Gerência de Memória

Sistemas Operacionais

Prof. MSc. Rodrigo D. Malara

Memória

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

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

•
•
•

Gerenciamento de Memória

- Principal operação – trazer programas para dentro da memória principal para ser executados pelo processador.
- Dividir dinamicamente a parte “User” da memória principal de forma que acomode vários processos.
- Alocar a memória de forma eficiente para empacotar tantos processos na memória quanto possível para evitar que o processador fique ocioso.
- Deve ser capaz de rodar um programa em que seu tamanho seja maior que o disponível pela memória real.

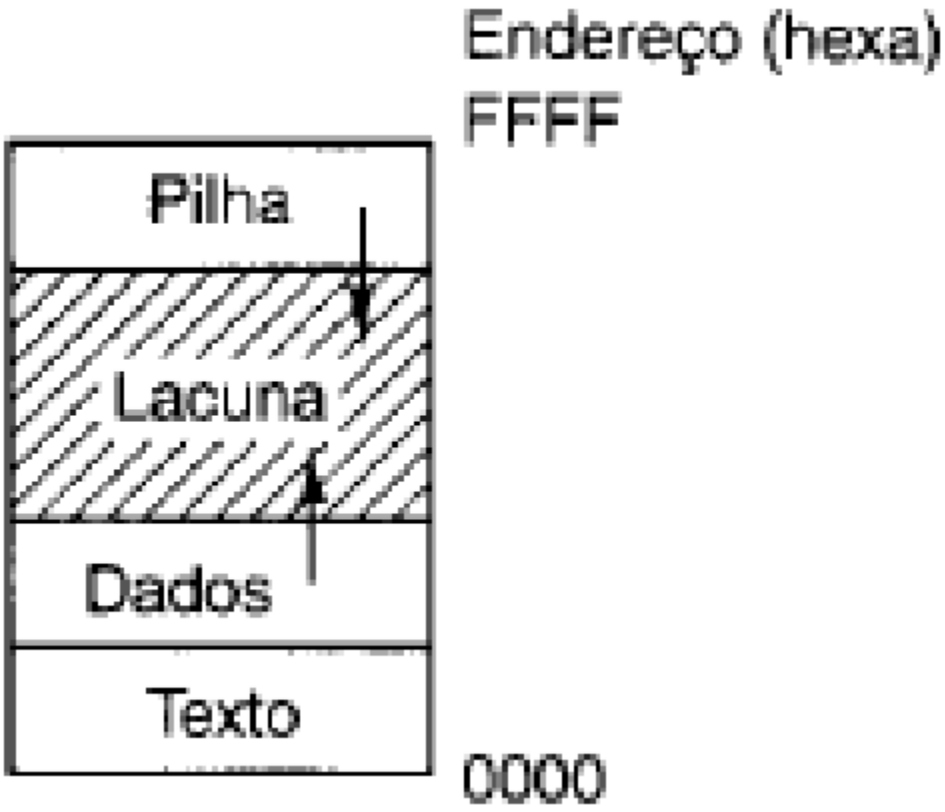
Requisitos

- Relocação
- Proteção
- Compartilhamento
- Organização lógica
- Organização física

Requisito - Relocação

- Capacidade de mover um processo de uma região da memória principal para uma outra sem invalidar as referencias de memória dentro do processo;
- O programador não sabe onde o programa é colocado na memória quando ele é executado
- O hardware do processador e o SO devem ser capazes de traduzir os endereços de referencia de memória no código do programa para o endereço físico da memória.

Técnica de Endereçamento
Imagem do Processo



Requisito - Proteção

- Cada processo deve ser protegido contra interferências não desejáveis de outros processos de forma acidental ou intencional
- Processos não devem ser capazes de referenciar localizações de memória de outro processo sem permissão;
- O hardware é responsável por fazer a verificação, caso a violação ocorra, o mesmo deve abortar tais instruções no ponto da execução;
- O SO não pode prever todas as referencias de memória que o programa fará (custo muito alto).

Requisito - Compartilhamento

- Permitir que vários processos acessem a mesma área de memória principal
- É vantajoso, que :
  - processos que são executados em um mesmo programa acessem a mesma cópia do programa e
  - Processos que estão cooperando em uma mesma tarefa compartilhem acesso a uma mesma estrutura de dados
- Deve permitir o compartilhamento sem comprometer o requisito de proteção

Requisitos - Organização Lógica

- Capacidade de manipular com programas e dados do usuário organizado em módulos
  - Memória -> seqüências de Bytes e palavras
  - Programas -> módulos
- Vantagens:
  - Módulo podem ser escritos e compilados independentemente, as referencias de um módulo para o outro são resolvidas em tempo de execução;
  - Com um overhead adicional, diferentes graus de proteção (read only, execute only) podem ser dados para diferentes módulos
  - É possível introduzir mecanismo de compartilhamento entre módulos

Requisito – Organização Física

- Organização da memória do computador:
  - Dois níveis:
    - Memória principal -> mais rápida, volátil e custo alto RAM
    - Memória secundária -> lenta, armazenamento permanente e barata – HDD ou SSD
- Capacidade de mover informações entre os 2 níveis de memória

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, usando memória RAM 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 (monoprogramação)
- monitor residente (monoprogramação)
- swapping (monoprogramação)
- partições múltiplas
- paginação
- ~~segmentação~~
- ~~sistemas combinados paginação e segmentação~~ ~~(muito complexos)~~

Monoprogramação

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)

