Gerenciamento de memória
Memória Virtual

Memória Virtual e Paginação

Requisitos

- referências em um processo são endereços lógicos
  - mapeamento para endereço real durante a execução devido a *swap*

- um processo é dividido em páginas não alocadas em MP continuamente
  - tabelas de páginas e mapeamento de endereços dinamicamente permitem a execução do processo

Memória Virtual e Paginação

- Somente parte do processo é trazido para MP
  - devido aos requisitos
  - princípio da localidade

Como funciona?

- O conjunto de páginas de um processo presentes na MP é chamado de conjunto residente
  - inicialmente, algumas páginas são carregadas a partir da página 0

- Quando é necessário um endereço que não está presente na MP uma interrupção é gerada
  - SO coloca processo em estado bloqueado
  - SO faz pedido de E/S para trazer mais página(s)
  - enquanto isso, SO escolhe outro processo para executar

e ...

- Quando a página é trazida para a memória, o primeiro processo (o que ocasionou falta de páginas)
  - passa para a fila dos prontos
  - controlador envia interrupção (evento realizado)

- Transparência para usuário

- Memória virtual – mais vantajoso: MP + MS

Vantagens desta divisão

- Mais processos (seus pedaços!) podem estar na MP
  - maior eficiência pois maior chance de um deles estar em estado pronto
- Processos podem ser maiores que a MP (tão grandes quanto a MS)
  - limite: tamanho do MS
  - tarefa do SO e hw trazer partes do processo
- Reduz o tempo de *swapping*
  - não é toda a imagem que está em MP
  - somente páginas requisitadas são carregadas

Thrashing

- MP contém páginas de diferentes processos
- páginas são carregadas, outras são *swapped out*
  - Possibilidade de enviar para MS uma página de outro processo logo antes desta ser utilizada
- O processador pode gastar a maior parte do tempo fazendo *swapping* em vez de processando instruções do usuário
- Como evitar:
  - tentando adivinhar qual página será mais necessária

Princípio da localidade

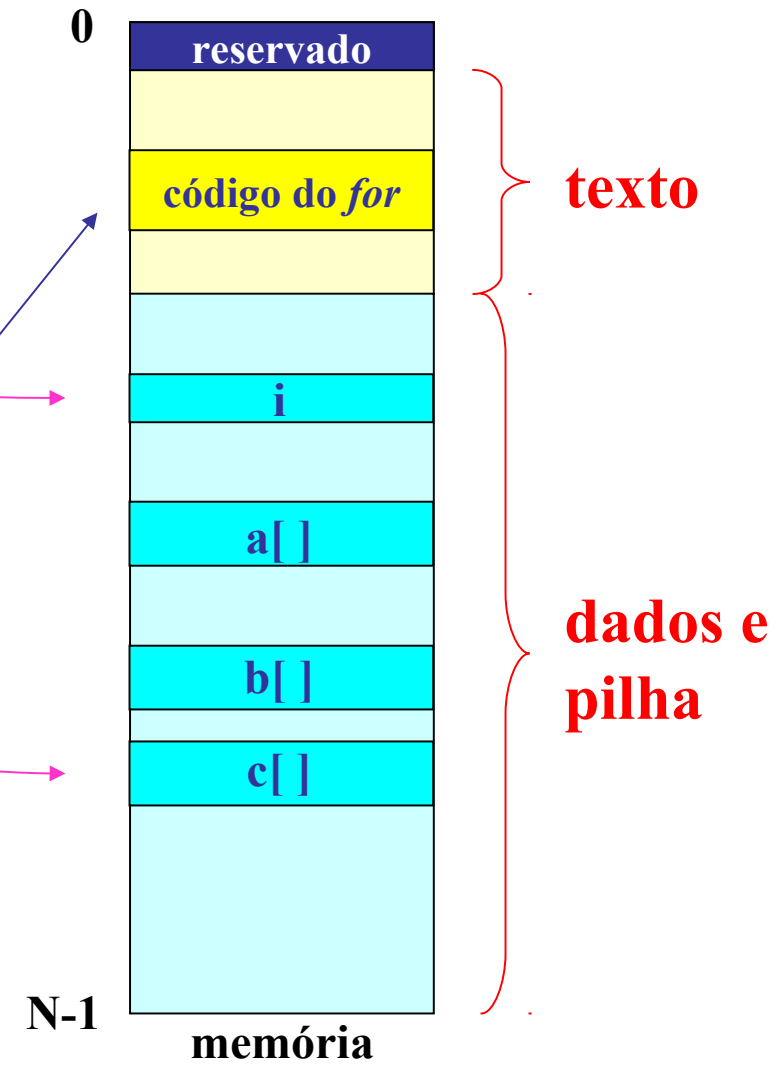
- Só partes do processo serão utilizadas em um dado intervalo de tempo
- Localidade espacial e temporal
- Palpites inteligentes podem ser feitos quanto aos pedaços que serão necessários no futuro próximo
- memória virtual pode funcionar eficientemente

Localidade

- Localidade de referência (P. Denning, 1968):
  - localidade espacial:
    - se um item é referenciado, itens com endereço próximo tendem a ser referenciados em seguida
  - localidade temporal:
    - se um item é referenciado, ele tenderá a ser referenciado novamente em breve

Exemplo de localidade

```
exemplo ( ... )
{
        ...
        for (i=1; i<N; i++) {
                a[i] = b[i] + c[i];
                b[i] += 2;
        }
        ...
}
```



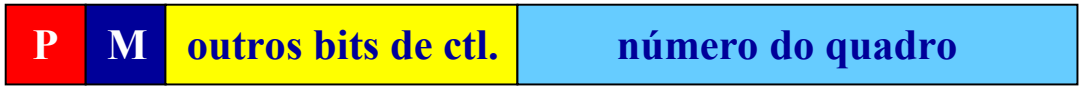
Paginação – Memória Virtual

- Cada processo tem sua tabela de páginas (TP)
- TP: mapeamento página x quadro
- Bit de presença
  - 0 – página não está em MP
  - 1 – está em MP
- Bit de modificação
  - 0 – página em MP não foi modificada
  - 1 – foi modificada

