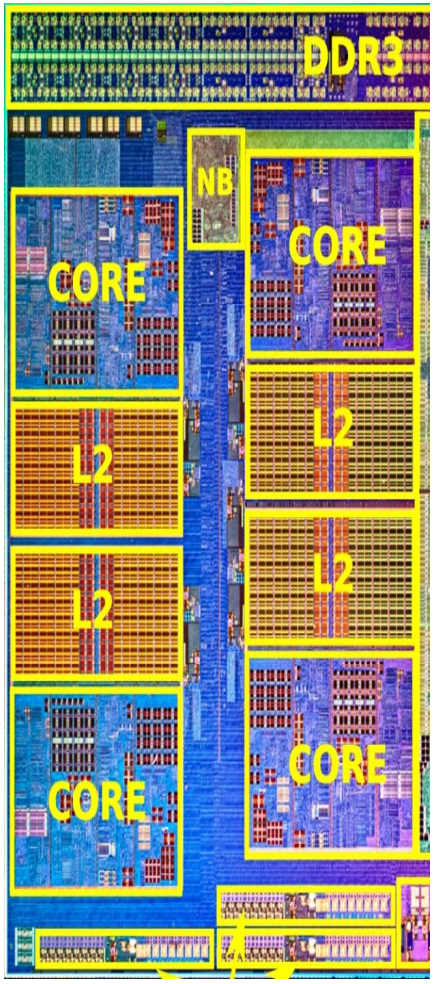
0273359 – Arquitetura e Organização de Computadores 1



Fonte: http://www.techspot.com/article/904-history-of-the-personal-computer-part-5/

Memória Virtual

Luciano de Oliveira Neris
luciano@dc.ufscar.br

Adaptado de slides do prof. Marcio Merino Fernandes
Figuras: David Patterson, John Hennessy
Arquitetura e Organização de Computadores – 4Ed, Elsevier, 2014

Departamento de Computação
Universidade Federal de São Carlos

Memória Virtual

- Memória Virtual: técnica na qual o disco é usado para "aumentar" o tamanho da memória principal do computador (RAM).
- Múltiplos processos são executados ao "mesmo tempo"
- Permitir o compartilhamento seguro e eficiente da memória entre múltiplos programas.
- Cada processo tem seu espaço de endereços (address space)
  - Conjunto de todos os endereços lógicos que um processo pode referenciar.

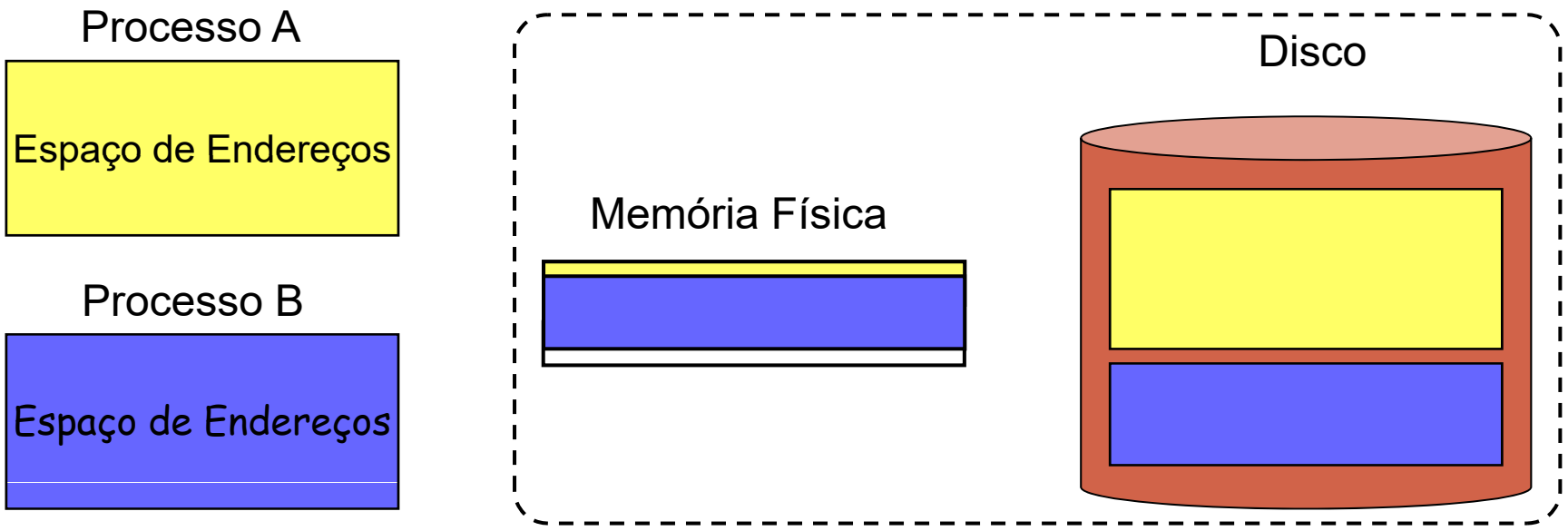
Memória Virtual

- Normalmente, a soma de todos os espaços de endereços é bem maior que a memória disponível
- Porém, apenas uma pequena parte desses espaços está ativa em um dado momento (novamente, o princípio da localidade)
- Dessa forma, é viável implementar um sistema de memória virtual.
  - Devido ao custo elevado associado a um acesso ao disco (milhões de ciclos), é importante organizar esta parte da hierarquia de tal forma a minimizar a taxa de ausência de dados na memória principal.

Compartilhamento

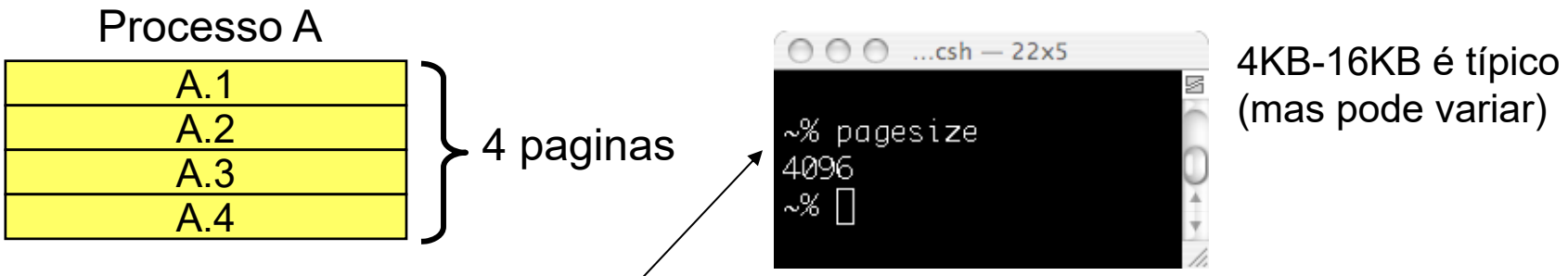
- Considere um computador com 1GB de RAM
- Considere dois processos A e B, cada um com espaço de endereços de 2GB.
- Ambos os processos podem ter parte do seu espaço de endereços na memória RAM
  - ex: 200 MB p/ PA e 600 MB p/ PB
- Quando um processo referenciar (acessar) um endereço que não está na RAM, este deve ser trazido do disco p/ a RAM, substituindo algum outra porção de endereços.

Compartilhamento



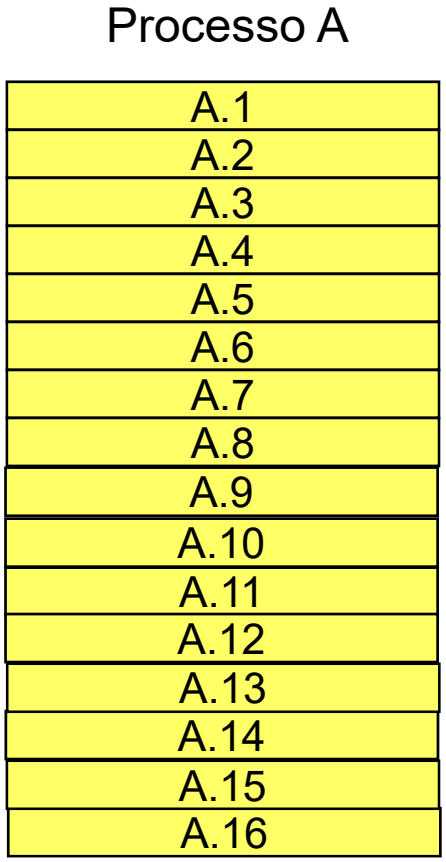
Páginas de Memória (Paginação)

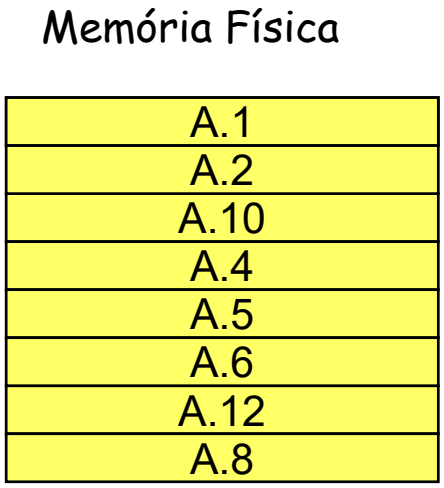
- Cada espaço de endereços é subdividido em páginas de memória (memory pages)

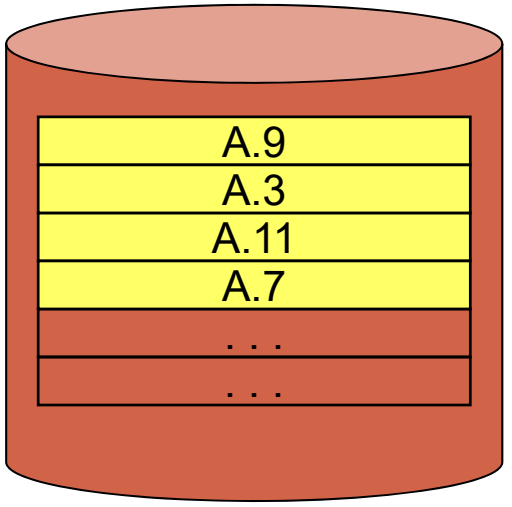


- O tamanho do espaço de endereços de um processo é um múltiplo do tamanha de uma página.
  - Linux: multiplo de 4KB
- O tamanho da página tem impacto no desempenho do sistema.

