão crescendo muito mais rapidamente."

Necessidade de Memória Virtual:

"Há uma necessidade de executar programas que são grandes demais para se encaixar na memória."

Limitações da Troca de Processos:

"A troca de processos não é uma opção atraente."

História dos Problemas de Memória:

"O problema dos programas maiores do que a memória existe desde o início da computação."

Solução de Sobreposições nos Anos 1960:

"Uma solução adotada nos anos 1960 foi dividir os programas em módulos pequenos, chamados de sobreposições."

Fundamentos de Memória Virtual e Paginação

Desvantagens das Sobreposições:

"Dividir programas grandes em módulos pequenos era uma tarefa cansativa, chata e propensa a erros."

Transição para a Memória Virtual:

"Não levou muito tempo para alguém pensar em passar todo o trabalho para o computador. O método encontrado ficou conhecido como memória virtual."

Conceito Básico de Memória Virtual:

"A ideia básica é que cada programa tem seu próprio espaço de endereçamento, o qual é dividido em blocos chamados de páginas."

Mapeamento de Páginas na Memória Física:

"Elas [páginas] são mapeadas na memória física, mas nem todas precisam estar na memória física ao mesmo tempo para executar o programa."

Funcionamento da Memória Virtual:

"Quando o programa referencia uma parte do espaço de endereçamento que está na memória física, o hardware realiza o mapeamento necessário rapidamente."

Fundamentos de Memória Virtual e Paginação

Tratamento de Referências a Páginas não Presentes:

"Quando o programa referencia uma parte de seu espaço de endereçamento que não está na memória física, o sistema operacional é alertado para ir buscar a parte que falta e reexecuta a instrução que falhou."

Generalização do Conceito de Registradores Base e Limite:

"De certa maneira, a memória virtual é uma generalização da ideia do registrador base e registrador limite."

Benefícios da Memória Virtual em Sistemas de Multiprogramação:

"A memória virtual funciona bem em um sistema de multiprogramação, com pedaços e partes de muitos programas na memória simultaneamente."

Utilização Eficiente da CPU em Memória Virtual:

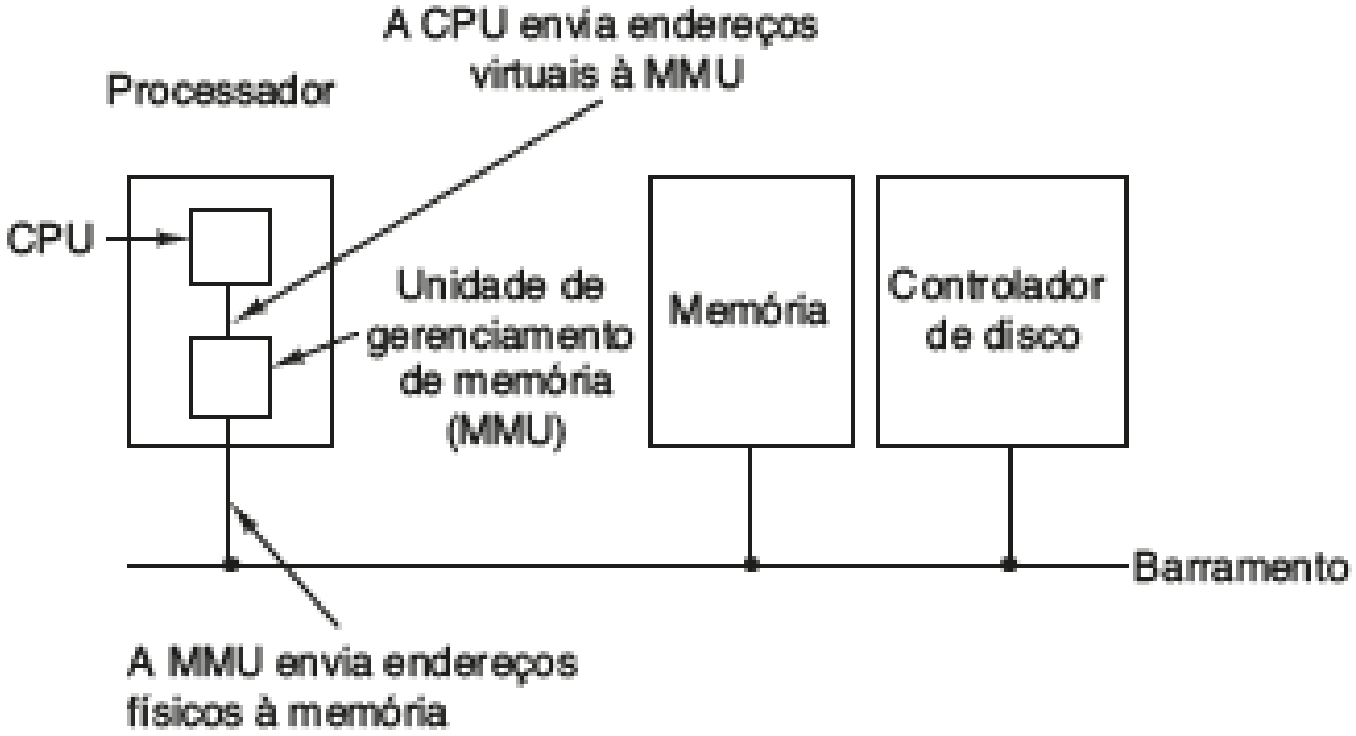
"Enquanto um programa está esperando que partes de si mesmo sejam lidas, a CPU pode ser dada para outro processo."

Fundamentos de Memória Virtual e Paginação

A maioria dos sistemas de memória virtual usa uma técnica chamada de paginação, que descreveremos agora. Em qualquer computador, programas referenciam um conjunto de endereços de memória. Quando um programa executa uma instrução como
MOV REG,1000
ele o faz para copiar o conteúdo do endereço de memória 1000 para REG (ou vice-versa, dependendo do computador). Endereços podem ser gerados usando indexação, registradores base, registradores de segmento e outras maneiras.

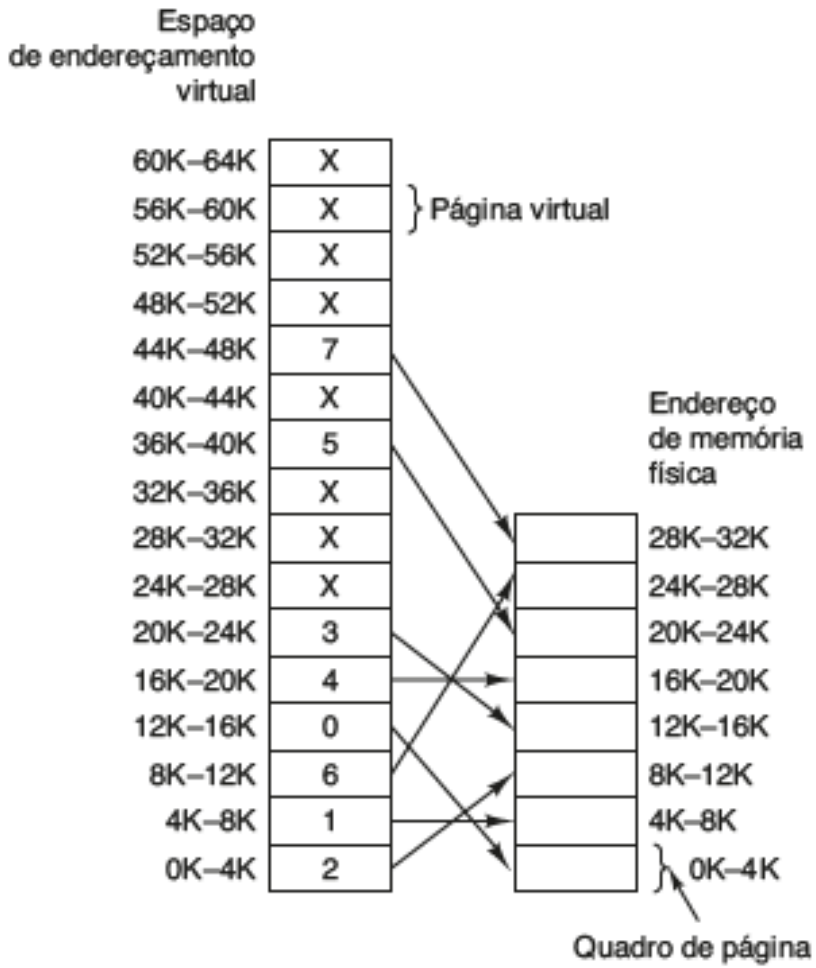
Esses endereços gerados por computadores são chamados de endereços virtuais e formam o espaço de endereçamento virtual. Em computadores sem memória virtual, o endereço virtual é colocado diretamente no barramento de memória e faz que a palavra de memória física com o mesmo endereço seja lida ou escrita. Quando a memória virtual é usada, os endereços virtuais não vão diretamente para o barramento da memória. Em vez disso, eles vão para uma MMU (Memory Management Unit — unidade de gerenciamento de memória) que mapeia os endereços virtuais em endereços de memória física, como ilustrado na Figura 3.8.

Fundamentos de Memória Virtual e Paginação



Fundamentos de Memória Virtual e Paginação

Um exemplo muito simples de como esse mapeamento funciona é mostrado na Figura 3.9. Nesse exemplo, temos um computador que gera endereços de 16 bits, de 0 a 64 K − 1. Esses são endereços virtuais. Esse computador, no entanto, tem apenas 32 KB de memória física. Então, embora programas de 64 KB possam ser escritos, eles não podem ser totalmente carregados na memória e executados. Uma cópia completa da imagem de núcleo de um programa, de até 64 KB, deve estar presente no disco, entretanto, de maneira que partes possam ser carregadas quando necessário. O espaço de endereçamento virtual consiste em unidades de tamanho fixo chamadas de páginas. As unidades correspondentes na memória física são chamadas de quadros de página. As páginas e os quadros de página são geralmente do mesmo tamanho.



Tabelas de Páginas e Otimizações de Paginação

Em uma implementação simples, o mapeamento de endereços virtuais em endereços físicos pode ser resumido como a seguir: o endereço virtual é dividido em um número de página virtual (bits mais significativos) e um deslocamento (bits menos significativos). Por exemplo, com um endereço de 16 bits e um tamanho de página de 4 KB, os 4 bits superiores poderiam especificar uma das 16 páginas virtuais e os 12 bits inferiores especificariam então o deslocamento de bytes (0 a 4095) dentro da página selecionada.

O número da página virtual é usado como um índice dentro da tabela de páginas para encontrar a entrada para essa página virtual. A partir da entrada da tabela de páginas, chega-se ao número do quadro (se ele existir).

Assim, o propósito da tabela de páginas é mapear as páginas virtuais em quadros de páginas. Matematicamente falando, a tabela de páginas é uma função, com o número da página virtual como argumento e o número do quadro físico como resultado. Usando o resultado dessa função, o campo da página virtual em um endereço virtual pode ser substituído por um campo de quadro de página, desse modo formando um endereço de memória física.

