Memória virtual

Uso da memória (1)

Quando um programa é criado (compilado), não são conhecidos os outros programas que serão executados em simultâneo

Se lhe for atribuída uma zona fixa da memória (física), é possível que essa zona de memória (ou parte dela) seja também usada por outros programas

O programa deverá esperar que todos os programas que usam a sua zona de memória terminem? Quando é que isso acontecerá?

O programa correrá em simultâneo com os outros programas que usam a sua zona de memória? Como se garante, então, que não há interferência entre eles?

A solução é usar memória virtual

Uso da memória (2)

Os endereços usados por um programa referem-se a um espaço de endereçamento virtual (*virtual address space*)

A memória organiza-se em páginas (blocos, com entre 4KB e 16KB, normalmente)

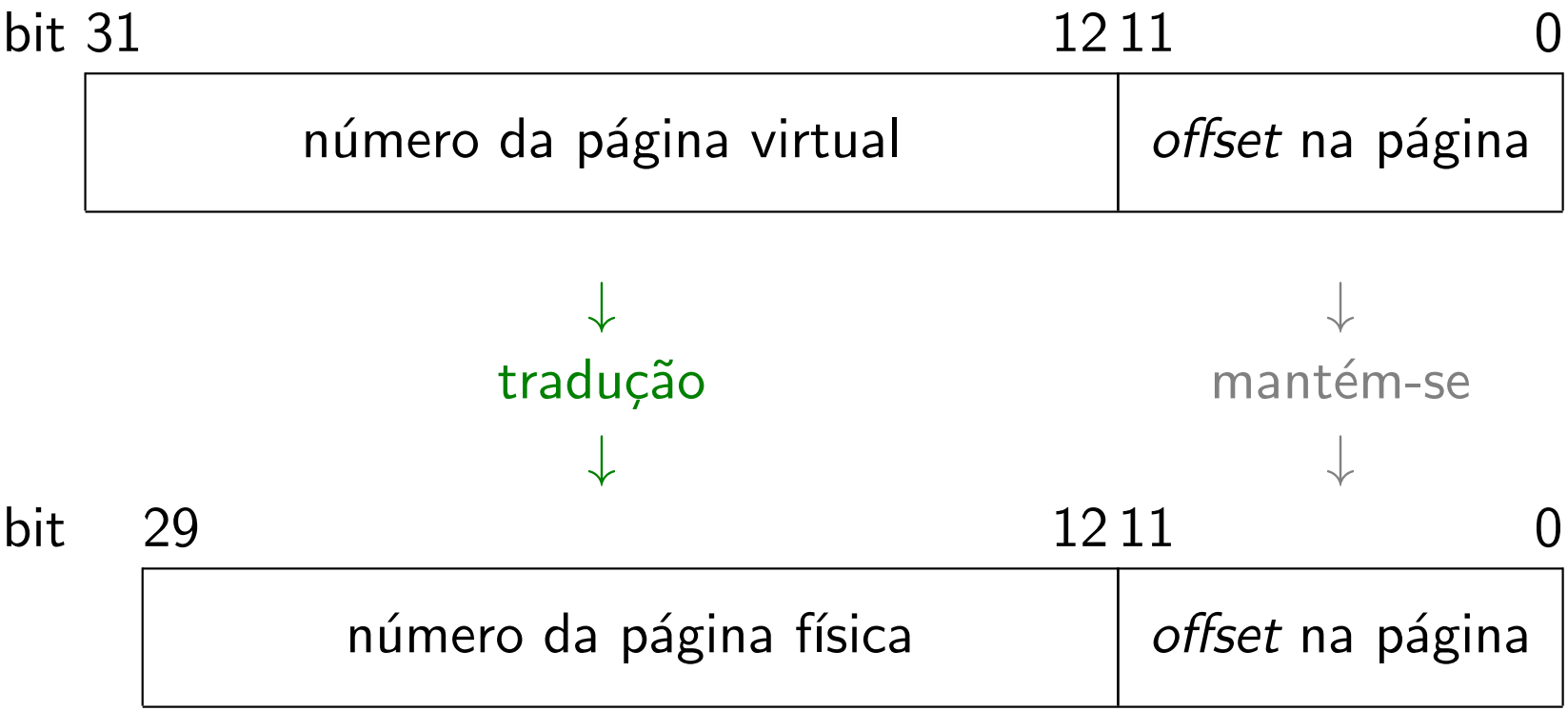
As páginas da memória virtual são mapeadas para páginas da memória física

