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

Gestão de Memória

MV por Paginação a pedido:

A tabela de páginas

Paginação no x86 (1)

- Alguns números que dão que pensar
  - Dimensão do EE: 2^{32} = 4G (bytes)
  - Dimensão de uma página: 2^{12} = 4 KB
  - Logo, número de entradas na tabela de páginas (PTEs)
  - #PTEs = 2^{32} / 2^{12} = 2^{20} = 1M PTEs
- Uma PTE (Page Table Entry) no x86: (TPC identifique alguns dos bits)

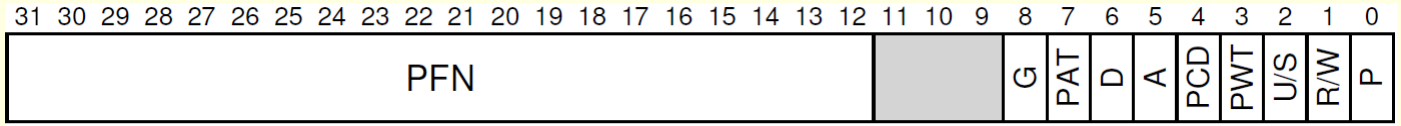


Figure 18.5: **An x86 Page Table Entry (PTE)**

Dimensão PTE = 32 bits = 4 Bytes
Dimensão da Tabela de Páginas: 4B x 1M PTEs = 4 MB

100 processos a correr → **400 MBs só de PTs**

Paginação no x86 (2)

- Uma PTE (Page Table Entry) no x86:

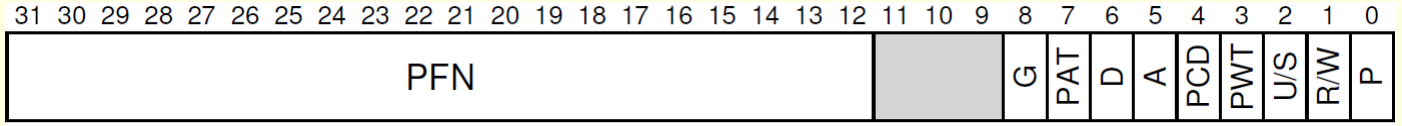


Figure 18.5: **An x86 Page Table Entry (PTE)**

- Bits (referem a página…):
  - P: Está Presente (/ausente) em RAM
  - R/W: Pode ser R(lida+executada) ou W(escrita+lida)
  - U/S: Só pode ser acedida em Supervisor (U= em qualquer)
  - D: Dirty, foi modificada

  - Note-se a combinação de permissões: R → rx e W → rw

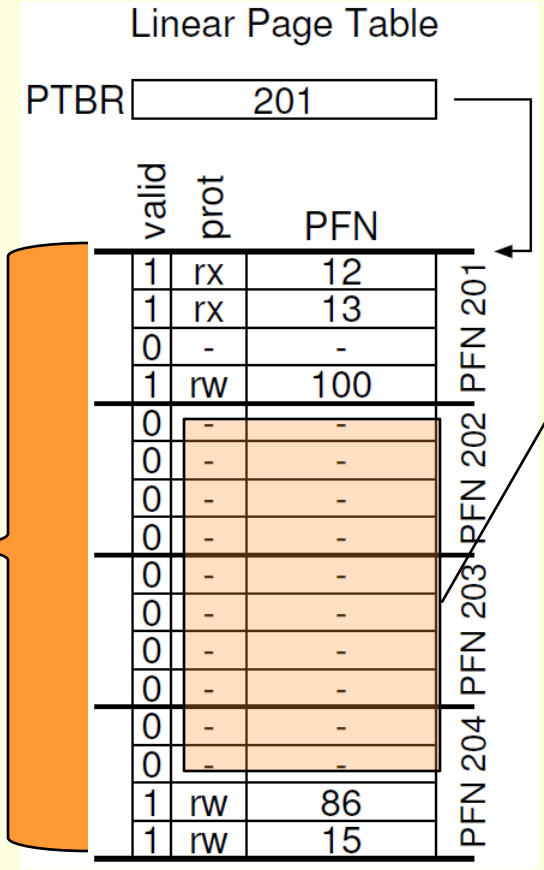
Tabela de Páginas do x86 (fictícia)

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)

Mas… muitas são inválidas (“buracos” no EE)

Para quê gastar espaço com elas???

