



Sistemas Operacionais

Unidade Cinco

Prof. Flávio Márcio de Moraes e Silva



Organização da Memória

- Recurso limitado em relação às memórias secundárias
- Tarefa do SO otimizar o seu uso
- Questões a resolver:
 - Como dividir a memória? Blocos do mesmo tamanho ou de tamanhos diferentes?
 - Executar um programa em uma área de memória de tamanho exato ou onde couber?
 - Um programa deve estar em um bloco contíguo na memória ou pode ser dividido em pequenos blocos?



Gerenciamento de Memória

- Os projetistas de SO devem escolher quais estratégias usar de modo a obter o máximo de desempenho da memória
- As **Estratégias de Gerenciamento de Memória** determinam a organização de memória e como deve se comportar sob várias cargas
 - São realizadas por software e hardware de propósito especial



Gerenciamento de Memória

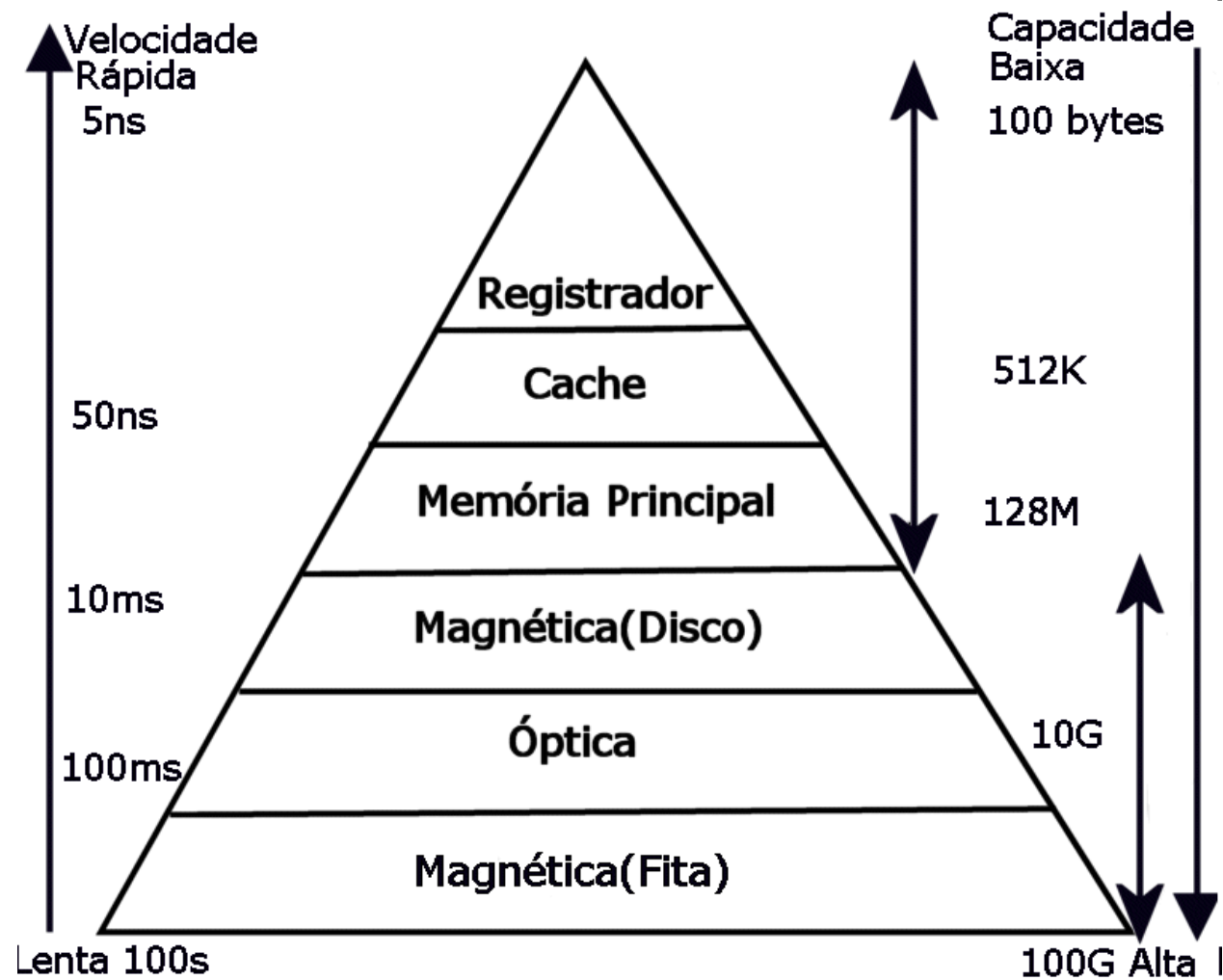
- O **Gerenciador de Memória** é responsável pelas estratégias de gerenciamento de memória e sua organização
 - Define como a memória é alocada e como responder as mudanças de necessidade de espaço de um processo



