Sistemas Operacionais

Gerenciamento de Memória

Prof. Arliones Hoeller

arliones.hoeller@ifsc.edu.br

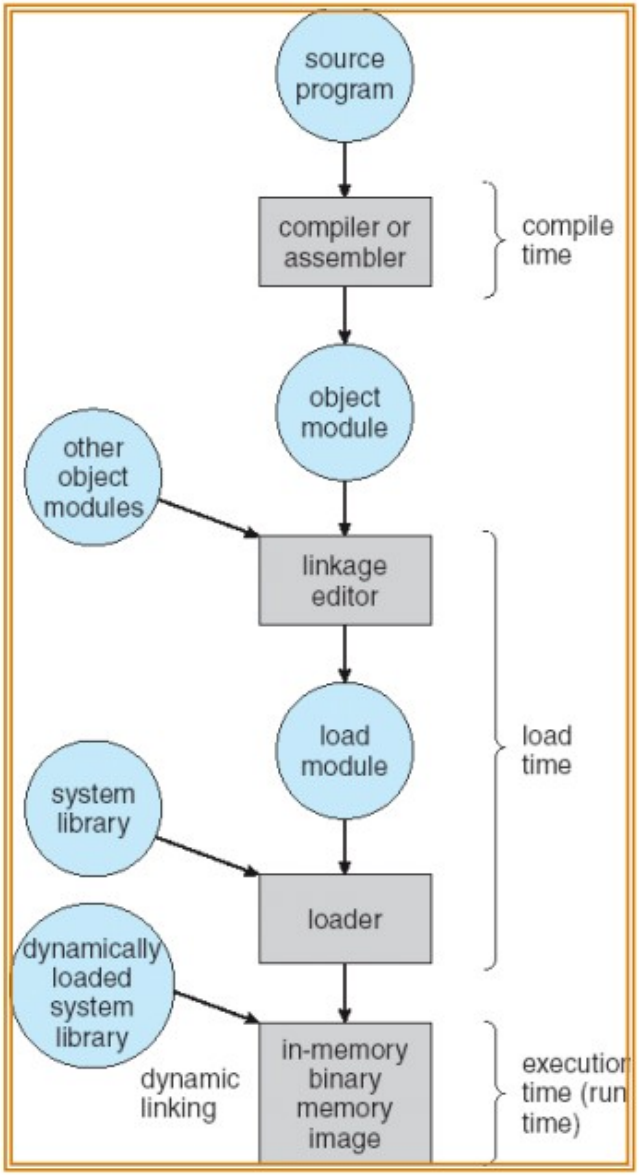
Abril de 2014

baseado no material do Prof. Fröhlich em
http://www.lisha.ufsc.br/~guto

Gerenciamento de Memória

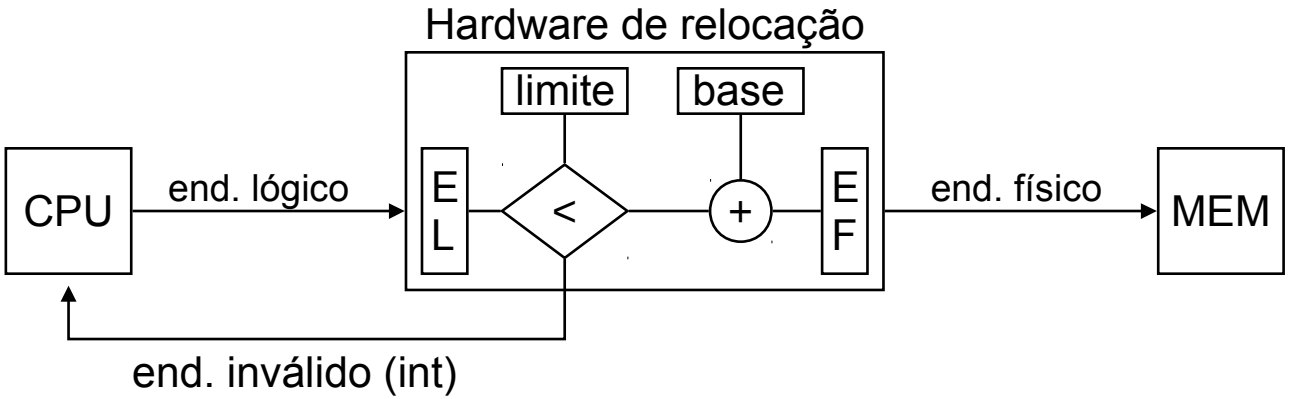
- Processador busca instruções na memória principal
  - Programas precisam ser carregados na memória antes de executar
- Endereços apontados por programas (ex: variáveis) precisam ser vinculados à memória
  - Na compilação: endereços absolutos
  - No carregamento: código relocável
  - Durante execução: hardware de relocação
- Programas maiores que a memória disponível
  - Sobreposições realizadas durante execução do programa
  - Pode envolver suporte de SO
- Programas comumente replicados
  - Podem ser organizados em bibliotecas compartilhadas

