Sistemas Operacionais

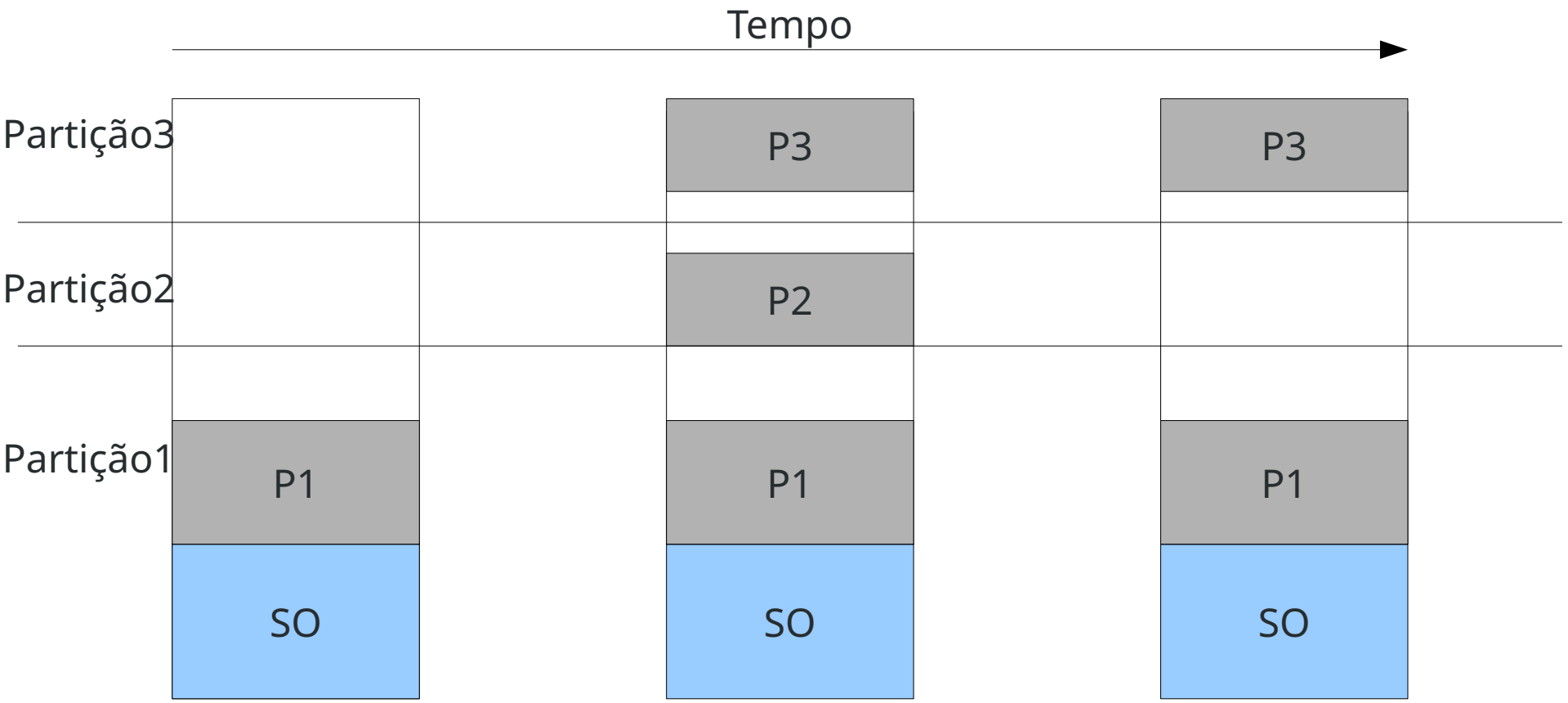
Módulo 8 – Memória Virtual

Prof. Daniel Sundfeld
daniel.sundfeld@unb.br

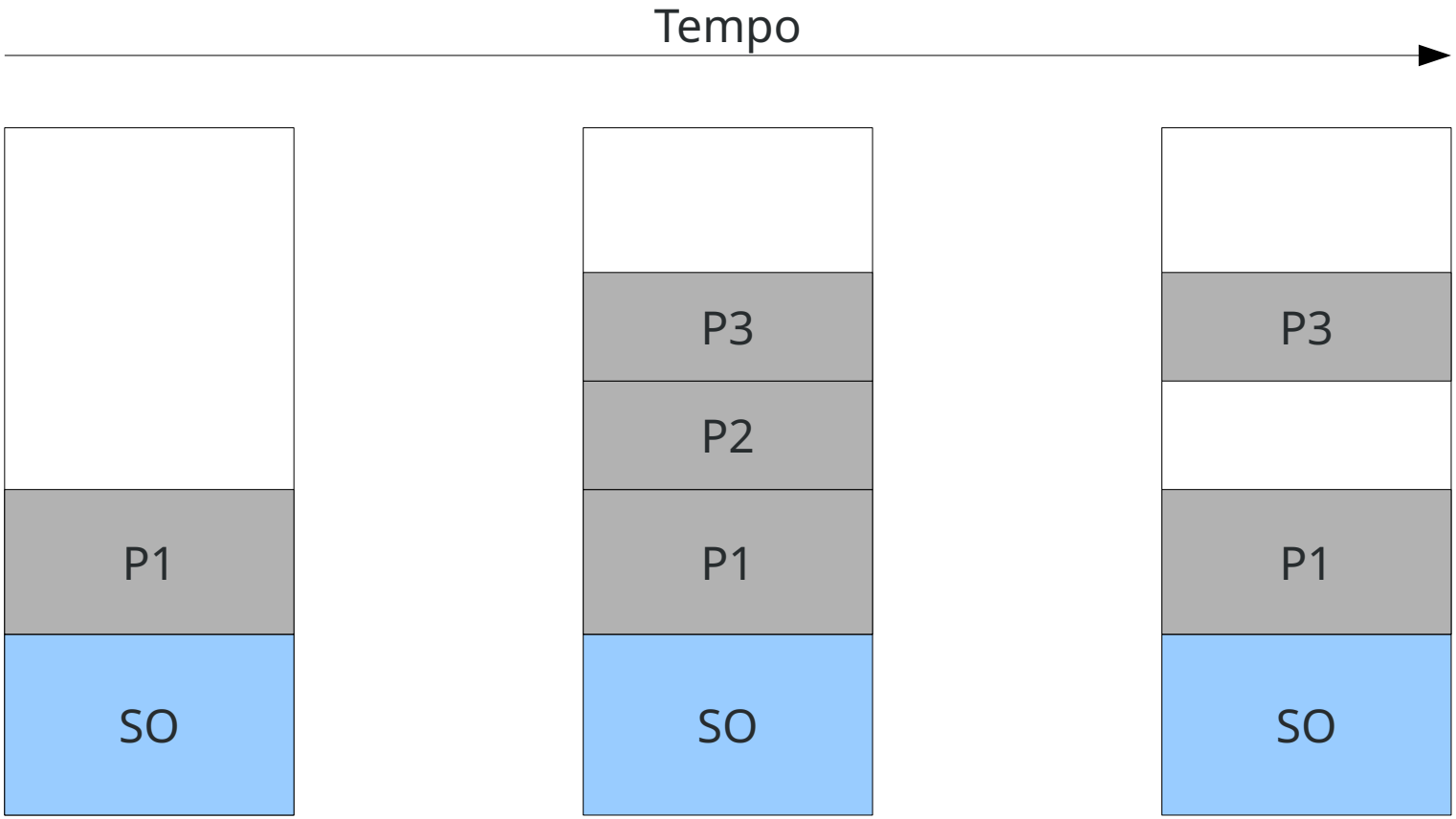
Aula Anterior

- Funções básicas
- Esquemas de gerenciamento de memória
  - Sem abstração de memória
  - Com abstração de memória
    - Partições Fixas
    - Partições Variáveis
- Algoritmos de controle de memória
- Swapping e Overlay

