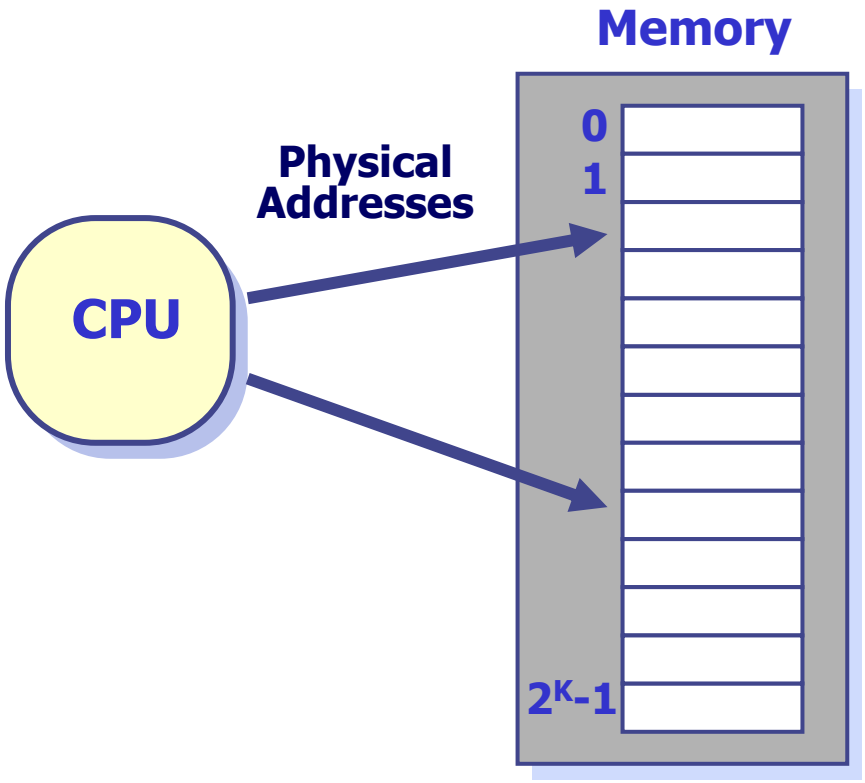
Aulas 24, 25 e 26

- Memória Virtual
  - Motivações para a sua utilização
  - Endereço virtual, endereço físico
  - *Page Table*. Tradução de endereços. *Page Fault*
  - "Translation-lookaside buffer" (TLB)
  - Integração da memória virtual com TLB e *cache*

José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

Sistema apenas com memória física

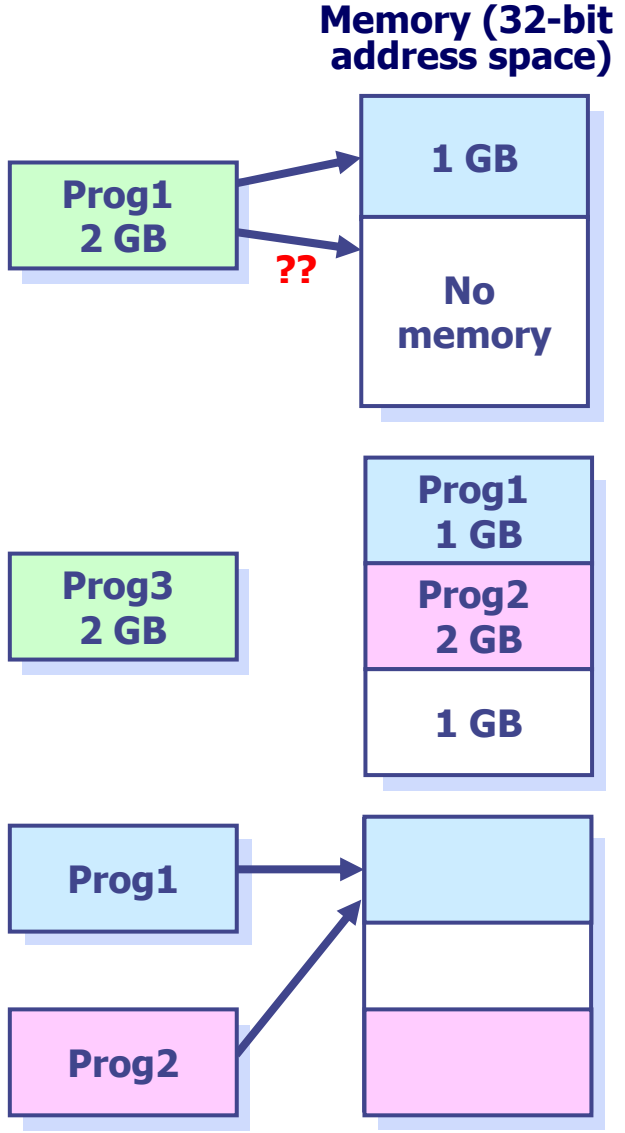
- Num sistema apenas com memória física



- Os endereços gerados pelo CPU apontam diretamente para as posições de memória a que este pretende aceder

Sistema apenas com memória física – problemas

- Memória insuficiente
  - O que acontece se o programa Prog1 tentar aceder a mais do que 1 GB de memória?
- Gestão da memória disponível
  - Supondo que o sistema tem 4 GB de memória, os programas Prog1 e Prog2 usam 3 GB. Quando Prog1 terminar ficam 2GB disponíveis, mas Prog3 não pode executar porque não existe um bloco de 2 GB
  - Fragmentação da memória (espaço não utilizado de forma contínua)
- Segurança
  - Um programa pode escrever fora da zona de memória que lhe está atribuída



Sistema apenas com memória física – problemas

- Num sistema multi-processo, a memória disponível tem de ser partilhada pelos processos em execução
  - Cada processo tem as suas próprias necessidades de memória
  - Como gerir adequadamente a memória disponível entre os vários processos?
  - O que fazer se a memória necessária para executar todos os processos for superior à memória física disponível?
- A memória pode facilmente ser corrompida se um processo escrever (de forma involuntária ou intencional) na zona de memória atribuída a outro processo
  - O processo que viu a sua zona de memória alterada falhará por razões que nada têm a ver com a sua lógica de funcionamento
  - Como garantir que um processo não escreve na zona de memória atribuída a outro processo?

Memória virtual

- É uma abstração que permite uma utilização eficiente do sistema de memória em sistemas multi-processo (mantendo o modelo de memória hierárquica)
- Esta técnica é usada em sistemas de computação geral baseados em microprocessador, com sistema operativo (não usado em sistemas simples baseados em microcontrolador)
- O conceito de "memória virtual" não é novo: descrito pela primeira vez por Kilburn et al. em 1962

Motivações para a utilização de Memória virtual

- **Eficiência na utilização da memória**
  - Memória deve ser partilhada, de forma eficiente, pelos vários processos em execução
  - Em memória apenas deve residir a informação necessária de cada processo
- **Segurança**
  - Devem existir mecanismos de segurança que impeçam que um processo altere as zonas de memória dos outros processos
- **Transparência**
  - Um processo deve ter acesso à memória de que necessita (eventualmente mais do que a memória física DRAM)
  - Um processo deve correr como se toda a memória lhe pertencesse
- **Partilha de memória**
  - Vários processos devem poder aceder à mesma zona de memória (de forma controlada)