Paginação: endereçamento

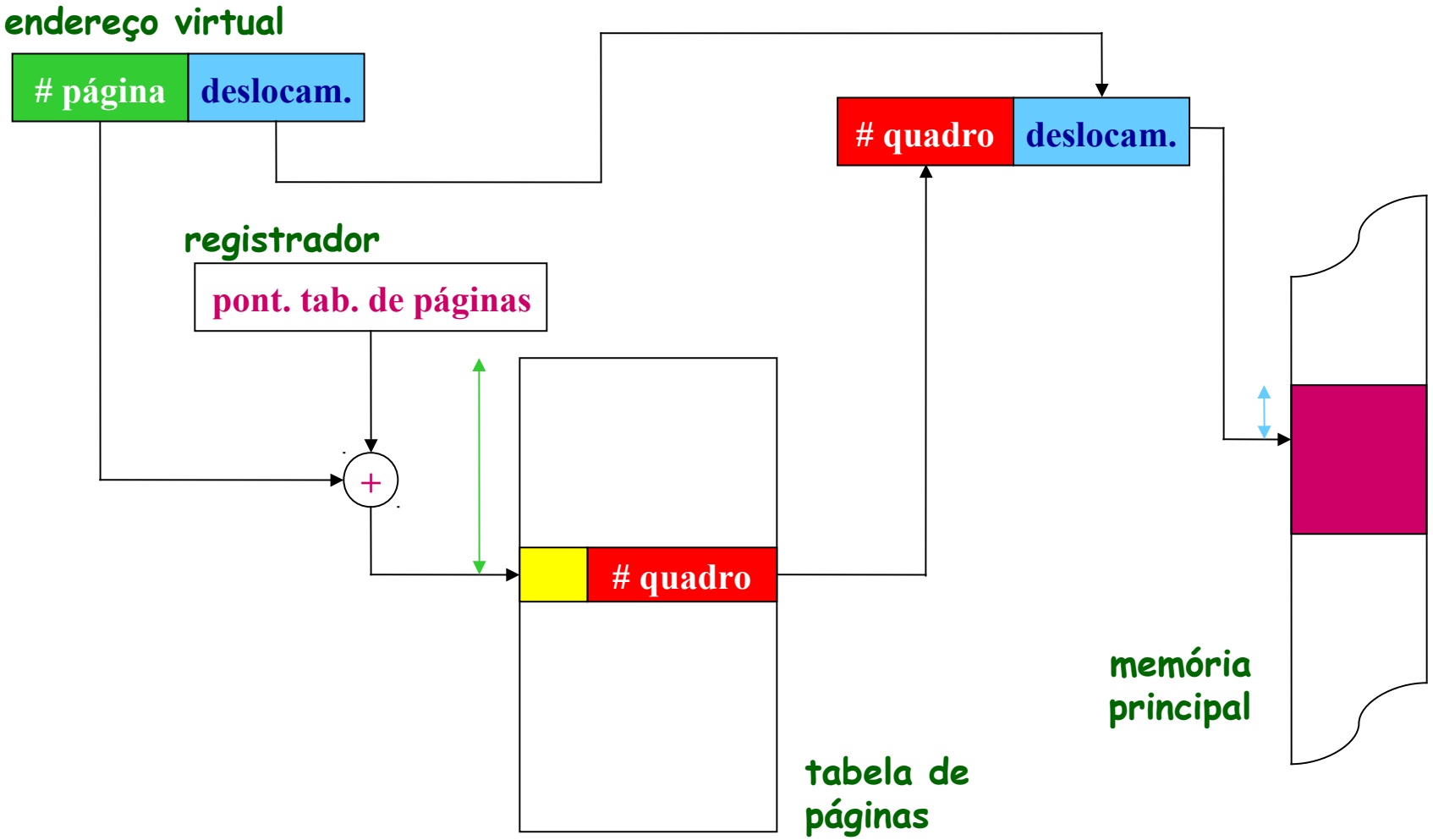
endereço virtual

linha da tabela de páginas

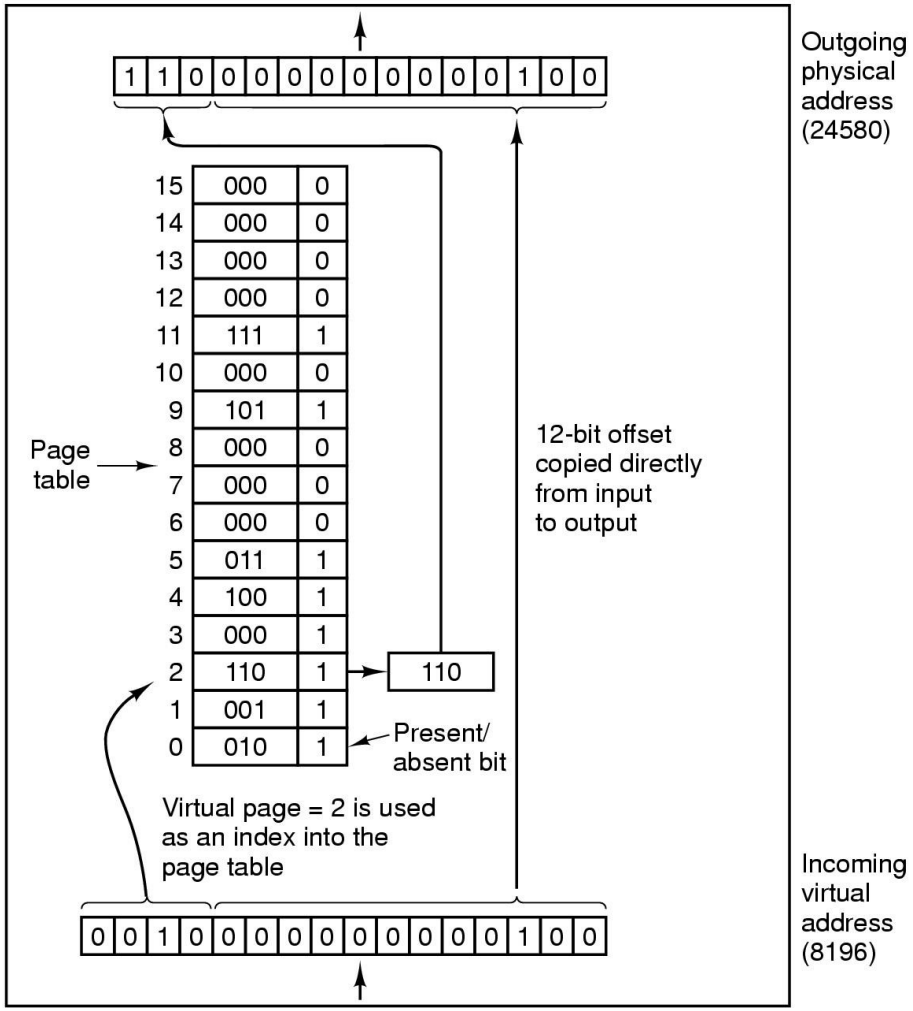


e.g., referenciada, proteção, compartilhamento, desabilita colocação na *cache*, etc.

Paginação: endereçamento



Paginação: exemplo



Estrutura da Tabela de Páginas

- número de entradas – depende do tamanho do processo
- onde armazenar?
  - registradores
  - MP
  - MV
- em dois níveis
  - uma tabela de X entradas
  - cada uma aponta para uma tabela de Y entradas