Fragmentação Interna



Fragmentação Externa



Roteiro

- Introdução
- Endereçamento Virtual e Mapeamento
- Memória Virtual
- Memória Virtual por Paginação
  - Alocação/Substituição e Algoritmos de Substituição de Páginas
  - Implementação: Tamanho das Páginas/Proteção da Memória
- Memória Virtual por Segmentação
- Comparação de Paginação e Segmentação
- Memória Virtual com Paginação e Segmentação

Introdução

- O uso de partições variáveis é mais flexível e permite alocar áreas de memórias para os processos em execução
- No entanto, o problema da fragmentação externa era constante e às vezes pode precisar de intervenção do sistema operacional

Introdução

- Uma possível minimização para o problema da fragmentação externa é o uso de memória virtual
- Implementado na grande maioria dos sistemas operacionais modernos
- Semelhante à implementação de um overlay, mas é implementada pelo SO, e não pelo programador
- O processo não referencia mais endereços físicos e sim endereços virtuais

Introdução

- Desta forma, as memória primária e secundária são combinadas dando a ilusão ao usuário de que há uma memória amplamente disponível
- Fundamenta-se no fato de não vincular os endereços vinculados do programa a endereços físicos da memória disponível
- Em uma máquina de 32 bits, um processo pode endereçar 2^(32) – 1 endereços virtuais, não dependendo do tamanho da memória física

Principais vantagens

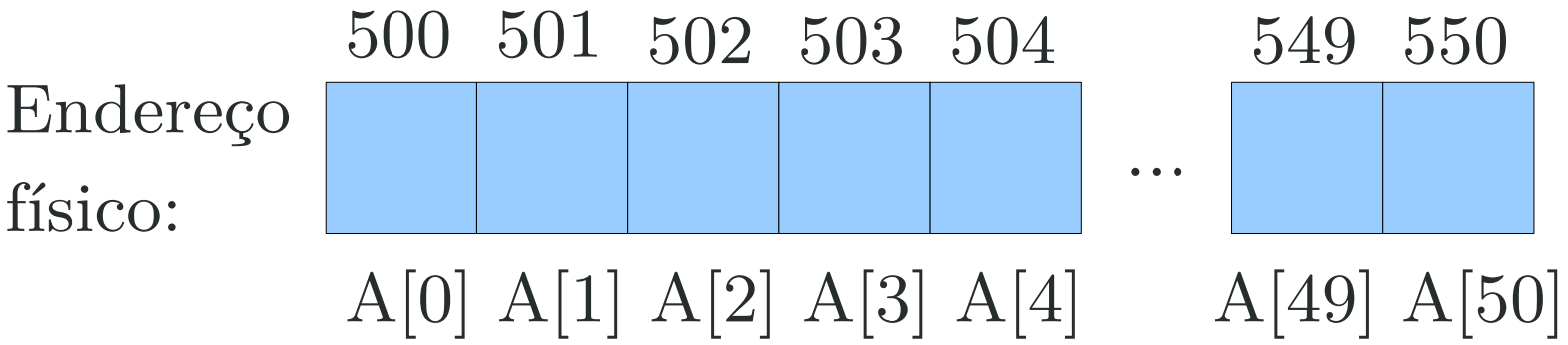
- Permite que um número maior de processos na memória principal
- Leva ao uso mais eficiente do processador
- Minimiza o problema da fragmentação da memória principal

Implementação

- Muitas das funções e modificações necessárias para usar a memória virtual são implementadas diretamente no hardware
- Por isso, os SOs devem levar em consideração a arquitetura de computadores disponível
- Especialmente o esquema de endereçamento do processador

Endereçamento Virtual

- Um endereço virtual pode ser comparado ao acessar um vetor em memória
- O programador não se preocupa em saber quais são os endereços que ele está acessando: o compilador faz essa tarefa



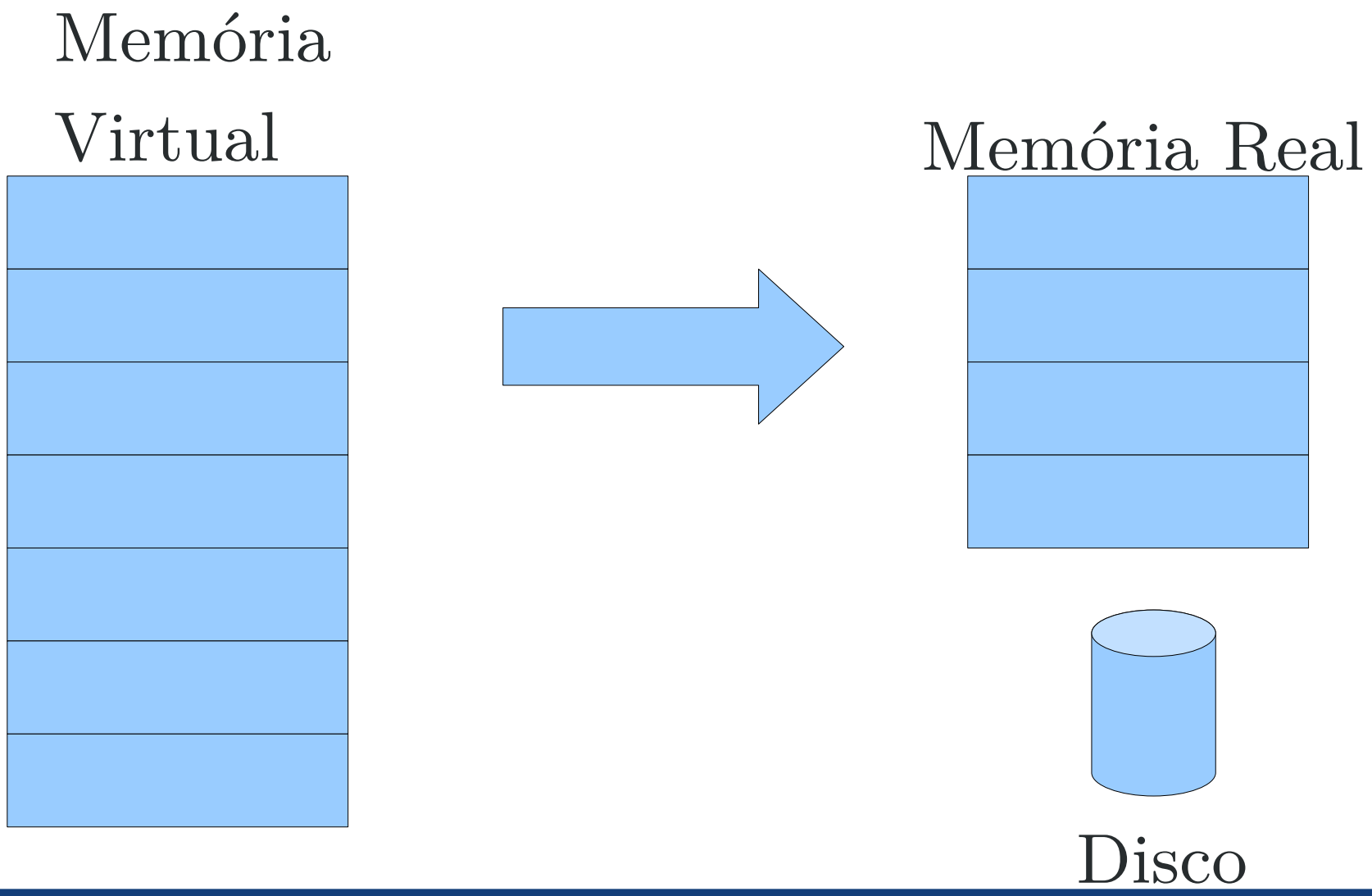
Endereçamento Virtual

- Analogamente, cria-se um ambiente com uma memória virtual
- O processo em execução faz referências ao espaço de endereçamento virtual
- O sistema operacional é responsável por traduzir essas instruções para endereços reais