Exemplo – 1 Processo



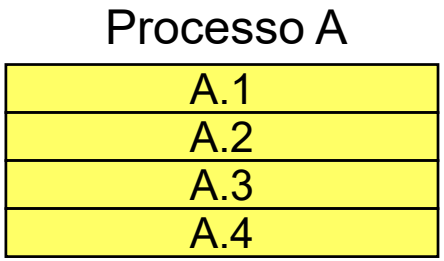


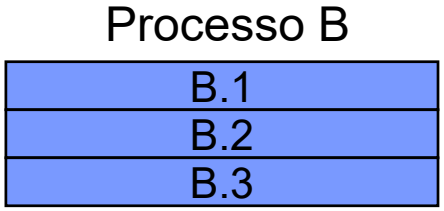


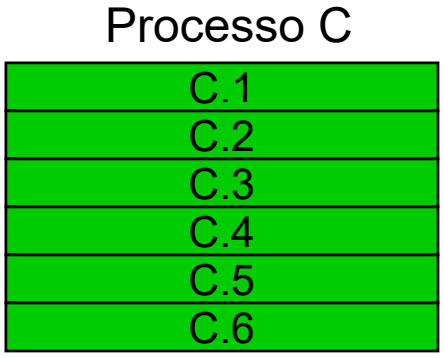
Disco

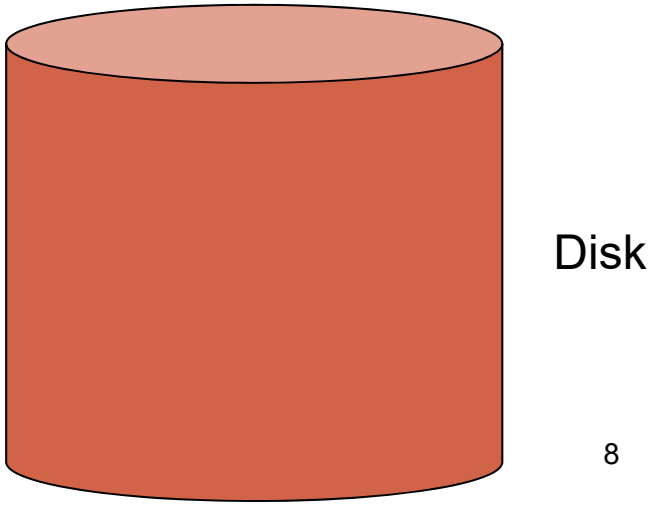
- Um dado processo pode ter um espaço de endereços maior que a memória física.
- A memória virtual passa a ilusão de uma memória principal maior
  - ....c/ perda de desempenho

Exemplo – Vários Processos

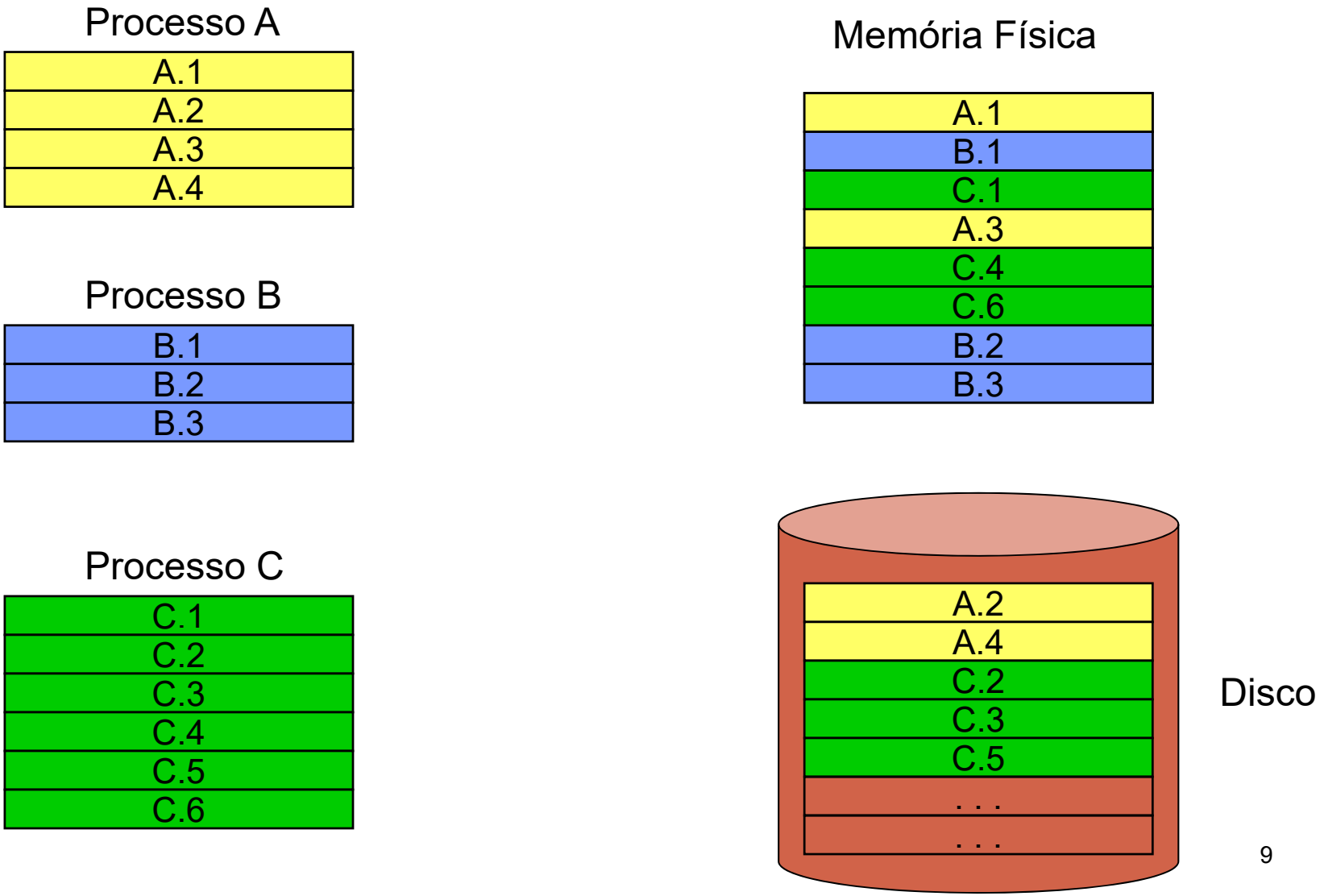




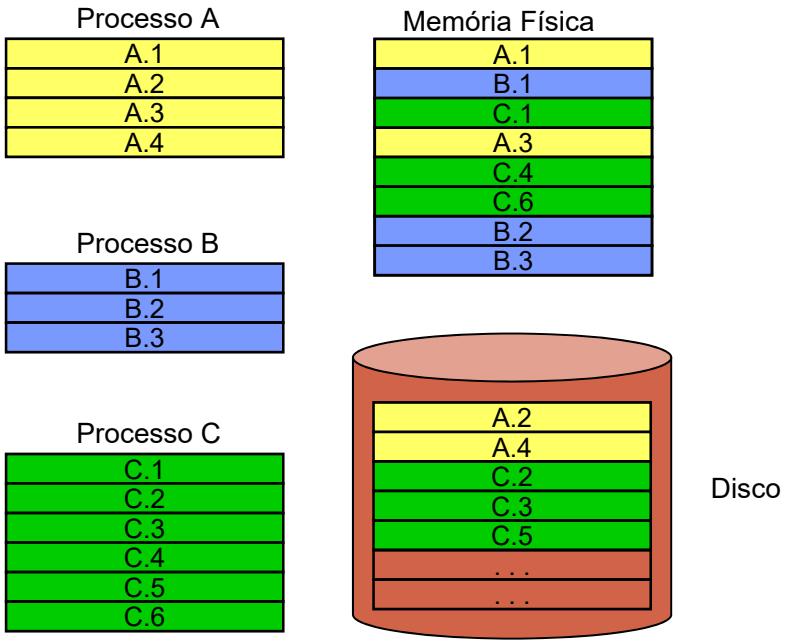




Exemplo – Vários Processos



Proteção de Memória



- O mecanismo de memória virtual deve garantir a proteção de memória:
  - O processo A não pode ler/escreer no espaço de endereços (memória) do processo B
  - Isso é feito por meio do esquema de tradução de endereços.

Tradução de Endereços

- **Pergunta**
  - Já que não sabemos antecipadamente quais processos estarão na memória RAM ao mesmo tempo, como fazemos para proteger seus respectivos espaços ?
    - ...no passado isso era feito manualmente pelos programadores!

Tradução de Endereços

- **Resposta**
  - Utilize um nível adicional de endereçamento, chamado de endereço virtual.
  - Cada processo tem um espaço de endereços virtuais únicos, o qual é independente do endereço físico real.
  - Antes da execução de um processo, endereços virtuais são traduzidos em endereços reais (físicos).