Tabelas de Páginas e Otimizações de Paginação

A estrutura de uma entrada na tabela de páginas é crucial para o funcionamento da memória virtual em computadores. Esta estrutura varia de acordo com a máquina, mas geralmente inclui campos comuns essenciais para a gestão da memória. Um exemplo típico de entrada na tabela de páginas pode ser de 32 bits, contendo várias informações importantes:

Número do Quadro de Página: Este é o campo mais importante, pois o objetivo principal do mapeamento de páginas é localizar este valor, que indica onde a página está localizada na memória física.

Bit Presente/Ausente: Este bit indica se a entrada é válida (1) ou não (0). Se estiver definido como 0, significa que a página virtual correspondente não está na memória física, e qualquer tentativa de acesso resultará em uma falta de página.

Bits de Proteção: Estes bits definem os tipos de acesso permitidos à página, podendo ser um único bit (leitura/escrever ou somente leitura) ou um conjunto de bits para controlar acessos de leitura, escrita e execução.

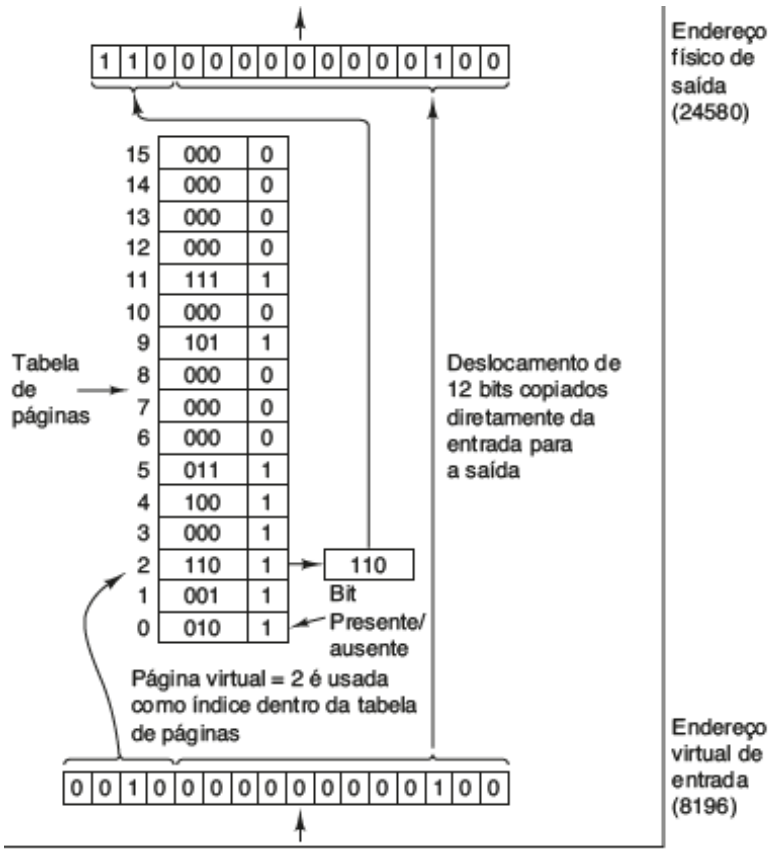
Bit Modificada: Automaticamente configurado quando uma página é escrita, esse bit é crucial para determinar se a página precisa ser atualizada no disco antes de ser descartada da memória, indicando se a página foi alterada ("suja") desde que foi carregada.

Bit Referenciada: Configurado sempre que a página é acessada (leitura ou escrita), esse bit ajuda o sistema operacional a escolher qual página substituir durante uma falta de página, indicando o uso recente da página.

Tabelas de Páginas e Otimizações de Paginação

Bit de Cache: Este bit permite desabilitar o mecanismo de cache para páginas específicas, o que é essencial para páginas que mapeiam registradores de dispositivos ao invés da memória regular, garantindo que o hardware não utilize cópias desatualizadas da cache.

A tabela de páginas é projetada para armazenar somente as informações necessárias para a tradução de endereços virtuais para físicos. Informações adicionais, como o endereço de disco da página quando não está na memória, são mantidas em tabelas de software dentro do sistema operacional, pois não são necessárias para a operação do hardware. A memória virtual, como abstração, divide o espaço de endereçamento virtual em páginas e mapeia cada uma delas para quadros de página na memória física, ou opta por não mapeá-las temporariamente, criando uma nova abstração da memória física semelhante à forma como um processo é uma abstração do processador físico (CPU).



Algoritmos de Substituição de Páginas: Introdução e Otimizações

Os algoritmos de substituição de páginas são métodos usados por sistemas operacionais para gerenciar a memória virtual, especialmente quando ocorre uma falta de página. Uma falta de página acontece quando um programa tenta acessar dados que não estão atualmente na memória principal. Para resolver isso, o sistema operacional precisa escolher uma página na memória para ser removida e substituída pela página necessária. Este processo é essencial porque a capacidade de memória física é limitada, e não é possível manter todas as páginas de todos os programas na memória ao mesmo tempo.
Há vários fatores a serem considerados na escolha de qual página remover:

Modificação da Página: Se a página a ser removida foi modificada (suja) durante sua estadia na memória, ela deve ser reescrita no disco para atualizar sua cópia. Caso contrário, se não foi modificada (limpa), pode ser simplesmente descartada sem a necessidade de reescrever, já que sua cópia no disco está atualizada.
Frequência de Uso: Preferencialmente, as páginas que são menos usadas devem ser escolhidas para remoção. Remover uma página frequentemente usada pode resultar em um custo extra, pois ela pode precisar ser rapidamente recarregada.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Outras Aplicações: Os algoritmos de substituição de páginas também são aplicados em outras áreas da computação, como a gestão de caches de memória em computadores e servidores da web.

Páginas Limpa ou Suja: As páginas podem estar limpas ou sujas, e isso afeta a decisão de qual página substituir.

Propriedade da Página: Uma questão adicional é se a página a ser removida deve pertencer ao processo que causou a falta de página ou se pode ser de outro processo. Isso pode levar a limitar o número de páginas disponíveis para cada processo ou permitir uma alocação mais flexível.

Esses algoritmos são fundamentais para manter a eficiência e o desempenho do sistema operacional, equilibrando a utilização da memória física limitada com as demandas dos programas em execução.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo FIFO:
O algoritmo de substituição de páginas "primeiro a entrar, primeiro a sair" (First In, First Out - FIFO) é um método simples e de baixo custo usado pelos sistemas operacionais para gerenciar a memória virtual. Este algoritmo funciona seguindo a lógica de que as páginas que entraram primeiro na memória são as primeiras a serem removidas quando é necessário espaço para novas páginas.

Para ilustrar o funcionamento do FIFO, pode-se comparar com a gestão de estoque em um supermercado. Imagine um supermercado com espaço limitado nas prateleiras, que só pode exibir um número fixo de produtos diferentes. Quando um novo produto (como um iogurte orgânico inovador) é introduzido e precisa ser estocado, o supermercado precisa remover um produto antigo para liberar espaço. Uma abordagem seria remover o produto que está nas prateleiras há mais tempo, assumindo que ele é o menos interessante para os clientes.

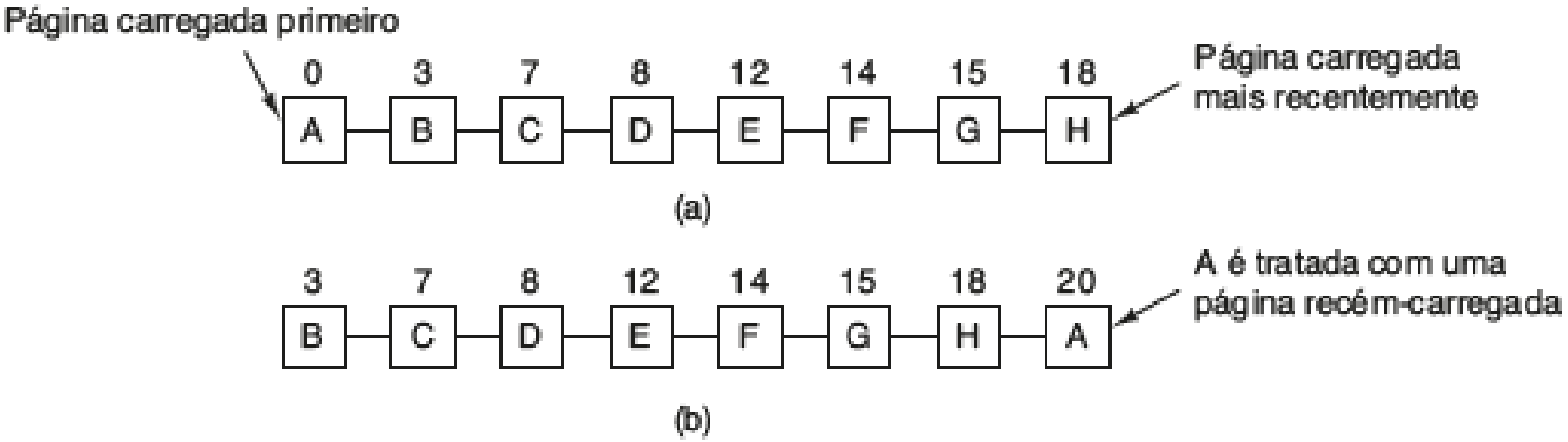
Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo FIFO:
Aplicando este conceito à memória do computador, o sistema operacional mantém uma lista encadeada das páginas que estão atualmente na memória, ordenadas pela ordem de chegada. A página que está na memória há mais tempo (a primeira da lista) é removida quando ocorre uma falta de página, e a nova página é adicionada ao final da lista.

No entanto, como no exemplo do supermercado, onde remover o produto mais antigo pode significar descartar itens ainda úteis (como farinha ou sal), o algoritmo FIFO em computadores pode também remover páginas que ainda são úteis. Este é um problema significativo do FIFO, pois a página mais antiga pode estar em uso ativo. Por essa razão, o algoritmo FIFO em sua forma mais pura raramente é usado em sistemas de gerenciamento de memória modernos, que normalmente empregam métodos mais sofisticados para determinar quais páginas remover.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo FIFO:



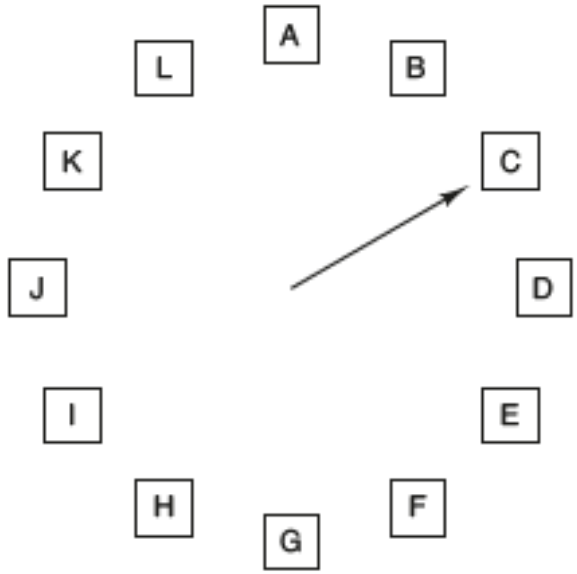
Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo relógio:
Embora segunda chance seja um algoritmo razoável, ele é desnecessariamente ineficiente, pois ele está sempre movendo páginas em torno de sua lista. Uma abordagem melhor é manter todos os quadros de páginas em uma lista circular na forma de um relógio, como mostrado na Figura 3.16. Um ponteiro aponta para a página mais antiga.