Esta forma de Page Table é designada linear – é simplesmente um vector

V2P com Tabela de Páginas linear

A partir do endereço virtual, obtém-se o endereço físico

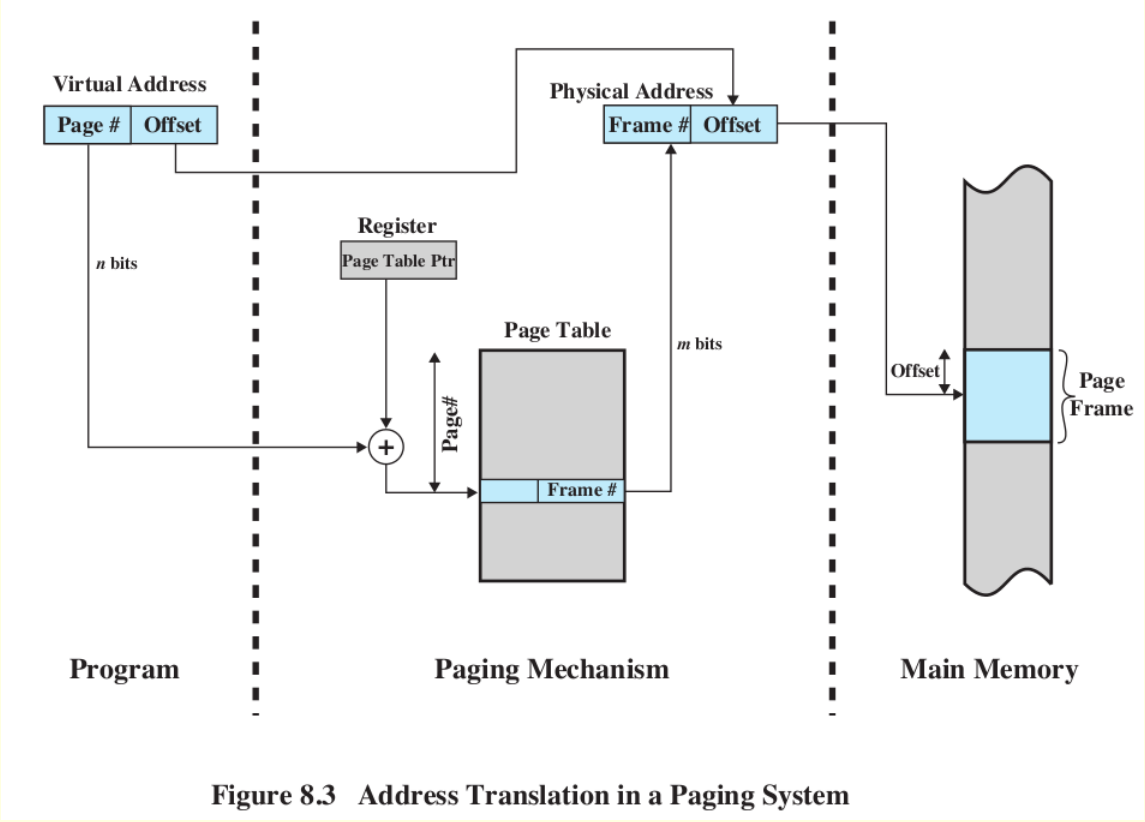
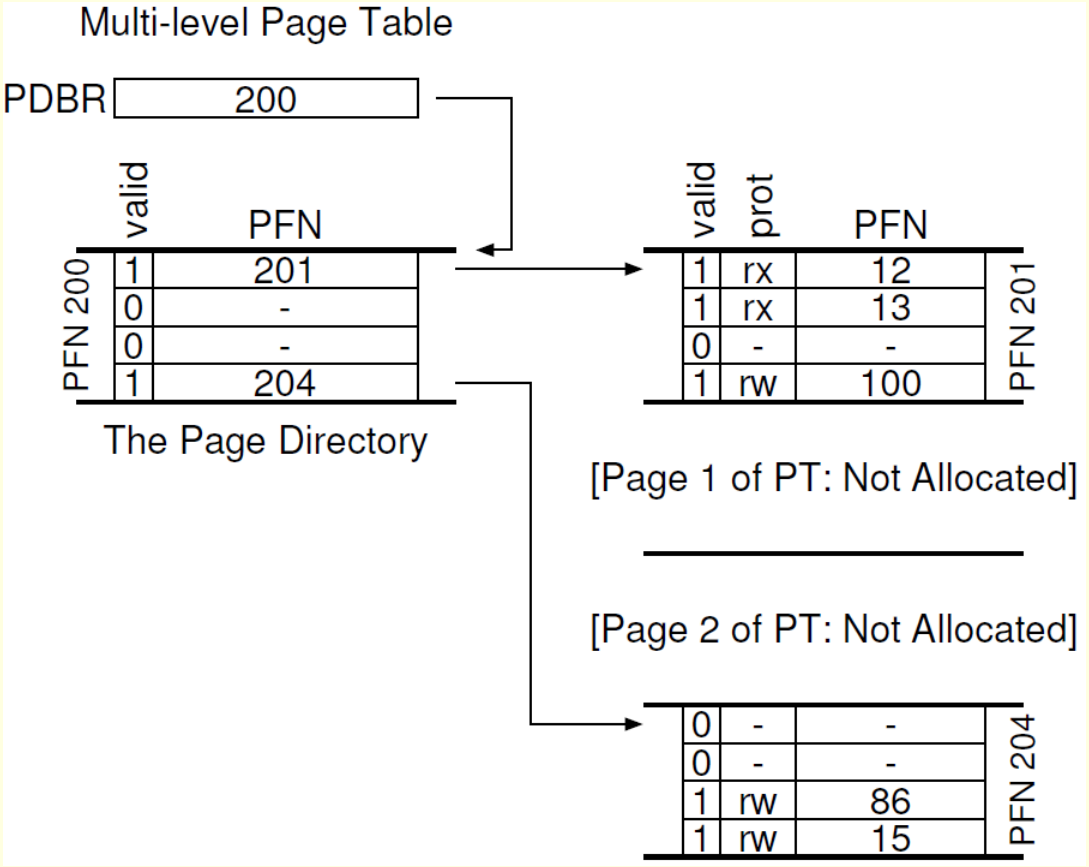