Hierarquia de Memória

- Qualquer programa só é executado na memória principal (RAM)
- Programas devem ser armazenados em mídias baratas, como unidades de disco
 - Só vão para a memória principal quando necessário
- Memórias secundárias chegam a ser 10^6 vezes mais lentas que a memória principal
- A hierarquia de memória é caracterizada pela sua velocidade e custo

Hierarquia de Memória





Estratégias de Gerenciamento de Memória

- **Estratégias de Posicionamento**
 - Determinam em qual lugar da memória colocar o programa
- **Estratégias de Busca**
 - Determinam quando transferir a próxima porção de um programa para a memória principal
 - **Sob demanda:** a medida que necessário busca
 - **Antecipada:** tenta carregar parte de um programa ou dados antes que sejam referenciados
- **Estratégias de Substituição**
 - Determina qual programa ou parte de um programa remover da memória quando ela estiver cheia



Memória Virtual

- É a separação da memória lógica da memória física
 - Dão a ilusão aos processos de que tem mais memória do que a contida no computador
- Tipos de endereçamento de memória virtual
 - **Referenciados por processo:** denominados endereços virtuais ou lógicos
 - **Endereços físicos:** estão fisicamente disponíveis na memória principal
- Sempre que um endereço virtual é acessado, o SO o traduz para um endereço real



MMU(*Memory Management Unit*)

- MMU(*Memory Management Unit*): componente de hardware responsável por prover os mecanismos básicos, usados pelo SO, para gerência de memória
 - Integrada ao chip do processador (circuito integrado)
 - Transforma endereços virtuais em físicos
 - Método 1: registradores de limite inferior e superior. Endereços lógicos e físicos iguais.
 - Método 2: registradores de limite e de base. Endereço físico = registrador base + endereço lógico.
- Delimita a região de memória acessada pelos processos



Alocação de Memória

- **Contígua**

- Em sistemas antigos os programas deveriam ser executados em uma porção de memória contígua
 - Caso não exista memória contígua suficiente para o programa ser executado, o mesmo não é executado

- **Não Contígua**

- Programas atuais podem ser divididos em blocos ou segmentos
 - Assim programas maiores podem ocupar lacunas deixadas por programas menores



Estratégias de Particionamento

Partições fixas

Memória Física

Sistema Operacional - 225 Kbytes
Partição 1 - 200 Kbytes
Partição 2 - 100 Kbytes
Partição 3 - 50 Kbytes
Partição 4 - 25 Kbytes