Quando ocorre uma falta de página, a página indicada pelo ponteiro é inspecionada. Se o bit R for 0, a página é removida, a nova página é inserida no relógio em seu lugar, e o ponteiro é avançado uma posição. Se R for 1, ele é zerado e o ponteiro avançado para a próxima página. Esse processo é repetido até que a página seja encontrada com R = 0. Sem muita surpresa, esse algoritmo é chamado de relógio.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo relógio:



Quando ocorre uma falta de página, a página indicada pelo ponteiro é inspecionada. A ação executada depende do bit R:

R = 0: Remover a página
R = 1: Zerar R e avançar o ponteiro

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo LRU:

O algoritmo de simulação do LRU (Least Recently Used) em software, conhecido como algoritmo de envelhecimento (Aging), é uma adaptação prática do LRU que pode ser implementada em sistemas onde o hardware não suporta diretamente o LRU. Esse algoritmo visa aproximar a funcionalidade do LRU utilizando uma estratégia de contagem e envelhecimento das páginas na memória.

Funcionamento e Aplicação:

Contadores Associados às Páginas: Cada página na memória é associada a um contador de software, inicializado com zero. A cada interrupção de relógio (tick do sistema), o sistema operacional atualiza esses contadores.

Atualização dos Contadores: Durante a interrupção de relógio, os contadores são deslocados um bit à direita, e o bit de referência (R) da página é adicionado ao bit mais à esquerda do contador. O bit R é 0 ou 1, dependendo se a página foi referenciada ou não desde a última interrupção de relógio.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo LRU:

Envelhecimento das Páginas: Esse processo de deslocamento de bits e adição do bit R simula o envelhecimento das páginas. Páginas que não são referenciadas por um período mais longo acabam tendo contadores mais baixos.

Seleção de Páginas para Remoção: Quando ocorre uma falta de página, o sistema operacional escolhe a página com o menor valor no contador para substituição. Isso tende a remover páginas que foram menos usadas recentemente.

Limitações do Algoritmo:

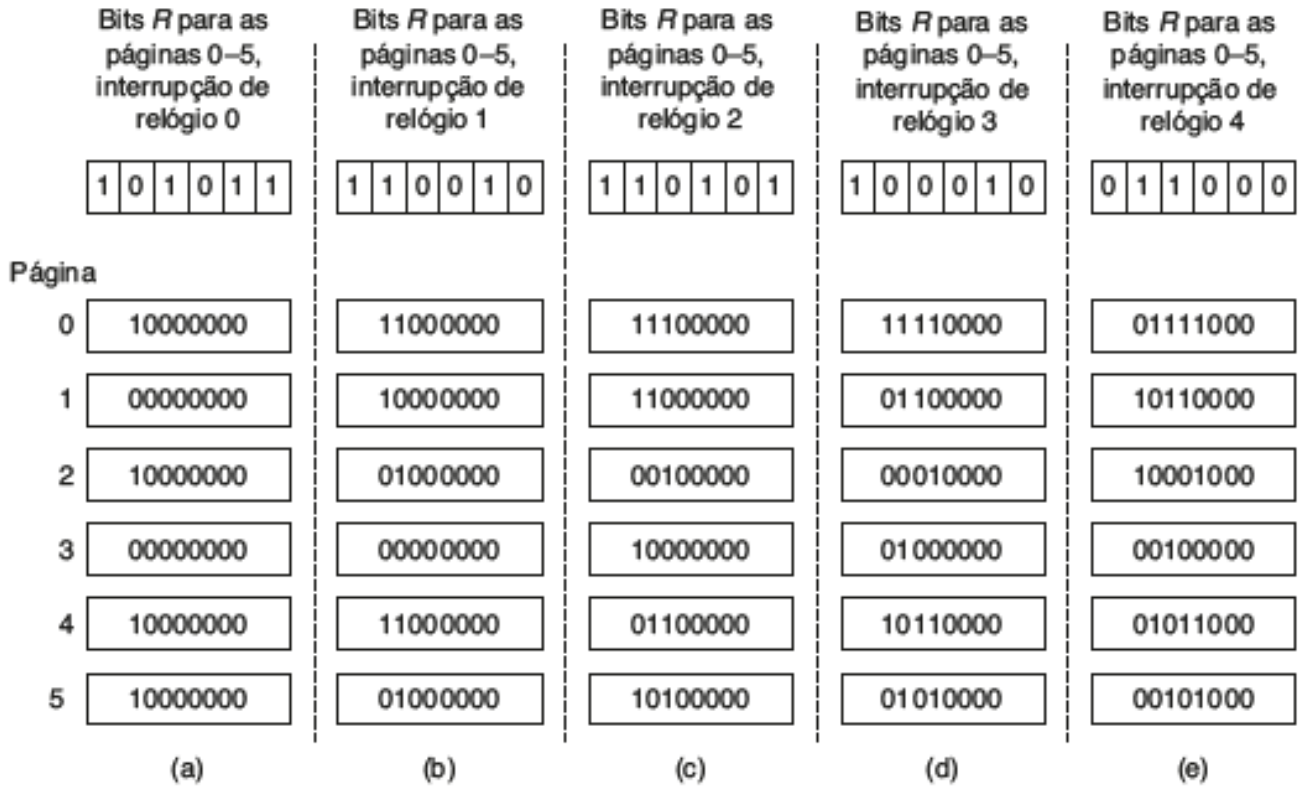
Incapacidade de distinguir exatamente a ordem das referências dentro de um intervalo de tempo. Por exemplo, não é possível saber qual página entre duas foi referenciada por último.

O número finito de bits nos contadores limita a capacidade de distinguir o histórico de referências distantes no tempo.

Apesar dessas limitações, o algoritmo de envelhecimento é uma aproximação eficaz do LRU em sistemas onde não é possível implementar o LRU puro por meio de hardware. Ele é útil para sistemas operacionais que precisam gerenciar a memória de forma eficiente, equilibrando a necessidade de reter páginas importantes na memória e remover páginas menos utilizadas para otimizar o desempenho do sistema

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo LRU:



Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo WSclock:

O algoritmo WSClock é uma melhoria do algoritmo básico de substituição de páginas que combina características do algoritmo de relógio com conceitos de conjunto de trabalho. Este algoritmo é conhecido por sua eficiência e simplicidade de implementação, sendo amplamente utilizado na prática. O funcionamento do WSClock pode ser explicado da seguinte maneira:

Estrutura de Dados: O algoritmo utiliza uma lista circular de quadros de páginas, semelhante ao algoritmo do relógio. Cada entrada na lista contém informações como o momento do último uso da página, além dos bits R (referenciado) e M (modificado).

Exame das Páginas: A cada falta de página, a página apontada na lista circular é examinada. Se o bit R for 1, indica que a página foi usada recentemente, então o bit R é zerado, e o ponteiro avança para a próxima página.

Seleção de Páginas para Remoção: Se a página apontada tem R = 0 e é "velha" (idade maior do que um valor τ) e limpa (não modificada), ela é considerada fora do conjunto de trabalho e pode ser removida para dar espaço à nova página.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo WSclock:

Tratamento de Páginas Sujas: Se a página com R = 0 está suja, ela não pode ser removida imediatamente, pois precisa ser reescrita no disco. Em vez de interromper o processo para esta operação, a escrita é escalonada, e o ponteiro avança para examinar a próxima página.

Limitação de Escritas no Disco: Para evitar tráfego excessivo de disco, um limite pode ser definido para o número de páginas que podem ser reescritas por ciclo de relógio.

Completação do Ciclo: Se o ponteiro completa um ciclo completo sem encontrar uma página adequada para remoção:

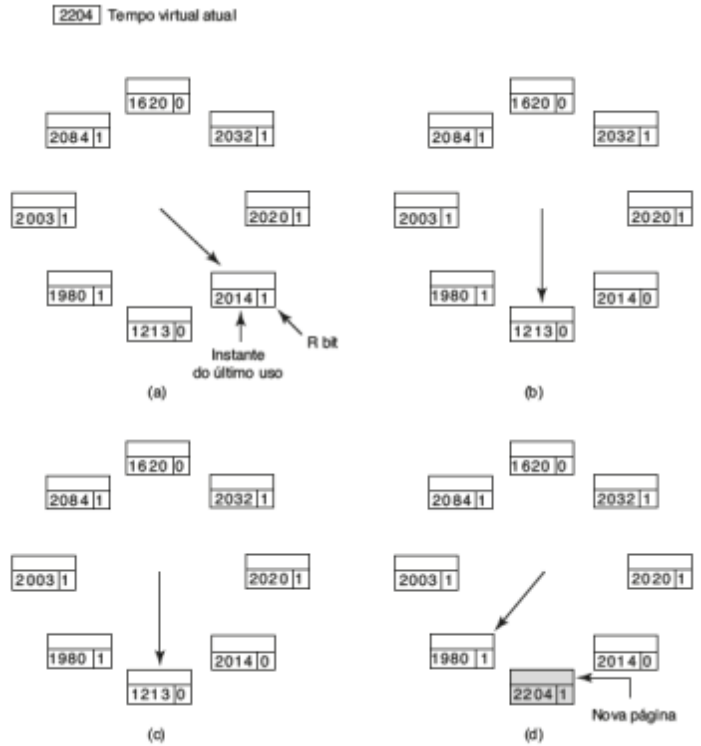
Se pelo menos uma escrita foi escalonada, o ponteiro continua se movendo, procurando uma página limpa que possa ser removida.

Se nenhuma escrita foi escalonada, significa que todas as páginas estão em uso. Neste caso, qualquer página limpa pode ser reivindicada, ou a página atualmente apontada é escolhida para ser reescrita e removida.

O algoritmo WSClock, portanto, fornece uma maneira eficiente de gerenciar a substituição de páginas na memória, minimizando o número de escritas no disco e considerando tanto a frequência de uso das páginas quanto se elas foram modificadas ou não.

Algoritmos de Substituição de Páginas: Introdução e Otimizações

Algoritmo WSclock:



Resumo e Comparação dos Algoritmos de Substituição de Página

O algoritmo ótimo remove a página que será referenciada por último. Infelizmente, não há uma maneira para determinar qual página será essa, então, na prática, esse algoritmo não pode ser usado. No entanto, ele é útil como uma medida-padrão pela qual outros algoritmos podem ser mensurados. O algoritmo NRU divide as páginas em quatro classes, dependendo do estado dos bits R e M. Uma página aleatória da classe de ordem mais baixa é escolhida. Esse algoritmo é fácil de implementar, mas é muito rudimentar. Há outros melhores.

O algoritmo FIFO controla a ordem pela qual as páginas são carregadas na memória mantendo-as em uma lista encadeada. Remover a página mais antiga, então, torna-se trivial, mas essa página ainda pode estar sendo usada, de maneira que o FIFO é uma má escolha.

O algoritmo segunda chance é uma modificação do FIFO que confere se uma página está sendo usada antes de removê-la. Se ela estiver, a página é poupada. Essa modificação melhora muito o desempenho. O algoritmo do relógio é simplesmente uma implementação diferente do algoritmo segunda chance. Ele tem as mesmas propriedades de desempenho, mas leva um pouco menos de tempo para executar o algoritmo.

