

Sistemas Operacionais

Profª. Dra. Kalinka Regina Lucas Jaquie Castelo Branco
kalinka@icmc.usp.br

Apresentação baseada nos slides do Prof. Dr. Antônio Carlos Sementille e da Profª. Dra. Luciana A. F. Martimiano e nas transparências fornecidas no site de compra do livro "Sistemas Operacionais Modernos"

Gerenciamento de Memória

- Recurso importante;
- Tendência atual do software
 - Lei de *Parkinson*: "Os programas se expandem para preencher a memória disponível para eles" (adaptação);
- Hierarquia de memória:
 - Cache;
 - Principal;
 - Disco;

2

Gerenciamento de Memória

- Idealmente os programadores querem uma memória que seja:
 - Grande
 - Rápida
 - Não Volátil
 - Baixo custo
- Infelizmente a tecnologia atual não comporta tais memórias
- A maioria dos computadores utiliza Hierarquia de Memórias que combina:
 - Uma pequena quantidade de memória cache, volátil, muito rápida e de alto custo
 - Uma grande memória principal (RAM), volátil, com centenas de MB ou poucos GB, de velocidade e custo médios
 - Uma memória secundária, não volátil em disco, com gigabytes (ou terabytes), velocidade e custo baixos

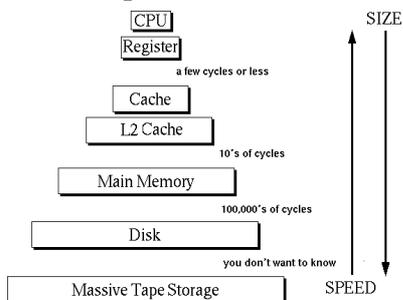
3

Gerenciamento de Memória Hierarquia de Memória

- **Cache**
 - Pequena quantidade
 - k bytes
 - Alto custo por byte
 - Muito rápida
 - Volátil
- **Memória Principal**
 - Quantidade intermediária
 - M bytes
 - Custo médio por byte
 - Velocidade média
 - Volátil
- **Disco**
 - Grande quantidade –
 - G bytes
 - Baixo custo por byte
 - Lenta
 - Não volátil

4

Hierarquia de Memória



5

Gerenciamento de Memória

! Cabe ao Gerenciador de Memória gerenciar a hierarquia de memória

! Controla as partes da memória que estão em uso e quais não estão, de forma a:

- ! alocar memória aos processos, quando estes precisarem;
- ! liberar memória quando um processo termina; e
- ! tratar do problema do *swapping* (quando a memória é insuficiente).

6

Gerenciamento de Memória

- Para cada tipo de memória:
 - Gerenciar espaços livres/ocupados;
 - Alocar processos/dados na memória;
 - Localizar dado.
- Entre os níveis de memória, deve-se:
 - Gerenciar as trocas.
- **Gerenciador de memória:**
 - Responsável por alocar e liberar espaços na memória para os processos em execução;
 - Responsável por gerenciar chaveamento entre a memória principal e o disco, e memória principal e memória *cache*;

{ 7 }

Gerenciamento Básico de Memória

- Sistemas de Gerenciamento de Memória, podem ser divididos em 2 classes:
 - Sistemas que, durante a execução levam e trazem processos entre a memória principal e o disco (troca de processos e paginação)
 - Sistemas mais simples, que não fazem troca de processos e nem paginação

{ 8 }

Gerenciamento Básico de Memória

Monoprogramação sem trocas de processos ou Paginação

Monoprogramação sem trocas de processos ou Paginação

Sistemas Mono-usuários: gerência de memória é bem simples, pois toda a memória é alocada à próxima tarefa, incluindo a área do S.O.

Erros de execução podem vir a danificar o S.O.

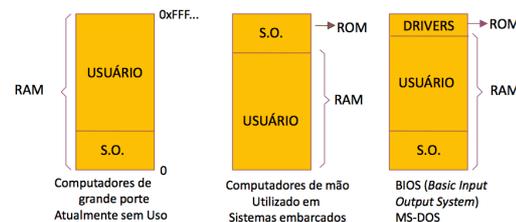
Neste caso, a destruição do S.O. é um pequeno inconveniente, resolvido pelo recarregamento do mesmo.

{ 9 }

Gerenciamento Básico de Memória

Monoprogramação sem trocas de processos ou Paginação

Gerenciamento mais simples



Três esquemas simples de organização de memória

- Um sistema operacional com um processo de usuário

{ 10 }

Multiprogramação com partições Fixas

Sistemas Monoprogramados: raramente usados atualmente.

Sistemas modernos: permitem **multiprogramação**

A maneira mais comum de realizar a multiprogramação é dividir simplesmente a memória em n partições (provavelmente de tamanhos diferentes).

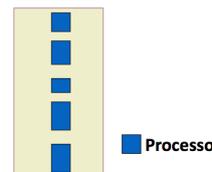
Esta divisão pode ser feita de maneira manual, quando o sistema é inicializado

Ao chegar, um *job*, pode ser colocado em uma fila de entrada associada à menor partição, grande o suficiente para armazená-lo

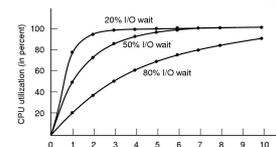
{ 11 }

Modelagem da Multiprogramação

- Modelo de Multiprogramação
 - Múltiplos processos sendo executados;
 - Eficiência de CPU;



Memória Principal - RAM



A utilização da CPU como uma função do número de processos na memória.

Necessidade de Particionamento da Memória Principal.

{ 12 }

Modelagem da Multiprogramação

O uso da Multiprogramação pode melhorar a utilização da CPU

Considerando a utilização da CPU de modo probabilístico:

- Suponha que um processo gaste uma fração p de seu tempo esperando pela finalização de sua solicitação de E/S
- Com n processos simultâneos na memória, a probabilidade de todos os n processos estarem esperando por E/S (situação em que a CPU está ociosa) é p^n
- A utilização da CPU é dada pela fórmula:
$$\text{utilização da CPU} = 1 - p^n$$

13

Gerenciamento de Memória

- Multiprogramação → Vários processos na memória:
 - Proteção: Como proteger os processos uns dos outros e o *kernel* de todos os processos?
 - Necessidade de realocação: Como tratar a realocação? O processo pode estar em diferentes posições da memória.
- Todas as soluções envolvem equipar a CPU com um hardware especial → **MMU (*memory management unit*)**.

14

Gerenciamento de Memória

- **MMU** (do inglês **Memory Management Unit**) é um dispositivo de **hardware** que transforma endereços virtuais em endereços físicos.
- Na MMU, o valor no registro de realocação é adicionado a todo o endereço lógico gerado por um processo do utilizador na altura de ser enviado para a **memória**. O programa do utilizador manipula endereços lógicos; ele nunca vê endereços físicos reais.

VAMOS VOLTAR NISSO MAIS A FRENTE OK?

15

Relocação e Proteção

- Pode não ter certeza de onde o programa será carregado na memória
 - As localizações de endereços de localização das variáveis e do código das rotinas não podem ser absolutos
 - Deve-se manter um programa fora das partições de outros processos
- Uso de valores de base e limite
 - Os endereços das localizações são somados a um valor de base para mapear um endereço físico
 - Valores de localizações maiores que um valor limite são considerados erro

16

Gerenciamento de Memória

- **Realocação:**
 - Quando um programa é *linkado* (programa principal + rotinas do usuário + rotinas da biblioteca → executável) o *linker* deve saber em que endereço o programa irá iniciar na memória;
 - Nesse caso, para que o *linker* não escreva em um local indevido, como por exemplo na área do SO (100 primeiros endereços), é preciso de realocação:
 - $\#100 + \Delta \rightarrow$ que depende da partição!!!
- **Proteção:**
 - Com várias partições e programas ocupando diferentes espaços da memória é possível acontecer um acesso indevido.