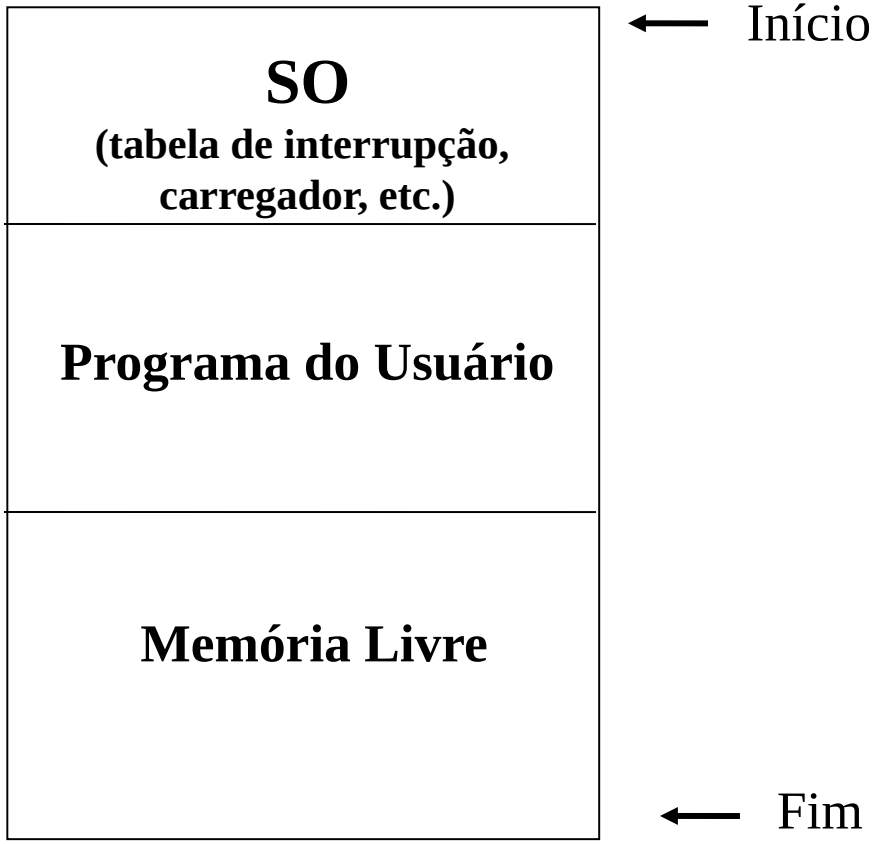
Máquina Pura

- 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 – ex: MS-DOS

- 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**

Alocação Contígua - Overlays

- Programas de usuário limitados pelo tamanho da memória principal disponível.
- - Solução: Overlay
  - Programador divide programa em módulos;
  - Permitir execução independente de cada módulo, usando a mesma área de memória;
- Área de Overlay
  - Área de memória comum onde módulos compartilham mesmo espaço.

Alocação Contígua - Overlays

- Exemplo de Overlay
- Um programa que consista em um módulo principal e outros dois módulos independentes, um correspondente ao cadastro e outro, à impressão. Quando o programa é carregado, apenas o módulo principal é introduzido na memória, os demais aguardam em memória secundária. Quando um dos outros módulos for referenciado pelo módulo principal, aquele será carregado na memória principal, na área de overlay

Alocação Contígua - Overlays

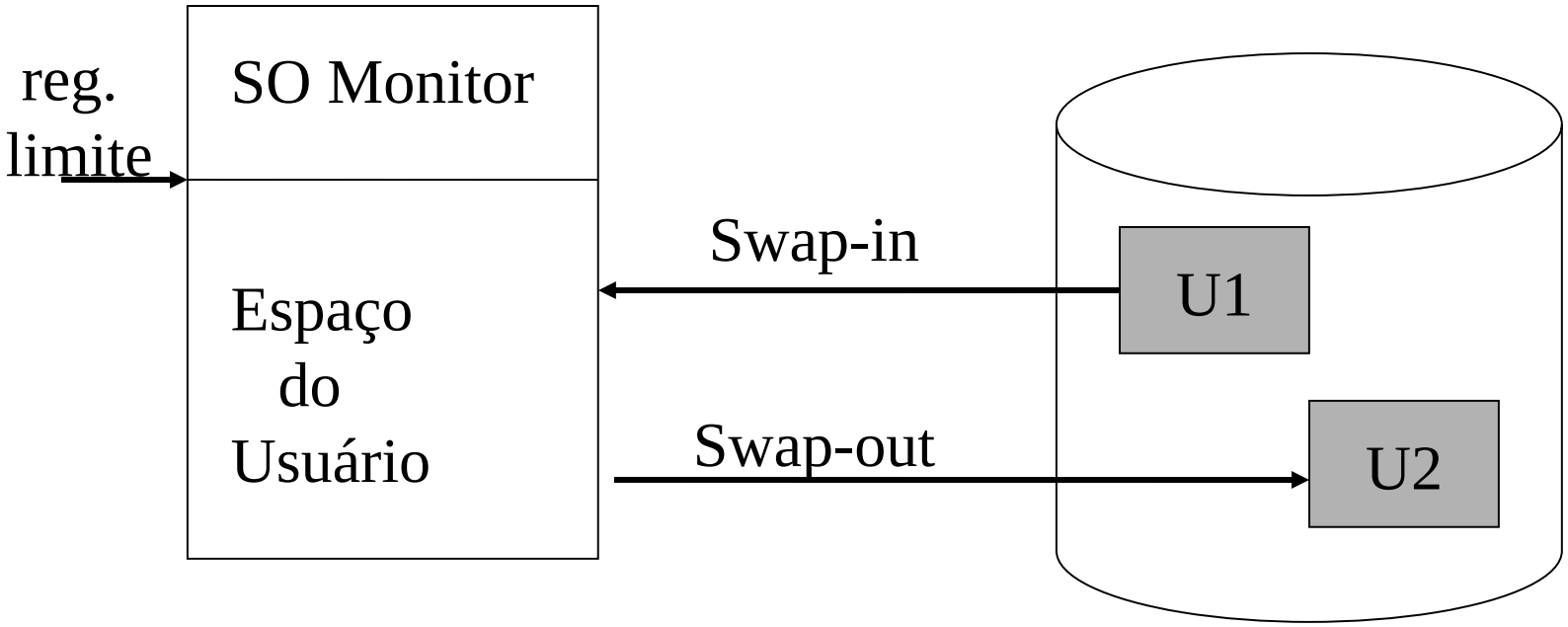
- A definição das áreas de overlay é feita pelo próprio programador, através de comandos específicos da linguagem de programação.
- O tamanho da área de overlay deve ser igual ou maior ao tamanho do maior módulo.
- Como vantagem, a técnica de overlay possibilita um melhor aproveitamento da memória principal.
- Sua utilização exige cuidado, pois pode trazer implicações tanto na manutenção como no desempenho das aplicações, devido à possibilidade de transferências excessivas dos módulos entre a memória principal e a memória secundária.

Multiprogramação

Multiprogramação através de Swapping

- É implementada por um SO do tipo monitor residente. Ex: MS-DOS, DR-DOS
- 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
- O programador não precisa interferir