Estrutura da Tabela de Páginas

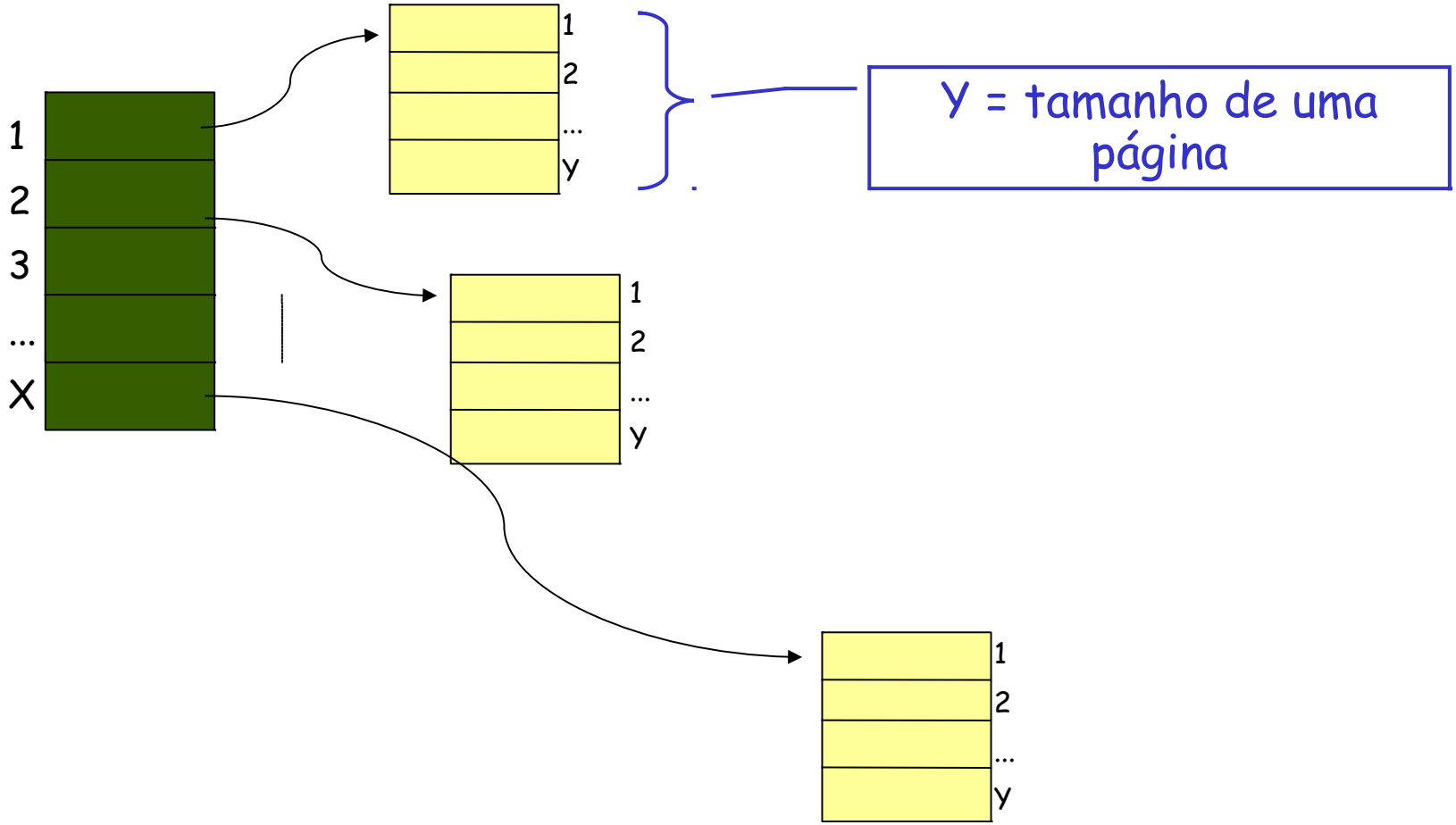
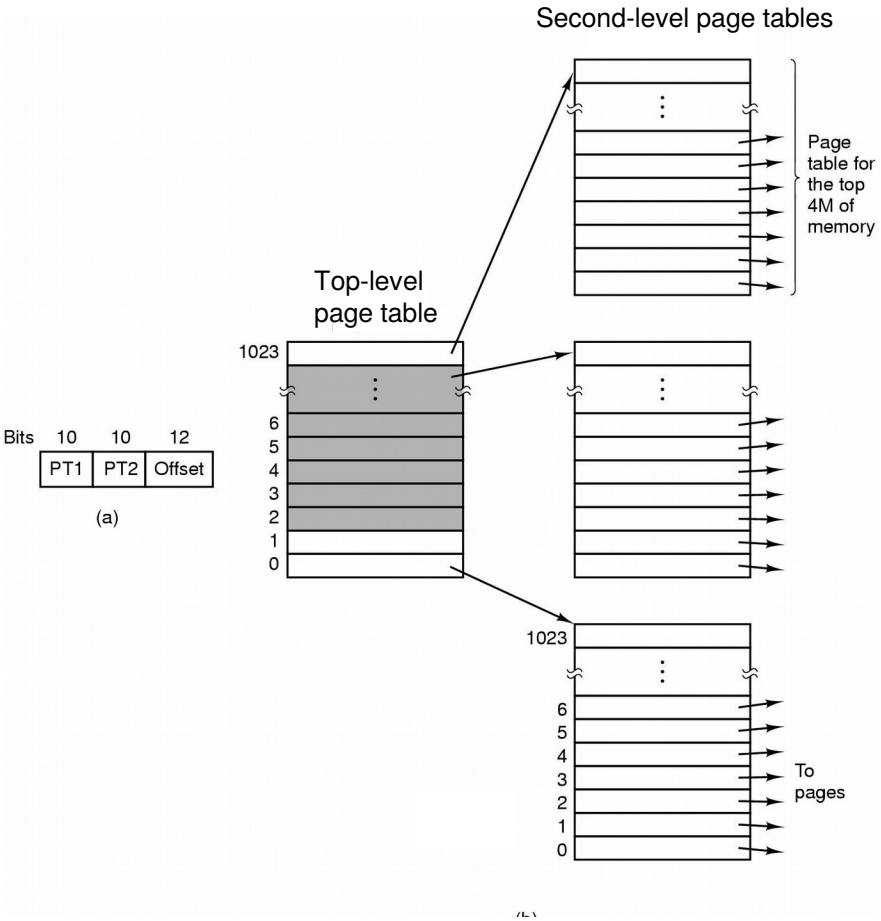


Tabela de páginas em 2 níveis



Translation Lookaside Buffer (TLB)

- para acessar MP com paginação e MV são necessários dois acessos à memória
  1) acesso a TP
  2) acesso ao endereço real
- Para diminuir o gargalo: TLB