17

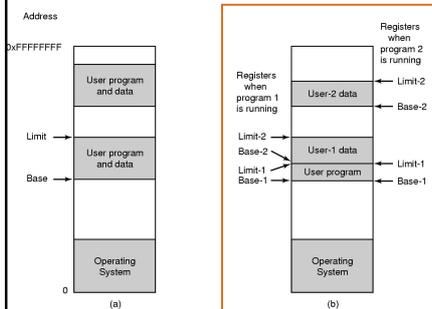
Gerenciamento de Memória

- Solução para ambos os problemas:
 - 2 registradores → base e limite
 - Quando um processo é escalonado o registorador-base é carregado com o endereço de início da partição e o registorador-limite com o tamanho da partição;
 - O registorador-base torna impossível a um processo uma remissão a qualquer parte de memória abaixo de si mesmo.
 - Automaticamente, a MMU adiciona o conteúdo do registorador-base a cada endereço de memória gerado;
 - Endereços são comparados com o registorador-limite para prevenir acessos indevidos.

18

Gerenciamento de Memória

Registadores base e limite



b) MMU mais sofisticada → dois pares de registradores: segmento de dados usa um par separado;
 ■ MMU modernas têm mais pares de registradores.

Gerenciamento de Memória

Partições/Alocação

- Particionamento da memória pode ser realizado de duas maneiras:
 - Partições fixas (ou alocação estática);
 - Partições variáveis (ou alocação dinâmica);
- **Partições Fixas:**
 - Tamanho e número de partições são fixos (estáticos);
 - Não é atrativo, porque partições fixas tendem a desperdiçar memória (Qualquer espaço não utilizado é literalmente perdido)
 - Mais simples.

20

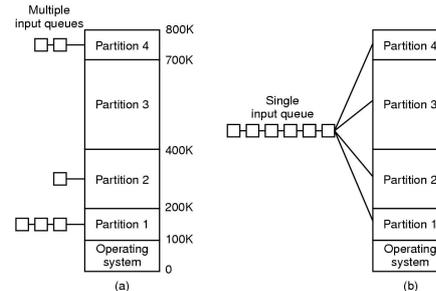
Gerenciamento de Memória

Partições Fixas

- Partições Fixas:
 - Filas múltiplas:
 - Problema: filas não balanceadas;
 - Fila única:
 - Facilita gerenciamento;
 - Implementação com Lista:
 - Melhor utilização da memória, pois procura o melhor processo para a partição considerada;
 - Diferentes algoritmos podem ser considerados para alocar os processos.

21

Multiprogramação com partições Fixas



- Partições de memória fixa
 - fila separada para cada partição
 - uma única fila de entrada

22

Gerenciamento de Memória

Partições Fixas

- Partições Fixas: problemas com fragmentação:
 - **Interna:** desperdício dentro da área alocada para um processo;
 - Ex.: processo de tamanho 40K ocupando uma partição de 50K;
 - **Externa:** desperdício fora da área alocada para um processo;
 - Duas partições livres: PL1 com 25k e PL2 com 100k, e um processo de tamanho 110K para ser executado;
 - Livre: 125K, mas o processo não pode ser executado;

23

Gerenciamento de Memória

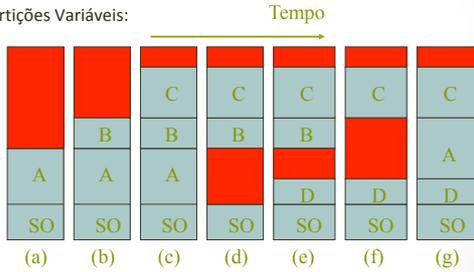
Partições Variáveis

- **Partições Variáveis:**
 - Tamanho e número de partições variam;
 - Otimiza a utilização da memória, mas complica a alocação e liberação da memória;
 - Partições são alocadas dinamicamente;
 - SO mantém na memória uma lista com os espaços livres;
 - Menor fragmentação interna e grande fragmentação externa;
 - Solução: Compactação;

24

Gerenciamento de Memória Partições Variáveis

- Partições Variáveis:



■ Memória livre

25

Gerenciamento de Memória

- Minimizar espaço de memória inutilizados:
 - Compactação: necessária para recuperar os espaços perdidos por fragmentação; no entanto, muito custosa para a CPU;
- Técnicas para alocação dinâmica de memória:
 - Bitmaps;
 - Listas Encadeadas.

26

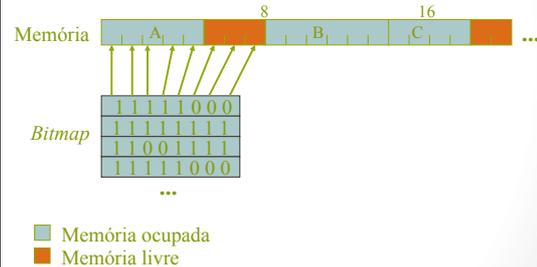
Gerenciamento de Memória

- Técnica com *Bitmaps*:
 - Memória é dividida em unidades de alocação em kbytes;
 - Cada unidade corresponde a um *bit* no *bitmap*:
 - 0 → livre
 - 1 → ocupado
 - Tamanho do *bitmap* depende do tamanho da unidade e do tamanho da memória;
 - Ex.:
 - unidades de alocação pequenas → *bitmap* grande;
 - unidades de alocação grandes → perda de espaço.

27

Gerenciamento de Memória

- Técnica com *Bitmaps*:

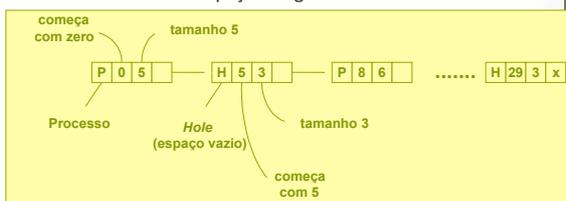


28

Gerenciamento de Memória

- Técnica com Listas Encadeadas:
 - Uma lista para os espaços vazios e outra para os espaços cheios, ou uma lista para ambos!

“espaço = segmento”



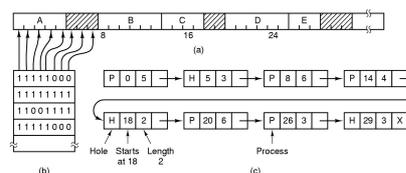
29

Gerenciamento de Memória com Listas encadeadas

Outra maneira de gerenciar memória é manter uma lista encadeada de segmentos de memória alocados e segmentos disponíveis

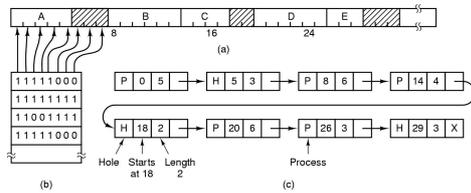
Cada elemento desta lista especifica:

- um segmento disponível (H), ou alocado a um processo (P),
- o endereço onde se inicia este segmento
- e um ponteiro para o próximo elemento



30

Gerenciamento de memória com Mapas de Bits



- (a) Uma parte da memória com 5 processos e 3 buracos
- As regiões em branco (1 no bitmap) marcam as unidades já alocadas
 - As regiões sombreadas (0 no bitmap) marcam unidades desocupadas
- (b) O Bitmap correspondente
- (c) A mesma informação como uma lista encadeada

31

Gerenciamento de Memória

- Algoritmos de Alocação** → quando um novo processo é criado:
 - FIRST FIT**
 - 1º segmento é usado;
 - Rápido, mas pode desperdiçar memória por fragmentação.
 - NEXT FIT**
 - 1º segmento é usado;
 - Mas na próxima alocação inicia busca do ponto que parou anteriormente;
 - Possui desempenho inferior.

32

Gerenciamento de Memória

- BEST FIT**
 - Procura na lista toda e aloca o espaço que mais convém;
 - Menor fragmentação;
 - Mais lento.
- WORST FIT**
 - Aloca o maior espaço disponível.
- QUICK FIT**
 - Mantém listas separadas para os espaços mais requisitados.

33

Gerenciamento de Memória

- Cada algoritmo pode manter listas separadas para processos e para espaços livres:
 - Vantagem:**
 - Aumenta desempenho;
 - Desvantagens:**
 - Aumenta complexidade quando espaço de memória é liberado – gerenciamento das listas;
 - Fragmentação.