Resumo e Comparação dos Algoritmos de Substituição de Página

O LRU é um algoritmo excelente, mas não pode ser implementado sem um hardware especial. Se o hardware não estiver disponível, ele não pode ser usado.

O NFU é uma tentativa rudimentar, não muito boa, de aproximação do LRU. No entanto, o algoritmo do envelhecimento é uma aproximação muito melhor do LRU e pode ser implementado de maneira eficiente. Trata-se de uma boa escolha. Os últimos dois algoritmos usam o conjunto de trabalho. O algoritmo do conjunto de trabalho proporciona um desempenho razoável, mas é de certa maneira caro de ser implementado.

O WSClock é uma variante que não apenas proporciona um bom desempenho, como também é eficiente de ser implementado. Como um todo, os dois melhores algoritmos são o do envelhecimento e o WSClock. Eles são baseados no LRU e no conjunto de trabalho, respectivamente. Ambos proporcionam um bom desempenho de paginação e podem ser implementados eficientemente. Alguns outros bons algoritmos existem, mas esses dois provavelmente são os mais importantes na prática.

Questões de Projeto em Sistemas de Paginação

Para projetar um sistema e fazê-lo funcionar bem, você precisa saber bem mais. É como a diferença entre saber como mover a torre, o cavalo e o bispo, e outras peças do xadrez, e ser um bom jogador. Os projetistas de sistemas operacionais têm de considerar cuidadosamente vários fatores a fim de obter um bom desempenho de um sistema de paginação.

Questões de Projeto em Sistemas de Paginação

Políticas de alocação local versus global

As políticas de alocação de memória em sistemas operacionais, local versus global, referem-se a como a memória é distribuída entre processos concorrentes. A diferença chave entre as duas abordagens está em como os quadros de páginas são alocados e gerenciados:

Alocação Local: Nesta abordagem, cada processo recebe uma alocação fixa de quadros de páginas na memória. Quando ocorre uma falta de página, o algoritmo de substituição de páginas considera apenas as páginas alocadas para esse processo específico. Este método pode levar a ultrapaginação (trocas excessivas de páginas de/para o disco) se o conjunto de trabalho do processo crescer além da alocação fixa. Além disso, pode haver desperdício de memória se o conjunto de trabalho diminuir e as páginas não forem realocadas para outros processos.

Alocação Global: Neste método, os quadros de páginas são alocados dinamicamente entre todos os processos em execução. Isso significa que o número de quadros de páginas atribuídos a cada processo pode variar ao longo do tempo. Algoritmos globais geralmente fornecem melhor desempenho, pois podem adaptar-se às variações no tamanho do conjunto de trabalho dos processos. Se um processo precisa de mais páginas devido ao aumento do seu conjunto de trabalho, ele pode receber páginas de outros processos que não as estão utilizando ativamente.

Questões de Projeto em Sistemas de Paginação

Políticas de alocação local versus global

Gerenciamento Dinâmico da Alocação: Em um sistema que utiliza alocação global, é crucial gerenciar continuamente quantos quadros de páginas são atribuídos a cada processo. Métodos como monitorar o tamanho do conjunto de trabalho ou usar o algoritmo de frequência de faltas de página (PFF) podem ajudar a ajustar dinamicamente a alocação de páginas.

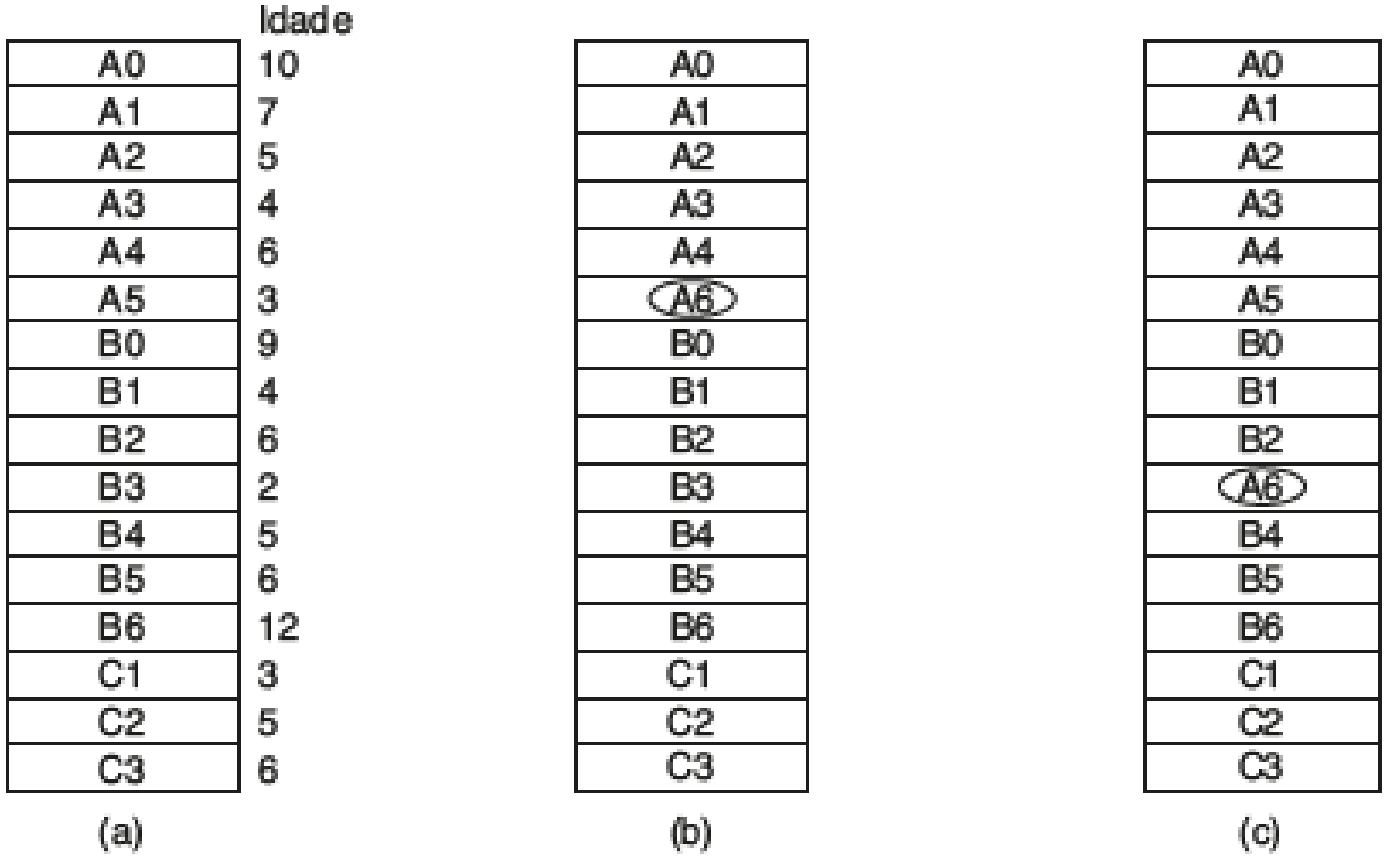
Limitações e Considerações: Ambas as abordagens têm suas limitações e situações onde são mais eficazes. A alocação local pode ser mais simples, mas pode não utilizar eficientemente a memória disponível. Por outro lado, a alocação global requer um gerenciamento mais complexo, mas pode se adaptar melhor às necessidades variáveis dos processos.

Aplicação aos Algoritmos de Substituição de Páginas: Alguns algoritmos, como FIFO e LRU, podem funcionar tanto em contextos locais quanto globais. Contudo, algoritmos como o conjunto de trabalho e WSClock são mais específicos para processos individuais e se encaixam melhor em um contexto de alocação local.

Em resumo, a escolha entre alocação local e global depende do contexto específico do sistema operacional e da natureza dos processos em execução. Uma abordagem global, por ser mais dinâmica e adaptável, é geralmente preferida em sistemas modernos, onde a variabilidade do uso de memória é uma consideração importante.

Questões de Projeto em Sistemas de Paginação

Políticas de alocação local versus global



Questões de Projeto em Sistemas de Paginação

Controle de carga

O controle de carga no processo de paginação é uma estratégia utilizada para gerenciar a demanda por memória em sistemas operacionais, principalmente em situações de ultrapaginação. A ultrapaginação ocorre quando os conjuntos de trabalho combinados de todos os processos em execução excedem a capacidade da memória, levando a um excesso de trocas de páginas de/para o disco (swap). Para lidar com isso, o sistema operacional pode adotar as seguintes medidas:

Identificação de Ultrapaginação: Quando o algoritmo de frequência de faltas de página (PFF) indica que vários processos precisam de mais memória e nenhum pode liberar memória, isso é um sinal de ultrapaginação.

Redução de Processos Ativos: Uma solução é reduzir o número de processos competindo por memória. Isso pode ser feito transferindo alguns processos para o disco e liberando suas páginas de memória, redistribuindo-as para os processos restantes que estão ultrapaginando.

Troca de Processos para o Disco: Essa técnica envolve mover processos inteiros para o disco temporariamente. Ao fazer isso, o sistema operacional pode aliviar a pressão sobre a memória.

Alternância de Processos: Periodicamente, alguns processos podem ser trazidos de volta da memória secundária para a principal, enquanto outros são transferidos para o disco, gerenciando assim a carga de trabalho da memória.

Questões de Projeto em Sistemas de Paginação

Controle de carga

Escalonamento de Dois Níveis: Esta abordagem combina a ideia de trocar processos com o escalonamento de curto prazo, onde apenas um número suficiente de processos é mantido na memória para manter a frequência de faltas de páginas em um nível aceitável.

Consideração do Grau de Multiprogramação: É importante equilibrar o número de processos na memória principal. Muitos processos podem levar à ultrapaginação, enquanto poucos podem resultar em ociosidade da CPU.

Seleção de Processos para Troca: Ao decidir quais processos transferir para o disco, o sistema operacional pode considerar fatores como o tamanho do processo, frequência de paginação, e se o processo é limitado pela CPU ou por E/S.

Em resumo, o controle de carga no processo de paginação envolve gerenciar dinamicamente a quantidade de processos na memória principal para evitar ultrapaginação, mantendo ao mesmo tempo a eficiência do processador e a velocidade geral do sistema.

Questões de Projeto em Sistemas de Paginação

Tamanho da página

O tamanho da página é um parâmetro crucial no gerenciamento de memória de um sistema operacional, influenciando diretamente a eficiência da memória e do desempenho do sistema. Aqui está um resumo de como o tamanho da página afeta o gerenciamento de memória e como é determinado:

Escolha do Tamanho da Página: O sistema operacional pode definir o tamanho da página, mesmo que o hardware suporte um tamanho específico. Por exemplo, páginas de hardware de 4096 bytes podem ser tratadas como páginas de 8192 bytes pelo sistema operacional, combinando pares de páginas.

Fragmentação Interna: Com páginas grandes, há um risco maior de fragmentação interna, onde o espaço extra na última página de um segmento é desperdiçado. Isso sugere um tamanho de página menor para minimizar o desperdício.