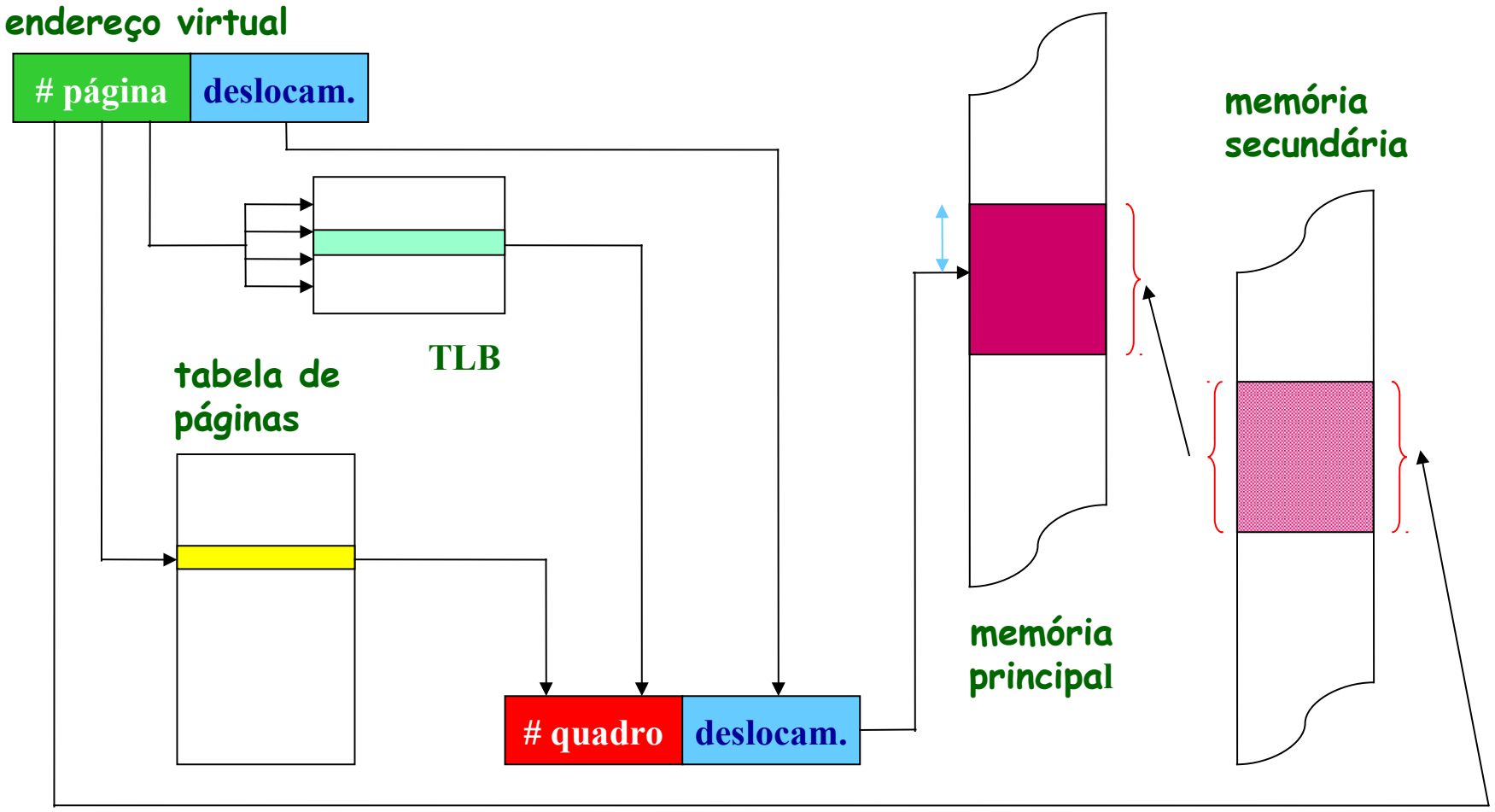
  cache especial para tabela de páginas

- *TLB contém as entradas da TP mais utilizadas*

Translation Lookaside Buffer (TLB)

- Pedido de acesso a endereço lógico:
  - no da pág. no endereço - entrada naTP
  - a TLB é consultada
- se está na TLB acessa a info de qual frame
- senão
  - acessa a TP na MP
  - se P == 1 então
    - acessa a info de qual frame
    - traz esta entrada da TP para TLB
  - senão *page fault*

Paginação: TLB



Entrada da TLB

- Procura na TLB: hardware especial
  - olha todas as entradas em paralelo
  - cache associativa

- Pergunta: Como funciona a TLB+MV e cache + MP?

Memória associativa: TLB

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Tamanho da Página

- é importante para evitar fragmentação interna
- o menor possível, mas...
  - muitas páginas por processo → TP longa
  - MV para tabela de páginas
  - pode gerar duas faltas de páginas:
    - de entradas de tabela de páginas
    - da página do processo caso não esteja em MP

Tamanho da Página

- também está relacionado com o tamanho da MP
- deve-se evitar falta de páginas
- atualmente, os próprios programas dos usuários estão mais modulares

Memória virtual: suporte de *s/w*

- Políticas:
  - busca
  - onde colocar pedaços na MP
  - que páginas retirar
  - tamanho do conjunto residente
  - política de limpeza
  - controle de carregamento
  - que processo suspender

Tamanho do conjunto residente

- quantas páginas de um processo devem ser trazidas para MP
- quanto menos páginas, mais processos estarão em MP
  - grau de *swaping out* de processo pode ser menor
  - mas mais falta de páginas de um processo
- também pode ser limitado pelo princípio da localidade

Tamanho do conjunto residente

Alocação fixa

- cada processo recebe um número fixo de quadros
- em caso de falta de páginas, uma das residentes é trocada

Alocação variável

- número de páginas varia durante a execução do processo
- a substituição pode englobar páginas de outros processos

Adaptativo

- se muita falta de páginas então aumenta o nº de frames/processo, senão diminui
- pode ser muito custoso gerenciar essa adaptação

Escopo da substituição

Local

- somente páginas do processo são substituídas
  - alocação fixa
  - fácil implementação

Global

- qualquer frame é candidato
- desempenho melhor, apesar de overhead maior
- um processo aumenta com número de páginas em MP, outro diminui (ou até processo sai de MP)

Escopo de Substituição - Combinações

Alocação fixa, escopo local

- problema: prever o melhor número de páginas por processo no momento de carga que tal que o número de falta de páginas seja mínimo

Alocação variável, escopo global

- muito adotado
- parte de um número inicial de frames alocado a um processo

Alocação variável, escopo local

- de tempos em tempos, o número de frames é realocado, mas a substituição é só local

Política de troca/substituição

- Trata da seleção da página a ser retirada da MP
- Algumas páginas podem ficar permanentemente em memória:
  - estruturas do núcleo
  - *buffers* de E/S
  - SO de tempo real

Política de troca/substituição

Questões que devem ser consideradas

- número de quadros/processo
- escopo local diminui o grau de multiprogramação
- das páginas candidatas, qual escolher

Políticas de substituição

- ideal: escolher um quadro que contem uma página que não será mais referenciada
- geralmente baseada em histórico passado

Política de substituição - Ótima

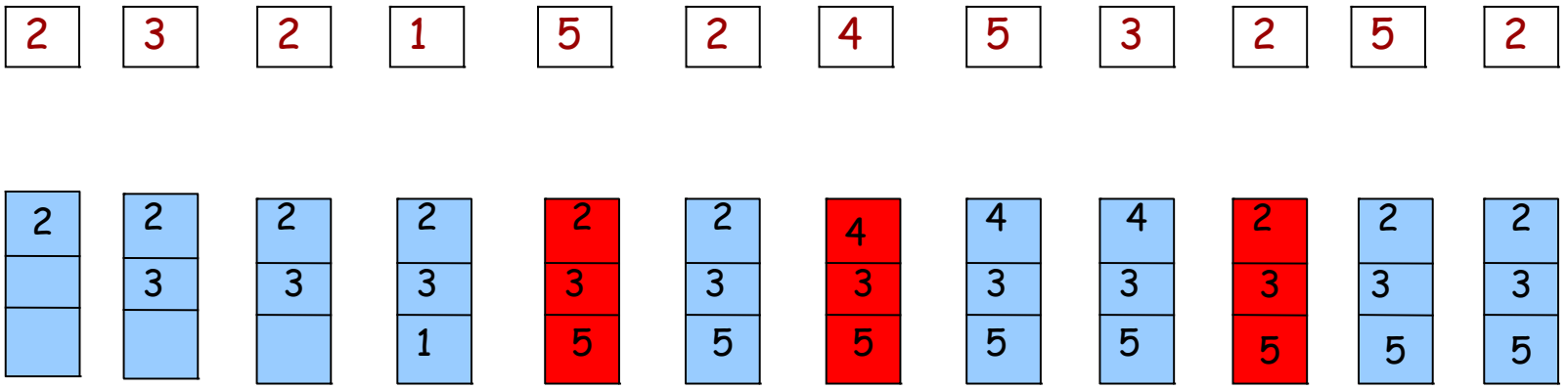
Política ótima

- seleciona a página cujo tempo para o próximo acesso será o mais longo (comparação)
- menor # de falta de páginas
- difícil implementação. Como adivinhar