Swapping


reg.
limite
SO Monitor
Espaço
do
Usuário
Swap-in
Swap-out
U1
U2

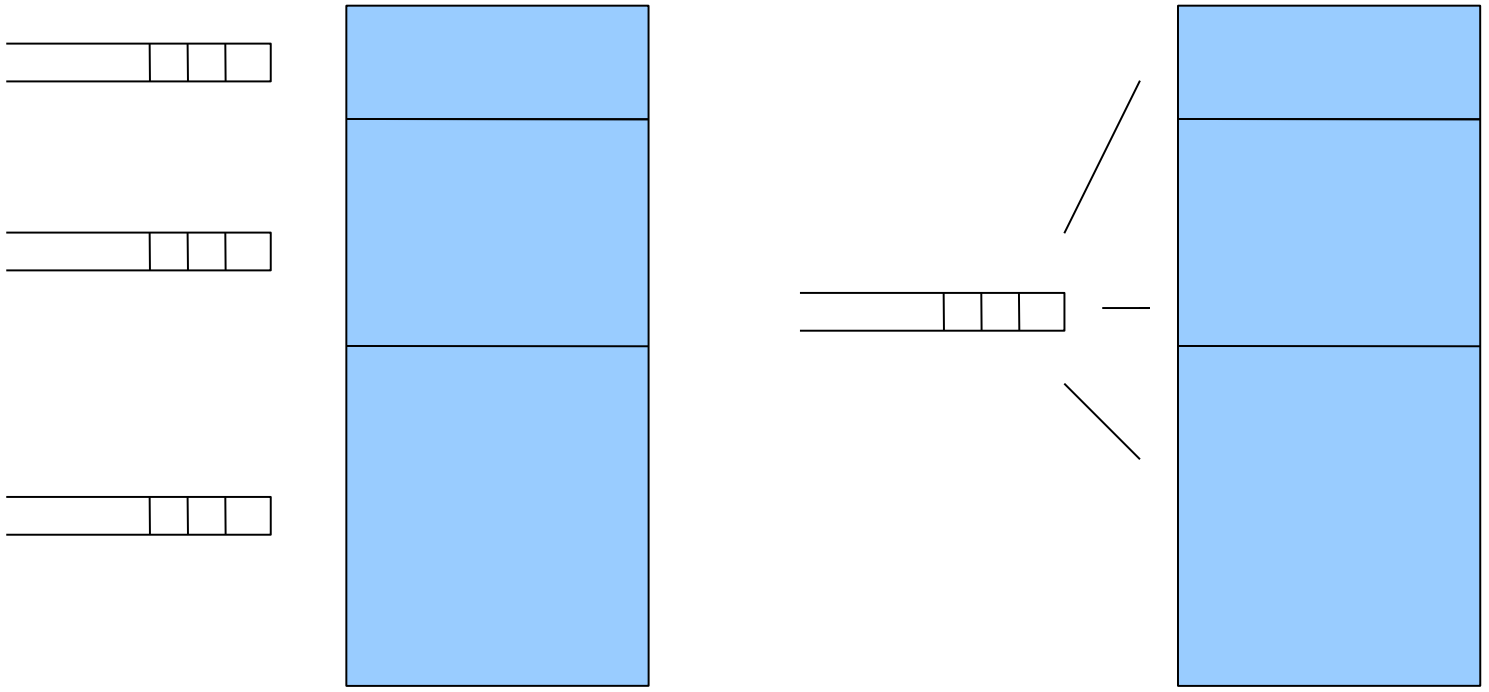
Partições Múltiplas

- 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 (baseado em Jobs)

