UFRJ – IM - DCC

Sistemas Operacionais I

Unidade III – Memória Primária

ORGANIZAÇÃO DA UNIDADE

- Processador - Escalonamento
- Memória Primária
  - Fundamentos
  - Formas de Particionamento
  - Swapping
  - Técnicas de Overlay
- Memória Virtual

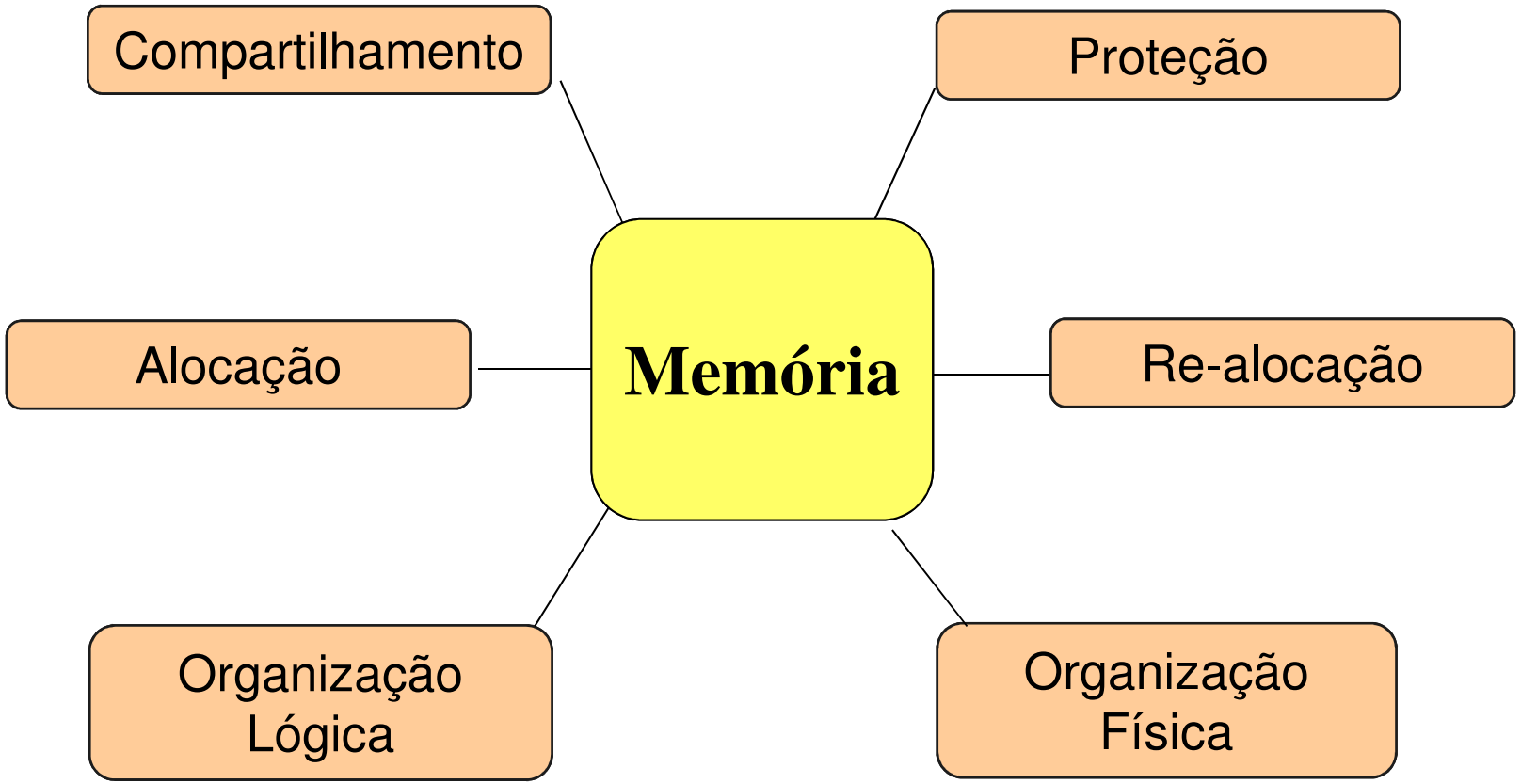
Gerenciar a memória consiste na tarefa de subdividir e alocar espaços para acomodar os processos em execução.

- ***Espaços são requeridos (alocados) e liberados, a medida que os processos são executados.***

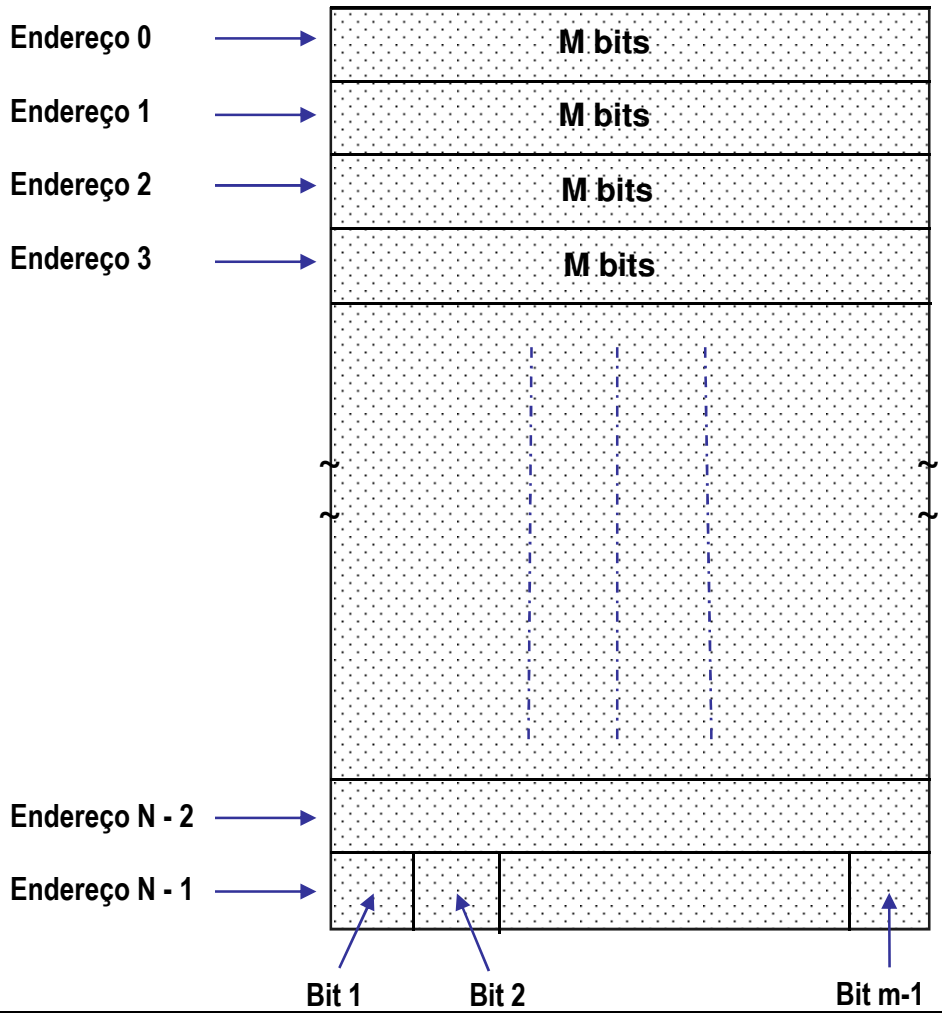
- ***Os espaços de memória ocupados pelos processos precisam ser preservados (protegidos).***

- ***Os processos podem ter a necessidade de aumentar o espaço ocupado ou mesmo compartilhar espaço com outros.***