Endereçamento Virtual

- Não existe alguma relação direta entre os endereços no espaço real, é possível que o programa referencie endereços que estão fora dos limites
- Assim o sistema operacional combina a memória principal e secundária para criar um espaço de endereçamento maior que o disponível fisicamente

Endereçamento Virtual



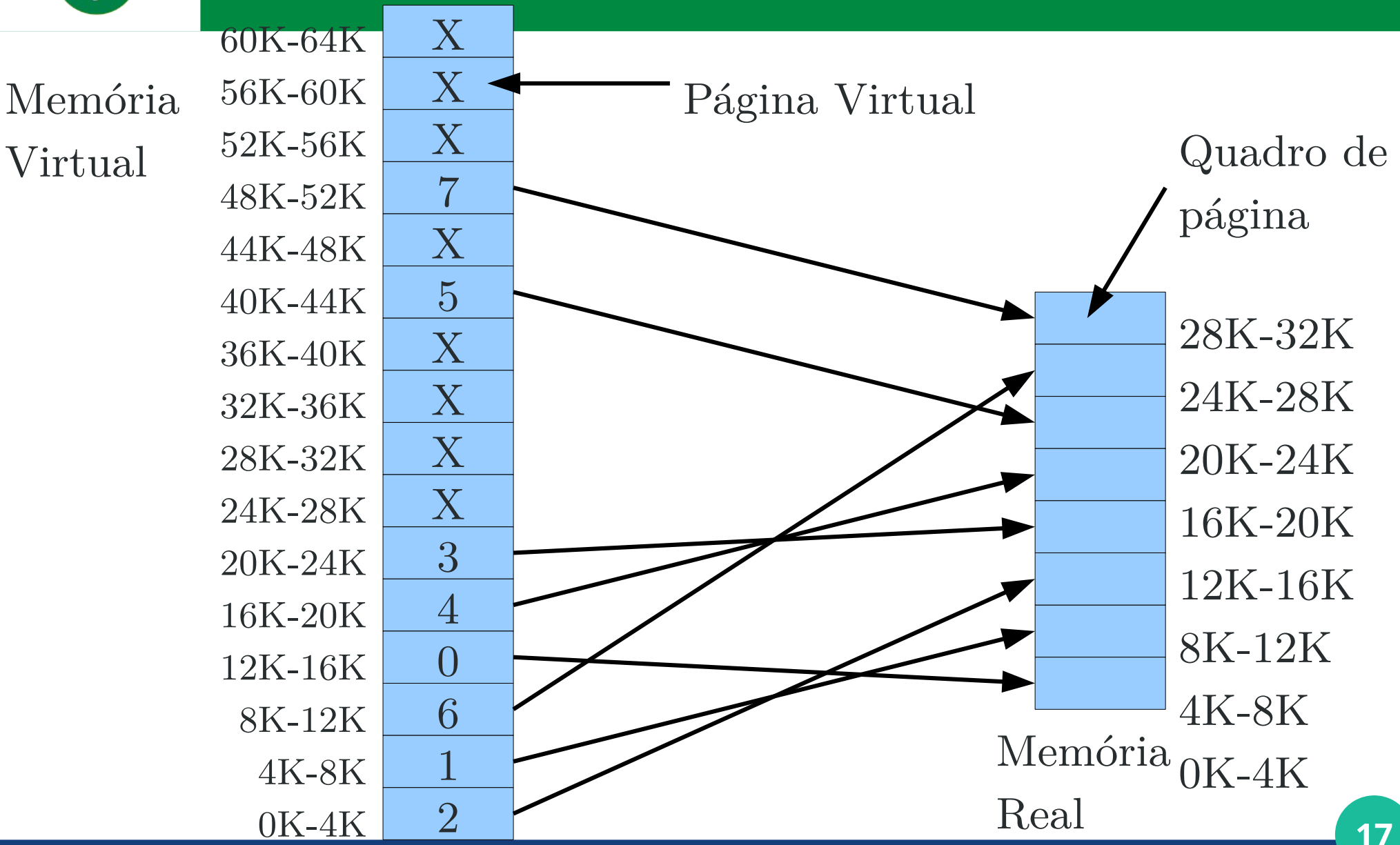
Endereçamento Virtual

- Ao desenvolver um programa, os programadores ignoram o espaço de endereçamento virtual: eles determinam apenas a quantidade de memória necessária
- Durante o processo de **compilação** e **ligação**, é gerado um código executável que utiliza endereços virtuais
- Esses endereços são usados pelo SO durante a **execução** para mapear em endereços reais

Endereçamento Virtual

- Desta forma, o compilador cuida de criar instruções:
  `LOAD R1, 1000`
- Durante a execução, o Sistema Operacional cuida dos detalhes de mapear o endereço virtual em um endereço real

Mapeamento



Fonte: Tanenbaum

Mapeamento

- Temos duas grandes vantagens nesse esquema de mapeamento:
- **1:** Pode-se usar memória secundária e memória primária para criar uma memória maior que a disponível
- **2:** E algo muito importante: os processos **não** precisam necessariamente serem alocados em memória contínua!!

Mapeamento

- Porém é necessário um mecanismo para mapear os endereços do espaço virtual para o espaço real
- Para não comprometer o desempenho, esse mecanismo é implementado diretamente no hardware
- Antigamente era um chip a parte, nos sistemas modernos foi incorporado à CPU

Mapeamento

- MMU: Unidade de Mapeamento de Memória
- Mapeia endereços virtuais em endereços de memória física
- Desta forma, os endereços virtuais não vão diretamente para o barramento de memória
- Eles vão para a unidade antes da CPU fazer a referência à memória. Cada processo possui o seu espaço de endereçamento virtual e na execução os endereços são traduzido a cada instrução

Mapeamento

- A tabela de mapeamento é uma estrutura de dados para cada processo
- Essa tabela é alterada a cada troca de contexto entre processos
- Normalmente implementado como um registrador na CPU que aponta a entrada da tabela que deve ser utilizada para o processo corrente

