Gerência de Memória

Ambientes Operacionais

Memória

- A **memória** pode ser vista como um *array* (vetor) de células de armazenamento (palavras ou bytes), cada célula com seu endereço

0	2
1	31
2	4
3	35
4	26
5	124
6	42
7	12
8	42

⋮

Memórias física, lógica e virtual

- Memória física
  - É a memória implementada pelo hardware
- Memória lógica de um processo
  - É a memória endereçada pelas instruções de máquina do processo
- Memória Virtual
  - É uma memória implementada pelo SO, com o auxílio da memória secundária (disco). Comumente, é implementada através de paginação ou segmentação.
  - Normalmente, é maior que a memória física do computador

Gerência de Memória

- Rotinas do SO que controlam o uso da memória.
  - Controle de quais partes da memória encontram-se livres e quais estão em uso
  - alocação da memória de acordo com as necessidades dos processos
  - liberação da memória alocada após o término de um processo
  - transferência do processo, ou parte dele, entre a memória principal e a memória secundária

Mecanismos para Gerência de Memória

- máquina pura
- monitor residente
- swapping
- partições múltiplas
- paginação
- segmentação
- sistemas combinados

Máquina Pura

- É o esquema mais simples, pois não existe gerência de memória
- O usuário lida diretamente com o hw e possui total controle sobre toda a memória
- Fornece maior flexibilidade para o usuário, máxima simplicidade e custo mínimo, pois não exige sw ou hw especiais
- O software para essas máquinas é desenvolvido através de compiladores que executam em outras máquinas (compiladores cruzados)

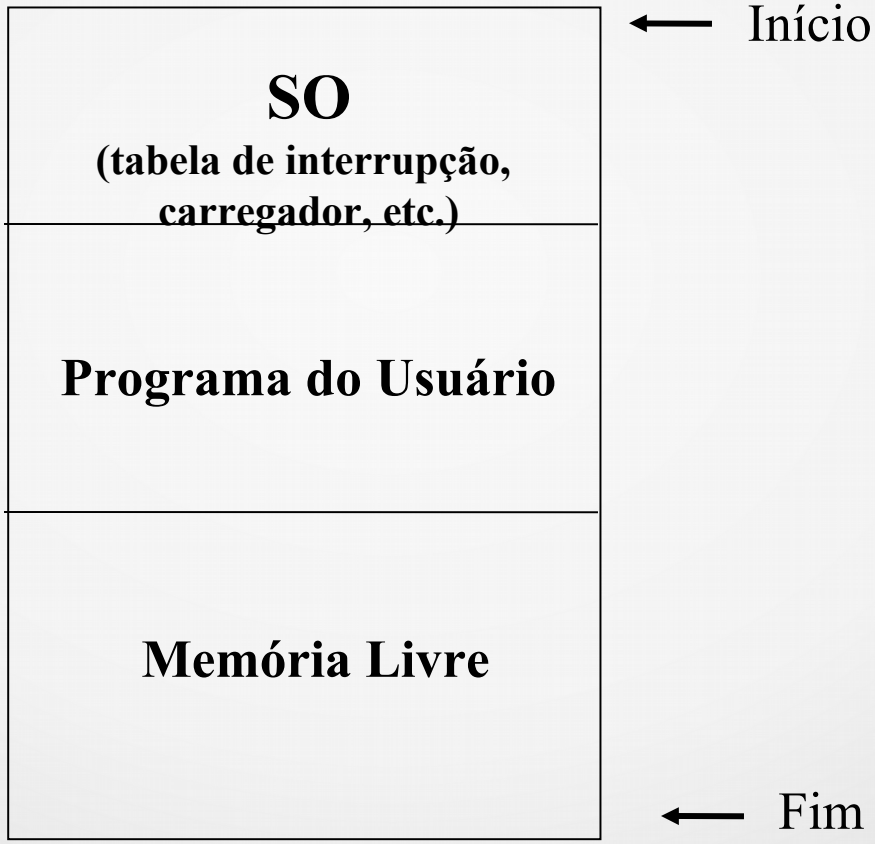
Máquina Pura

- Problemas:
  - não existe a infra-estrutura do SO (rotinas de E/S, por exemplo)
  - não há monitor residente para controlar chamadas de sistema ou erros
- Viável apenas em sistemas dedicados, onde o computador controla um equipamento específico.

Sistemas monoprogramados

- Com monoprogramação a gerência de memória fica simples
- O espaço é dividido entre o SO e o processo do usuário que está sendo executado

Monoprogramação



Monoprogramação

- Vantagens:
  - simplicidade
  - custo baixo de implementação e uso
  - não ocorrência de *overheads* decorrentes do gerenciamento de memória
  - flexibilidade

Monitor Residente

- Normalmente, este esquema é usado em sistemas monoprogramados
- Memória dividida em duas partes:
  - área do SO
  - área do usuário
- Registrador limite: contém o primeiro endereço do programa usuário