Carregando Programas

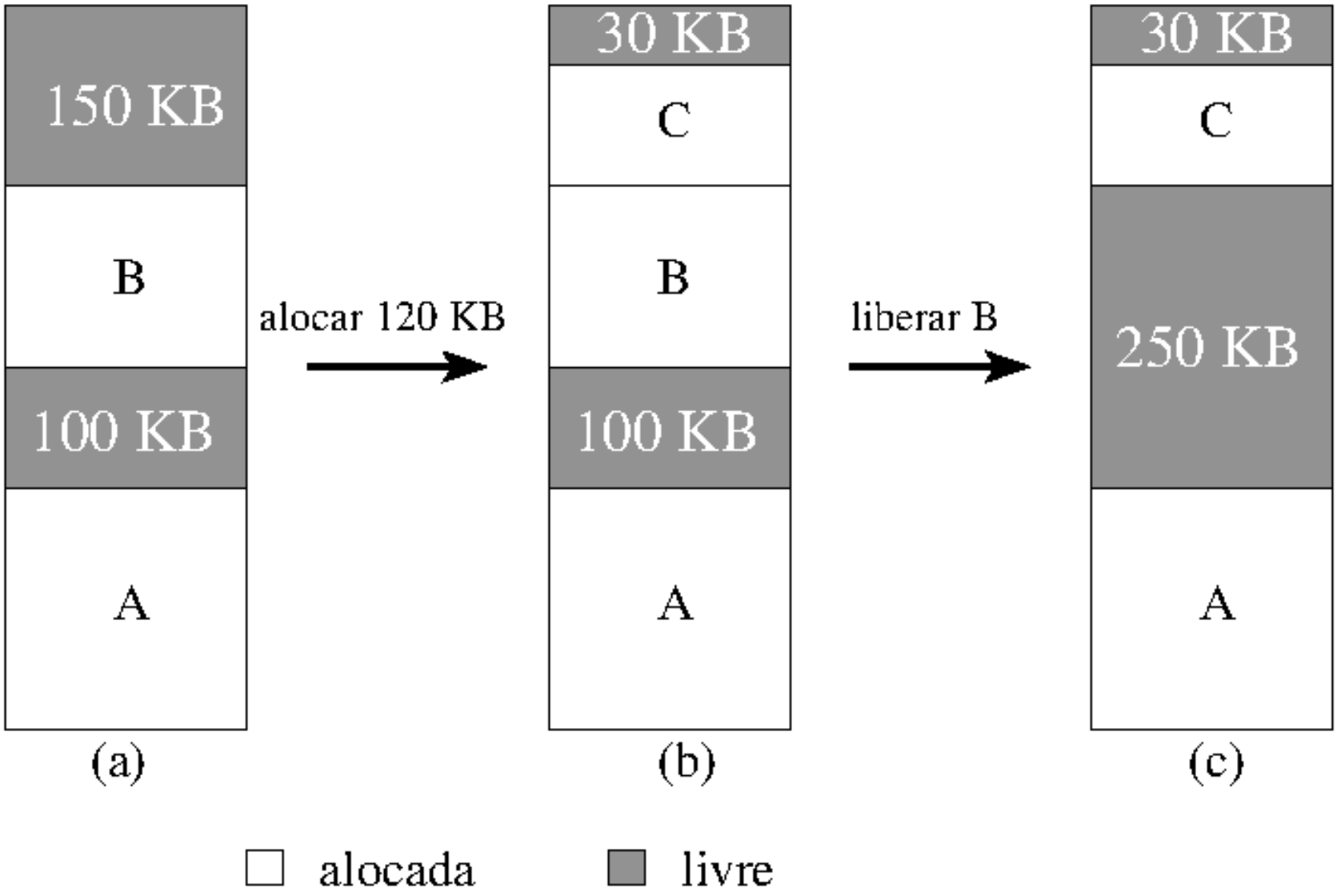


Alocação de Memória para um único processo

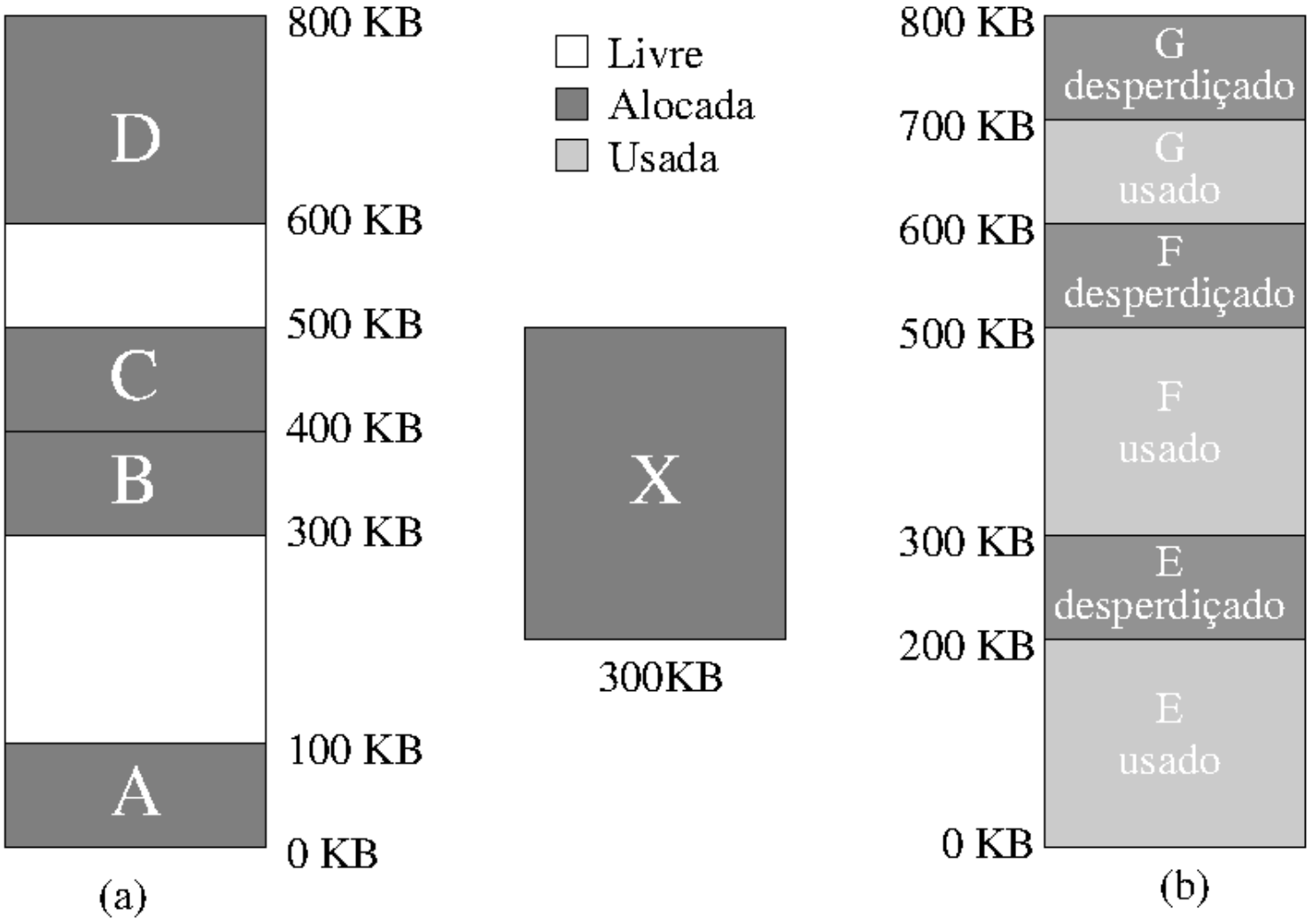
- Sem suporte de SO
  - Sistemas simples, dedicados (ex: embarcados)
  - Sem gerenciador de memória
- Com suporte de SO
  - Memória do SO é protegida por um *registrador-base*
  - Processos de usuário são carregados após SO
- Relocação dinâmica com suporte de hardware
  - Compilador e CPU geram endereços de memória *lógicos*
  - Hardware de relocação somam **endereço lógico** ao *endereço-base*, gerando um **endereço de memória física**