34

Gerenciamento de Memória

Alocação de segmentos livres

Principais Consequências

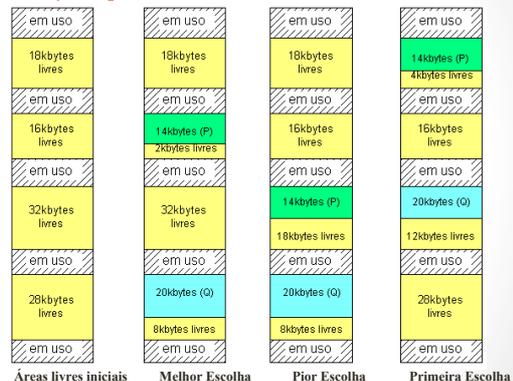
IA melhor escolha: deixa o menor resto, porém após um longo processamento poderá deixar “buracos” muito pequenos para serem úteis.

IA pior escolha: deixa o maior espaço após cada alocação, mas tende a espalhar as porções não utilizadas sobre áreas não contínuas de memória e, portanto, pode tornar difícil alocar grandes jobs.

IA primeira escolha: tende a ser um meio termo entre a melhor e a pior escolha, com a característica adicional de fazer com que os espaços vazios migrem para o final da memória.

Gerenciamento de Memória

Alocação de segmentos livres



36

Gerenciamento de Memória

O que fazer quando não existe espaço suficiente para todos os processos ativos?

- **Swapping**
 - Chaveamento de processos inteiros entre a memória principal e o disco.
- **Overlays – Memória Virtual**
 - Programa de são divididos em pedaços menores
 - Pedaços são chaveados entre a memória principal e o disco.

37

Gerenciamento de Memória

- **Swapping**
 - Chaveamento de processos inteiros entre a memória principal e o disco;
 - Transferência do processo da memória principal para a memória secundária (normalmente o disco): *Swap-out*;
 - Transferência do processo da memória secundária para a memória principal: *Swap-in*;
 - Pode ser utilizado tanto com partições fixas quanto variáveis.

38

Gerenciamento de Memória

- Programas maiores que a memória eram divididos em pedaços menores chamados de *overlays* – programador;
 - **Desvantagem:** custo muito alto.
- **Memória Virtual**
 - Sistema operacional é responsável por dividir o programa em *overlays*;
 - Sistema operacional realiza o chaveamento desses pedaços entre a memória e o disco.

39

Gerenciamento de Memória Memória Virtual (MV)

- Programas maiores que a memória eram divididos em pedaços menores chamados *overlays*;
 - Programador define áreas de *overlay*;
 - **Vantagem:** expansão da memória principal;
 - **Desvantagem:** custo muito alto.

40

Gerenciamento de Memória Memória Virtual (MV)

- Sistema operacional é responsável por dividir o programa em *overlays*;
- Sistema operacional realiza o chaveamento desses pedaços entre a memória principal e o disco;
- Década de 60: ATLAS → primeiro sistema com MV (Universidade Manchester - Reino Unido);
- 1972: sistema comercial: IBM System/370.

41

Gerenciamento de Memória Memória Virtual (MV)

- Com MV existe a sensação de se ter mais memória principal do que realmente se tem;
- O *hardware* muitas vezes implementa funções da gerência de memória virtual:
 - SO deve considerar características da arquitetura.

42

Gerenciamento de Memória Memória Virtual

- Espaço de Endereçamento Virtual de um processo é formado por todos os endereços virtuais que esse processo pode gerar;
- Espaço de Endereçamento Físico de um processo é formado por todos os endereços físicos/reais aceitos pela memória principal (RAM).

43

Gerenciamento de Memória Memória Virtual

- Um processo em Memória Virtual faz referência a endereços virtuais e não a endereço reais de memória RAM;
- No momento da execução de uma instrução, o endereço virtual é traduzido para um endereço real, pois a CPU manipula apenas endereços reais da memória RAM → **MAPEAMENTO**.

44

Gerenciamento de Memória Mapeamento MV

- MMU: Realiza **mapeamento** dos endereços lógicos (usados pelos processos) para endereços físicos;



45

Gerenciamento de Memória Memória Virtual

- Endereços virtuais formam um espaço de endereçamento virtual;
- Mapeamento entre endereços reais e virtuais é realizado pela MMU;
- Técnicas de MV:
 - Paginação;
 - Segmentação.

46

Gerenciamento de Memória Memória Virtual

- Paginação:
 - Blocos de tamanho fixo chamados de **páginas**;
 - SO mantém uma lista de todas as páginas;
 - O espaço de endereçamento virtual é dividido em páginas virtuais.
- Segmentação:
 - Blocos de tamanho arbitrário chamados **segmentos**;
 - SO mantém uma lista de todos os segmentos;
 - O espaço de endereçamento virtual é dividido em segmentos virtuais.

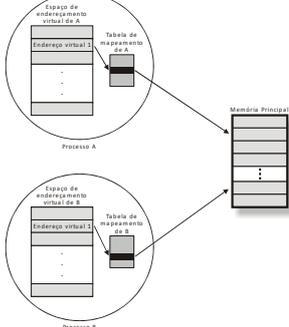
47

Gerenciamento de Memória Memória Virtual - Paginação

- Memória Principal e Memória Secundária são organizadas em páginas de mesmo tamanho;
- Página é a unidade básica para transferência de informação;
- Tabela de páginas: responsável por armazenar informações sobre as páginas virtuais:
 - argumento de entrada → número da página virtual;
 - argumento de saída (resultado) → número da página real (ou moldura de página - *page frame*).

48

Gerenciamento de Memória Memória Virtual - Paginação



49

Gerenciamento de Memória Memória Virtual

- Exemplo:
 - Páginas de 4Kb
 - 4096 bytes/endereços (0-4095);
 - 64Kb de espaço virtual;
 - 32Kb de espaço real;
- Temos:
 - 16 páginas virtuais;
 - 8 páginas reais;

50

Gerenciamento de Memória Memória Virtual

Espaço Virtual X Tamanho da Página

Espaço de Endereçamento Virtual	Tamanho da página	Número de páginas	Número de entradas nas tabela de páginas
2^{32} endereços	512 bytes	2^{23}	2^{23}
2^{32} endereços	4 kbytes	2^{20}	2^{20}
2^{64} endereços	4 kbytes	2^{52}	2^{52}
2^{64} endereços	64 kbytes	2^{48}	2^{48}

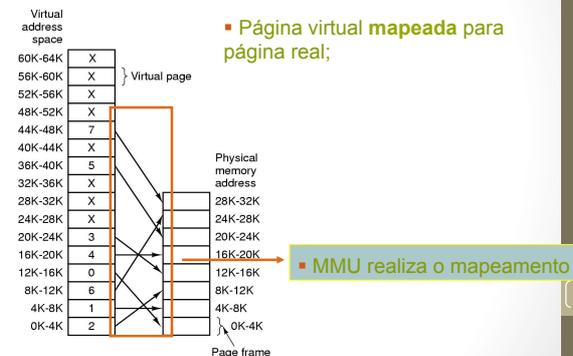
51

Gerenciamento de Memória Memória Virtual - Paginação

- Problemas:
 - Fragmentação interna;
 - Definição do tamanho das páginas;
 - Geralmente a MMU que define e não o SO;
 - Páginas maiores: leitura mais eficiente, tabela menor, mas maior fragmentação interna;
 - Páginas menores: leitura menos eficiente, tabela maior, mas menor fragmentação interna;
 - Sugestão: 1k a 8k;
- Mapa de bits ou uma lista encadeada com as páginas livres.