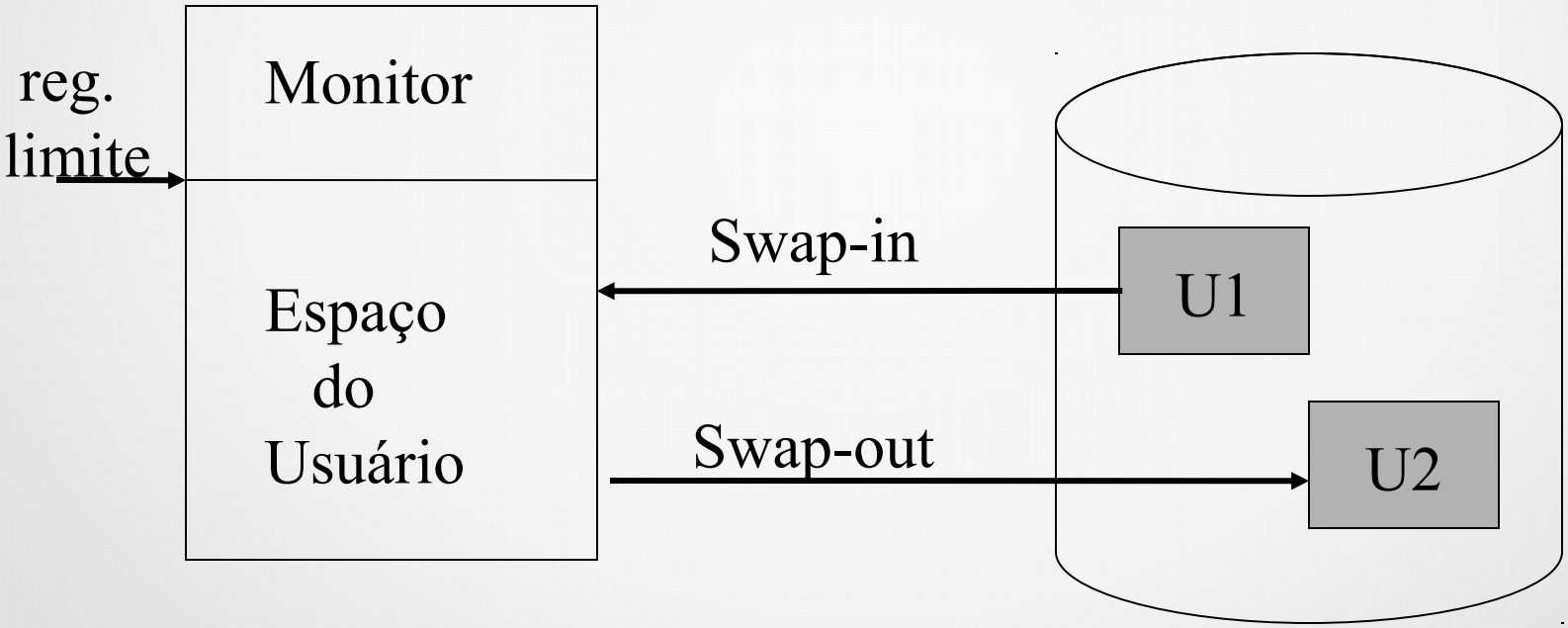
O problema da Relocação

- **Relocação** é a transformação dos endereços relativos do programa em endereços absolutos
- Se o conteúdo do registrador limite é previamente conhecido, os endereços absolutos podem ser gerados na compilação
- Se o endereço inicial do programa só vai ser conhecido no momento da carga, deve haver relocação:
  - **Relocação estática** : realizada pelo carregador
  - **Relocação dinâmica** : os endereços não são modificados, pois usa-se um **registrador base**

Multiprogramação através de Swapping

- É implementada por um SO do tipo monitor residente
- O esquema de gerenciamento de memória é estendido para implementar **swapping**
- O programa que perde a CPU é copiado p/ disco, enquanto o programa que ganha a CPU é transferido do disco p/ a memória principal

Swapping



Partições Múltiplas

- Com multiprogramação, é conveniente ter vários programas na memória ao mesmo tempo para que a CPU seja rapidamente alternada entre eles
- Solução: dividir a memória em partições (cada partição irá conter um programa)
  - **partições fixas** (normalmente o hw usa registradores **limite inferior** e **limite superior**)
  - **partições variáveis** (normalmente o hw usa registradores **base** e **limite**)

Partições Fixas

- Divide-se a memória em um número fixo de blocos (do mesmo tamanho ou não)
- Quando um processo é criado, ele é colocado em uma fila (em disco) à espera que uma partição de tamanho suficiente se torne disponível

Partições Fixas

- Para definir a partição onde o programa vai ser colocado, existem duas opções:
  - Montar uma fila individual para cada partição
  - Montar uma fila única para todas as partições

Fragmentação

- São perdas (desperdício) de memória:
  - **fragmentação interna**: memória é perdida dentro da partição alocada (é um desperdício de espaço dentro da partição usada pelo processo)
  - **fragmentação externa**: ocorre quando existe espaço disponível mas este é pequeno demais para os processos que estão à espera (perda de espaço fora das partições alocadas)

Partições Fixas

- O controle de partições fixas é conceitualmente simples. Necessita levar em conta:
  - tamanhos das partições de memória
  - algoritmo para gerenciar a lista de processos em espera

Partições Fixas

- Exemplo: memória de 256K
  - espaço do SO: 64K
  - espaço para processos pequenos: 16K
  - espaço para processos médios: 48K
  - espaço para processos grandes: 128K

Paramos aqui – 14/02/2017

Partições Variáveis

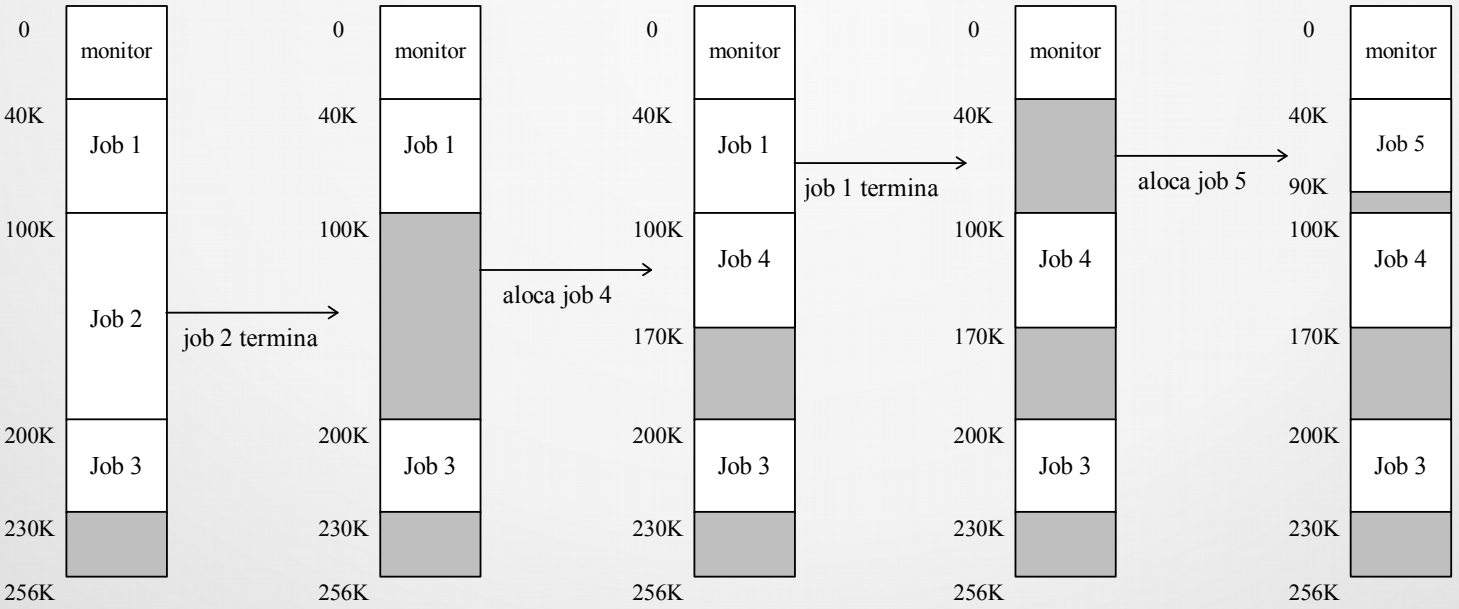
- os tamanhos das partições variam de acordo com a necessidade
- Tanto o tamanho quanto o número de partições variam dinamicamente
- Elimina a fragmentação interna
- Introduz a fragmentação externa
- Mais difícil de implementar