Alocação de memória



Fragmentação



Externa

Interna

Alocação de memória a múltiplos processos sem suporte de hardware

- Lista de blocos de memória livres
  - First-fit: aloca o **primeiro** bloco grande o suficiente para armazenar o processo
  - Best-fit: aloca o **menor** bloco que é grande o suficiente para armazenar o processo
  - Worst-fit: aloca **maior** bloco disponível
- Fragmentação externa
  - Grande volume de blocos pequenos
  - Há memória livre suficiente para satisfazer uma requisição mas a memória livre não é contígua

$$Fragmentação\ Externa\ de\ Memória = 1 - \frac{Maior\ Bloco\ de\ Memória\ Livre}{Total\ de\ Memória\ Livre}$$

Alocação de memória a múltiplos processos sem suporte de hardware

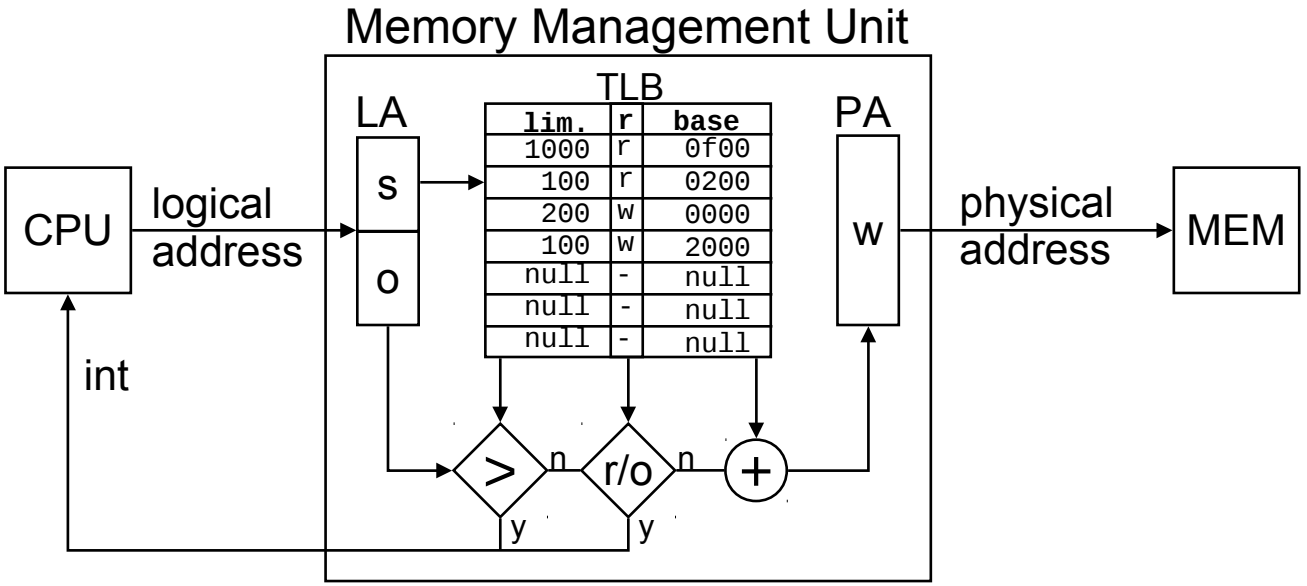
- Proteção
  - Através de registradores *base* e *limite*
- Compactação
  - Relocação dinâmica de processos para agrupar blocos livres de memória
  - Suporte a relocação
    - Apenas endereços relativos (implícitamente relocáveis)
    - Re-link pelo carregador de aplicação do SO (caro)
    - Suporte externo a relocação

Alocação de memória a múltiplos processos com suporte de hardware

- Separação entre os conceitos de **espaço de endereçamento** e **memória**
  - Compiladores, processadores e processos operam em um espaço de endereçamento que é mapeado à memória por uma **MMU**
- Memory Management Unit (MMU)
  - Hardware que traduz endereços lógicos em físicos
  - Logo: mapeia o espaço de endereçamento de cada processo à memória
- Estratégias típicas
  - Paginação
  - Segmentação
  - Paginação segmentada

Segmentação

- Memória organizada em segmentos e alocada em palavras
- Endereços bi-dimensionais: (segmento, offset)
- Espaços de endereçamento de processos
  - Organizados em segmentos
  - Mapeados à memória física por tabelas de segmentos com **base** e **limite** para cada segmento