Particionamento fixo – tamanho variável



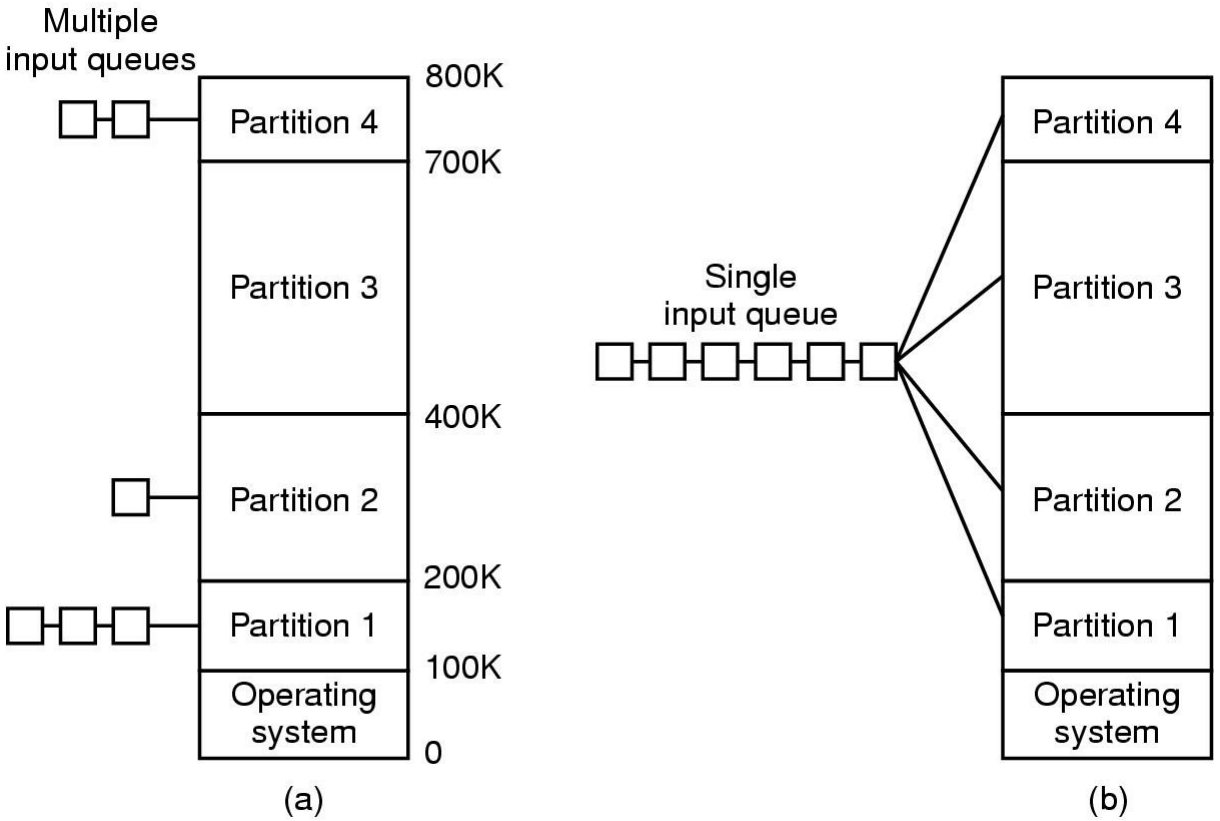
Partições Fixas

- 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

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

Particionamento fixo – tamanho variável



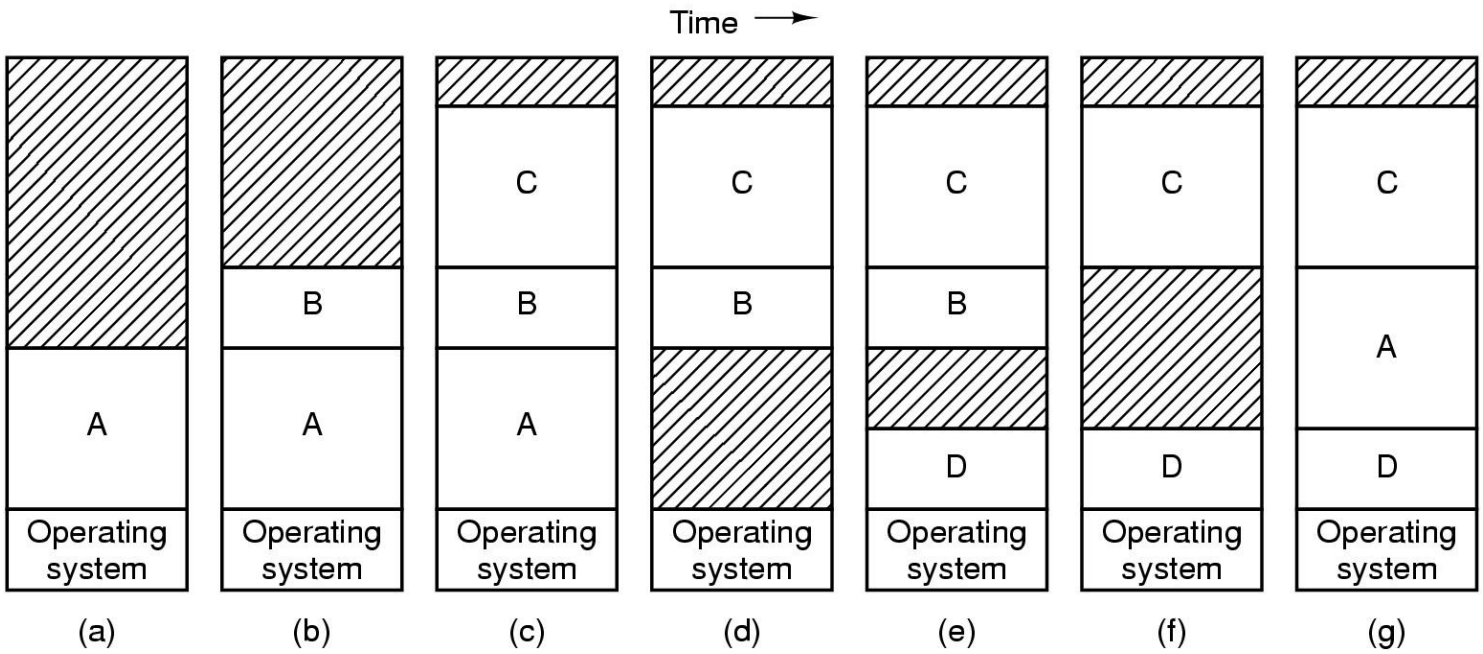
Partições Variáveis

- 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

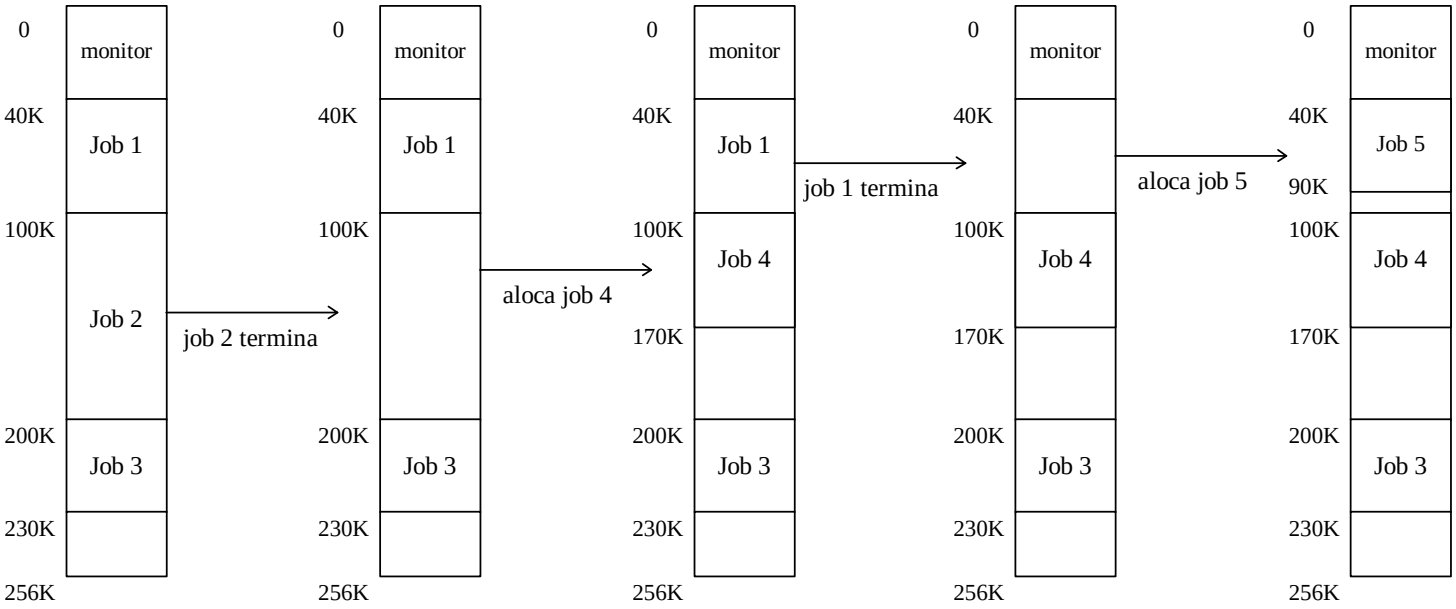
- O SO mantém uma lista indicando quais partes da memória estão disponíveis e quais estão ocupadas.
- As áreas disponíveis são denominadas **lacunas** (***holes***)
- Quando um processo chega para ser executado, a lista de lacunas é consultada e é escolhida uma lacuna de tamanho suficiente

Particionamento dinâmico



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

Fragmentação

- São perdas (desperdício) de memória devido a fragilidades do mecanismo de gerenciamento de memória

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) – partições fixas
  - **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 variáveis

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)
  - necessidade de relocação dinâmica (pois a relocação estática impossibilita a compactação)

Paginação

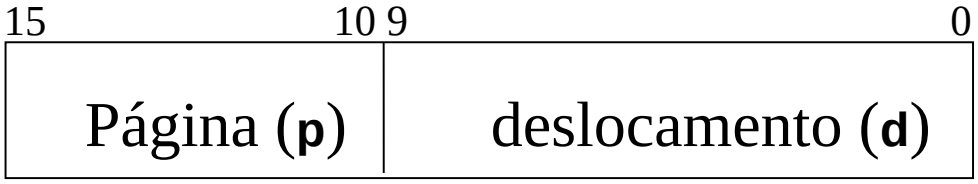
Paginação

- 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
- Será abordado com mais detalhes depois