Mapeamento

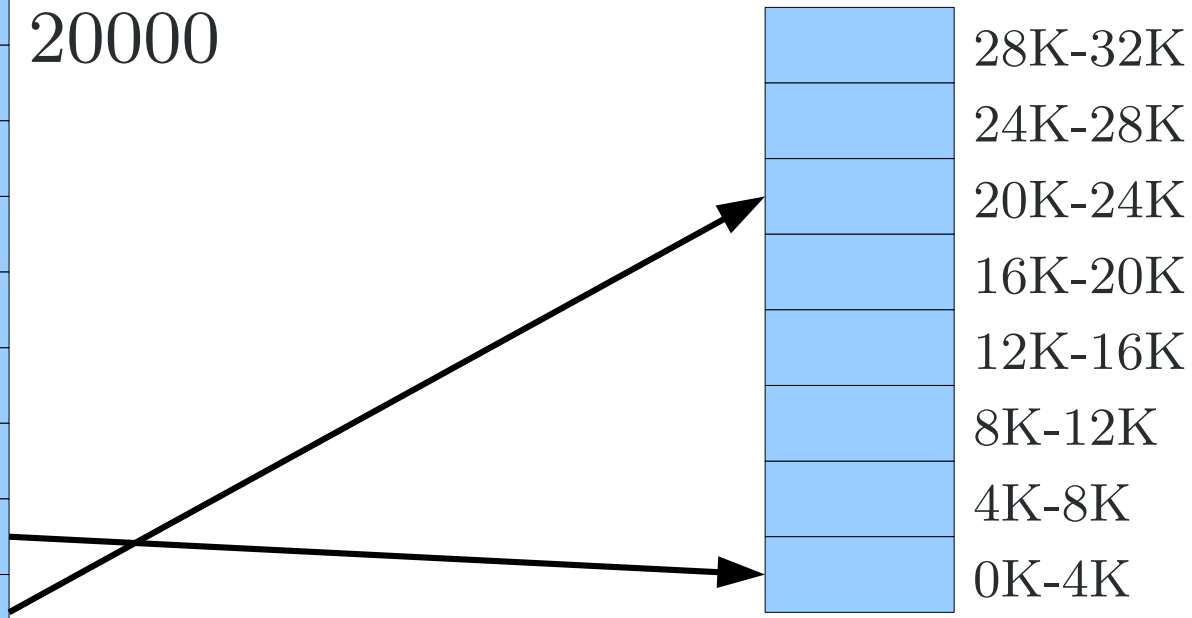
60K-64K	X
56K-60K	X
52K-56K	X
48K-52K	7
44K-48K	X
40K-44K	6
36K-40K	X
32K-36K	X
28K-32K	X
24K-28K	X
20K-24K	3
16K-20K	4
12K-16K	0
8K-12K	5
4K-8K	1
0K-4K	2

Ex., instrução executada pela CPU:

LOAD R1, 8000

Posição de memória real acessada:

20000



Fonte: Tanenbaum

Mapeamento

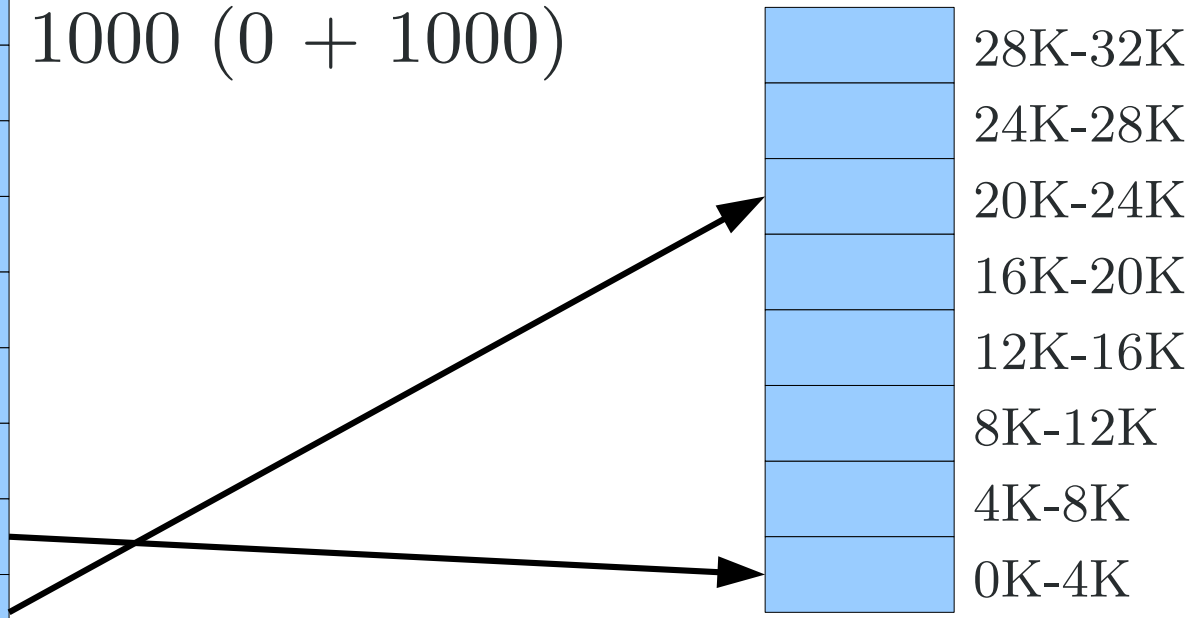
60K-64K	X
56K-60K	X
52K-56K	X
48K-52K	7
44K-48K	X
40K-44K	6
36K-40K	X
32K-36K	X
28K-32K	X
24K-28K	X
20K-24K	3
16K-20K	4
12K-16K	0
8K-12K	5
4K-8K	1
0K-4K	2

Ex., instrução executada pela CPU:

LOAD R1, 13000 (12000 + 1000)

Posição de memória real acessada:

1000 (0 + 1000)



Fonte: Tanenbaum

Mapeamento

- Mas qual o tamanho dessa tabela? Se cada célula da memória fosse uma entrada da tabela, o espaço ocupado pelas tabelas seria tão grande quanto o espaço de endereçamento virtual
- Problema parecido com os de algoritmos de gerência de memória para representar a memória livre
- Por isso, são utilizados blocos de memória no mapeamento, e não células individuais

Mapeamento

- Espaço Virtual e Tamanho Bloco:
- Quanto maior o bloco, menor o número de entradas na tabela de mapeamento

Espaço de Endereçamento Virtual	Tamanho do bloco	Número de entradas na tabela de mapeamento
2^32 endereços	512 endereços	2^23
2^32 endereços	4 K endereços	2^20
2^64 endereços	4 K endereços	2^52
2^64 endereços	64 K endereços	2^48

Mapeamento

- O tamanho da tabela de páginas inviabiliza mantê-la totalmente na MMU
- A tabela de página é armazenada na memória principal e um registrador contém o endereço de início da tabela de páginas
- A MMU possui uma memória associativa (cache) chamada de Translation Look-aside Buffer (TLB), onde algumas entradas de páginas são armazenadas

Mapeamento

- Quando um endereço virtual é apresentado à MMU, ela procura primeiro na TLB
- Se estiver presente, o mapeamento é imediato
- Senão, um acesso à memória é feito para recuperar a entrada da tabela de páginas
- A referência é colocada no TLB
- As próximas referências à essa página serão imediatas

Mapeamento

- Motivamos o uso de memória virtual para o melhor uso das memória por vários programas
- No entanto ela é uma método genérico e pode ser utilizada até em sistemas monoprogramados

Roteiro

- Introdução
- Endereçamento Virtual e Mapeamento
- Memória Virtual
- Memória Virtual por Paginação
  - Alocação/Substituição e Algoritmos de Substituição de Páginas
  - Implementação: Tamanho das Páginas/Proteção da Memória
- Memória Virtual por Segmentação
- Comparação de Paginação e Segmentação
- Memória Virtual com Paginação e Segmentação

Memória Virtual

- Existem dois mecanismos que implementam a memória virtual:
  - **paginação**: divide a memória física e a memória virtual em unidades de **tamanho fixo;**
  - **segmentação:** divide as memórias física e virtual em unidades de **tamanho variável**.