Segmentação e Fragmenção

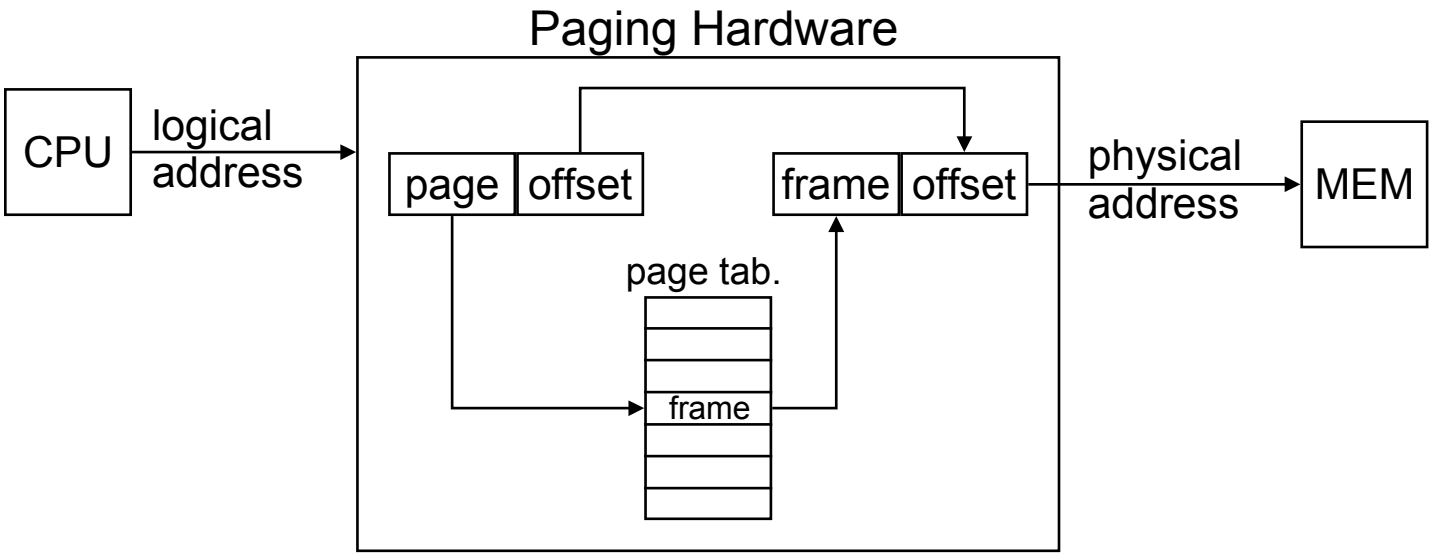
- Fragmentação externa
  - Um segmento por processos -> *-fit
  - Segmentos de tamanho fixo -> paginação
  - Segmentos de tamanho de palavras -> tabelas de segmentos grandes e dobra tempo de acesso à memória
  - Um segmento por objeto
    - Proposta da Intel
    - Não implementado por compiladores comuns
- Sem fragmentação interna
  - Limite pode ser ajustado para utilizar fração de segmento

Implementação de Segmentação

- Tabelas de Segmentos
  - Registradores: limitado a poucos segmentos (poucos espaços de endereçamento ou segmentos grandes)
  - Memória: lento (dobra tempo de acesso à memória)
  - Translation Look-aside Buffer (TLB)
    - Cache de traduções de páginas
    - Rápido e caro (memória completamente associativa)
    - Bom desempenho se taxa de acerto é alto (política de troca)
- Compartilhamento de segmentos
  - Tabelas de segmentos de processos distintos podem referenciar segmentos em comum
- Proteção
  - Segmentos são marcados com bits de permissão que são verificados pela MMU
  - O limite dos segmentos também é verificado pela MMU

Paginação

- Memória organizada e alocada em **quadros** (frames)
- Espaços de endereçamento dos processos
  - Organizados em **páginas**
  - Mapeados a **quadros** (memória física) através de **tabelas de páginas**



Paginação e Fragmentação

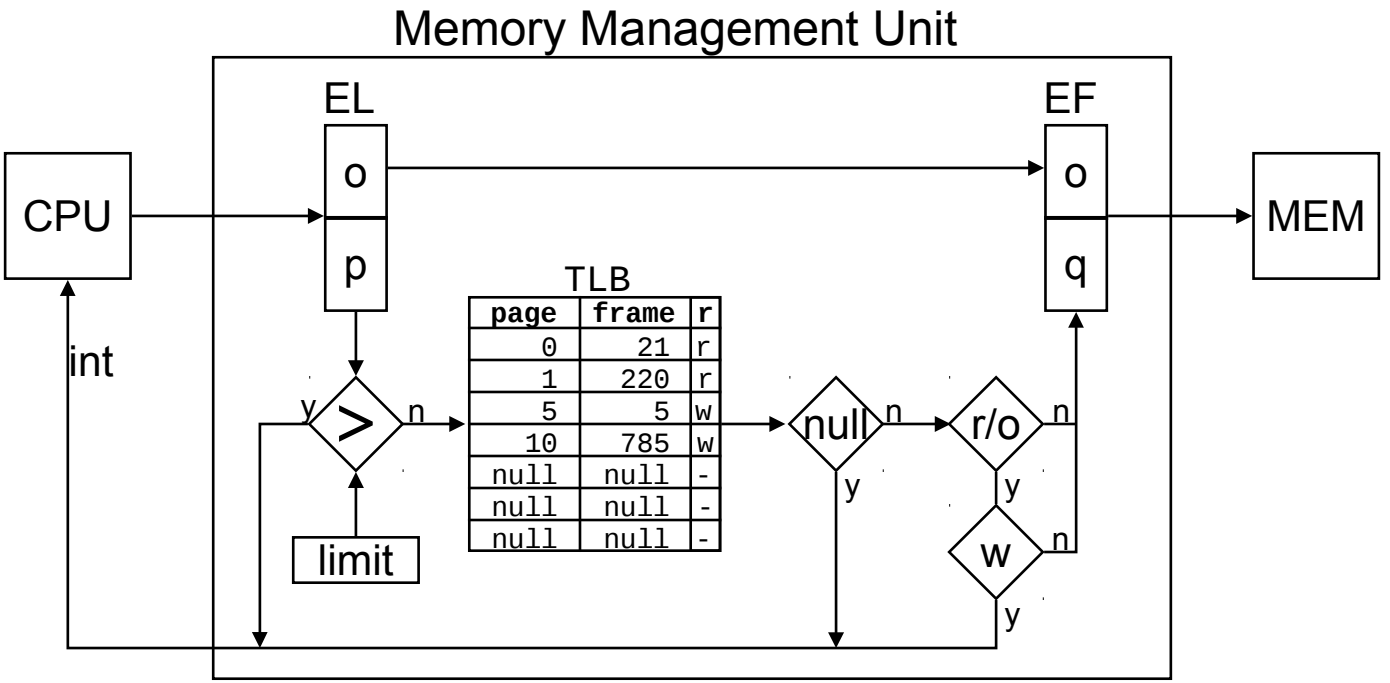
- Não há fragmentação externa
- Fragmentação interna
  - Fração não utilizada de uma página que não pode ser alocado a outro processo
- Fragmentação interna X tamanho de página
  - Página pequena -> menos fragmentação -> mais memória para armazenar tabelas de página
  - Exemplo
    - Requisição de alocação de 1 GB
    - Páginas de 4 KB -> 262.144 páginas
    - Páginas de 4 MB → 256 páginas

