Engenharia de Computação

Arquitetura de Sistemas Operacionais

Gerência de Memória

Prof. Martín Vigil

Adaptado de Prof. Anderson Luiz Fernandes Perez

Introdução

- O Sistema Operacional é o responsável pelo gerenciamento da memória, ou seja, seu uso e otimização.
- Um processo ocupa uma porção de memória denominado *espaço de endereçamento do processo*.
- Um espaço de endereçamento de um processo é o conjunto de posições de memória que um programa executado por este processo pode referenciar.

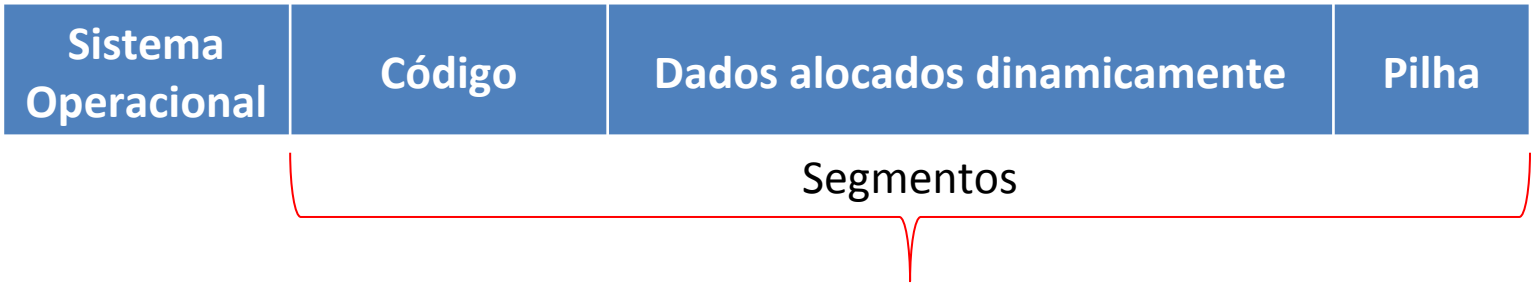
Introdução

- O espaço de endereçamento está associado ao processo, todas as informações acessadas ou guardadas neste espaço são acessíveis a partir do contexto do processo.
- O espaço de endereçamento organiza a memória de maneira a definir a área para uso do sistema operacional e a área para uso dos processos em execução.

Introdução

- Espaço de Endereçamento

- Espaço de Endereçamento Visto pelos Programadores



Introdução

- Existe uma clara noção de confinamento do processo ao seu espaço de endereçamento válido.
- O sistema operacional tem em cada instante um mapa preciso de quais posições de memória o programa pode acessar e de que forma.
- O confinamento garantido pelo SO é chamado de mecanismo de proteção de memória.

Modelo Computacional

- Os programas referenciam a memória para ler instruções e ler e escrever dados.
- Dados e instruções podem ter tamanho variável, desta forma uma operação de leitura de uma instrução ou a escrita de um dado transfere uma quantidade de informações da memória para a CPU.
- Os processadores estão organizados em palavras (múltiplos de bytes). Ex.: 4 bytes (32 bits), 8 bytes (64 bits).
- Os endereços de memória referenciam sempre bytes, indiferente da arquitetura do processador.

Modelo Computacional

- Um endereço de memória permite acessar um byte que conterá parte ou a totalidade do dado ou instrução que se quer acessar.

endereço → valor

Ou

endereço virtual → endereço real → valor

Modelo Computacional

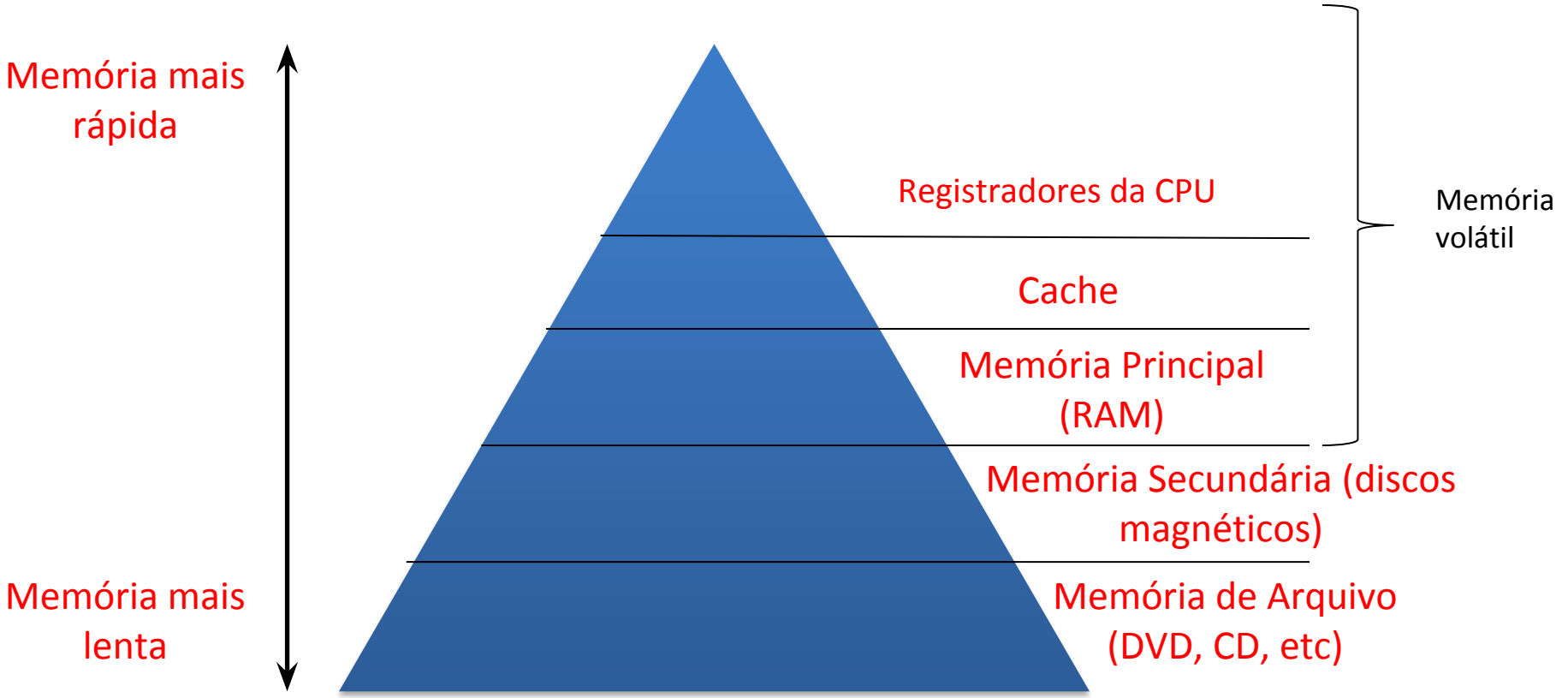
- Normalmente o sistema operacional disponibiliza um conjunto de chamadas de sistema para manipulação da memória.
- Chamadas de sistema para manipular memória:
  - Alocar (ex. malloc e calloc);
  - Liberar (ex. free);
  - Proteger;
  - Mapear;
  - Desmapear;
  - Associar;
  - Desassociar.

Modelo Computacional

- malloc() está disponível quando não há SO? Ex: Arduino

Modelo Computacional

- Hierarquia de Memória

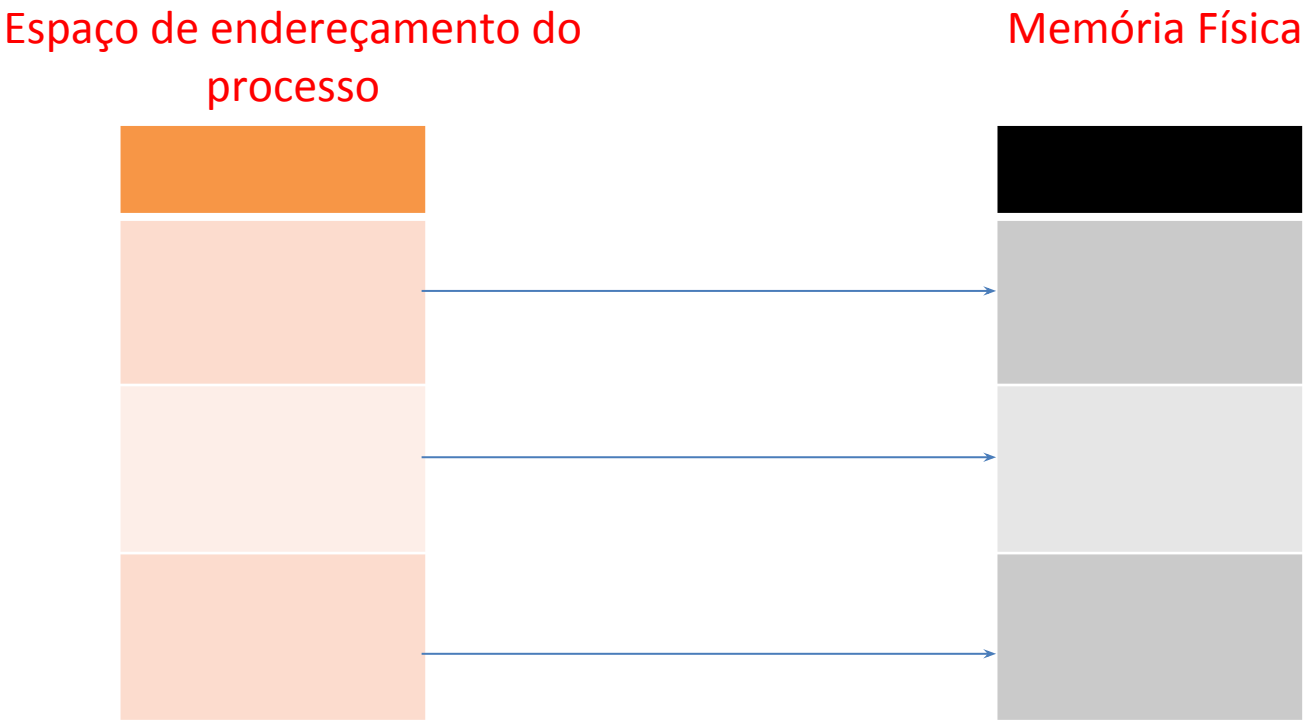


Endereços Reais e Virtuais

- Os computadores antigos suportavam apenas endereçamento real, ou seja, os endereços de memória acessados por um programa têm relação direta com os endereços de memória primária do computador.
- As desvantagens do endereçamento real são:
  - A dimensão do programa é limitada pela dimensão da **memória primária** do computador.
  - Um programa só pode funcionar para os **endereços físicos** para o qual foi escrito.
  - Difícil suportar a multiprogramação.

Endereços Reais e Virtuais

- Endereçamento Real (exemplo)

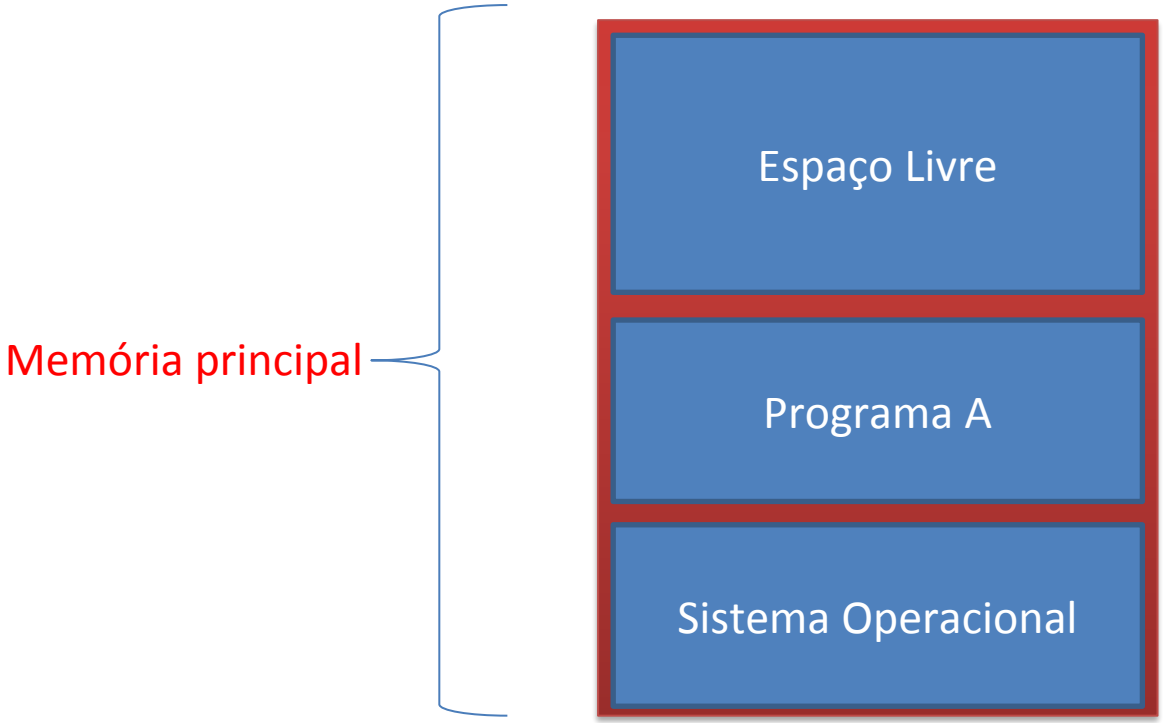


Endereços Reais e Virtuais

- Endereçamento Real
  - Apesar das desvantagens do modelo de endereçamento de memória real, este método é utilizado em alguns sistemas embarcados.
  - O endereçamento real pode ser aplicado em sistemas monoprogramados e sistemas multiprogramados.

Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Monoprogramados (sem overlay)

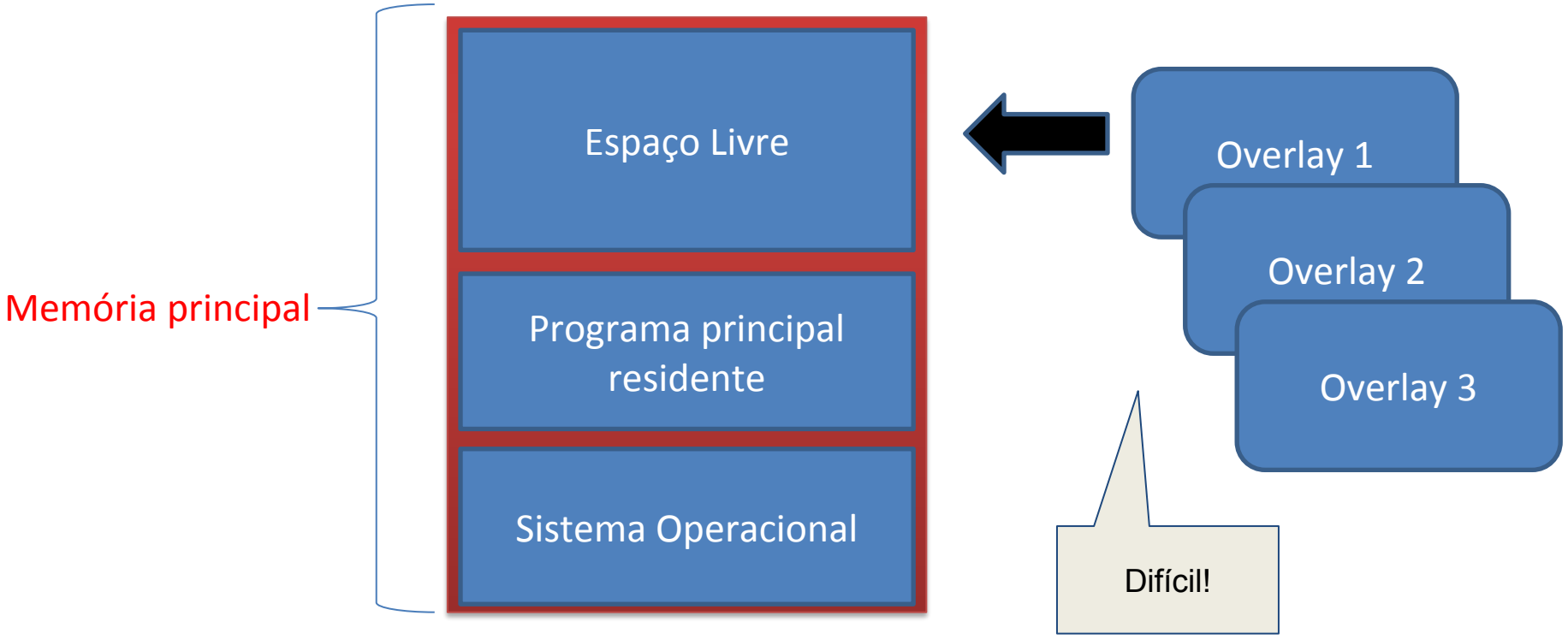


Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Monoprogramados
    - O mapa de memória é basicamente composto de duas partes:
      - Uma ocupada pelo sistema operacional;
      - Outra ocupada pelo programa carregado na memória principal.
    - O problema de falta de espaço na memória principal para alocar um programa pode ser resolvido com o uso de overlays.
    - Um overlay é uma rotina do programa que é carregada na memória somente quando for necessária.

Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Monoprogramados (com overlay)

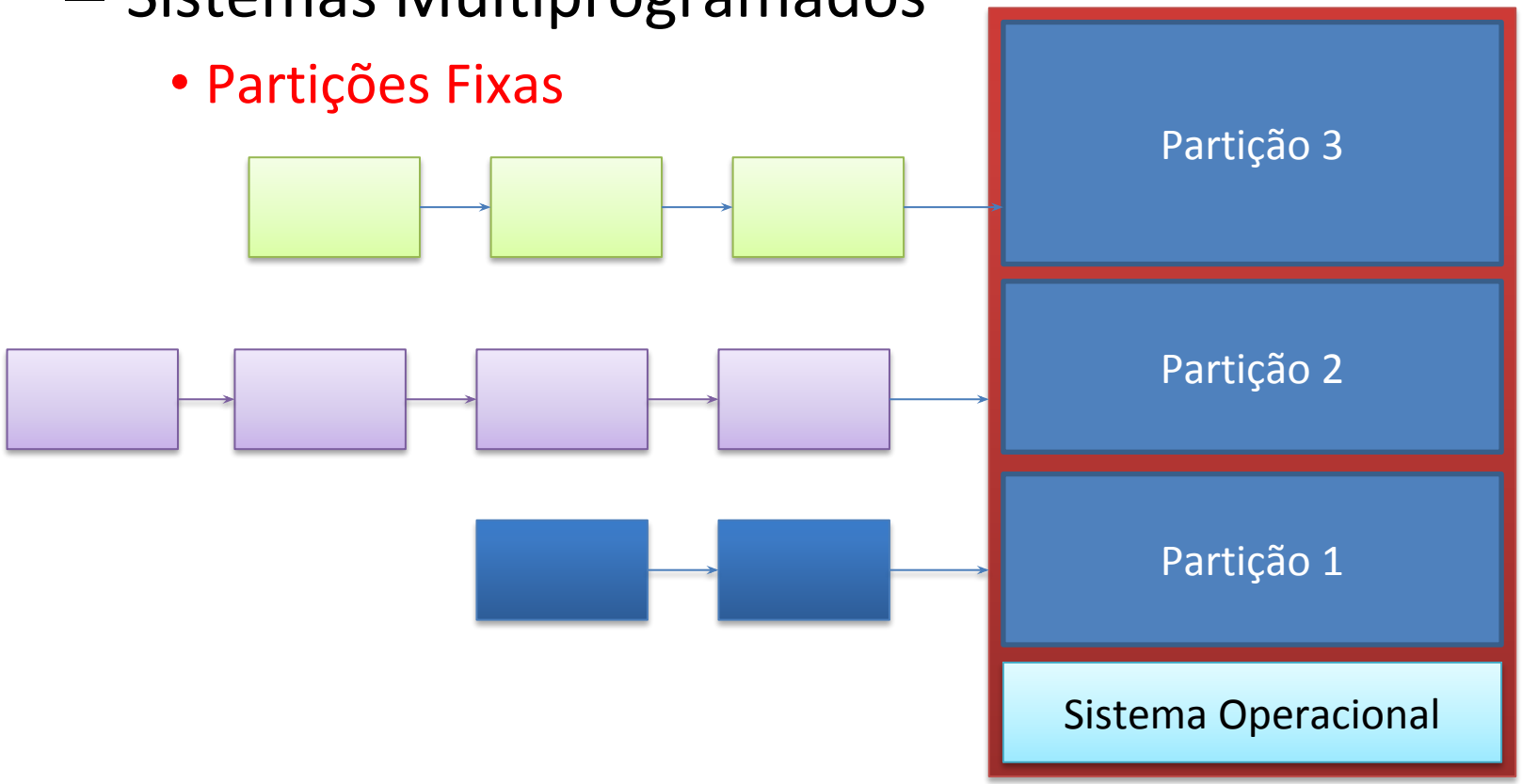


Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Os sistemas multiprogramados permitem que vários programas estejam alocados na memória principal.
    - Para suportar a multiprogramação a memória é dividida em partições que podem ser de tamanho fixo ou variável.
    - Quando um determinado programa for bloqueado em uma operação de entrada e saída de dados, outro programa, que está alocado em outra partição, é colocado em execução.
    - O grau de multiprogramação é dado pelo número de partições existentes no sistema.

Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Fixas



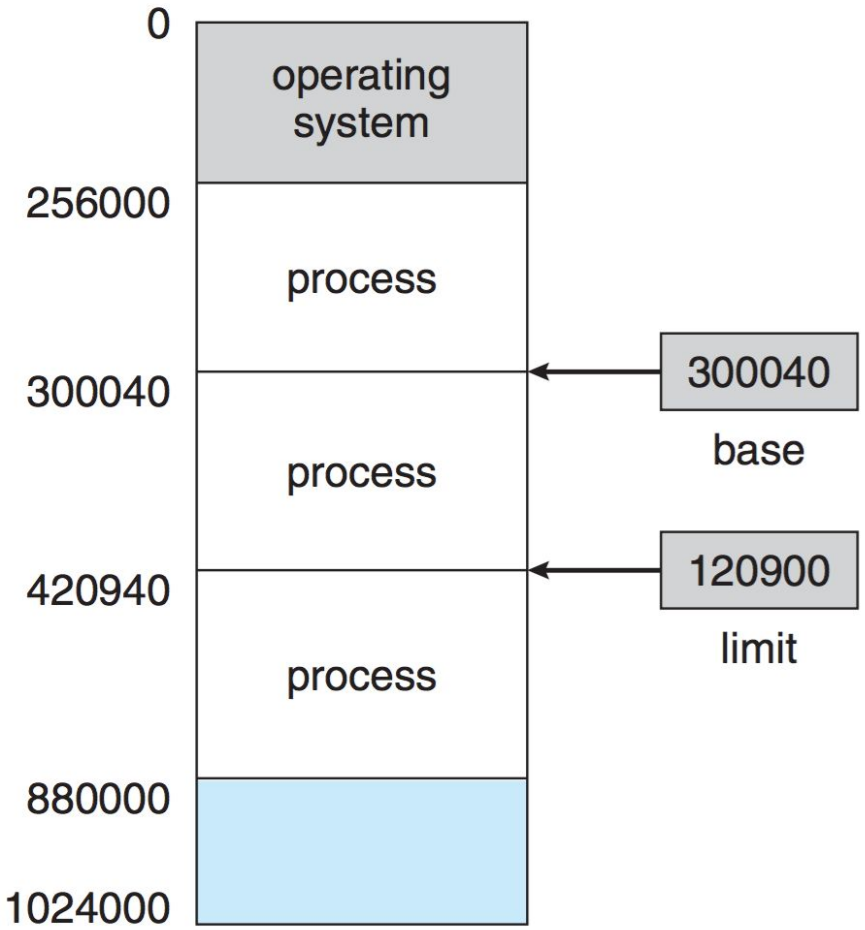
Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Fixas
      - Inicialmente na alocação de um programa a uma partição era feita diretamente, ou seja, os endereços gerados para os programas correspondiam aos endereços reais da máquina.
      - Um forma de resolver o problema da geração de endereços foi a adoção da técnica de relocação.
      - Na relocação o compilador gera para todo o programa o endereço zero de memória, quando o programa for carregado em memória o SO decide em qual partição este será alocado, desta forma, adiciona um deslocamento ao endereço gerado pelo compilador.

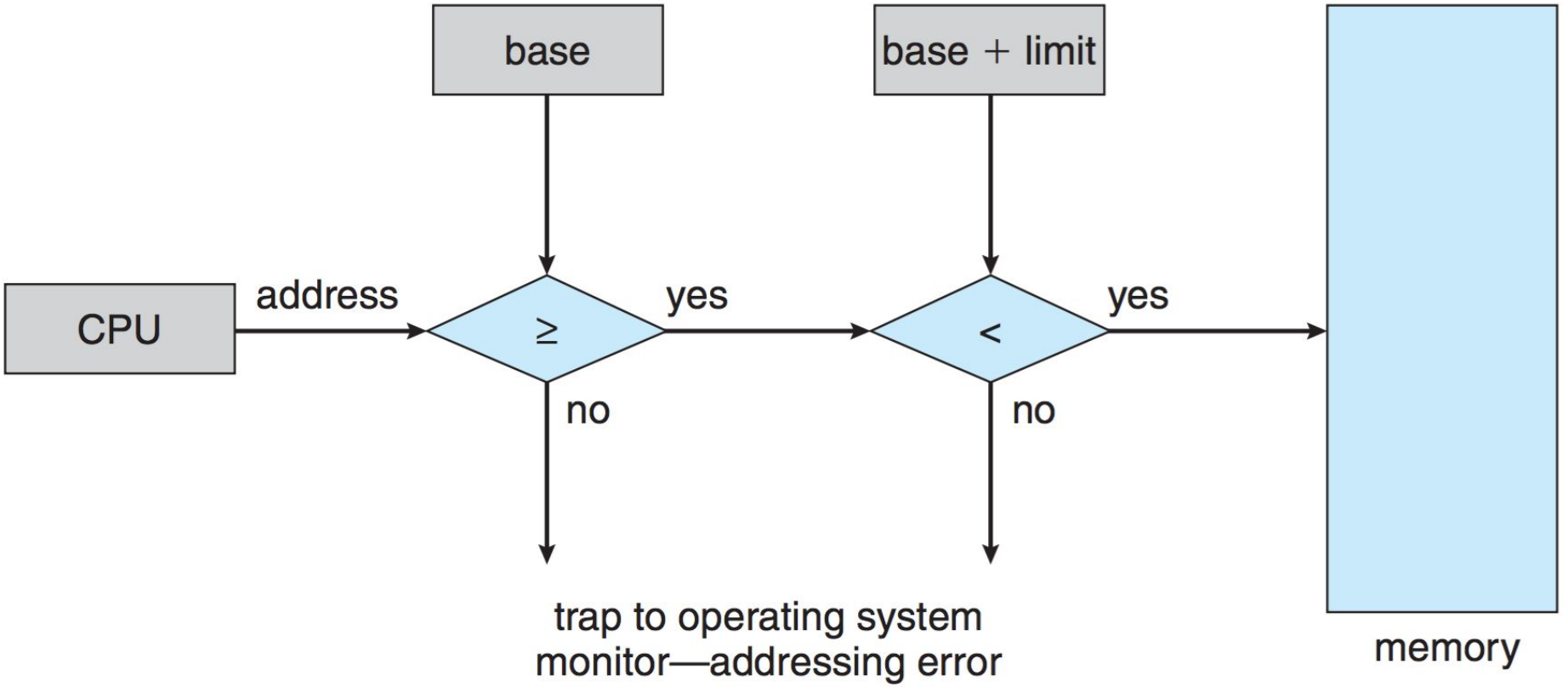
Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Fixas
      - Para a alocação de programas usava-se um registrador especial chamado de registrador base.
      - O registrador base era carregado com o endereço físico do início da partição.
      - O hardware somava o endereço do registrador base com o endereço do programa para obter o endereço real da alocação do programa.
      - Um outro registrador chamado de registrador de limite era utilizado para controlar o acesso a memória e impedir que programas acessassem partições de memória de outro programa.

Endereços Reais e Virtuais



Endereços Reais e Virtuais



Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Fixas pode gerar fragmentação **interna**.

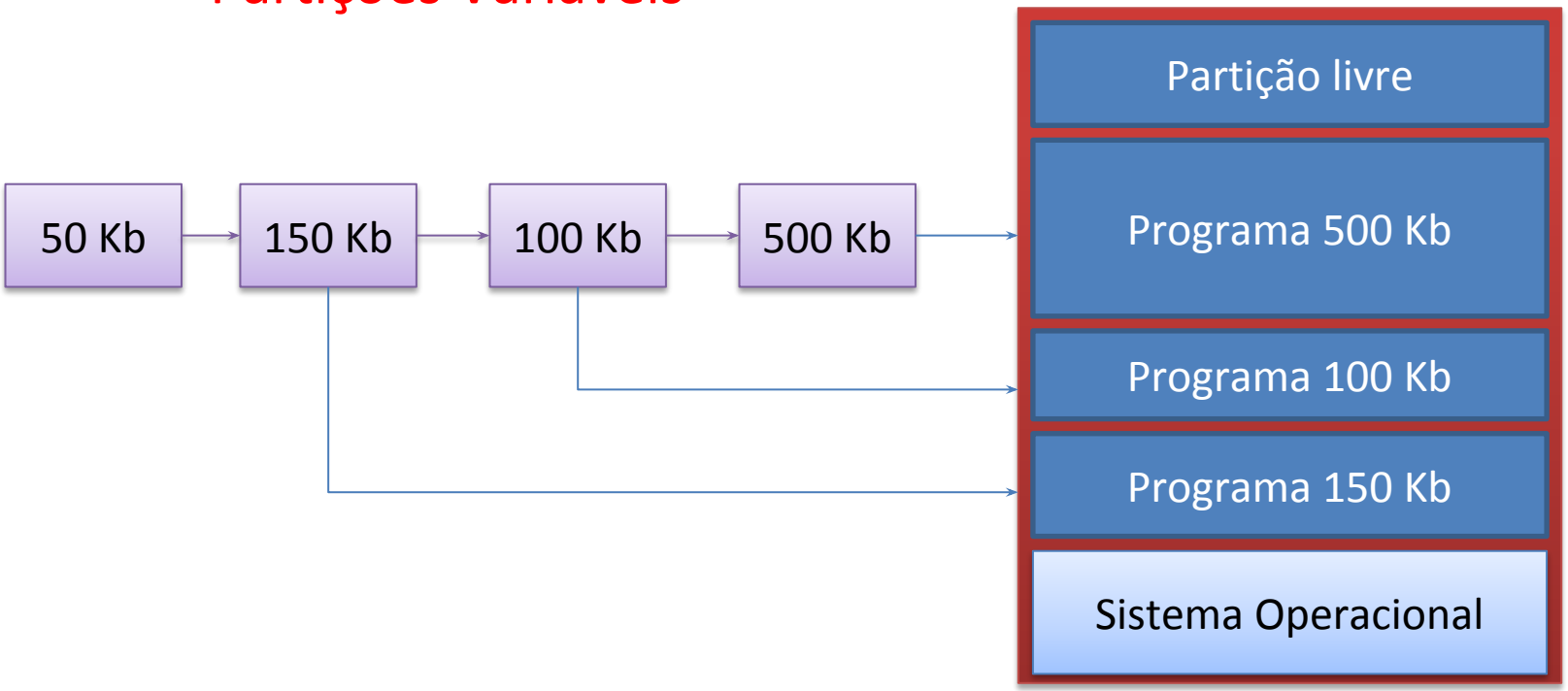
Partição

Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Variáveis
      - Na técnica de partições variáveis existe um espaço reservado para o SO e o restante da memória é considerado espaço livre.
      - A qualquer tempo, caso um processo demande memória, o espaço livre pode ser dividido e aproveitado para alocar programas.
      - Neste técnica um programa ocupa exatamente o quanto necessita da memória principal.