Partições Variáveis

- O SO mantém uma lista indicando quais partes da memória estão disponíveis e quais estão ocupadas.
- As áreas disponíveis são denominadas **lacunas** (***holes***)
- Quando um processo chega para ser executado, a lista de lacunas é consultada e é escolhida uma lacuna de tamanho suficiente

Partições Variáveis

JOB	1	2	3	4	5
Memória	60K	100K	30K	70K	50K
Tempo	10	5	20	8	15



Partições Variáveis - Características

- Vai existir um conjunto de áreas livres (lacunas) espalhadas pela memória
- Para executar um programa, o conjunto é pesquisado à procura de uma área maior ou igual à necessidade
- se a área é maior, a parte restante vai continuar livre
- quando um processo termina, a área é liberada. Se a área é adjacente a outra área livre, as duas áreas são aglutinadas em uma única lacuna

Partições Variáveis - Algoritmos de Alocação

- Algoritmos para escolha da área livre (alocação dinâmica da memória):
  - **first-fit**: aloca o primeiro espaço livre de tamanho suficiente
  - **best-fit**: aloca o menor espaço livre que seja suficiente. Produz a menor sobra de espaço
  - **worst-fit**: aloca o maior espaço livre. Produz a maior sobra de espaço livre (a sobra é mais útil que a gerada por best-fit)

Partições Variáveis - Compactação

- Para resolver o problema da fragmentação externa, a solução é a **compactação da memória**:
  - os programas são deslocados na memória de forma que todo os espaço livre fique reunido em uma única lacuna
  - custo de tempo de CPU (overhead) *sobrecarga
  - necessidade de relocação dinâmica (pois a relocação estática impossibilita a compactação)

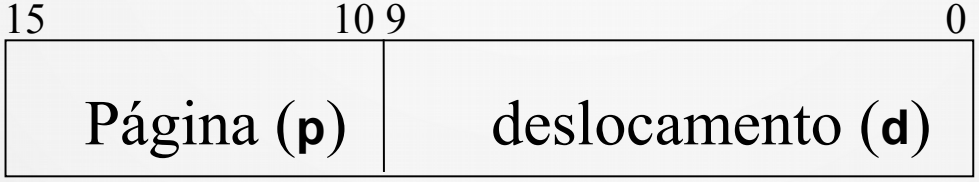
Paginação

- A memória física é dividida em um número de partições de mesmo tamanho, denominadas **páginas físicas, quadros** ou ***frames***
- A memória lógica é dividida em partições do mesmo tamanho, denominadas **páginas lógicas** (ou, simplesmente, páginas)
- Cada página lógica é carregada em um *frame* quando o processo é carregado na memória principal
- Nessa ocasião, uma **tabela de páginas** é criada
- Permite que o espaço físico ocupado por um processo seja não contíguo

Paginação

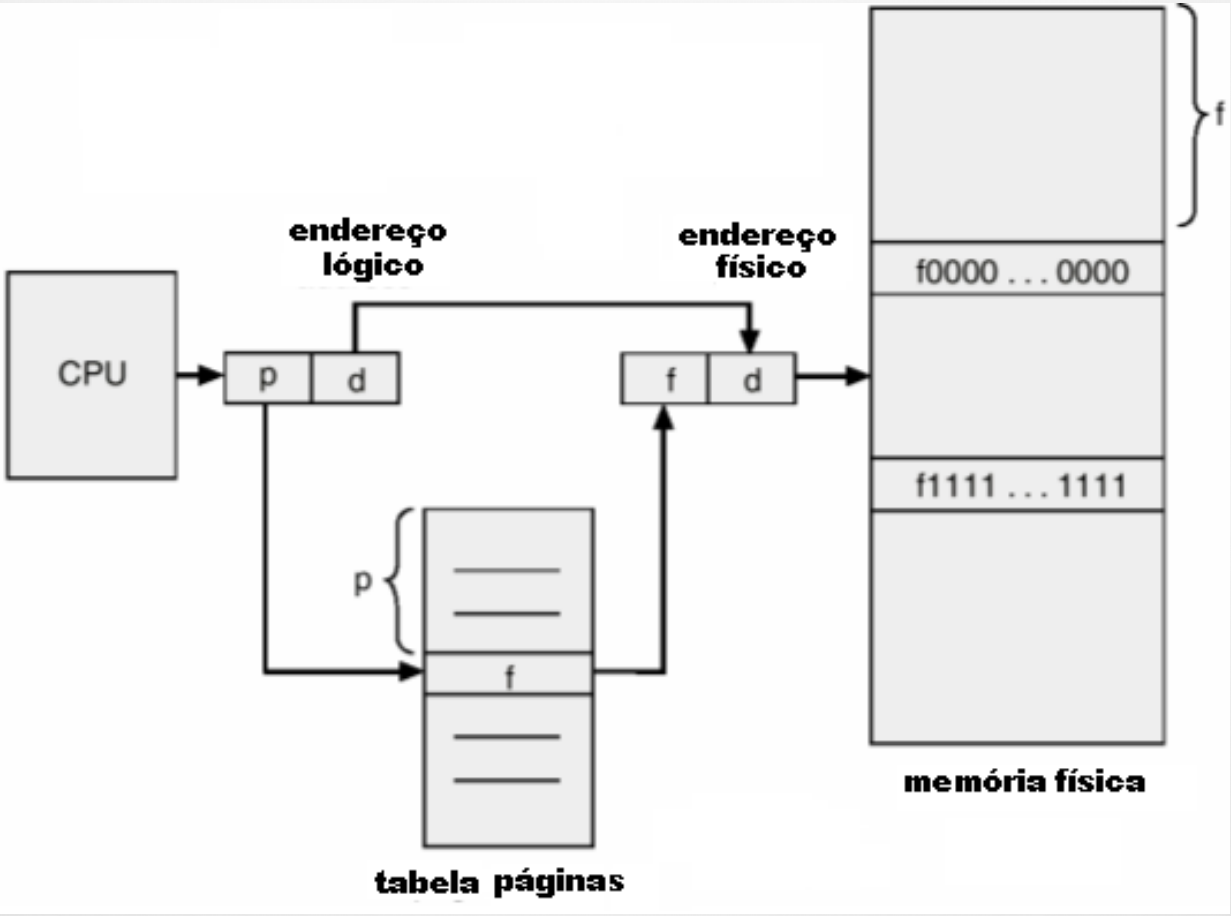
- Se um processo tem tamanho K, os seus endereços lógicos (endereços especificados nas suas instruções) vão desde 0 até K-1. Este é o **espaço de endereçamento** do processo.
- Cada endereço lógico é quebrado em duas partes:
  - número de página **p**
  - deslocamento **d**