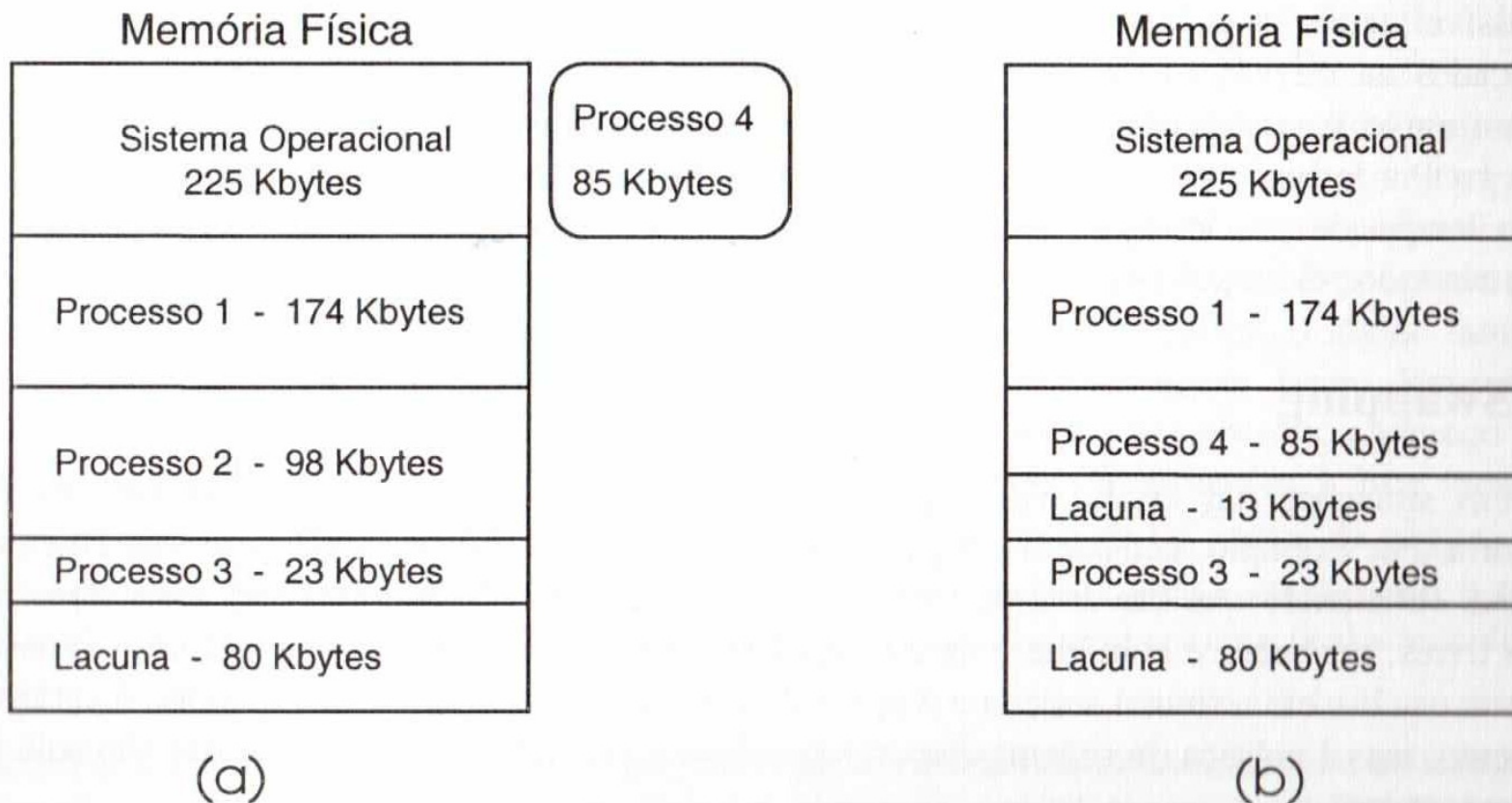
- Forma mais simples de gerência de memória
- Dividir a memória em partições de tamanho diferentes, mas fixos. Reservando um espaço ao SO.
- Problemas:
 - Fragmentação interna: colocar um processo em uma partição um pouco maior que o necessário. Ex: processo de 80kb na partição de 100kb.
 - Fragmentação externa: existem duas partições livres a de 100kb e a de 25kb, mas como o espaço de 125kb não é contíguo, não se pode abrigar um processo de 110kb.

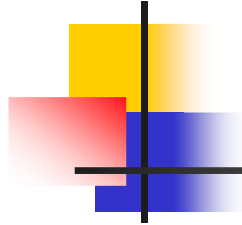


Partições variáveis

- Ajustar o tamanho das partições às necessidades dos processos
- SO mantém uma lista encadeada de lacunas (regiões livres de memória)
- Para alocar novo processo, formas de percorrer a lista de lacunas:
 - O primeiro que couber ([first fit](#))
 - O melhor que couber ([best fit](#))
 - O que pior couber ([worst fit](#))
 - first fit com busca após a última lacuna ([circular fit](#))
- Ao encerrar um processo, cria-se uma nova lacuna. Lacunas adjacentes são concatenadas
- Resolve fragmentação interna, mas aumenta a externa (muitas lacunas pequenas geradas)
- Alternativas:
 - Organização de lacunas em parágrafos de tamanho fixo 32kb (int 4kb)
 - Compactação ou reorganização da memória movendo processos (custo muito alto)