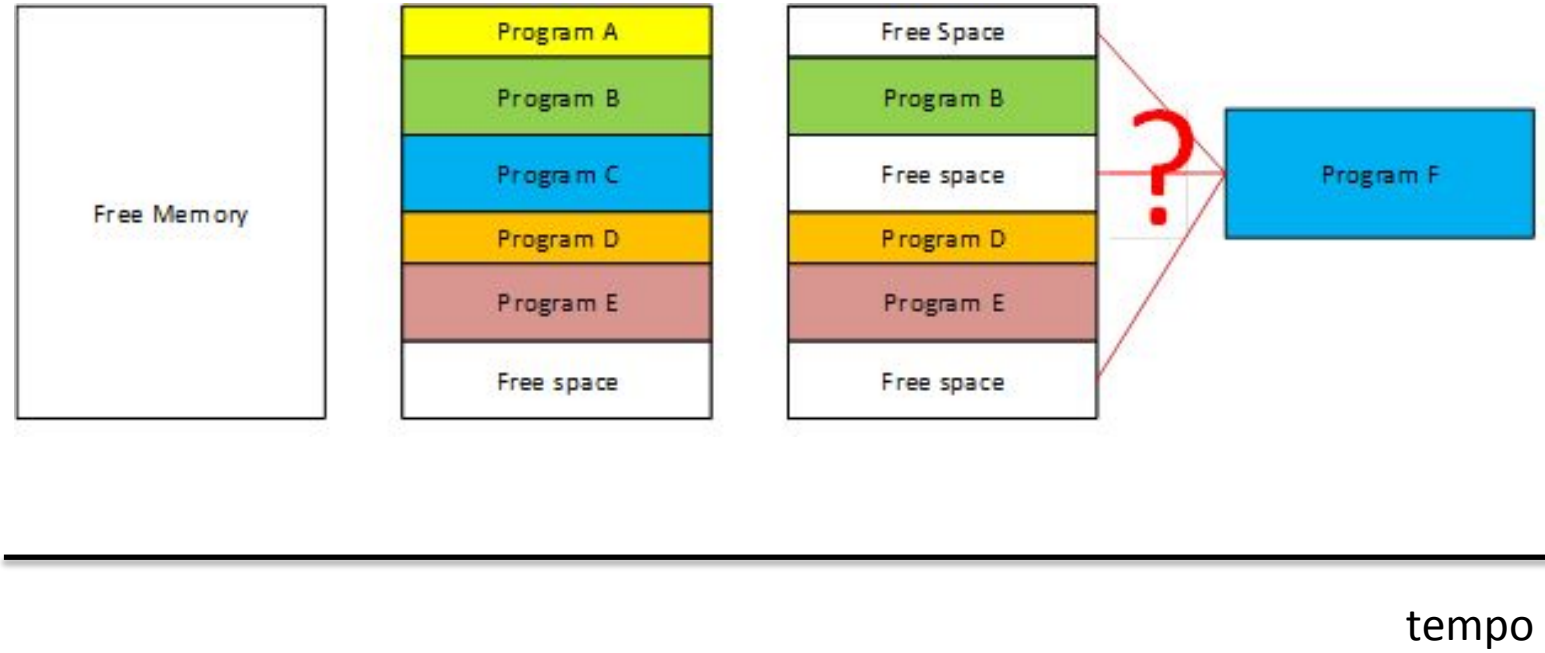
Endereços Reais e Virtuais

- Endereçamento Real
  - Sistemas Multiprogramados
    - Partições Variáveis



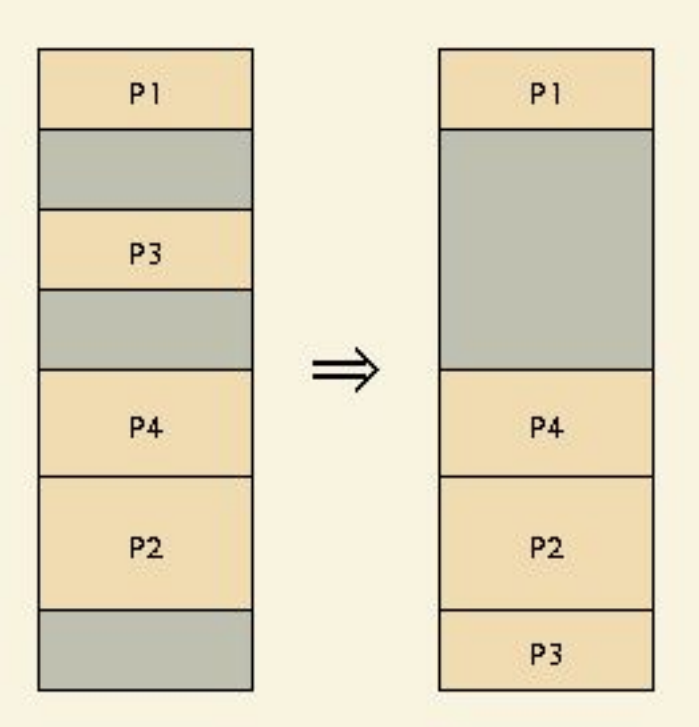
Endereços Reais e Virtuais

- Partições variáveis podem causar o problema de fragmentação externa



Endereços Reais e Virtuais

- O problema de fragmentação externa pode ser resolvido pela compactação de memória



Custo computacional significativo

Endereços Reais e Virtuais

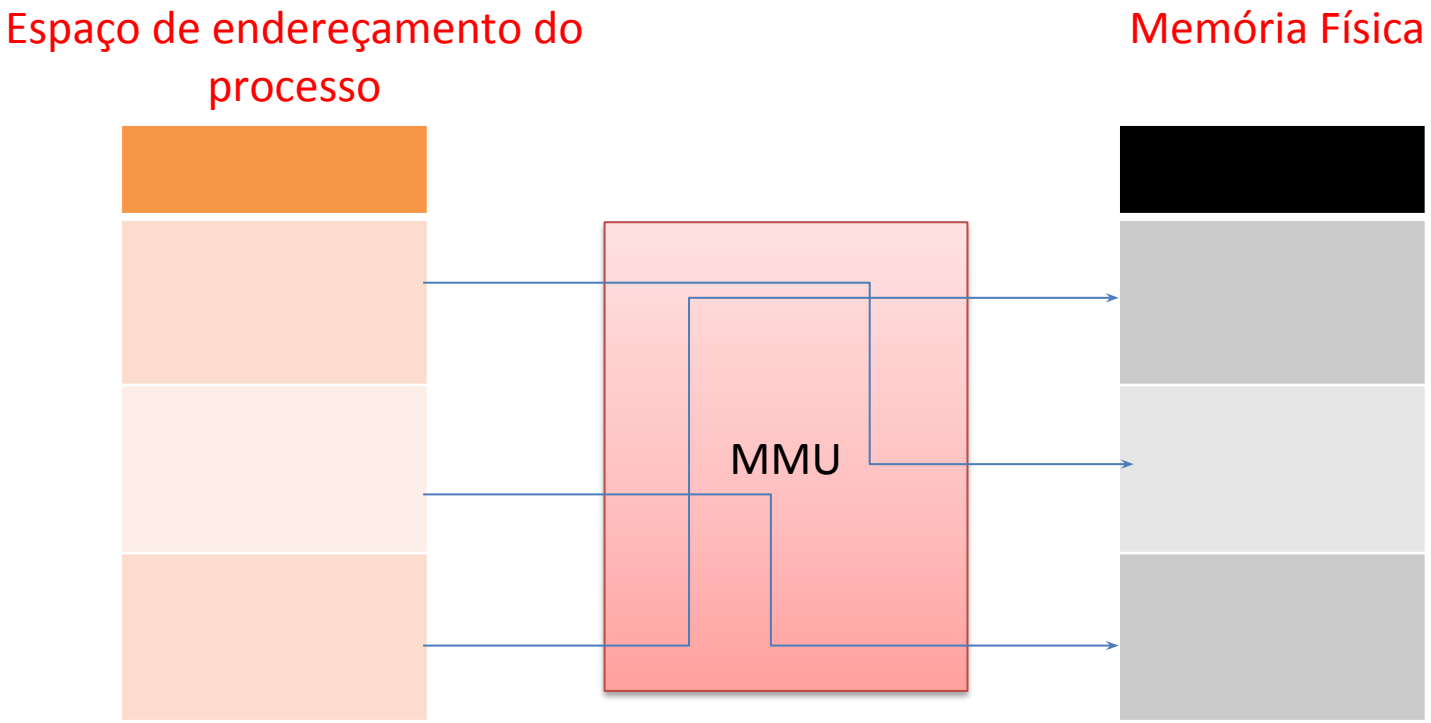
- Endereçamento Virtual
  - O sistema de endereçamento virtual desvincula os endereços gerados pelo programa dos endereços físicos acessados na memória principal.
  - O programador passa a ter um espaço de endereçamento independente.
  - O espaço de endereçamento passa a ser maior do que o tamanho da memória principal.

Endereços Reais e Virtuais

- Os endereços gerados pelo programa (virtuais ou lógicos) são convertidos pelo processador, em tempo de execução, em endereços físicos (reais).
- No sistema de endereçamento virtual a CPU é dotada de um hardware especial chamado MMU (*Memory Management Unit*).

Endereços Reais e Virtuais

- Endereçamento Virtual (exemplo)



Endereços Reais e Virtuais

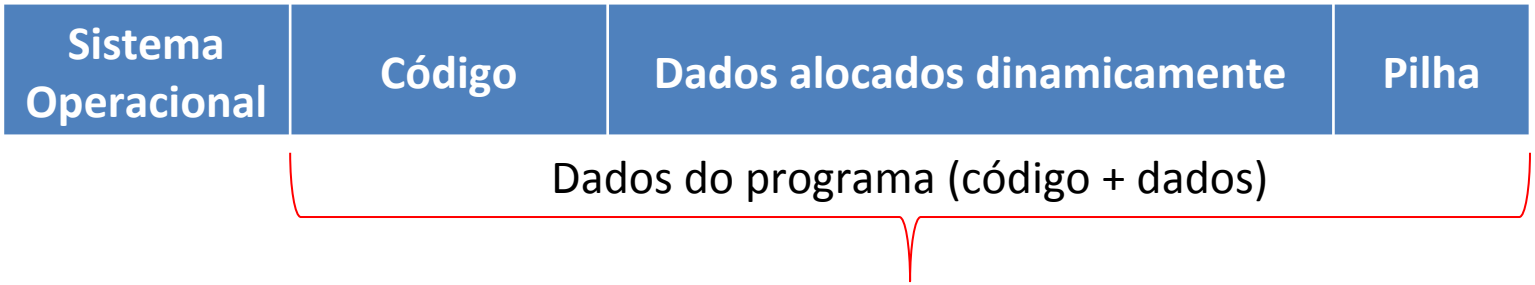
- Um endereço virtual não se refere nem a memória primária e nem a memória secundária, mas a um endereço lógico.
- O hardware de gerenciamento de memória (MMU) e o SO são responsáveis por traduzir/mapear o endereço virtual em endereço real na memória primária ou secundária.
- Caso o dado a ser acessado esteja em memória secundária, este é carregado na memória primária para ser acessado pelo processo.
- O uso de memória secundária permite alocar um programa que seja **maior** que a memória primária disponível.

Endereços Reais e Virtuais

- Endereçamento Virtual
  - A tradução de endereços virtuais em endereços reais é feito pelo tradutor de endereços (MMU – *Memory Management Unit*).
  - O mecanismo de tradução mantém uma tabela onde cada endereço virtual corresponde a um endereço real.
  - Para melhorar o desempenho e evitar que a tabela fique muito grande, ela indexa blocos, ou seja, o endereço de um bloco virtual corresponde a um endereço de um bloco real.
  - Um endereço passa a ter a forma bloco + deslocamento.