- Endereço lógico:



- Acontece **relocação dinâmica**, pois cada endereço lógico é traduzido em endereço físico em tempo de execução

Paginação


endereço
lógico
endereço
físico
CPU
p
d
f
d
f
f0000 . . . 0000
f1111 . . . 1111
p
f
memória física
tabela páginas

Implementação da Tabela de Páginas

- Conjunto de registradores dedicados
- Memória Principal
- Memória Associativa

Tabela de Páginas em Conjunto de Registradores Dedicados

- Na mudança de processo em execução estes registradores são carregados com os valores correspondentes ao novo processo

Tabela de Páginas na Memória Principal

- Cada descritor de processo contém o endereço de sua respectiva tabela de páginas.
- A UCP possui um registrador que aponta para a tabela de páginas atual
- Para acessar um dado na memória são necessários dois acessos: um de mapeamento (acesso à tabela) e outro para acessar o dado

Tabela de Páginas em Memória Associativa

- Memória de alta velocidade, onde cada posição possui dois campos:
  - **chave** e **valor**
- Pesquisa rápida, mas o hw é caro
- **chave** = número de página lógica
- **valor** = página física correspondente

Tabela de Páginas – bits de controle

- Cada entrada da tabela possui alguns bits adicionais para implementar proteção
  - um bit para indicar se a página é de **apenas leitura** (***read only***)
  - um bit para indicar se a página é **válida** ou **inválida**

Segmentação

- Divisão do espaço de endereçamento em um número de partições com tamanhos distintos
- Aproxima-se mais da visão do programador: um programa é uma coleção de segmentos de tamanho variável

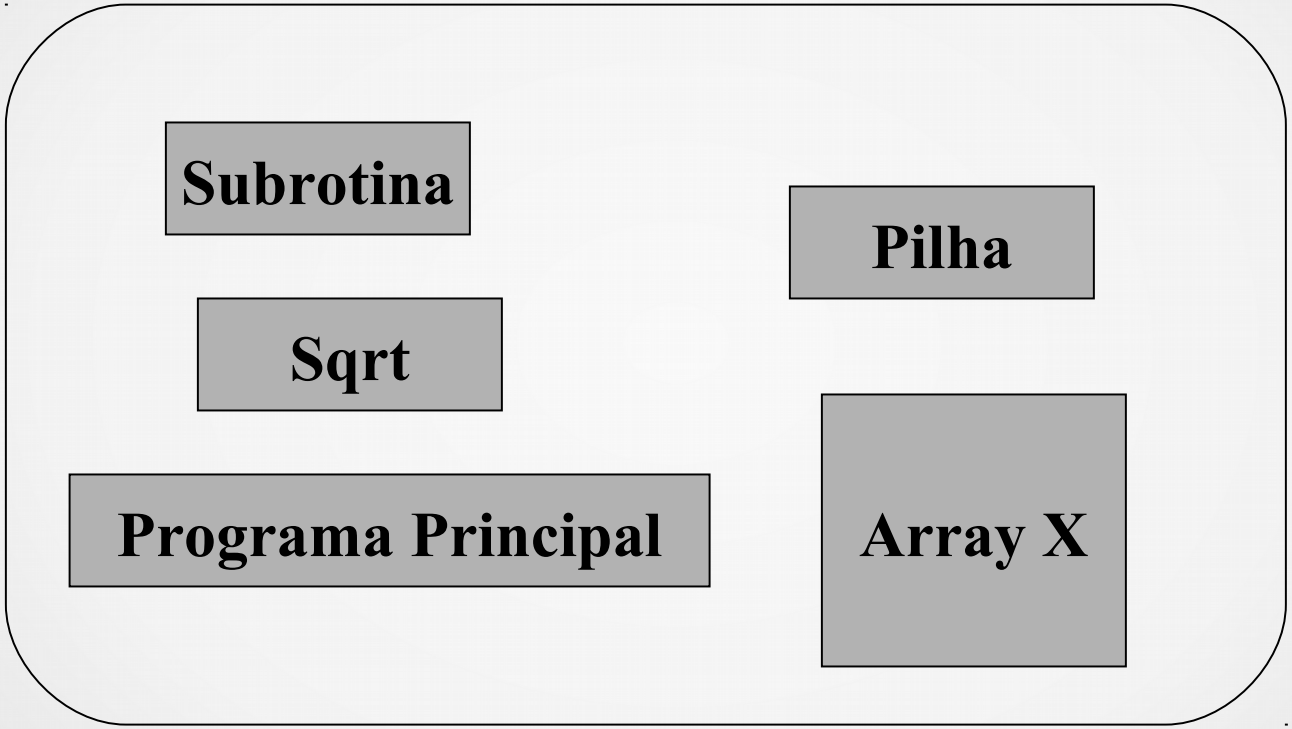
Segmentação

- A memória lógica é constituída por um conjunto de segmentos, cada um com um nome e um tamanho (na prática, os segmentos são identificados por números e não por nomes)
- Uma posição da memória lógica é referida por um par (**s**, **d**)
  - **s** é o número do segmento
  - **d** é o deslocamento (*offset*) dentro do segmento

Segmentação

- Os compiladores e montadores criam automaticamente os segmentos que constituem o programa
- Na carga do programa cada segmento recebe um número de segmento específico