Tamanho da Tabela de Páginas: Páginas menores resultam em um maior número de páginas necessárias para um programa, o que aumenta o tamanho da tabela de páginas. Isso pode afetar negativamente o desempenho, especialmente em sistemas com espaço limitado de tabela de páginas, como o Translation Lookaside Buffer (TLB).

Desempenho de E/S: Transferências de disco são normalmente feitas uma página por vez. Páginas menores podem levar a mais operações de E/S, o que pode ser menos eficiente do que transferir páginas maiores, que gastam menos tempo em operações de posicionamento.

Questões de Projeto em Sistemas de Paginação

Tamanho da página

Uso do TLB: Páginas maiores podem ser benéficas para o desempenho do TLB, pois ocupam menos entradas do TLB. Isso é importante, pois as entradas do TLB são limitadas e críticas para o desempenho.

Cálculo do Tamanho Ótimo da Página: O tamanho ótimo da página pode ser calculado considerando o equilíbrio entre o custo da tabela de páginas e a fragmentação interna. Uma fórmula que leva em conta esses fatores sugere que o tamanho ideal da página é proporcional à raiz quadrada do produto do tamanho médio do processo pelo tamanho da entrada da tabela de páginas.

Aplicação Prática: Sistemas operacionais modernos tendem a usar tamanhos de páginas que variam de acordo com as necessidades específicas. Páginas grandes podem ser usadas para o núcleo do sistema, enquanto páginas menores podem ser mais adequadas para processos do usuário.

O tamanho da página é uma consideração importante no gerenciamento de memória, e a escolha ideal depende de um equilíbrio entre vários fatores, incluindo a minimização da fragmentação interna, a eficiência da tabela de páginas e do TLB, e o desempenho das operações de E/S.

Questões de Projeto em Sistemas de Paginação

Espaços separados de instruções e dados

Espaço Único de Endereçamento: "A maioria dos computadores tem um único espaço de endereçamento tanto para programas quanto para dados." Neste modelo, tudo funciona bem, desde que o espaço de endereçamento seja suficientemente grande.

Espaços de Endereçamento Separados para Instruções e Dados: Em sistemas com espaço de endereçamento limitado, "uma solução é ter dois espaços de endereçamento diferentes para instruções (código do programa) e dados, chamados de espaço I e espaço D, respectivamente." Cada um desses espaços varia de 0 até um valor máximo, geralmente 2^16 - 1 ou 2^32 - 1.

Requisitos do Ligador: "O ligador precisa saber quando endereços I e D separados estão sendo usados" para realizar a realocação apropriada dos dados.

Paginação Independente: "Cada um [dos espaços de endereçamento] tem sua própria tabela de páginas, com seu próprio mapeamento de páginas virtuais para quadros de página física." Isso permite que instruções e dados sejam paginados independentemente.

Uso do Hardware: Dependendo de se uma instrução ou dado está sendo acessado, o hardware utiliza a tabela de páginas correspondente ao espaço I ou D.

Questões de Projeto em Sistemas de Paginação

Espaços separados de instruções e dados

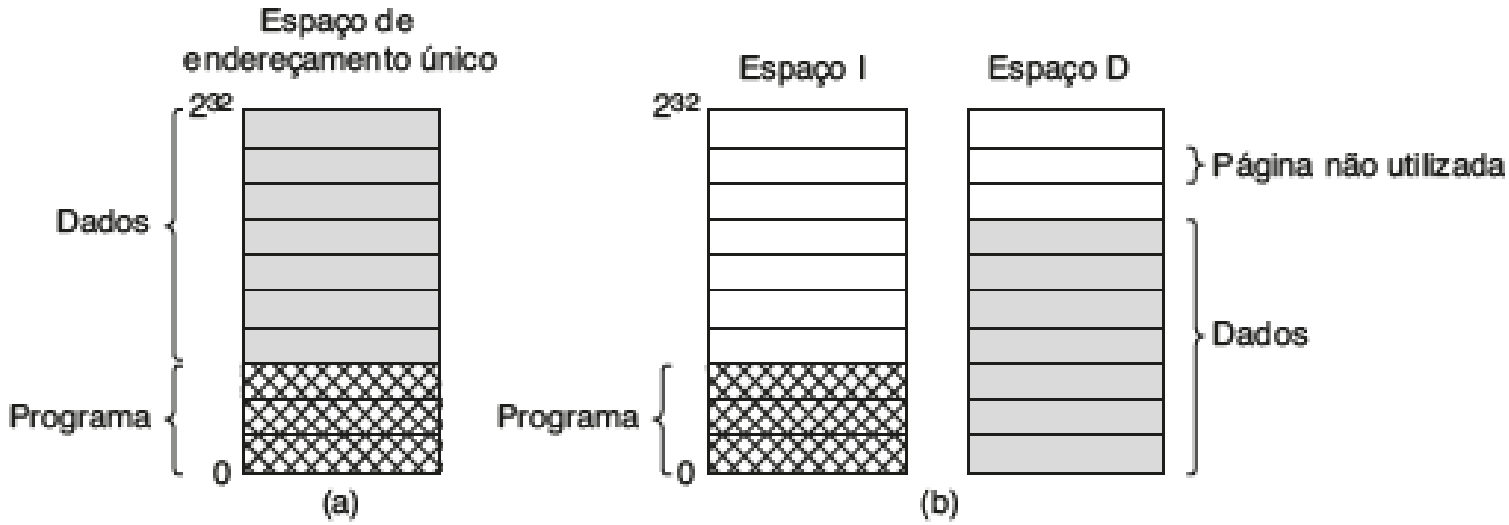
Simplicidade para o Sistema Operacional: "Ter espaços de I e D separados não apresenta quaisquer complicações especiais para o sistema operacional" e efetivamente duplica o espaço de endereçamento disponível.

Aplicação em Espaços de Cache: Embora os espaços de endereçamento sejam amplos, "espaços de I e D separados são comuns" em caches L1, onde a memória ainda é considerada escassa.

Essa abordagem de espaços de endereçamento separados para instruções e dados foi pioneiramente implementada no PDP-11 e continua a ser uma prática comum, especialmente para otimizar o uso da memória em caches L1.

Questões de Projeto em Sistemas de Paginação

Espaços separados de instruções e dados



Questões de Projeto em Sistemas de Paginação

Páginas compartilhadas

O compartilhamento de páginas em sistemas de multiprogramação é uma questão de projeto importante e pode ser implementado de várias maneiras:

1. Eficiência do Compartilhamento: "É claramente mais eficiente compartilhar as páginas, para evitar ter duas cópias da mesma página na memória ao mesmo tempo". Isso é particularmente útil quando vários usuários estão executando o mesmo programa ou usando a mesma biblioteca.
2. Diferenciação entre Páginas Compartilháveis e Não Compartilháveis: "Nem todas as páginas são compartilháveis". Páginas somente de leitura, como código de programa, podem ser compartilhadas facilmente, mas o compartilhamento de páginas com dados é mais complicado.
3. Espaços I e D Separados: Em sistemas com espaços de endereçamento de instruções (I) e dados (D) separados, "dois ou mais processos usam a mesma tabela de páginas para seu espaço I, mas diferentes tabelas de páginas para seus espaços D". Cada processo tem ponteiros para essas tabelas separadas, facilitando o compartilhamento de código.
4. Compartilhamento sem Espaços I e D Separados: "Mesmo sem espaços I e D separados, os processos podem compartilhar programas", embora o mecanismo seja mais complicado.

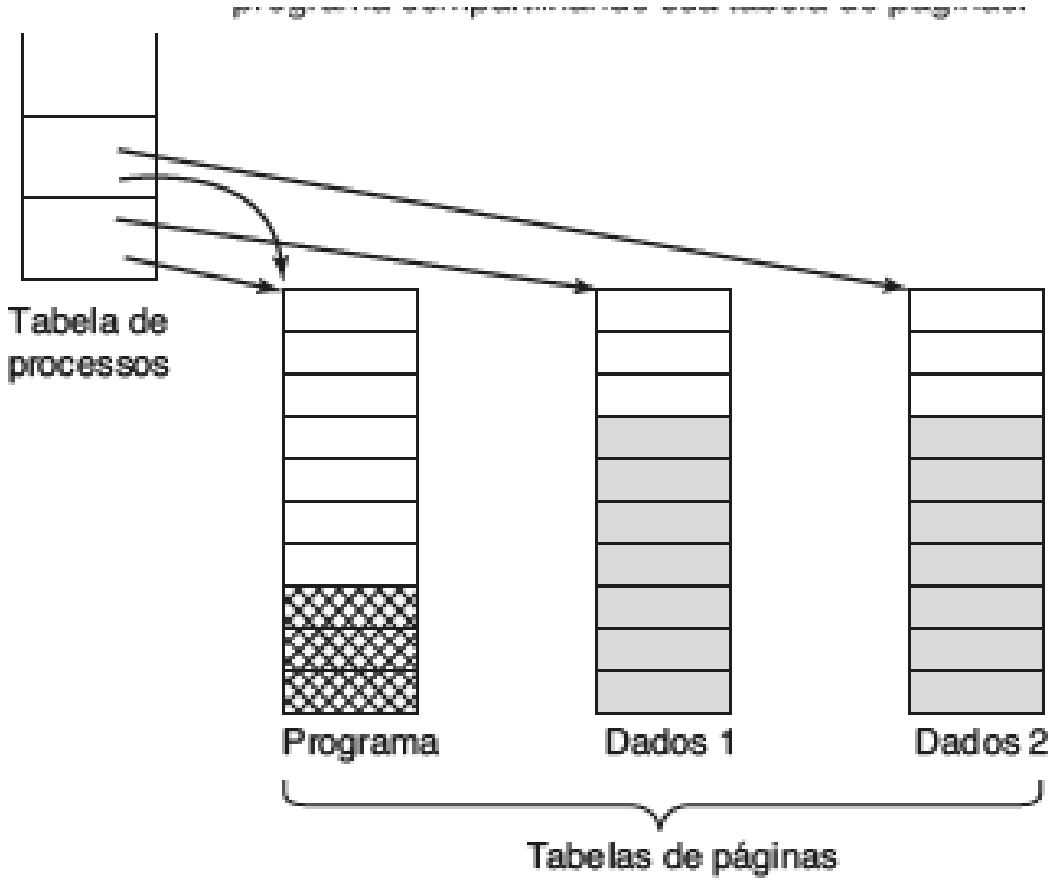
Questões de Projeto em Sistemas de Paginação

Páginas compartilhadas

5. Desafios do Compartilhamento de Páginas: Quando páginas são compartilhadas, a remoção de um processo pode levar a um grande número de faltas de páginas em outros processos compartilhando as mesmas páginas. Além disso, é importante que o sistema operacional saiba que páginas ainda estão em uso para evitar a liberação acidental do espaço de disco.
6. Estruturas de Dados Especiais para Controle de Compartilhamento: Para gerenciar páginas compartilhadas de maneira eficiente, "estruturas de dados especiais são necessárias", principalmente se a unidade de compartilhamento for páginas individuais.
7. Compartilhamento de Dados e 'Copiar na Escrita': "Compartilhar dados é mais complicado do que compartilhar códigos, mas não é impossível". No UNIX, após um fork, processo pai e filho compartilham o código do programa e os dados, com páginas de dados mapeadas como somente leitura. Se um processo modifica os dados, ocorre uma interrupção, e uma cópia da página é feita, permitindo que cada processo tenha sua própria versão. Isso é conhecido como estratégia de "copiar na escrita" (copy on write), que melhora o desempenho reduzindo o número de cópias necessárias.