Endereços Reais e Virtuais

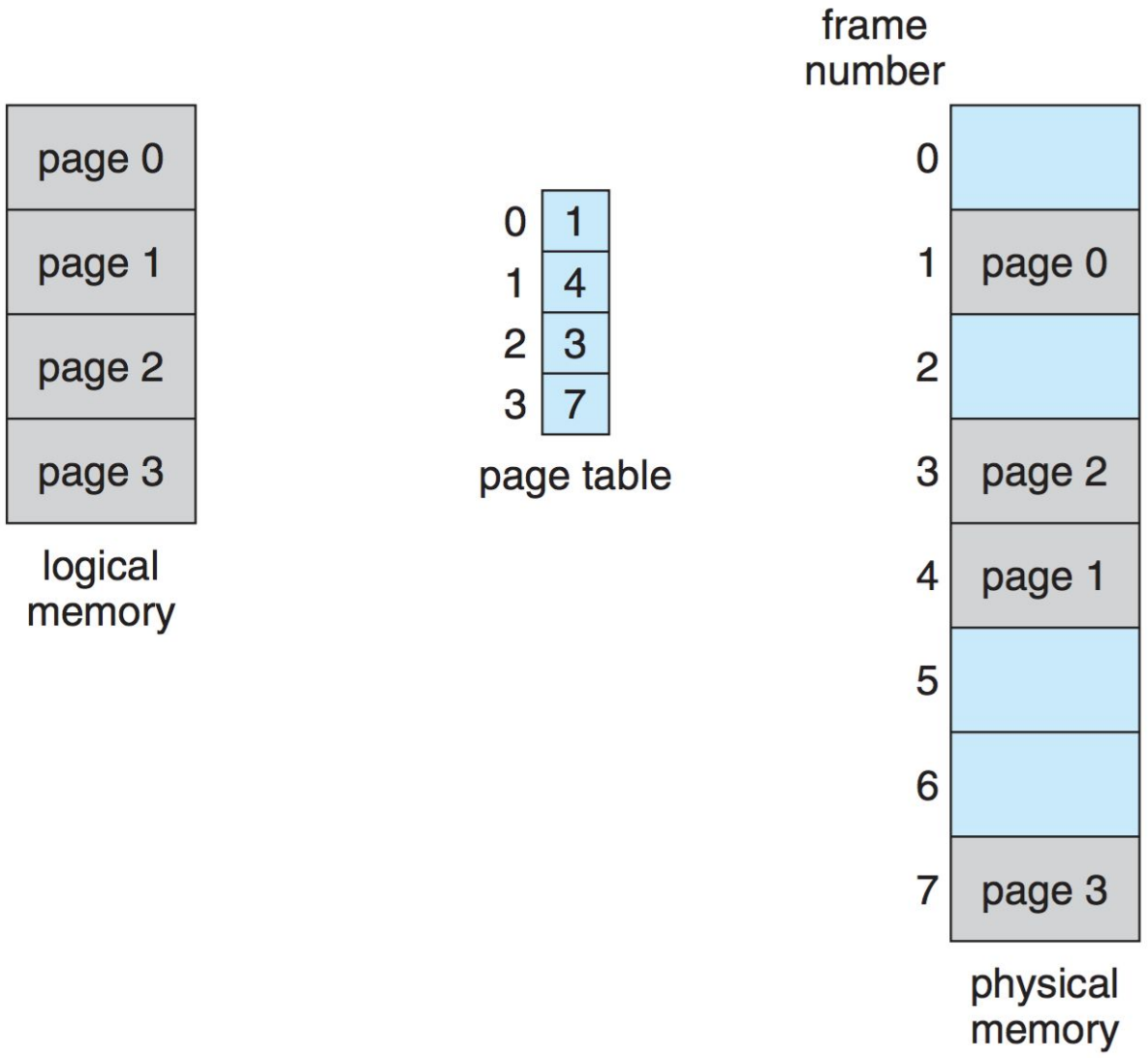
- Endereçamento Virtual
  - Visão da Memória pelo Programador



Endereços Reais e Virtuais

- Endereçamento Virtual
  - Paginação
    - A ideia da paginação é oferecer ao programador um espaço de endereçamento (virtual) contíguo.
    - A memória física é dividida em quadros de tamanho fixo
    - Memória lógica é dividida em páginas
    - Tamanho da página = tamanho do quadro = definido pelo hardware

Endereços Reais e Virtuais



Endereços Reais e Virtuais

- Endereçamento Virtual
  - Paginação
    - Endereço lógico

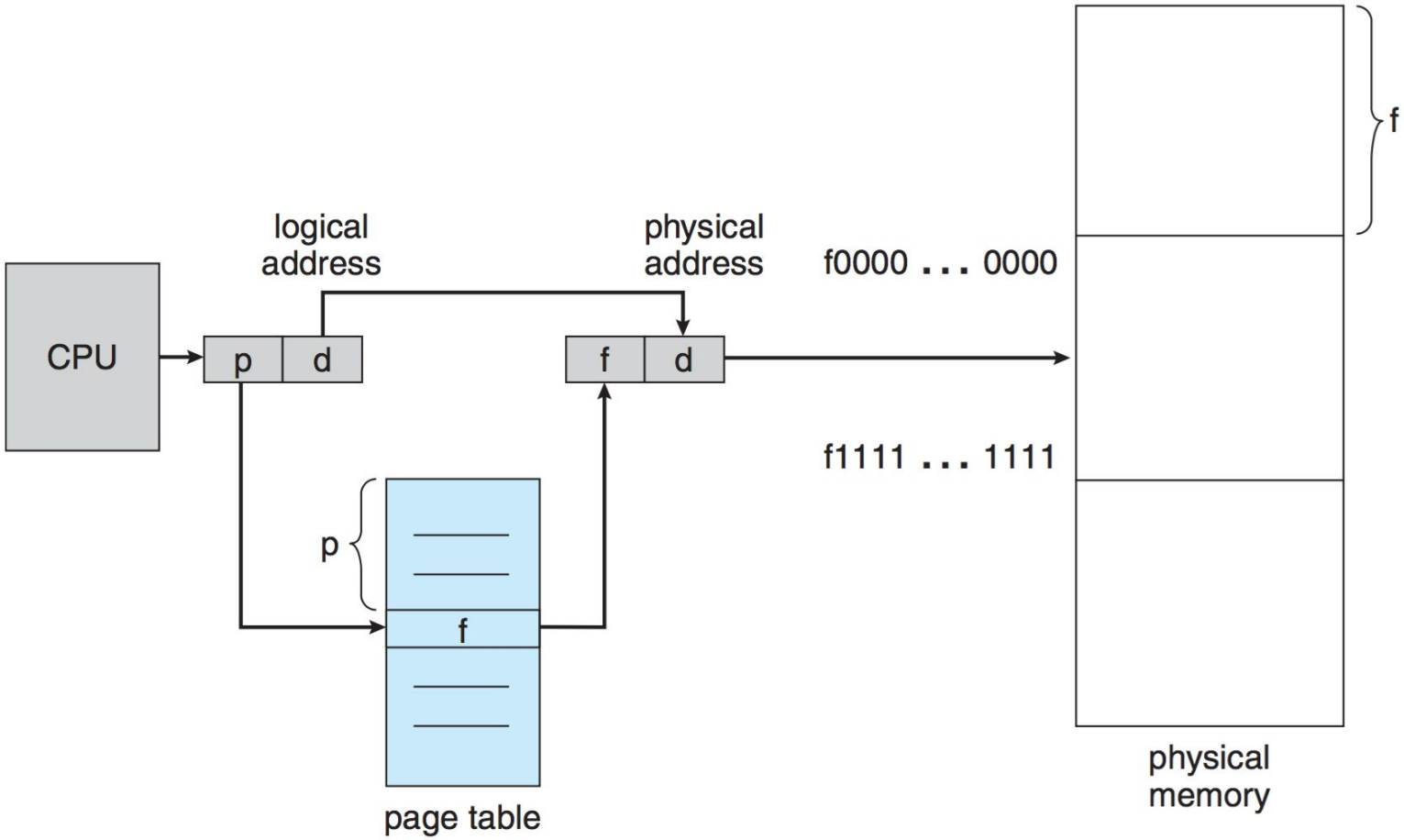
< #página | deslocamento >

Endereços Reais e Virtuais

- Endereçamento Virtual
  - Paginação
    - Exemplo de endereço lógico de 10 bits, onde
      - 7 bits para endereçar páginas/quadros = 2^7 páginas/quadros
      - 3 bits para endereçar bytes =2^3 bytes

#página	deslocamento
7 bits	3 bits

Endereços Reais e Virtuais

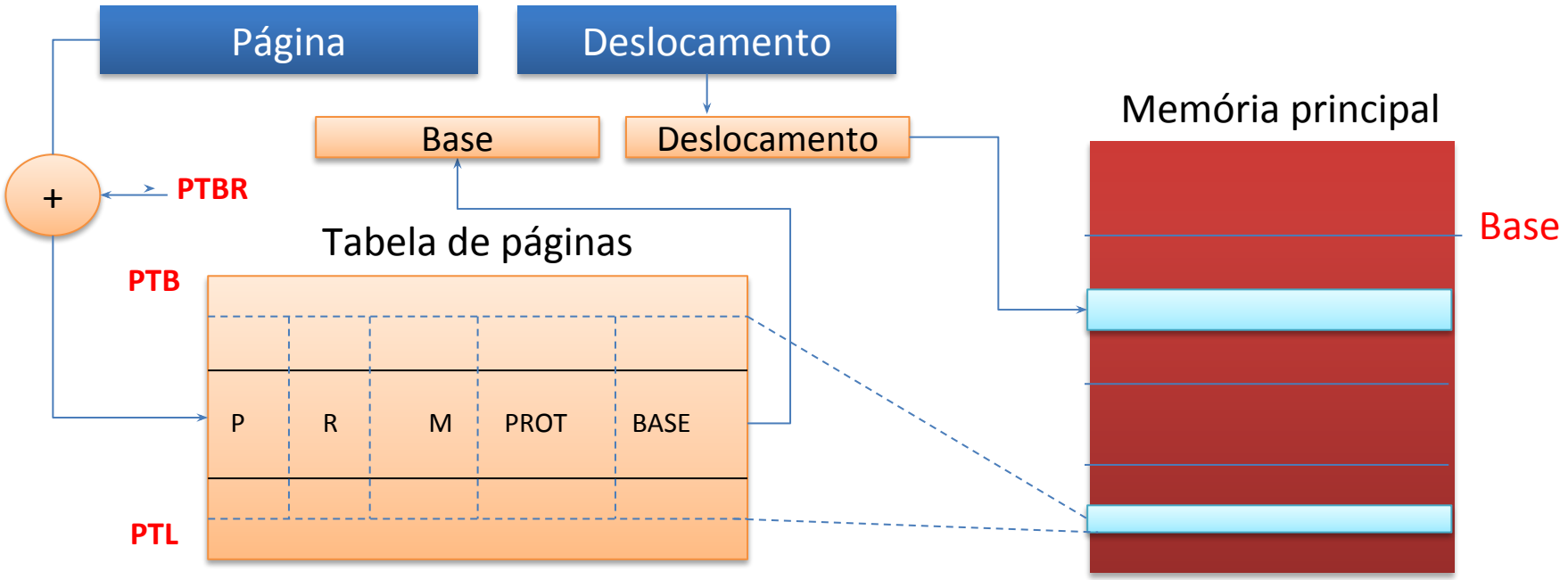


Endereços Reais e Virtuais

- Tabela de Páginas
  - Existe uma tabela de páginas por processo
  - Tabelas ficam na RAM
    - Page Table Base Register (PTBR): início da tabela
  - Linhas têm o mesmo tamanho
  - Colunas:
    - Base: onde começa o quadro
    - PROT: info de proteção
    - Bit P: indica se a página está carregada na memória primária.
    - Bit R: indica se a página foi acessada (referenciada).
    - Bit M: indica se a página foi acessada e modificada.

Endereços Reais e Virtuais

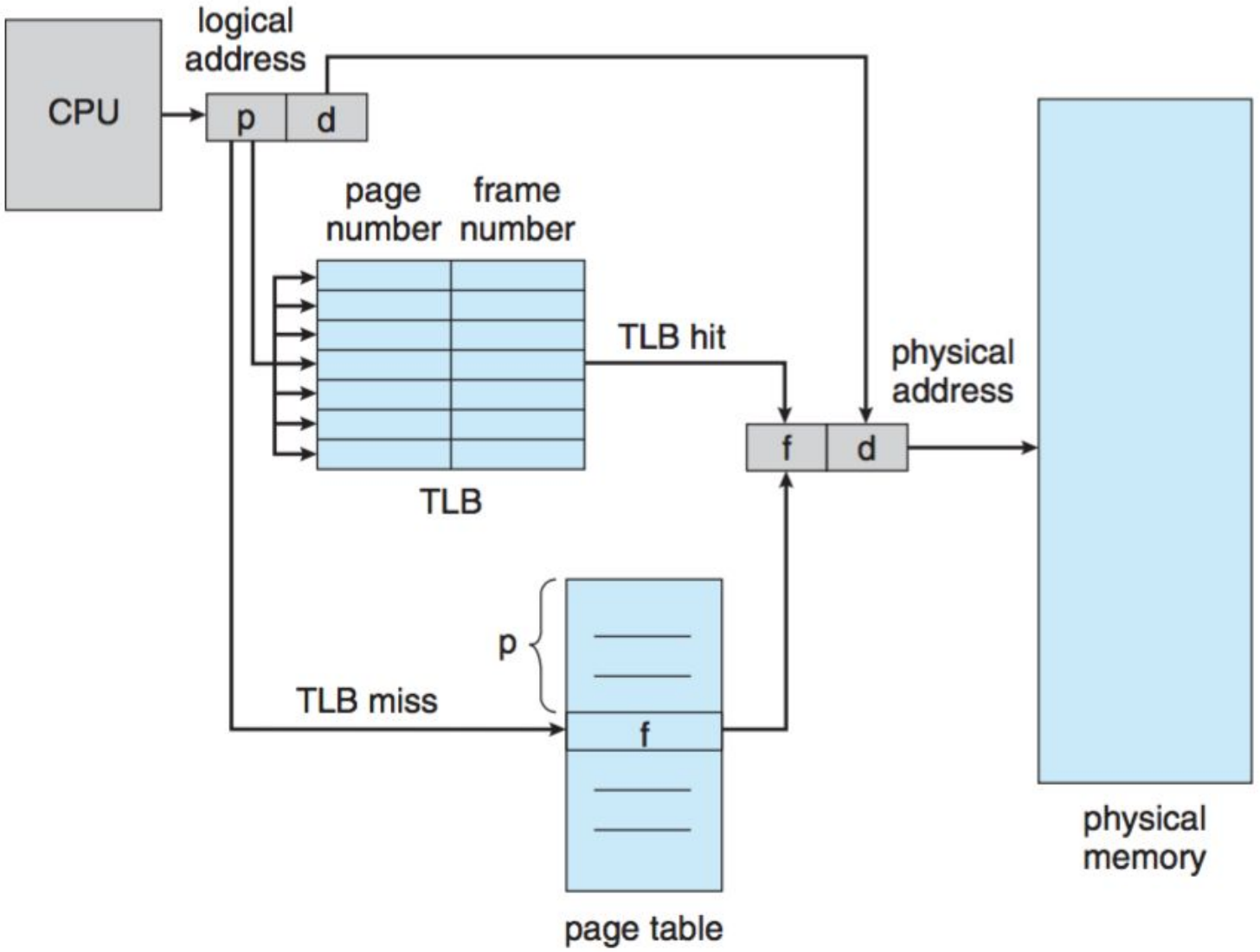
- Endereçamento Virtual
  - Paginação
    - Tabela de Páginas



Endereços Reais e Virtuais

- Tabela de Páginas
  - Ineficiente: dois acessos a memória:
    - 1) Acessar a tabela de páginas para localizar quadro
    - 2) Buscar dado desejado no quadro
  - Solução: usa-se uma memória associativa de acesso muito rápido, esta memória é chamada TLB *(Translation Lookaside Buffer).*
  - A TLB guarda os descritores da *n* últimas páginas acessadas pelo programa.
  - Quando o programa gera um endereço, a memória associativa consulta simultaneamente todas as posições, verificando se existe uma entrada cujo endereço da página seja igual ao endereço gerado pelo programa.
    - Se a página for encontrada na TLB seu endereço é colocado na saída,
    - Caso contrário é indicado ao hardware de memória que a página não foi encontrada. O SO busca a página faltante e insere na na TLB. Se ela estiver cheia, então uma página na TLB deverá ser substituída.