Paginação

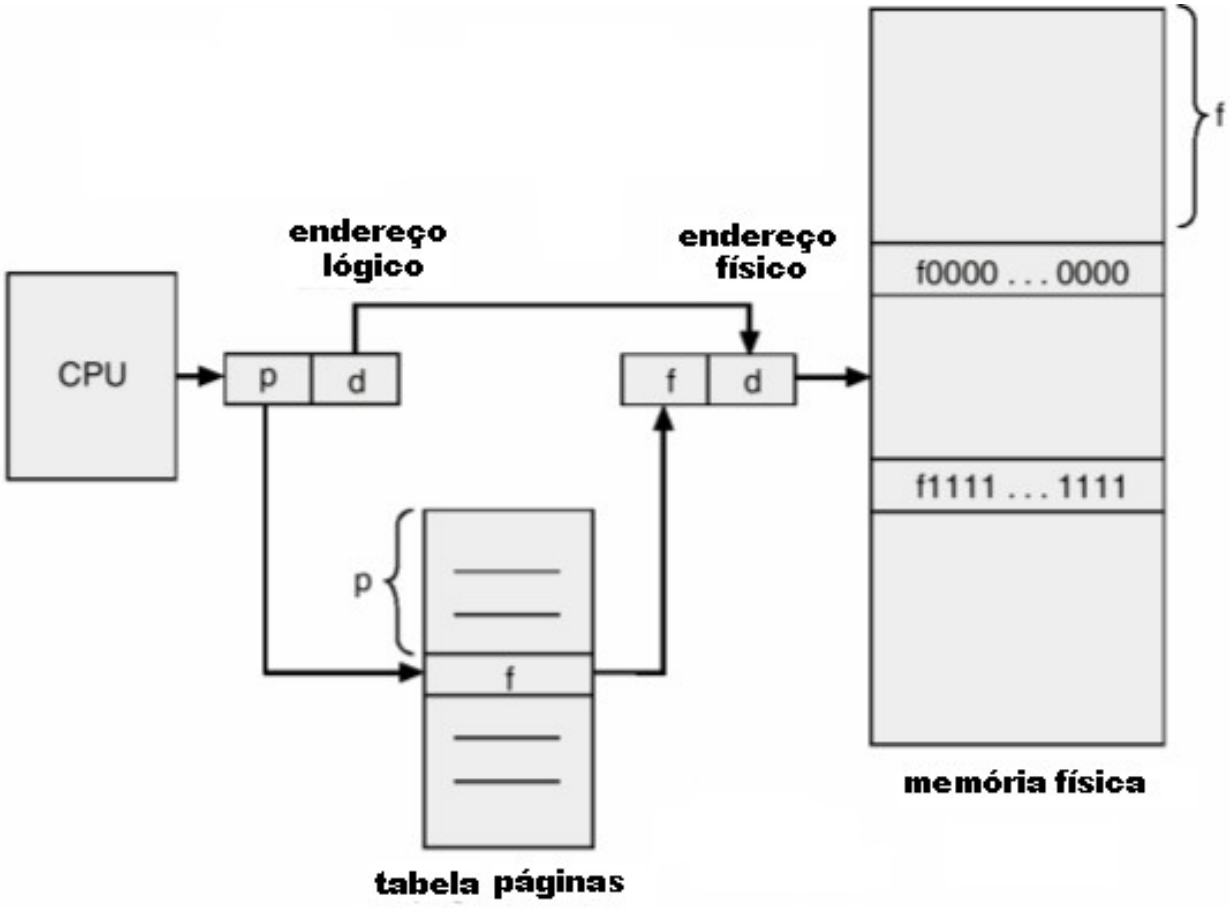
- Se um processo tem tamanho K, os seus endereços lógicos (endereços especificados nas suas instruções) vão desde 0 até K-1. Este é o **espaço de endereçamento** do processo.
- Cada endereço lógico é quebrado em duas partes:
  - número de página **p**
  - deslocamento **d**

- Endereço lógico:

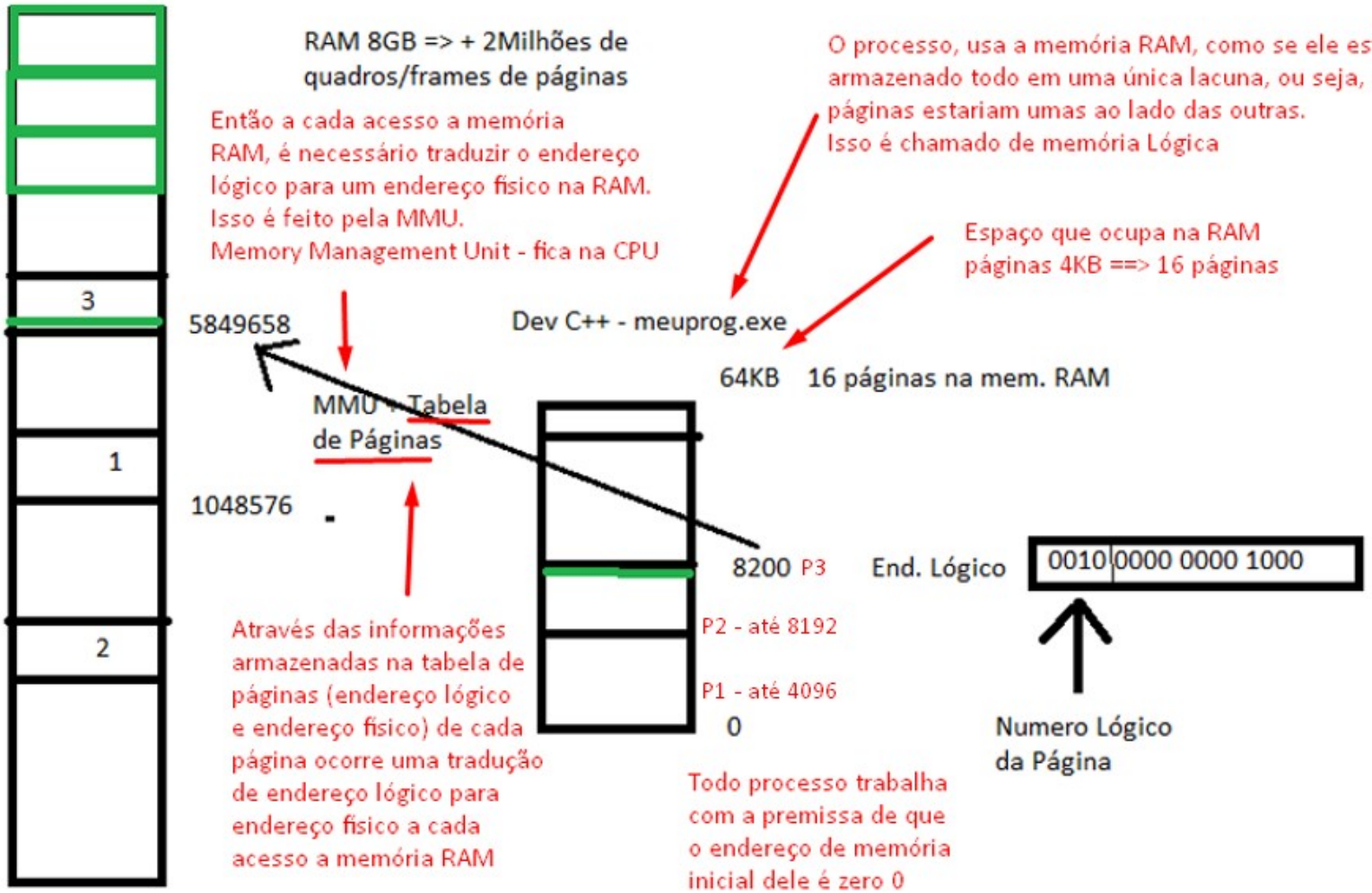


- Acontece **relocação dinâmica**, pois cada endereço lógico é traduzido em endereço físico em tempo de execução