Memória virtual - soluções

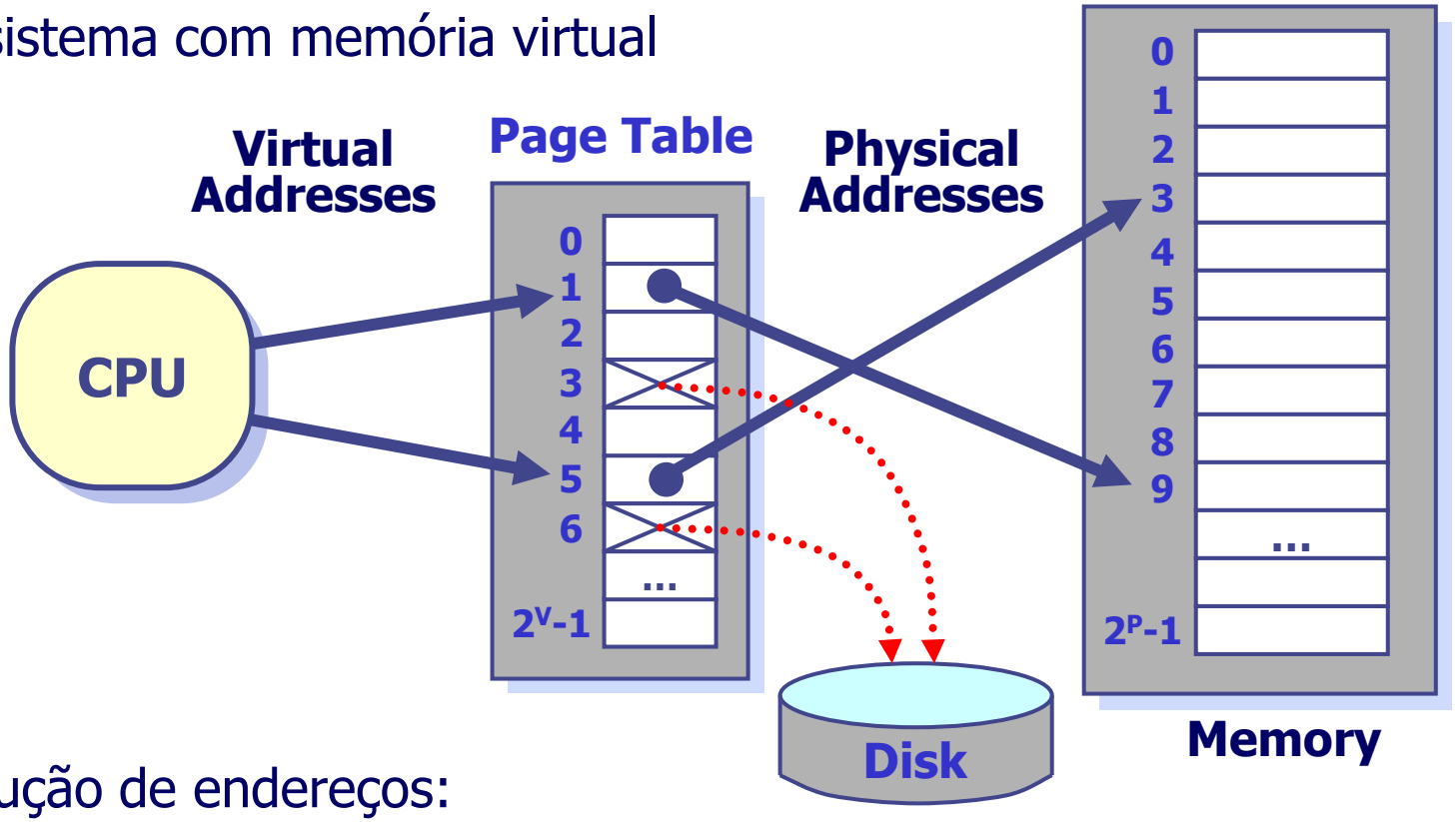
- Vários programas podem estar em execução "simultânea" no computador
- A memória pode estar fragmentada
- A quantidade de memória total necessária para executar todos os programas pode ser superior à memória física disponível
- **Utilização eficiente da memória física**
  - Apenas uma fração da memória física disponível está ativamente em uso num determinado momento
  - Apenas as zonas ativas de todos os programas em execução (instruções e dados) precisam de ser mantidas em memória, enquanto as restantes podem ser armazenadas em disco (memória secundária), libertando espaço na memória física
  - O sistema operativo pode alocar mais memória para um processo conforme necessário ou libertar memória que já não está a ser usada, de acordo com as necessidades do sistema e os padrões de utilização de memória dos processos

Memória virtual - soluções

- **Segurança**
  - Múltiplos processos partilham a mesma memória física
  - É necessário usar mecanismos de proteção que assegurem que um dado processo apenas acede (leitura e/ou escrita) às zonas de memória que lhe foram atribuídas
  - Endereços usados pelos processos não são endereços da memória física
- **Transparência**
  - Cada processo tem o seu próprio espaço de endereçamento que pode usar na totalidade, de forma exclusiva
  - Cada processo executa como se fosse o único a ocupar a memória
- **Partilha de memória**
  - Com a implementação de mecanismos de proteção torna-se possível a partilha de zonas de memória entre processos

Princípio de funcionamento

- Um sistema com memória virtual



- Tradução de endereços:
  - Os endereços virtuais gerados pelo CPU são convertidos para endereços físicos através de uma tabela, designada por "Page Table"
  - A tradução de endereços tem que ser temporalmente eficiente

Nomenclatura-base

- **Endereço virtual** (*virtual address*)
  - Um endereço gerado pelo CPU
  - O **espaço de endereçamento virtual** é a coleção de todos os endereços virtuais (0 a 2^V-1, num espaço de endereçamento de V bits)
- **Endereço físico** (*physical address*)
  - Um endereço da memória principal (e.g. DRAM)
  - O **espaço de endereçamento físico** é a coleção de todos os endereços físicos (0 a 2^P-1, num espaço de endereçamento de P bits)
- **Tradução de endereços** (*address translation*)
  - O processo pelo qual um endereço virtual é traduzido num endereço físico
  - Um endereço virtual é traduzido num endereço físico quando o CPU acede à memória (para ler uma instrução ou para aceder a uma palavra de dados)

Princípio de funcionamento

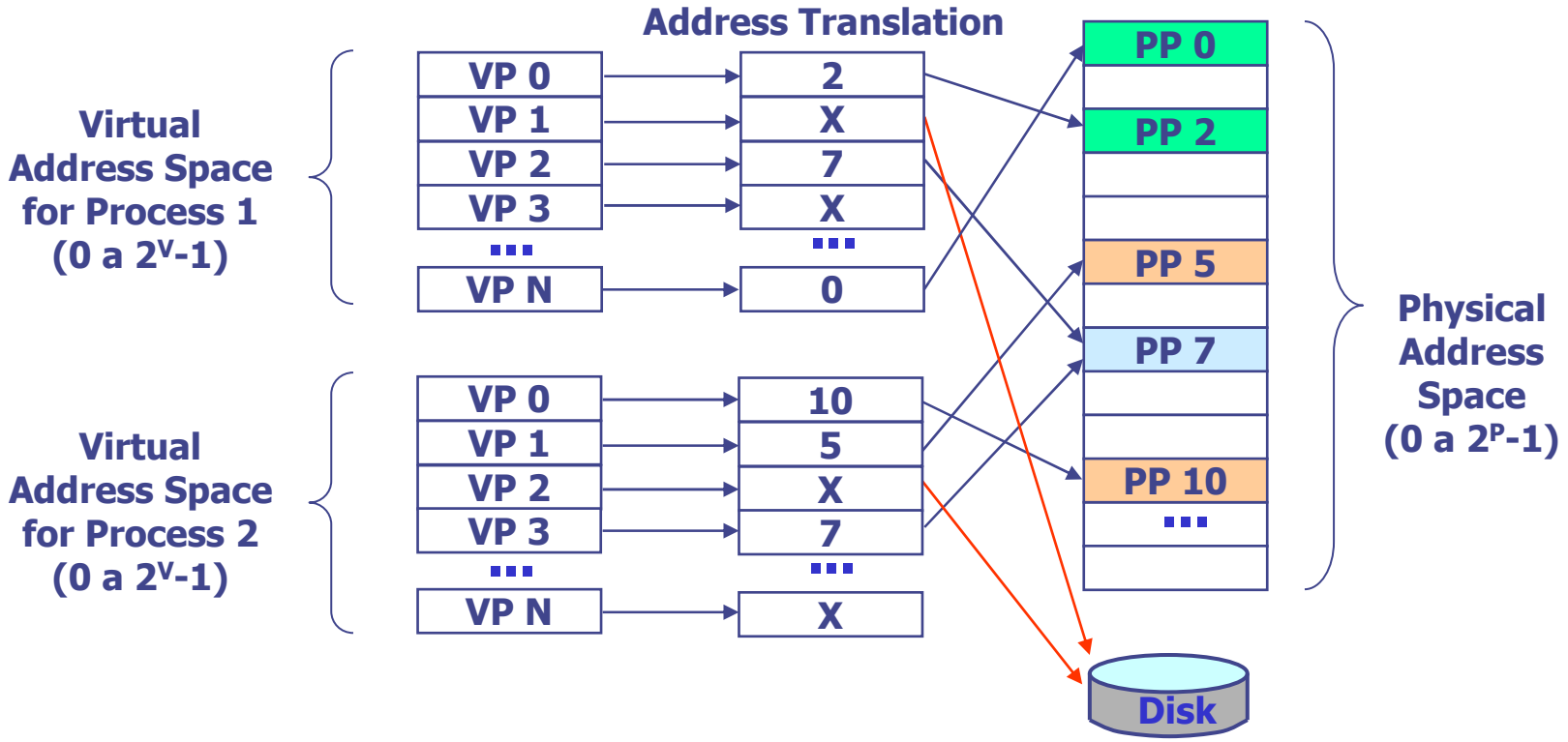
- Os espaços de endereçamento virtual e físico são divididos em blocos; os blocos têm a mesma dimensão nos dois espaços de endereçamento
  - Na terminologia de memória virtual estes blocos designam-se por "**páginas**" (tamanho varia, mas é geralmente uma potência de 2)
- Cada processo tem o seu próprio espaço de endereçamento -> **espaço de endereçamento virtual**
  - O espaço de endereçamento virtual de cada processo pode seguir um modelo comum, ou seja, a sua estrutura pode ser similar entre os processos
  - Todos os processos podem iniciar no mesmo endereço, ter a primeira instrução no mesmo endereço, os dados na mesma zona de memória e a *stack* na mesma posição dentro do seu espaço de endereçamento virtual
  - Contudo, o mapeamento físico dessas zonas será independente para cada processo, permitindo que os endereços virtuais de diferentes processos possam corresponder a diferentes endereços físicos

Princípio de funcionamento

- A atribuição de **páginas físicas** a **páginas virtuais** é gerida pelo sistema operativo
- Quando um processo é carregado na memória, o sistema operativo atribui páginas de memória virtual do processo a páginas de memória física disponíveis, construindo a *Page Table* do processo. A *Page Table* é a estrutura responsável por manter esse mapeamento
- O programa pode ser carregado em qualquer local da memória física, sem a preocupação com a localização específica dos dados e instruções, desde que o sistema tenha o mapeamento correto na *Page Table*
- Durante a execução do processo, a tradução de endereços virtuais em endereços físicos é feita por hardware, num componente do sistema designado por MMU (Memory Management Unit)