Endereços Reais e Virtuais



Endereços Reais e Virtuais

- Endereçamento Virtual
  - Vantagens da Paginação
    - Sem fragmentação externa
    - Programador trabalha apenas com endereço lógico
    - Permite criar algoritmos para trocar páginas de acordo com comportamento dos processos
  - Desvantagens
    - Fragmentação interna

Algoritmos de Gerenciamento de Memória

- Os algoritmos de gerenciamento de memória são necessários para:
  - Decidir onde se deve **alocar** um bloco de memória (ex. página);
  - Quando **transferir** um bloco da memória secundária para a memória primária e vice-versa;
  - Qual bloco **substituir** da memória quando não existir mais espaço livre na memória primária.

Algoritmos de Gerenciamento de Memória

- A operação de alocação de memória é realizada quando:
  - Um processo é criado ou finalizado;
  - Um processo solicita mais espaço de memória (alocação dinâmica. Ex. função *malloc()* linguagem C) e em virtude das chamadas de subrotinas (pilha de execução);

Algoritmos de Gerenciamento de Memória

- Alocação de Páginas
  - Em um sistema baseado em paginação, a alocação de página da memória é realizada em ordem FIFO.
  - As páginas livres são mantidas em uma lista, gerenciada, geralmente, conforme uma fila.
  - Na alocação retira-se a primeira página da fila e na liberação insere-se a página no fim da fila.

Algoritmos de Gerenciamento de Memória

- Diagrama de Estados das Páginas
  - Uma página pode estar em um dos seguintes estados:
    - Livre: a página não está sendo usada, ou seja, está disponível para ser alocada. Todas as páginas inicialmente encontram-se neste estado.
    - Ocupada: página alocada pelo sistema operacional e que pode conter dados e código de um programa.
    - Modificada: página que foi alterada e que depois deixou de ser utilizada.
    - Modificada em escrita: a página está sendo escrita na memória secundária. Operação que deverá ser realizada sempre que uma página é liberada da memória principal e tiver sido escrita por um processo.

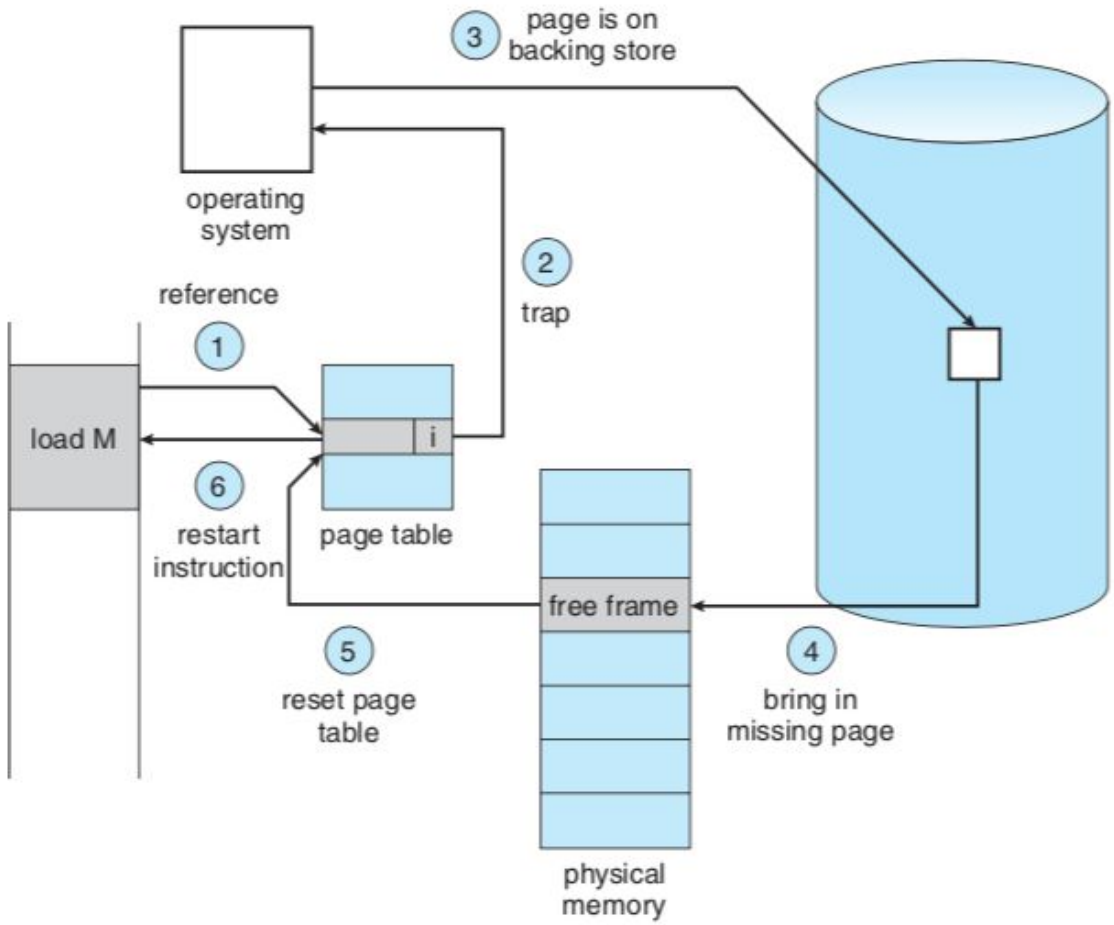
Algoritmos de Gerenciamento de Memória

- Diagrama de Estados das Páginas

Algoritmos de Gerenciamento de Memória

- Transferência de Páginas
  - A transferência de páginas da memória secundária para a memória primária é por demanda.
  - Uma vez que um processo necessite de uma página ainda não alocada na memória principal, é gerado um erro por falta (exceção) de página, então o SO deve carregar a página faltante da memória secundária para a memória primária.
  - Toda a vez que uma página escrita tiver que deixar a memória principal, o sistema necessita gravar esta página na memória secundária em uma área chamada de *área de paginação.*

Algoritmos de Gerenciamento de Memória



Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - O SO deve manter duas listas de páginas:
    1. A lista de páginas livres, ou seja, páginas que o sistema mantém uma cópia na memória secundária. Essas páginas podem ser substituídas por outras página do processo.
    2. A lista de páginas livres modificadas, ou seja, página que foram escritas pelo processo. Neste caso, antes da página ser substituída por outra, esta deve ser copiada para a memória secundária.

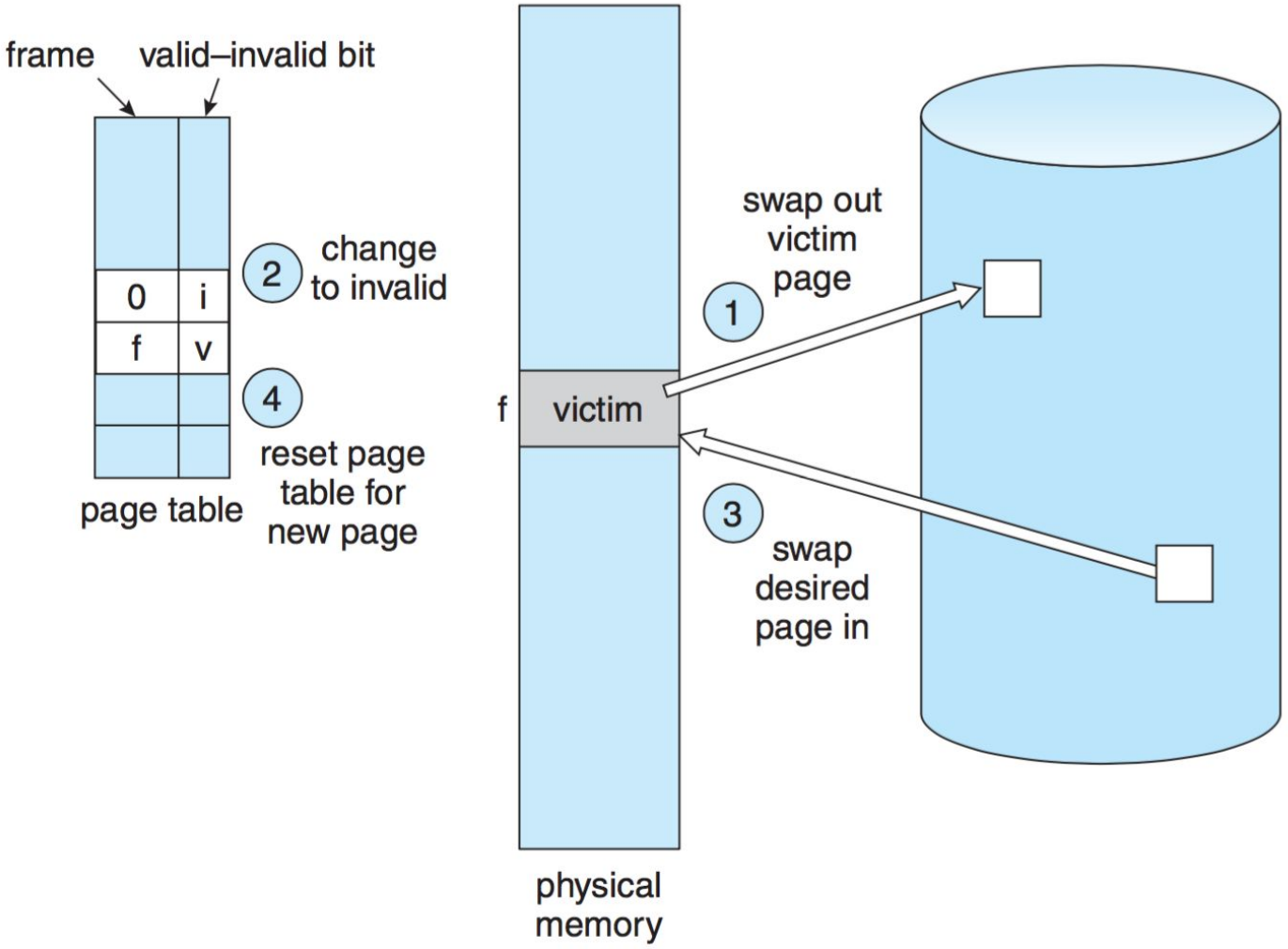
Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Quando ocorre um erro por falta de páginas, o SO retira a primeira página da lista de páginas livres.
  - Se o número de páginas livres ficar abaixo de um valor mínimo (*low water mark*), o SO deve acordar (colocar em execução) o paginador.
  - O paginador irá liberar páginas utilizando algum critério (*algoritmo de substituição de página*) de liberação até que o número de páginas livres suba até um valor máximo (*high water mark*).

Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Periodicamente a lista de páginas livres modificadas é escrita na memória secundária, então as páginas desta lista passam para a lista de páginas livres.
  - A escolha da página vítima, ou seja, a página que será substituída por outra, deve ser baseada em algum critério, de preferência que não prejudique a execução do programa, ou seja, que não degrade seu desempenho em virtude de uma alta demanda de paginação.

Algoritmos de Gerenciamento de Memória

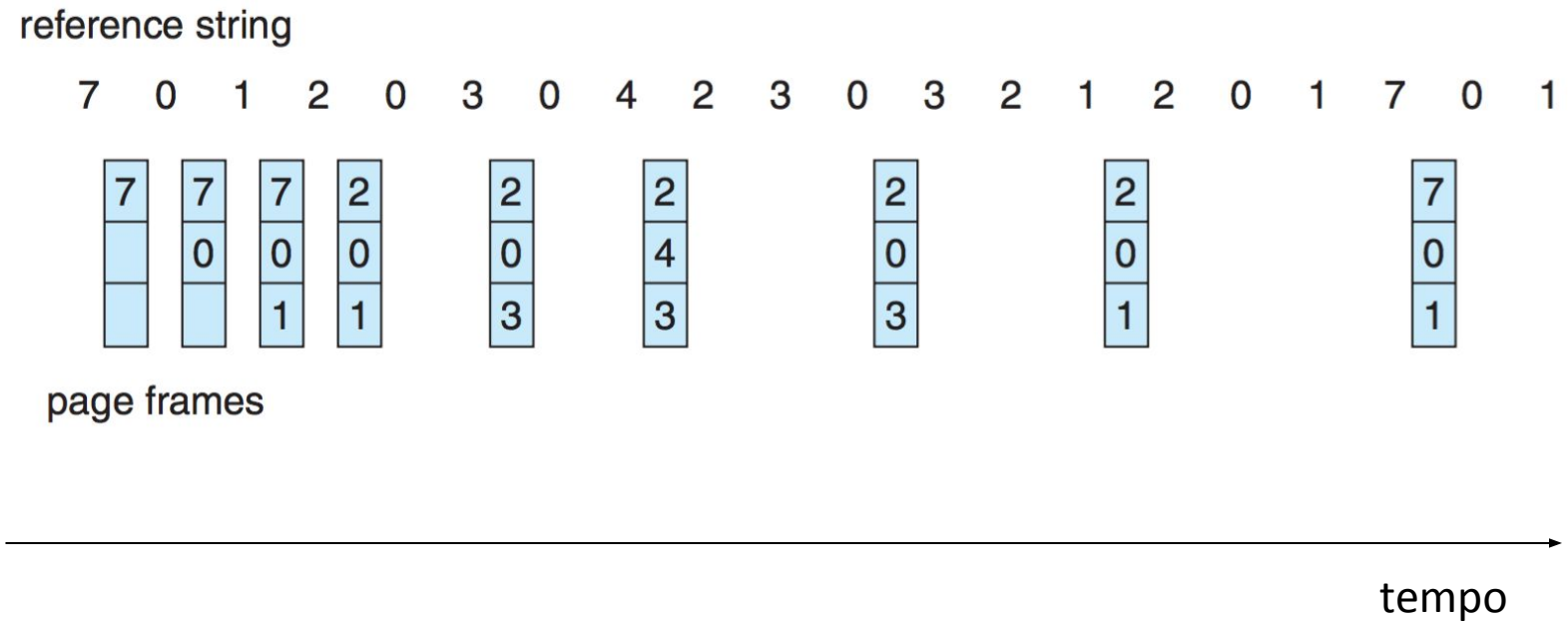


Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Algoritmos de Substituição de Páginas (1/6)
    - Ótimo
      - Retira-se da memória principal a página que não será utilizada pelo período mais longo
      - Requer prever o futuro, portanto, inviável na prática
      - Usado com referência de melhor performance (benchmark)

Algoritmos de Gerenciamento de Memória

- Exemplo: Ótimo (9 faltas)

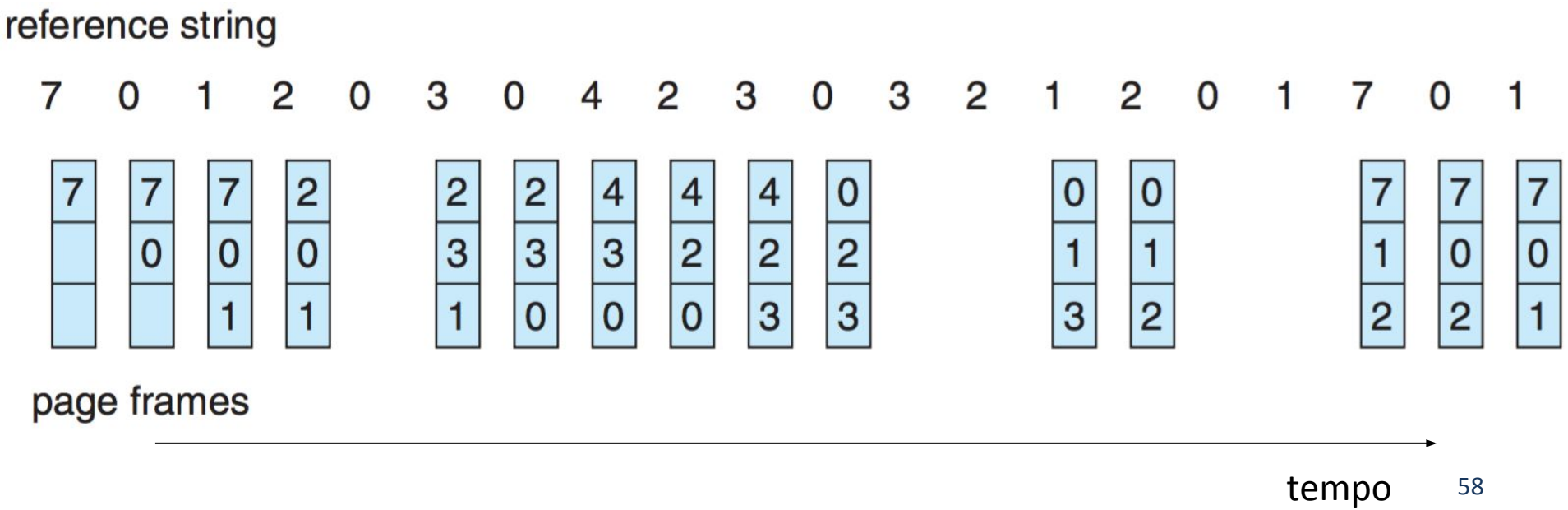


Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Algoritmos de Substituição de Páginas (3/6)
    - FIFO (*First In, First Out* – Primeiro a Entrar, Primeiro a Sair)
      - Considera o **passado** das páginas: instante em que a página foi trazida à memória
      - Simples: usa uma lista do tipo FIFO, onde a página a ser substituída ocupa a primeira posição da lista.
      - Toda página nova, que está sendo carregada na memória principal, é inserida no fim da lista.
      - Tende a degradar o desempenho de execução dos processos, uma vez que a página a ser substituída pode estar sendo constantemente utilizada.

Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Algoritmos de Substituição de Páginas (1/6)
    - FIFO: 15 faltas de página



FIFO

- Simples
- Pode remover página frequentemente usada
- Performance ruim
  - Raramente usado

Algoritmos de Gerenciamento de Memória

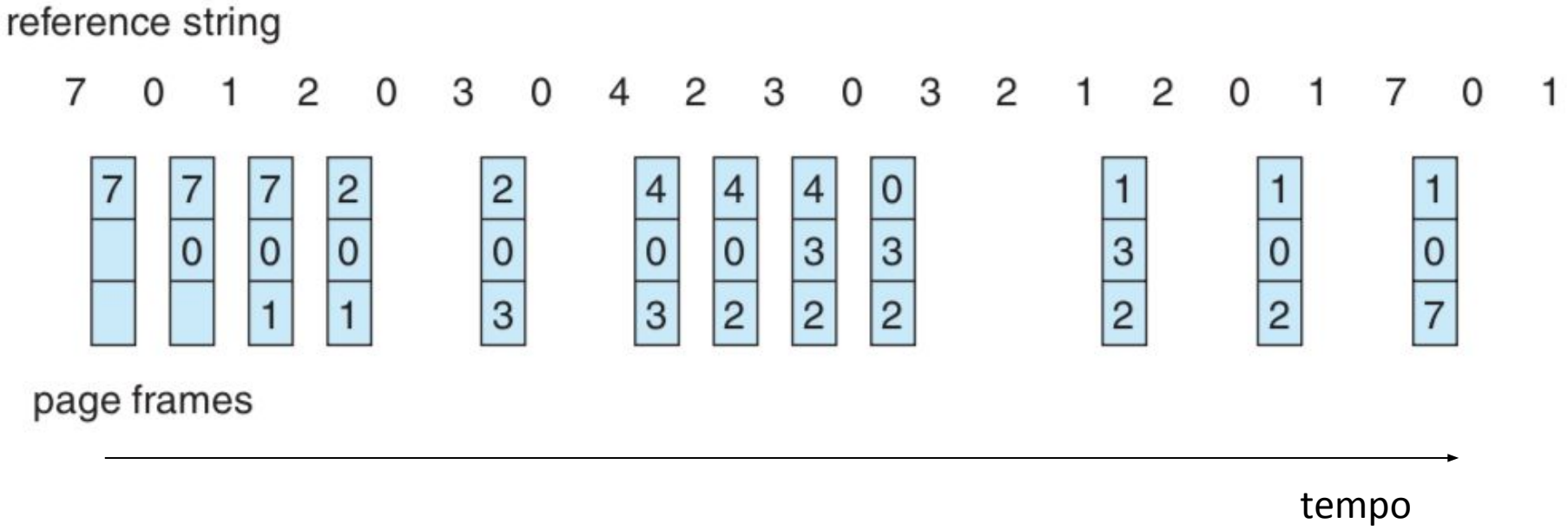
- Substituição de Páginas
  - Algoritmos de Substituição de Páginas (4/6)
    - Segunda Chance
      - É uma **melhoria** do algoritmo FIFO.
      - Permite alterar a posição de uma página muito antiga na lista que tenha sido referenciada (usada).
      - Se a página escolhida tiver o valor do bit R em 1, muda-se o valor para zero e então escolhe-se outra página.
      - Se a página escolhida tiver o valor do bit R em zero, efetua a substituição da página.

Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Algoritmos de Substituição de Páginas (1/6)
    - LRU (*Least Recently Used* – Menos Recentemente Usada)
      - Tentar identificar a página que será menos observando o passado
      - Substitui página cujo último uso é o mais antigo

Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Exemplo LRU: 12 faltas



Algoritmos de Gerenciamento de Memória

- Substituição de Páginas
  - Algoritmos de Substituição de Páginas (1/6)
    - LRU (*Least Recently Used* – Menos Recentemente Usada)
      - **Difícil de implementar exatamente.**
      - **Implementação via hardware**. Contador incrementado a cada instrução. Quando uma página é referenciada, o valor do contador é associado a página. A página de menor contador será a próxima página substituída.
      - **Implementação via software**. Há um contador inicialmente zerado para cada página. Periodicamente o SO soma o bit R da página ao respectivo contador.

LRU

- Hardware: raramente implementado
- Software: nunca esquece contador
  - Uma página pode ser inicialmente muito usado mas depois não

Não recentemente usada (NRU)

- Usa os bits referenciada (R) e modificada (M) da tabela de páginas
- Zera periodicamente (ex., a cada x interrupções de relógio) o bit R de todas as páginas
- Quando for substituir, inspeciona todas as páginas separando-as em 4 classes:
  - Classe 0: R = 0, M = 0
  - Classe 1: R = 0, M = 1
  - Classe 2: R = 1, M = 0
  - Classe 3: R = 1, M = 1
- Remove aleatoriamente uma página da classe de ordem mais baixa

Exemplo: somente 4 molduras de página

Moldura de página

7	11	0	0	1
6	00	0	0	0
5	00	0	0	0
4	00	0	0	0
3	00	0	1	1
2	01	1	1	1
1				
0				

Exemplo: somente 4 molduras de página

Moldura de página	R	M	A/P
7	10	0	1
6	11	0	1
5	00	0	0
4	00	0	0
3	00	0	0
2	00	0	0
1	00	0	1
0	01	1	1

Tabela de páginas

Nova página requisitada: 3

- **Classe 0**: 6, 7
- **Classe 1**: 1
- **Classe 2**: -
- **Classe 3**: 0

- **Página escolhida:** 6 ou 7 (aleatório)

Exemplo: somente 4 molduras de página

Moldura de página		R	M
7	10	0	0	1
6	11	0	0	0
5	00	0	0	0
4	00	0	0	0
3	00	0	0	0
2	00	0	0	0
1	00	0	1	1
0	01	1	1	1

Tabela de páginas

Nova página requisitada:

3

- **Classe 0**: 6, 7
- **Classe 1**: 1
- **Classe 2**: -
- **Classe 3**: 0

- **Página escolhida:**

Exemplo: somente 4 molduras de página

Moldura de página	R	M	A/P
7	10	0	0	1
6	11	0	0	0
5	00	0	0	0
4	00	0	0	0
3	11	1	0	1
2	00	0	0	0
1	00	0	1	1
0	01	1	1	1

Tabela de páginas

Nova página requisitada:

3

- **Classe 0**: 6, 7
- **Classe 1**: 1
- **Classe 2**: -
- **Classe 3**: 0

- **Página escolhida:**

NRU

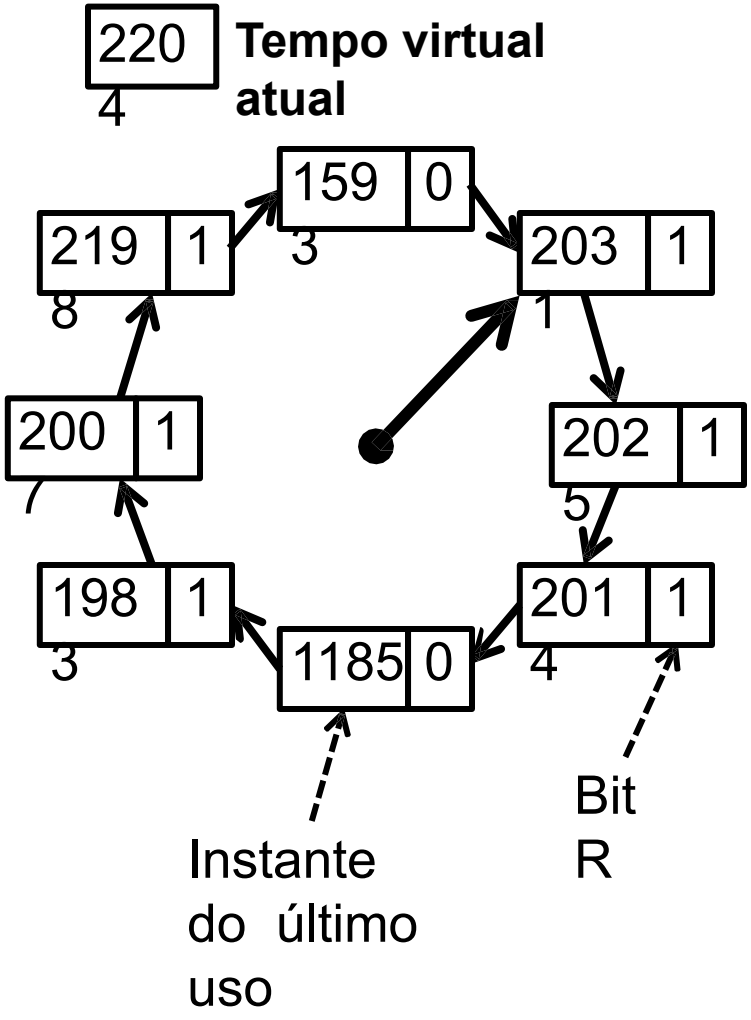
- Simples
- Performance adequada

WSClock

- **Conceitos**
  - **Localidade de referência**: processos só referenciam uma fração pequena de páginas em um determinado espaço de tempo
  - **Espaço de trabalho**: conjunto de páginas que um processo está usando atualmente na memória

WSClock

- Elementos algoritmo
  - T: intervalo de base (equivale a várias interrupções de relógio)
  - Tempo virtual atual: quant. de tempo de CPU que o processo usou desde que iniciou
  - Instante do último uso: instante do tempo virtual no qual a moldura de página foi referenciada pela última vez
  - Bits R e M



WSClock

- Funcionamento do algoritmo WSClock
  - Quando ocorre uma falta de página, a página que estiver sendo apontada é examinada primeiro
  - Se **R = 1**, ela foi referenciada antes da interrupção de relógio atual, então não é candidata a remoção
    - **Zera o bit R, copia o tempo virtual atual para o instante do último uso e avança o ponteiro**
  - Se **R = 0**
    - **idade** = tempo virtual atual - instante do último uso
    - Se **idade > T e M = 0**, substitui a página
    - Se **idade > T e M = 1**, escalona a escrita da página no disco e avança o ponteiro (ao final da escrita: **M = 0**)
  - Se o ponteiro deu uma volta completa
    - **Se pelo menos uma escrita foi escalonada**, o ponteiro poderá encontrar uma página "limpa" (M = 0), logo, a removerá
    - **Se nenhuma escrita foi escalonada**, remove qualquer página "limpa" (M = 0) ou aleatoriamente se não existir nenhuma

WSClock

Algoritmo	Características Principais
Ótimo	Teórico. Serve com comparação.
FIFO	Pode fazer má escolhas. Performance ruim
Segunda chance	Melhora do FIFO
LRU	Excelente mas difícil de implementar exatamente
NRU	Fácil de implementar, performance adequada
WSClock	Boa perfomance

Endereços Reais e Virtuais

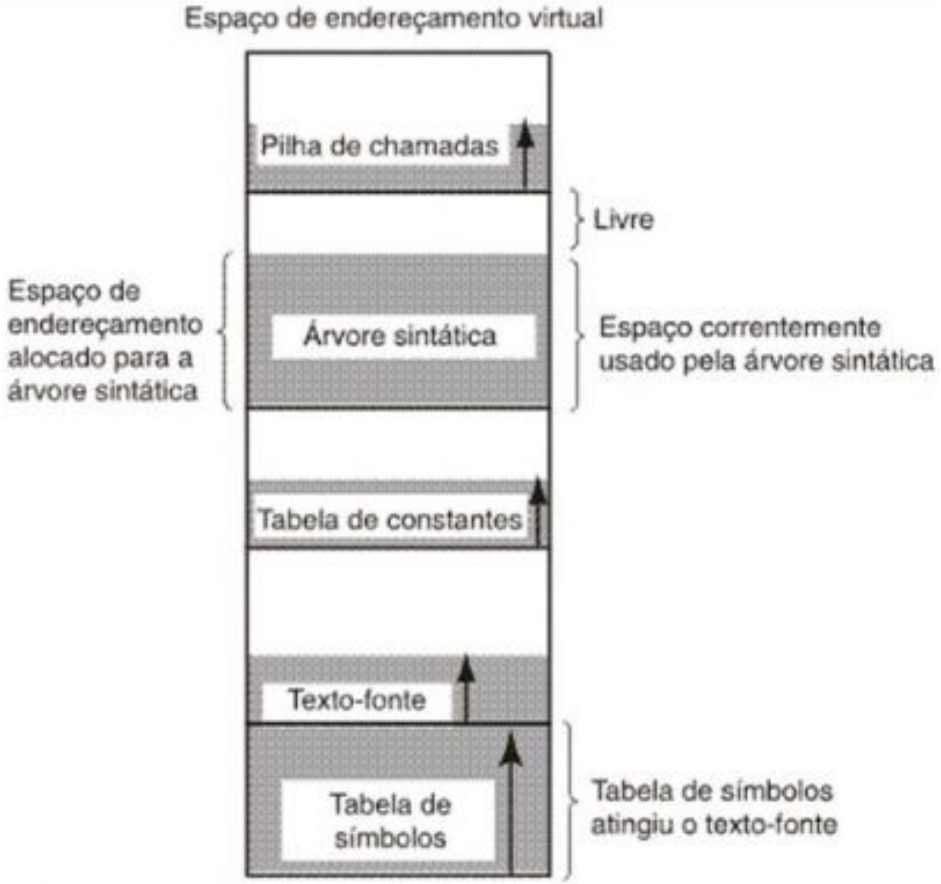
- Endereçamento Virtual
  - Paginação utiliza blocos de tamanho fixo
    - Existe fragmentação interna
    - Dificuldade de manipular estruturas de dados que mudam de tamanho dentro dos blocos (ex: a estrutura de dados “avança” sobre outros dados dentro do bloco)