Implementação de Paginação

- Tabelas de Páginas
  - Em registradores: limitado a poucas páginas
    - Espaços de endereçamento pequenos ou páginas grandes
  - Em memória: lento (dobra o tempo de acesso à memória)
  - Translation Look-aside Buffer (TLB)
    - Cache de traduções de páginas
    - Rápido e caro (memória completamente associativa)
    - Bom desempenho se taxa de acerto é alto (política de troca)
- Compartilhamento de páginas
  - Tabelas de páginas de processos diferentes podem apontar para quadros em comum
  - Explorado por bibliotecas de código compartilhado
- Proteção
  - Páginas são marcadas com bits de permissão verificados pela MMU
  - Registro de limite para reduzir tamanho da tabela de página

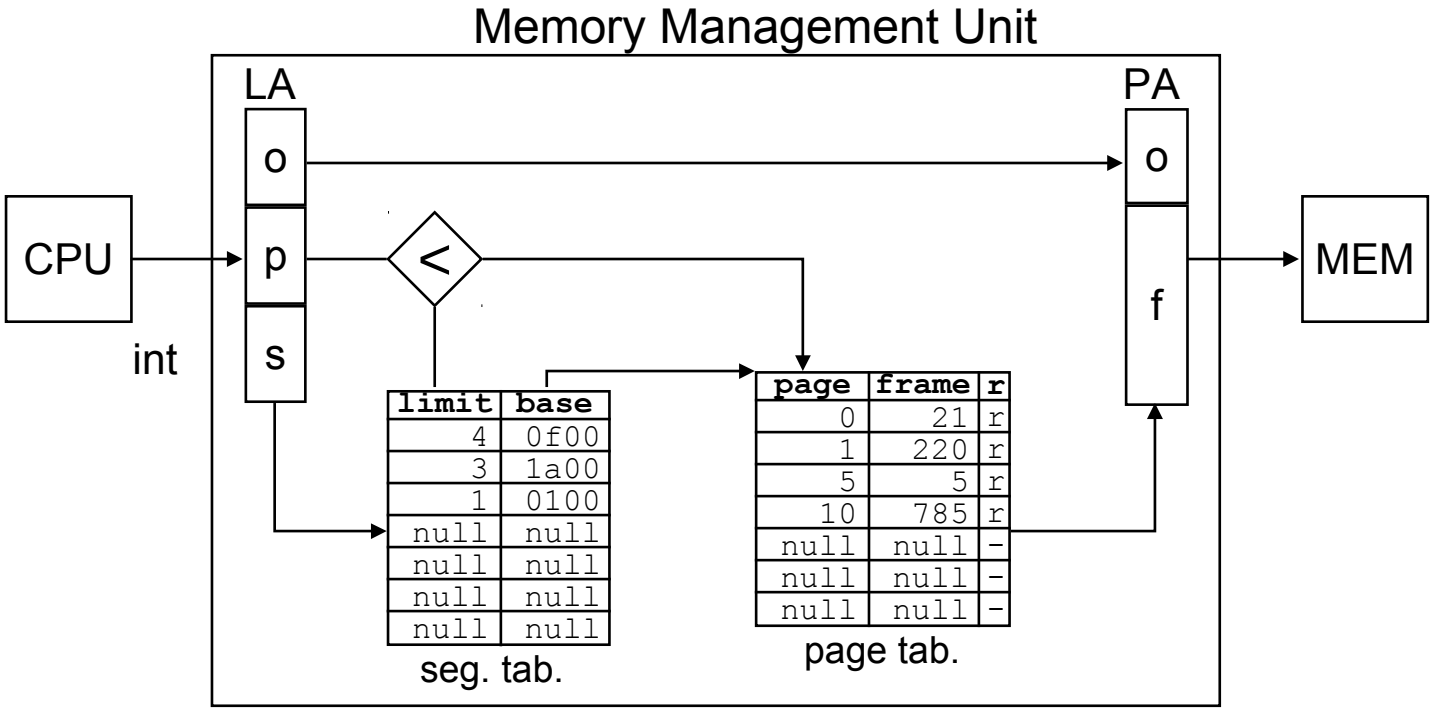
Exemplo de paginação

Tamanho dos quadros: 4 Kbytes
Tamanho da memória: 4 Gbytes (1 Mframes)
Espaço de endereçamento: 64 Mbytes (16 Kpages)
Endereço físico (EF): 32 bits (quadro = 20, offset = 12)
Endereço lógico (EL): 26 bits (página = 14, offset = 12)
Page table size: 16 Kentries