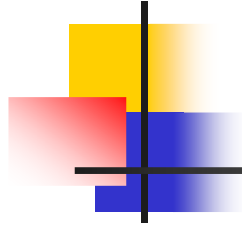
Partições Variáveis





Exercício

- Considere um sistema cuja gerência de memória é feita através de partições variáveis. Neste momento existe a seguinte lista de lacunas (áreas livres): 10k, 4k, 20k, 18k, 7k, 9k, 12k e 13k. Nessa ordem. Quais lacunas serão ocupadas pelas solicitações: 15k, 4k e 8k. Nessa ordem. Para os algoritmos abaixo:
- First fit
- Best fit
- Worst fit
- Circular fit



Exercício

- A gerência de memória é por partições variáveis.
- A memória tem 256kb. Totalmente livre.
- Chegaram 6 processos com os seguintes tamanhos: 32, 58, 64, 70, 12 e 20. Nessa ordem.
- Terminaram os processos com os tamanhos 32, 70 e 20.
- Chegaram três solicitações de tamanhos 40, 25 e 15. Nessa ordem.
- Mostre a memória após a execução dos algoritmos: First Fit, Best Fit, Worst Fit e Circular Fit.



Mapeamento de Bloco

- Fragmentação externa é gerada pelo fato de programas terem que ocupar regiões contíguas na memória.
- Solução: dividir processos em blocos lógicos, que podem ficar em áreas não contíguas.
 - O SO apenas monitora onde cada bloco da memória virtual foi posicionado na memória principal
- Blocos de tamanho **fixo** são denominados de **páginas** e sua organização de **paginação**
- Blocos com tamanhos **diferentes** são denominados de **segmentos** e sua organização de **segmentação**
- Alguns sistemas mesclam as duas técnicas