Questões de Projeto em Sistemas de Paginação

Páginas compartilhadas



Implementação e Questões Operacionais em Gerenciamento de Memória

Envolvimento do sistema operacional com a paginação
O envolvimento do sistema operacional com a paginação ocorre em quatro momentos principais, cada um com suas próprias ações e procedimentos necessários:

Criação do Processo: O sistema operacional deve determinar o tamanho inicial do programa e dos dados. Criação e inicialização de uma tabela de páginas para o processo. Alocação de espaço na memória para a tabela de páginas e na área de troca do disco. Inicialização da área de troca com o código do programa e os dados. "Por fim, informações a respeito da tabela de páginas e área de troca no disco devem ser gravadas na tabela de processos."

Execução do Processo: Reinicialização da MMU para o novo processo e limpeza do TLB. A tabela de páginas do processo em execução se torna a atual, normalmente através da cópia da tabela ou de um ponteiro para ela em registradores de hardware. "Opcionalmente, algumas ou todas as páginas do processo podem ser trazidas para a memória."

Implementação e Questões Operacionais em Gerenciamento de Memória

Envolvimento do sistema operacional com a paginação

Faltas de Páginas: Leitura dos registradores de hardware para determinar o endereço virtual que causou a falta. Cálculo da página necessária e sua localização no disco. Encontrar um quadro de página disponível e, se necessário, remover uma página antiga. Leitura da página necessária para o quadro de página. "Salvar o contador do programa para que ele aponte para a instrução que causou a falta de página e deixar que a instrução seja executada novamente."

Término do Processo: Liberação da tabela de páginas do processo, suas páginas na memória e o espaço de disco ocupado. Se algumas páginas são compartilhadas com outros processos, elas só podem ser liberadas quando o último processo que as utiliza for terminado.

Esses procedimentos garantem a eficiente gestão e manipulação da memória em um sistema operacional que utiliza paginação, desde a criação do processo até sua conclusão.

Implementação e Questões Operacionais em Gerenciamento de Memória

Tratamento de falta de página

Estamos enfim em uma posição para descrever em detalhes o que acontece em uma falta de página. A sequência de eventos é a seguinte:

1. O hardware gera uma interrupção para o núcleo, salvando o contador do programa na pilha. Na maioria das máquinas, algumas informações a respeito do estado da instrução atual são salvas em registradores de CPU especiais.
2. Uma rotina em código de montagem é ativada para salvar os registradores gerais e outras informações voláteis, a fim de impedir que o sistema operacional as destrua. Essa rotina chama o sistema operacional como um procedimento.
3. O sistema operacional descobre que uma falta de página ocorreu, e tenta descobrir qual página virtual é necessária. Muitas vezes um dos registradores do hardware contém essa informação. Do contrário, o sistema operacional deve resgatar o contador do programa, buscar a instrução e analisá- la no software para descobrir qual referência gerou essa falta de página.

Implementação e Questões Operacionais em Gerenciamento de Memória

Tratamento de falta de página

4. Uma vez conhecido o endereço virtual que causou a falta, o sistema confere para ver se esse endereço é válido e a proteção é consistente com o acesso. Se não for, é enviado um sinal ao processo ou ele é morto. Se o endereço for válido e nenhuma falha de proteção tiver ocorrido, o sistema confere para ver se um quadro de página está disponível. Se nenhum quadro de página estiver disponível, o algoritmo de substituição da página é executado para selecionar uma vítima.
5. Se o quadro de página estiver sujo, a página é escalonada para transferência para o disco, e um chaveamento de contexto ocorre, suspendendo o processo que causou a falta e deixando que outro seja executado até que a transferência de disco tenha sido completada. De qualquer maneira, o quadro é marcado como ocupado para evitar que seja usado para outro fim.
6. Tão logo o quadro de página esteja limpo (seja imediatamente ou após ele ter sido escrito para o disco), o sistema operacional buscará o endereço de disco onde a página desejada se encontra, e escalonará uma operação de disco para trazê-la. Enquanto a página estiver sendo carregada, o processo que causou a falta ainda está suspenso e outro processo do usuário é executado, se disponível.

Implementação e Questões Operacionais em Gerenciamento de Memória

Tratamento de falta de página

7. Quando a interrupção de disco indica que a página chegou, as tabelas de página são atualizadas para refletir a sua posição e o quadro é marcado como em um estado normal.
8. A instrução que causou a falta é recuperada para o estado que ela tinha quando começou e o contador do programa é reinicializado para apontar para aquela instrução.
9. O processo que causou a falta é escalonado, e o sistema operacional retorna para a rotina (em linguagem de máquina) que a chamou.
10. Essa rotina recarrega os registradores e outras informações de estado e retorna para o espaço do usuário para continuar a execução, como se nenhuma falta tivesse ocorrido.

Implementação e Questões Operacionais em Gerenciamento de Memória

Separação da política e do mecanismo

Uma ferramenta importante para o gerenciamento da complexidade de qualquer sistema é separar a política do mecanismo. Esse princípio pode ser aplicado ao gerenciamento da memória fazendo que a maioria dos gerenciadores de memória seja executada como processos no nível do usuário. Tal separação foi feita pela primeira vez em Mach (YOUNG et al., 1987) sobre a qual a discussão é baseada a seguir.

Um exemplo simples de como a política e o mecanismo podem ser separados é mostrado na Figura 3.29. Aqui o sistema de gerenciamento de memória está dividido em três partes:

1. Um tratador de MMU de baixo nível.
2. Um tratador de falta de página que faz parte do núcleo.
3. Um paginador externo executado no espaço do usuário.

Todos os detalhes de como a MMU funciona são encapsulados no tratador de MMU, que é um código dependente de máquina e precisa ser reescrito para cada nova plataforma que o sistema operacional executar.

Implementação e Questões Operacionais em Gerenciamento de Memória

Separação da política e do mecanismo

Quando um processo inicia, ocorrem várias interações com o sistema de paginação para garantir a eficiência e a correta alocação de memória. O processo começa com o "paginador externo sendo notificado para ajustar o mapa de páginas do processo e alocar o armazenamento de apoio no disco se necessário". Conforme o processo executa, ele pode necessitar mapear novos objetos em seu espaço de endereçamento, levando a novas notificações ao paginador externo.

Durante a execução, o processo pode causar faltas de página. Nestes casos, "o tratador de faltas calcula qual página virtual é necessária e envia uma mensagem para o paginador externo". O paginador externo, por sua vez, lê a página necessária do disco e a copia para seu próprio espaço de endereçamento. Após isso, ele informa o tratador de faltas sobre a localização da página.

O próximo passo envolve o tratador de páginas, que "remove o mapeamento da página do espaço de endereçamento do paginador externo e pede ao tratador da MMU para colocar a página no local correto dentro do espaço de endereçamento do usuário". Isso permite que o processo do usuário seja reiniciado.

Implementação e Questões Operacionais em Gerenciamento de Memória

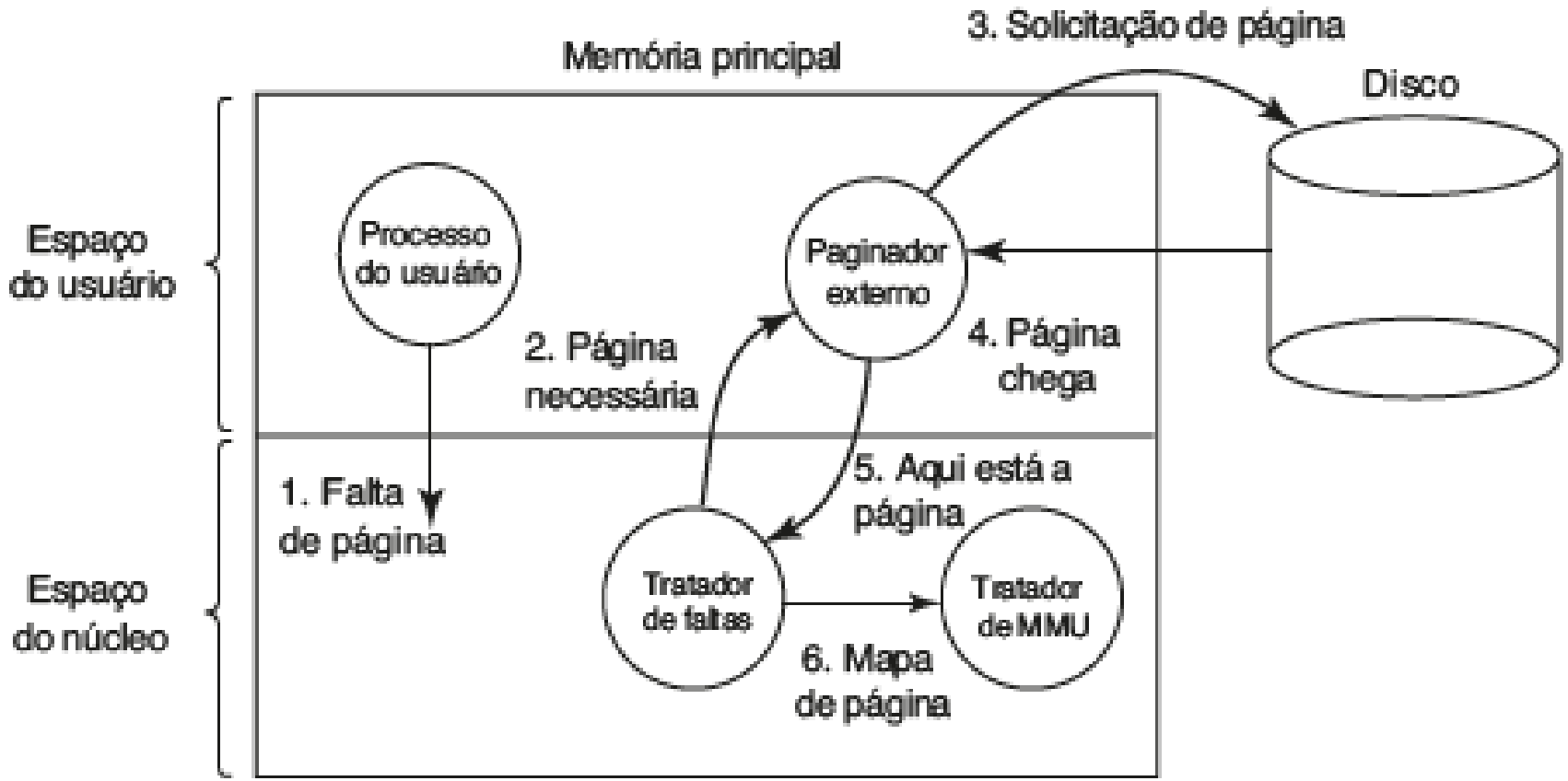
Separação da política e do mecanismo

Um aspecto importante desta implementação é a flexibilidade na localização do algoritmo de substituição de página. Embora seja mais lógico tê-lo no paginador externo, há desafios, como "o paginador externo não ter acesso aos bits R e M de todas as páginas", que são essenciais para muitos algoritmos de paginação. Como solução, é necessário um mecanismo para passar essa informação ao paginador externo ou localizar o algoritmo de substituição de página no núcleo.