Organização e Funcionalidades



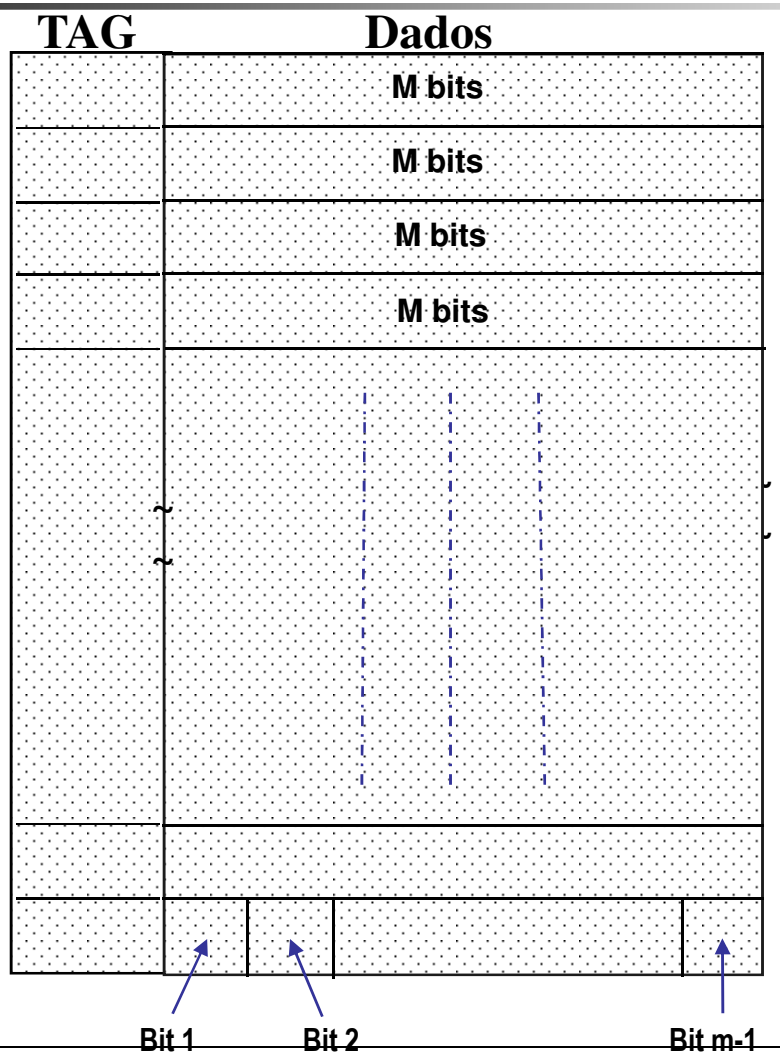
Organização Física



Memória de Acesso Direto

Todas as células têm a mesma quantidade M de bits

Organização Física



Memória de Acesso Associativo

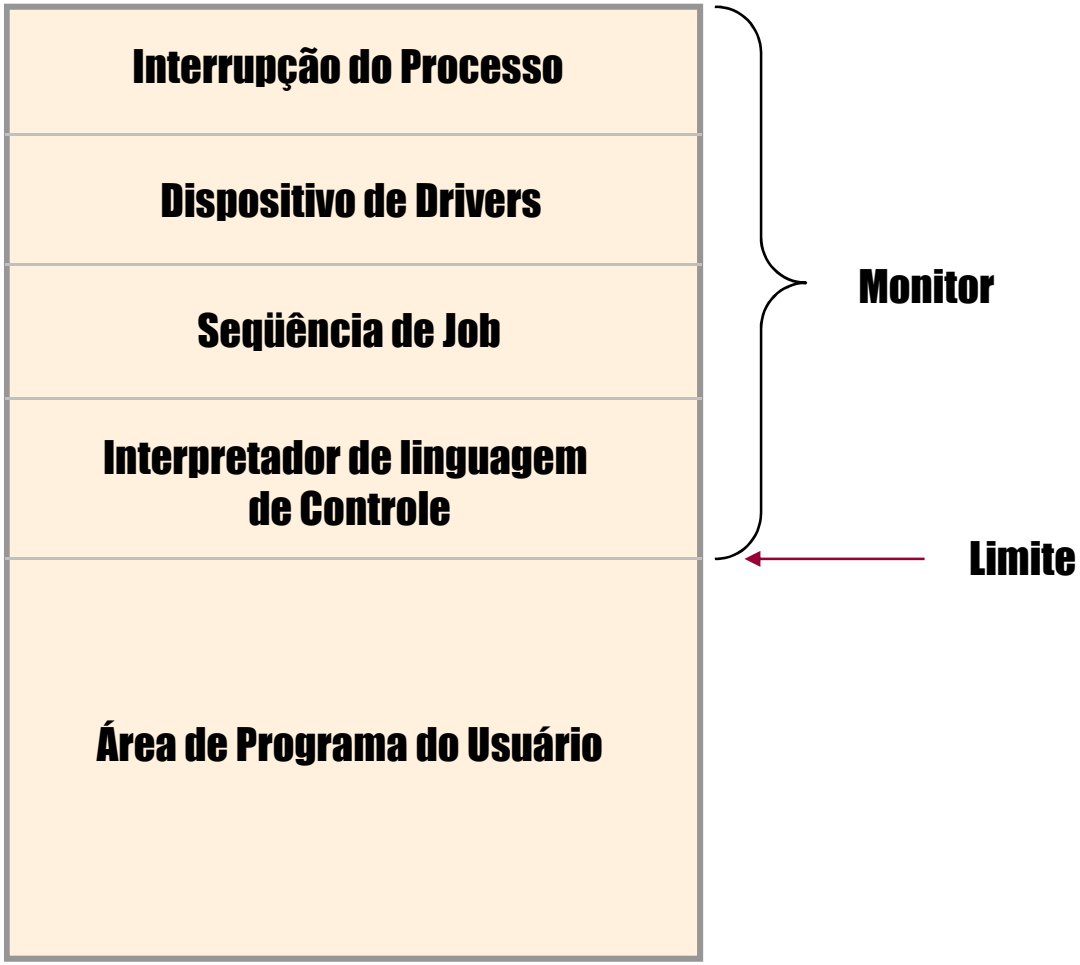
Organização Lógica

Consiste na forma como a memória é vista (particionada) logicamente pelo SO.

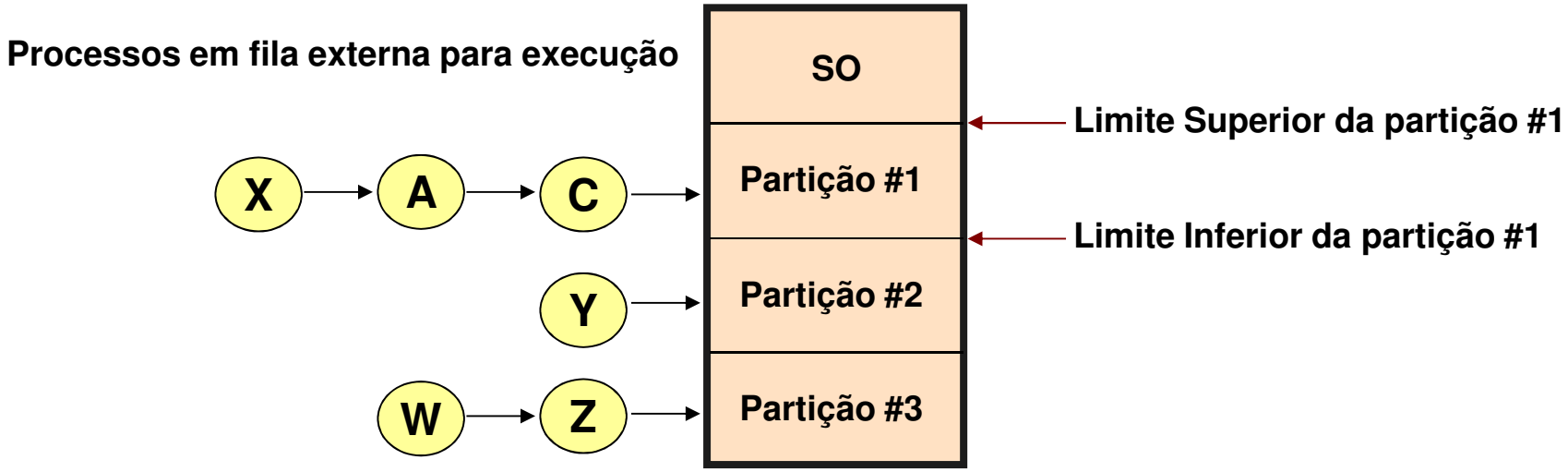
Formas de Particionamento:

- ***Contíguo Simples***
- ***Estático***
- ***Estático Relocável***
- ***Dinâmico***
- ***Paginado***
- ***Segmentado***

Particionamento Contíguo Simples

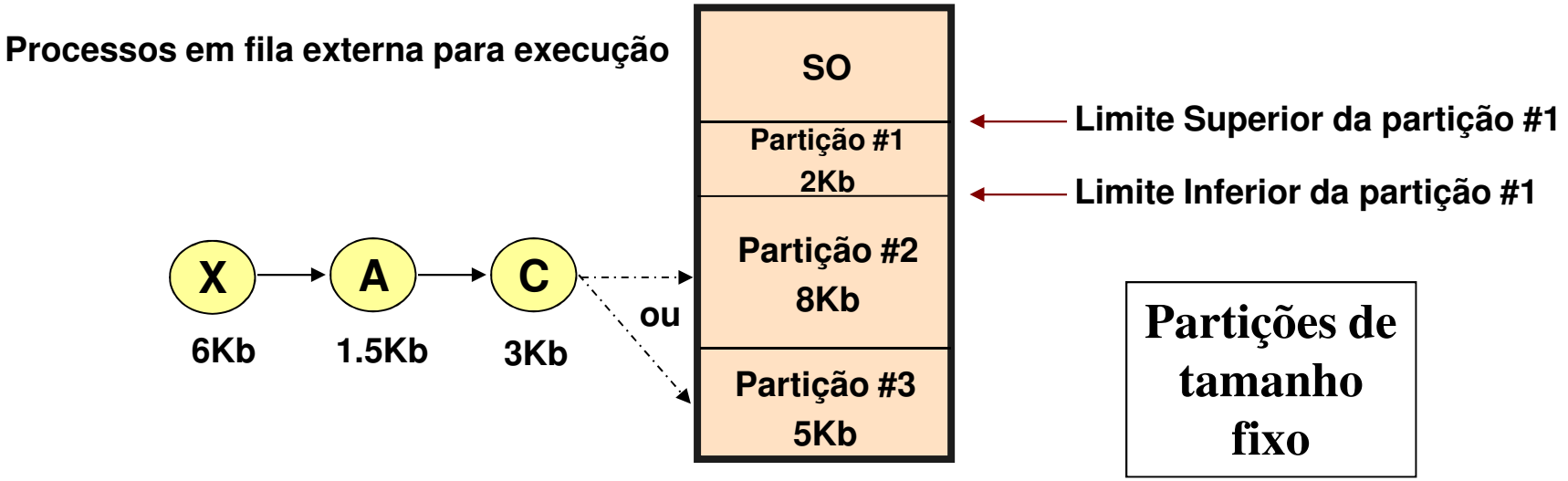


Particionamento Estático



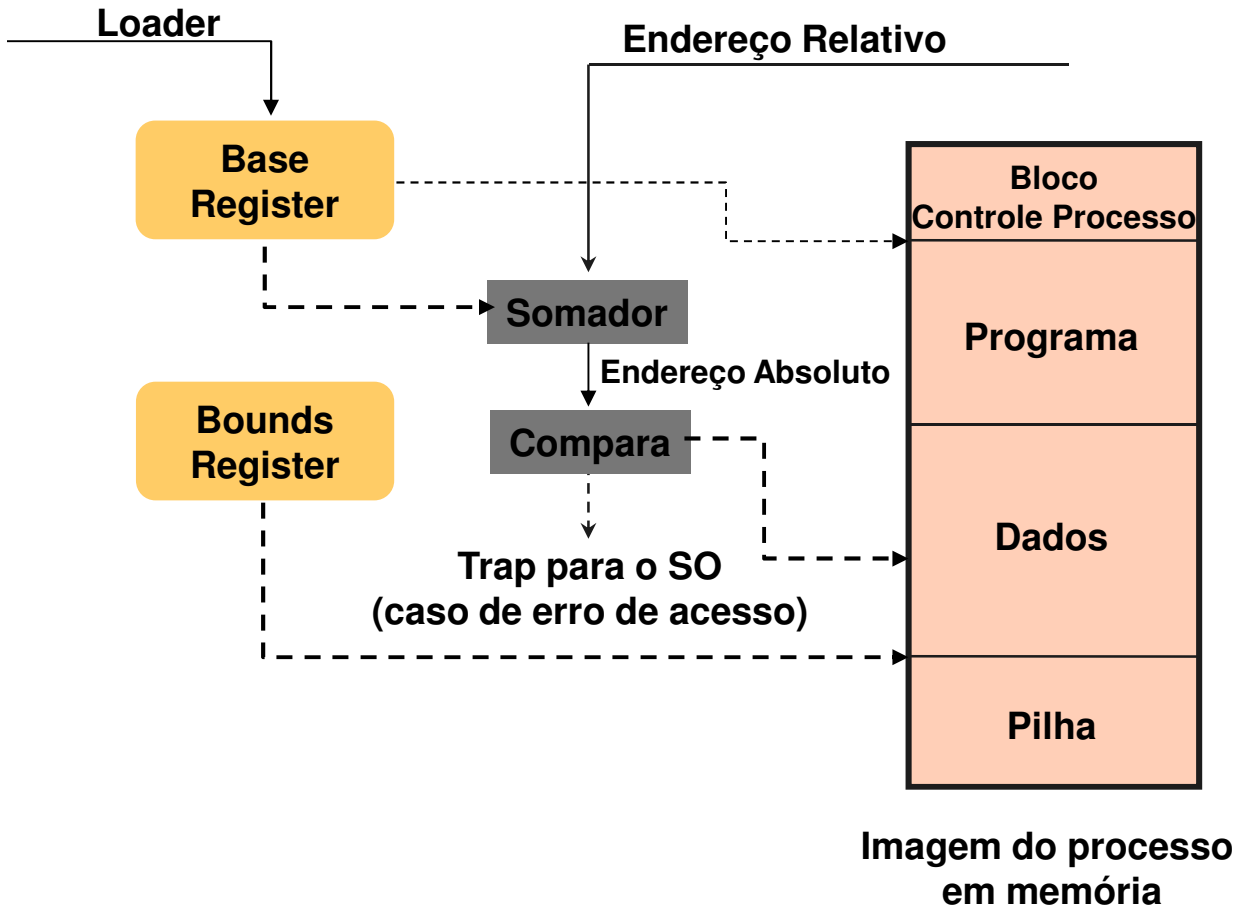
- Endereços absolutos são gerados de forma estática - Linkeditor ou Carregador
- Simples implementação
- Baixo desempenho
- Fragmentação interna