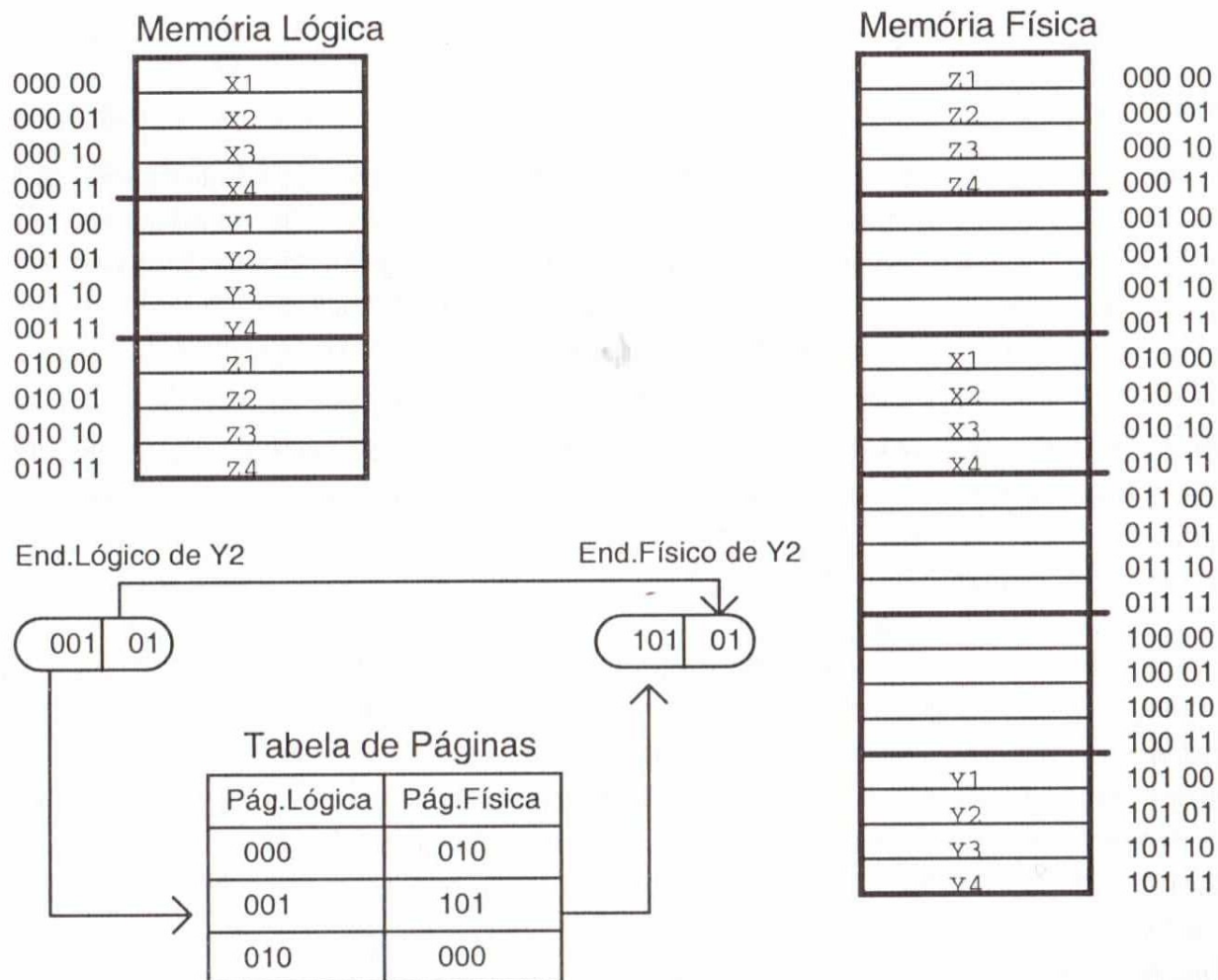
Paginação

- Uma funcionalidade de um processo só pode ser executada se a página referenciada estiver na memória principal
- Uma página ao ser transferida para a memória principal é armazenada em um bloco denominado **moldura de página** (*page frame*)
- Molduras de páginas começam em um endereço de memória física
- Criação de 1 tabela de páginas para cada processo

Tabela de Páginas

(mapear endereços lógicos em físicos)

- Exemplo:
- Memória lógica 12 bytes
- Memória física 24 bytes
- Páginas de 4 bytes
- 2 bits deslocar na página
- 3 bits entrada na tabela
- Máximo de 8 páginas
- Tabela de até 8 linhas
- Neste exemplo poderíamos ter memória lógica de até 32 bytes (8 x 4b), ou seja, programas com no máximo este tamanho.





Paginação

- Questão a resolver: qual o tamanho ideal de página?
 - Pequeno: muito trabalho para o gerenciador de memória, muitos acessos a disco e tabela de páginas grande.
 - Grande: resolve os problemas acima mas aumenta a fragmentação interna.



Paginação

- Vantagem

- Mais processos podem residir na memória principal ao mesmo tempo
 - Nem todas as páginas de um processo precisam estar na memória principal ao mesmo tempo (apenas as referenciadas)

- Desvantagem

- Aumento da complexidade do mecanismo de tradução de endereços
 - Deve indicar se a página reside ou não na memória principal
 - Caso exista a tabela de páginas fornecerá o número da moldura na memória principal
 - Caso não exista fornecerá a localização no armazenamento secundário
 - Quando uma página não existe na memória principal, é gerada uma falha ao SO para que carregue a página desejada



Tabela de páginas

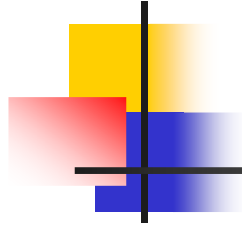
Informações adicionais

- Bit de referência (BR)
 - Esta ou não na memória RAM (0 = não e 1 = sim)
- Bit de modificação (BM)
 - Foi ou não alterada na RAM (0 = não e 1 = sim)
- Momento de entrada da página na memória
- Momento do último acesso a página
- Quantidade de acessos a página



Endereçando além de 4Gb com Windows de 32 bits

- Pentium-pro ou superiores: apesar de serem sistemas de 32bits, que na prática só poderiam endereçar 4Gb, conseguem endereçar até 64Gb usando instruções PAE (Physical Address Extensions) contidas no processador. Tal técnica é uma extensão lógica do barramento de endereços de memória (address bus) de 32 para 36bits.
- Na prática a PAE ativa recursos que fazem com que o Memory Manager (gerenciador de memória), passe a endereçar dados no formato de endereços de 64bits, limitados aos 36bits (64Gb) citados anteriormente.
- O Memory Manager utiliza uma tabela de índices, para criar um mapa para endereços de 64bits. Se uma requisição é feita para endereços até 4Gb esta é atendida diretamente, requisições entre 4GB e 64Gb são atendidas com base em consultas a esta tabela de referência para ponteiros de endereços.