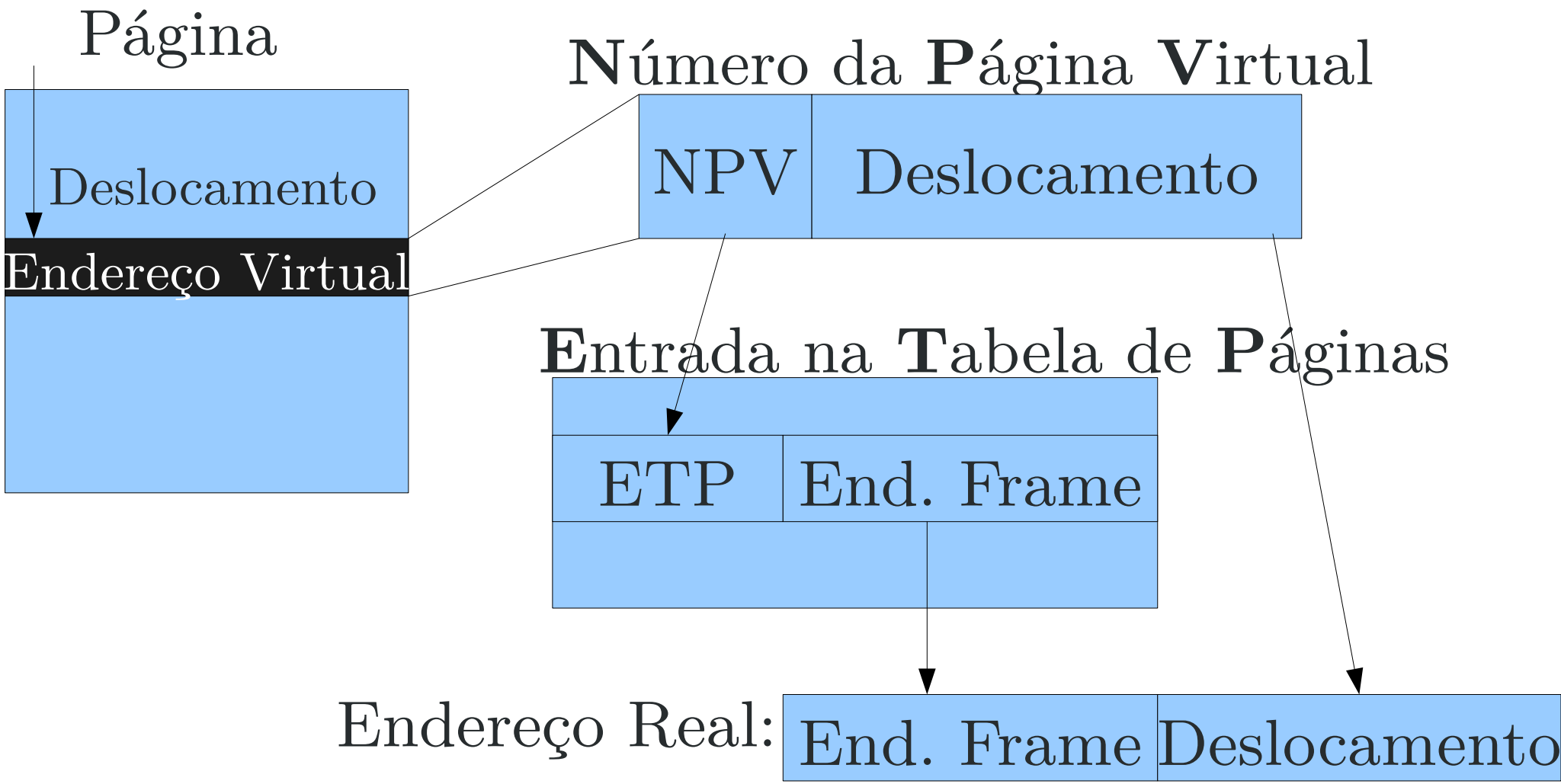
Memória Virtual por Paginação

- A memória é dividida em blocos de mesmo tamanho, chamados páginas
- Páginas no espaço virtual: páginas virtuais
- Páginas no espaço real: páginas reais, quadros de página, page frames ou frames
- As page frames são o espaço das páginas enquanto elas estiverem na memória

Memória Virtual por Paginação

- A CPU envia o endereço virtual (v) para a MMU
- Na MMU, o endereço virtual é dividido em (p,d) onde p é pagina e d é o deslocamento dentro da página. A MMU utiliza a página p para acessar a tabela de páginas e recuperar a frame f na qual a página p reside
- A MMU substitui p por f e coloca o endereço (f,d) no barramento

Memória Virtual por Paginação



Memória Virtual por Paginação

- A entrada na tabela de páginas contém diversas informações: localização da página virtual, se está na memória principal ou secundária, se ela foi referenciada, alterada ou se está protegida, entre outros
- O espaço de endereços é bem maior que a memória disponível. Por isso, é necessário mover páginas da memória secundária para a principal

Memória Virtual por Paginação

- Quando uma memória virtual é referenciada e ela não está na memória principal, é gerado um "**page fault**"
- Então, o SO irá transferir a página da memória secundária para a principal: **page in**
- Nesse processo, ele irá escolher uma página para ser substituída da memória principal, atualizar a tabela de páginas e reexecutar a instrução que causou o page fault

Memória Virtual por Paginação

- Quando um page fault ocorre, o processo é alterado para o estado de bloqueado, até que a operação de E/S para carregar a página na memória seja resolvida
- Após resolver, o processo é movido para a lista de processos prontos para serem escalonados
- O procedimento de mover uma página para disco é conhecido como **page out**

Busca de Páginas

- Ao se buscar páginas na memória, pode-se utilizar diferentes estratégias:
  - Paginação por demanda (demand paging): as páginas dos processos são transferidas somente quando ocorre alguma referência a ela. Somente carrega páginas necessárias a memória
  - Paginação antecipada (prepaging): ao carregar uma página para a memória principal, o sistema também carrega páginas próximas, buscando antecipar páginas que serão acessadas

Alocação de Páginas

- A política de alocação de páginas determina quantos frames cada processo pode ter na memória principal:
  - Política de alocação fixa;
  - Política de alocação variável.

Alocação de Páginas

- Na política de **alocação fixa**, cada processo tem um número máximo de frames a ocupar durante a execução do programa
- Se atingir o máximo, é forçado um page fault
- Apesar de justo, as aplicações possuem requisitos de memória muito diferenciado e diferentes políticas de uso, não sendo muito prático

Alocação de Páginas

- Se o número de páginas é muito pequeno, são forçados muitos page faults, mesmo com memória disponível
- Se o número de páginas for grande, cada processo ocupa uma grande parte da memória e reduz o número de processos residentes

Alocação de Páginas

- Na **alocação variável**, o número de páginas do processo varia durante a sua execução em função da taxa de paginação (número de page faults ocorrido) e a sua atual ocupação da memória
- Assim, processos que estão sofrendo muito page fault pode ampliar o limite máximo de frames, a fim de reduzir o número de page faults
- Processos com baixa taxa, podem ter seus frames liberados para outros processos

Substituição de Páginas

- Logicamente, pode ser necessário trocar páginas do disco para a memória principal
- Para isso, o sistema deve selecionar dentre as diversas páginas selecionadas a que deve ser movida para disco
- Os movimentos de page in e page out são colocados em um arquivo chamado arquivo de paginação

Substituição de Páginas

- Nem todas as páginas precisam ser copiadas para o arquivo: algumas páginas são de apenas leitura de um arquivo que já está em disco, como por exemplo o código executável
- Já as páginas modificáveis contém dados e variáveis que sofrem modificação ao longo da execução do programa
- Por isso, existe um bit para marcar páginas que sofreram modificações ou não

Substituição de Páginas

- Algumas páginas que podem ser modificadas, mas por acaso não tenham sido alteradas durante a execução do programa, não precisam ser copiadas novamente para o disco
- A política de substituição de páginas pode ser local ou global
- Na local, apenas páginas do processo que gerou o page fault são candidatas a realocação. Neste caso, os processos do sistema possuem o mesmo número de páginas

Substituição de Páginas

- Na política de substituição global, todas as páginas que estão em memória são analisadas como candidatas à substituição, independente do processo que gerou o page fault
- Na verdade, "todas" as páginas é muito forte. Algumas páginas são marcadas como bloqueadas, como por exemplos páginas utilizadas pelo sistema operacional, e não podem ser substituídas
  - Imagine o que acontece, se o SO tirar da memória o algoritmo de substituição de páginas!