Segmentação

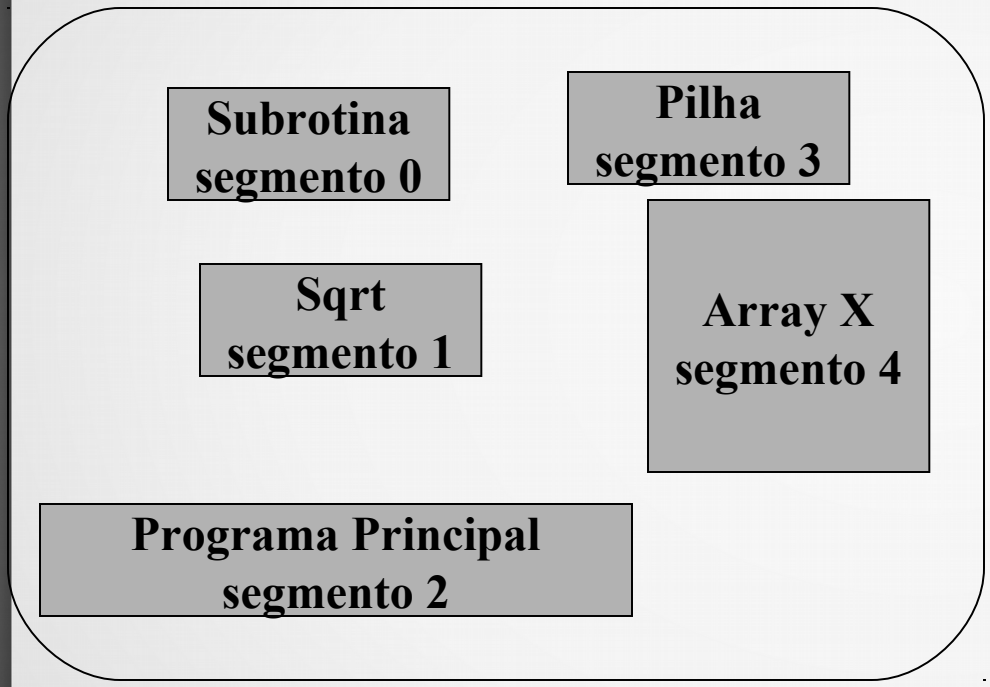


Espaço de Endereçamento Lógico do Processo

Segmentação

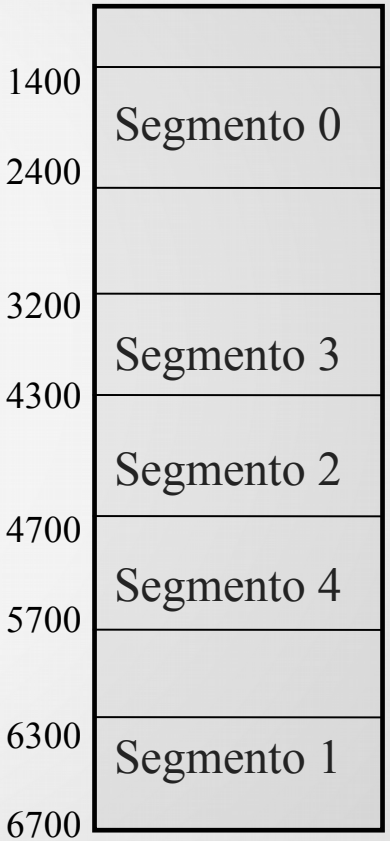
- É necessário mapear cada endereço lógico do tipo (**s**, **d**) para o endereços da memória física correspondente
- Para isso cada processo possui a sua **tabela de segmentos**
- A tabela de segmentos pode ser colocada em registradores rápidos ou na memória principal. Normalmente, é usado o esquema de memória associativa (na tabela associativa ficam os segmentos mais recentemente acessados e seus endereços)

Tabela de Segmentos



seg	base	limite
0	1400	1000
1	6300	400
2	4300	400
3	3200	1100
4	4700	1000

Tabela de segmentos



Memória Física

Segmentação

- Pode-se associar atributos aos segmentos, possibilitando assim uma proteção ou compartilhamento destes segmentos
- Bit de proteção associado a cada entrada da tabela de segmentos
- A segmentação facilita o compartilhamento entre usuários

Fragmentação

- A segmentação apresenta o problema de fragmentação externa. A alocação de espaço utiliza os métodos: first-fit, best-fit, etc.
- Na paginação, ocorre a fragmentação interna, pois, em média, a última página do processo é ocupada apenas pela metade

Segmentação e Paginação - resumo

- Qual o melhor?
  - Discussão antiga, sem vencedores
- Fragmentação
  - Paginação : apresenta fragmentação interna
  - Segmentação : apresenta fragmentação externa
- Administração
  - Paginação é mais simples
- Proteção (segurança) e compartilhamento
  - Segmentação é melhor, pois:
    - segmentos são unidades lógicas
    - páginas são mais misturadas (dados, código)