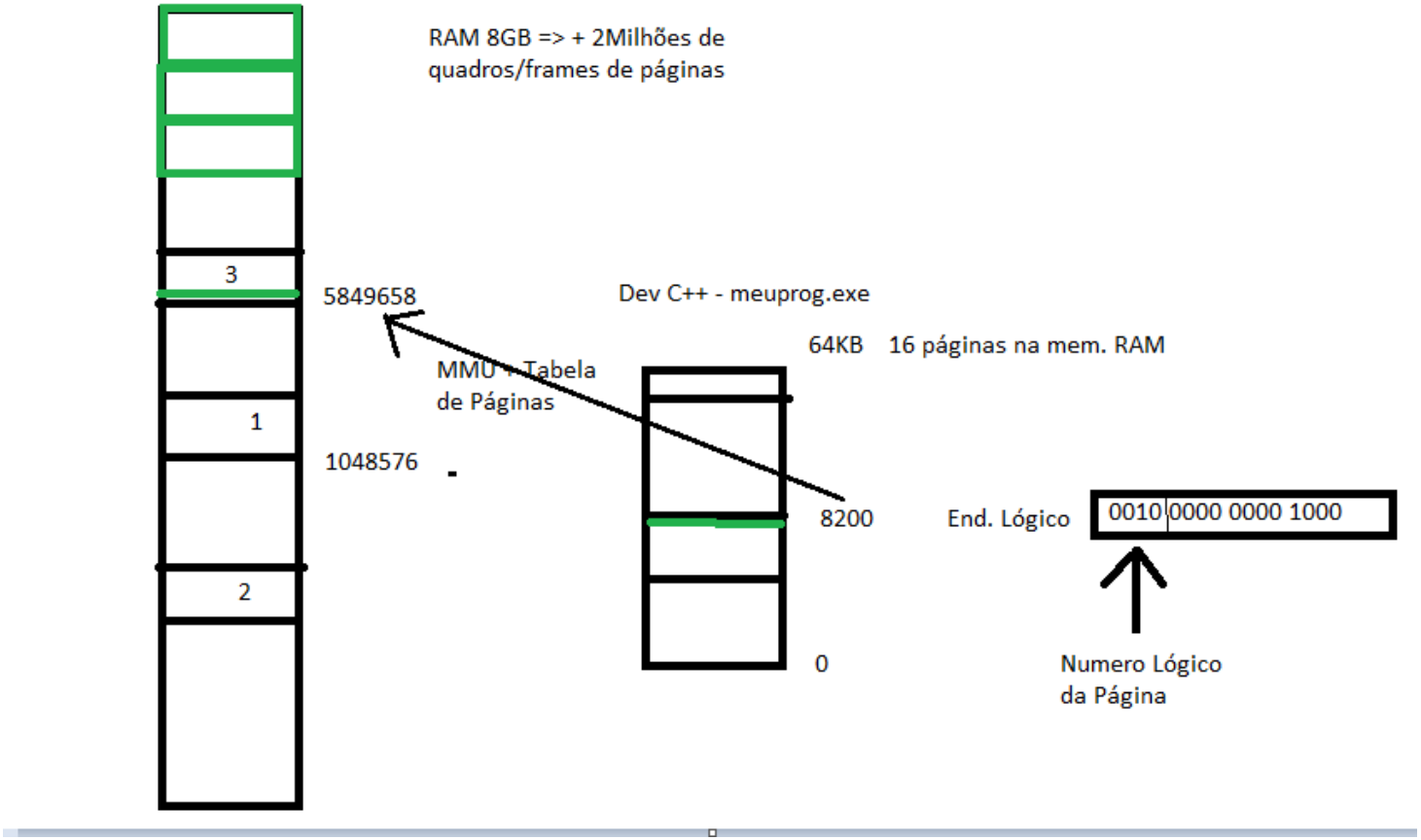
Paginação


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

Paginação


RAM 8GB => + 2Milhões de quadros/frames de páginas
Então a cada acesso a memória RAM, é necessário traduzir o endereço lógico para um endereço físico na RAM. Isso é feito pela MMU. Memory Management Unit - fica na CPU
O processo, usa a memória RAM, como se ele estivesse armazenado todo em uma única lacuna, ou seja, as 16 páginas estariam umas ao lado das outras. Isso é chamado de memória Lógica
Espaço que ocupa na RAM páginas 4KB ==> 16 páginas
3
5849658
Dev C++ - meuprog.exe
64KB 16 páginas na mem. RAM
MMU + Tabela de Páginas
1
1048576
8200 P3
End. Lógico
0010 0000 0000 1000
P2 - até 8192
P1 - até 4096
0
2
Através das informações armazenadas na tabela de páginas (endereço lógico e endereço físico) de cada página ocorre uma tradução de endereço lógico para endereço físico a cada acesso a memória RAM
Todo processo trabalha com a premissa de que o endereço de memória inicial dele é zero 0
Numero Lógico da Página

Paginação


RAM 8GB => + 2Milhões de
quadros/frames de páginas
3
5849658
1
1048576
2
Dev C++ - meuprog.exe
64KB 16 páginas na mem. RAM
MMU + Tabela
de Páginas
8200
0
End. Lógico
0010 0000 0000 1000
Numero Lógico
da Página

Implementação da Tabela de Páginas

- Memory Management Unit - MMU
  - Componente interno ao processador
  - Faz a tradução de endereços lógicos para físicos
- Conjunto de registradores dedicados
- Memória Principal
- Memória Associativa - CACHE

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
- Não temos registradores suficientes para armazenar a tabela de páginas de um processo

Tabela de Páginas na Memória Principal

- Cada descritor de processo contém o endereço de sua respectiva tabela de páginas.
- A CPU possui um registrador que aponta para a tabela de páginas atual
  - Para que a MMU (Memory Management Unit) possa encontrá-la quando for necessário converter um endereço lógico para endereço físico
- 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 Associativa de Alta Velocidade
  - Memória Cache
    - Próxima ao processador – mesmo material
- 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** ou melhor se está em RAM ou em disco

Memória Virtual com Paginação

- A memória RAM é dividida em frames para as páginas
- Uma parte de memória secundária (HD/SSD) será reservada para armazenar páginas de memória que não couberem na memória principal - RAM
  - Partição (LINUX/UNIX/MacOS)
  - Arquivo pagefile.sys (Windows)

Memória Virtual

- Na memória RAM teremos somente as páginas que estiverem sendo mais utilizadas pelos programas em execução
- Na partição/arquivo de swap teremos as páginas de processos que não estiverem sendo muito utilizadas