Working Set

- Existe um desafio a se implementar a memória virtual, para decidir quais páginas permanecem na memória e quais deveriam sair
- Caso os processos não tenham um número suficiente de páginas, é provável que ocorra um número muito alto de page faults durante a sua execução, ocorrendo muitas operações de E/S
- Esse problema é conhecido como *thrashing*

Working Set

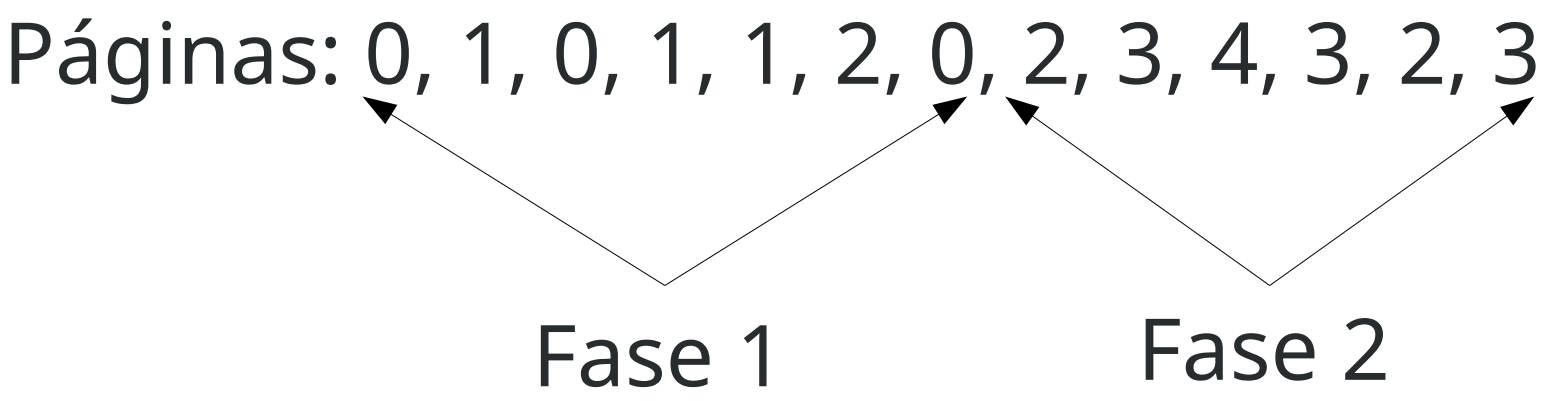
- Como o trashing, o sistema entra num estado onde os processos causam muitos page faults ao executar poucas instruções
- O working set surgiu com o intuito de diminuir o problema do trashing
- O princípio da localidade define dois tipos de localidade:
  - localidade espacial;
  - localidade temporal.

Working Set

- A localidade espacial diz que após uma referência a uma memória, é provável que as próximas referências sejam realizadas em regiões próximas
- A localidade temporal diz que após uma referência a uma posição de memória, é provável que ela seja referenciada em um curto espaço de tempo

Working Set

- As consequências desse princípio é que o programa pode ser divido em fases e, em cada uma dessas fases, apenas um subconjunto é referenciado durante a execução



Working Set

- O princípio da localidade é intuitivo, ao pensar que o programa é composto por loops: a tendência é que os mesmos dados sejam acessados
- No início da execução, existe um alto número de page faults, pois nenhum dado está na memória
- Após um período, o número de page faults cai novamente
- Durante a migração entre fases de um programa, o número de page faults se eleva, mas volta a cair

Working Set

- Petter Daning formulou o modelo em 1968.
- Dentro de uma janela do working set, o número de páginas distintas referenciadas é conhecido como o tamanho do working set

Páginas: 0, 1, 0, 1, 1, 2, 0, 2, 3, 4, 3, 2, 3

Δt1

Δt1: tamanho 3. Working set: 0, 1, 2

Working Set

- Esse modelo de working set auxilia a calcular quais são as páginas necessárias para execução
- Caso o limite de página de um processo seja maior que o working set, então há uma maior probabilidade de ocorrer page fault
- Casos onde o limite de páginas é muito pequeno leva a uma quantidade grande de page faults

Working Set

- Outra questão é como implementar no sistema operacional, qual tamanho de janela e como usar essas informações para estabelecer uma boa política de gerência de memória virtual?
- Esse modelo não deve ser implementando em ambientes com política de alocação de páginas fixa, ela deve ser variável para adaptar o número de página a medida do necessário

Working Set

- O working set é muito útil para analisar a taxa de paginação de um programa
  - O SO pode aumentar o número de páginas de um processo para permitir que um working set caiba na memória
  - O SO pode diminuir o número de páginas de um processo quando notar que eles não está sendo usado, de forma a liberar o frame para outro processo

Algoritmos de Substituição de Páginas

- No entanto, ainda há um problema fundamental na gerência de memória virtal
- A maior dificuldade não é decidir como e quando copiar, mas sim quais páginas retirar da memória quando outras são necessárias
- Para isso existem diversas técnicas para os algoritmos de substituição de páginas

Algoritmos de Substituição de Páginas

- Seu principal objetivo selecionar frames que serão substituídos com a menor chance de ser referenciado em um futuro próximo
- A partir do princípio de localidade, eles podem selecionar alguns para a remoção
- É necessário que o algoritmo execute rapidamente, que ele não seja lento, diminuindo o overhead do sistema operacional

Algoritmos de Substituição de Páginas

- Principais algoritmos:
  - Ótimo;
  - Aleatório;
  - First-in First-out (FIFO);
  - Least-Frequently-Used (LFU);
  - Least-Recently-Used (LRU);
  - Not-Recently-Used (NRU);
  - FIFO Circular.

Algoritmos de Substituição de Páginas

- **Ótimo**: algoritmo seleciona a página que não será mais referenciada no futuro
- Ou seleciona aquela que levará o maior intervalo de tempo para ser selecionada
- Na prática é impossível de ser implementado, mas é utilizado em testes de simulação como parâmetro de comparação para os outros algoritmos

Algoritmos de Substituição de Páginas

- **Aleatório:** o algoritmo seleciona aleatoriamente as páginas a serem substituídas
- Como o futuro é indeterminado, ele visa minimizar ao máximo o overhead de selecionar uma página
- Não possui uma eficiência muito boa

Algoritmos de Substituição de Páginas

- **FIFO (First-In-First-Out):** seleciona a página que está a mais tempo na memória, mantendo as páginas mais novas nela
- Implementando utilizando uma famosa estrutura de dados: fila
- No entanto considera apenas a criação da página, e não considera os acessos recentes: uma página que acabou de ser usada pode ser retirada

Algoritmos de Substituição de Páginas

- **LFU (Least-Frequently-Used):** o algoritmo seleciona a página menos utilizada
- Cria-se um overhead de manter um contador para cada acesso à memória realizado pelos programas
- Boa estratégia???