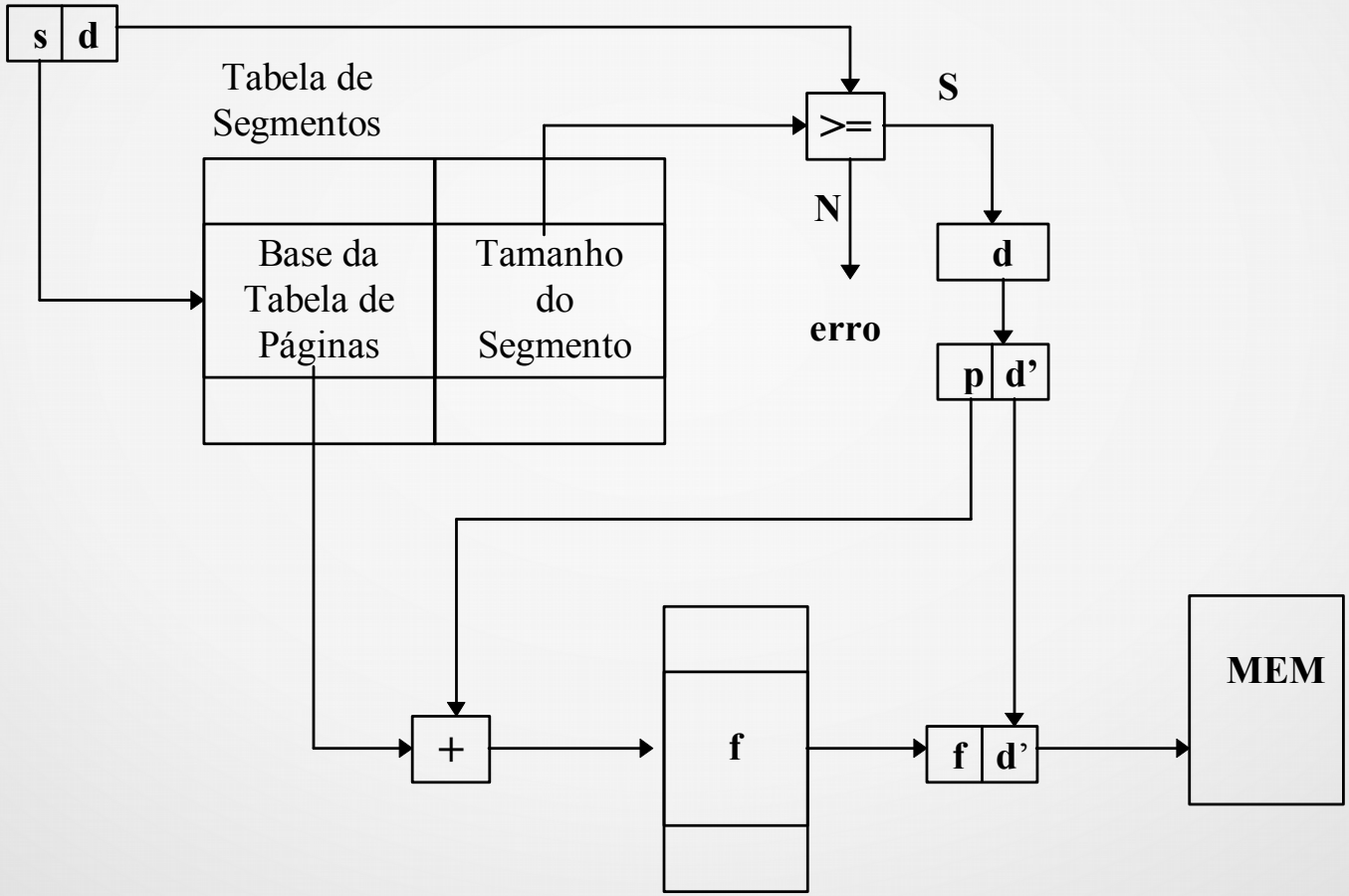
Segmentação e Paginação - resumo

- Espaço endereçamento
  - Paginação
    - Espaço de endereçamento lógico é um espaço único, contínuo, cujos endereços vão desde zero até MAX (onde MAX = tamanho do programa menos 1)
  - Segmentação
    - Espaço de endereçamento lógico é formado por um conjunto de segmentos. Cada segmento é um espaço contínuo, cujos endereços vão desde zero até MAX (onde MAX = tamanho do segmento menos 1)

Sistemas Combinados

- Existem sistemas onde a paginação e a segmentação são usadas em conjunto, procurando tirar proveito de ambos os esquemas.
  - Segmentação paginada (mais comum)
  - Paginação segmentada (menos comum)

Segmentação Paginada - Fluxograma



Memória Virtual

- Para alguns autores, memória virtual é sinônimo de memória lógica (i.é, é a memória endereçada pelas instruções de máquina de um processo)
- Para outros, só existe memória virtual quando o sistema consegue executar programas sem que eles estejam completamente carregados na memória física.

Paginação por demanda

- Um programa pode ser executado com poucas de suas páginas na memória
- Quando necessário, uma página é trazida do disco para memória e utilizada (**demanda**)

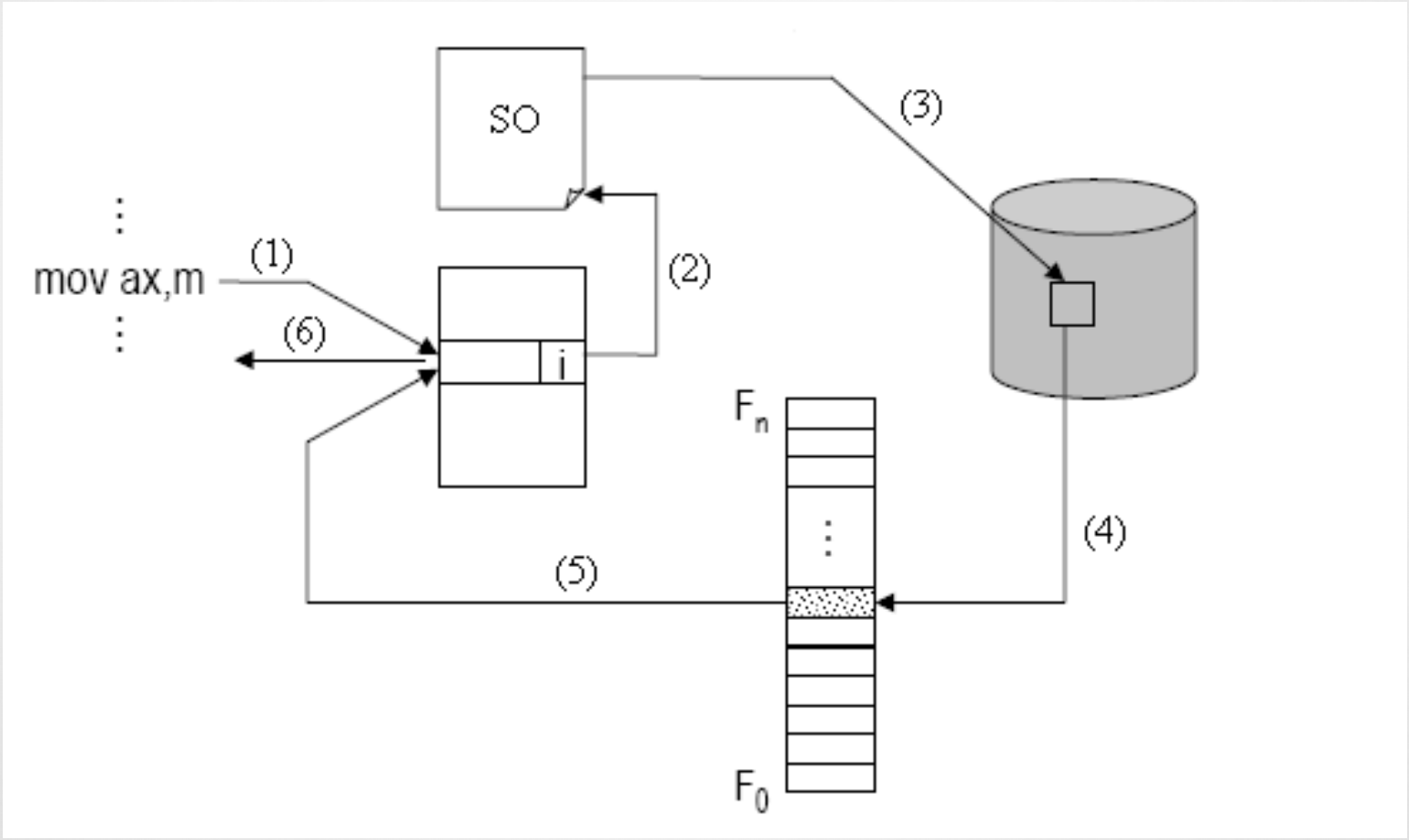
Paginação por demanda

- É uma extensão do mecanismo de paginação simples
- As páginas de um processo podem estar presentes na memória ou não. As páginas não presentes estão marcadas como inválidas
- Se uma página inválida é referida, o SO verifica se ela está em disco (***page fault***) ou se realmente é uma página fora do espaço lógico do processo

Tratamento de *page-fault*

- O processo que gerou a falta de página é suspenso, seu descritor de processo é removido da *ready list* e inserido em uma fila especial, a "fila dos processos esperando por carga de página lógica";
- Uma página física livre deve ser alocada;
- A página lógica acessada deve ser localizada no disco (a localização das páginas no disco é indicada no registro descritor do processo);
- Uma operação de leitura do disco deve ser solicitada, indicando o endereço da página lógica no disco e o endereço da página física alocada.