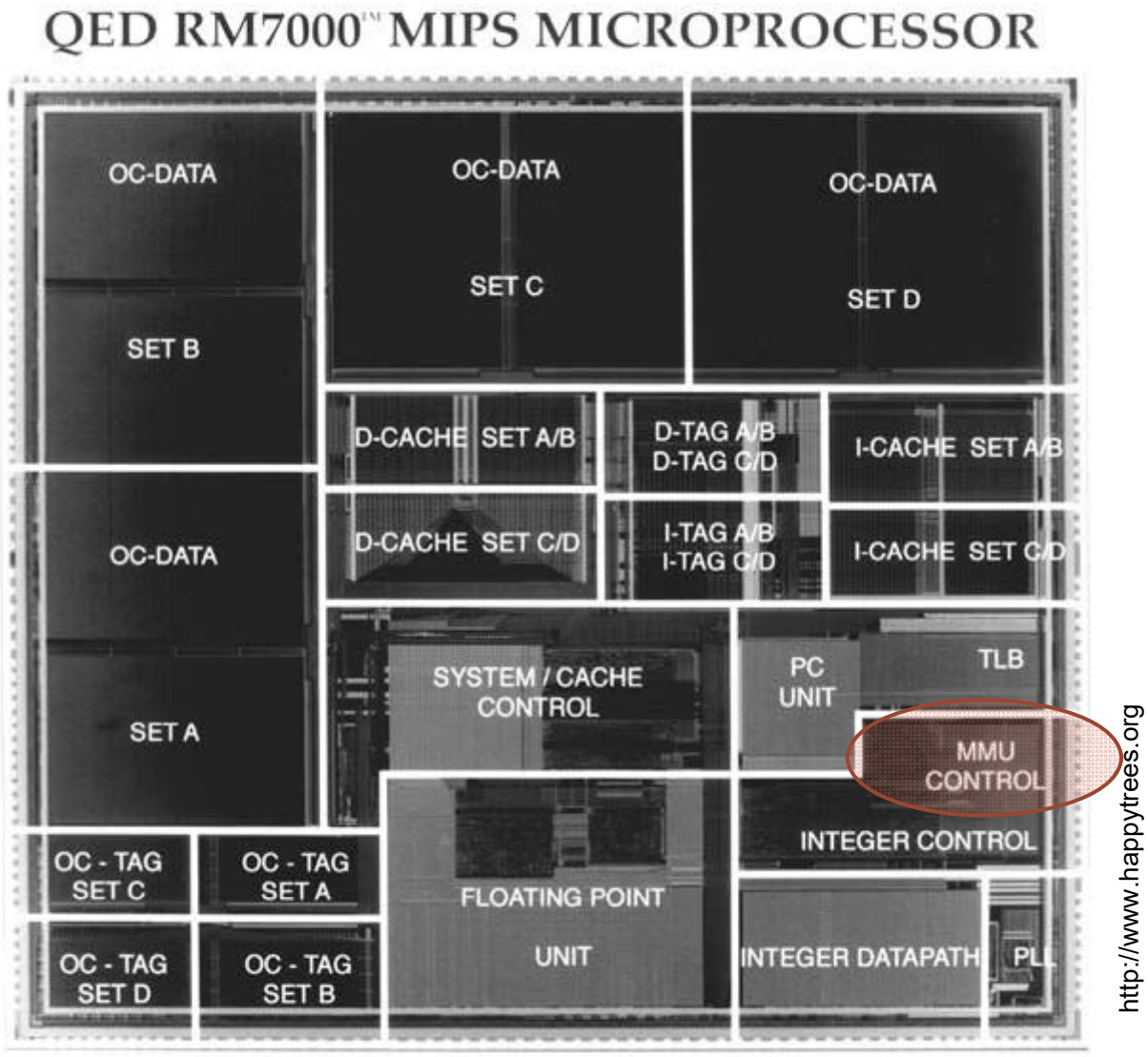
Tradução de Endereços

- Vantagens:
  - O programa não se preocupa c/ endereços físicos
  - Um programa pode ser carregado em qualquer lugar da memória

- Quem faz essa tradução ?
  - Memory Management Unit (MMU)→ Módulo de Hardware na CPU
    - *Também cuida da interface c/ a memória cache.*

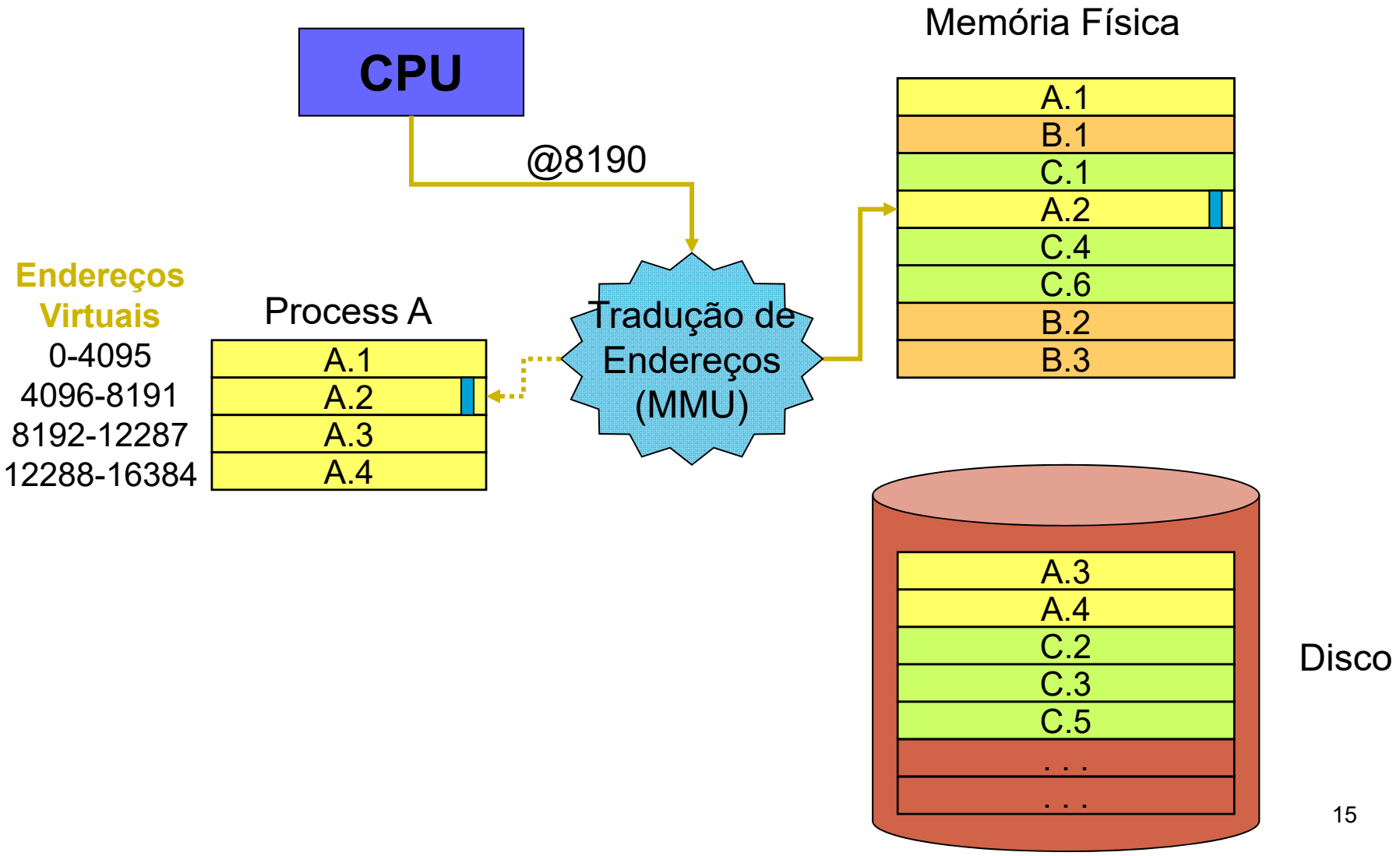
Tradução de Endereços

- **MMU**

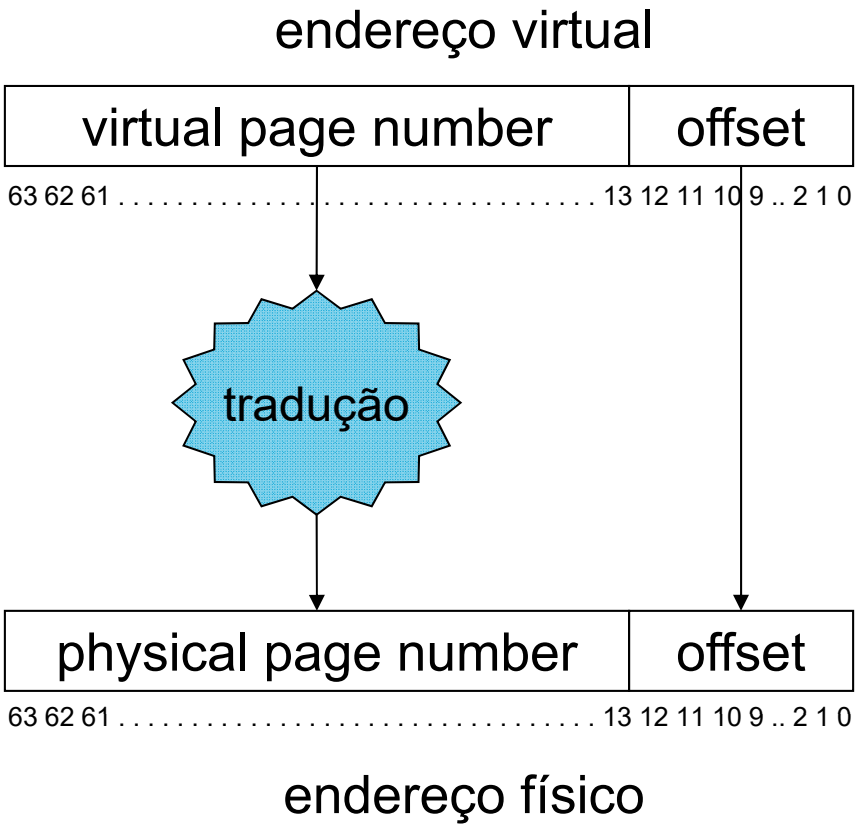


http://www.happytrees.org

Tradução de Endereços



Tradução de Endereços



- Pergunta: como é feita a tradução?
- Resposta: usando uma tabela de páginas (page table)