52

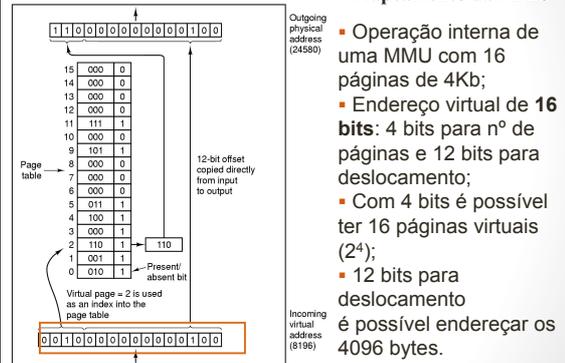
Gerenciamento de Memória Endereço Virtual → Endereço Real



53

Gerenciamento de Memória Memória Virtual - Paginação

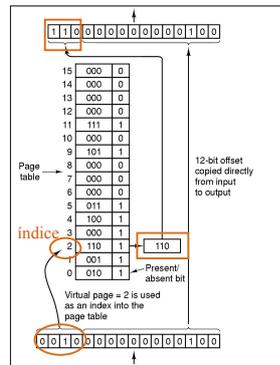
Mapeamento da MMU



54

Gerenciamento de Memória Memória Virtual - Paginação

Mapeamento da MMU



- Número da página virtual é usado como índice;
- Se página está na memória RAM, então o nº da página real (110) é copiado para os três bits mais significativos do endereço de saída (real), juntamente com o deslocamento sem alteração;
- Endereço real com 15 bits é enviado à memória.

55

Gerência de Memória Memória Virtual - Paginação

- Algumas questões que surgem com relação à Paginação:
 - Onde armazenar a tabela de páginas?

56

Gerência de Memória Memória Virtual - Paginação

- A Tabela de páginas pode ser armazenada de três diferentes maneiras:
 - Em conjunto de registradores, se a memória for pequena
 - Vantagem: rápido
 - Desvantagem: precisa carregar toda a tabela nos registradores a cada chaveamento de contexto.
 - Na própria memória RAM – MMU gerencia utilizando dois registradores:
 - Registrador Base da tabela de páginas (PTBR – *page table base register*): indica o endereço físico de memória onde a tabela está alocada
 - Registrador Limite da tabela de páginas (PTLR – *page table limit register*): indica o número de entradas da tabela (número de páginas)
 - Precisa de dois acessos à memória: um para acessar a tabela de páginas e outro para acessar a posição de memória.

57

Gerência de Memória Memória Virtual - Paginação

- Em memória *cache* da MMU
 - Também conhecida como TLB (*Translation Lookside Buffer – buffer de tradução dinâmica*);
 - Hardware especial para mapear endereços virtuais para endereços reais sem ter que passar pela tabela de páginas na memória principal;
 - Melhora o desempenho.

58

Gerências de Memória Memória Virtual - Paginação

- Projeto mais simples:
 - Uma única tabela de páginas que consista em um vetor de registradores rápidos em hardware (um registrador para cada entrada);
 - Quando o processo estiver para ser executado, o SO carregará esses registradores a partir de uma cópia da tabela de páginas desse processo mantida na memória;
 - Vantagem: Não requer nenhum acesso à memória durante a tradução
 - Desvantagens:
 - Caro!
 - Ter que carregar toda a tabela de páginas em cada troca de contexto.
- Tabela de páginas totalmente na memória.
- O Hardware necessário resume-se a um único registrador (que aponta para o início da tabela de páginas)
- Desvantagem:
 - A execução de uma instrução implicará em pelo menos dois acessos à memória
 - O primeiro para acessar a tabela de páginas (e descobrir o endereço físico desta instrução)
 - O segundo para buscar a respectiva instrução na memória. Isso sem falar nos operandos da instrução que podem estar em memória.

59

60

Gerências de Memória Memória Virtual - Paginação

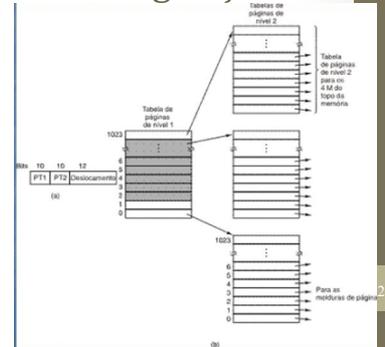
Tabela de Página Multinível

- O objetivo é evitar manter toda a tabela de páginas na memória durante todo o tempo
- Apresenta-se como uma solução para o dimensionamento da tabela de páginas
- Uso de dois apontadores e um deslocamento
- Exemplo: Tabela de dois níveis
 - O endereço de 32 bits de endereço dividido em 3 campos
 - PT1 [10 bits] : indexa o primeiro nível da tabela
 - PT2 [10 bits] : indexa o segundo nível da tabela
 - Deslocamento [12 bits] => paginas de 4 KB

61

Gerências de Memória Memória Virtual - Paginação

- 1o. nível com 1024 entradas
- Cada uma dessas entradas representa 4 MB
 - 4 GB / 1024



62

Gerências de Memória Memória Virtual - Paginação

- No exemplo anterior:
- Suponha que um processo utilize apenas 12 MB do seu espaço de endereços virtuais
 - 4MB da base da memória para código de programa
 - Outros 4 MB para dados
 - 4 MB do topo da memória para pilha
- Portanto:
 - A entrada 0 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa ao código do programa
 - A entrada 1 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa aos dados do processo
 - A entrada 1023 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa à pilha do processo

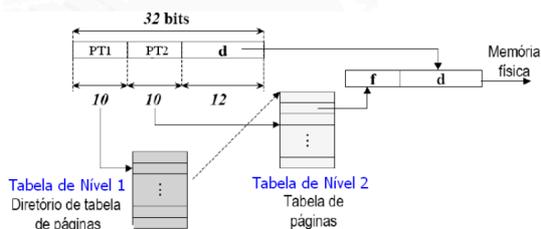
63

Gerências de Memória Memória Virtual - Paginação

- Quando um endereço virtual chega à MMU, a primeira extraí o campo PT1 e o utiliza como índice da tabela de páginas do nível 1
- A entrada da tabela de páginas de nível 1 aponta para a tabela de páginas do nível 2.
- Então PT2 é usado como índice nesta segunda tabela para localizar a entrada correspondente à página virtual
- Esta entrada indicará em qual moldura física encontra-se o endereço a ser acessado
- No exemplo:
 - Suponha que um processo utilize apenas 12 MB do seu espaço de endereços virtuais
 - A entrada 0 da tabela de nível 1 aponta para a tabela de páginas de nível 2 relativa ao código do programa

64

Gerências de Memória Memória Virtual - Paginação



- Considere o endereço virtual 0x00403004(4206596)
 - Qual será o endereço físico correspondente?

65

Gerências de Memória Memória Virtual - Paginação

PT1	PT2	Deslocamento
000000001	000000011	0000 0000 0100

- PT1: Entrada 1 da tabela do 1o nível
 - 2o bloco de 4M (4M a 8M de memória virtual)
- PT2: Entrada 3 da tabela do 2o nível
 - Esta entrada indica em qual moldura encontra-se esta página
 - O endereço físico do primeiro byte dessa moldura é somado ao deslocamento
 - Supondo a página encontra-se na moldura 1 (4k a 8k-1), o endereço físico correspondente será 4096 + 4 = 4100
- OU:

Nº da moldura	Deslocamento
0... 00001	0000 0000 0100

 = 4100₁₀

66

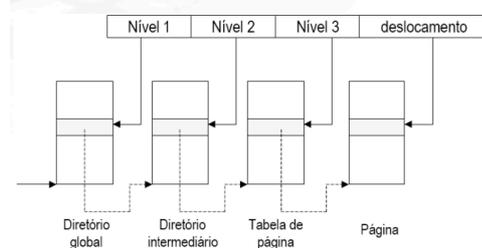
Gerências de Memória Memória Virtual - Paginação

- Para entender as vantagens, considere o exemplo anterior (endereço virtual de 32 bits – página de 4kB)
- Usando tabela de páginas tradicional:
 - 1 tabela de 2^{20} entradas (1 M entradas)
- Usando tabela de páginas em 2 níveis
 - 4 tabelas de 2^{10} entradas cada (1 K entradas)
- Se cada entrada da tab. de páginas ocupa 16 bits
 - primeiro caso: $2^{20} \times 2^4 = 16$ Mbits p/ armazenar a tabela de páginas
 - segundo caso: $4 \times 2^{10} \times 2^4 = 64$ Kbits p/ armazenar a tabela de 2 níveis

{ 67 }

Gerências de Memória Memória Virtual - Paginação

- Paginação de três níveis
 - Típicos de arquiteturas de processadores de 64 bits