Endereços Reais e Virtuais

- Exemplo: Compilador e seus componentes
  - Código fonte sendo analisado
  - Tabela de símbolos (nomes x atributos)
  - Tabela de constantes
  - Árvore de análise sintática
  - Pilha para chamada de funções

Endereços Reais e Virtuais

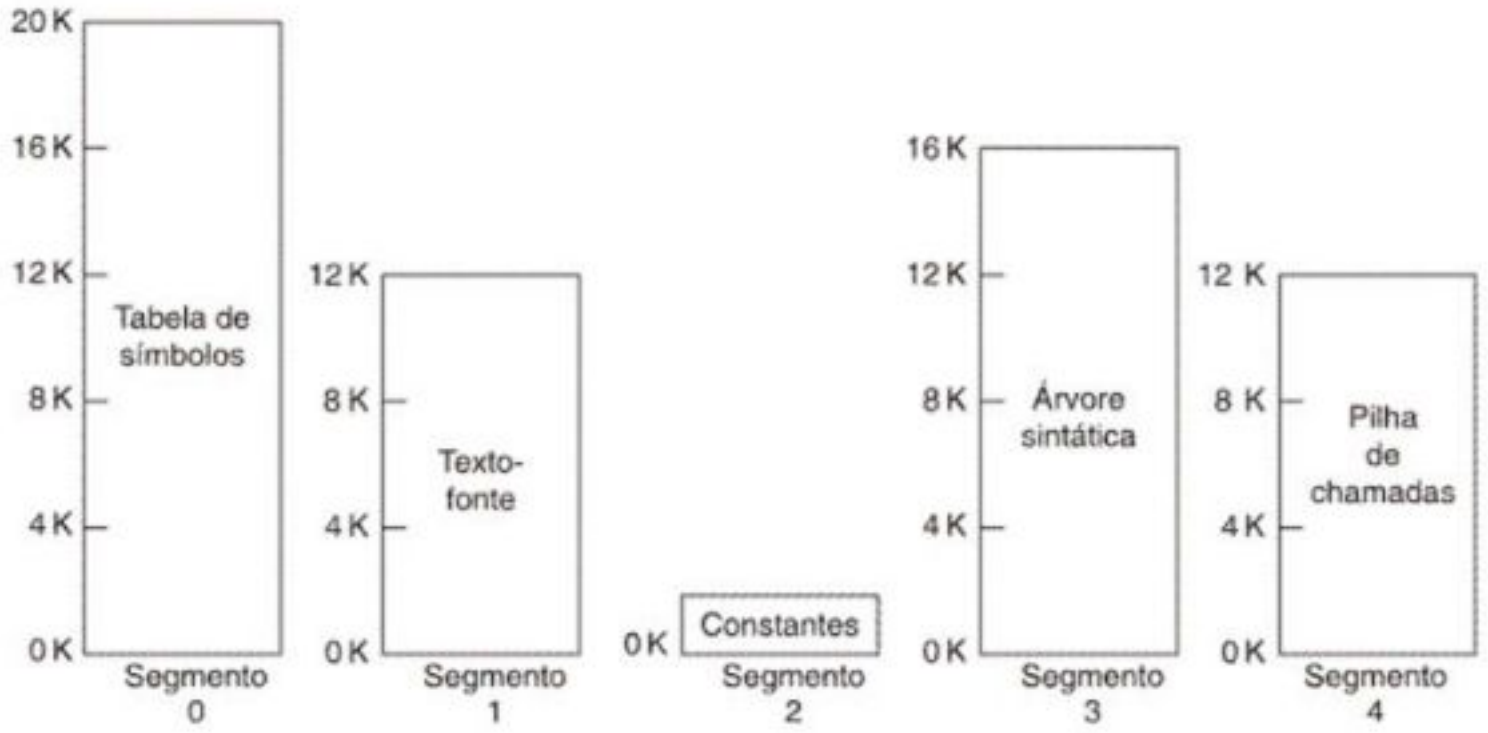
- Exemplo: Compilador e seus componentes



Necessidade de separar componentes em blocos distintos

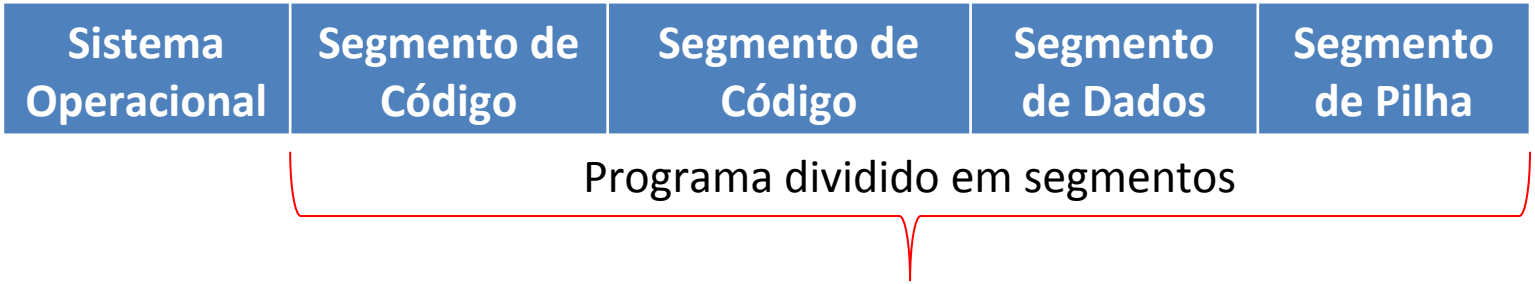
Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação: blocos de tamanho distinto e dinâmico



Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação



Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação
    - Os seguintes aspectos devem ser considerados na segmentação:
      - Carregamento em memória: o segmento é a **unidade mínima** a ser carregado em memória.
      - Proteção: impede um processo acessar dados do SO ou de outros processos
      - Eficiência: princípio da localidade, ou seja, ao ser acessado um endereço de um segmento existe uma grande probabilidade de que os próximos acessos sejam próximos a esse endereço.

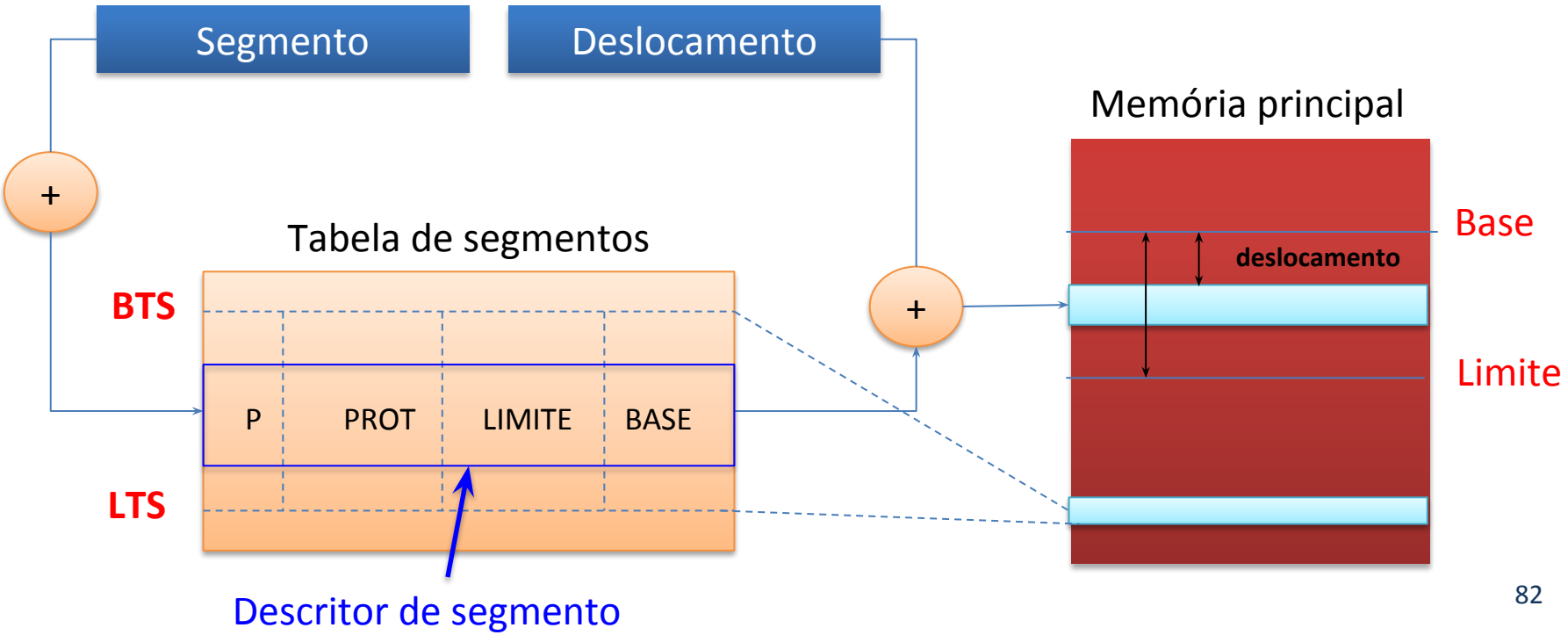
Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação
    - Endereço lógico

< #segmento, deslocamento>

Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação
    - Tabela de Segmentos: end. lógico end. real



Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação
    - Tabela de Segmentos
      - Cada linha é um descritor de segmento.
      - Base: endereço físico onde começa um segmento.
      - Limite: endereço físico onde termina um segmento.
      - PROT: proteção (ex. escrita/leitura e nível de acesso necessário)
      - P: segmento está na memória principal
      - O registrador de BTS (Base da Tabela de Segmentos) e LTS (Limite da Tabela de Segmentos), contém respectivamente o endereço real do início e do fim da tabela.
      - Quando o programa gera um endereço lógico, o número do segmento é multiplicado pelo número de bytes do descritor e somado com o registrador BTS, obtendo-se a entrada na tabela de segmentos correspondente a este segmento.

Algoritmos de Gerenciamento de Memória livre

- Alocação de Segmentos
  - A alocação de segmentos é mais difícil do que a alocação de páginas.
  - Os segmentos podem ter tamanhos distintos
  - O tamanho de um segmento pode mudar
  - Segmento é carregado integralmente na memória
  - O SO controla os blocos livres em memória mantendo uma lista que armazena o endereço do bloco e o seu tamanho.

Algoritmos de Gerenciamento de Memória de livre

- Alocação de Segmentos
  - **First-Fit**: pesquisa na lista até encontrar o primeiro segmento livre com tamanho suficiente.
  - **Next-fit**: variação do first fit, porém começando a pesquisa de onde parou da última vez.
  - **Best fit**: encontra o segmento livre cujo tamanho é o menor tamanho suficiente
  - **Worst fit**: encontra o maior segmento livre

Algoritmos de Gerenciamento de Memória de memória livre

- Alocação de Segmentos
  - Best fit é mais lento que first fit e resulta em mais memória perdida (buracos pequenos inutilizáveis) que o first fit e next fit
    - First fit gera, na média, buracos maiores
  - Best fit e worst fit precisam de listas ordenadas (custo adicional)

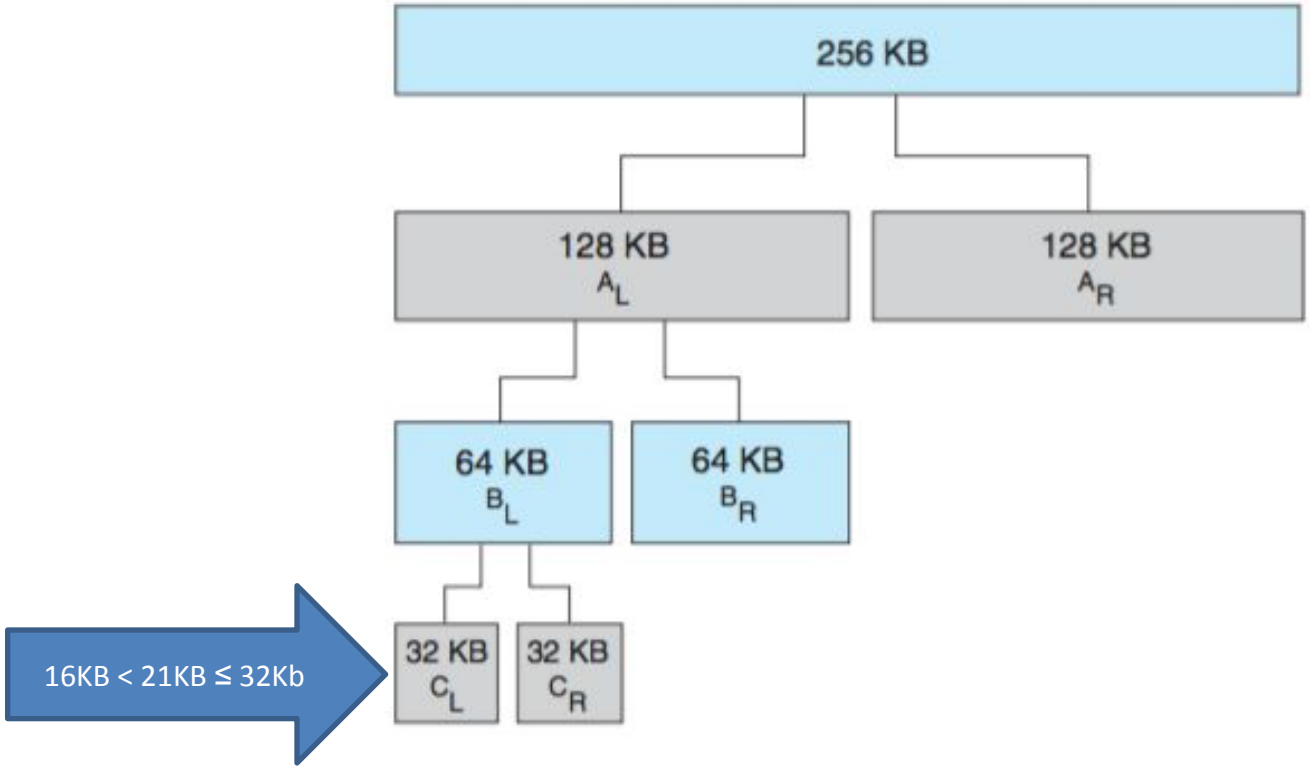
Algoritmos de Gerenciamento de Memória livre

- Alocação de Segmentos
  - **Buddy**: organiza a memória em blocos de tamanho b^n contíguos, normalmente b = 2.
    - Para satisfazer um pedido de tamanho T, busca pelo menor segmento livre T' com tamanho $2^{k.}$

```
Buddy(T, T') {
  se tamanho(T) > tamanho T' retorne NULL
  enquanto tamanho(T) ≤ tamanho(T')/2 < tamanho(T')
      T' = T'[0,..., tamanho(T')/2 -1]
  retorne T'
}
```

Algoritmos de Gerenciamento de Memória

- Exemplo de Buddy: alocar 21KB



Algoritmos de Gerenciamento de Memória

- Buddy
  - Vantagem: ao desalocar um segmento, pode-se uni-lo ao irmão até a raiz da árvore
  - Desvatagem: fragmentação interna ao alocar segmentos de tamanho 2^k

Algoritmos de Gerenciamento de Memória

- A operação de transferência é realizada quando:
  - Um processo necessita de um determinado segmento ou página que não foi alocado na memória principal.
  - Os algoritmos de transferência podem atuar de três maneiras distintas:
    1. A pedido (*on request*): o programa ou o SO invoca uma chamada de sistema que permite carregar outro bloco.
    2. Por necessidade (*on demand*): um programa necessita de um bloco de memória que ainda não foi carregado na memória (exceção por falta de segmento ou página).
    3. Por antecipação (*pre-fetching*): o bloco é carregado na memória principal antes de ser solicitado (acessado).

