# *Fundamentos de Sistemas de Operação*

*Unix Windows NT Netware MacOS DOS/VS Vax/VMS Linux Solaris HP/UX AIX Mach Chorus*

## *Gestão de Memória:*

## MV por Paginação a pedido

(revisões/continuação)

# *Memória Virtual* (revisão)



- Memória virtual (VM) – separação dos endereços lógicos (virtuais) dos físicos
  - Só alguns dos programas precisam de estar carregados em memória.
  - EE virtuais podem ser muito maiores do que a RAM.
  - Pode permitir partilha de partes do espaço de endereçamento entre processos (tal como vimos na paginação).
  - Permite uma criação de processos mais eficiente (COW).
  - É completamente transparente ao utilizador/programador
- VM pode ser suportada através de:
  - Paginação-a-pedido
  - Segmentação a pedido (não estudada)

# *Memória Virtual: motivações (rev.)*



- Usar a RAM como Cache para o Disco
  - O EE "completo" "existe" no disco. É trazido para a RAM quando for necessário
  - O espaço de endereçamento do processo pode exceder a dimensão da memória; a soma dos espaços de endereçamento dos vários processos pode exceder o tamanho da memória física.
- Simplificar a gestão de memória
  - Múltiplos processos residem em memória central cada um com o seu próprio espaço de endereçamento. Só zonas (código, dados, stack, ….) "activas" é que estão em RAM – permite "*overbooking*".
  - Fornece protecção - os processos não interferem uns com os outros, porque estão em espaços diferentes e diferentes secções do espaço de um processo têm permissões diferentes.

# *Memória Virtual por paginação-a-pedido*

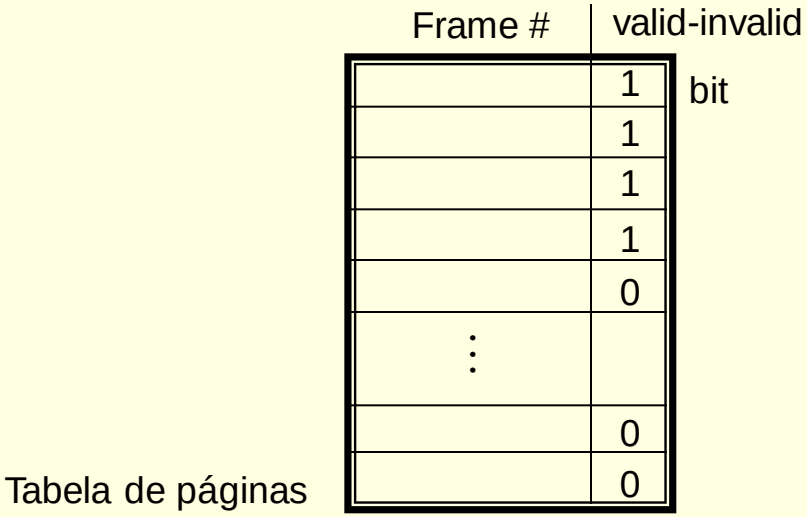


- A página só é carregada para memória **quando é referenciada** (naturalmente há um carregamento inicial)
  - Menos I/O
  - Menos RAM utilizada
  - Tempo de resposta pior (sistema “mais lento”?!)
  - Mais programas em RAM
- Página referenciada por uma instrução
  - Referência válida ⇒ executar a instrução
  - ATU indica “Não-em-memória” (Intel só usa um bit)
    - Página válida ⇒ Necessário trazer página para RAM
    - Página inválida ⇒ Abortar a execução