Particionamento Estático Relocável

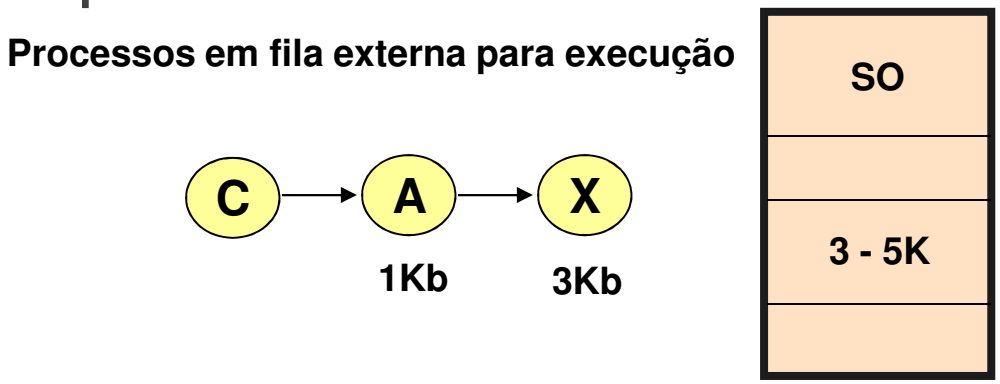


- Endereçamento relativo
- Endereços absolutos dinâmicos, calculados em tempo de execução
- Baixo desempenho
- Fragmentação interna

Endereço Relativo x Absoluto

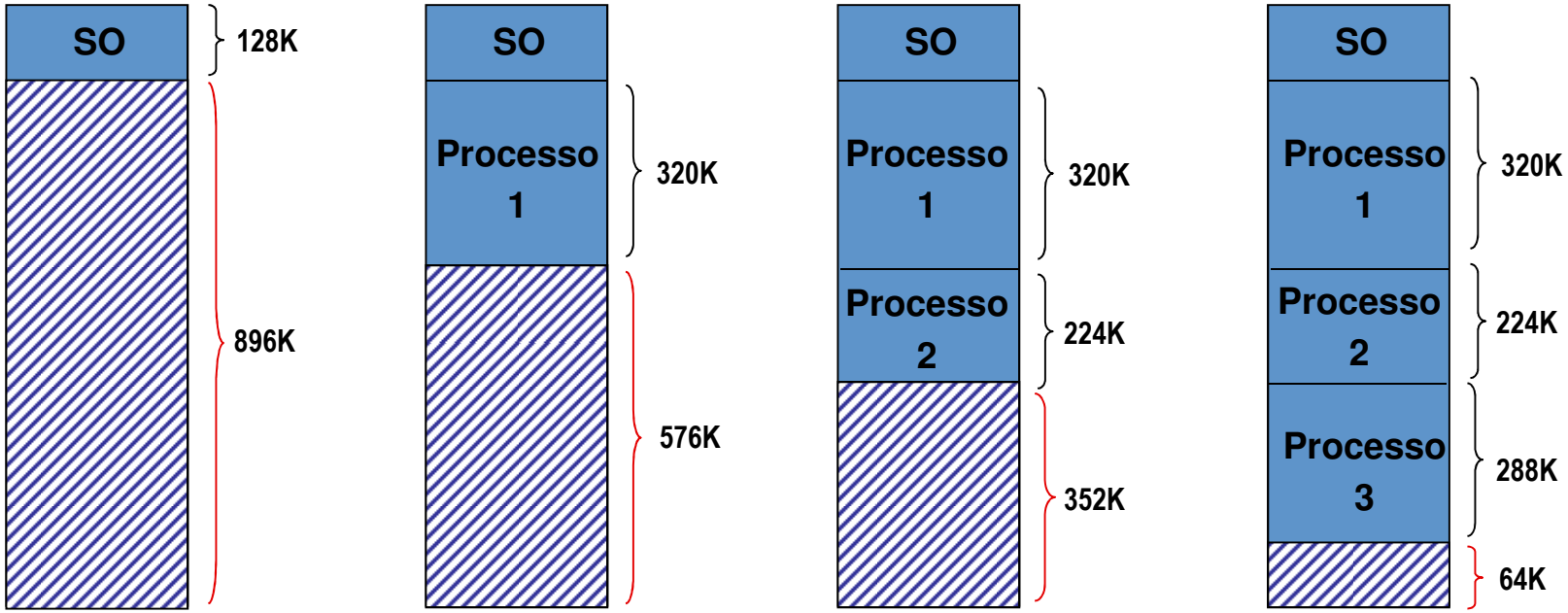


Particionamento Dinâmico



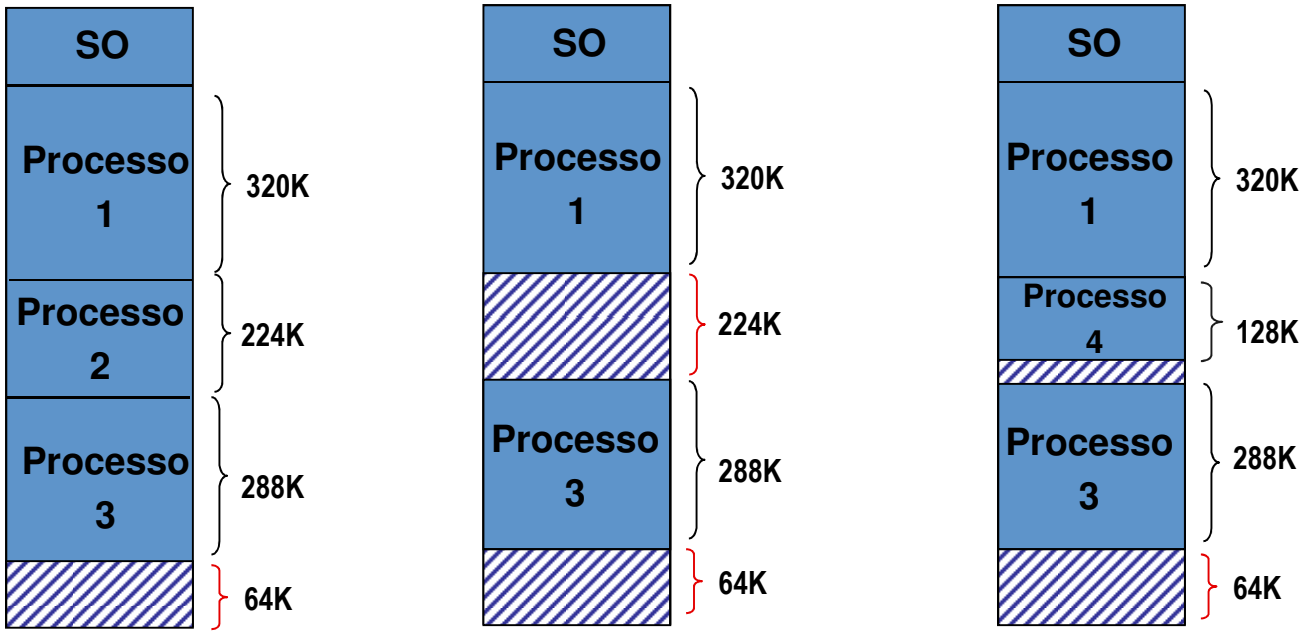
- A quantidade e o tamanho das partições são variáveis
- Para cada processo, é alocado o espaço exato que for necessário (não tem fragmentação interna)
- Eventualmente, são criados "buracos" de tamanho pequeno, sem utilidade de uso (fragmentação externa)
- Os processos, de tempos em tempos, precisam ser re-alocados para eliminar os "buracos" (compactação)

Particionamento Dinâmico - Exemplo



- **Após carregar os três processos, um espaço de 64K fica disponível. Assim, não sobra espaço suficiente para um Processo 4 de 128K.**
- **Eventualmente, todos os três processos poderão estar bloqueados.**
- **Neste caso, o SO pode retirar o Processo 2 (o mais adequado) para trazer o Processo 4 (= 128K)**

Particionamento Dinâmico - Exemplo ...



(*) Colcha de retalhos – quando a memória fica repleta de pequenos espaços ociosos espalhados de forma não contígua.

Esquemas de Alocação

O S.O. deve escolher o bloco de memória livre onde será colocado o processo

Algoritmo First-Fit

- Procura a partir da memória o primeiro bloco livre que sirva
- Pode criar muitos pequenos blocos livres no início da memória
- Considerado o algoritmo mais rápido

Algoritmo Worst-Fit

- Escolhe o maior bloco livre a partir início da memória.
- Tende deixar buracos maiores que o Best-fit

Algoritmo Next-Fit

- Escolhe o próximo bloco livre a partir da última alocação em que caiba o processo
- Tende a acabar com o grande bloco livre no final da memória

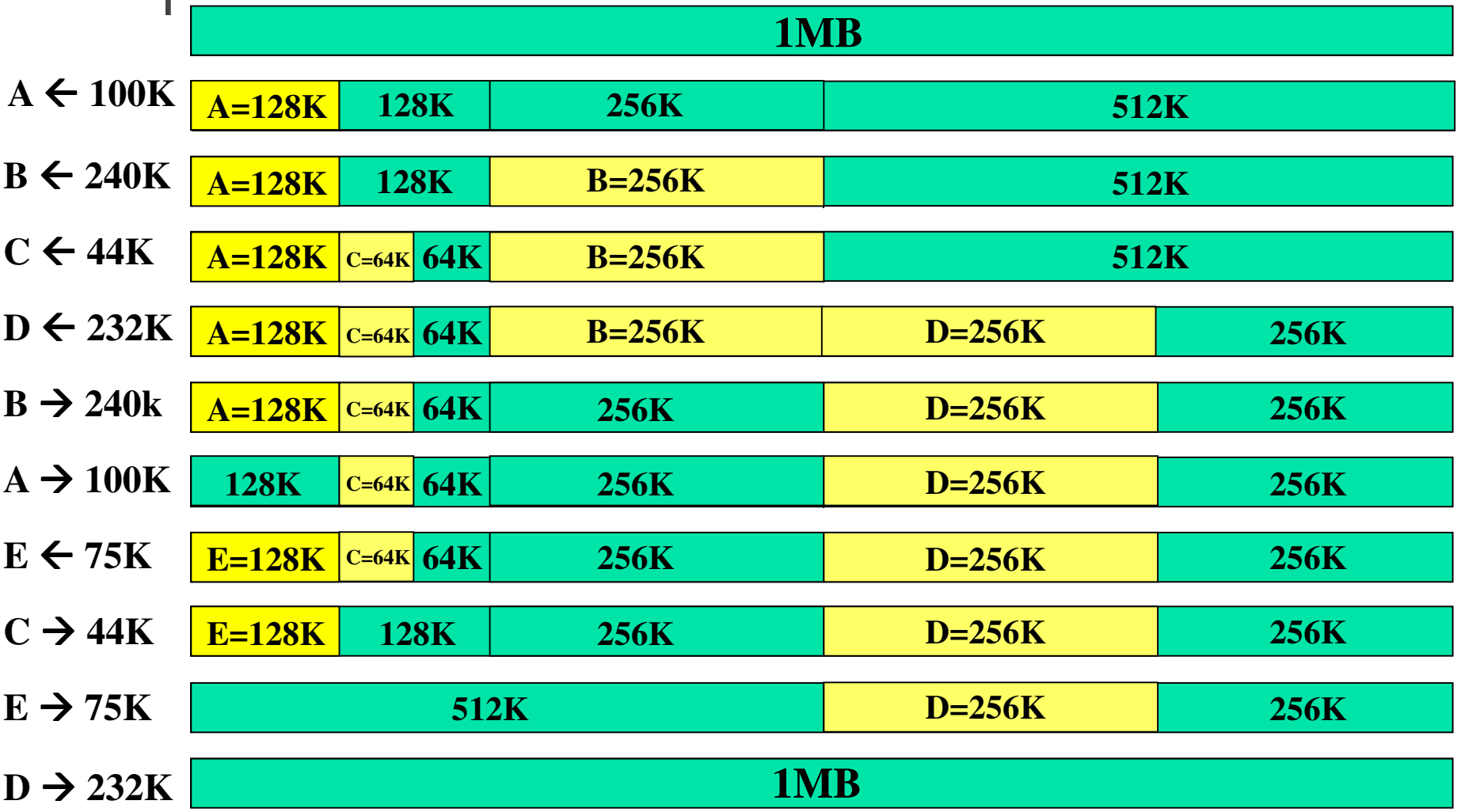
Algoritmo Best-Fit

- Escolhe o menor bloco que comporte o processo
- Cria muitos buracos pequenos, exigindo mais compactações
- Oferece o pior desempenho