A principal vantagem desse método é a modularidade e flexibilidade do código, mas existe um "custo extra causado pelos diversos chaveamentos entre o núcleo e o usuário, assim como a sobrecarga nas trocas de mensagens entre as partes do sistema". Atualmente, essa abordagem é controversa, mas a tendência é que, com o aumento da velocidade dos computadores e a complexidade dos softwares, a troca de algum desempenho por software mais confiável se torne aceitável para a maioria dos implementadores.

Implementação e Questões Operacionais em Gerenciamento de Memória

Separação da política e do mecanismo



Implementação e Questões Operacionais em Gerenciamento de Memória

Segmentação na Gestão de Memória

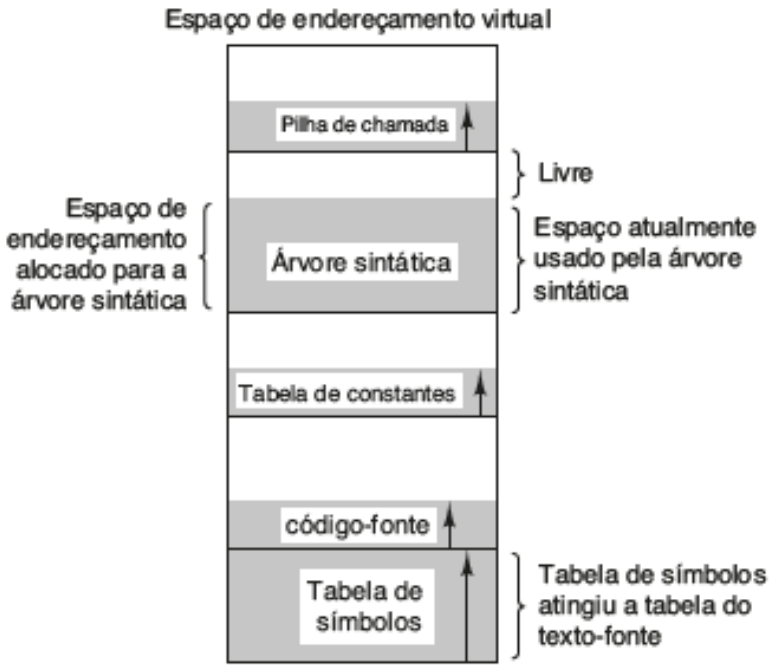
A memória virtual discutida até aqui é unidimensio nal, pois os endereços virtuais vão de 0 a algum endere ço máximo, um endereço depois do outro. Para muitos problemas, ter dois ou mais espaços de endereços vir tuais pode ser muito melhor do que ter apenas um. Por exemplo, um compilador tem muitas tabelas construí das em tempo de compilação, possivelmente incluindo:

1. O código-fonte sendo salvo para impressão (em sistemas de lote).
2. A tabela de símbolos, contendo os nomes e atri butos das variáveis.
3. A tabela contendo todas as constantes usadas, in teiras e em ponto flutuante.
4. A árvore sintática, contendo a análise sintática do programa.
5. A pilha usada pelas chamadas de rotina dentro do compilador.

Cada uma das quatro primeiras tabelas cresce continuamente à medida que a compilação prossegue. A última cresce e diminui de maneiras imprevisíveis durante a compilação. Em uma memória unidimensional, essas cinco tabelas teriam de ser alocadas em regiões contíguas do espaço de endereçamento virtual, como na Figura 3.30.

Implementação e Questões Operacionais em Gerenciamento de Memória

Segmentação na Gestão de Memória



Implementação e Questões Operacionais em Gerenciamento de Memória

Segmentação

O conceito de memória segmentada em sistemas operacionais é uma solução direta e geral para gerenciar espaços de endereçamento que precisam expandir ou contrair dinamicamente.

Uso de Segmentos: Em situações onde um programa tem um excesso de variáveis, mas quantidades normais de outros componentes, pode ocorrer a falta de espaço na tabela de símbolos. A solução é "fornecer à máquina espaços de endereçamento completamente independentes, que são chamados de segmentos".

Características dos Segmentos: Cada segmento é uma sequência linear de endereços, começando do zero e podendo atingir o endereço máximo permitido. "Diferentes segmentos podem e costumam ter comprimentos diferentes" e podem mudar de tamanho durante a execução do programa.

Independência dos Segmentos: Segmentos operam de forma independente, permitindo que, por exemplo, "uma pilha em um determinado segmento precise de mais espaço de endereçamento para crescer, ela pode tê-lo".

Implementação e Questões Operacionais em Gerenciamento de Memória

Segmentação

Endereçamento de Memória Segmentada: Para especificar um endereço na memória segmentada, "o programa precisa fornecer um endereço em duas partes, um número de segmento e um endereço dentro do segmento".

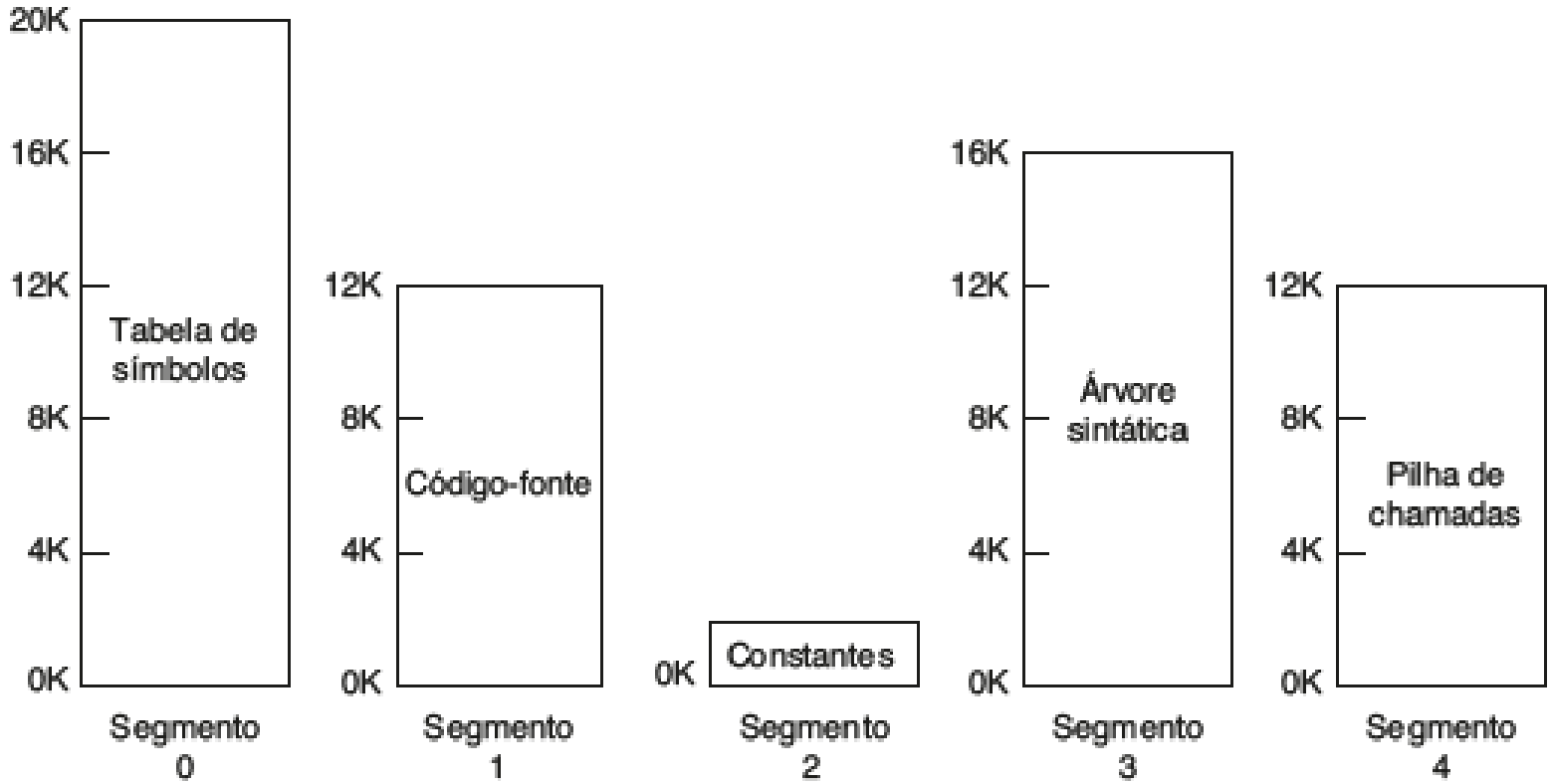
Vantagens da Memória Segmentada: A memória segmentada simplifica a manipulação de estruturas de dados dinâmicas. Além disso, facilita a ligação de rotinas compiladas separadamente e a compartilhamento de rotinas ou dados entre vários processos, como no caso de "bibliotecas gráficas extremamente grandes compiladas em quase todos os programas".

Proteção de Segmentos: Segmentos podem ter diferentes tipos de proteção. Por exemplo, "um segmento de rotina pode ser especificado como somente de execução", enquanto um conjunto de dados pode ser somente leitura/escrita.

Comparação com Paginação: Embora a segmentação e a paginação sejam métodos diferentes de gestão de memória, sistemas de paginação puros muitas vezes simulam a segmentação para gerenciar bibliotecas compartilhadas de forma eficiente.

Implementação e Questões Operacionais em Gerenciamento de Memória

Segmentação



Segmentação com Paginação

MULTICS

A implementação da memória virtual no sistema operacional MULTICS, um dos sistemas mais influentes, exemplifica a combinação eficiente de segmentação e paginação.

Paginação de Segmentos Grandes: Em casos onde os segmentos são grandes, "talvez seja inconveniente, ou mesmo impossível, mantê-los na memória principal em sua totalidade", levando à necessidade de paginar os segmentos.

O Sistema MULTICS: MULTICS, iniciado como um projeto de pesquisa no M.I.T., foi implementado em 1969 e operou até 2000. "O sistema operacional MULTICS foi um dos mais influentes de todos os tempos", com impactos em diversos aspectos da computação.

Estrutura de Memória Virtual do MULTICS: MULTICS funcionava em máquinas Honeywell 6000, fornecendo "uma memória virtual de até 2^18 segmentos, cada um com até 65.536 palavras (36 bits) de comprimento".

Segmentos como Memórias Virtuais: No MULTICS, cada segmento era tratado como uma memória virtual individual e paginada, combinando "as vantagens da paginação com as vantagens da segmentação".

Descritores de Segmentos e Tabelas Paginadas: Cada programa no MULTICS tinha uma tabela de segmentos paginada, com um descritor por segmento. O descritor de segmento incluía informações como se o segmento estava na memória e um ponteiro para a tabela de páginas do segmento.