Paginação por demanda

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

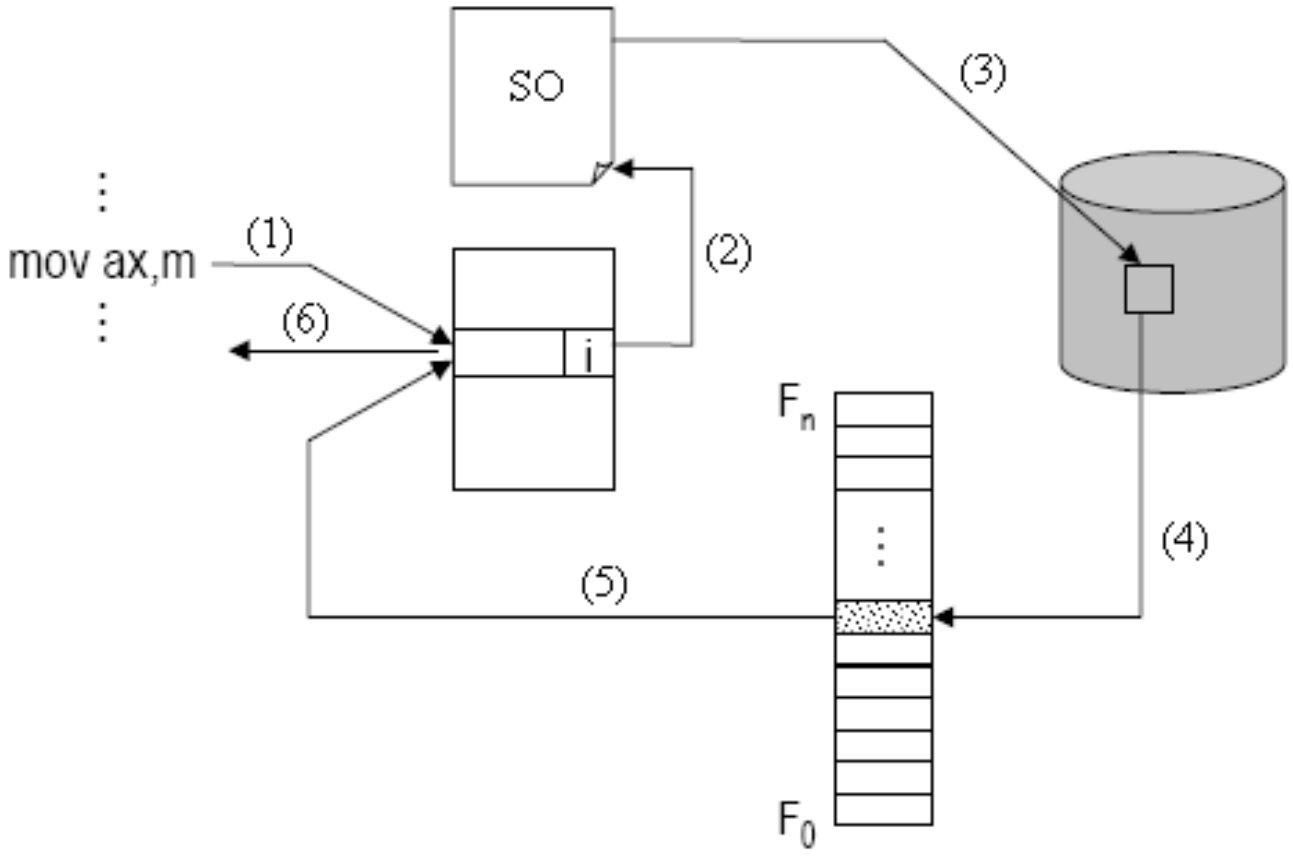
Paginação por demanda

- 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 de execução (bloqueado) e inserido em uma fila especial, a "fila dos processos esperando por carga de página lógica";
- Uma página física (frame) livre deve ser alocada;
  - Se necessário, retirar uma página não tão utilizada da RAM e enviar para disco
- 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.
  - E depois disso o processo será desbloqueado, ficando pronto para execução

Tratamento de *page-fault*

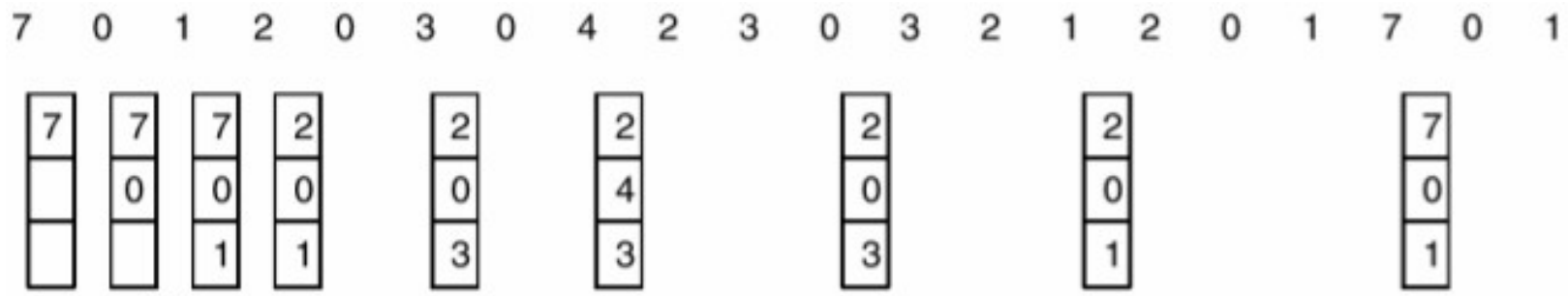


Substituição de Páginas

- O SO deve escolher uma página para ser substituída (**página vítima**)
- Bits auxiliares
  - Bit de sujeira ou modificação (*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 que a página foi acessada recentemente (este bit é desligado pelos algoritmos de substituição de páginas)
  - Bit de trava (*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.

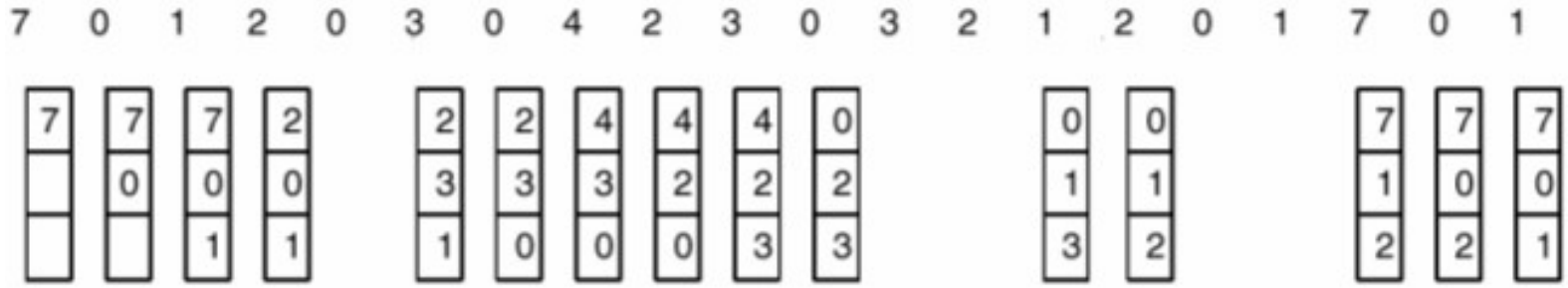
Algoritmos de substituição

- Ótimo (teórico)
  - Substitui a página que será acessada por último, a que tem a menor chance de ser usada