Tratamento de *page-fault - representação*

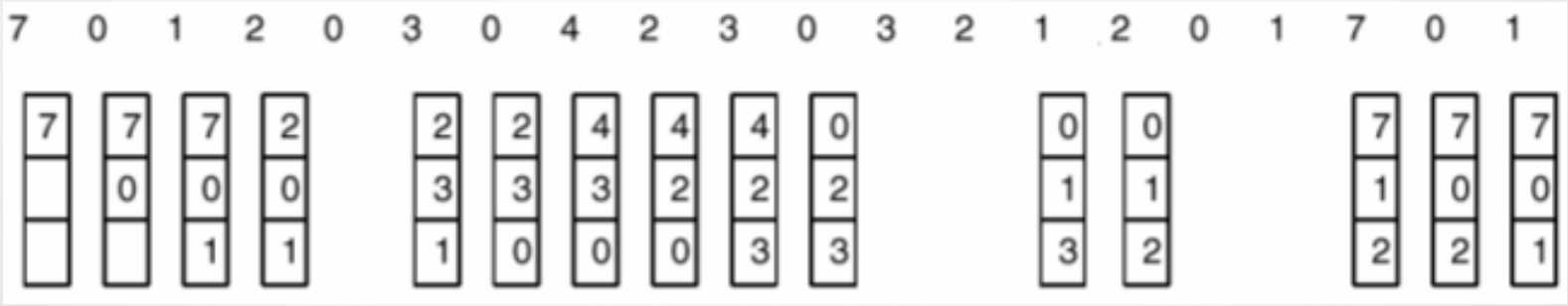


Substituição de Páginas

- O SO deve escolher uma página para ser substituída (**página vítima**)
- Bits auxiliares
  - Bit de sujeira (*dirty bit*)
    - Se ligado, indica se a página foi alterada na memória (nesse caso, sua cópia no disco não está atualizada)
  - Bit de referência (*reference bit*)
    - Se ligado, indica se a página foi acessada recentemente (este bit é desligado pelos algoritmos de substituição de páginas)
  - Bit de tranca (*lock bit*)
    - Se ligado, indica que a página não pode ser escolhida como vítima (swapped-out), p.ex., página envolvida em operação de E/S.

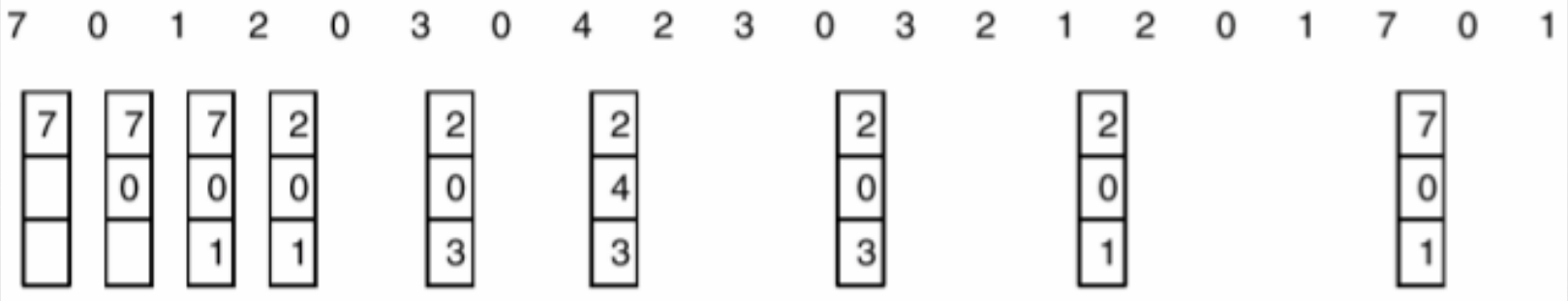
Algoritmos de substituição

- FCFS (ou FIFO)
  - Mais antiga sai. O SO deve memorizar a ordem em que as páginas são trazidas para a memória