{ 68 }

Gerências de Memória Memória Virtual - Paginação

- Espaço de endereçamento virtual pode ser exageradamente grande em máquinas de **64 bits**.
 - Páginas de 4KB
 - 2^{52} entradas na tabela
 - Se cada entrada ocupa 8 B => tabela de ~30.000.000 GB
- O armazenamento da tabela torna-se viável se a mesma for **invertida**, isto é, ter o tamanho da quantidade de molduras (memória real) e não da quantidade de páginas (memória virtual)
- Se memória real é de 256 Mbytes , e páginas de 4 KB:
 - Tem-se 65536 entradas

{ 69 }

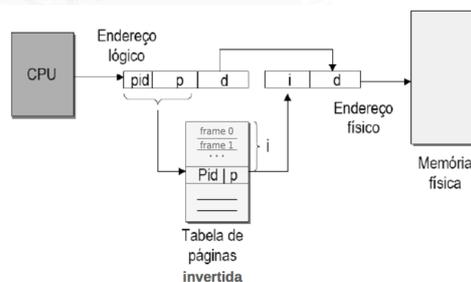
Gerências de Memória Memória Virtual - Paginação

Tabelas de Páginas Invertidas

- Uma entrada por moldura de memória real
- Cada entrada na tabela informa
 - Par: (PID, # página virtual) alocado naquela moldura
- Entretanto
 - Tradução de virtual/físico mais complicada
 - Quando o processo *n* endereça a página *p*
 - *p* não serve de índice da tabela
 - Toda a tabela deve ser pesquisada em busca de uma entrada (*p*,*n*)
- Solução muito lenta
 - A busca é feita para toda referência à memória

{ 70 }

Gerências de Memória Memória Virtual - Paginação



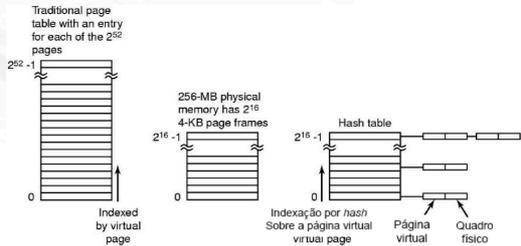
{ 71 }

Gerências de Memória Memória Virtual - Paginação

- Aceleração pode ser obtida
 - TLB para páginas mais referenciadas
 - Indexar a tabela por hash
 - Uma função hash que recebe o número da página e retorna um entre N valores possíveis, onde N é a quantidade de molduras (memória instalada).
 - Páginas com mesmo hash serão encadeadas em uma lista
 - Cada entrada da tabela armazena um par (página/quadro)

{ 72 }

Gerências de Memória Memória Virtual - Paginação



Comparação de uma *page table* tradicional com uma *page table invertida*

73

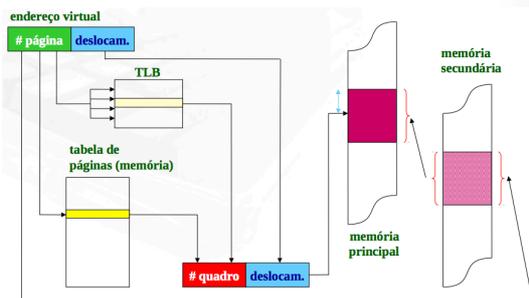
Gerências de Memória Memória Virtual - Paginação

TLB – *Translation Lookaside Buffer*

- Como diminuir o número de referências à MP introduzido pelo mecanismo de paginação?
- Os programas tendem a fazer um grande número de referências a um mesmo pequeno conjunto de páginas virtuais
 - Princípio da localidade temporal e espacial
- Solução: equipar a MMU com uma TLB
 - Também chamada de Memória Associativa
 - Dispositivo de hardware implementado com um reduzido número de entradas
- Contém algumas entradas (linhas) da tabela de páginas do processo em execução

74

Gerências de Memória Memória Virtual - Paginação



75

Gerência de Memória Memória Virtual - Paginação

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Exemplo de TLB

- Loop acessando pag. 19, 20, 21
- Dados principais: pag. 129, 130, 141
- Pilha: 860, 861

Memória associativa (TLB) - De 64 a 4096 entradas

76

Gerência de Memória Memória Virtual - Paginação

HIT Ratio (Taxa de Sucesso)

- Razão de referências à memória que podem ser satisfeitas a partir da TLB
- \uparrow Hit Ratio \Rightarrow \uparrow performance
- Tempo de acesso com HIT (sucesso) à memória via TLB

$$T_{\text{HIT}} = T_{\text{TLB}} + T_{\text{MEM}}$$
- Tempo de acesso com MISS (insucesso) à memória via TLB

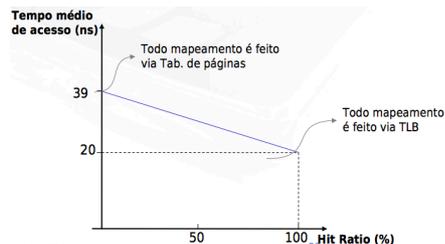
$$T_{\text{MISS}} = T_{\text{TLB}} + T_{\text{MEM}} + T_{\text{MEM}}$$
- Tempo médio de acesso = $hr \cdot T_{\text{HIT}} + (1-hr) \cdot T_{\text{MISS}}$
 - hr é o Hit Ratio

77

Gerência de Memória Memória Virtual - Paginação

HIT Ratio (Taxa de Sucesso)

- Suponha: $T_{\text{HIT}} = 20\text{ns}$; $T_{\text{MISS}} = 39\text{ns}$; H.R. = 90%
- Tempo médio de acesso = $0,9 \times 20 + 0,1 \times 39 = 21,9\text{ns}$



78

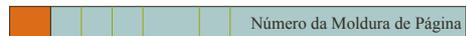
Gerência de Memória Memória Virtual - Paginação

- Algumas questões que surgem com relação à Paginação:
 - Onde armazenar a tabela de páginas?
 - Qual a estrutura de uma entrada na tabela de páginas?

{ 79 }

Gerenciamento de Memória Memória Virtual - Paginação

- Estrutura de uma tabela de páginas: 32 bits (mais comum)

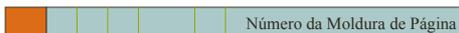


Identifica a página real;
Campo mais importante;

{ 80 }

Gerenciamento de Memória Memória Virtual - Paginação

- Estrutura de uma tabela de páginas: 32 bits (mais comum)

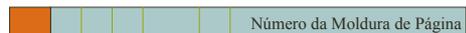


Bit de Residência:
Se valor igual 1, então entrada válida para uso;
Se valor igual 0, então entrada inválida, pois página virtual correspondente não está na memória;

{ 81 }

Gerenciamento de Memória Memória Virtual - Paginação

- Estrutura de uma tabela de páginas: 32 bits (mais comum)



Bits de Proteção:
Indicam tipos de acessos permitidos:
1 bit → 0 – leitura/escrita
 1 – leitura
3 bits → 0 – Leitura
 1 – Escrita
 2 – Execução

{ 82 }

Gerenciamento de Memória Memória Virtual - Paginação

- Estrutura de uma tabela de páginas: 32 bits (mais comum)



Bit de Modificação (Bit M):
Controla o uso da página;
Se valor igual a 1, página foi escrita;
 página é copiada para o disco
Se valor igual a 0, página não foi modificada;
 página não é copiada para o disco;

{ 83 }

Gerenciamento de Memória Memória Virtual - Paginação

- Estrutura de uma tabela de páginas: 32 bits (mais comum)

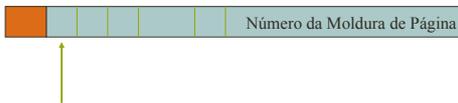


Bit de Referência (Bit R):
Controla o uso da página;
Auxilia o SO na escolha da página que deve deixar a MP (RAM);
Se valor igual a 1, página foi referenciada (leitura/escrita);
Se valor igual a 0, página não referenciada;

{ 84 }

Gerenciamento de Memória Memória Virtual - Paginação

- Estrutura de uma tabela de páginas: 32 bits (mais comum)



Bit de Cache:
Necessário quando os dispositivos de entrada/saída são mapeados na memória e não em um endereçamento específico de E/S;

85

Gerência de Memória Memória Virtual - Paginação

- Algumas questões que surgem com relação à Paginação:
 - Onde armazenar a tabela de páginas?
 - Qual a estrutura de uma entrada na tabela de páginas?
 - Quantas páginas reais serão alocadas a um processo?