Exercício

- Para um SO com páginas de 1kb, endereço lógico de 16bits e endereço físico de 20bits.
- Responda:
- Qual o maior programa possível (espaço lógico)?
- Qual o tamanho da memória RAM física?
- Qual o número de bits de entrada da tabela de páginas?
- Qual o número de linhas da tabela de páginas?



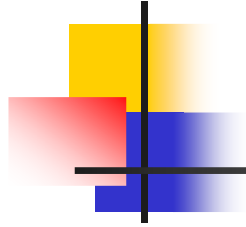
Solução

- Para um SO com páginas de 1kb, endereço lógico de 16bits e endereço físico de 20bits. Temos:
 - Espaço de endereçamento lógico (maior programa possível): deslocamento interno na página de 1kb = 10 bits. $16 - 10 = 6$ bits.
 $2^6 \times 1\text{kb} = 64\text{Kb}$
 - Espaço de endereçamento físico (memória principal): $2^{10} \times 1\text{kb} = 1\text{Mb}$
 - Bits de entrada na tabela de páginas: 6 bits
 - Número de linhas da tabela de páginas: $2^6 = 64$ linhas



Segmentação

- Na segmentação os dados e instruções são divididos em blocos chamados segmentos
 - Cada segmento deve ter uma localização contígua
 - Não precisam ser do mesmo tamanho. Podem ser tão grandes ou tão pequenos quanto precisarem ser
- Segmentação é um conceito lógico, e não físico (como na paginação)
- Segmentos de um programa são mantidos em memória só se precisarem ser executados em um determinado instante



Segmentação

- Se o segmento não estiver na memória principal, o sistema de memória virtual localizará o segmento na memória secundária e o trará para a memória principal
- Um segmento poderá ser colocado, de forma contígua, em qualquer lugar na memória principal, desde que seja suficientemente grande para contê-lo

Segmentação

Segmento 00 - Código

00000	C1
00001	C2
00010	C3
00011	C4
00100	C5
00101	C6

Segmento 01 - Dados

00000	D1
00001	D2
00010	D3
00011	D4

Segmento 10 - Pilha

00000	P1
00001	P2
00010	P3

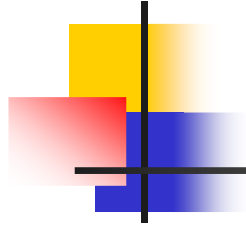
Memória Lógica

Tabela de Segmentos

Segmento	Base	Limite
00	01000	0110
01	00000	0100
10	10100	0011

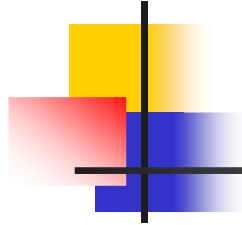
Memória Física

D1	00000
D2	00001
D3	00010
D4	00011
	00100
	00101
	00110
	00111
C1	01000
C2	01001
C3	01010
C4	01011
C5	01100
C6	01101
	01110
	01111
	10000
	10001
	10010
	10011
P1	10100
P2	10101
P3	10110
	10111



Sistemas de Segmentação/Paginação

- Segmentos são arranjados ao longo de várias páginas, ou seja, várias páginas formam um segmento
- Nem todas as páginas de um segmento precisam estar na memória principal



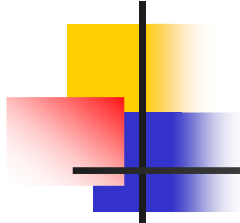
SWAPPING

- é utilizado quando não é possível manter todos os processos simultaneamente na memória
- um processo sofre swap-out quando ele é retirado da lista de prontos, inserido na lista de suspensos e copiado da memória para o disco
- um processo sofre swap-in quando ele retorna para a memória
- o swapping permite que o SO execute mais processos do que a memória normalmente suportaria
- o custo para os processos é alto
- o processo deve ficar no disco um tempo razoável para justificar o swapping
- é mais aceitável para processos que executam em segundo plano do que para processos interativos



Exercício

- Suponha um SO que trabalha com segmentação paginada. Cada processo pode ter no máximo 16 segmentos. Um segmento pode conter no máximo 8 páginas. Cada página ocupa 2 bytes. Suponha um programa com: segmento de código de 4 páginas, segmento de dados de 2 páginas e segmento de pilha de 3 páginas. Cada segmento aponta para uma tabela de páginas, que por sua vez mapeia páginas lógicas em páginas físicas.
- Mostre um esboço possível para a memória lógica, a tabela de segmentos, as tabelas de páginas e a memória física; de forma similar ao mostrado nos exemplos anteriores.