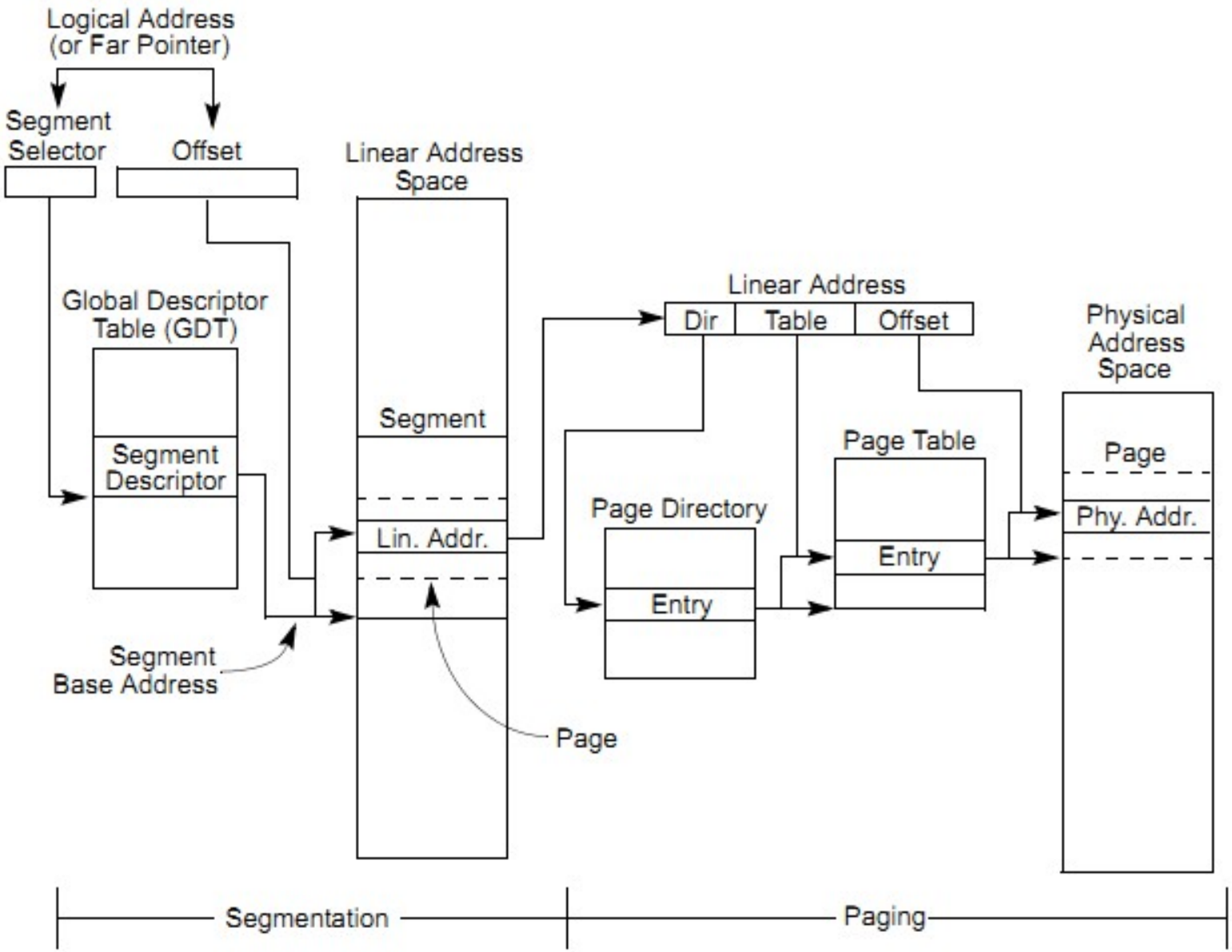


Paginação Segmentada

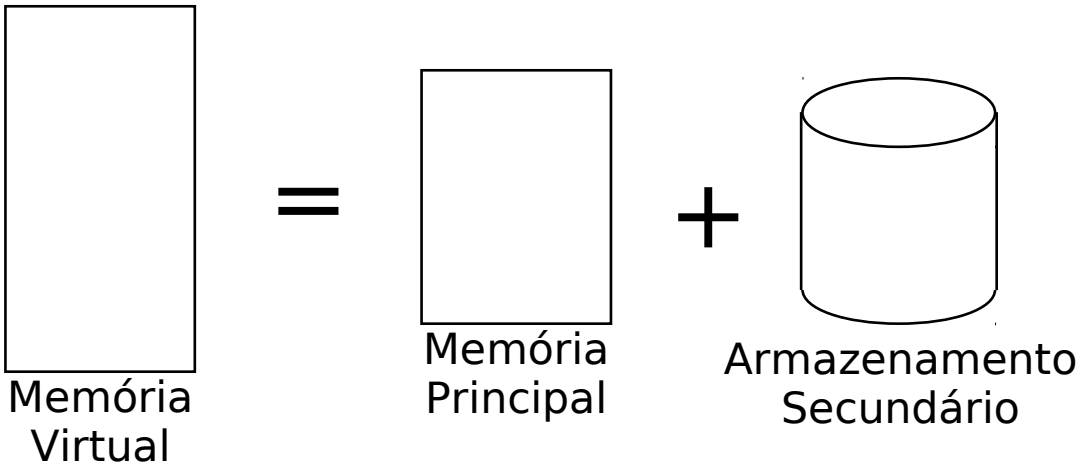
- Mescla de segmentação e paginação
- Espaço de endereçamento dos processos são divididos em segmentos, sendo cada um paginado
  - Tabelas de segmentos apontam para tabelas de páginas



Exemplo: x86


Logical Address
(or Far Pointer)
Segment
Selector
Offset
Linear Address
Space
Global Descriptor
Table (GDT)
Segment
Descriptor
Segment
Lin. Addr.
Linear Address
Dir
Table
Offset
Physical
Address
Space
Page Table
Page
Phy. Addr.
Page Directory
Entry
Entry
Segment
Base Address
Page
Segmentation
Paging

Memória Virtual



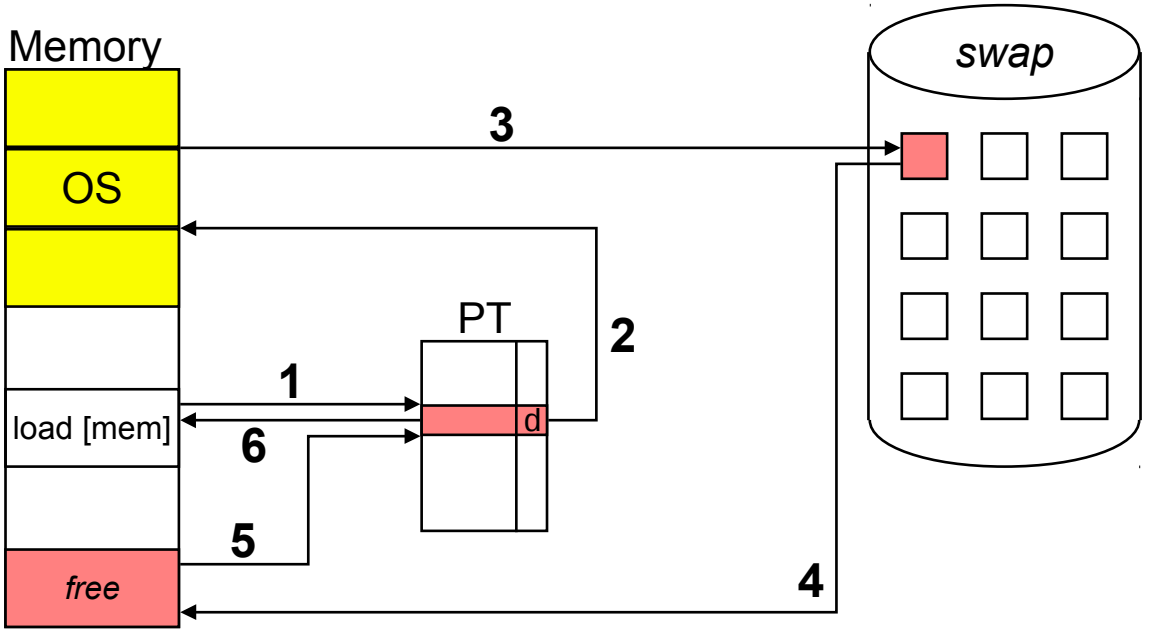
- Permite que um processo seja executado mesmo se não estiver totalmente carregado na memória
- Permite que processos aloquem mais memória que a quantidade de memória física
- Pode melhorar a utilização da CPU
- Pode reduzir o sobrecusto (*overhead*) de swap
- Pode matar seu sistema!