# *Memória Virtual: bit de validade (rev.)*



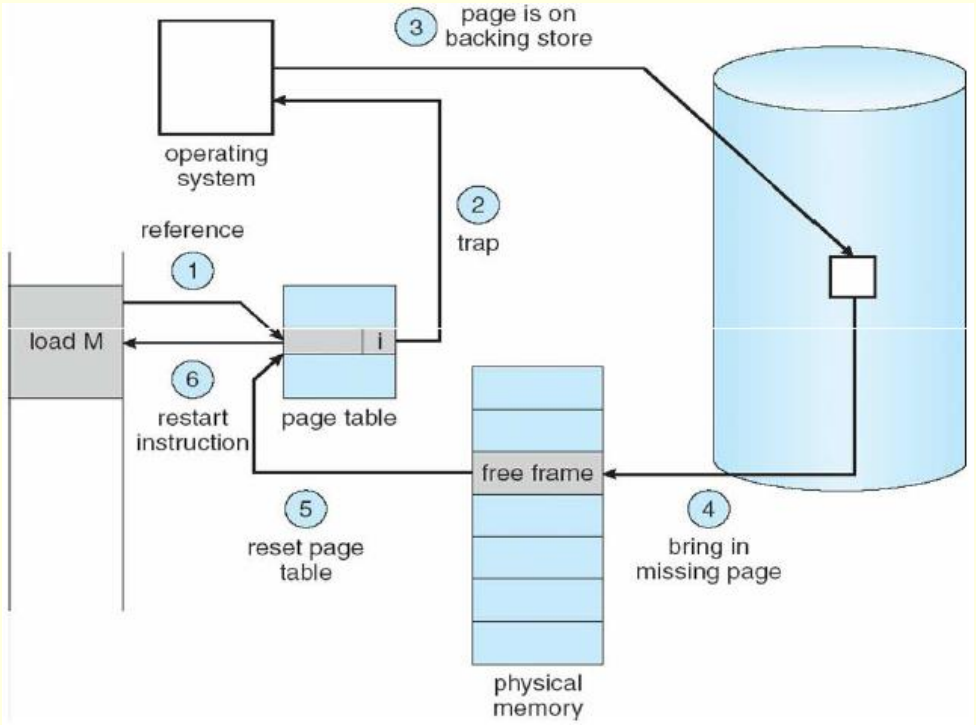
- Cada página tem associado um bit de Validade (V) (V==1 ⇒ em RAM, V==0 ⇒ em disco ou inválida-Intel)
- Inicialmente V==0 em todas as páginas.
- Na transformação de endereços, se V na entrada da tabela de páginas está a 0 ⇒ page fault (falta de página).



# *Memória Virtual: falta de página*



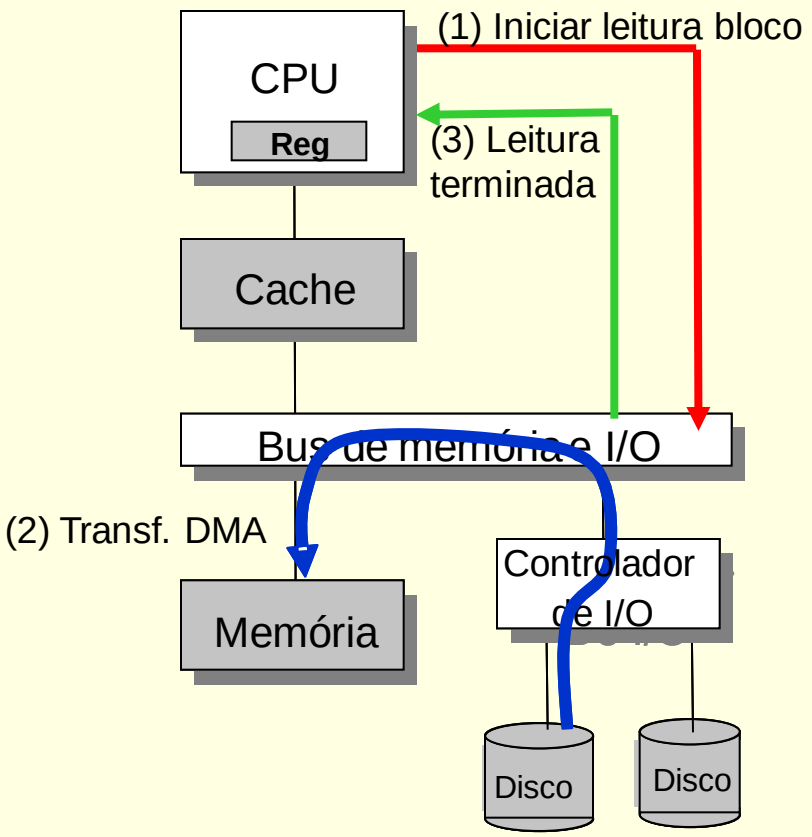
- Tabela de páginas indica que o conteúdo referenciado pelo endereço virtual não está em RAM
- É invocado um procedimento de excepção para trazer os dados para memória:
  - o processo corrente é suspenso, outros podem continuar
  - o SO tem controlo completo sobre a localização das páginas



# Memória Virtual: atendimento da falta de página



- CPU (executa código do driver do disco) pede ao controlador:
  - ler bloco de dim. P a partir do end. X e escrever na RAM começando no end. Y (driver adormece)
  - Ocorre a leitura
    - Dados no buffer do controlador
    - Direct Memory Access (DMA)
    - sob controlo do I/O controller
  - I / O Controller assinala o fim
    - Interrupção, driver acorda
    - SO passa o processo a READY