Figure 8.3 Address Translation in a Paging System

Fonte:
US Naval Academy

TPs: como reduzir o tamanho?



Neste exemplo, a PT está guardada nas frames 201 e 204.
Como as entradas 2 e 3 são inválidas, não se gastou espaço para elas

A frame 200 não tem dados da PT (não tem PTEs) mas sim apontadores para as frames. É uma directoria.

Dos 16x4KB de EE apenas 5 páginas (12,13,100,86 e 15) são válidas.

Neste exemplo, a PT tem 2 níveis…

V2P com PT com 2 níveis (e.g., x86)

A partir do endereço virtual, obtém-se o endereço físico, mas

Entrada da PTD = PTBR + 10 MSB da página

1º end. Indirecto = [Entrada da PTD] + 10 LSB da página

Frame = [1º endereço Indirecto]

Fonte:
US Naval Academy

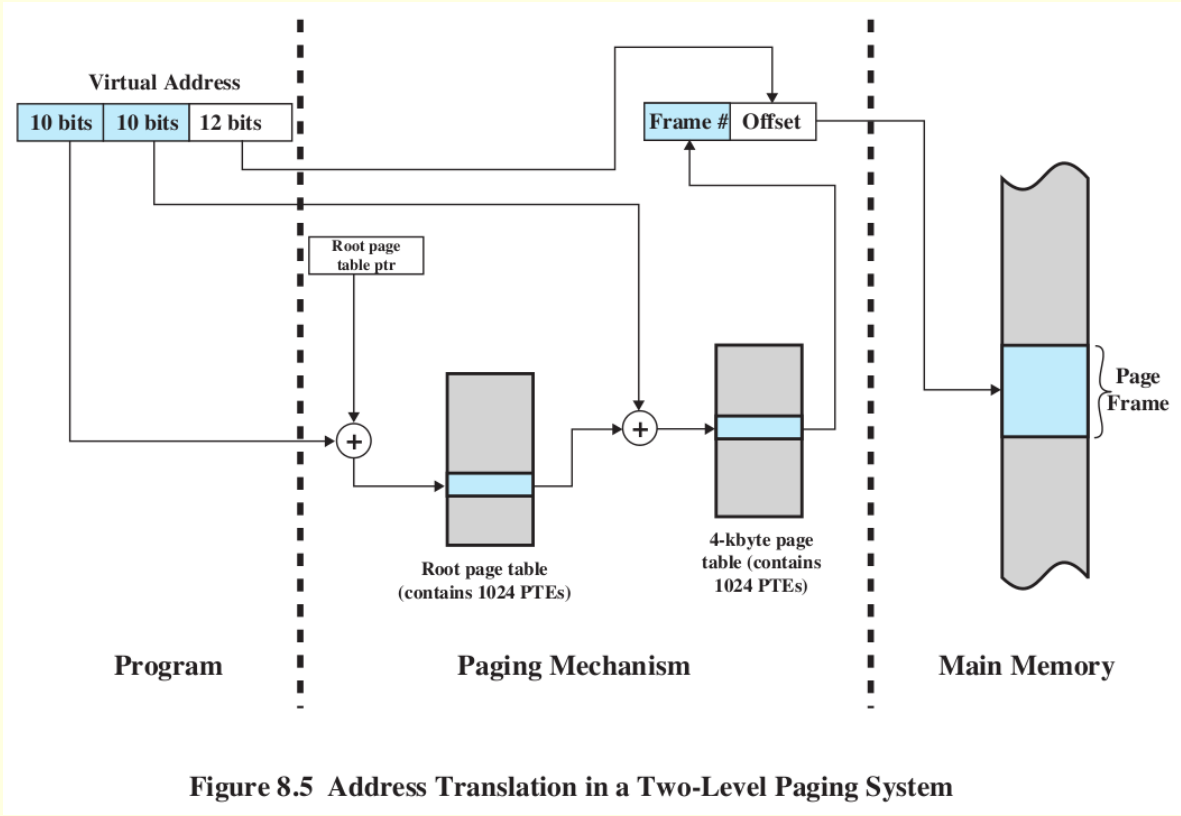


Figure 8.5 Address Translation in a Two-Level Paging System

Tradução de endereços

- *Se a tradução V2P fosse em software, seria…*

```
PDindex = vAddr >> (10+12)                          1ns
PDE = *(PTBR + Pdindex)                            50ns

PTindex = (vAddr >> 12) & MASKmsb10bits             2ns