Uma página de memória virtual pode residir em qualquer das páginas da memória física

Tradução de endereços (1)

Para efectuar um acesso à memória, é necessário traduzir o endereço virtual para o endereço físico correspondente

Endereço virtual de 32 bits



Endereço físico de 30 bits

Páginas com 2^{12} *bytes* = 4KB

Tradução de endereços (2)

A tabela de páginas mantém a correspondência entre as páginas virtuais e físicas de um programa

Cada programa em execução constitui um processo e tem a sua tabela de páginas, residente no espaço de memória do sistema operativo

O *page table register* do processador contém o endereço da tabela de páginas do programa que está a ser executado

Implementação da tabela de páginas (1)

Tradicional (Completa)

Uma entrada por cada página virtual, identificando a página física correspondente

Prós Acesso directo

Contras Para os espaços de endereçamento recentes, necessita de muita memória

Crescente

Uma entrada por cada página virtual

Cresce à medida que são acedidas mais páginas

Prós Acesso directo

Contras Grande parte dos programas acede a páginas no início e no fim do espaço de endereçamento

Implementação da tabela de páginas (2)

Crescente em dois sentidos

Uma entrada por cada página virtual

Dividida em duas partes, uma para as páginas de endereços baixos, outra para as páginas de endereços altos

Cresce em dois sentidos à medida que são acedidas mais páginas

Usada no MIPS

Prós Acesso simples

Contras Não se adapta a programas que usam o espaço de endereçamento de forma descontínua

Implementação da tabela de páginas (3)

Invertida

Uma entrada por cada página física

Acesso através de uma função (de *hash*) aplicada ao número da página virtual

Prós Adaptada ao espaço de endereçamento físico

Contras Acesso mais complexo

Multi-nível

Uma parte do número da página virtual é usada para indexar a tabela do 1º nível

Tabelas do 2º nível identificam as páginas físicas correspondentes

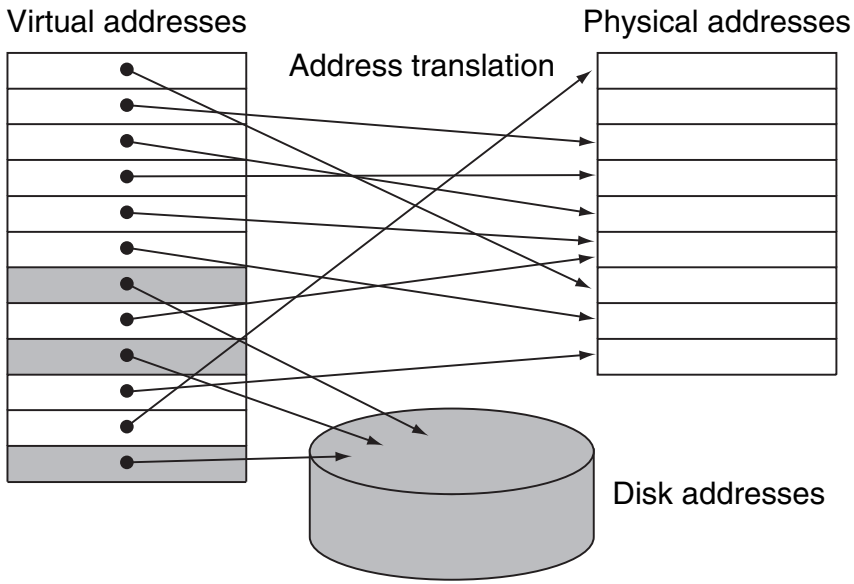
Só existem as correspondentes a páginas usadas

Prós Adapta-se à parte utilizada do espaço de endereçamento

Contras Acesso em dois ou mais passos

Page fault

As páginas virtuais que não cabem na memória física são guardadas em memória secundária