Princípio de funcionamento

- Espaço de endereçamento virtual de V bits
- Espaço de endereçamento físico de P bits

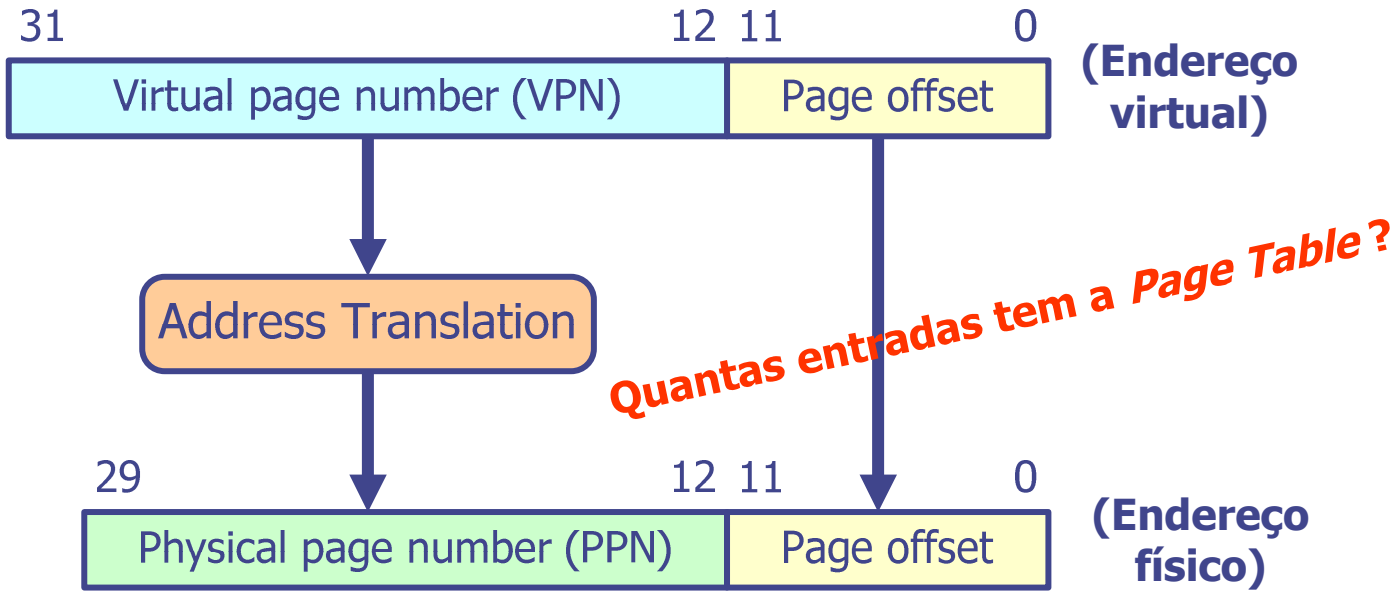


VP ≡ Virtual page
PP ≡ Physical page

Neste exemplo a página física 7 é partilhada pelos dois processos com endereços virtuais diferentes (exemplo de um zona de memória partilhada: leitura ou leitura/escrita)

Princípio de funcionamento

- Mapeamento entre um endereço virtual (V bits) e um endereço físico (P bits) - exemplo c/ **V = 32 bits** e **P = 30 bits** e páginas de 4 kBytes

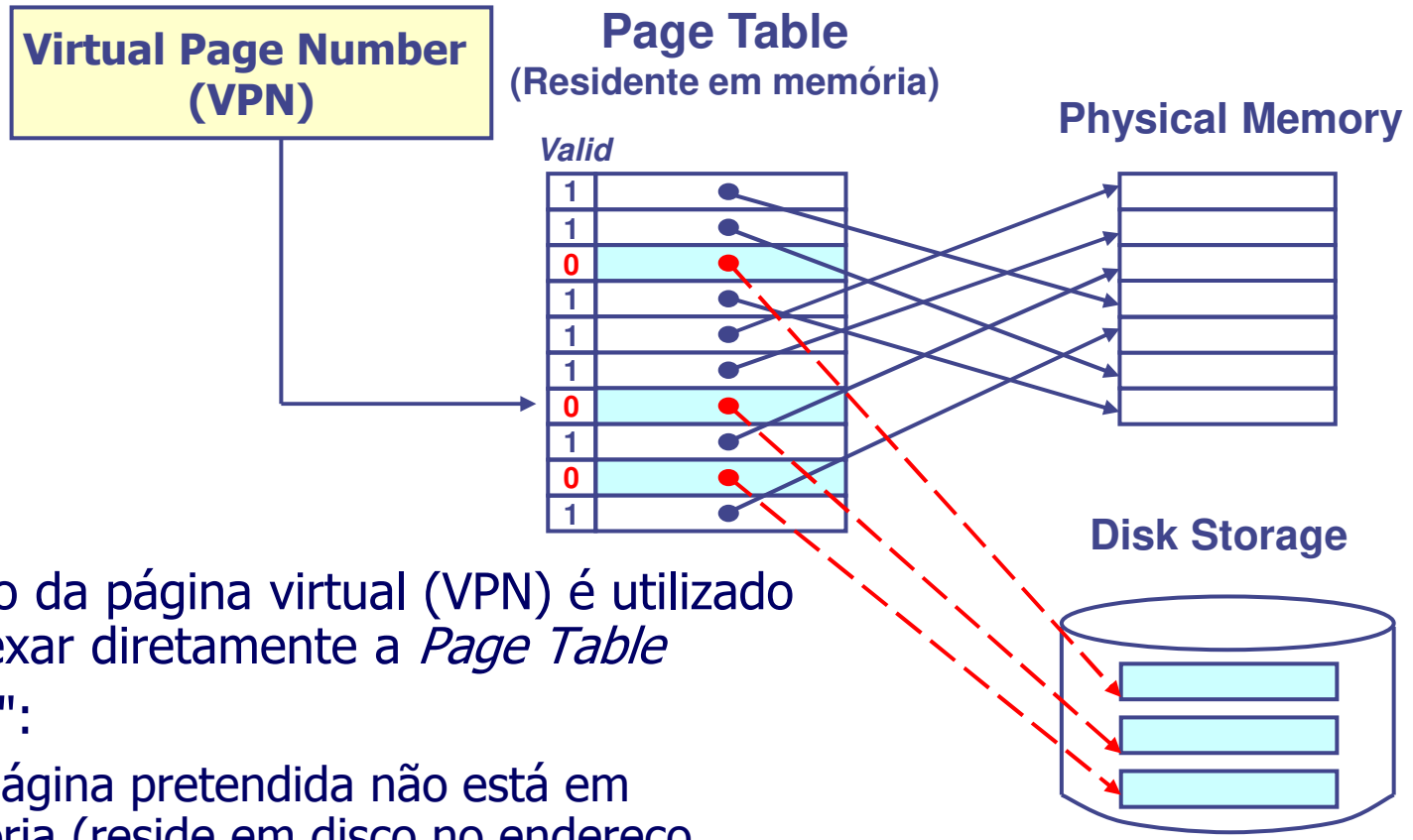


  - Espaço de endereçamento virtual: 2^{32} = 4 GB
  - Espaço de endereçamento físico: 2^{30} = 1 GB
  - Dimensão da página: 2^{12} = 4 kB
  - Número de páginas da memória virtual: 2^{20} = 1 M
  - Número de páginas da memória física: 2^{18} = 256k

Tradução de endereços

- A *Page Table* contém um número de entradas igual ao número máximo de páginas virtuais (para o exemplo dado anteriormente, a tabela teria 2^{20} entradas). Por esta razão não é necessária uma "tag"
- Cada entrada da *Page Table* é responsável por mapear uma página virtual para uma página física (ou indicar que a página não está em memória); ou seja, cada entrada contém o endereço da página física correspondente à página virtual (e ainda outras informações sobre o estado da página, ou atributos de acesso à página)
- O número da página virtual (VPN) é utilizado para indexar diretamente a *Page Table*
- A tradução de endereços tem que ser rápida, uma vez que ocorre em cada acesso do CPU à memória. Realizada por hardware (MMU)