Segmentação com Paginação

MULTICS

Endereçamento em MULTICS: Um endereço no MULTICS consistia em duas partes: o número do segmento e o endereço dentro dele, subdividido em número de página e palavra dentro da página.

Processo de Referência de Memória: Ao ocorrer uma referência de memória, o sistema "usava o número do segmento para encontrar o descritor do segmento", seguido por uma série de verificações e passos para localizar a palavra desejada na memória.

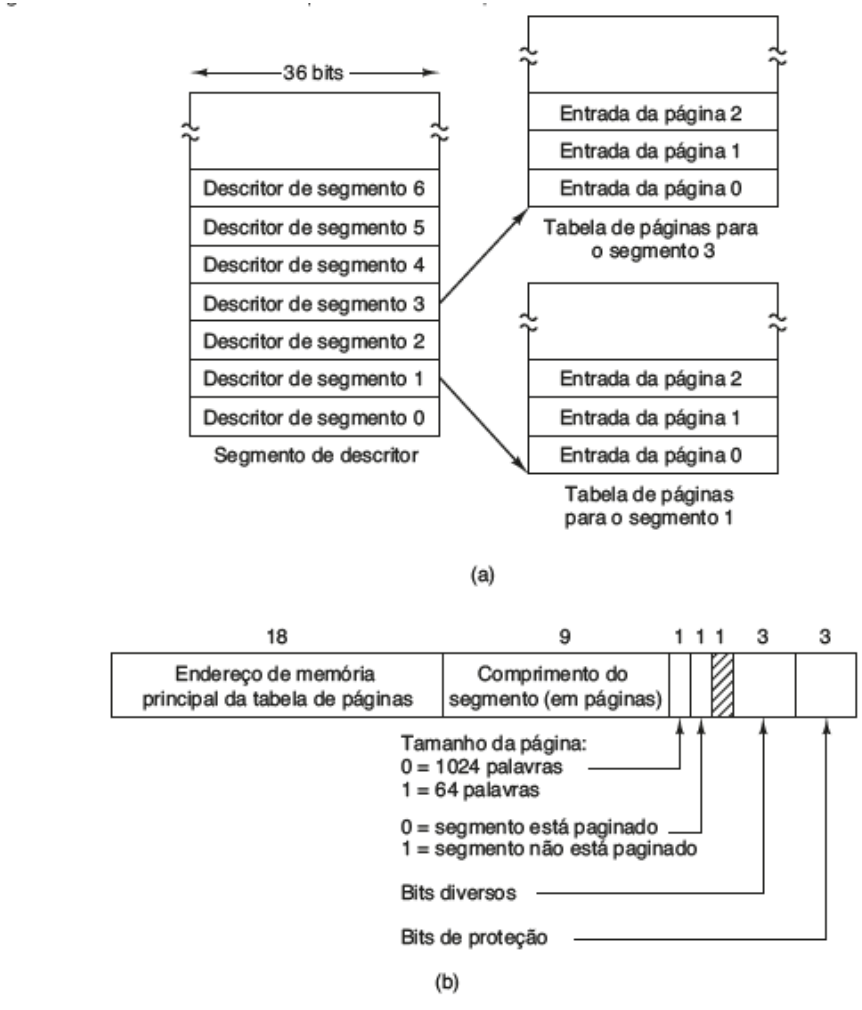
Uso do TLB no MULTICS: O MULTICS incluía um TLB de alta velocidade para melhorar a eficiência. "Quando um endereço era apresentado ao computador, o hardware de endereçamento primeiro conferia para ver se o endereço virtual estava no TLB."

Eficiência do TLB: Com o TLB, endereços das páginas mais usadas eram armazenados para acesso rápido, permitindo que "programas cujo conjunto de trabalho era menor do que o tamanho do TLB executassem eficientemente".

O MULTICS é um exemplo histórico de como a segmentação e a paginação podem ser combinadas de maneira eficaz para criar um sistema de memória virtual robusto e eficiente, com inovações como o uso do TLB que continuam a influenciar o design de sistemas operacionais modernos.

Segmentação com Paginação

MULTICS



Segmentação com Paginação

Intel x86

O sistema de memória virtual do x86, até a introdução do x86-64, representou uma evolução significativa na combinação de segmentação e paginação.

Evolução do x86: Até o advento do x86-64, o sistema de memória virtual do x86 foi semelhante ao MULTICS, com "16K segmentos independentes, cada um com até 1 bilhão de palavras de 32 bits". No x86-64, a segmentação foi considerada obsoleta, exceto no modo legado.

Estrutura da Memória Virtual no x86: O coração da memória virtual do x86 consistia em duas tabelas, a LDT e a GDT. "Cada programa tem seu próprio LDT, mas há uma única GDT, compartilhada por todos os programas no computador".

Funcionamento dos Segmentos no x86: Para acessar um segmento, "um programa x86 primeiro carrega um seletor para aquele segmento em um dos seis registradores de segmentos da máquina". Este seletor é um número de 16 bits que identifica o segmento como local ou global.

Descritores de Segmentos: Cada descritor de segmento no x86 inclui o endereço base do segmento, tamanho e outras informações. "Um descritor consiste em 8 bytes".

Conversão de Endereços no x86: O sistema usa o seletor e o deslocamento para encontrar o descritor correspondente e, subsequentemente, converte um par (seletor, deslocamento) em um endereço físico.

Segmentação com Paginação

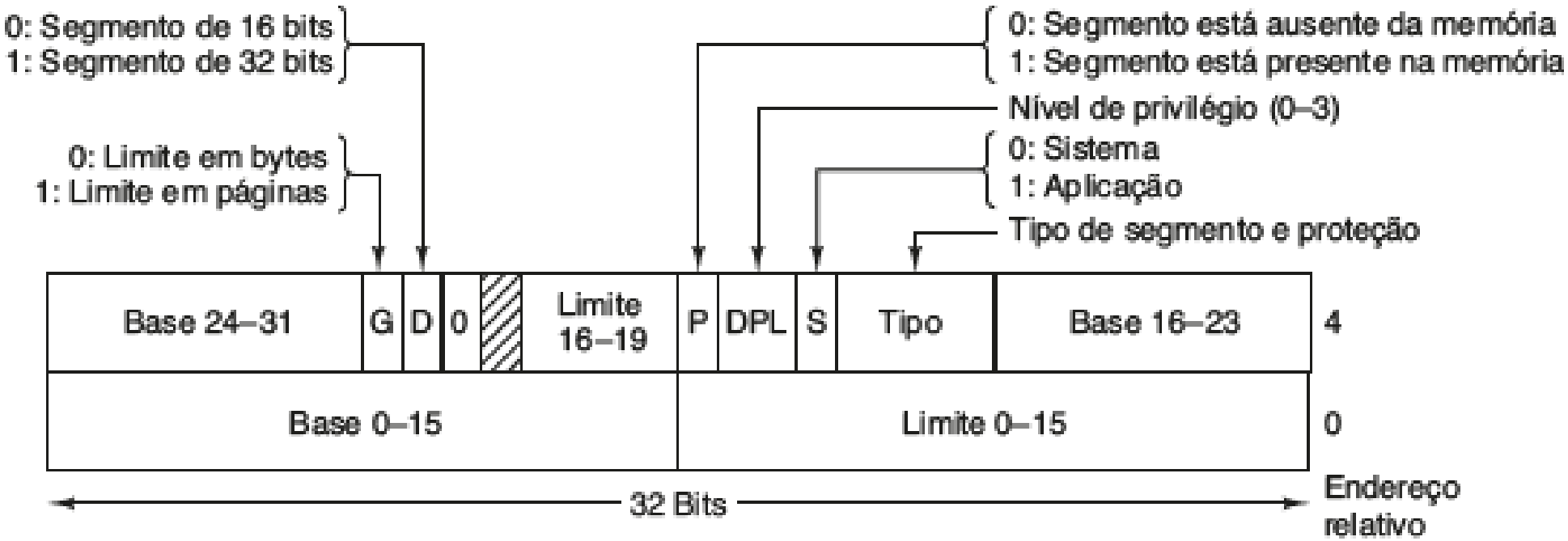
Intel x86

Paginação e TLB no x86: Quando a paginação está habilitada, "o endereço linear é interpretado como um endereço virtual e mapeado no endereço físico usando as tabelas de páginas". O x86 também utiliza uma TLB para mapear eficientemente os endereços mais usados.
Simplicidade e Eficiência do Design: O design do sistema de memória do x86 é reconhecido por sua "simplicidade e limpeza", equilibrando eficientemente a implementação de segmentação e paginação.
Adaptação aos Sistemas Operacionais Modernos: Sistemas operacionais modernos para x86, como o Windows e o UNIX, utilizam o modelo de memória virtual do x86 de uma forma simplificada, estabelecendo todos os registradores de segmentos com o mesmo seletor para um espaço de endereçamento paginado de 32 bits.

A arquitetura de memória virtual do x86 até o x86-64 representou uma abordagem inovadora para o gerenciamento de memória em sistemas computacionais, combinando segmentação e paginação de maneira eficiente e influenciando significativamente o design de sistemas operacionais contemporâneos.

Segmentação com Paginação

Intel x86



Pesquisa Atual em Gerenciamento de Memória

O gerenciamento de memória, com foco nos algoritmos de paginação para CPUs uniprocessadores, uma vez campo fértil de pesquisa, tem agora encontrado novas fronteiras e desafios. De acordo com Moruz et al. (2012), mesmo em sistemas de propósito geral, há ainda espaço para inovações, enquanto aplicações específicas como o processamento de transações on-line demandam abordagens especializadas (Stoica e Ailamaki, 2013).

Um exemplo dessa evolução é observado nos uniprocessadores, onde a substituição de discos rígidos por SSDs levanta questões de desempenho e necessidade de novos algoritmos (Chen et al., 2012). Além disso, as memórias com mudança de fase não voláteis, com suas características únicas de desempenho e desgaste (Lee et al., 2013; Saito e Oikawa, 2012; Bheda et al., 2011, 2012), exigem uma reavaliação dos métodos de paginação.

As máquinas virtuais, por exemplo, renovaram o interesse no gerenciamento de memória, como destacado por Bugnion et al. (2012). Jantz et al. (2013) apresentam a ideia de permitir que aplicações influenciem as decisões sobre a paginação física de páginas virtuais. Em ambientes de nuvem, a variabilidade da memória física disponível para máquinas virtuais requer algoritmos adaptáveis, um aspecto explorado por Peserico (2013).

Pesquisa Atual em Gerenciamento de Memória

A pesquisa também avançou para sistemas com múltiplos núcleos, como observado em trabalhos de Boyd-Wickizer et al. (2008) e Baumann et al. (2009). A complexidade da memória cache compartilhada nesses sistemas é um fator importante, conforme apontado por Lopez-Ortiz e Salinger (2012). Paralelamente, sistemas NUMA, com tempos de acesso variáveis à memória, são explorados por Dashti et al. (2013) e Lankes et al. (2012).

No contexto dos dispositivos móveis, como smartphones e tablets, Joo et al. (2012) analisam a paginação de RAM para armazenamento em flash. Finalmente, o gerenciamento de memória em sistemas em tempo real continua relevante, com pesquisas focadas em atender suas exigências específicas, como destacado por Kato et al. (2011).