Tabela de Páginas

- Tabela de Páginas: Armazenada na memória principal
  - Lista contendo ponteiros para o endereço físico de páginas, indexada pelo endereço virtual.
  - Para agilizar o acesso, a CPU tem um registrador de tabela de páginas, apontado para o endereço da tabela na memória principal.
- Se a página virtual está presente na memória RAM:
  - Ponteiro aponta p/ o endereço físico, além de alguns bits de status (refereciada, dirty, etc.)
- Se a página virtual não está presente na memória RAM:
  - Ponteiro aponta p/ um local no espaço de swap do disco

Tradução Usando a Tabela de Páginas

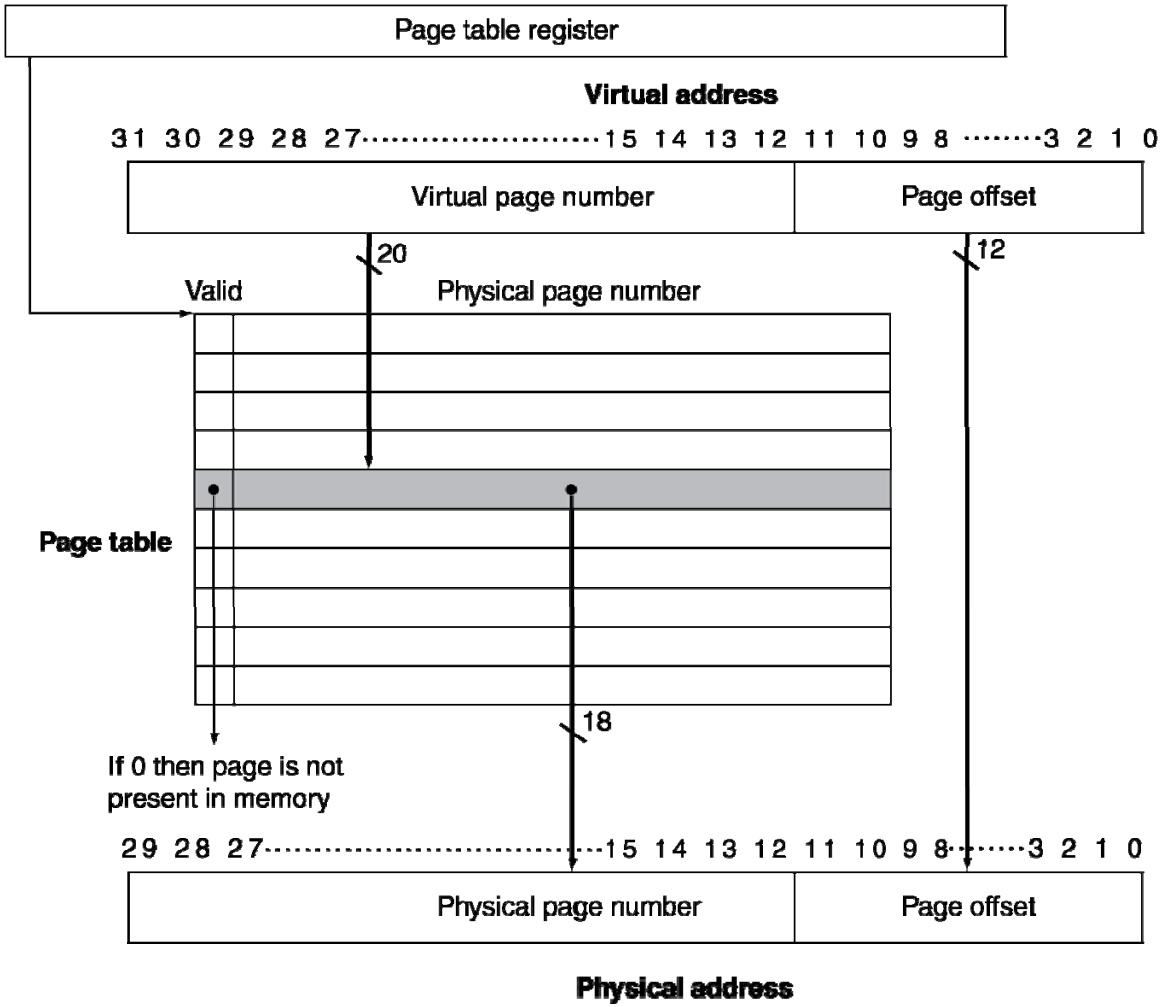


Tabela de Páginas

Tabela de Páginas:

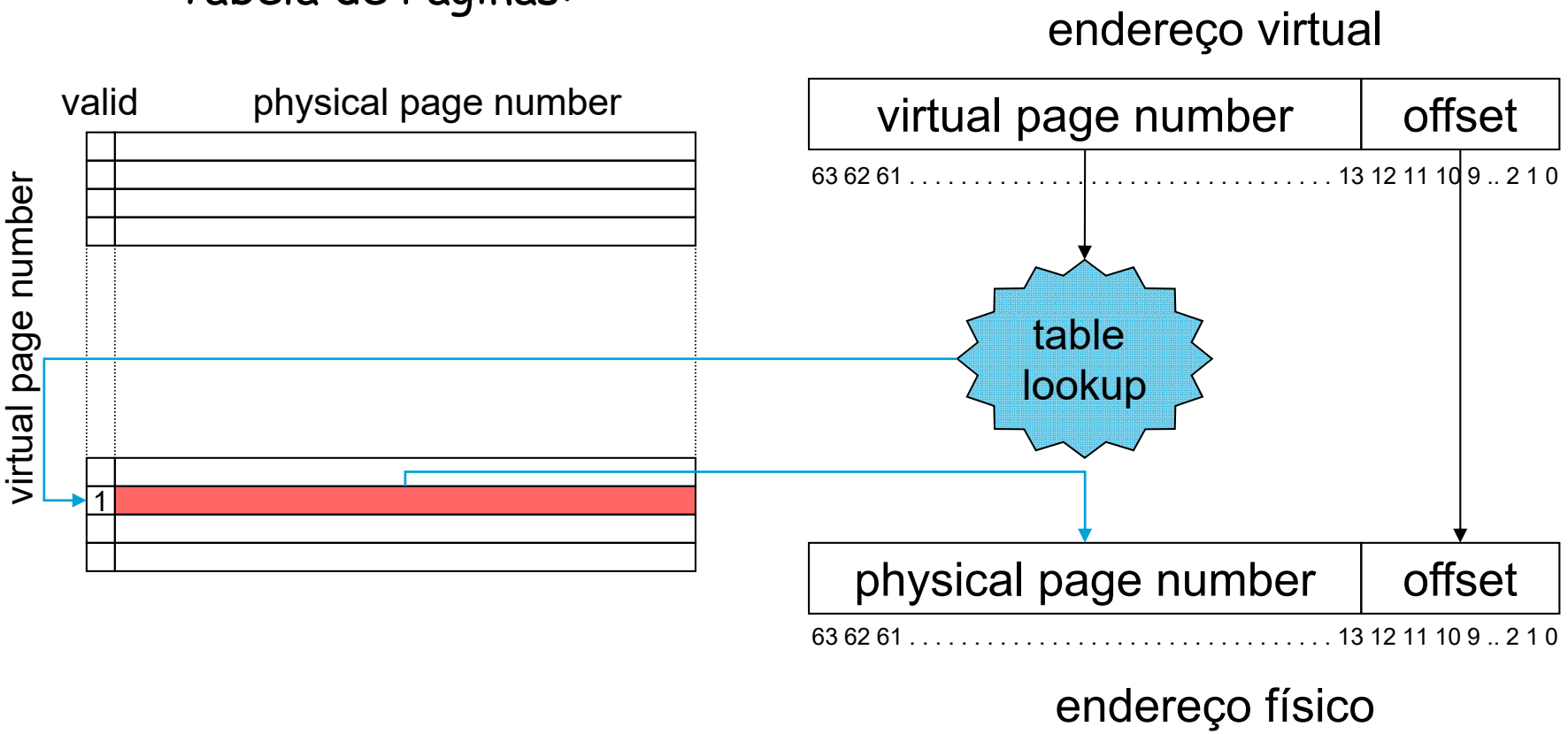


Tabela de Páginas

- Cada processo tem sua própria tabela de páginas
  - O S.O. possui mecanismos para proteger essas tabelas.
- A tabela é armazenada na RAM, e apontada por um registrador especial
- O S.O. é responsável por atualizar a tabela, e fazer como que cada processo utilize a tabela correta.
- Cada tabela tem uma entrada (ponteiro) p/ cada página

Tabela de Páginas

- Tamanho da Tabela de Páginas:
  - Exemplo:
    - Endereços virtuais: 32-bits (total de endereços possíveis= 2^{32})
    - Tamanho da página: 4KB (2^{12} Bytes)
    - Número de páginas= 2^{32} / 2^{12} = 2^{20} páginas
      - A tabela de páginas possui 2^{20} linhas
    - Tamanho da tabela de páginas = 4 x 2^{20} = 4 MB

32bits (4 bytes) por endereço na tabela de páginas

Tamanho normalmente desnecessário

  - Existem esquemas eficientes p/ reduzir o tamanho da tabela.

O que é um processo ?