Tradução de endereços

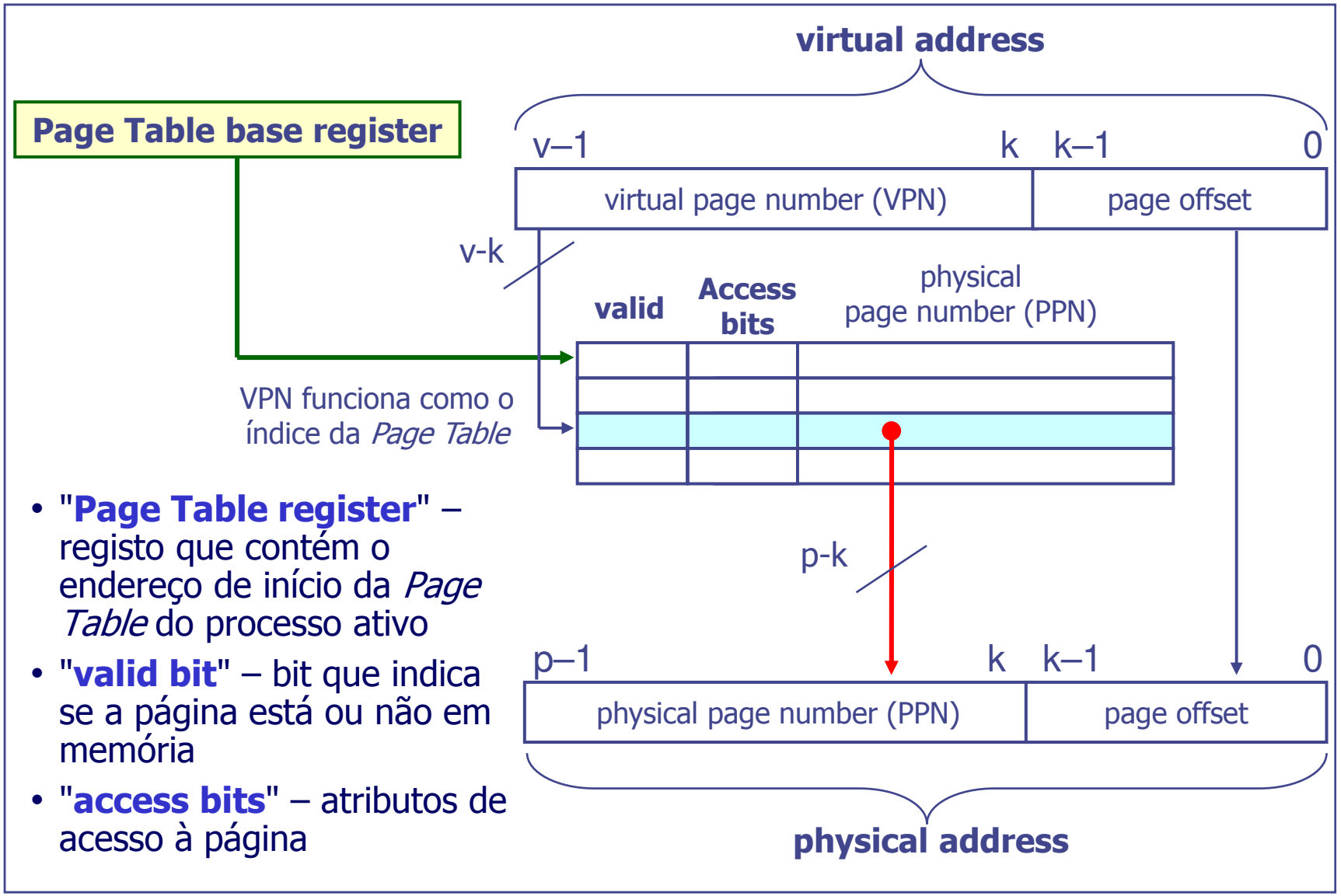


- O número da página virtual (VPN) é utilizado para indexar diretamente a *Page Table*
- "Valid bit":
  - **0**: a página pretendida não está em memória (reside em disco no endereço especificado – *Page Fault*, ou não existe)
  - **1**: a *Page Table* fornece o número da página física (a partir do qual se obtém o endereço inicial da página em memória)

Operação

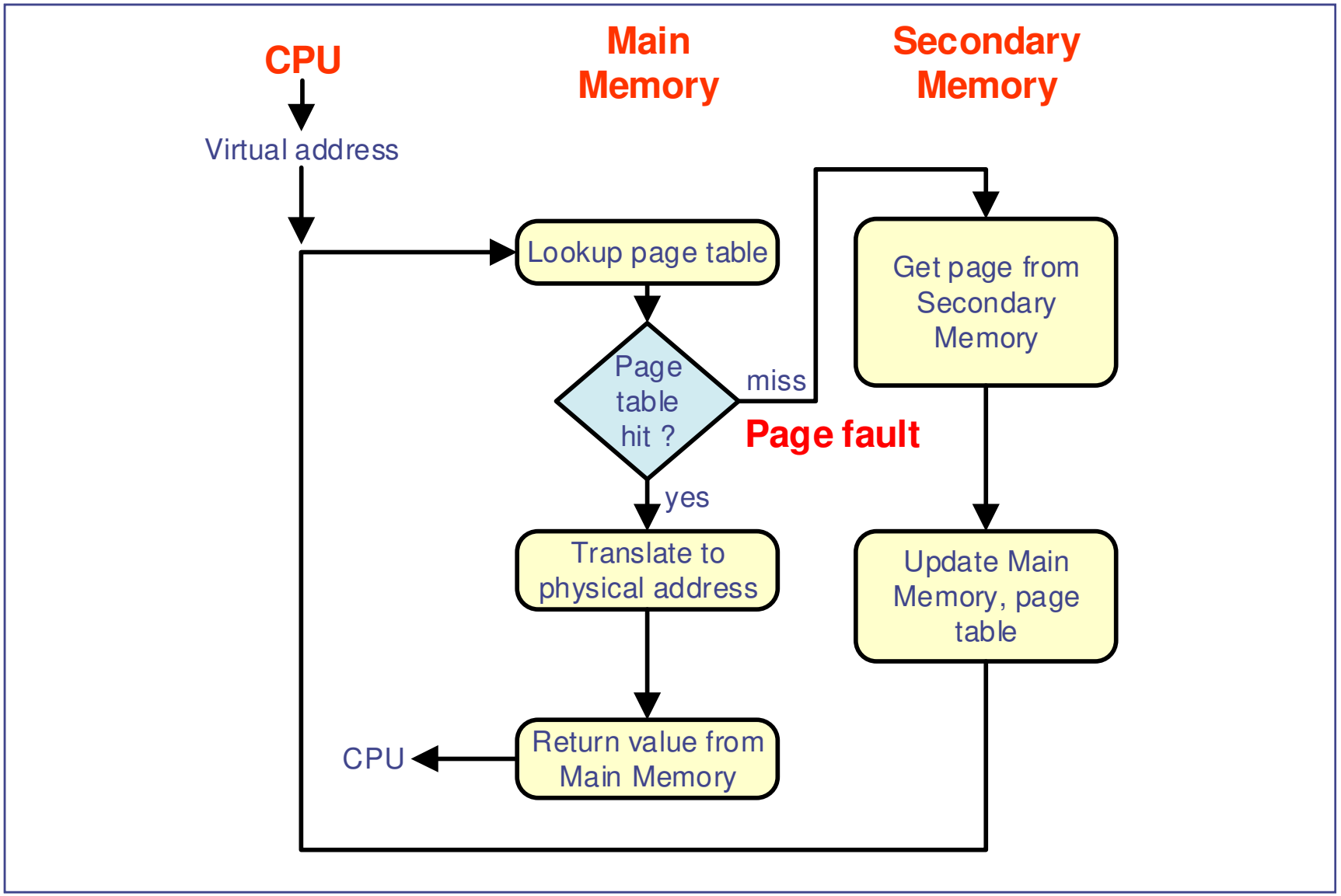
- Tradução
  - Uma *Page Table* por processo
  - O Virtual Page Number (VPN), obtido do endereço gerado pelo processador, forma o índice de acesso à *Page Table*
- Obtenção do endereço físico da página
  - A entrada da tabela correspondente ao VPN contém informação sobre a página
  - Se "valid bit = 1" a página reside na memória
    - O endereço de acesso é obtido da entrada da tabela correspondente ao VPN concatenado com o "page offset"
  - Se "valid bit = 0" a página reside no disco ou não existe
    - *Page Fault*
    - Copiar para a memória a página pretendida (pode envolver a substituição de uma página residente em memória)
    - "valid bit" passa a "1"
- Proteção
  - Os atributos de acesso à página (Read, Read/Write, Execute) são verificados em cada tradução (para cada VPN)

Tradução de endereços



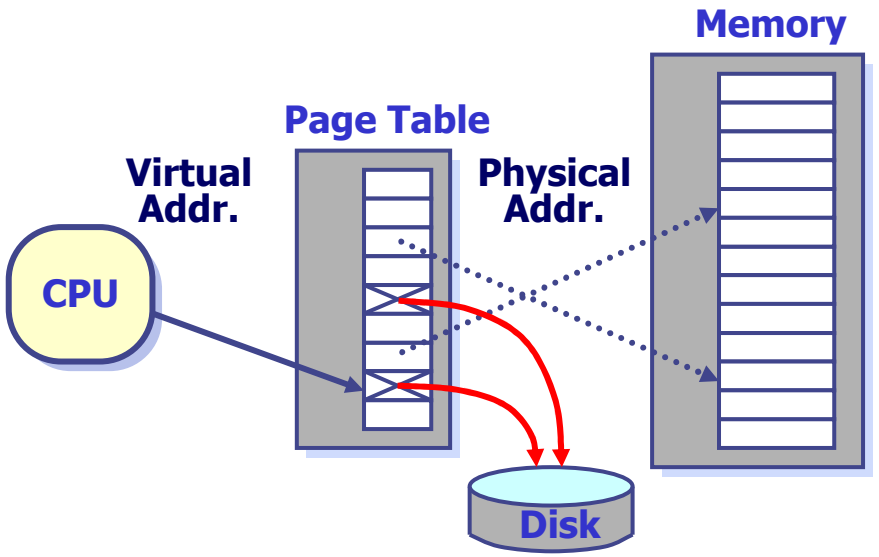
- "**Page Table register**" – registo que contém o endereço de início da *Page Table* do processo ativo
- "**valid bit**" – bit que indica se a página está ou não em memória
- "**access bits**" – atributos de acesso à página