Swap space

Se a página virtual não está em memória física, ocorre uma *page fault*

Neste caso, a página virtual tem de ser carregada para uma página física

Memória física e memória virtual

A memória física contém parte do espaço de memória virtual dos programas em execução

A memória física comporta-se como uma cache da memória virtual

- Com organização *fully associative*
- Usando LRU para a escolha da página a substituir

  A cada página virtual está associado um *reference bit* para controlar os acessos à página

- Usando a estratégia *write-back* para lidar com as operações de escrita

  A cada página virtual está associado um *dirty bit*, que indica se o conteúdo da página foi alterado e se é necessário copiá-lo para memória secundária quando a página for substituída na memória física

Translation-lookaside buffer

Como a tabela de páginas reside em memória, se for sempre necessário consultá-la, cada acesso à memória implica dois acessos à memória

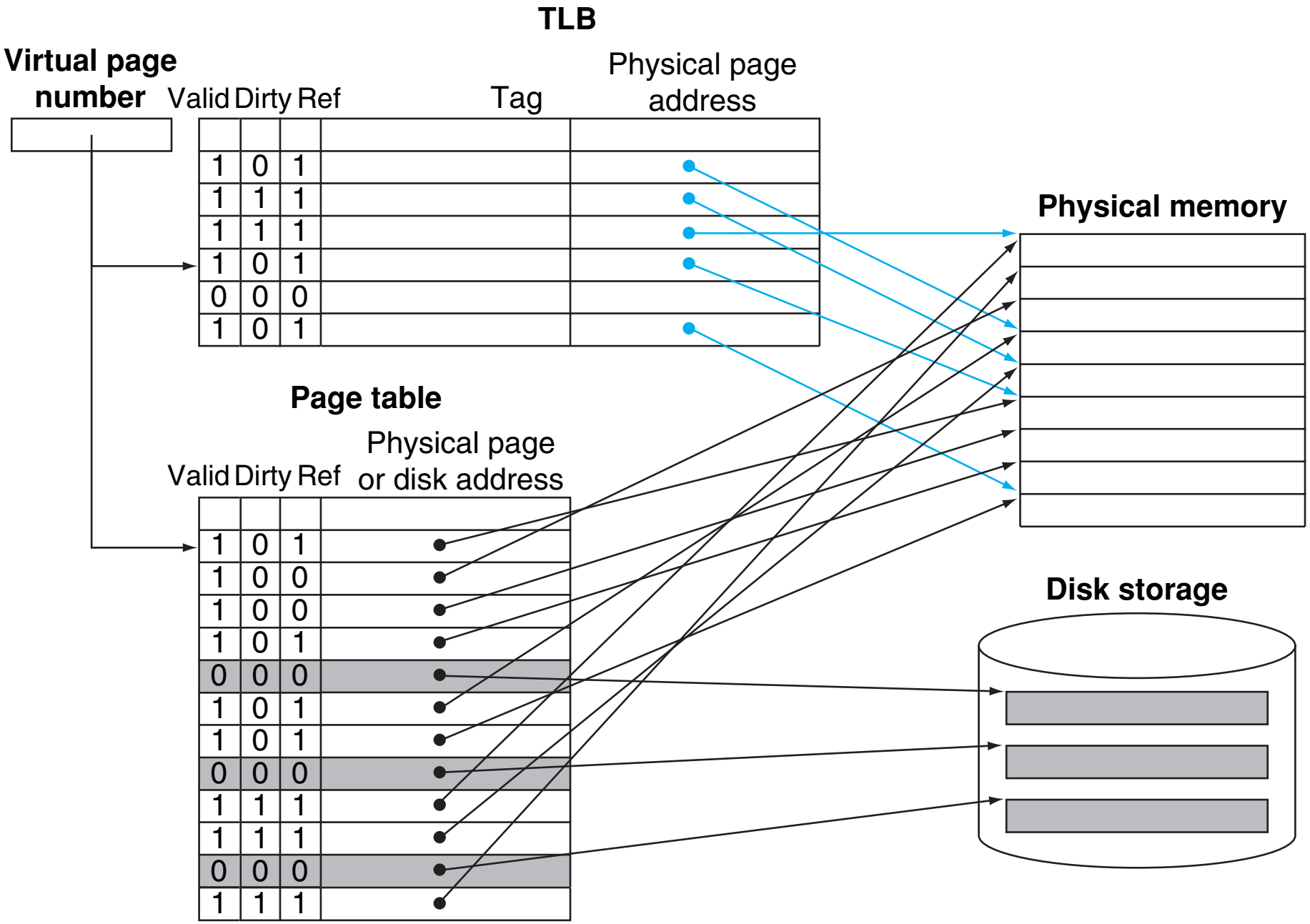
O *translation-lookaside buffer* (TLB) é uma cache da tabela de páginas, que contém traduções de páginas virtuais em físicas