# *Memória Virtual: bit de "página modificada"*



- Cada página tem associado um bit de Modificada (M), ou *dirty* (suja) (M==1 ⇒ foi modificada desde que veio do disco, M==0 ⇒ versão da página em RAM igual à cópia em disco)
- Inicialmente M==0 em todas as páginas que são carregadas; durante a execução, algumas poderão ser modificadas (variáveis que mudam de valor, stack/heap,...)
- Se fôr preciso libertar uma frame em que M==1, a página tem de ser copiada para disco (para a área de paginação do SO)
  - Unix: disco tem zona de **`swap`**
  - Windows: disco tem ficheiro **`pagefile.sys`**

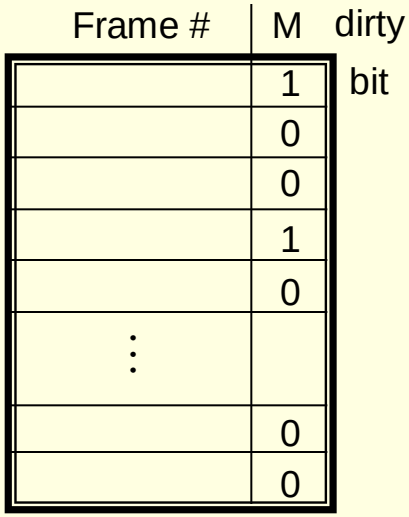



Tabela de páginas

# *Memória Virtual: tabela "completa"*



- Se o SO tiver de "sacrificar" uma página (diz-se vítima) presente em RAM para libertar a frame (porque é necessária para esse mesmo ou para um outro processo), não é muito interessante sacrificar uma que foi modificada, porque esta terá de ser escrita em disco (operação "lenta")
- Bits de protecção (bits, não letras!!!)
  - Associados a cada página, realizam o controle de acessos e são diferentes para as páginas de:
    - Constantes: r--
    - Variáveis: rw-
    - Heap: rw-
    - Stack: rwx
    - Código: r-x

Tabela de páginas

| Frame # | P | V | M |
|---|---|---|---|
| | r-- | 1 | 0 |
| | rw- | 1 | 0 |
| | rwx | 0 | 0 |
| | rwx | 1 | 1 |
| | --- | 0 | 0 |
| ⋮ | | | |
| | r-x | 0 | 0 |
| | r-x | 1 | 0 |

# *Tabela de Páginas do x86* (fictícia)

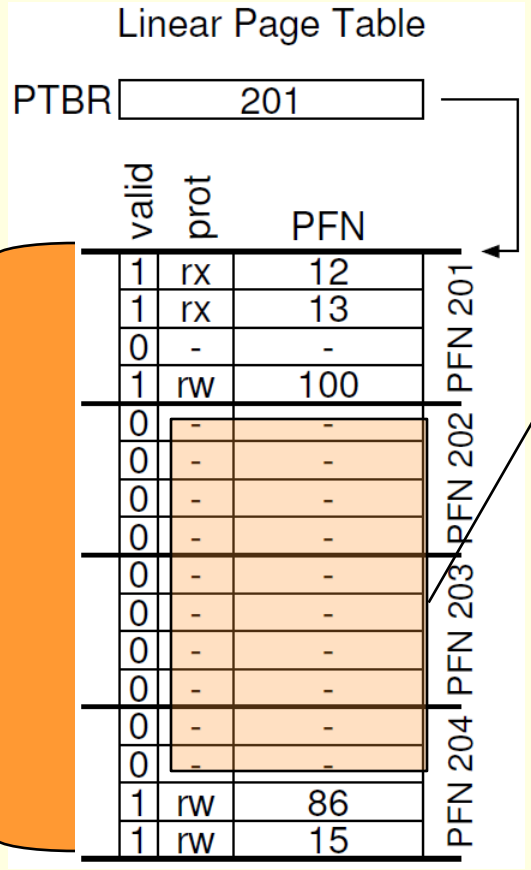


O PTBR (Page Table Base Register) é um reg. do CPU que aponta para o início da Tabela de Páginas do processo. É guardado no PCB.

No Intel CR3=PTBR

**1 M entradas**

A PT é guardada em RAM, logo é guardada em frames
Neste exemplo, a PT está guardada nas frames 201 a 204 (daí PTBR = 201)




Cada frame só tem espaço para 4 PTEs. O EE deste exemplo é 16 x 4KB (assumindo que 1 página=4KB)

**Páginas inválidas ("buracos" no EE)**

# *Fundamentos de Sistemas de Operação*

Unix Windows NT Netware MacOS DOS/VS Vax/VMS
Linux Solaris HIP/UX AIX Mach
Chorus

## *Revisões*
## *de preparação para o 1º teste*