Particionamento Dinâmico - Buddy System

- a memória é organizada em blocos de 2^k;
- $L \leq K \leq U$
- 2^L – menor bloco que pode ser alocado
- 2^U – maior bloco (tamanho da memória)
- reduz problema da colcha de retalhos (re-agrupamento)
- insere fragmentação interna

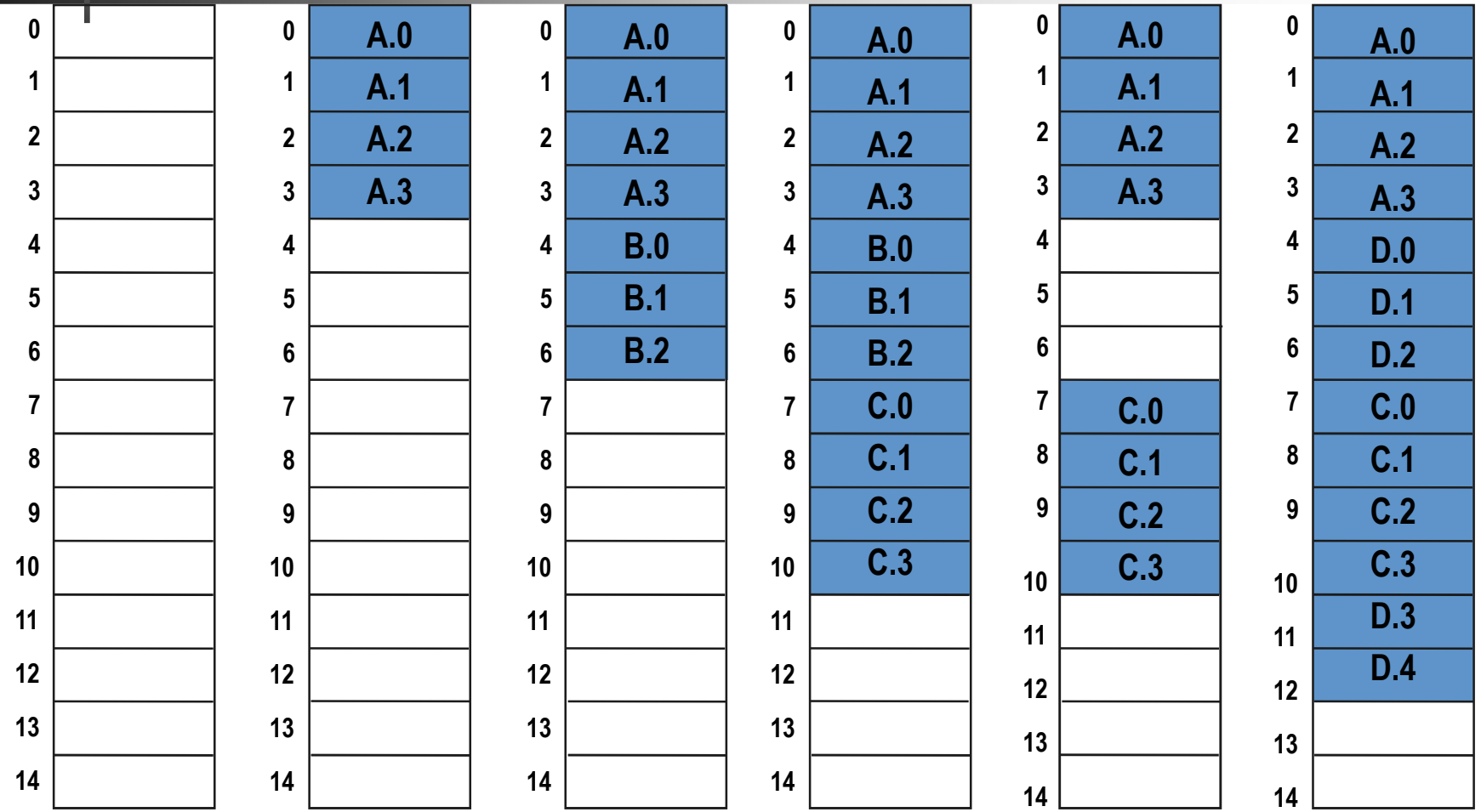
Particionamento Dinâmico - Buddy System



Particionamento Paginado

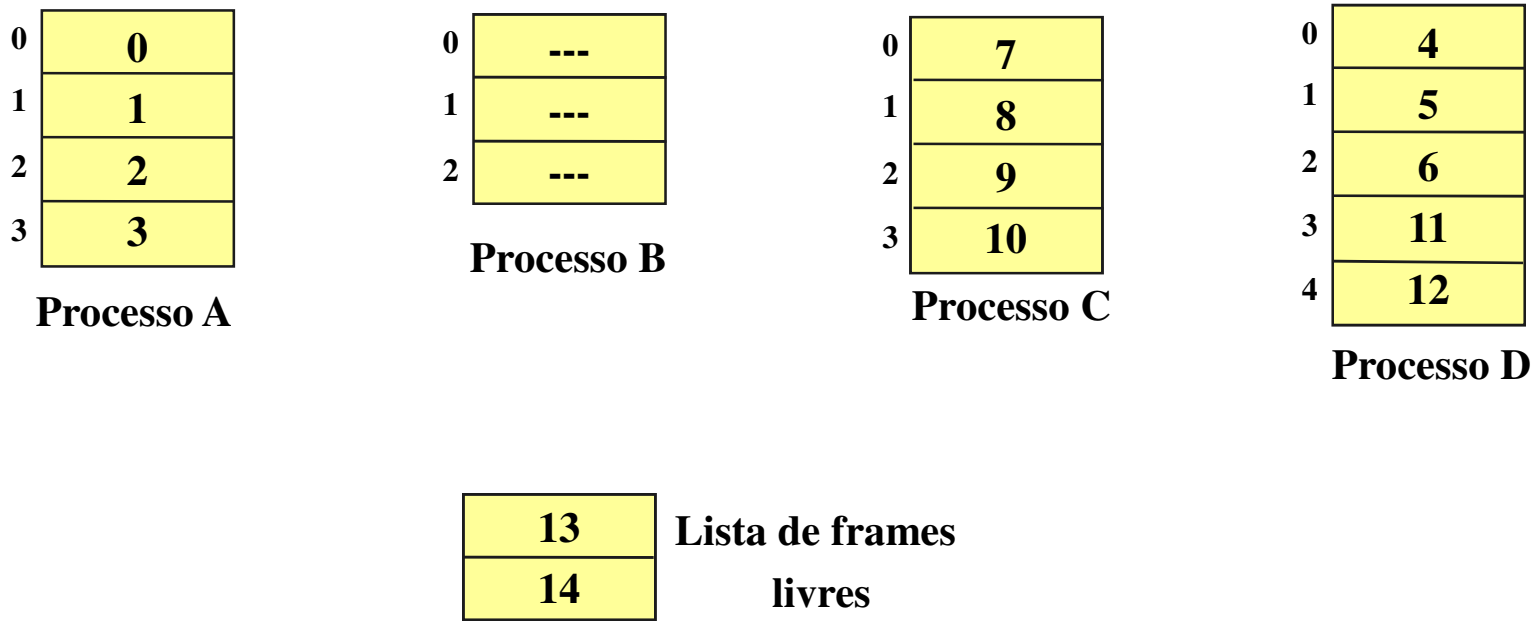
- A memória é particionada em pedaços de tamanho igual, assim como os processos;
- Os pedaços que compõem os processos são chamados de páginas e os pedaços de memória são as molduras de página (frames);
- Quando um processo é carregado, suas páginas são alocadas em quaisquer molduras disponíveis, não necessariamente contíguas;
- O S.O. precisa manter uma tabela de páginas por processo e uma lista de molduras disponíveis.

Particionamento Paginado - Exemplo



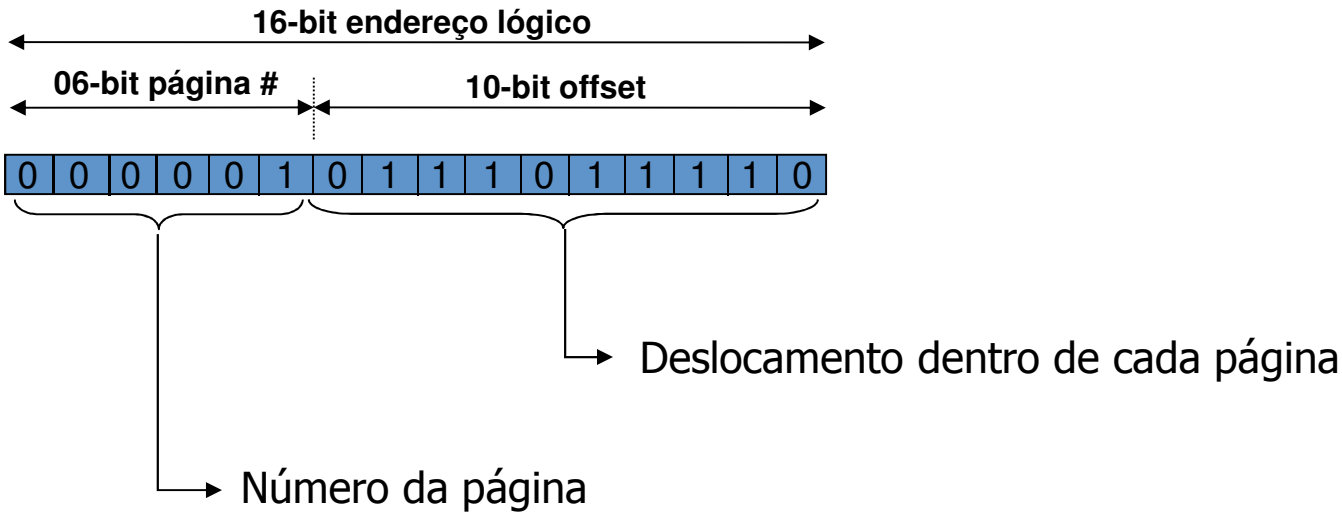
Tabelas de Páginas

Precisa ser mantida uma para cada processo, de forma a associar a página do processo com o frame correspondente em memória utilizado.