- Intuitivamente, um processo é um programa sendo executado
- Um definição mais precisa é que um processo é o estado de um programa em execução.
- O estado de um programa em execução é:
  - O valor registrador PC (program counter)
  - O valor dos registradores da CPU
  - A tabela de páginas
- O S.O. trabalha c/ o estado de um programa da seguinte maneira:
  - salvando
  - restaurando

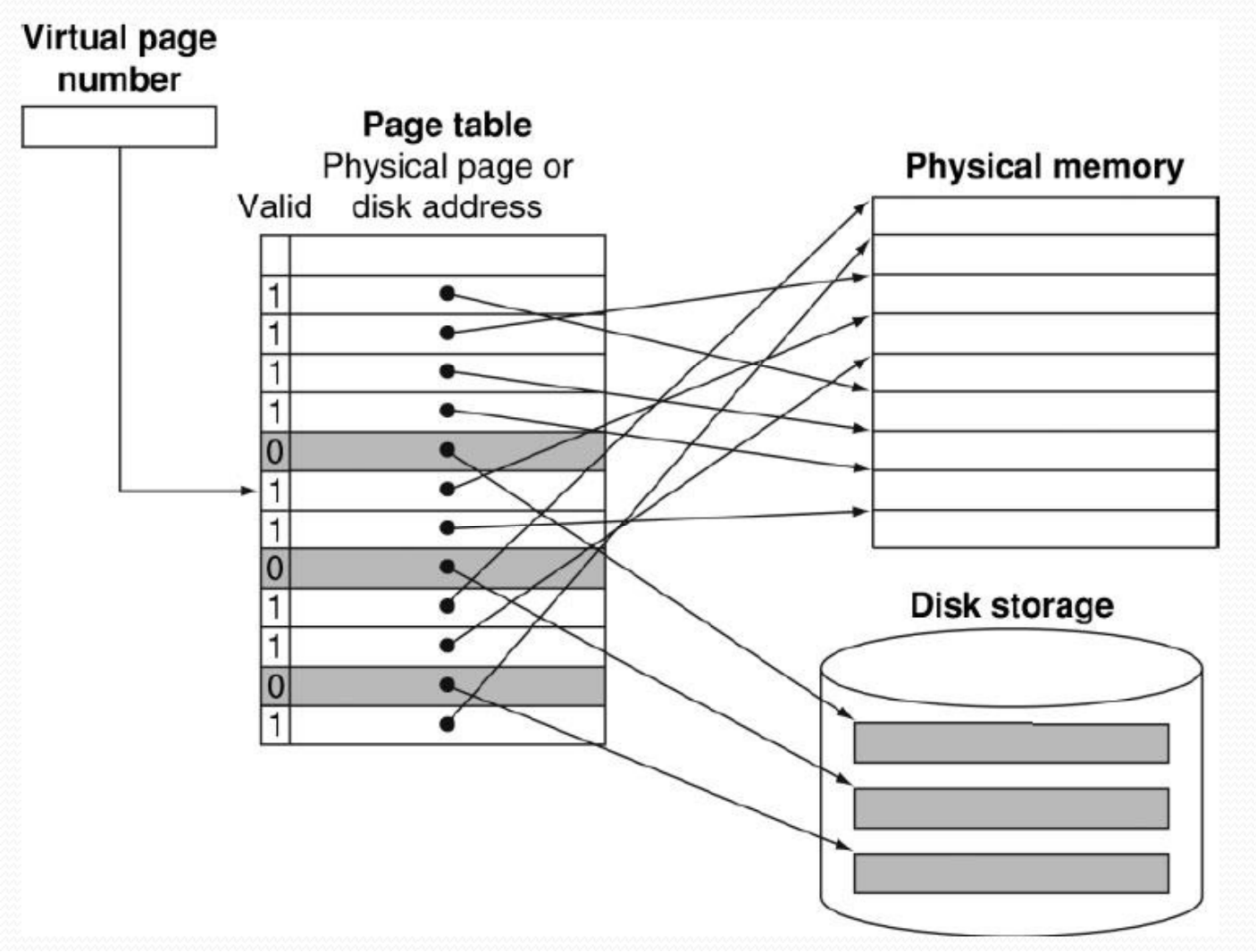
Falta de Página (Page Faults)

- Quando o ponteiro p/ um endereço físico não existe na tabela ocorre a chamada falta de página (page fault)
- Nesses casos, o bit de validade é igual a 0, significando que a página deve ser trazida do disco p/ a memória RAM, substituindo uma outra página física.
- Esse processo é conhecido como "swap de memória".
- Sistemas com pouca memória RAM gastam muito tempo fazendo swap.

Falta de Página (Page Faults)

- Apenas o endereço virtual não permite determinar onde a página desejada está armazenada no disco.
- Deve haver uma forma de rastrear a localização no disco de cada página no espaço de endereçamento virtual.
  - O sistema operacional usualmente cria um espaço (no disco) para todas as páginas de um processo quando este é criado. Este espaço é conhecido como espaço de swap.
  - Neste momento, ele também cria uma estrutura de dados para gravar onde cada página virtual está armazenada no disco. Esta estrutura de dados pode, inclusive, fazer parte – em um sentido lógico – da própria tabela de páginas, já que ambas estruturas são indexadas pelo endereço de página virtual.

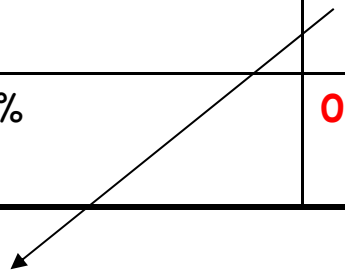
Falta de Página (Page Faults)



Desempenho: Memória Virtual x Cache

Exemplo

Parameter	L1 Cache	Virtual Memory
block (page) size	16-128 bytes	4096-65,536 bytes
Hit time	1-3 cycles	50-150 cycles
Miss penalty	8-150 cycles	**1M-10M cycles**
Miss rate	0.1-10%	**0.00001-0.001%**



não encontrou a página na memória principal:
falta de página (page fault)

Acelerando a Tradução

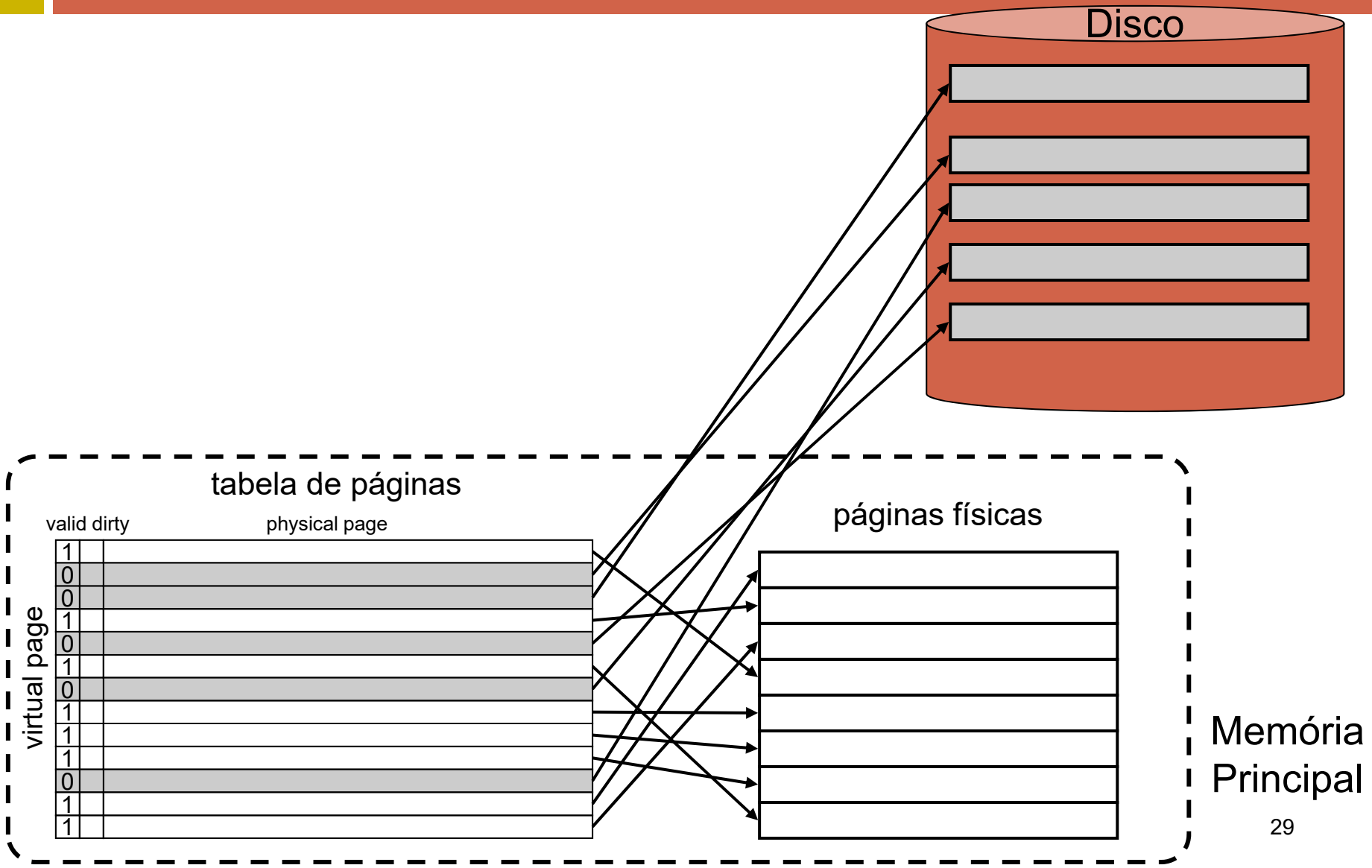
- A tradução de endereços p/ acesso à memória pode exigir vários ciclos:
  - Pesquisar a tabela (table lookup) p/ obter o endereço físico
  - Ler os bytes na página física da memória

- A princípio, tabelas de páginas são armazenadas na memória principal, e não no cache (logo, o acesso não é tão rápido)
  - ...pode até ser que estejam no disco, devido ao swap!