"Page Fault"

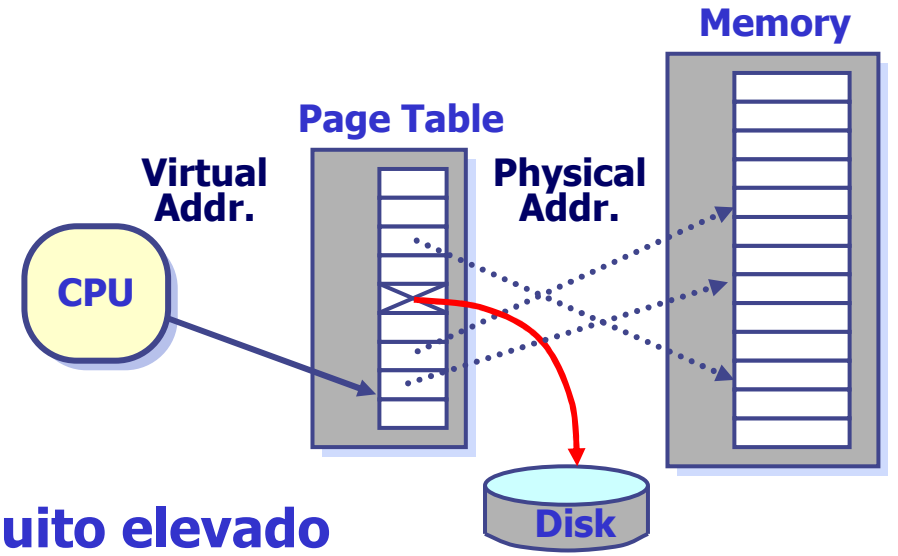

CPU
Main Memory
Secondary Memory
Virtual address
Lookup page table
Page table hit ?
miss
Page fault
Get page from Secondary Memory
yes
Translate to physical address
Update Main Memory, page table
CPU
Return value from Main Memory

"Page Fault"

- Ocorrência de um *Page Fault*



- Após um *Page Fault*



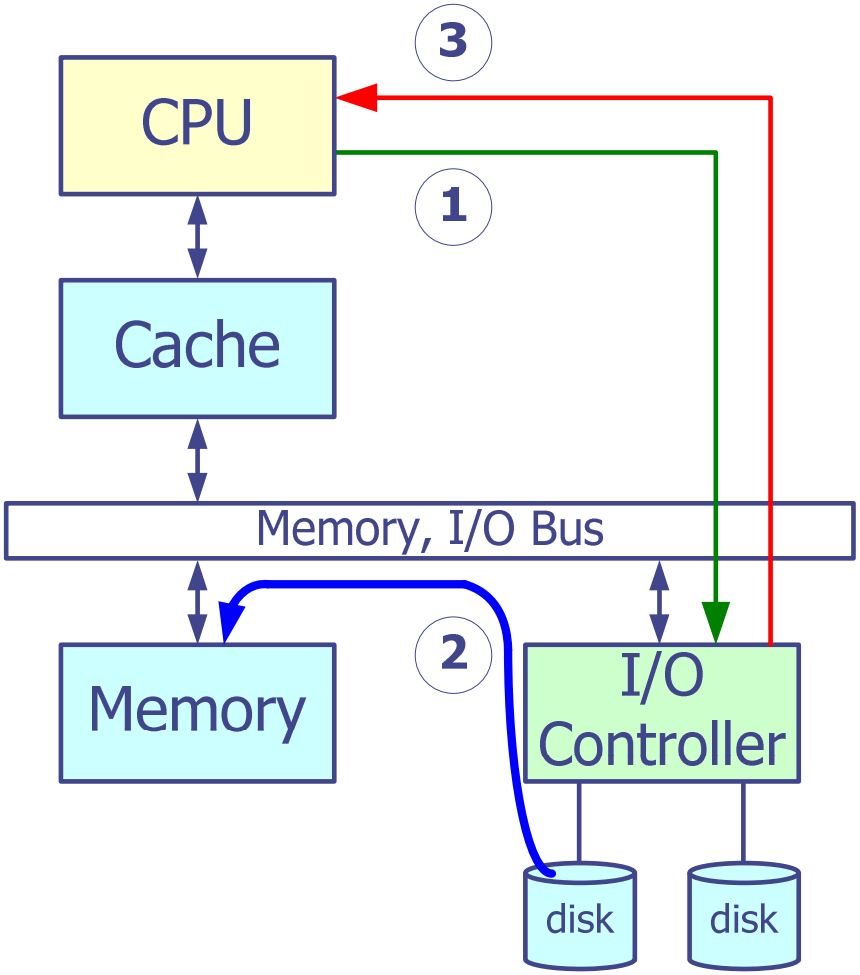
custo de um *Page Fault* é muito elevado

"Page Fault"

- Se a página que o CPU pretende aceder não está em memória ("Valid bit" da *Page Table* com o valor 0), é gerado um *Page Fault*
- Um *Page Fault* tem um custo elevado em termos temporais
- Os *Page Faults* são tratados por software:
  - O gestor de memória (Memory Management Unit - MMU) gera uma exceção que transfere o controlo para o Sistema Operativo
  - O Sistema Operativo decide onde colocar, na memória, a página em falta e desencadeia a respetiva transferência a partir do disco. Se não houver espaço na memória o SO tem que começar por substituir uma página existente e isso pode envolver a cópia dessa página para o disco
  - O processamento completo de um *Page Fault* pode demorar dezenas de milhões de ciclos de relógio
  - O Sistema Operativo suspende o processo corrente e lança outro
  - O processo suspenso é retomado quando o *Page Fault* for resolvido, numa decisão também da responsabilidade do Sistema Operativo

Parte do processamento de um Page Fault

1. SO configura o "I/O controller" para transferir, para memória, a página em falta (bloco):
   - Dimensão do bloco
   - Endereço de início no disco
   - Endereço destino na memória (escolhido pelo SO)
2. A transferência processa-se por DMA
3. O "I/O controller" sinaliza o fim da transferência:
   - Gera uma interrupção
   - O SO retoma a execução do processo suspenso



Política de substituição de páginas na memória

- Quando ocorre um *Page Fault* e a memória física está toda ocupada, é necessário decidir qual a página que tem que sair
- A política normalmente usada é:
  - LRU (Least Recently Used - é substituída a página que está há mais tempo sem ser referenciada)
  - NRU (Not Recently Used) - é uma versão simplificada do LRU
- Implementação do NRU
  - Cada VPN de uma *Page Table* contém um "reference bit"
  - Periodicamente os "reference bits" são colocados a zero pelo Sistema Operativo
  - Quando uma página é acedida (*touched*) o "reference bit" é colocado a 1
  - As páginas candidatas a serem substituídas são as que têm o "reference bit" a 0 (é substituída aleatoriamente uma dessas páginas)

Política de escrita

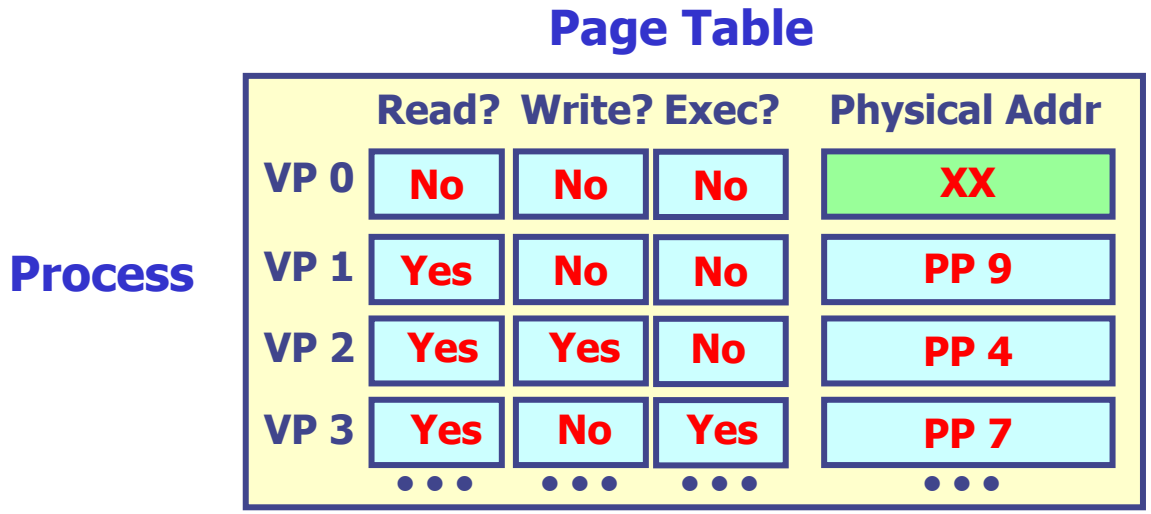
- **Write-back** - a utilização de um esquema do tipo "write-through" seria impraticável uma vez que a escrita de uma página no disco demora um tempo que penalizaria fortemente o desempenho do sistema
  - Escrita em disco página a página
  - Página só é escrita em disco quando necessita de ser retirada da memória física
- **Dirty bit**
  - Se "dirty bit" = 0, a página não necessita de ser escrita em disco aquando da sua substituição (*overwrite*)
  - Se "dirty bit" = 1, a página que vai ser substituída foi alterada. Antes de ser substituída, essa página é copiada para o disco; após essa operação, a nova página é então copiada do disco para a zona da memória que ficou livre
- **Write-allocate**
  - Escrita numa página que não reside em memória física: carrega essa página para a memória e escreve