Política de substituição - Ótima

Política ótima

- pág. referenciadas: 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2



Política de substituição - LRU

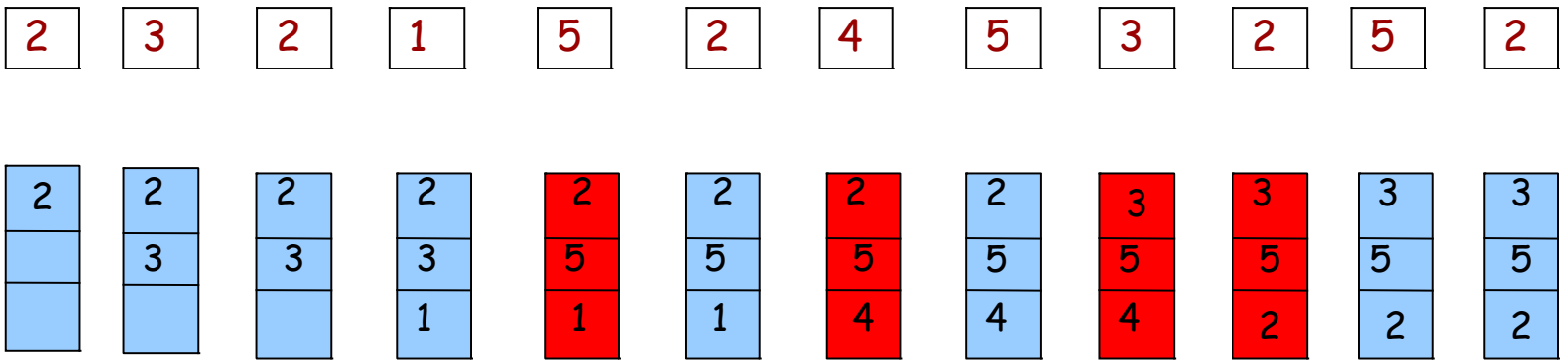
LRU (Least Recently Used)

- pelo princípio da localidade deve ser a de menor probabilidade de ser acessada. Implementação: etiqueta de tempo
- como guardar o tempo? (mais info)

Política de substituição - LRU

LRU (Least Recently Used)

- pág. referenciadas: 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2



Política de substituição – LRU e MRU

LRU

- tem desempenho ruim quando aplicações que requerem acesso seqüencial: vetores e matrizes
- variante: *most recently used* (MRU)

Política de substituição - FIFO

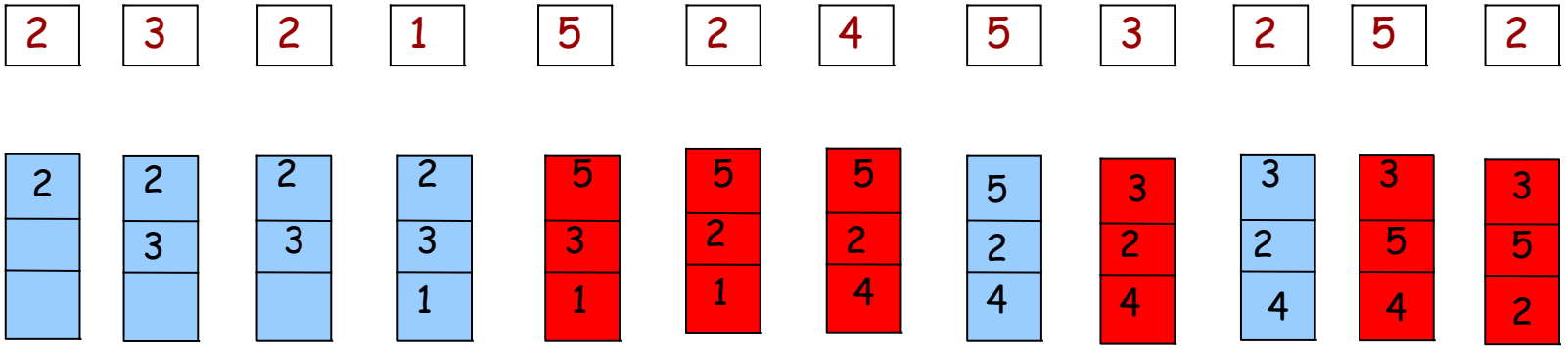
FIFO

- frames de um processo considerados como buffer circular
- ordem de substituição – *round robin*
- ponteiro indica a próxima a ser substituída
  - fácil implementação
  - baixo desempenho

Política de substituição - FIFO

FIFO

- pág. referenciadas: 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2



Política de substituição

FIFO X LRU

- grau de dificuldade de implementar (sobrecarga) *versus* desempenho

Relógio

- noção de tempo e uso
- bit adicional de controle

Desempenho:

- LRU, Relógio, FIFO

Política de Limpeza/Atualização

Sob demanda

- página modificada é escrita em MS somente quando substituída
- vantagem:
  - rapidez
- desvantagem:
  - afeta a execução do processo que sofre falta de páginas (espera pela gravação em MS de outra página)

Política de Limpeza

Pré Limpeza

- grava páginas modificadas em MS antes dos frames serem requisitados na substituição (em *bandos)*
- número de páginas gravadas a ser especificado
- vantagens:
  - gravação em conjunto de frames
- desvantagens:
  - pode não ser escolhidos para substituição
  - pode ser modificada depois da pré-limpeza e antes da substituição (tem que manter consistente)

Política de Limpeza

Bufferização de Páginas

- limpar páginas substituíveis
- páginas substituídas são associadas a duas listas:
  - modificadas
  - não modificadas
- modificadas gravadas periodicamente e colocadas na lista de não modificadas
- frames das páginas não modificadas são candidatas a substituição

Política de busca

- Determina qual página deve ser trazida para MP
- O objetivo é minimizar o número de faltas de página
- Políticas:
  - demanda
    - quando a página for necessária
    - falta de página
  - pré-paginação (*Working Set*)
    - conjunto de páginas é trazido
  - vantagens e desvantagens?

Tratando a falta de páginas

- Interrupção do h/w para o núcleo;
- salva PC e registradores de *status*
- Salvamento dos registradores de uso geral;
- chama SO
- SO descobre qual página virtual é necessária
- SO verifica validade do endereço e proteção e consegue um quadro

Tratando a falta de páginas

- Se quadro foi alterado, salva-o em disco
- SO busca página do disco
- Quando página chega:
  - tabela de páginas é atualizada
  - quadro é marcado
- Processo que causou falta é re-escalonado
- Registradores são restaurados e execução continua

Controle de carga

- Determina o número de processos residentes em MP
- Poucos processos → possibilidade de processador vazio
- muitos processos → possibilidade de *thrashing*
  - cada processo tem poucas páginas residentes
- Como decidir? Algumas medidas ....
  - tempo médio entre as faltas = tempo médio de processamento de uma falta
  - manter a utilização do processador em paginação em 50%

Suspensão de processos (swapped out)

- Usada para reduzir o nível de multiprogramação
  - o de menor prioridade: política de escalonamento de médio prazo
  - processo causador de falta de páginas
    - conjunto residente necessário ausente de qualquer maneira
    - coloca o processo em MS de qualquer modo
  - último processo ativado
  - maior processo: libera um maior número de quadros
  - menor processo: menos *overhead* para tornar suspenso