PTE = *(PDE + Ptindex)                             50ns

if (access violates ProtectionlBits) go SIGSEGFAULT
PFN = PTE & MASKoutControlBits                      1ns

Offset = vAddr & MASKmsb20bits                      1ns
pAddr = PFN | Offset                                1ns
```

- *Lentoooooo!* *~100ns*

Acelerando a Tradução V2P: TLB

- *TLB é parte da MMU…*

These are typical performance levels of a TLB:

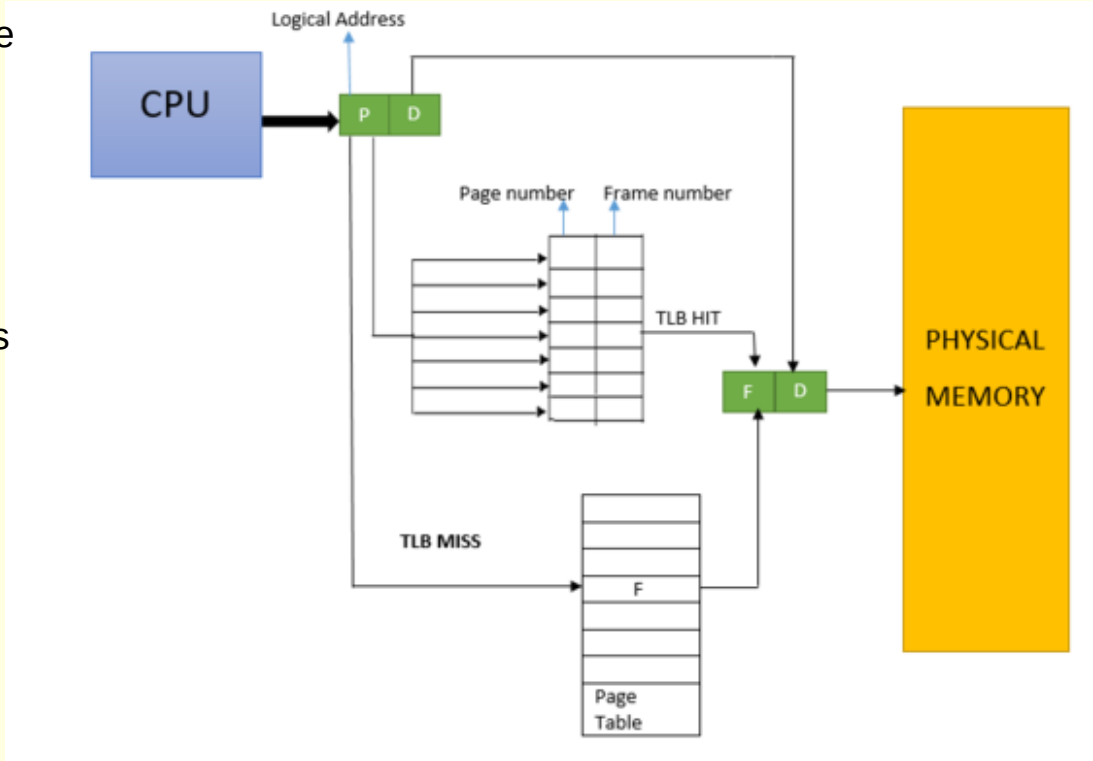
size: 4,096 entries

hit time: 0.5 – 1 clock cycle

miss penalty: 10 – 100 clock cycles

miss rate: 0.01 – 1% typical, 20%–40% for bad apps or sparse/graph applications)

Fonte: Wikipedia



Tradução V2P com TLB: penalizações

- *TLB funciona bem se…*
  - *É uma cache, logo o processo tem de exibir localidade de referência*
- *TLB funciona "mal" se*
  - *Há comutação de processos: ao novo processo não servem as traduções cached do outro (as PFNs são outras)*
- *Uma solução?*
  - *Reduzir o nº de páginas = aumentar o tamanho da página*
  - *Linux: HugePages: 1MB a 1GB!!!*