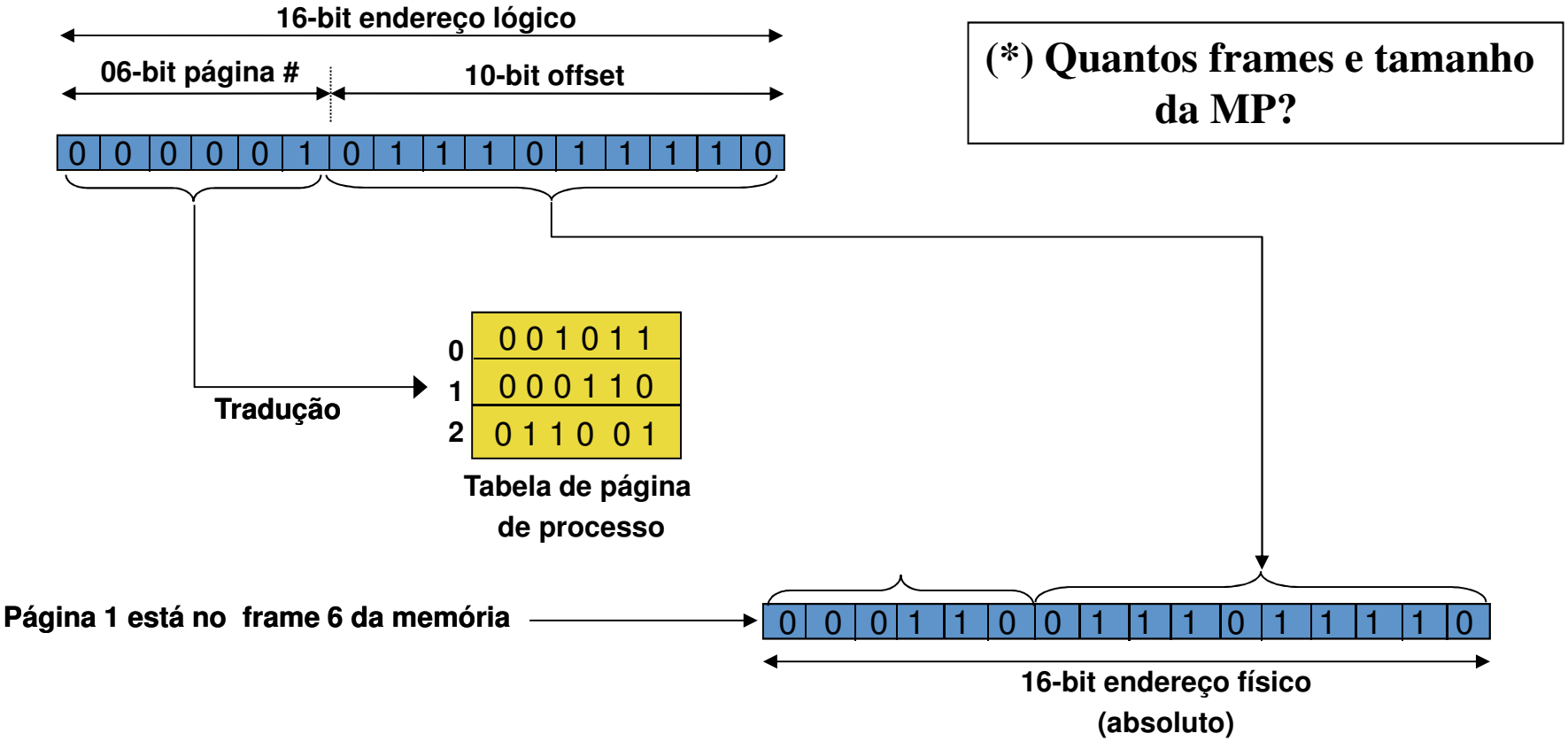
Endereço Lógico

Exemplo:



Tradução Endereço Lógico → Físico

Tradução em Paginação

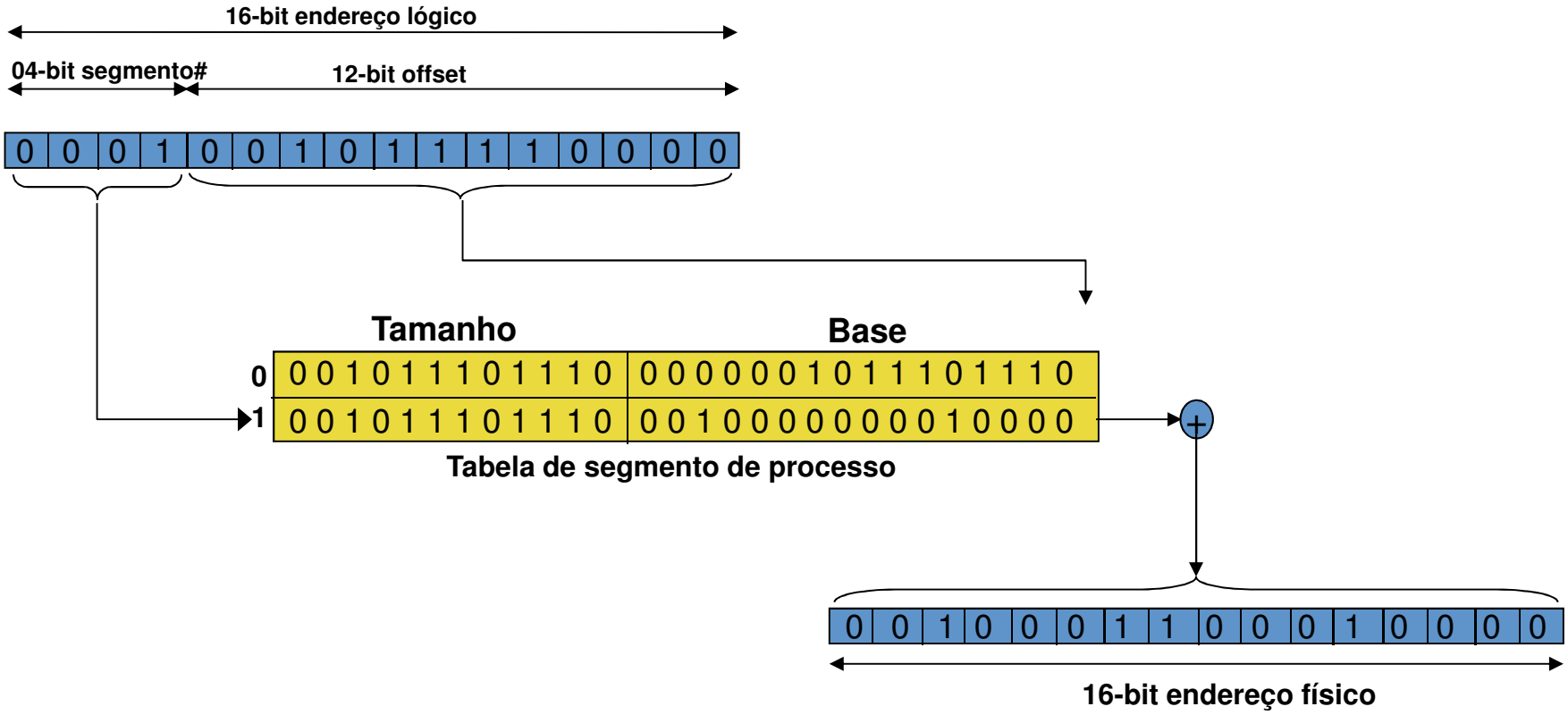


Particionamento Segmentado

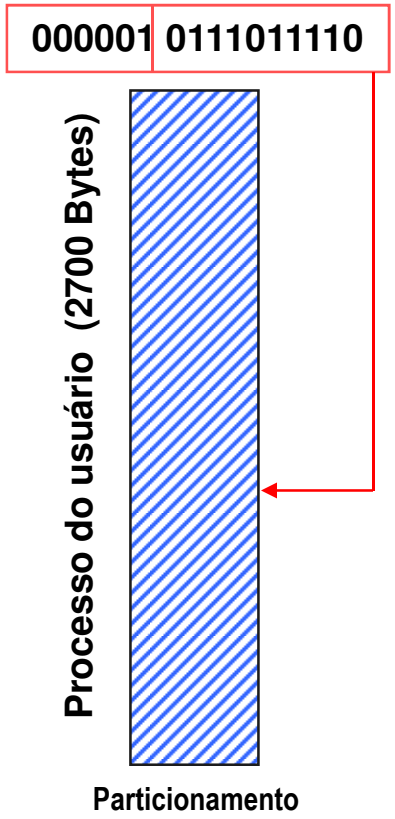
- Cada programa é subdividido em blocos de diferentes tamanhos, chamados segmentos.
- Quando um processo é carregado para a memória principal, cada segmento diferente pode ocupar qualquer lugar.
- O SO mantém uma tabela de segmentos de cada processo. Cada entrada contém:
  - o início do endereço físico daquele segmento
  - o tamanho do segmento (por proteção)
- Apresenta fragmentação externa