Gerenciamento da memória virtual: Thrashing

- Estratégias de paginação/segmentação exigem operações de E/S, que devem ser evitadas.
- Thrashing: é a excessiva transferência de páginas/segmentos entre a memória principal e a memória secundária. Problema existente tanto em paginação quanto na segmentação.
- Solução: o SO deve limitar o número de páginas de um processo na memória.



Gerenciamento da memória virtual: Working set

- Working Set de um processo é o conjunto de páginas que deve permanecer na memória principal para que este execute de forma eficiente, evitando a elevada taxa de paginação (thrashing).
- Sempre que um processo é criado, todas as suas páginas estão na memória secundária.
- O Working Set deve ter um limite máximo de páginas permitidas.



Gerenciamento da memória virtual: Memória Compartilhada

- Bastante útil para programas concorrentes.
- É bastante simples implementar o compartilhamento de código e dados entre vários processos, bastando que as entradas das tabelas de páginas/segmentos apontem para as mesmas páginas/segmentos na memória principal.
- Reduz o número de programas na memória principal e aumenta o número de usuários compartilhando o mesmo recurso.



Gerenciamento da Memória Virtual: Princípio da Localidade

- Páginas relacionadas à página que esta sendo executada, tendem a ser as próximas a executar
 - O sistema de paginação tende a fornecer subconjuntos de páginas
 - Favorece a forma como os programas são escritos
 - Variáveis relacionadas tendem a ficar uma perto da outra



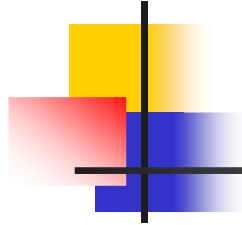
Gerenciamento da Memória Virtual: Princípio da Localidade

- **Localidade Espacial** – páginas próximas a em execução tendem a ser as próximas a executar
- **Localidade Temporal** – páginas utilizadas recentemente tendem a ser utilizadas em um futuro próximo



Paginação por Demanda

- Armazena a página do programa que contém a primeira instrução
 - As demais páginas só são carregadas na memória quando requisitadas
- **Vantagem**
 - O espaço da memória principal não é desperdiçado
 - Só estarão na memória as páginas realmente utilizadas
- **Desvantagem**
 - O processo deve esperar a cada nova página requisitada
 - Pode ser custoso se existirem vários processos na memória



Paginação Antecipada

- O SO carrega páginas na memória sem ter a certeza que serão utilizadas (previsão de uso)
 - Apenas quando houver espaço em memória disponível
- Autores criticam a paginação antecipada por causa da dificuldade de prever o caminho de execução que o programa tomará
 - Pode gerar sobrecarga pelo carregamento de páginas erradas (indevidas)
- Com o aumento do desempenho e da capacidade do *hardware*, essa técnica tornou-se interessante



Paginação Antecipada

- Uma estratégia que não defina com exatidão as páginas necessárias ao processo, pode causar desempenho pior que a paginação por demanda
- Estratégias de paginação antecipada são combinadas com a paginação por demanda
 - Quando um processo gera uma falha de página, o SO carrega a página faltante e outras páginas próximas a requisitada (**localidade espacial**)



Substituição de páginas

- O que fazer quando não houver mais molduras disponíveis para uma página requisitada
- O SO deve escolher qual página deve ser substituída (retirada ou sobrescrita) por uma nova página
 - Se a página escolhida não tiver sido modificada, então retire-a (sem ir a memória secundária)
 - Se a página escolhida tiver sido modificada, deverá ser salva antes de ser substituída
- Estratégias de substituição tentam reduzir o número de faltas de página



Algoritmos de substituição

- Classe global: a página a ser substituída pode estar relacionada a qualquer um dos processos presentes na memória.
- Classe local: a substituição esta relacionada ao comportamento dos processos durante a execução, princípio da localidade.
- Exemplos a seguir.