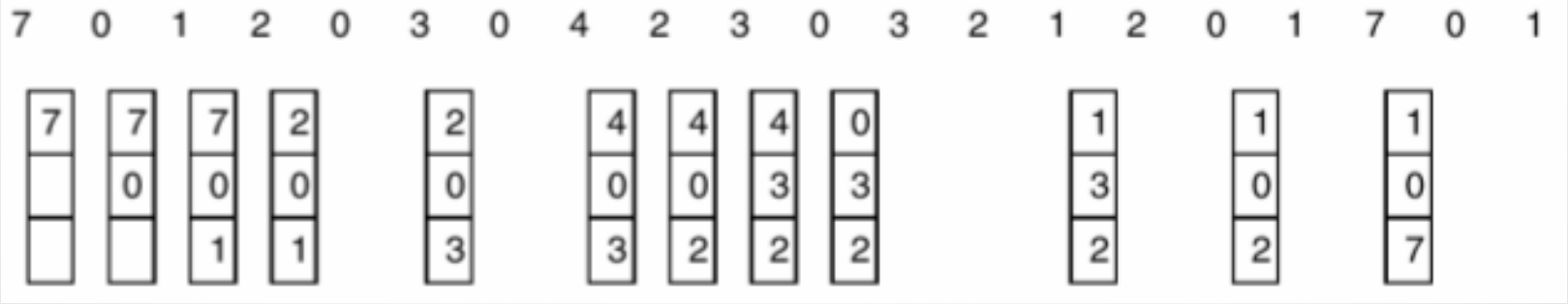
Algoritmos de substituição

- Ótimo (teórico)
  - Substitui a página que será acessada por último



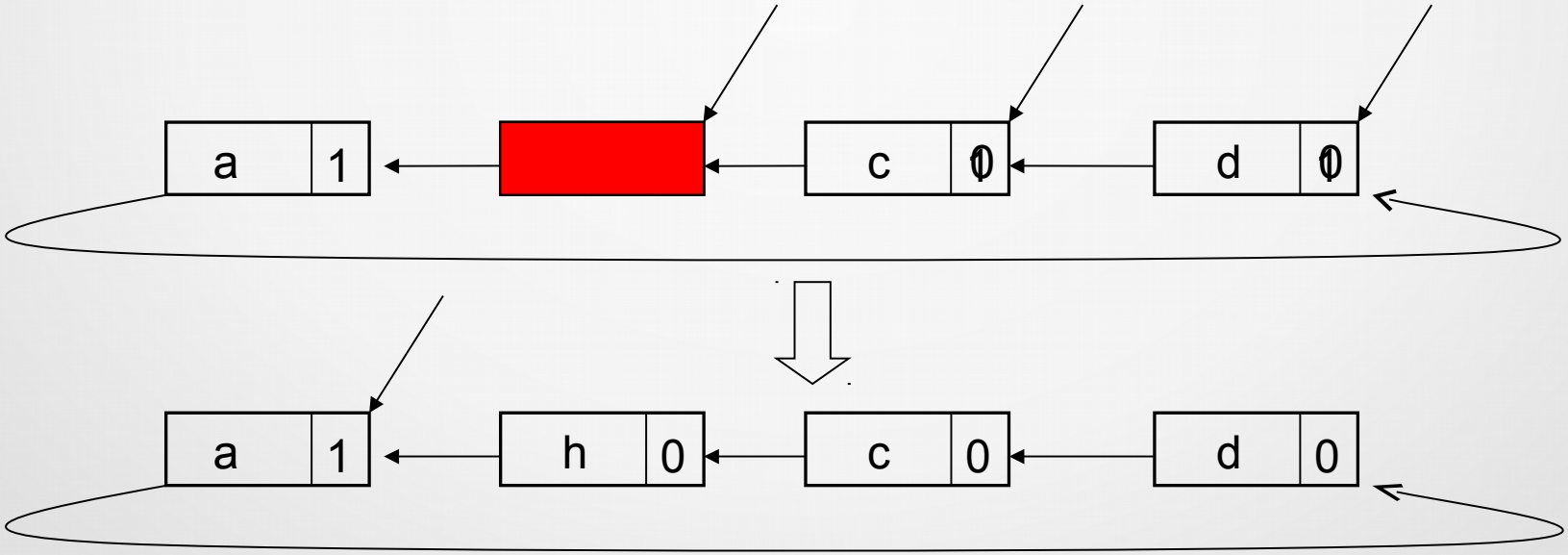
Algoritmos de substituição

- LRU
  - A página usada há mais tempo sai (precisa de HW especial)



Algoritmos de Substituição

- Segunda Chance
  - Utiliza o bit de referência. A tabela de páginas é vista como uma lista circular



Alocação de Páginas Físicas

- Quantas páginas o processo deve ter na memória física a cada instante?
- Alocação local
- Alocação global

Alocação de Páginas Físicas

- Alocação local
  - Não afeta outros processos
  - Conjunto fixo de páginas por processo. Em caso de falta de página, escolhe-se uma destas para ser substituída
  - Problema: quantas páginas? (processos diferem muito entre si)

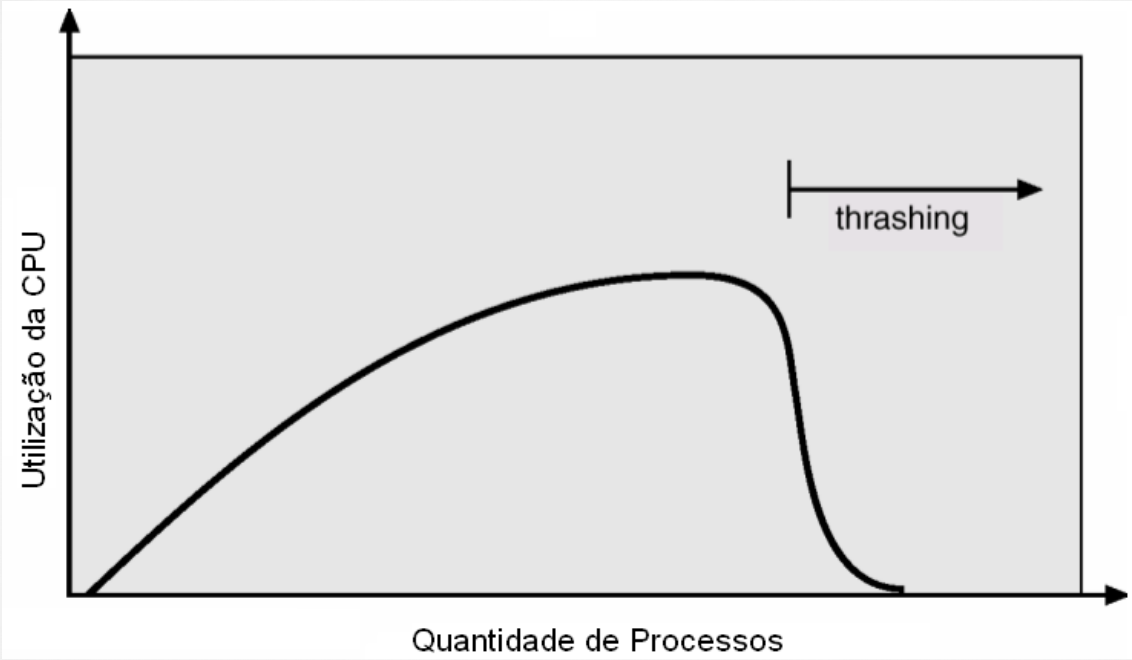
Alocação de Páginas Físicas

- Alocação global
  - Não distingue entre os processos na hora de escolher uma página vítima
  - Problema: processos de baixa prioridade são prejudicados

Thrashing

- Se um processo tem poucas páginas, sua taxa de *page faults* é alta. Isto ocasiona:
  - **Má utilização da CPU**
- **Thrashing** ≡ processo gasta mais tempo fazendo *swap in* e *swap out* de páginas do que realizando suas operações

Thrashing



Aumentando o número de processos na memória, diminui o número de páginas físicas que cada um recebe. O funcionamento do sistema pode chegar a um ponto em que a UCP não tem o que executar, pois todos os processos ficam bloqueados esperando por busca de página no disco.

Working set

- O ***working set*** de um processo no tempo **t**, para um período de observação (ou janela temporal) **τ**, é definido como o conjunto das páginas que o processo refere no intervalo de tempo (**t**-**τ**; **t**)

Working set
Algoritmo FFP (Freqüência de Falta de Páginas)

- São utilizados 3 parâmetros de configuração: **tax_max**, **tax_min** e **período de contabilização** de *page faults*.
- Seja **x** o número de *page faults* geradas pelo processo no período de contabilização:
  - Se **x** > **tax_max**, o SO aloca mais páginas físicas para esse processo
  - Se x < **tax_min**, o SO libera algumas páginas físicas desse processo

    As páginas liberadas são inseridas na **lista de páginas físicas livres** do sistema.

Desempenho da paginação por demanda

$$t_e = (1-p) * t_{am} + p * t_{tf}$$

- onde **p** é a taxa de *page faults*, t_{am} é o tempo médio de acesso à memória principal quando não ocorre *page fault* e t_{tf} é o tempo médio necessário para o tratamento completo de uma falta de página
- t_e é o tempo efetivo de acesso à memória