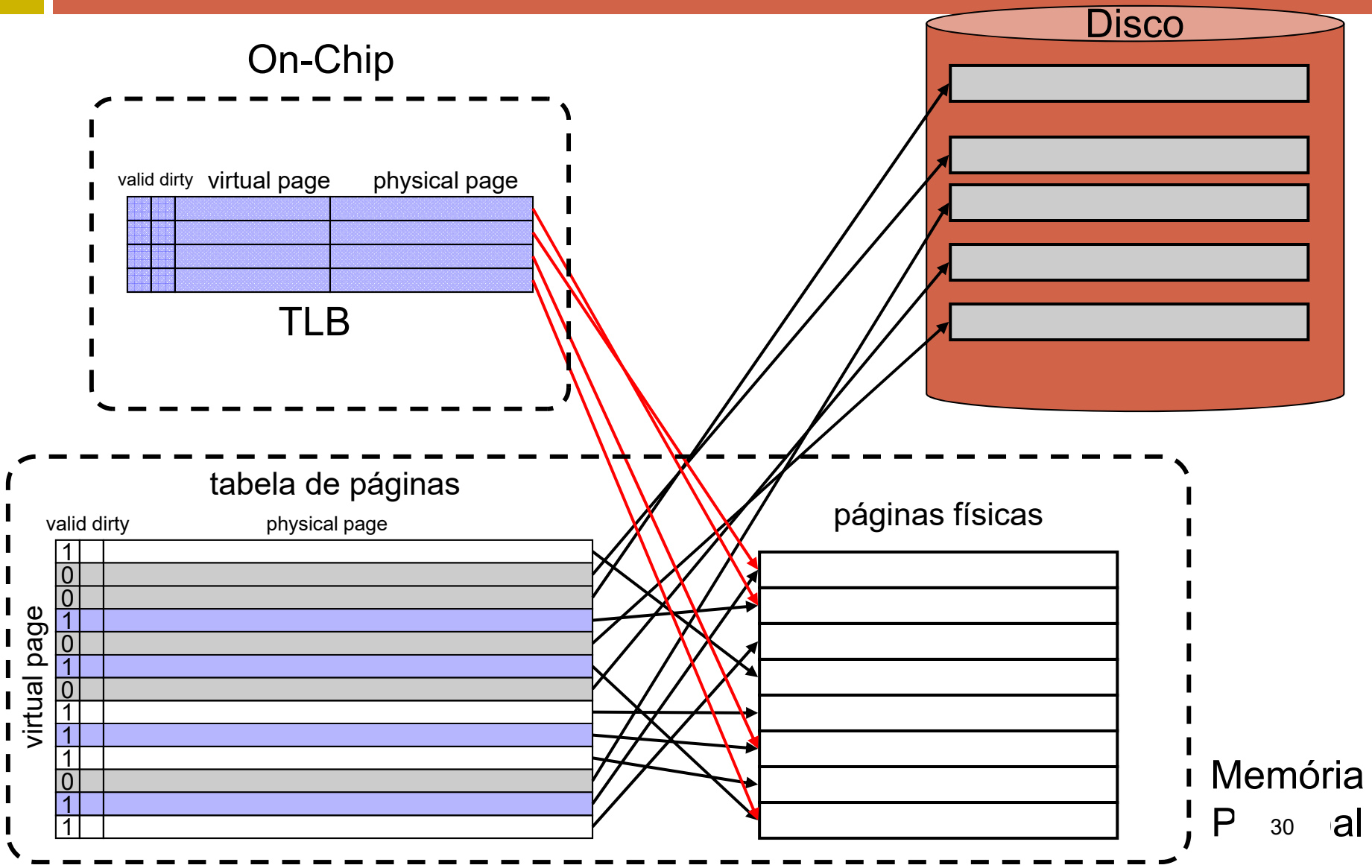
Acelerando a Tradução

- Porém, normalmente os endereços acessados pela CPU possuem localidade espacial.
- Assim, normalmente dois acessos consecutivos irão referenciar a mesma página física, ou seja, seria desnecessário pesquisar a tabela novamente.
- Idéia: Armazene em memóra cache dedicada os resultados de traduções recentes.
- Esse cache dedicado é chamado Translation Lookaside Buffer (TLB)

TLB: Translation Lookaside Buffer



TLB: Translation Lookaside Buffer



TLB: Translation Lookaside Buffer

- TLB é uma memória cache
- Porém, diferente do cache de dados e instruções, não armazena cópias da memória principal, mas sim endereços virtuais e a tradução correspondente.
- A TLB possui informações sobre a qual processo uma dada entrada pertence.

TLB: Translation Lookaside Buffer

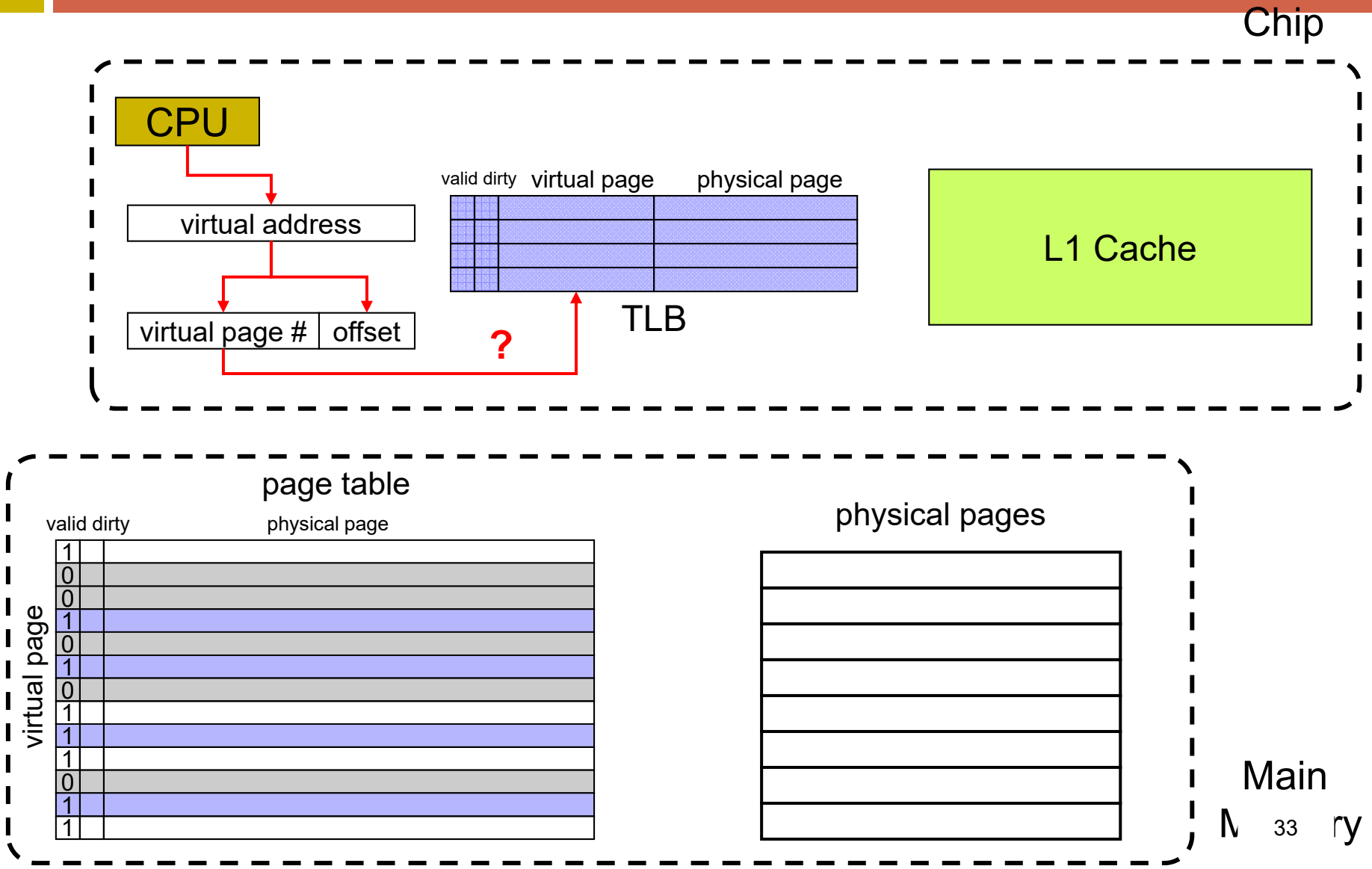
- Política de substituição da TLB

- O que é um TLB miss ?
  - Apenas um miss da tradução em si, ou uma falta de página ?
  - Pode ser tratado por hardware ou software.