Algoritmos de Substituição de Páginas

- **LFU (Least-Frequently-Used):** o algoritmo seleciona a página menos utilizada
- Cria-se um overhead de manter um contador para cada acesso à memória realizado pelos programas
- Aparentemente boa estratégia, mas leva a uma injustiça: as páginas antigas, com um grande contador, nunca são retiradas e as páginas novas sempre são retiradas afinal seus contadores são muito pequenos

Algoritmos de Substituição de Páginas

- **LRU (Least-Recently-Used):** algoritmo seleciona página que está a mais tempo sem ser utilizada.
- Leva em conta o princípio da localidade
- Deve-se manter o tempo de acesso associado para cada frame
- Pode-se utilizar uma lista encadeada para manter os frames na ordem que foram acessados
- Possui um alto custo de implementação

Algoritmos de Substituição de Páginas

- **Not-Recently-Used (NRU):** simplificação do LRU. Na tabela de páginas, existe um bit de referência
- Quando a página é acessada ou carregada à memória, o bit de referência é alterado para 1
- Periodicamente, o sistema coloca todos os bits em 0 e, após um determinado tempo, devido aos acessos, os bits voltam a ser 1
- Assim o sistema distingue quais não foram usadas recentemente

Algoritmos de Substituição de Páginas

- **Not-Recently-Used (cont):** estratégia possui sucesso quando combinada ao bit de modificação.
- Assim o algoritmo priorizará retirar páginas que não foram modificadas e acessadas, evitando um page-out
- Se não encontrar, priorizará páginas que foram modificadas mas não foram acessadas, por terem menos chance de acordo com o princípio da localidade

Algoritmos de Substituição de Páginas

- **FIFO circular (clock):** algoritmo que utiliza FIFO porém está numa estrutura circular, semelhante à um relógio
- Presente em muitos sistemas Unix
- Existe um ponteiro que guarda a posição da página mais antiga na lista
- Cada página possui um bit de referência

Algoritmos de Substituição de Páginas

- **FIFO circular (clock)**
- Quando é necessário substituir uma página, verifica-se se o valor R do frame apontado
- Se R = 0, a página é selecionada para remoção
- Se R = 1, foi usada recentemente. R = 0 e passa-se ao próximo membro da lista
- O processo continua até encontrar R = 0 ou dar a volta na lista circular

Algoritmos de Substituição de Páginas

- **FIFO circular (clock)**
- Se o processo deu a volta, todos os bits foram colocados como 0 e deve-se utilizar a mais antiga, como em um FIFO puro
- Não há overhead de zerar todos os bits de "tempos em tempos"

Roteiro

- Introdução
- Endereçamento Virtual e Mapeamento
- Memória Virtual
- Memória Virtual por Paginação
  - Alocação/Substituição e Algoritmos de Substituição de Páginas
  - Implementação: Tamanho das Páginas/Proteção da Memória
- Memória Virtual por Segmentação
- Comparação de Paginação e Segmentação
- Memória Virtual com Paginação e Segmentação

Tamanho das Páginas

- O tamanho utilizado para uma página varia de acordo com o hardware, mas possui valores muito variáveis, de 512 B, 4 KB, 16 MB ou 1 GB

  Alguns sistemas permitem configurar o tamanho de uma página

- Selecionar páginas grandes diminui o tamanho da tabela de páginas, mas aumenta o problema da fragmentação interna

Tamanho das Páginas

- Páginas pequenas necessitam de tabelas de mapeamento maior, mas podem possui uma melhor utilização da memória
- Outro fator que influencia é o tempo de leitura e gravação na memória secundária: em discos o tempo de operar duas página de 512 bytes pode ser muito maior que operar uma de 1024
- Os sistemas modernos tendem a optar por páginas com alguns milhares de bytes: tipicamente 4KB (32 bits) ou 8KB (64 bits)*

Fonte: Linux System Programming. 2Nd Edition. O'Reilly Media. 2013.

Paginação em Múltiplos Níveis

- O tamanho da tabela de páginas pode ser um problema, ocupando até 4 MB de espaço
- Utilizar vários níveis em uma tabela de paginação pode ser uma solução para amenizar esse problema
- Nela, o endereço virtual é dividido em múltiplos números de página virtual de nível

Endereço Virtual:

NPV1	NPV2	Desloc.

Paginação em Múltiplos Níveis

- O fato de um programa poder endereçar tantas posições de memória não quer dizer que ele vai endereçar
- Com as tabelas de múltiplos níveis, são mantidas em memória uma quantidade menor de tabelas
- Existem tabelas de páginas de 1 nível (PDP-11), 2 níveis (VAX), 3 níveis (SunSPARC) e 4 níveis (Motorola 68030). No entanto um grande número de páginas causa um overhead muito grande

Proteção de Memória

- Em qualquer sistema multiprogramável, deve existir a proteção de memória entre os programas e o sistema operacional
- O SO deve proibir que programas alterem páginas que eles não possuem acesso
- Páginas do processo deve ser protegidas dele mesmo contra, se elas tiverem código executável

Proteção de Memória

- O mecanismo de memória virtual protege a memória de acesso indevidos, afinal cada processo possui sua própria tabela de mapeamento e a tradução de endereços é realizada pela MMU
- Assim, os programas não conseguem alterar dados de outros programas, eles apenas referenciam frames que eles possuem
- Para proteger acesso de leitura e gravação, deve-se adicionar alguns bits na tabela de mapeamento

Proteção de Memória

- Neste caso os bits (L, G) podem representar
  - (0, 0) Sem acesso
  - (1, 0) Acesso leitura
  - (1, 1) Acesso leitura e gravação
- A cada acesso à memória é necessário verificar os bits de acesso
- O hardware pode auxiliar nessa tarefa, mas essa tarefa é árdua