Swapping

- Processos podem ser temporariamente suspensos, tendo a memória a eles alocada copiada para um armazenamento secundário (ex.: disco) e depois liberada para outros processos (*swap out*)
  - Processos bloqueados
  - Processos de baixa prioridade
- Tais processos podem posteriormente ser retomados restaurando seus espaços de endereçamento da cópia no armazenamento secundário (*swap in*)

Paginação por Demanda

- Swapping orientado a páginas
  - Flags nas tabelas de páginas indicam se a uma página está na memória ou no disco
  - A MMU dispara uma exceção (*page-fault*) sempre que uma página acessada não está presente



1 – reference
2 – page-fault
3 – I/O request
4 – page in
5 – page table update
6 – instruction resume

Tratamento de Page-Fault

Page-fautl trap	1 µs
Save context	10 µs
Dispatch PF handler	1 µs
Locate page on disk	50 µs
Read page from disk	10 ms
Waiting queue	0 s
Seek	7 ms
Latency	2 ms
Transfer	1 ms
Scheduler	15 µs
Disk I/O completion	1 µs
Save context	10 µs
Dispatch disk I/O handler	1 µs
Update page table	15 µs
Ready queue waiting	0 s
Scheduler	15 µs
TOTAL	10,109 ms

Desempenho da Paginação por Demanda

- Fórmula

```
eat = (1 - p) x mat + p x pft
```

eat = effective access time
mat = memory access time
pft = page-fault handling time
p = page-fault probability

- Exemplo

mat = 50 ns
pft = 10 ms = 10.000.000 ns
eat = (1 - p) x 50 + p x 10.000.000
eat = 50 + 9.999.950 x p

SE p = 0.001, ENTÃO eat = 10.049,95 ns

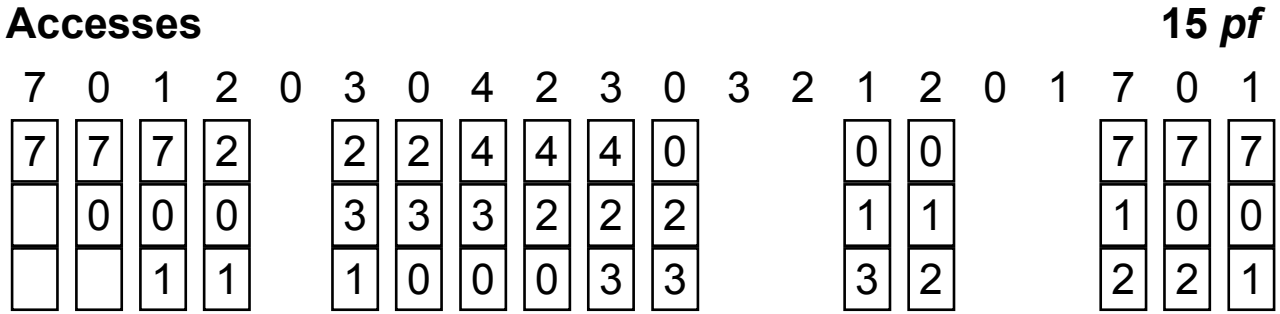
SE eat = 50 ns ENTÃO p = 0

Substituição de Páginas

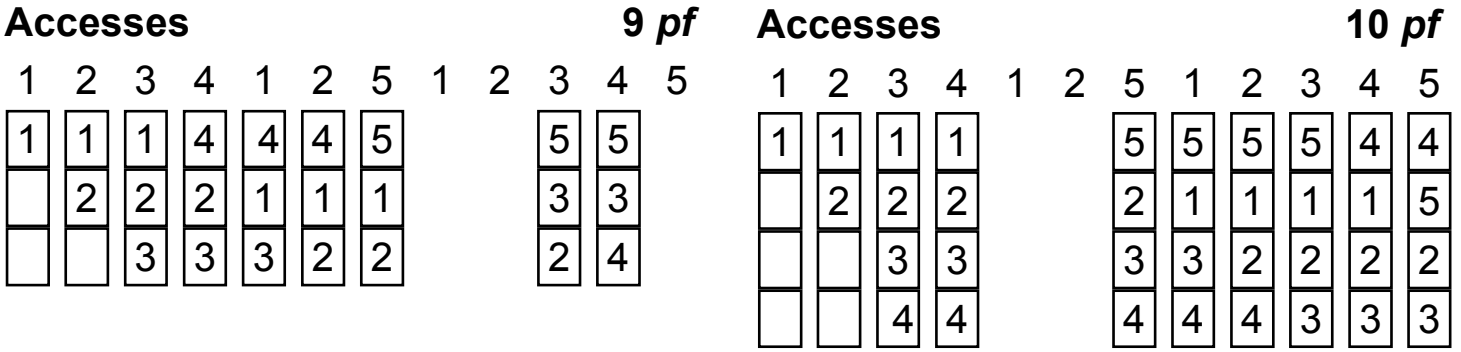
- Sempre que necessário, o SO seleciona páginas a serem movidas para o disco para liberar memória
- Critérios dos algoritmos
  - Minimizar taxa de page-fault
  - Minimizar I/O
- *Dirty bit*
  - O estado de cada página (modificado ou não) é mantido na entrada correspondente da tabela de páginas
  - A MMU automaticamente *seta* este bit quando uma página é modificada (escrita)
  - Páginas modificadas (*dirty* - “sujas”) precisam ser escritas no disco antes de serem reusadas