Implementação de mecanismos de proteção

- Um processo não pode interferir de maneira nenhuma com o funcionamento de outro (seja de forma involuntária ou intencional). Isso é garantido porque:
  - Os espaços de endereçamento de cada processo são independentes
  - Cada processo tem a sua própria *Page Table*, gerida exclusivamente pelo Sistema Operativo
- Dentro do mesmo processo:
  - Diferentes zonas do espaço de endereçamento podem ter diferentes permissões de acesso. Por exemplo, uma zona de instruções não tem permissão de escrita, uma zona de dados não tem permissão de execução, etc.
- Os atributos de acesso a cada página (Read, Read/Write, Execute) são verificados em cada tradução de endereços (para cada VPN)

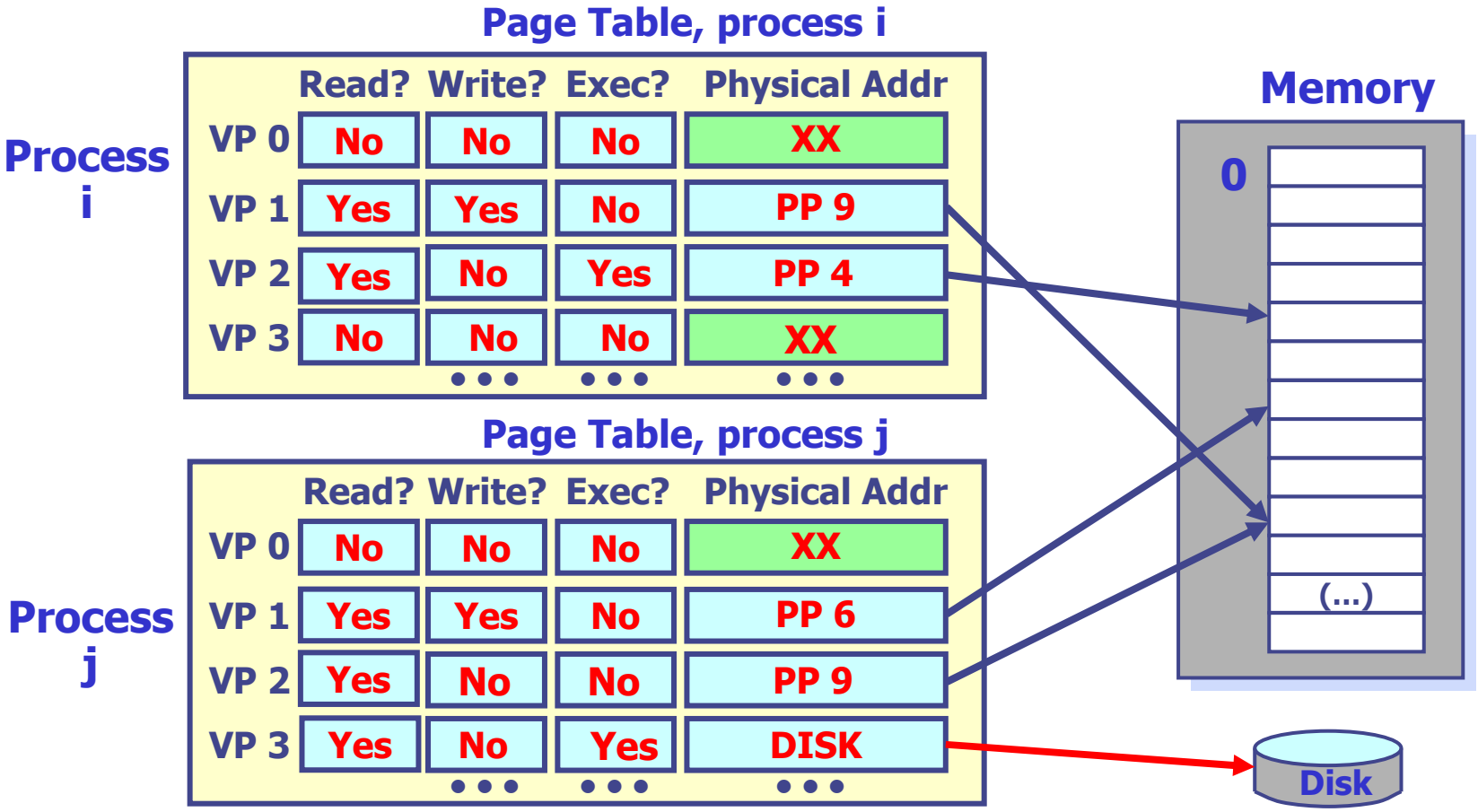
Implementação de mecanismos de proteção

- Cada entrada da *Page Table* contém informação relativa a atributos de acesso do processo respetivo:
  - Read
  - Read / Write
  - Execute



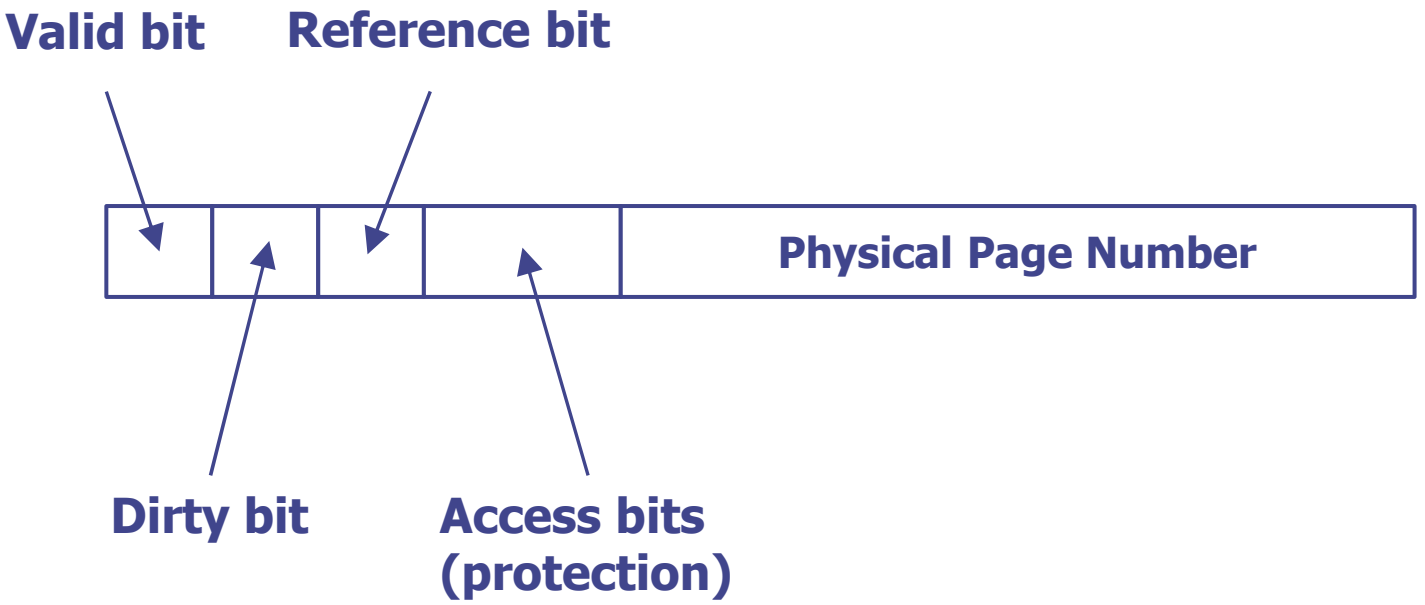
Implementação de mecanismos de proteção

- O hardware gera uma exceção se ocorrer uma tentativa de violação dos atributos de acesso (e.g. no SO Windows: "General protection fault", no SO Linux: "Segmentation fault")