Estratégias de Substituição de Páginas

- **Substituição Aleatória de Páginas** (Random Page Replacement – RAND)
 - Fácil de implementar
 - Baixa sobrecarga do SO
 - Cada página na memória tem a mesma probabilidade de ser substituída
 - **Desvantagem**
 - Pode substituir a próxima página a ser utilizada
 - **Vantagem**
 - Toma a decisão de qual página substituir rapidamente
 - Como há várias páginas na memória, é pouco provável que substitua a próxima página a ser utilizada



Estratégias de Substituição de Páginas

- **Estratégia FIFO** (Primeiro a Entrar, Primeiro a Sair)
 - Substitui a página que reside na memória a mais tempo
 - **Desvantagem**
 - Pode substituir uma página intensamente utilizada



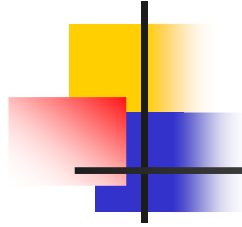
Estratégias de Substituição de Páginas

- **Estratégia MRU** (Menos Recentemente Utilizada)
 - Substitui a página que passou mais tempo na memória sem ser referenciada
 - Baseia-se na localidade temporal
 - Causa sobrecarga no SO
 - Usa uma fila para ordenar a mais recentemente utilizada para a menos utilizada
 - Pode falhar, pois a página menos recentemente utilizada pode ser a próxima a ser utilizada



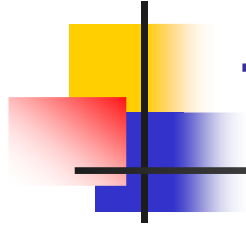
Estratégias de Substituição de Páginas

- **Estratégia MFU** (Menos Frequentemente Utilizada)
 - O SO substitui a página menos frequentemente utilizada
 - Páginas pouco referenciadas no presente, provavelmente serão pouco referenciadas no futuro
 - Pode ser implementada utilizando um contador por página
 - Causa sobrecarga no SO
 - **Problema 1**
 - Pode substituir uma página que acabou de ser colocada na memória
 - **Problema 2**
 - Páginas muito utilizadas no passado e que no presente não serão utilizadas podem ficar desnecessariamente na memória



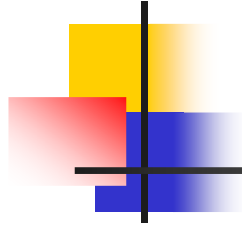
Liberação de Página

- Um processo informa as páginas que deseja utilizar
- Quando não for mais utilizar uma determinada página, o processo deve liberá-la
 - Liberação voluntária de página
- Os usuários não podem tomar essa decisão, mas os compiladores e o SO sim
- Você como programador pode fazer isto, como: alocação e desalocação adequada de recursos na programação.



Tamanho de Página

- Uma característica importante de um SO é o tamanho das páginas e de suas molduras
 - Não há um padrão industrial de tamanho de página
- Os resultados mostram a necessidade de páginas pequenas
- Mas com o aumento da memória e do tamanho dos programas, páginas maiores tornam-se desejáveis



Tamanho de Página

- Paginação leva a uma menor fragmentação, pois apenas poderá haver fragmentação interna na última página.
- A fragmentação é consequência do tamanho da página.
- Páginas pequenas, tabelas maiores, maior taxa de paginação e aumento do número de acessos à memória secundária. Mas menor fragmentação interna.
- Atualmente o tamanho da página esta associado ao hardware e varia de sistema para sistema, normalmente entre 512 bytes e 64 kb.



Exercício

- O WSL (Working Set List) de um processo é o número máximo de páginas do processo, que podem estar na memória. Para este exercício suponha um WSL de quatro páginas. Suponha um computador, onde endereço virtual é de 16 bits e as páginas têm tamanho 2Kb. Inicialmente, nenhuma página está na memória principal. Um programa faz referência a endereços virtuais situados nas páginas 0, 7, 2, 7, 5, 8, 9, 2 e 4, nesta ordem.
- A) Quantos bits do endereço virtual destinam-se ao número da página? E ao deslocamento interno na página?
- B) Indique o comportamento da política de substituição MRU, mostrando, a cada referência, quais páginas estão em memória, as faltas de páginas causadas e as páginas escolhidas para saírem da memória.
- C) Faça o procedimento da letra B agora para a política FIFO.