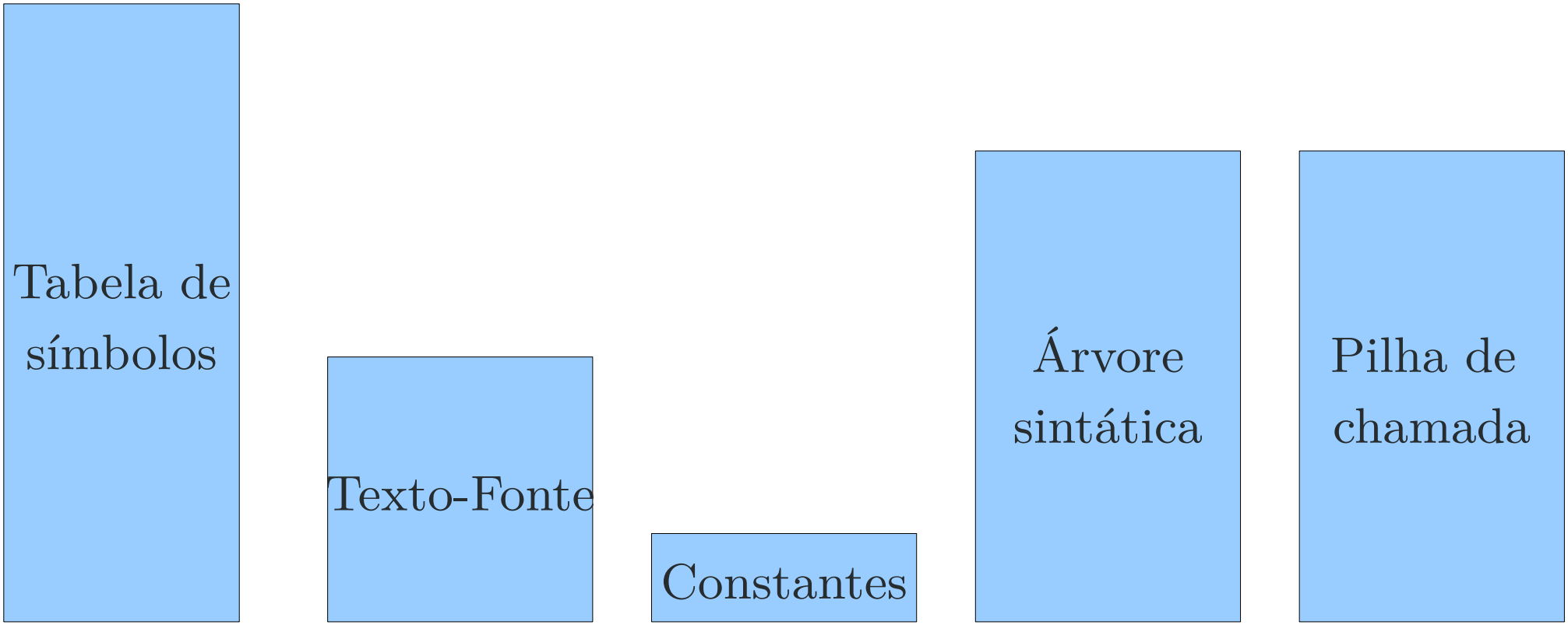
Roteiro

- Introdução
- Endereçamento Virtual e Mapeamento
- Memória Virtual
- Memória Virtual por Paginação
  - Alocação/Substituição e Algoritmos de Substituição de Páginas
  - Implementação: Tamanho das Páginas/Proteção da Memória
- Memória Virtual por Segmentação
- Comparação de Paginação e Segmentação
- Memória Virtual com Paginação e Segmentação

Memória Virtual por Segmentação

- Opondo-se à paginação, a segmentação divide o espaço de endereçamento virtual em blocos de tamanhos diferentes chamados de segmentos
- O programa é dividido em sub-rotinas e estruturas de dados e cada uma delas é colocada em um segmento na memória principal

Memória Virtual por Segmentação



- Na segmentação existe uma relação lógica entre o programa e a sua alocação na memória principal

Fonte: Tanenbaum 2016

Memória Virtual por Segmentação

- Definição de segmentos é frequentemente definida por compiladores
- Cada segmento consiste de uma sequência linear de endereços que podem alterar de tamanho durante a execução
- Existem limites para o tamanho máximo de segmentos e quantidade de segmentos por processo

Memória Virtual por Segmentação

- O espaço de endereçamento independente facilita o processo de compilação
- Em sistemas páginas, a alteração em um módulo exige a recompilação e religação de todos os módulos: mudar o tamanho de uma rotina pode influenciar no tamanho de todas

Memória Virtual por Segmentação

- O mecanismo de mapeamento da memória virtual por segmentação é muito semelhante ao de páginas: segmento, deslocamento, fim do segmento
- No entanto, na tabela de segmentos é incluído um campo para armazenar o tamanho do segmento

Memória Virtual por Segmentação

- Na paginação, uma estrutura que ocupar o tamanho superior ao de uma página, terá que ser associada a múltiplas páginas na memória
- Os algoritmos de substituição, modificação, as consideram independentes, no entanto o programa sabe que elas altamente relacionadas
- Na segmentação, apenas o tamanho do segmento deve ser alterado

Memória Virtual por Segmentação

- A segmentação não é útil quando o programa não está modularizado
- Se as aplicações não forem divididas em módulos, grandes segmentos de memória são carregados

Roteiro

- Introdução
- Endereçamento Virtual e Mapeamento
- Memória Virtual
- Memória Virtual por Paginação
  - Alocação/Substituição e Algoritmos de Substituição de Páginas
  - Implementação: Tamanho das Páginas/Proteção da Memória
- Memória Virtual por Segmentação
- Comparação de Paginação e Segmentação
- Memória Virtual com Paginação e Segmentação

Comparação entre Paginação e Segmentação

- Fragmentação:
- Na paginação ocorre o problema da fragmentação interna: como ela possui tamanho fixo, não é garantia que será completamente utilizada
- Na segmentação ocorre a fragmentação externa: onde áreas livres se encontram entre os segmentos, mas nenhum espaço é grande o suficiente para alocar um novo segmento

Comparação entre Paginação e Segmentação

- Proteção de Dados:
- A proteção dos dados na segmentação é mais simples de ser implementado
- Áreas de dados são segmentos leitura e escrita
- Áreas de código são segmentos leitura.
- Como estão separados em segmentos, é possível!
- Na paginação eles podem estar na mesma página e fica mais complexa essa proteção

Comparação entre Paginação e Segmentação

- Compartilhamento:
- O compartilhamento de dados na segmentação é mais simples
- O mapeamento é feito por estruturas lógicas e não páginas
- Não é necessário lidar com dados que ocupem várias páginas
- Na segmentação, basta que as tabelas possuam a mesma entrada

Comparação entre Paginação e Segmentação

- A segmentação permite compartilhar rotinas ou dados entre vários processos
- Por exemplo: código de bibliotecas compartilhadas ou código de processos iguais
- Esses trechos são colocados em segmentos que podem ser compartilhados entre os usuários, eliminando a necessidade de ter um segmento em cada espaço de endereçamento de processo

Comparação entre Paginação e Segmentação

- Em sistemas computacionais, os programas são associados a diversas bibliotecas compartilhadas (.dll no Windows, .so no Linux)
- Esse compartilhamento de dados é muito importante para reduzir o espaço utilizado por diversos processos em memória

Consideração	Paginação	Segmentação
Tamanho de Blocos	Iguais	Diferentes
Proteção	Complexa	Mais simples
Compartilhamento	Complexo	Mais simples
Estruturas de dados dinâmicas	Complexo	Mais simples
Fragmentação Interna	Pode existir	Não Existe
Fragmentação Externa	Não Existe	Pode Existir
Programação Modular	Dispensável	Indispensável
Alteração do Programa	Mais trabalho	Mais simples

Comparação entre Paginação e Segmentação

- A segmentação foi inventada para permitir que programas sejam divididos em espaços de endereçamento logicamente independentes e que o tamanho possa variar
- A paginação procura obter um grande espaço de endereçamento linear sem a necessidade de comprar mais memória física

Memória Virtual por Segmentação com Paginação

- Às vezes, os segmentos podem adquirir tamanhos muito grandes e deseja-se dividi-los, sem perder a noção de que os dados deles estão agrupados
- A técnica de paginação pode ser combinada com a segmentação: os endereços dos segmentos são divididos em múltiplas páginas
- Técnica busca combinar as vantagens de ambas

Memória Virtual por Segmentação com Paginação

- Então o endereço é divido em um número de segmento virtual e um número de página virtual

Endereço Virtual:

NVS	NVP	Desloc.

- Desta forma, o programador ainda está ciente que seus segmentos são divididos na memória, e o sistema consegue tratar páginas desses segmentos, quando eles se tornam muito grandes
- Essa arquitetura é a utilizada no MULTICS e os mais famosos computadores pessoais: Intel x86 e Intel x86-64

Referências

- Capítulo 3 – TANENBAUM, A. S. *Sistemas Operacionais Modernos*. 4ª ed. Prentice Hall, 2016.
- Capítulo 10 – MACHADO, F. B.; MAIA, L. P. *Arquitetura de Sistemas Operacionais*. 5ª ed. Rio de Janeiro: LTC, 2013.