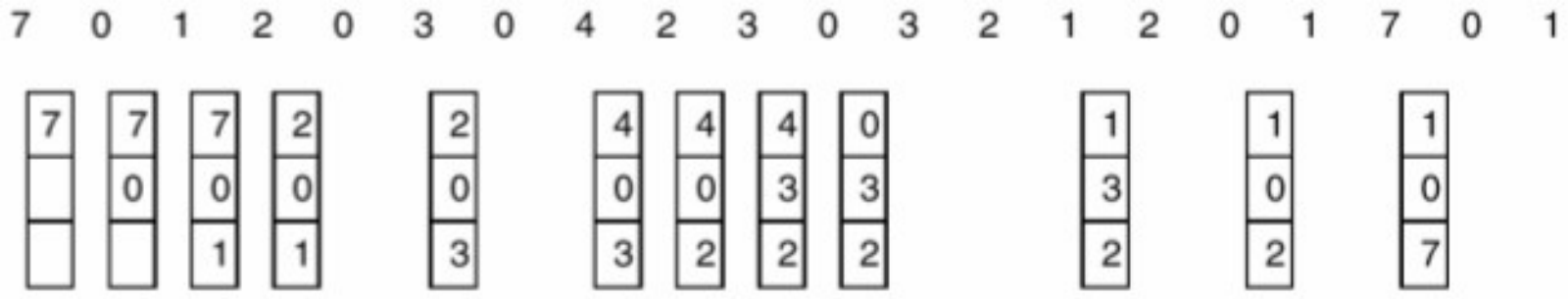
Algoritmos de substituição

- FCFS (ou FIFO)
- First-Come-First-Served ou First-In-First-Out
  - A página mais antiga deve sair.
  - O SO deve armazenar a ordem ou um timestamp em que as páginas são/foram trazidas para a memória



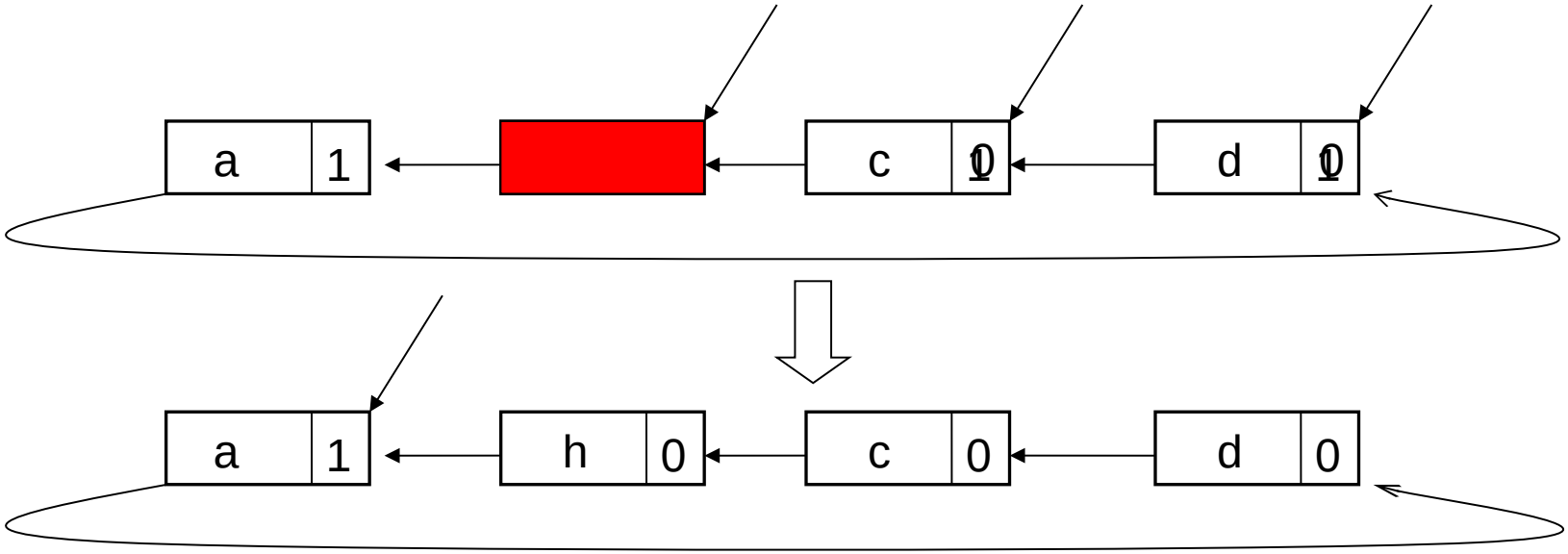
Algoritmos de substituição

- LRU
  - Least Recently Used ou Menos Usada Recentemente
  - A página usada há mais tempo sai



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)
    - Ex: x% das páginas de cada processo em memória RAM
    - Depende da quantidade de RAM disponível para definir x

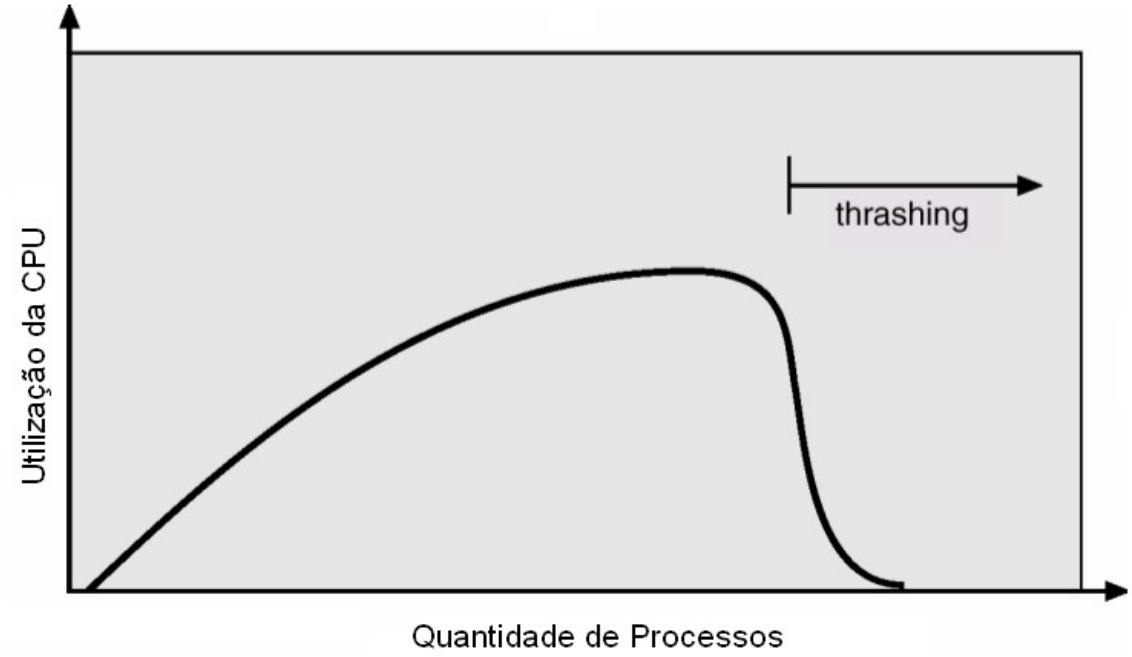
Alocação de Páginas Físicas

- Alocação global
  - Não distingue entre os processos na hora de escolher uma página vítima
  - Fato: processos de baixa prioridade ou que estão sendo pouco utilizados podem ser prejudicados (páginas enviadas para armazenamento secundário)

Thrashing

- Se um processo tem poucos frames de página para operar, 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.