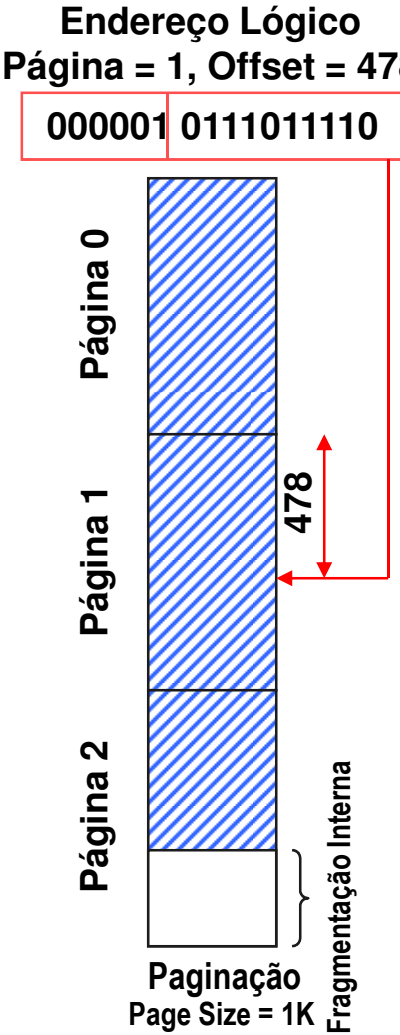
Tradução Endereço Lógico → Físico

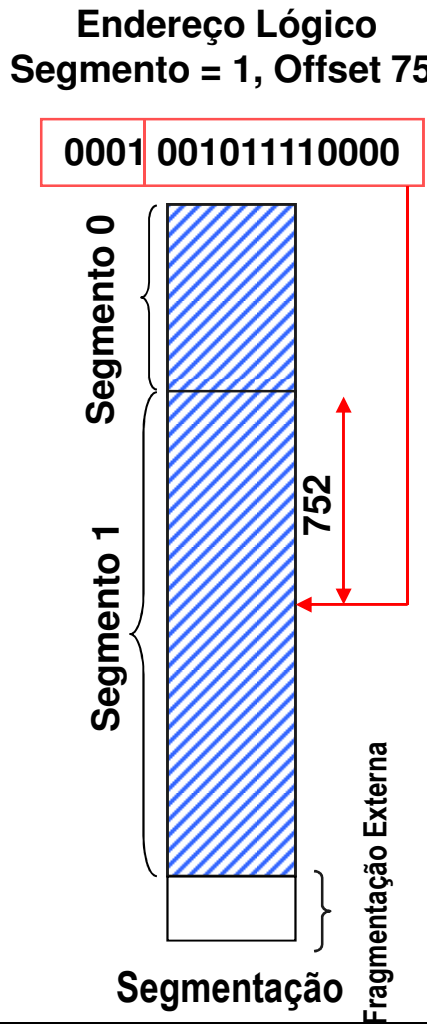
Tradução em Segmentação



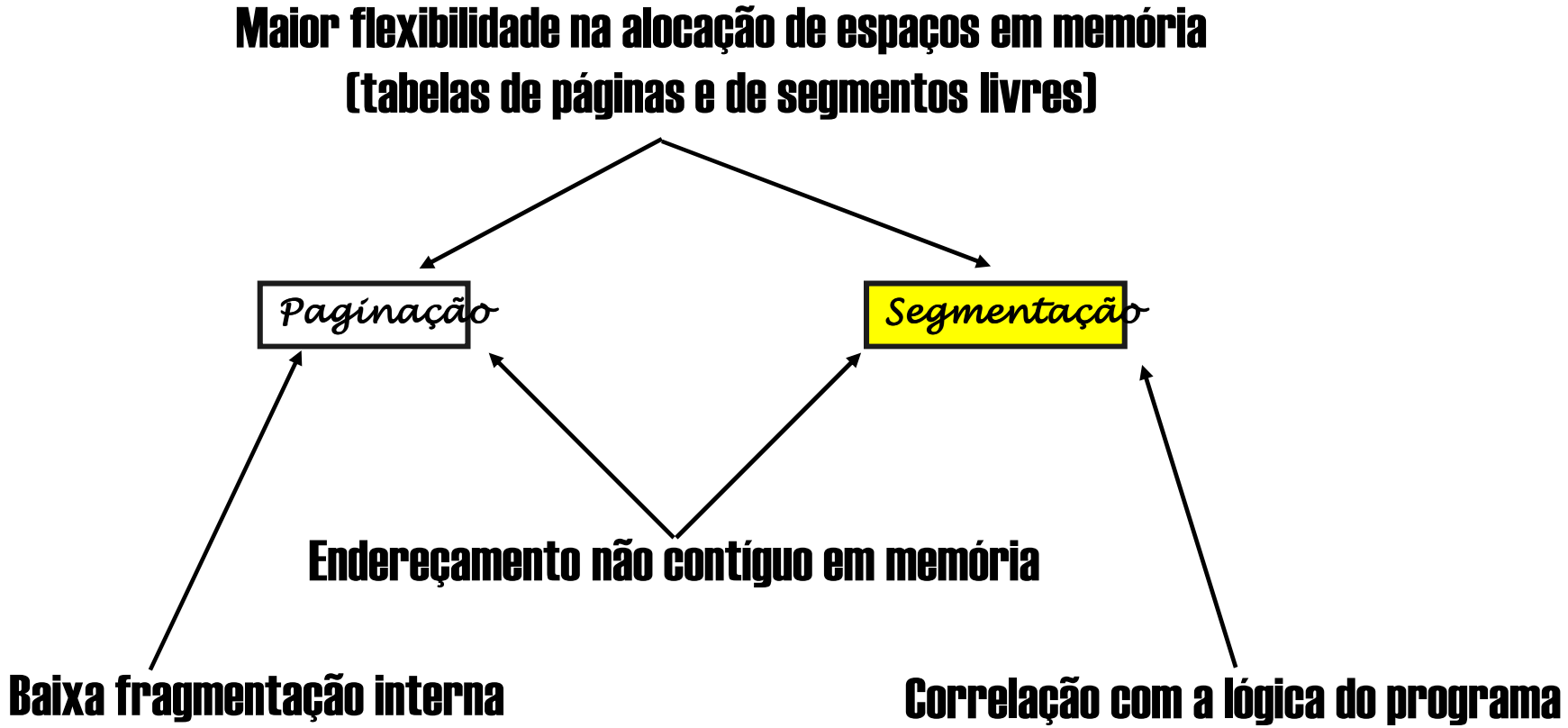
Tradução de Endereços



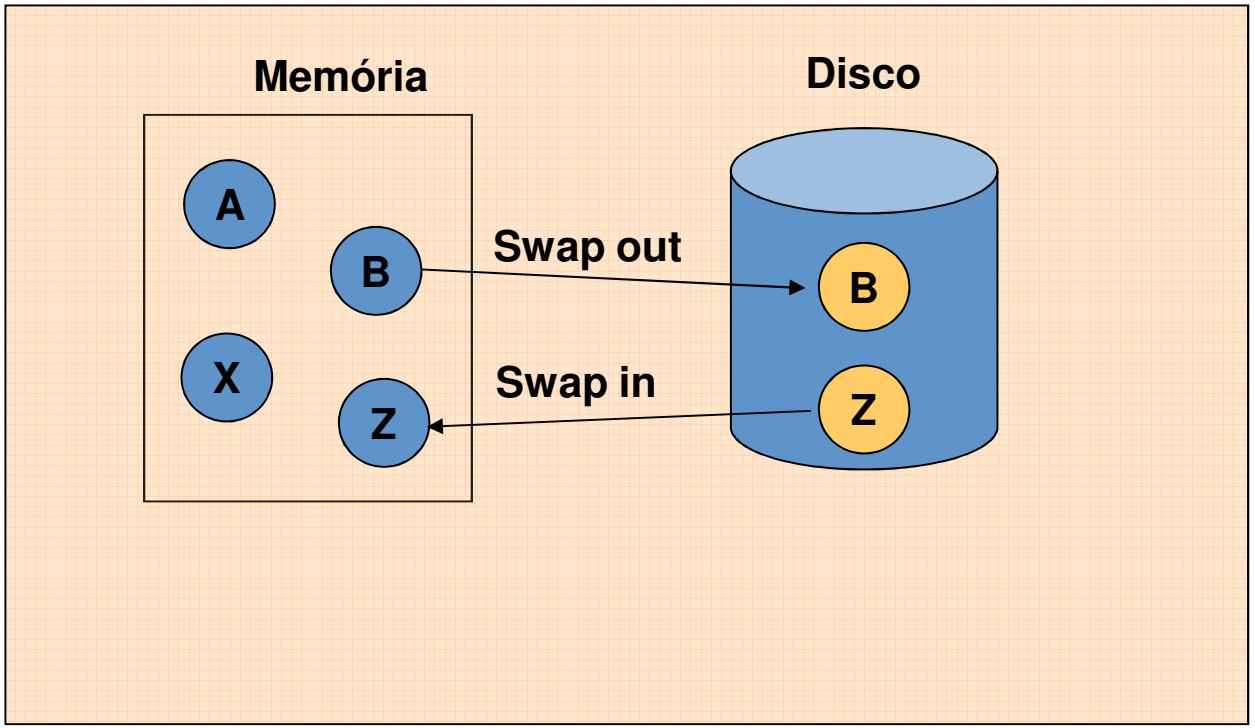




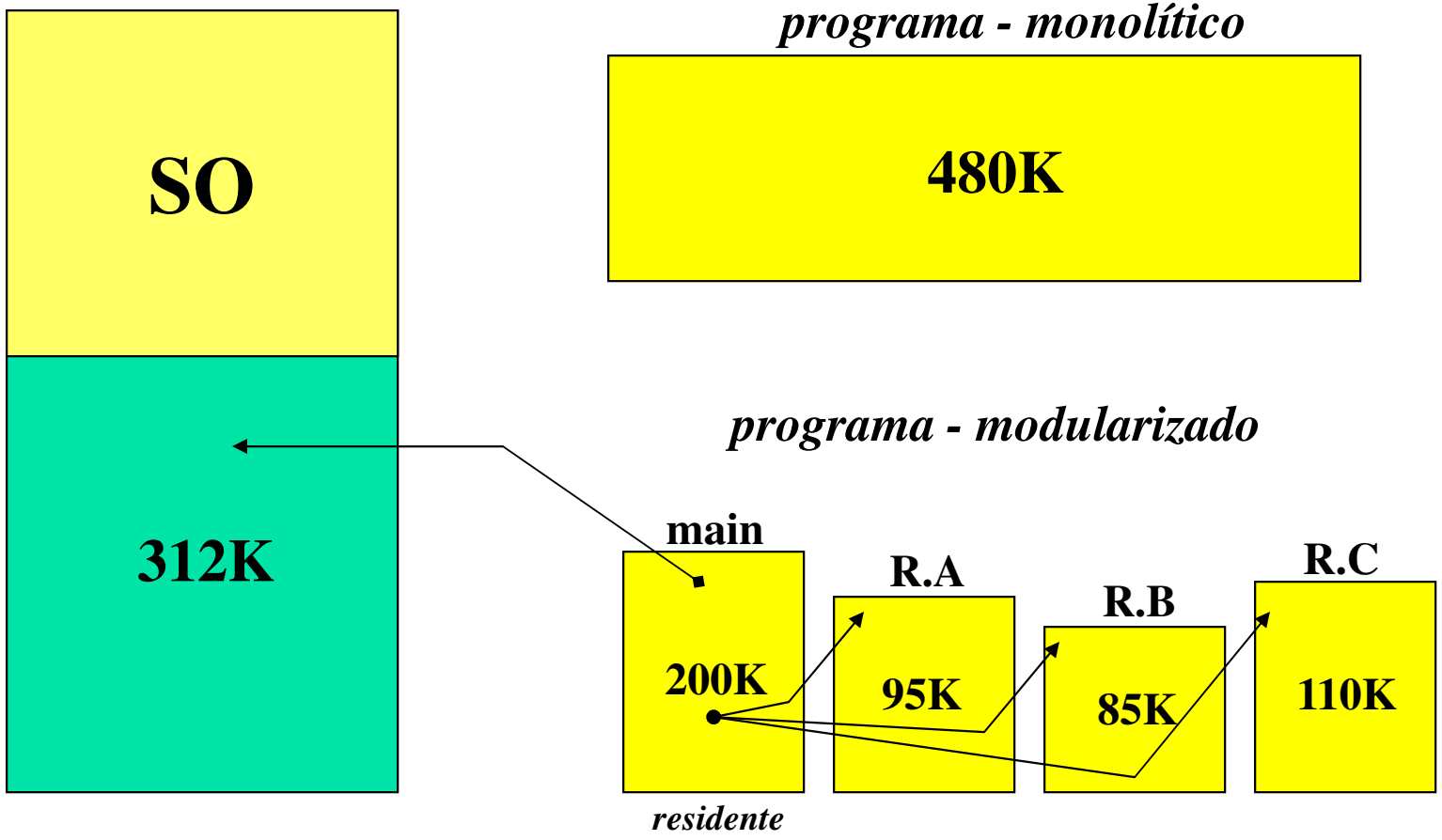
Vantagens da Paginação e Segmentação



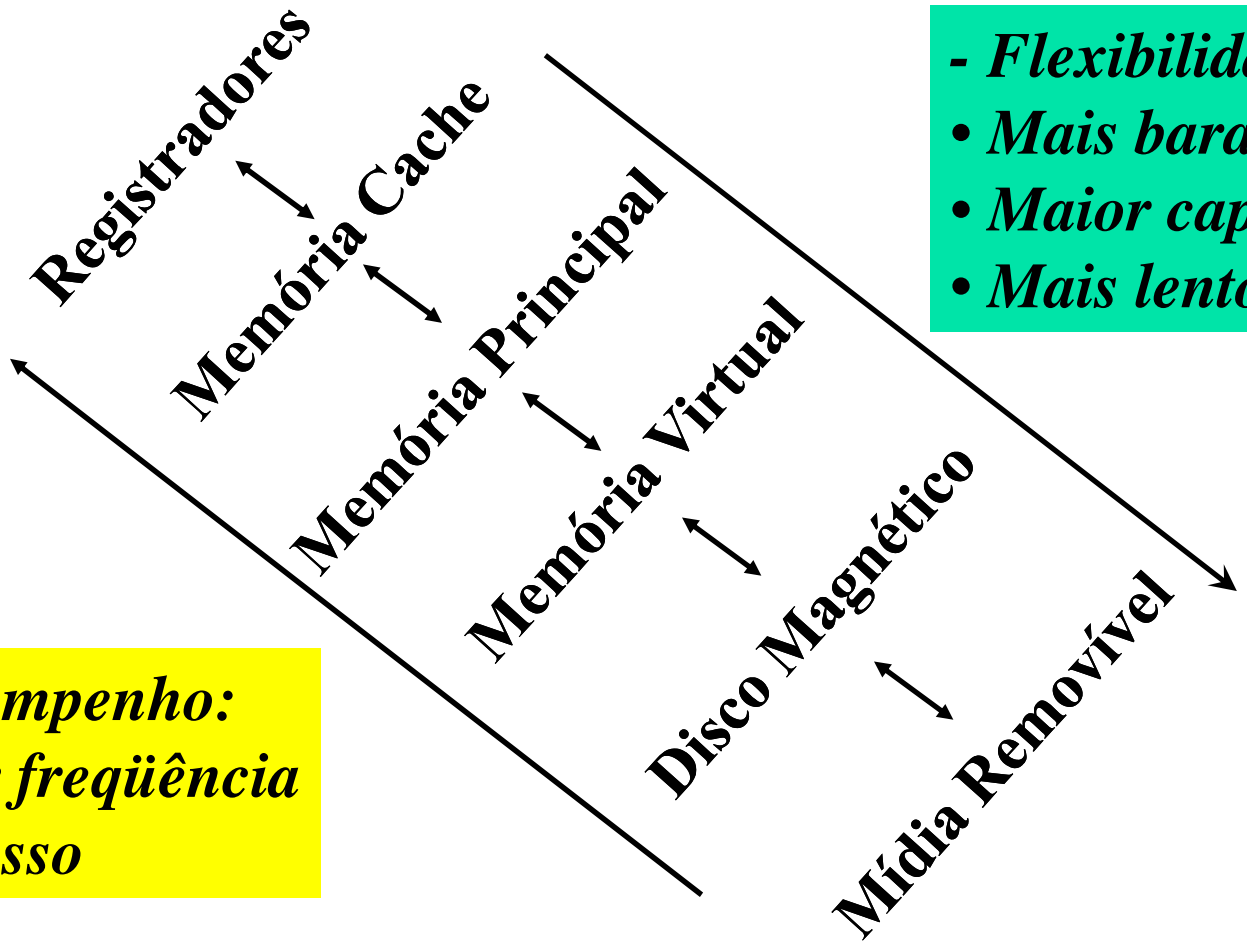
Swapping



Overlay


programa - monolítico
SO
480K
programa - modularizado
312K
main
R.A
R.B
R.C
200K
95K
85K
110K
residente

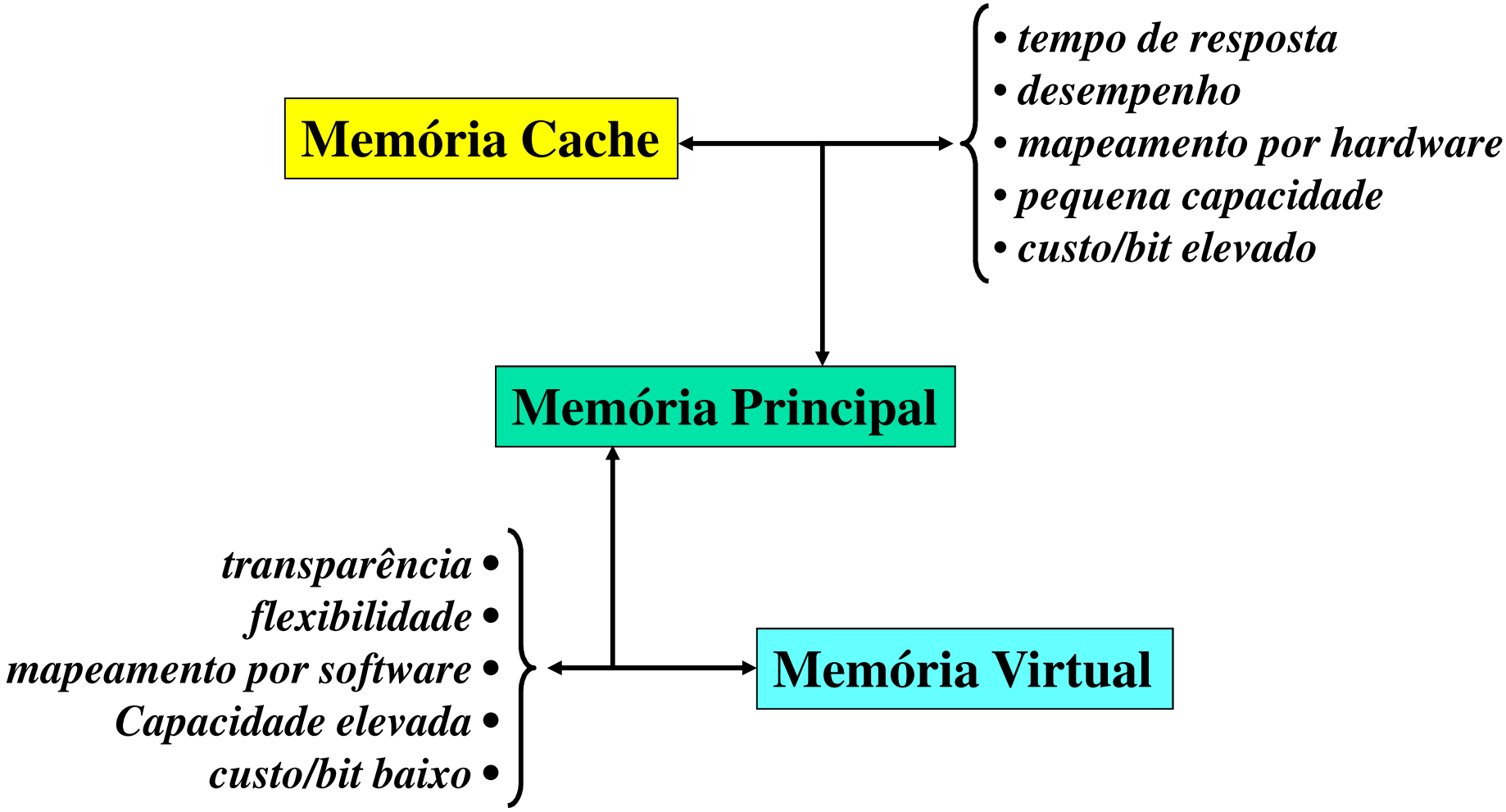
Hierarquia de Memória



- Flexibilidade:
- ***Mais barato***
- ***Maior capacidade***
- ***Mais lento***

- Desempenho:
Maior freqüência de acesso

Hierarquia de Memória