Segmentação: suporte de *h/w*

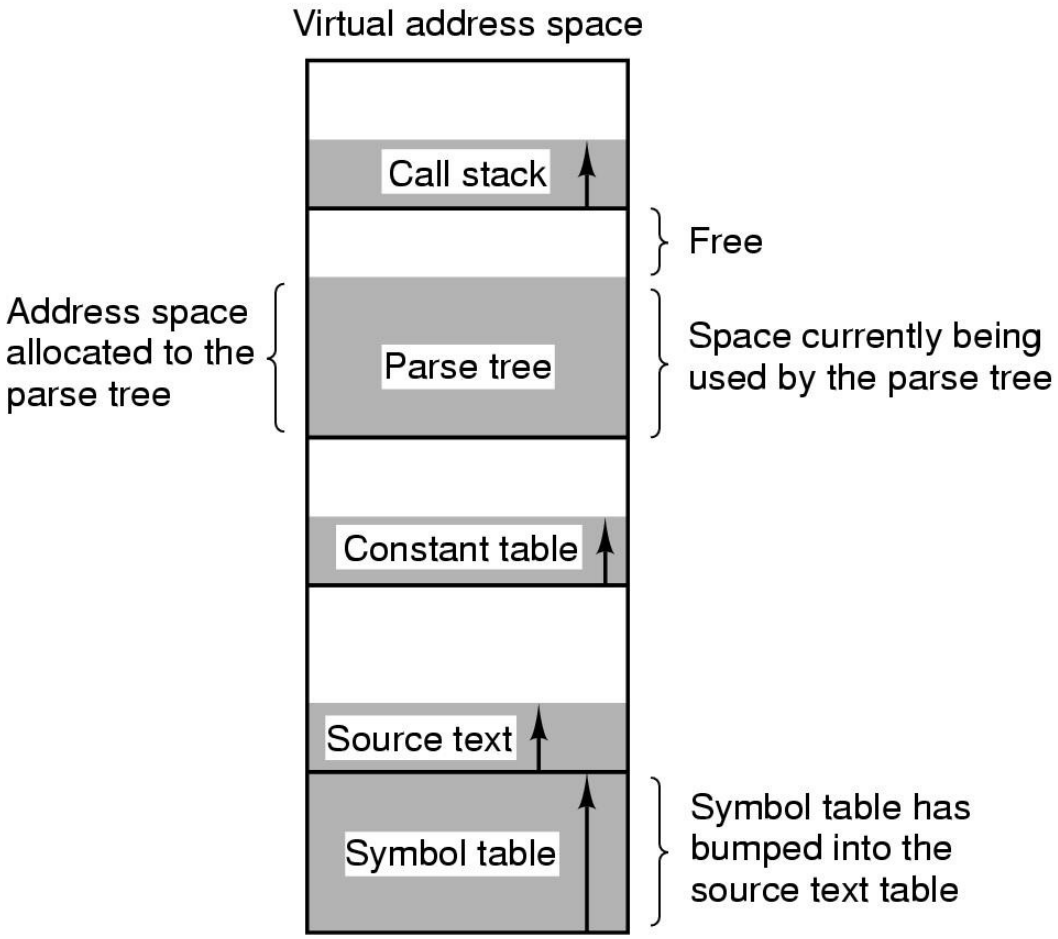
- Segmentação
  - pode ser dinâmica
  - compartilhamento e proteção

- Segmentação pura
- Segmentação com paginação

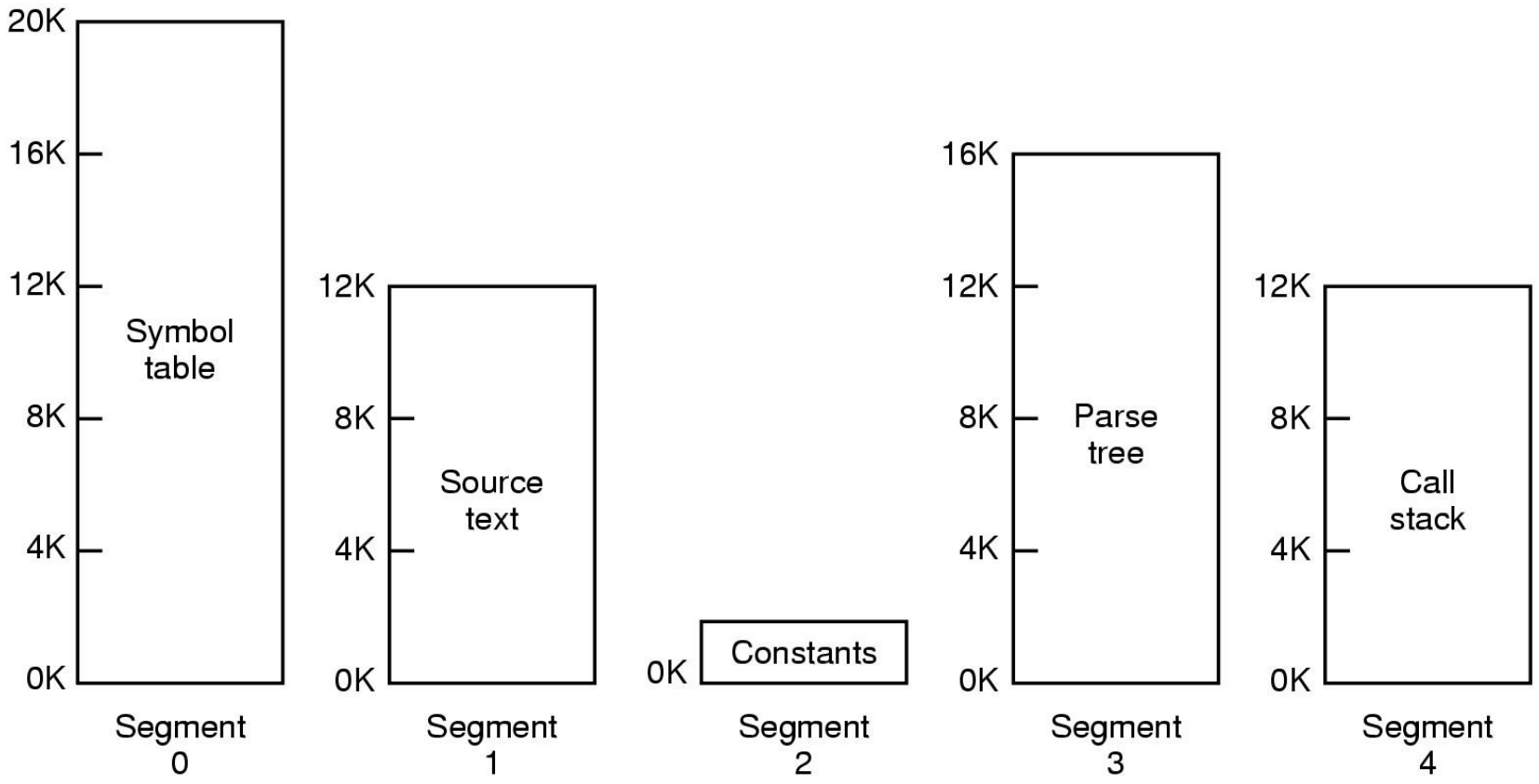
Vantagens de Segmentação

- programas com alocação dinâmica de estrutura de dados
  - um segmento pode ser alocado para as estr. de dados
  - SO manipula o aumento e diminuição do segmento