86

Gerenciamento de Memória Paginação - Alocação de Páginas

Quantas páginas reais serão alocadas a um processo?

- Duas estratégias:
 - **Alocação fixa ou estática:** cada processo tem um número máximo de páginas reais, definido quando o processo é criado;
 - O limite pode ser igual para todos os processos;
 - **Vantagem:** simplicidade;
 - **Desvantagens:** (i) número muito pequeno de páginas reais pode causar muita paginação; (ii) número muito grande de páginas reais causa desperdício de memória principal.

87

Gerenciamento de Memória Paginação - Alocação de Páginas

Quantas páginas reais serão alocadas a um processo?

- **Alocação variável ou dinâmica:** número máximo de páginas reais alocadas ao processo varia durante sua execução;
 - **Vantagem:** (i) processos com elevada taxa de paginação podem ter seu limite de páginas reais ampliado; (ii) processos com baixa taxa de paginação podem ter seu limite de páginas reais reduzido;
 - **Desvantagem:** monitoramento constante.

88

Gerência de Memória Memória Virtual - Paginação

- Algumas questões que surgem com relação à Paginação:
 - Onde armazenar a tabela de páginas?
 - Qual a estrutura de uma entrada na tabela de páginas?
 - Quantas páginas reais serão alocadas a um processo?
 - **Quando uma página deve ser carregada para a memória?**

89

Gerenciamento de Memória Paginação - Busca de Página

Políticas de busca determinam quando uma página deve ser carregada para a memória.

- **Paginação simples:**
 - Todas as páginas virtuais do processo são carregadas para a memória principal;
 - Assim, sempre todas as páginas são válidas.
- **Paginação por demanda (Demand Paging):**
 - Apenas as páginas efetivamente acessadas pelo processo são carregadas na memória principal;
 - Quais páginas virtuais foram carregadas → Bit de controle (bit de residência);
 - Página inválida.
- **Paginação antecipada (Anticipatory Paging):**
 - Carrega para a memória principal, além da página referenciada, outras páginas que podem ou não ser necessárias para o processo.

90

Gerência de Memória Memória Virtual - Paginação

- Algumas questões que surgem com relação à Paginação:
 - Onde armazenar a tabela de páginas?
 - Qual a estrutura de uma entrada na tabela de páginas?
 - Quantas páginas reais serão alocadas a um processo?
 - Quando uma página deve ser carregada para a memória?
 - **Como trazer uma página para a memória?**

91

Gerenciamento de Memória Paginação - Busca de Página

- Página inválida: MMU gera uma interrupção de proteção e aciona o sistema operacional;
 - Se a página está fora do espaço de endereçamento do processo, o processo é abortado;
 - Se a página ainda não foi carregada na memória principal, ocorre uma **falta de página** (*page fault*).

92

Gerenciamento de Memória Paginação - Busca de Página

- **Falta de Página**:
 - Processo é suspenso e seu descritor é inserido em uma **fila especial** – fila dos processos esperando uma página virtual;
 - Uma página real livre deve ser alocada;
 - A página virtual acessada deve ser localizada no disco;
 - Operação de leitura de disco, indicando o endereço da página virtual no disco e o endereço da página real alocada.

93

Gerenciamento de Memória Paginação - Busca de Página

- Após a leitura do disco:
 - Tabela de páginas do processo é corrigida para indicar que a página virtual agora está válida e está na página real alocada;
 - *Pager*: carrega páginas específicas de um processo do disco para a memória principal;
 - O descritor do processo é retirado da **fila especial** e colocado na fila do processador.

94

Gerência de Memória Memória Virtual - Paginação

- Algumas questões que surgem com relação à Paginação:
 - Onde armazenar a tabela de páginas?
 - Qual a estrutura de uma entrada na tabela de páginas?
 - Quantas páginas reais serão alocadas a um processo?
 - Quando uma página deve ser carregada para a memória?
 - Como trazer uma página para a memória?
 - **Como liberar espaço na memória?**

95

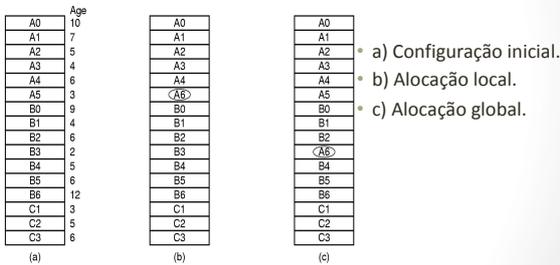
Gerenciamento de Memória Paginação - Troca de Páginas

A liberação é feita por meio da **troca de página**.

- **Política de Substituição Local**: páginas dos próprios processos são utilizadas na troca;
 - Dificuldade: definir quantas páginas cada processo pode utilizar;
- **Política de Substituição Global**: páginas de todos os processos são utilizadas na troca;
 - Problema: processos com menor prioridade podem ter um número muito reduzido de páginas, e com isso, acontecem muitas faltas de páginas;

96

Gerenciamento de Memória Paginação - Troca de Páginas



97

Gerenciamento de Memória Troca de Páginas

- Algoritmos de substituição local alocam uma fração fixa de memória para cada processo; enquanto que algoritmos de substituição global alocam molduras de páginas entre os processos em execução, variando o número de páginas no tempo;

98

Gerenciamento de Memória Troca de Páginas



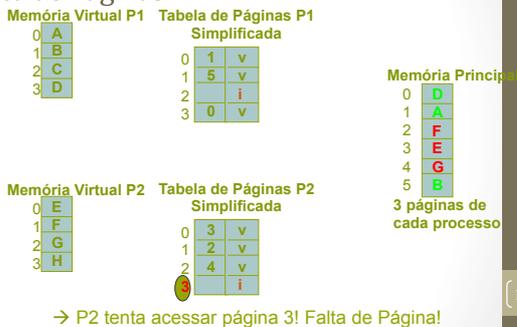
99

Gerenciamento de Memória Troca de Páginas

- Se todas as páginas estiverem ocupadas, uma página deve ser retirada: página vítima;
- Exemplo:
 - Dois processos P1 e P2, cada um com 4 páginas virtuais;
 - Memória principal com 6 páginas.

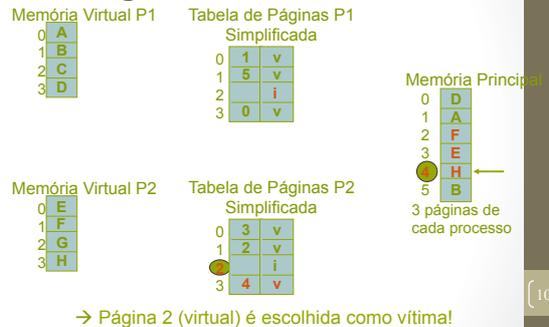
100

Gerenciamento de Memória Troca de Páginas



101

Gerenciamento de Memória Troca de Páginas



102

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmos:
 - Ótimo;
 - NRU;
 - FIFO;
 - Segunda Chance;
 - Relógio;
 - LRU;
 - *Working set*;
 - *WSclock*.

{ 103 }

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo Ótimo:
 - Retira da memória a página que tem menos chance de ser referenciada;
 - Praticamente impossível de se saber;
 - Impraticável;
 - Usado em simulações para comparação com outros algoritmos.

{ 104 }

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Not Recently Used Page Replacement* (NRU) → troca as páginas não utilizadas recentemente:
 - 02 bits associados a cada página → R e M
 - Classe 0 → não referenciada, não modificada;
 - Classe 1 → não referenciada, modificada;
 - Classe 2 → referenciada, não modificada;
 - Classe 3 → referenciada, modificada;
 - R e M são atualizados a cada referência à memória.
 - Uma vez que a página foi lida, o valor do bit R será sempre igual a 1 até que o SO reinicialize-o

{ 105 }

Gerenciamento de Memória Troca de Páginas - Paginação

- NRU:
 - Periodicamente, o bit R é limpo para diferenciar as páginas que não foram referenciadas recentemente;
 - A cada *tick* do relógio ou interrupção de relógio;
 - Classe 3 → Classe 1;
 - Vantagens: fácil de entender, eficiente para implementar e fornece bom desempenho.