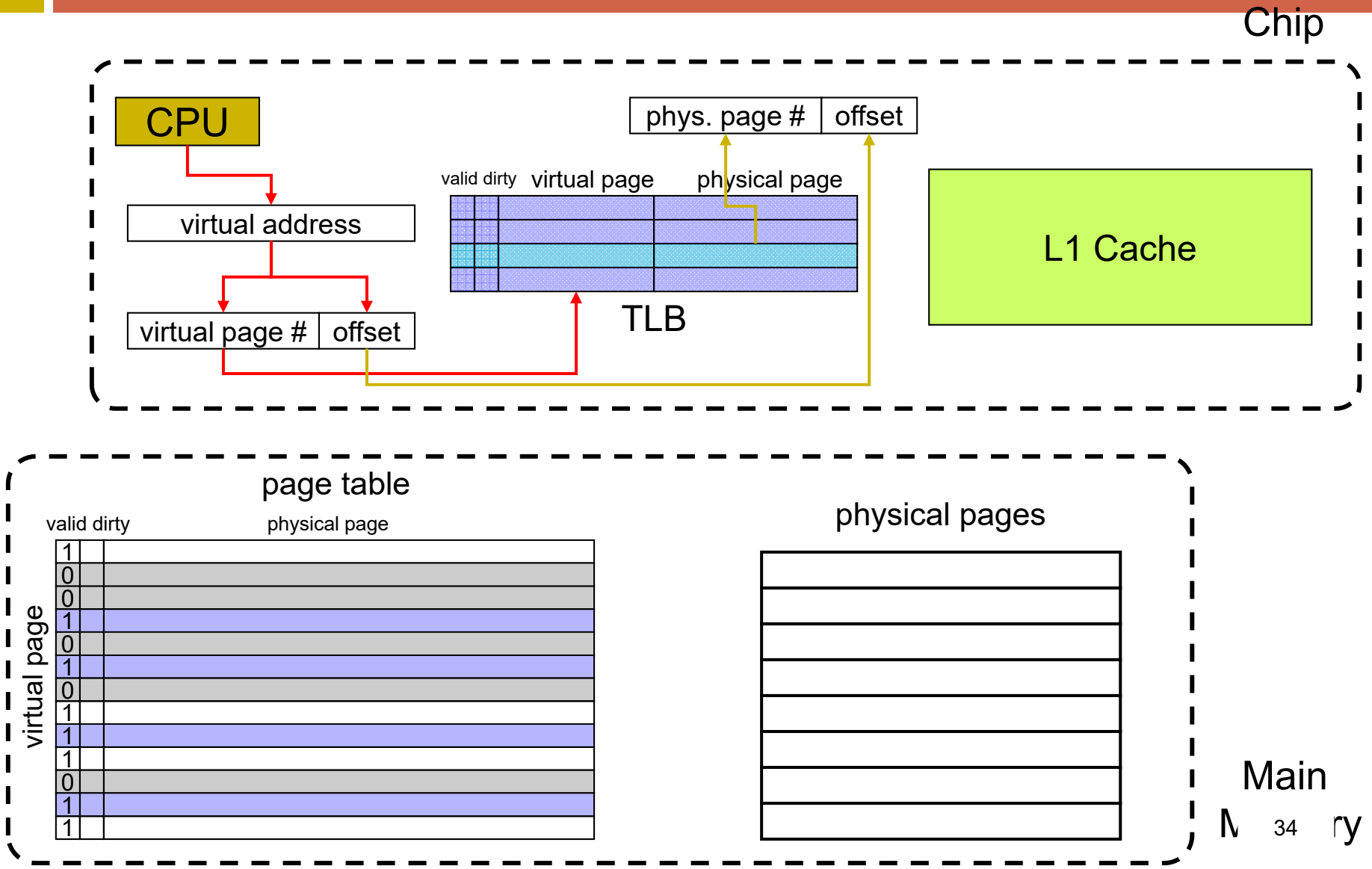
- TLB miss é muito mais frequente que falta de páginas
  - A TLB é pequena (6 a 512 entradas)
  - O número de páginas na memória é grande(ex: 2GB / 4KB > 500,000 pages)

- TLBs possuem várias configurações:
  - Pequena, totalmente associativa
  - Grande, baixa associatividade

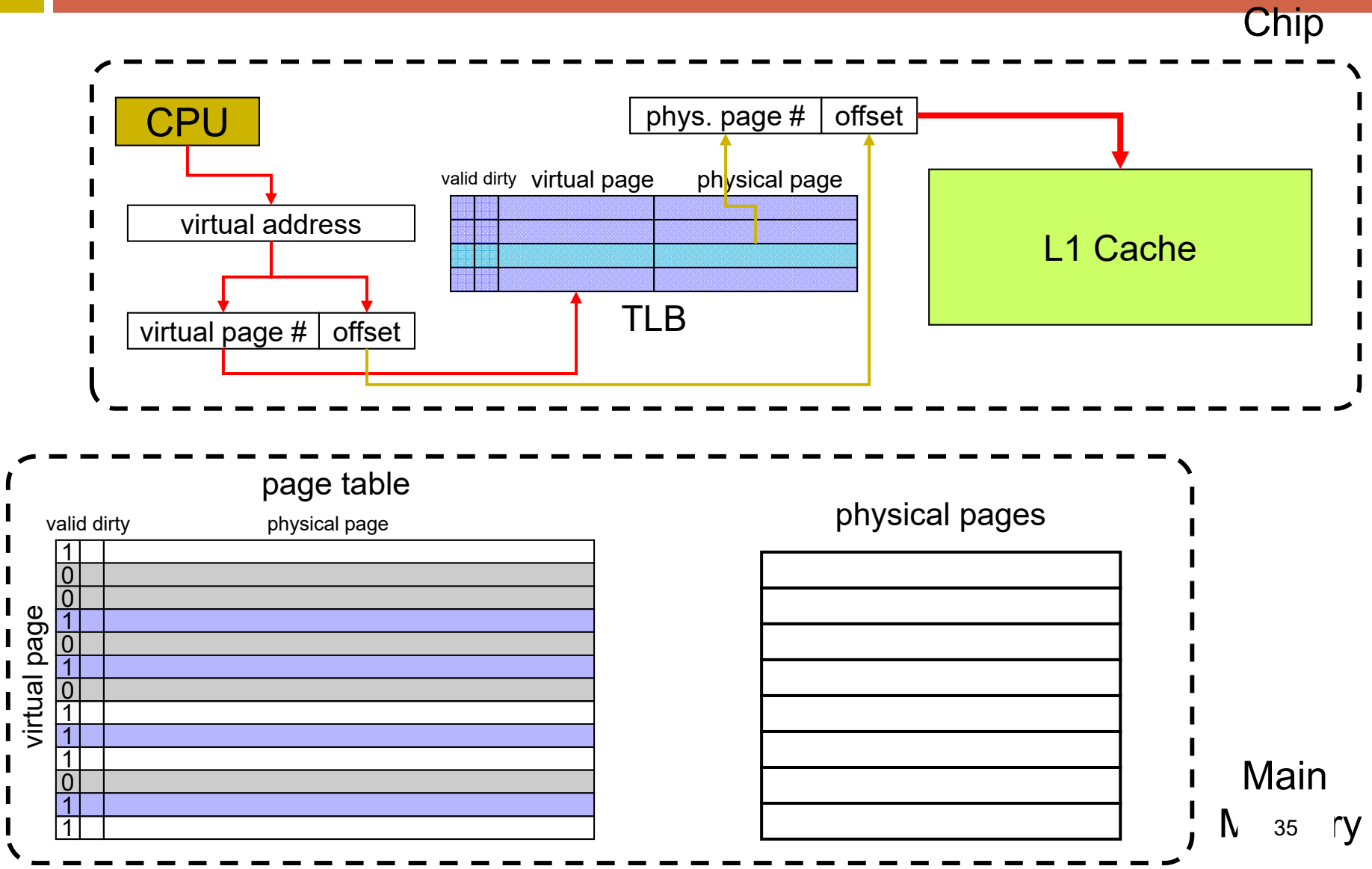
Situação Ideal



Situação Ideal



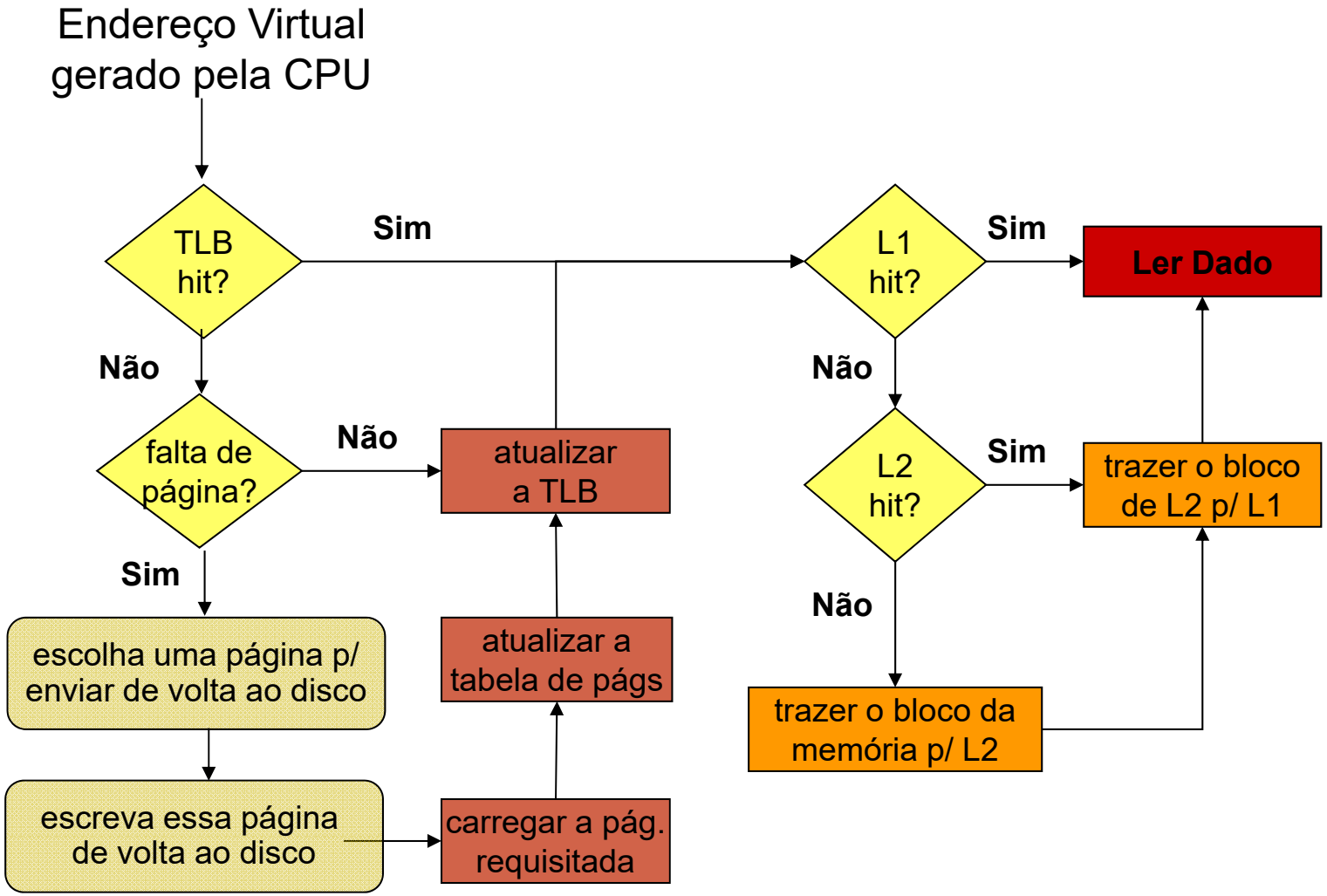
Situação Ideal



Situação Ideal

- A situação ideal mostrada anteriorme depende de:
  - TLB hit
  - L1 cache hit
- Porém, outras situações podem ocorrer
  - TLB miss
  - Page fault
  - Cache miss
- Felizmente, esses casos são relativamente raros, caso contrário os sistemas seriam bem mais lentos do que estamos acostumados.