Algoritmos de Gerenciamento de Memória

- Transferência e Substituição de Segmentos
  - Quando um processo é criado é possível que não haja espaço na memória primária para conter os seus segmentos.
  - Os segmentos que não cabem na memória ficam em uma área na memória secundária chamada de área de transferência (*swap*).
  - Ocorre então a substituição de segmentos, quando o sistema precisa decidir quais segmentos serão transferidos para a memória secundária (swap out) para liberar espaço para mover outros segmentos para a memória principal (swap in)

Algoritmos de Gerenciamento de Memória

- Substituição de Segmentos
  - Critérios para escolher segmentos a serem substituídos:
    1. Estado e prioridade: geralmente processos em estado de aguardando ou menos prioritários são os primeiros candidatos a irem para a memória secundária.
    2. Tempo de permanência dos segmentos na memória primária: segmentos devem permanecer na memória principal por um determinado tempo, o suficiente para o processo poder usá-lo sem necessitar trocas com a memória secundária.
    3. Tamanho do segmento: o segmento escolhido para ser *swap out* deve ter tamanho suficiente para liberar uma determinada quantidade de memória suficiente para alocar os segmentos *swap in*.

Algoritmos de Gerenciamento de Memória

- Transferência de Segmentos
  - Quando há a transferência de segmentos entre a memória primária e a secundária o segmento inteiro é transferido
  - Alguns sistemas, quando há falta de memória principal, transferem todos os segmentos os segmentos de um o programa para a memória secundária
  - A transferência de um programa da memória principal para a memória secundária é chamada de swap, ou seja, o programa foi *swap out*.

Endereços Reais e Virtuais

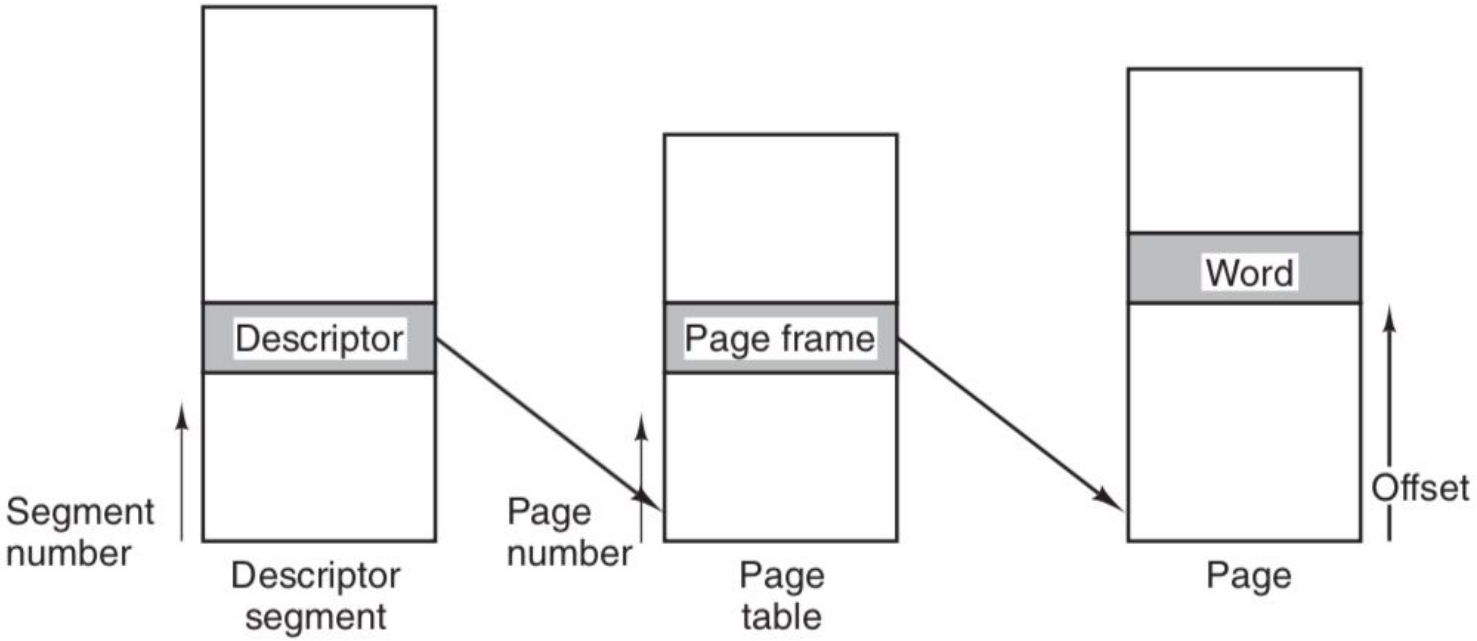
- Endereçamento Virtual
  - Segmentação: um segmento pode crescer ao ponto de não caber na memória
  - Segmentação paginada: dividir segmento em páginas e carregá-las conforme necessário

Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação Paginada
    - O programa é dividido em segmentos lógicos e cada segmento é dividido em páginas.
    - Um endereço de memória é formado por: segmento + página + deslocamento.
    - O número do segmento é utilizado para obter o endereço físico do início da tabela de páginas do segmento, sendo a tradução do par página-deslocamento feita como no sistema de paginação.

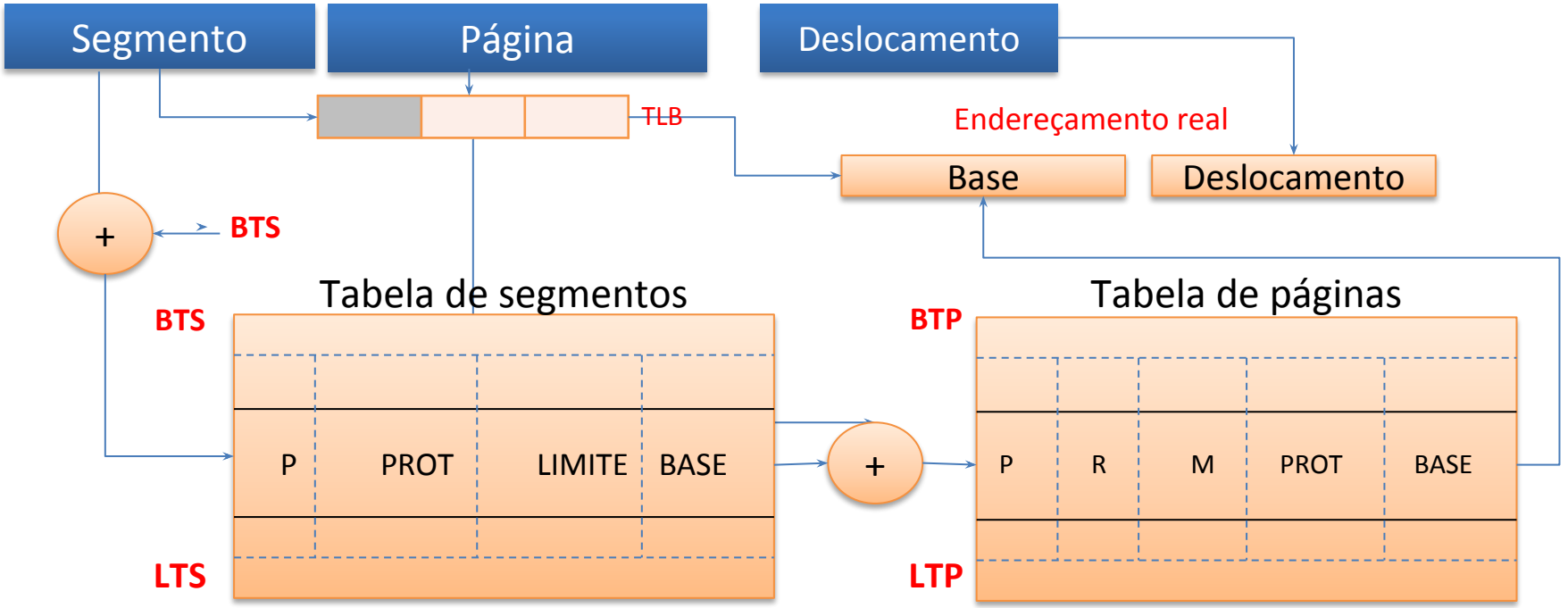
Endereços Reais e Virtuais

- Endereçamento Virtual
  - Segmentação Paginada



Endereços Reais e Virtuais

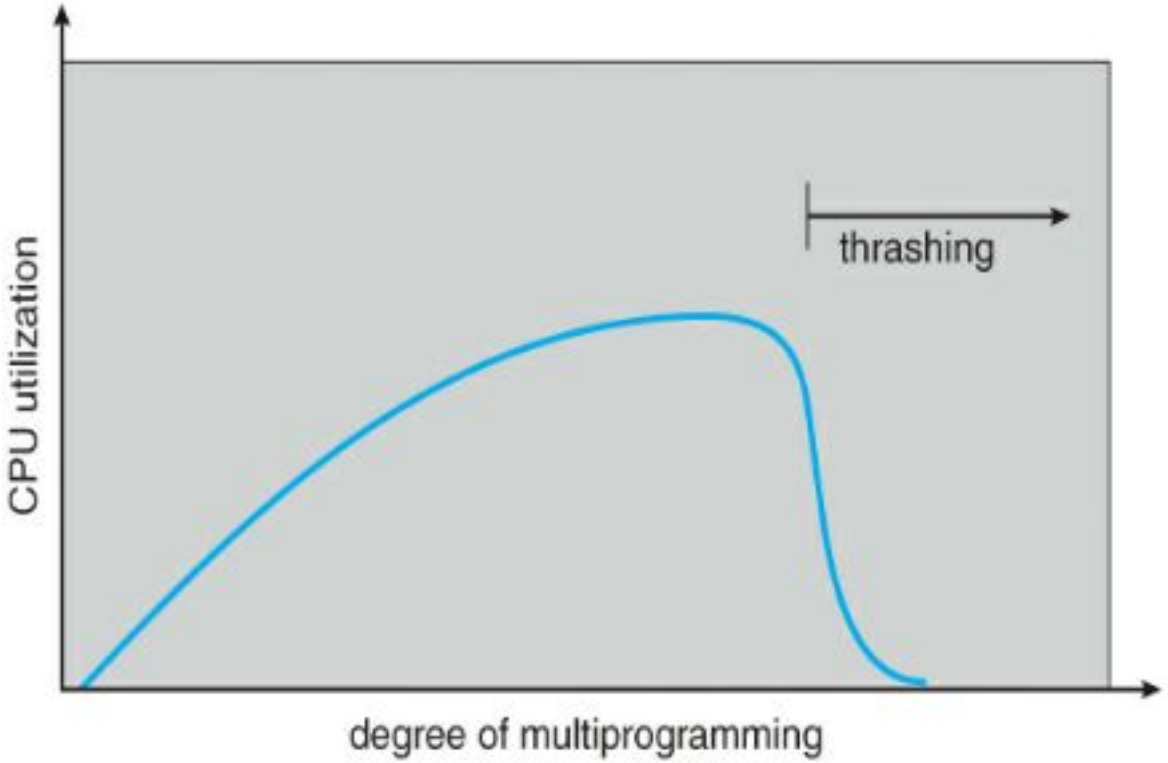
- Endereçamento Virtual
  - Segmentação Paginada
    - Tabela de Segmentos/Páginas com TLB



Trashing

- Ocorre quando um processo tem um espaço de trabalho maior que a quantidade de frames que ele pode utilizar
- Uma página em uso cede o frame para outra
- Processo passa maior parte do tempo esperando swap in/out
- SO percebe ociosidade da CPU e aumenta nível de multiprogramação

Trashing

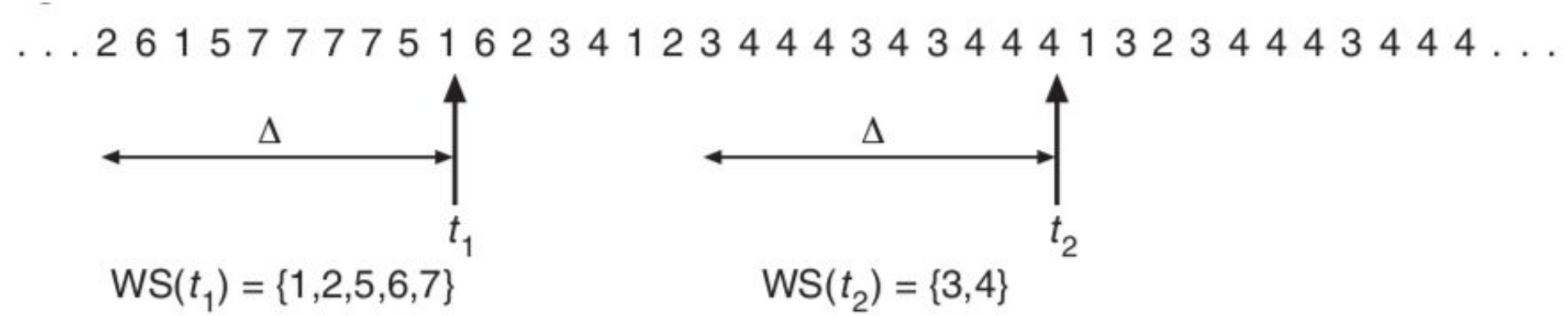


Trashing: Solução 1

- Passos:
  - 1. Fazer swap out de alguns processos para o disco para liberar molduras de página
  - 2. Alocar as molduras de página livres aos processos que precisam de mais memória
  - 3. Repetir os passos 1 e 2 até que o problema de thrashing seja resolvido
- Necessário fazer swap in / swap out de processos periodicamente
  - Levar em consideração o tipo de processo (CPU- e I/O-bound) na escolha dos processos

Trashing: Solução 2

- Janela Espaço de trabalho
  - Prover ao processo a quantidade de frames que ele precisa
  - Identificar conjunto de páginas referenciadas WS() nos últimos Δ segundos
  - Suspender processos se a demanda total for maior que a total de frames possíveis



Trashing: Solução 3

- Frequência de falta de páginas
  - Melhor solução entre as 3