A tabela de páginas só é consultada se a tradução

$$\text{número da página virtual} \rightarrow \text{número da página física}$$

não está presente no TLB

O TLB como cache da tabela de páginas



Acesso ao TLB

TLB *hit*

O número da página física contido no TLB é concatenado com o *offset* na página para obter o endereço físico

TLB *miss*

É gerada a excepção TLB *miss*

Tratada por *software* ou *hardware*

No MIPS é tratada por *software*, a partir do endereço $8000\ 0000_{16}$

```
mfc0 $k1, Context   # copia endereço da posição (na tabela
                    # de páginas) correspondente à página virtual
                    # acedida para $k1
lw   $k1, 0($k1)    # carrega conteúdo dessa posição para $k1
mtc0 $k1, EntryLo   # copia conteúdo dessa posição para
                    # o registo EntryLo
tlbwr               # escreve EntryLo e EntryHi (que contém o tag)
                    # na posição Random do TLB
eret                # termina o tratamento da excepção TLB miss
```

Registos do coprocessador 0 do MIPS

Alguns. . .

Register	CP0 register number	Description
EPC	14	Where to restart after exception
Cause	13	Cause of exception
BadVAddr	8	Address that caused exception
Index	0	Location in TLB to be read or written
Random	1	Pseudorandom location in TLB
EntryLo	2	Physical page address and flags
EntryHi	10	Virtual page address
Context	4	Page table address and page number

Acesso à tabela de páginas

Page fault

Se o conteúdo da tabela de páginas indica que a página virtual não está em memória, é gerada a excepção *page fault*