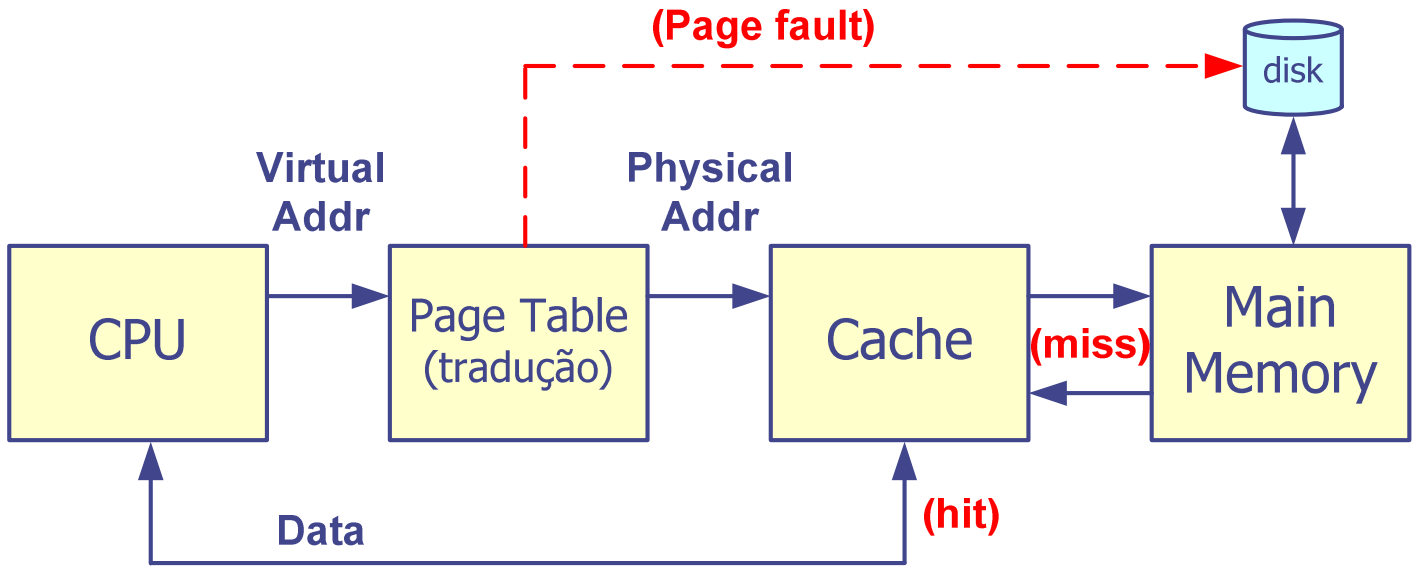


Estrutura de uma entrada da Page Table

- A estrutura exata de uma entrada da Page Table depende do hardware, mas a informação presente é semelhante de máquina para máquina
- A dimensão também depende do hardware da máquina (uma dimensão comum é 32 bits – alguns bits poderão não ser usados)



Memória virtual + cache



- A *Page Table* reside na memória principal. Isso significa que cada acesso à memória virtual implica dois acessos à memória física:
  - 1 acesso para indexar a *Page Table* e obter o endereço físico
  - 1 acesso para ler/escrever os dados
- Será esta uma solução viável? Como resolver este problema?

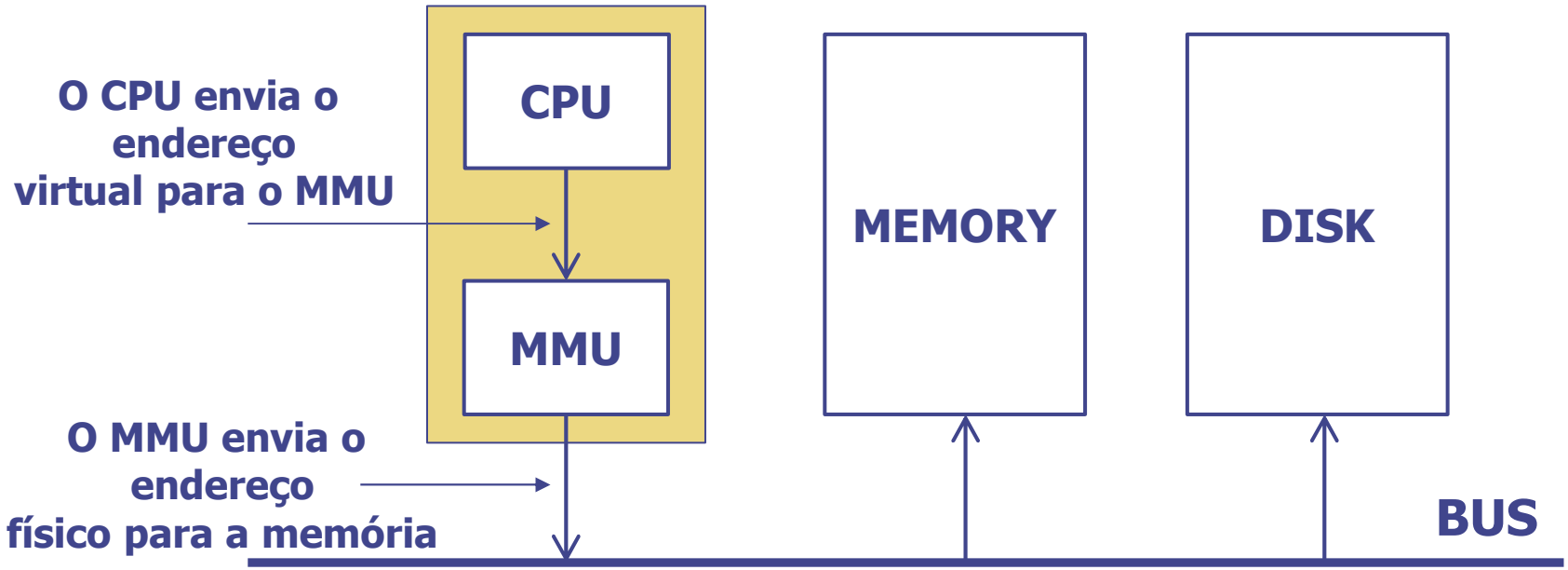
Translation-lookaside buffer (TLB)

- Os acessos à memória virtual exibem localidade espacial e temporal, o que significa que, frequentemente, dados próximos ou recentemente acedidos são reutilizados
- Uma forma de resolver o problema dos acessos lentos à *Page Table* é armazenar uma parte dela numa memória rápida, semelhante a uma cache, contendo as entradas mais recentemente utilizadas
- Esta memória é conhecida como "Translation Lookaside Buffer" (TLB), e é uma memória *cache* dedicada à tradução de endereços virtuais para endereços físicos
- O TLB é tipicamente implementado como uma memória associativa de busca paralela, permitindo acessos rápidos para verificar se uma entrada da *Page Table* está presente
- Páginas que estão residentes em disco (ou seja, não carregadas na memória física) não são referenciadas no TLB

Translation-lookaside buffer (TLB)

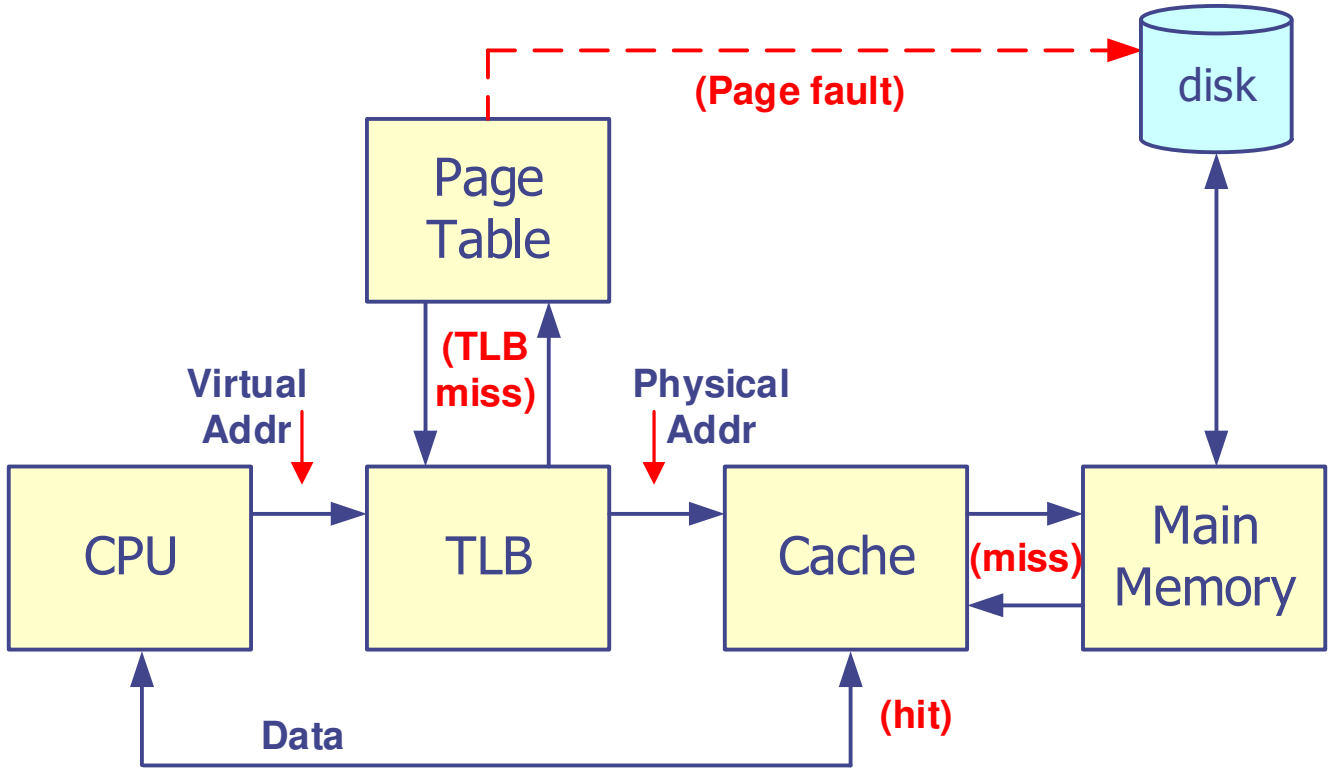
- Em cada acesso, no processo de tradução, verifica-se se o número da página virtual (VPN) está presente no TLB
- Se VPN está no TLB: "**Hit**" – o endereço da página física é obtido de imediato do TLB
- Se VPN não está no TLB: "**Miss**" – a *Page Table* é verificada e pode originar, ou não, um *Page Fault.*
  - Se o acesso à *Page Table* não originar um *Page Fault*, o endereço físico e os atributos da página são copiados da *Page Table* para o TLB, o que pode envolver a substituição de uma entrada no TLB
  - Se o acesso à *Page Table* originar um *Page Fault*, é necessário realizar todo o processamento do *Page Fault* que culmina com a atualização da *Page Table*. De seguida, o endereço físico e os atributos da página são copiados da *Page Table* para o TLB, o que pode envolver a substituição de uma entrada no TLB