Espaço de uma dimensão



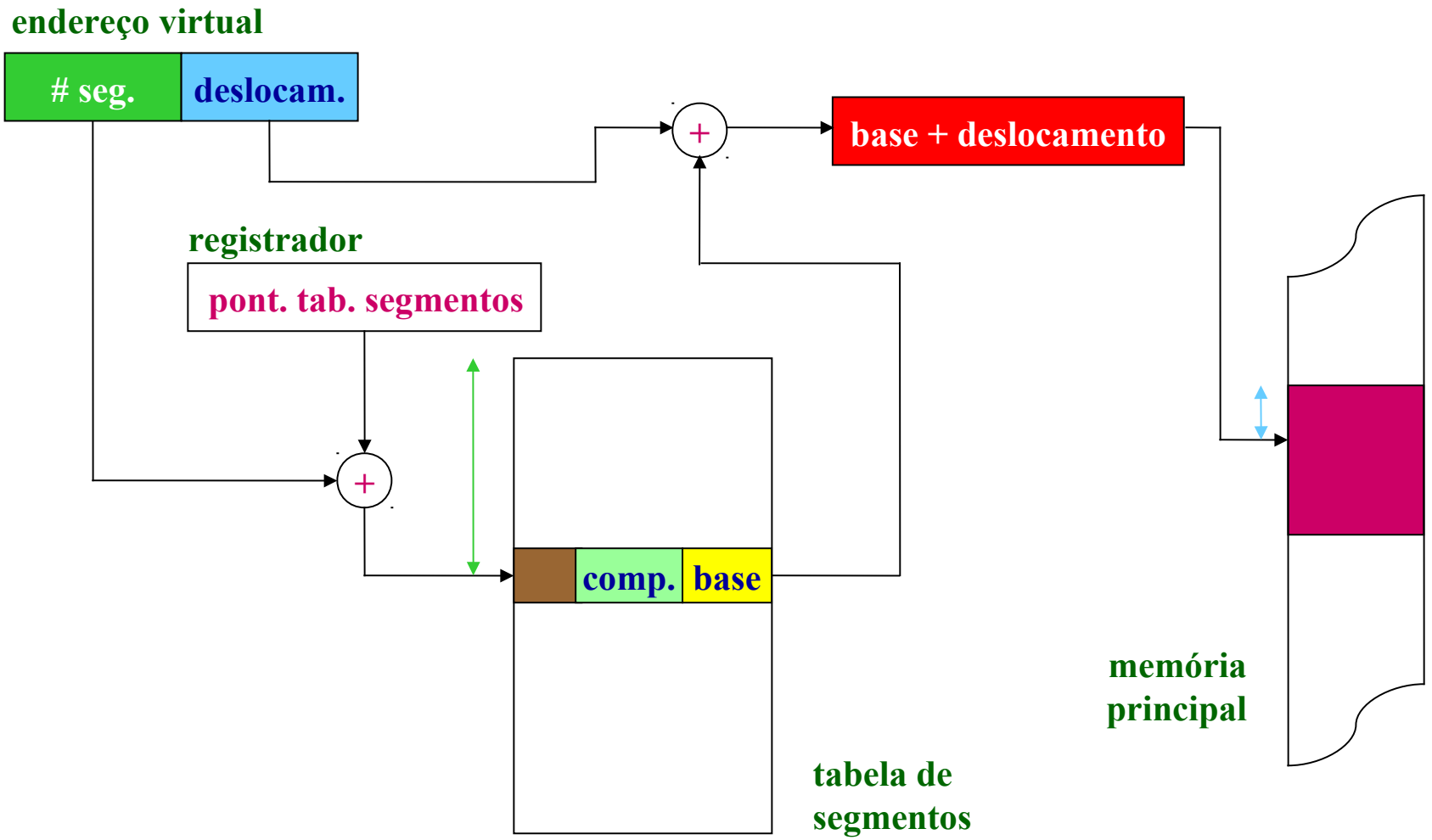
Memória segmentada



Organização

- uma tabela de segmentos por processo
- Campos de entrada
  - bit P: presença ou não em MP
  - bit M: modificado ou não
  - endereço inicial do segmento
  - tamanho do segmento