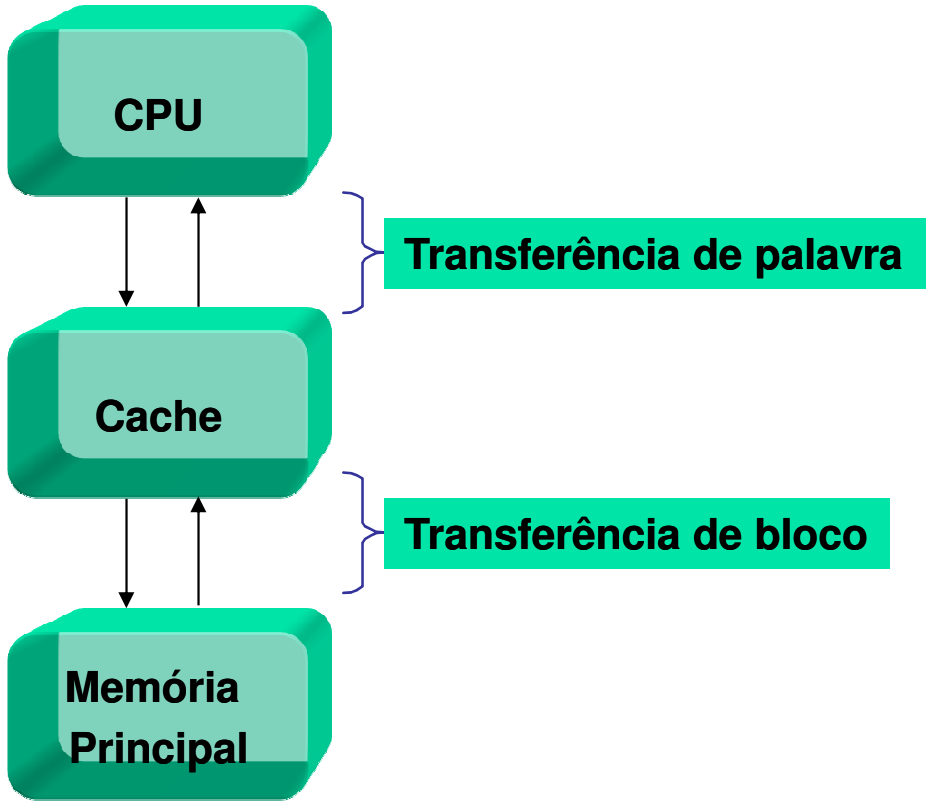


Memória Cache

- **É usada com a finalidade de aumentar o desempenho (reduzir o tempo de resposta) da memória.**
- **É transparente ao S.O.**
- **Tem capacidade de armazenamento bem menor que a Memória Principal**
- **Segue os princípios similares aos da Memória Virtual**
- **Motivação:**
  - o programa faz constantes acessos à memória
  - os processadores são mais rápidos que as memórias convencionais
  - os acessos são geralmente localizados (localidade de referência)

O Sistema Cache

- *A informação é sempre buscada primeiro na Cache.*
- *Quando não está na Cache é carregada para ela.*
- *É pelo menos uma ordem de grandeza mais rápida que a memória principal.*