First-In First-Out (FIFO)

- Substitui a página que está na memória há mais tempo (ex.: páginas recebem um *timestamp*)
- Implementadas utilizando uma fila FIFO
- Exemplo

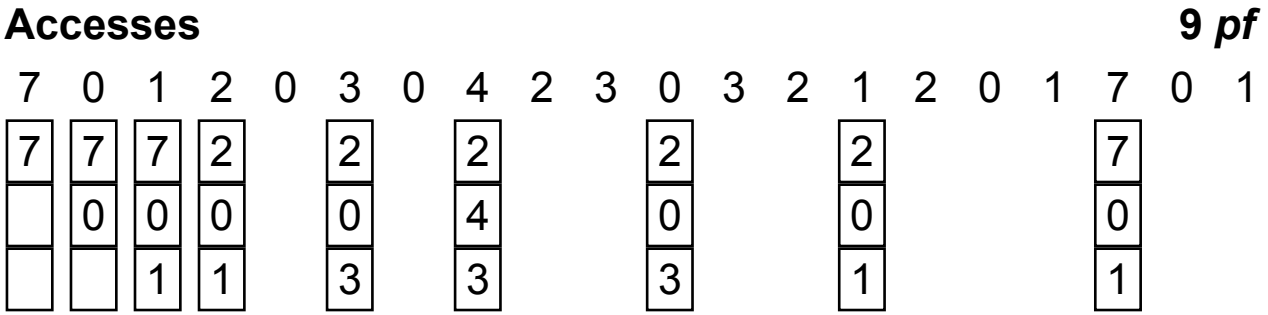


- Aumentar memória pode não diminuir taxa de PF



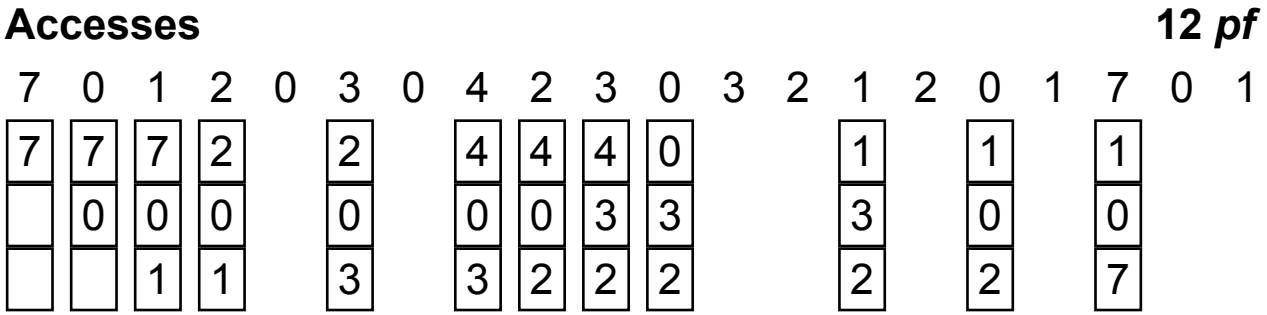
Algoritmo Ótimo

- Substitui a página que não será utilizada pelo mais longo período de tempo
- Não implementável, implica em conhecer o futuro
- Exemplo



Least Recently Used (LRU)

- Usa passado recente como aproximação do futuro próximo
- Substitui a página que não tem sido usada há mais tempo
- Exemplo



- Implementação
  - *Timestamp* para cada página
  - Lista encadeada de páginas

Aproximações do LRU

- Bit de referência
  - Cada página tem um bit de referência atribuído que é setado pela MMU quando a página é acessada
  - SO limpa este bit periodicamente
  - Ordem de uso não é conhecida
  - A página alvo é qualquer uma com o bit de referência limpo
- Palavra de referência
  - Bits adicionais de referência que são “shiftados” pela MMU
  - A página alvo é aquela com menores valores de referência
- Segunda chance
  - Páginas são rastreadas por uma FIFO circular
  - Se a página apontada tem o bit de referência limpo, ela é tomada para substituição
  - Senão, o bit é limpo e o ponteiro ajustado para a próxima página

Aproximações do LRU

- Least Frequently Used (LFU)
  - Usa um contador de referência para cada página que é incrementado pela MMU
  - A página alvo é a com o menor contador
  - Páginas acessadas intensamente no passado mas não mais acessadas, levarão mais tempo para serem substituídas
- Bits de Referência e Modificação
  - Além dos acessos, a MMU marca páginas que foram modificadas
  - Ordem de substituição
    - Não-acessada, não-modificada
    - Não-acessada, modificada
    - Acessada, não-modificada
    - Acessada, modificada

Alocação de Quadros

- Conjunto mínimo de quadros (*frames*)
  - Instruções e dados podem ser espalhados por várias páginas
  - Dependente de arquitetura
  - Instruções precisam ser reiniciadas após um page-fault
- Quadros por processo
  - Proporcional ao tamanho do processo (i.e. memory footprint)
  - Igual a todos processos
- Interferência de processos
  - Um processo pode causar a substituição de uma página inicialmente alocada a outro processo
  - Um processo pode apenas substituir suas próprias páginas

Thrashing

- Um processo está em “thrashing” se ele gasta mais tempo substituindo páginas que executando
- Causas
  - SO monitora utilização da CPU e permite mais processos ao mesmo tempo
  - Se utilização da CPU estava baixa devido a page-faults, aumentar o número de processos pode causar trashing
  - Thrashing só ocorre se substituição de páginas global (i.e. de outros processos) é permitida
- Prevenção
  - A qualquer tempo, um processo em execução precisa ter um conjunto de páginas disponíveis que complete suas demandas: seu conjunto de páginas de trabalho

Considerações Finais

- Carregamento de processos
  - Sob demanda
  - De uma vez para memória principal
  - De uma vez para disco (swap)
- Tamanho de páginas
  - Grande: menos page-faults, menos I/O, menos tabelas de páginas
  - Pequena: menor fragmentação interna
- Resultados de I/O
  - Páginas que receberão resultados de I/O precisam ser fixas
- Programação e geração de código
  - Embora a funcionalidade de memória virtual seja transparente aos programas, padrões de acesso à memória podem influenciar a funcionalidade (ex.: grandes matrizes)