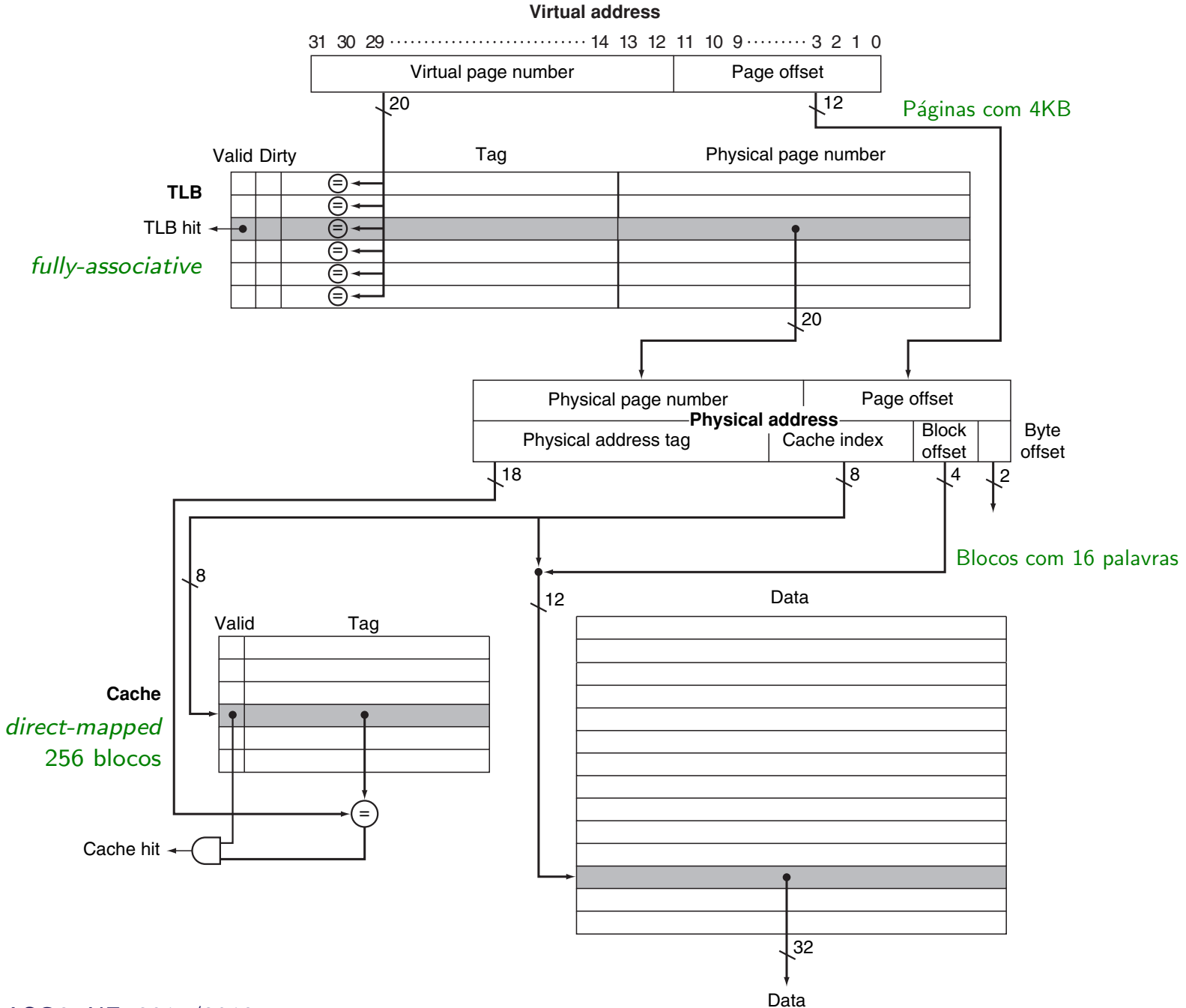
Esta excepção é sempre tratada por *software*:

- Escolha da página física onde colocar a página virtual

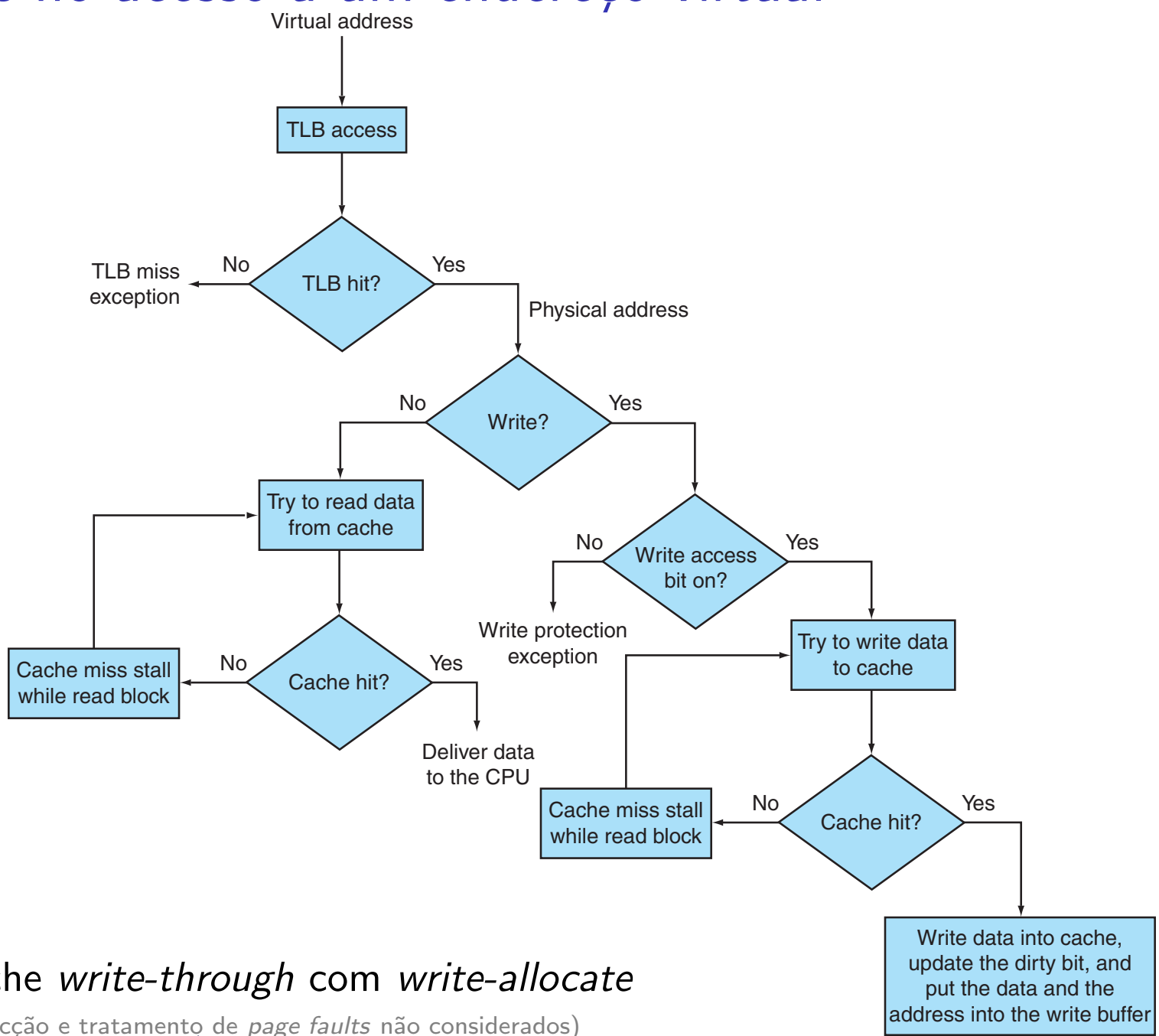
  (Usando LRU, em geral, se não há páginas livres)
- Se a página física está ocupada e o seu *dirty* bit indica que a página foi alterada, ela é escrita em memória secundária
- Carregamento da página pretendida

  (Escrita e carregamento demoram milhões de ciclos)
- Actualização das tabelas de páginas dos processos envolvidos
- Actualização do TLB

O TLB e a cache



Passos no acesso a um endereço virtual



Cache *write-through* com *write-allocate*

(Detecção e tratamento de *page faults* não considerados)

Alguns valores típicos

Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	2500–25,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	125–2000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%

L3 2–8 MB, reduz *miss penalty* da L2 para 30–40 ciclos

ARM Cortex-A8 vs. Intel Core i7

Suporte de memória virtual

Characteristic	ARM Cortex-A8	Intel Core i7
Virtual address	32 bits	48 bits
Physical address	32 bits	44 bits
Page size	Variable: 4, 16, 64 KiB, 1, 16 MiB	Variable: 4 KiB, 2/4 MiB
TLB organization	1 TLB for instructions and 1 TLB for data Both TLBs are fully associative, with 32 entries, round robin replacement TLB misses handled in hardware	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, 7 per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB is four-way set associative, LRU replacement The L2 TLB has 512 entries TLB misses handled in hardware

Intel Nehalem vs. AMD Opteron X4 (Barcelona)

Suporte de memória virtual

Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
Virtual address	48 bits	48 bits
Physical address	44 bits	48 bits
Page size	4 KB, 2/4 MB	4 KB, 2/4 MB
TLB organization	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement The L2 TLB is four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, 7 per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB has 512 entries TLB misses handled in hardware	1 L1 TLB for instructions and 1 L1 TLB for data per core Both L1 TLBs fully associative, LRU replacement 1 L2 TLB for instructions and 1 L2 TLB for data per core Both L2 TLBs are four-way set associative, round-robin Both L1 TLBs have 48 entries Both L2 TLBs have 512 entries TLB misses handled in hardware