{ 106 }

Gerenciamento de Memória Troca de Páginas - Paginação

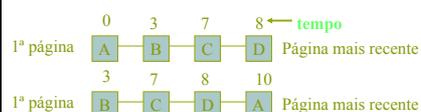
- Algoritmo *First-in First-out Page Replacement* (FIFO)
 - SO mantém uma listas das páginas correntes na memória;
 - A página no início da lista é a mais antiga e a página no final da lista é a mais nova;
 - Simples, mas pode ser ineficiente, pois uma página que está em uso constante pode ser retirada;
 - Pouco utilizado.

{ 107 }

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo da Segunda Chance
 - FIFO + bit R;
 - Página mais velha é candidata em potencial;

Se o bit R=0, então página é retirada da memória, senão, R=0 e se dá uma nova chance à página colocando-a no final da lista.



Se página A com R=1; e falta de página em tempo 10; Então R=0 e página A vai para final da lista;

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo do Relógio
 - Lista circular com ponteiro apontando para a página mais antiga
 - Algoritmo se repete até encontrar R=0;

Se R=0

- troca de página
- desloca o ponteiro

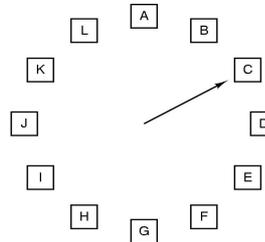
Se R=1

- R = 0
- desloca o ponteiro
- continua busca

109

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo do Relógio



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:
R = 0: Evict the page
R = 1: Clear R and advance hand

110

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo Least Recently Used Page Replacement (LRU)
 - Troca a página menos referenciada/modificada recentemente;
 - Alto custo
 - Lista encadeada com as páginas que estão na memória, com as mais recentemente utilizadas no início e as menos utilizadas no final;
 - A lista deve ser atualizada a cada referência da memória.

111

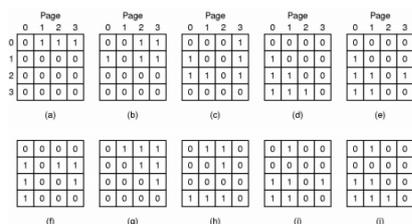
Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo Least Recently Used Page Replacement (LRU)
 - Pode ser implementado tanto por hardware quanto por software:
 - Hardware:** MMU deve suportar a implementação LRU;
 - Contador em hardware (64 bits) - conta instruções executadas;
 - Após cada referência à memória, o valor do contador é armazenado na entrada da tabela de páginas referente à página acessada;
 - Quando ocorre falta de página, o SO examina todos os contadores e escolhe a página que tem menor valor;
 - Tabela de páginas armazena o valor desse contador para saber quantas vezes a página foi usada.
 - Software:** duas maneiras
 - NFU (Not frequently used);
 - Aging (Envelhecimento);

112

Gerenciamento de Memória Troca de Páginas - Paginação

- LRU - Hardware
 - Matriz nxn bits
 - Quando se faz uma referência à página k ->
 - Todos os bits da linha k recebem valor 1
 - Todos os bits da coluna k recebem o valor 0



113

Gerenciamento de Memória Troca de Páginas - Paginação

- Software: NRU
 - Para cada página existe um contador -> iniciado com zero e incrementado a cada referência à página;
 - Página com menor valor do contador é candidata a troca;
 - Como esse algoritmo não se esquece de nada
 - Problema: pode retirar páginas que estão sendo referenciadas com frequência;
 - Compiler com vários passos: passo 1 tem mais tempo de execução que os outros passos -> páginas do passo 1 terão mais referências armazenadas;

114

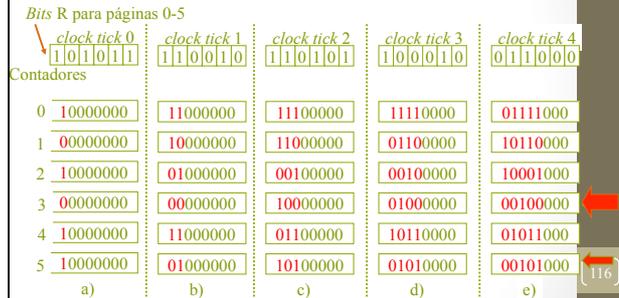
Gerenciamento de Memória Troca de Páginas - Paginação

- Software: Algoritmo *aging*
 - Modificação do NRU, resolvendo o problema descrito anteriormente;
 - Além de saber **quantas vezes** a página foi referenciada, também controla **quando** ela foi referenciada;
 - Geralmente, 8 bits são suficientes para o controle se as interrupções de relógio (*clock ticks*) ocorrem a cada 20ms (10^{-3});

115

Gerenciamento de Memória Troca de Páginas - Paginação

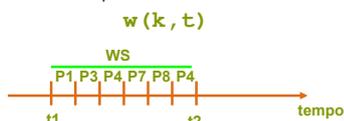
- Algoritmo *aging*



116

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Working Set (WS)*:
 - Paginação por demanda → páginas são carregadas na memória somente quando são necessárias;
 - Pré-paginação → *Working set*
 - Conjunto de páginas que um processo está efetivamente utilizando (referenciando) em um determinado tempo t .



117

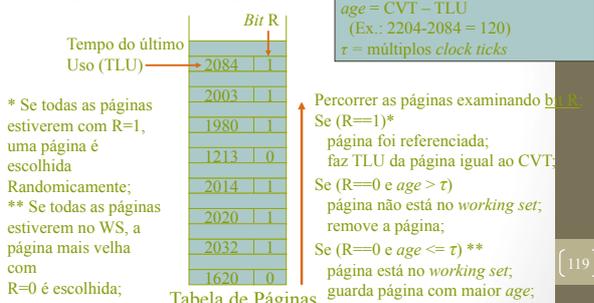
Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Working Set (WS)*:
 - Objetivo principal: reduzir a falta de páginas
 - Um processo só é executado quando todas as páginas necessárias no tempo t estão carregadas na memória;
 - SO gerencia quais páginas estão no *Working Set*;
 - Para simplificar → o *working set* pode ser visto como o conjunto de páginas que o processo referenciou durante os últimos τ segundos de tempo;
 - Utiliza *bit R* e o tempo de relógio (tempo virtual) da última vez que a página foi referenciada;

118

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Working Set*:



119

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock*:
 - *Clock + Working Set*;
 - Lista circular de páginas formando um anel a cada página carregada na memória;
 - Utiliza *bit R* e o tempo da última vez que a página foi referenciada;
 - *Bit M* utilizado para agendar escrita em disco.

120

Gerenciamento de Memória Troca de Páginas - Paginação

Tempo virtual atual: 2204

- Algoritmo *WSClock*:

a) $R=0$ e $age > t$
Tempo do último uso

b) $R=1$
 $R=0$ e ponteiro avança

(121)

Gerenciamento de Memória Troca de Páginas - Paginação

Tempo virtual atual: 2204

- Algoritmo *WSClock*:

c) $R=0$ e $age > t$
Tempo do último uso

d) $R=0$ e $age > t$
 $M=0$ (não agenda escrita) → troca

Nova página

(122)

Gerenciamento de Memória Troca de Páginas - Paginação

Tempo virtual atual: 2204

- Algoritmo *WSClock*:

e) $R=0$ e $age > t$
 $M=1$ (agenda escrita e continua processando)

Nova página

(123)

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock*:
 - Se todas estiverem com $M=1$; então escreve página atual no disco, e troca a página;
 - Melhor desempenho → menos acessos ao disco.

(124)

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmos de substituição local:
 - Working Set*;
 - WSClock*;
- Algoritmos de substituição local/global:
 - Ótimo;
 - NRU;
 - FIFO;
 - Segunda Chance;
 - LRU;
 - Relógio.