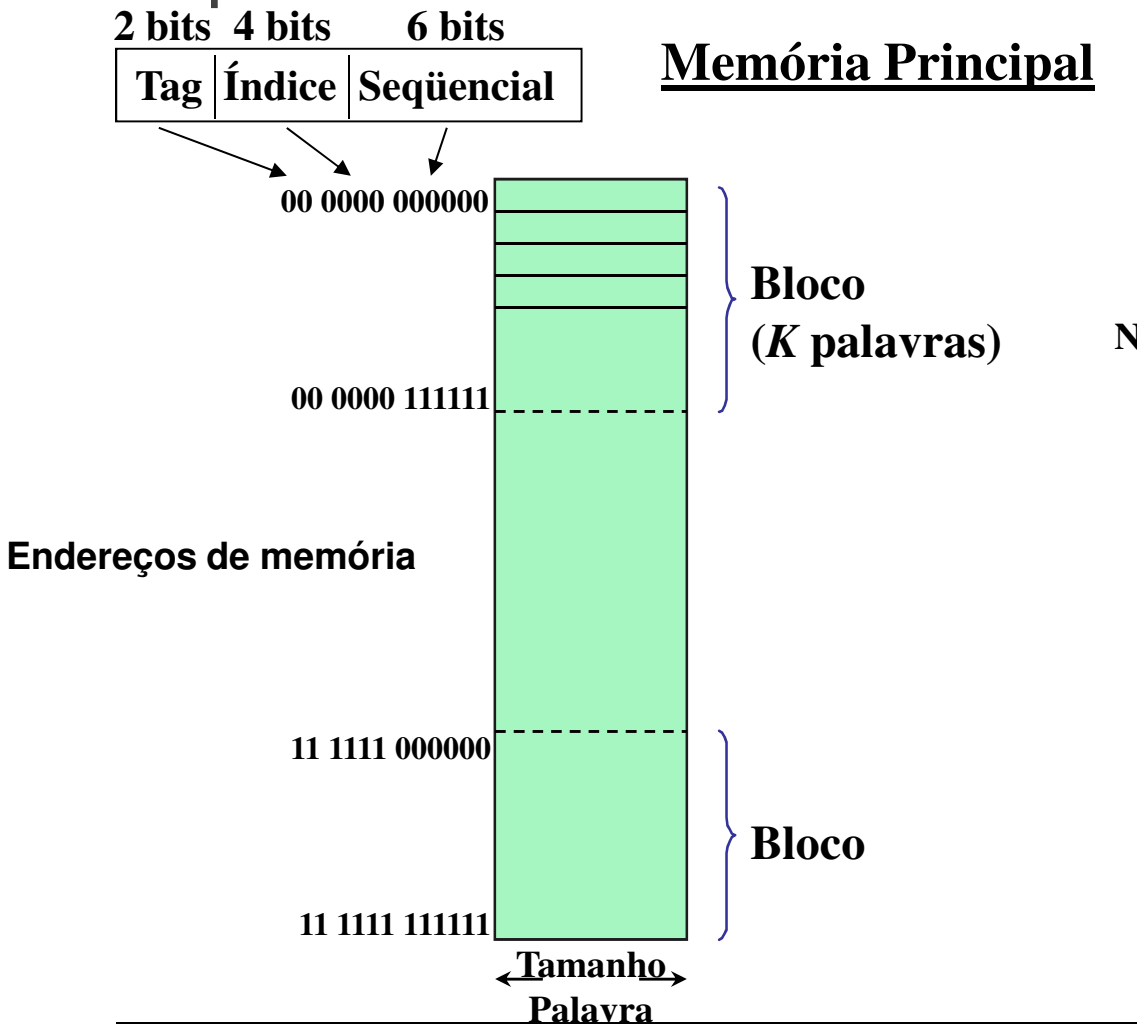
O Sistema Cache

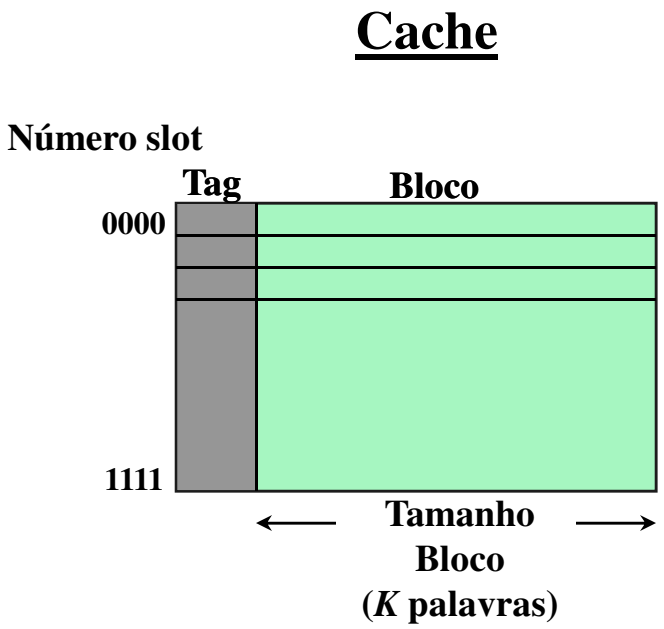
- **Tamanho da Cache**
  - **Caches pequenas têm desempenho ruim**
- **Organização Física**
  - **acesso associativo**
  - **acesso direto**
- **Tamanho do bloco**
  - **a quantidade de dados trocados entre a cache e memória principal**
  - **blocos maiores: maior taxa de acertos devido ao princípio da localidade**
  - **blocos ainda maiores: menor taxa de acertos já que a probabilidade de acesso a uma palavra não carregada por causa do bloco maior é maior**

O Sistema Cache

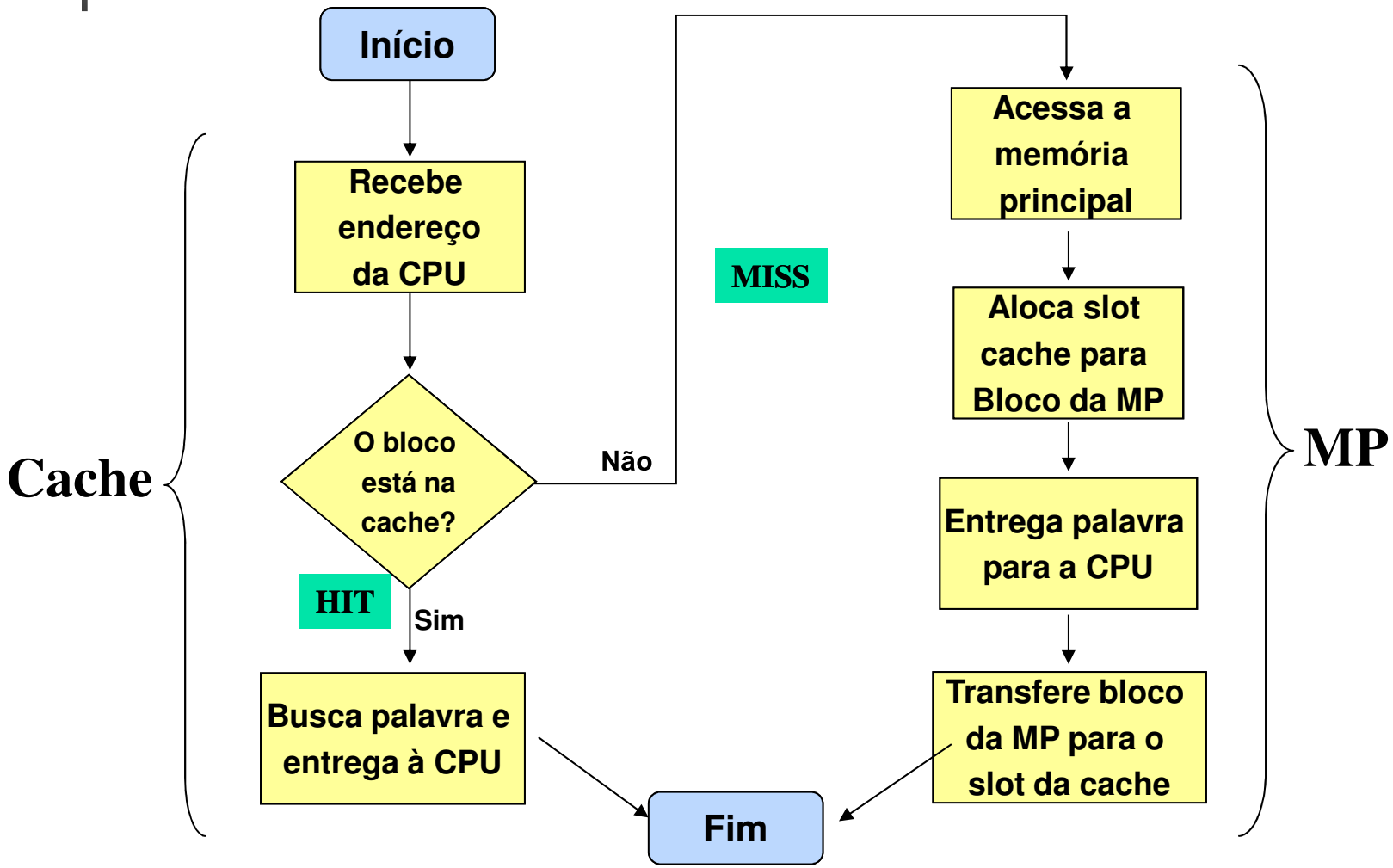
- **Função de mapeamento**
  - indica qual slot um bloco irá ocupar na cache
  - quanto mais flexível, maior o custo de hardware para localizar um bloco
- **Algoritmo de substituição**
  - indica o bloco a ser substituído
  - algoritmo preferencial: Least-Recently-Used (LRU)
- **Política de escrita**
  - quando atualizar a memória principal
  - a cada escrita: mais acessos à memória (write-through)
  - na substituição: problemas de consistência (write-back)

Mapeamento Memória x Cache

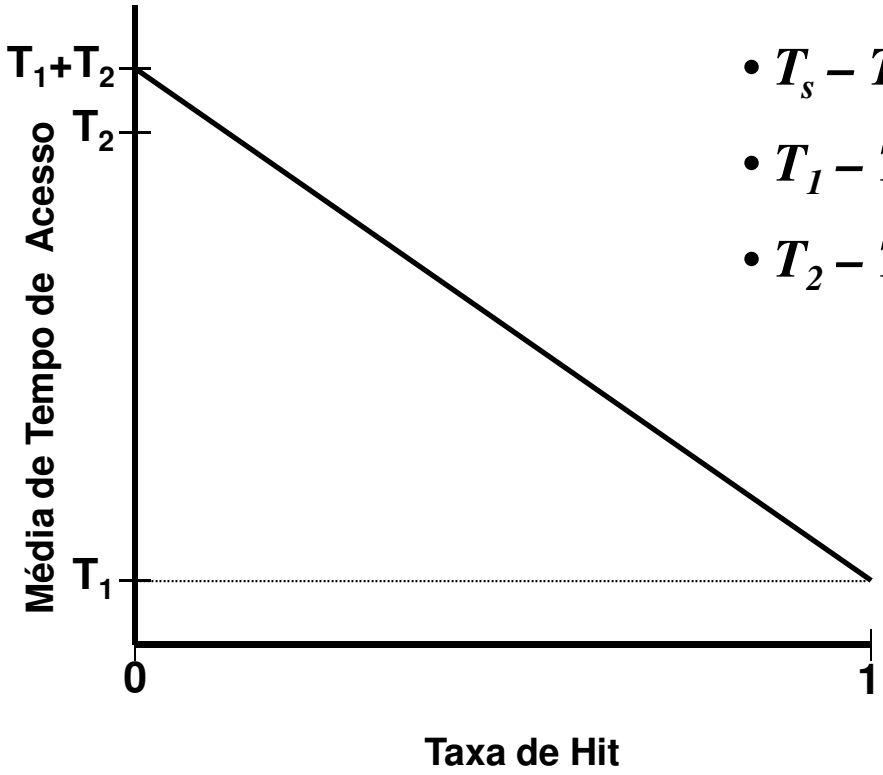




Operação de Leitura - Exemplo



Tempo Médio de Acesso



- T_s – *Tempo médio*
- T_1 – *Tempo da Cache*
- T_2 – *Tempo da MP*

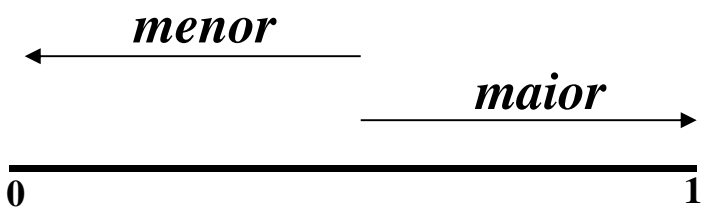
$$T_s = H \times T_1 + (1 - H) \times (T_1 + T_2) = T_1 + (1 - H) \times T_2$$

Medidas de Eficiência

a) Eficiência de Acesso:

$$\frac{T_1}{T_s} = \frac{T_1}{T_1 + (1 - Hit)T_2}$$

Eficiência de acesso



b) Eficiência de Custo:

$$C_s = \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2}$$

C_x = custo / bit
S_x = tamanho em bits

Níveis de Memória Cache