Possíveis Situações p/ Operação de Leitura



Exemplo de um Sistema Real

Intrinsth FastMath

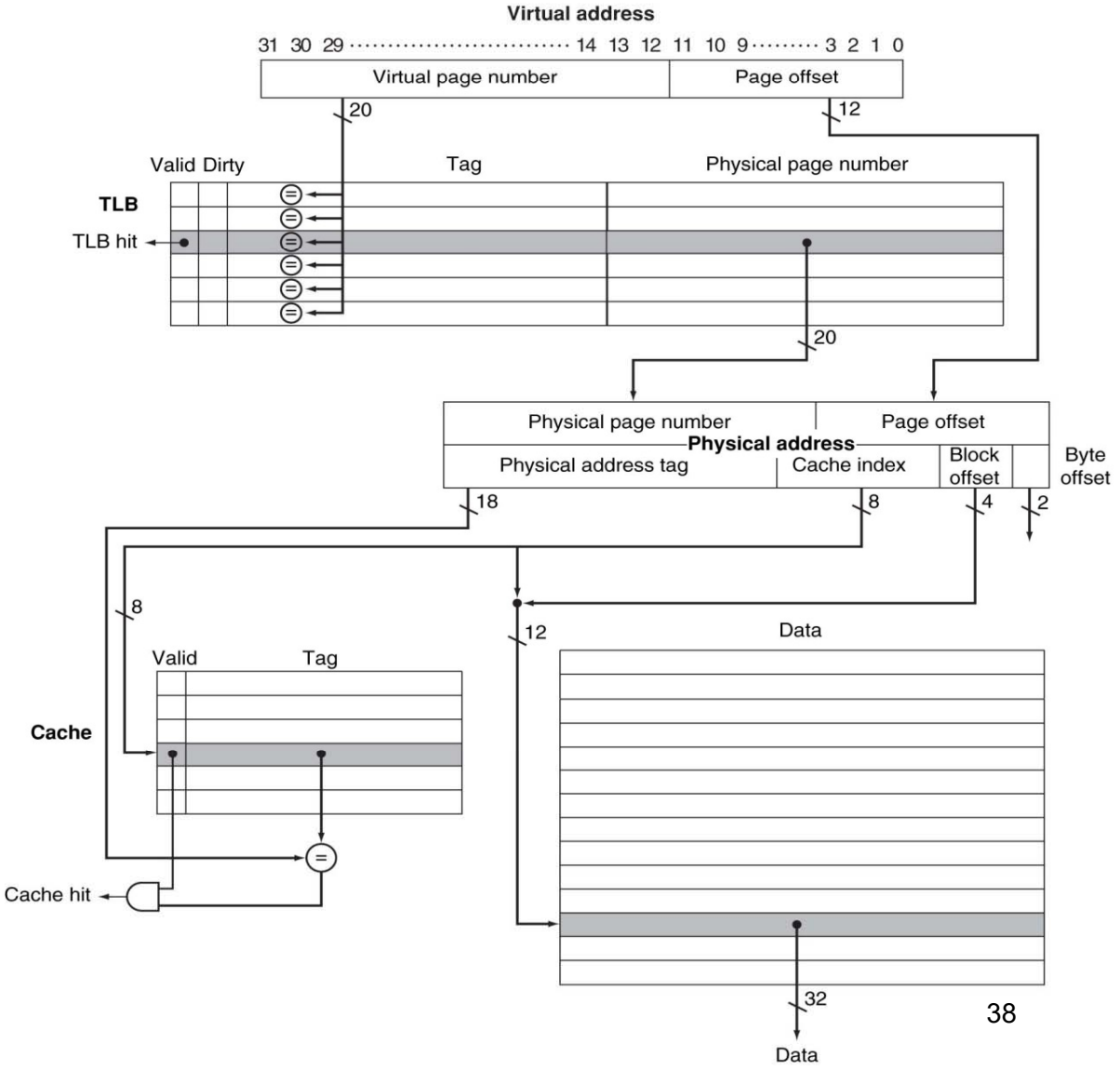


FIGURE 5.24 The TLB and cache implement the process of going from a virtual address to a data item in the Intrinsity Fast MATH. This figure shows the organization of the TLB and the data cache, assuming a 4 KB page size. This diagram focuses on a read; Figure 5.25 describes how to handle writes. Note that unlike Figure 5.9, the tag and data RAMs are split. By addressing the long but narrow data RAM with the cache index concatenated with the block offset, we select the desired word in the block without a 16:1 multiplexor. While the cache is direct mapped, the TLB is fully associative. Implementing a fully associative TLB requires that every TLB tag be compared against the virtual page number, since the entry of interest can be anywhere in the TLB. (See content addressable memories in the *Elaboration* on page 485.) If the valid bit of the matching entry is on, the access is a TLB hit, and bits from the physical page number together with bits from the page offset form the index that is used to access the cache.

Exemplo de um Sistema Real

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

FIGURE 5.38 Address translation and TLB hardware for the Intel Nehalem and AMD Opteron X4. The word size sets the maximum size of the virtual address, but a processor need not use all bits. Both processors provide support for large pages, which are used for things like the operating system or mapping a frame buffer. The large-page scheme avoids using a large number of entries to map a single object that is always present. Nehalem supports two hardware-supported threads per core (see Section 7.5 in Chapter 7).

Salvando e Restaurando o Estado da Máquina

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

FIGURE 5.27 MIPS control registers. These are considered to be in coprocessor 0, and hence are read using mfc0 and written using mtc0.

Salvando e Restaurando o Estado da Máquina

Save state			
Save GPR	addi	$k1,$sp, -XCPSIZE	# save space on stack for state
	sw	$sp, XCT_SP($k1)	# save $sp on stack
	sw	$v0, XCT_V0($k1)	# save $v0 on stack
	...		# save $v1, $ai, $si, $ti,... on stack
	sw	$ra, XCT_RA($k1)	# save $ra on stack
Save hi, lo	mfhi	$v0	# copy Hi
	mflo	$v1	# copy Lo
	sw	$v0, XCT_HI($k1)	# save Hi value on stack
	sw	$v1, XCT_LO($k1)	# save Lo value on stack
Save exception registers	mfc0	$a0, $cr	# copy cause register
	sw	$a0, XCT_CR($k1)	# save $cr value on stack
	...		# save $v1,....
	mfc0	$a3, $sr	# copy status register
	sw	$a3, XCT_SR($k1)	# save $sr on stack
Set sp	move	$sp, $k1	# sp = sp - XCPSIZE
Enable nested exceptions			
	andi	$v0, $a3, MASK1	# $v0 = $sr & MASK1, enable exceptions
	mtc0	$v0, $sr	# $sr = value that enables exceptions
Call C exception handler			
Set $gp	move	$gp, GPINIT	# set $gp to point to heap area
Call C code	move	$a0, $sp	# arg1 = pointer to exception stack
	jal	xcpt_deliver	# call C code to handle exception
Restoring state			
Restore most GPR, hi, lo	move	$at, $sp	# temporary value of $sp
	lw	$ra, XCT_RA($at)	# restore $ra from stack
	...		# restore $t0,....., $a1
	lw	$a0, XCT_A0($k1)	# restore $a0 from stack
Restore status register	lw	$v0, XCT_SR($at)	# load old $sr from stack
	li	$v1, MASK2	# mask to disable exceptions
	and	$v0, $v0, $v1	# $v0 = $sr & MASK2, disable exceptions
	mtc0	$v0, $sr	# set status register
Exception return			
Restore $sp and rest of GPR used as temporary registers	lw	$sp, XCT_SP($at)	# restore $sp from stack
	lw	$v0, XCT_V0($at)	# restore $v0 from stack
	lw	$v1, XCT_V1($at)	# restore $v1 from stack
	lw	$k1, XCT_EPC($at)	# copy old $epc from stack
	lw	$at, XCT_AT($at)	# restore $at from stack
Restore ERC and return	mtc0	$k1, $epc	# restore $epc
	eret	$ra	# return to interrupted instruction

FIGURE 5.28 MIPS code to save and restore state on an exception.

Virtualização

- Permitem isolamento e segurança
- Compartilham um computador entre vários usuários não relacionados entre si.
- Viabilizadas por meio do alto desempenho das máquinas atuais, o que tornou o overhead aceitável.
- Permimtem o uso de diferentes ISAs e Sistemas Operacionais em uma única máquina.
  - SVM (System Virtual Machines)
    - Virtual machine monitor ou hypervisor.
  - Virtual machines que rodam sob a supervisão do monitor são chamadas "guest VMs"

Conclusão

- A hierarquia de memórias é um dos aspectos mais complexos, interessantes e desenvolvidos nos computadores modernos.
- Assim como a memória cache, a memória virtual é extremamente necessária para que se consiga o desempenho necessário.
- Vários outros aspectos relacionados com memória virtual são tratados em cursos sobre sistemas operacionais.