(125)

Gerenciamento de Memória Troca de Páginas - Paginação

Resumo dos Algoritmos de Substituição de Páginas

Algoritmo	Comentário
Ótimo	Não implementável, mas útil como um padrão de desempenho
NRU	Muito rudimentar
FIFO	Pode descartar páginas importantes
Segunda Chance	Algoritmo FIFO bastante melhorado
Relógio	Realista
MRU	Excelente algoritmo porém difícil de ser implementado de maneira exata
NFU	Aproximação bastante rudimentar do MRU
Envelhecimento	Algoritmo bastante eficiente que se aproxima bem do MRU
Conjunto de trabalho	Implementação cara
WSClock	Algoritmo bom e eficiente

(126)

Gerenciamento de Memória Implementação da Paginação

Até agora – somente como uma página é selecionada para remoção. Mas onde essa página descartada da memória é colocada?

• Memória Secundária – Disco

- A área de troca (*swap area*) é gerenciada como uma lista de espaços disponíveis;
- O endereço da área de troca de cada processo é mantido na tabela de processos;
 - Cálculo do endereço: MMU;
- Possibilidade A - Assim que o processo é criado, ele é copiado todo para sua área de troca no disco, sendo carregado para memória quando necessário;
 - Área de troca diferente para dados, pilha e programa, pois área de dados pode crescer e a área de pilha crescerá certamente.

127

Gerenciamento de Memória Implementação da Paginação

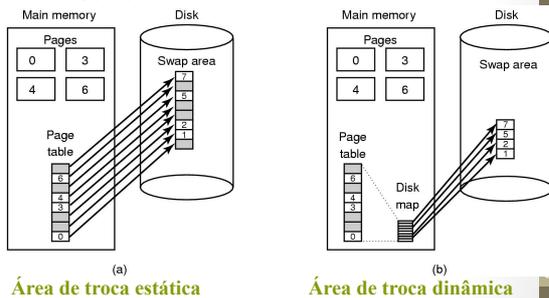
• Memória Secundária – Disco

- Possibilidade B - Nada é alocado antecipadamente, espaço é alocado em disco quando a página for enviada para lá. Assim, processo na memória RAM não fica “amarrado” a uma área específica.

128

Gerenciamento de Memória Implementação da Paginação

Como fica o disco – memória secundária



29

Gerenciamento de Memória Memória Virtual - Segmentação

- Segmentação: Visão do programador/compilador
 - Tabelas de segmentos com n linhas, cada qual apontando para um segmento de memória;
 - Vários espaços de endereçamento;
 - Endereço real \rightarrow base + deslocamento;
 - Alocação de segmentos segue os algoritmos já estudados:
 - *FIRST-FIT*;
 - *BEST-FIT*;
 - *NEXT-FIT*;
 - *WORST-FIT*;
 - *QUICK-FIT*.

130

Gerenciamento de Memória Memória Virtual - Segmentação

- Segmentação:
 - Facilita proteção dos dados;
 - Facilita compartilhamento de procedimentos e dados entre processos;
 - MMU também é utilizada para mapeamento entre os endereços lógicos e físicos;
 - Tabela de segmentos informa qual o endereço da memória física do segmento e seu tamanho.

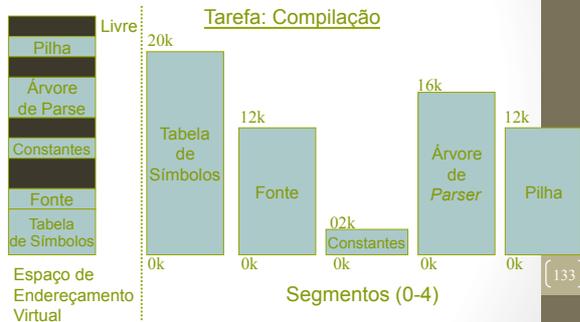
131

Gerenciamento de Memória Memória Virtual - Segmentação

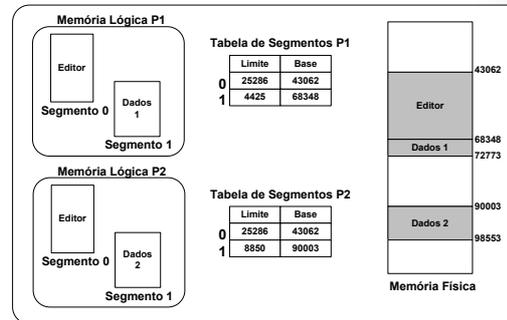
- Segmentação:
 - Problemas encontrados \rightarrow embora haja espaço na memória, não há espaço contínuo:
 - Política de realocação: um ou mais segmentos são realocados para abrir espaço contínuo;
 - Política de compactação: todos os espaços são compactados;
 - Política de bloqueio: fila de espera;
 - Política de troca: substituição de segmentos;
 - Sem fragmentação interna, com fragmentação externa.

132

Gerenciamento de Memória Memória Virtual - Segmentação



Gerenciamento de Memória Memória Virtual - Segmentação

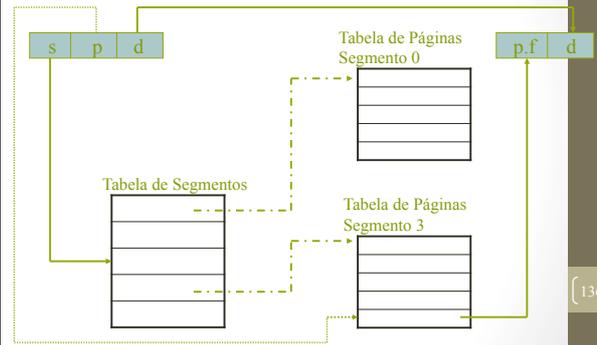


Gerenciamento de Memória Segmentação-Paginada

- Espaço lógico é formado por segmentos
 - Cada segmento é dividido em páginas lógicas;
 - Cada segmento possui uma tabela de páginas → mapear o endereço de página lógica do segmento em endereço de página física;
 - No endereçamento, a tabela de segmentos indica, para cada segmento, onde sua respectiva tabela de páginas está;

135

Gerenciamento de Memória Segmentação-Paginada



Gerenciamento de Memória Thrashing

- **Thrashing** (paginação excessiva)
 - Associado com o problema de definição do número de páginas/segmentos → troca de páginas/segmentos é uma tarefa cara e lenta;
 - Se o processo tiver um número de páginas muito reduzido, ele pode ficar muito tempo esperando pelo atendimento de uma falta de página → muitos processos bloqueados.

137

Gerenciamento de Memória Thrashing

- **Evitar o problema (paginação):**
 - Taxa máxima aceitável de troca de páginas;
 - Suspende alguns processos, liberando páginas físicas (*swapping*);
 - Risco de aumentar o tempo de resposta dos processos.
 - Determinar periodicamente o número de processos em execução e alocar para cada um mesmo número de páginas;
 - **Problema:** processos grandes teriam o mesmo número de páginas de processos pequenos, causando paginação excessiva.

138

Gerenciamento de Memória Thrashing

- Possível solução: Número de páginas proporcional ao tamanho do processo → alocação dinâmica durante a execução dos processos;
 - PFF (*Page Fault Frequency*): algoritmo informa quando aumentar ou diminuir a alocação de páginas de um processo, controlando o tamanho do conjunto de alocação;

(139)

Gerenciamento de Memória Memória Virtual

Consideração	Paginação	Segmentação
Programador deve saber da técnica?	Não	Sim
Espaços de endereçamento existentes	1	Vários
Espaço total de endereço pode exceder memória física?	Sim	Sim
É possível distinguir procedimento de dados e protegê-los?	Não	Sim

(140)

Gerenciamento de Memória Memória Virtual

Consideração	Paginação	Segmentação
Tabelas de tamanho variável podem ser acomodadas sem problemas?	Não	Sim
Compartilhamento de procedimentos entre usuário é facilitado?	Não	Sim
Por que?	Para obter espaço de endereçamento maior sem aumentar memória física	Para permitir que programas e dados possam ser divididos em espaços de endereçamento logicamente independentes; compartilhamento e proteção

(141)