Segmentação: endereçamento



Segmentação: endereçamento

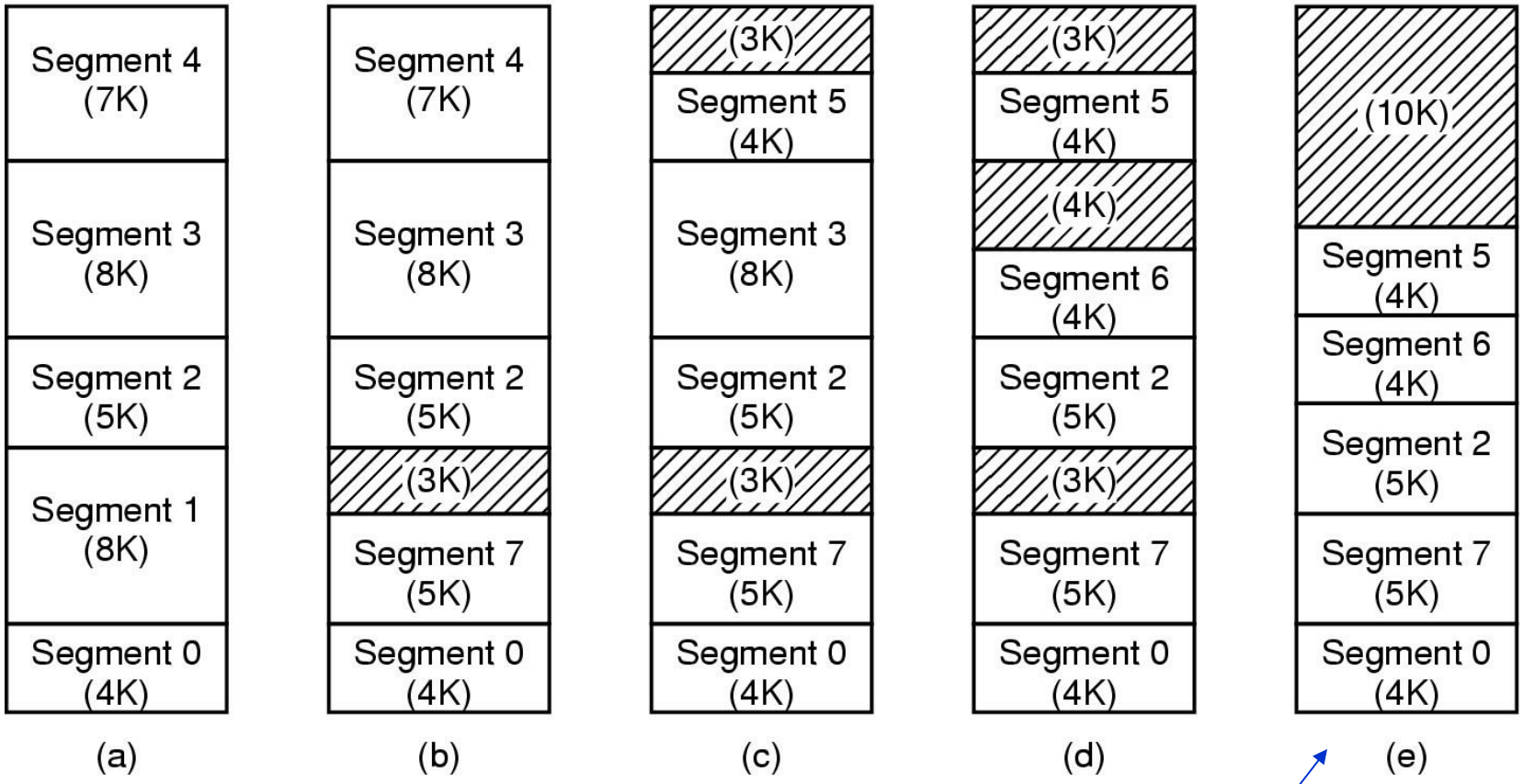
endereço virtual

linha da tabela de segmentos

Paginação x segmentação

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

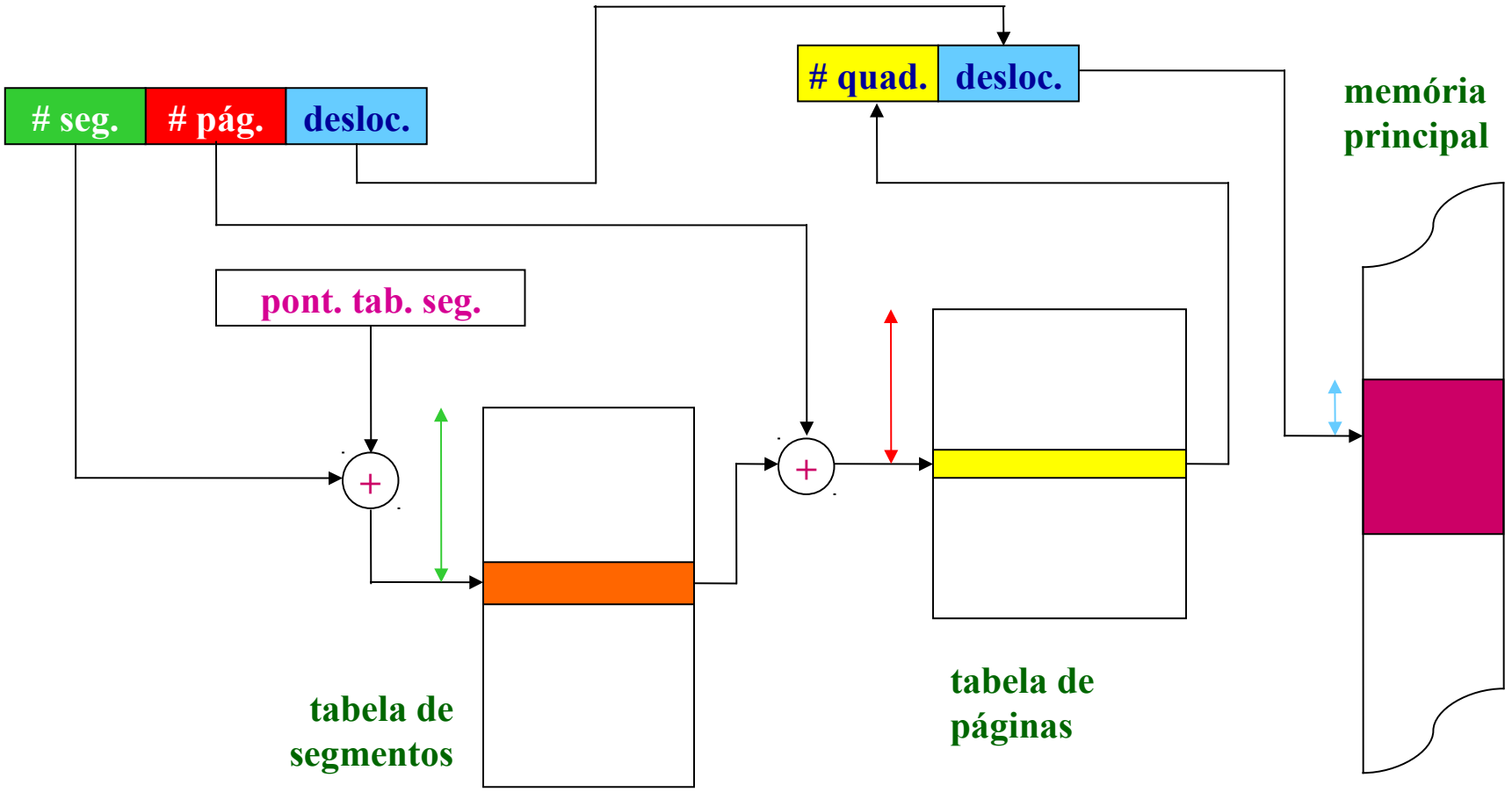
Segmentação pura



Paginação + segmentação

- para combinar as vantagens dos dois esquemas
- paginação:
  - evita a fragmentação externa
- segmentação:
  - mais fácil manipular alocação dinâmica de dados, por exemplo

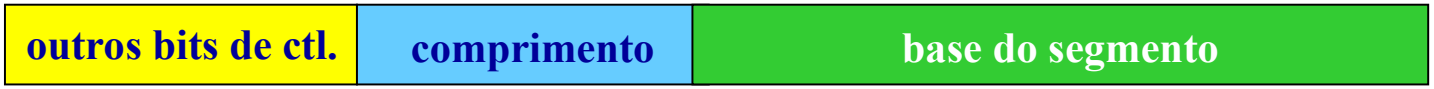
Segmentação com paginação



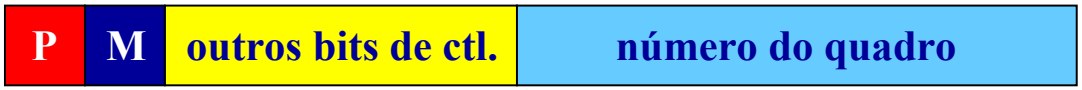
Segmentação com paginação

endereço virtual

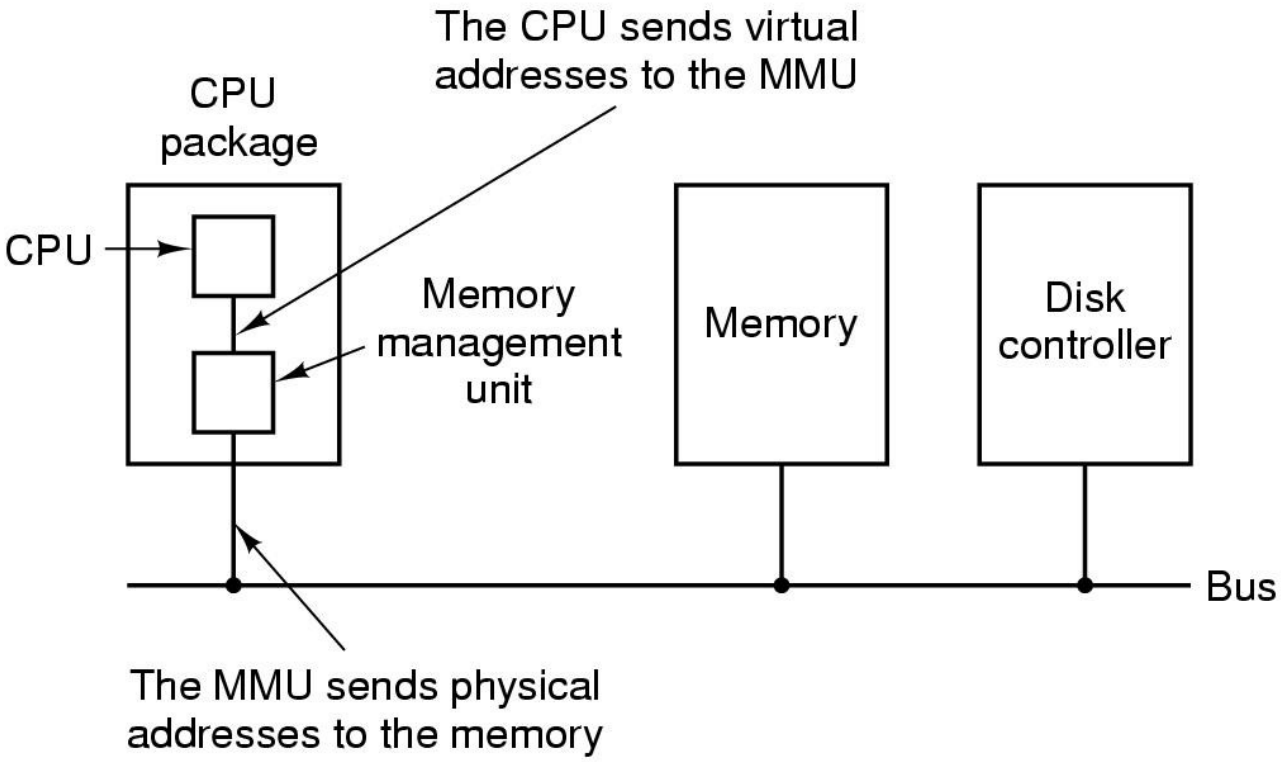
linha da tabela de segmentos



linha da tabela de páginas



A MMU


The CPU sends virtual addresses to the MMU
CPU package
CPU
Memory management unit
Memory
Disk controller
Bus
The MMU sends physical addresses to the memory

Mapeamento de endereço virtual para físico