Translation-lookaside buffer (TLB)



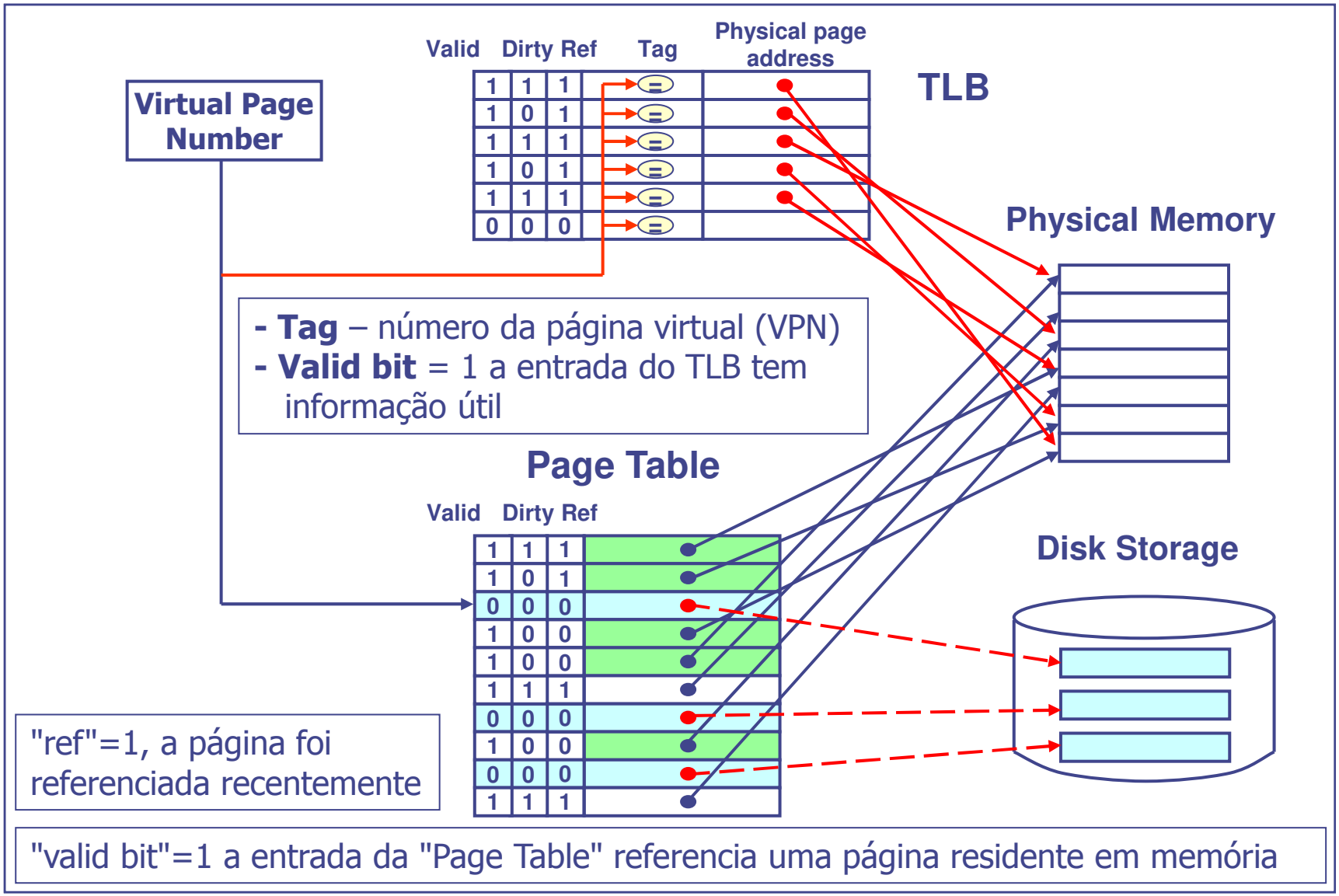
- Atualmente o MMU ("Memory Management Unit") faz parte do circuito integrado onde também está o CPU
- O TLB é implementado no MMU: armazena os mapeamentos recentes entre endereços virtuais e endereços físicos
- A Page Table reside na memória: contém o mapeamento completo entre endereços virtuais e endereços físicos

TLB + Memória virtual + cache

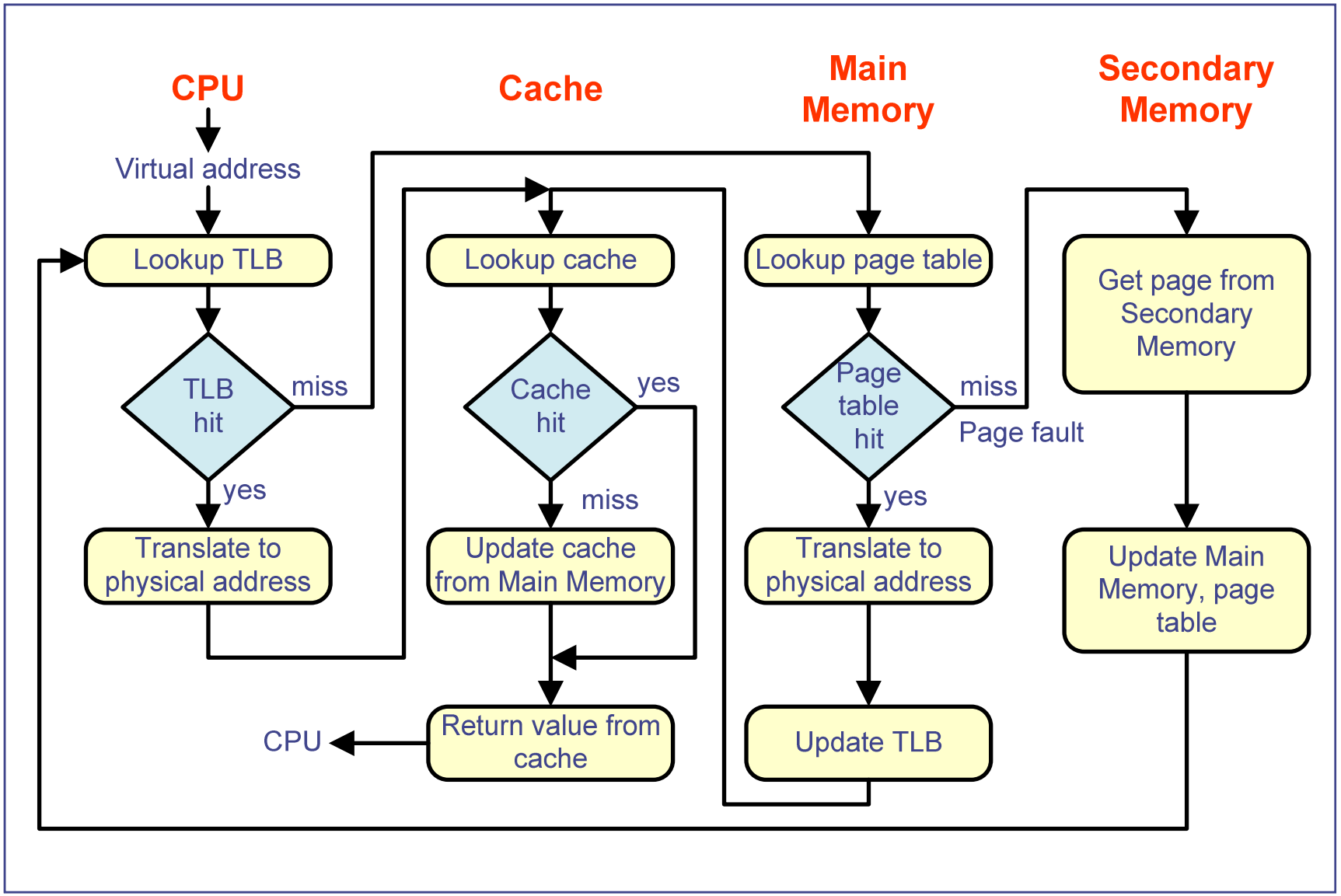


- Se a página virtual está no TLB (hit) a tradução é muito rápida
- Se a página virtual não está no TLB (miss) é necessário consultar a *Page Table*, que reside na memória principal, e atualizar o TLB
- No acesso à *Page Table* pode ocorrer um *Page Fault*

Translation-lookaside buffer (TLB)



Operação da hierarquia de memória

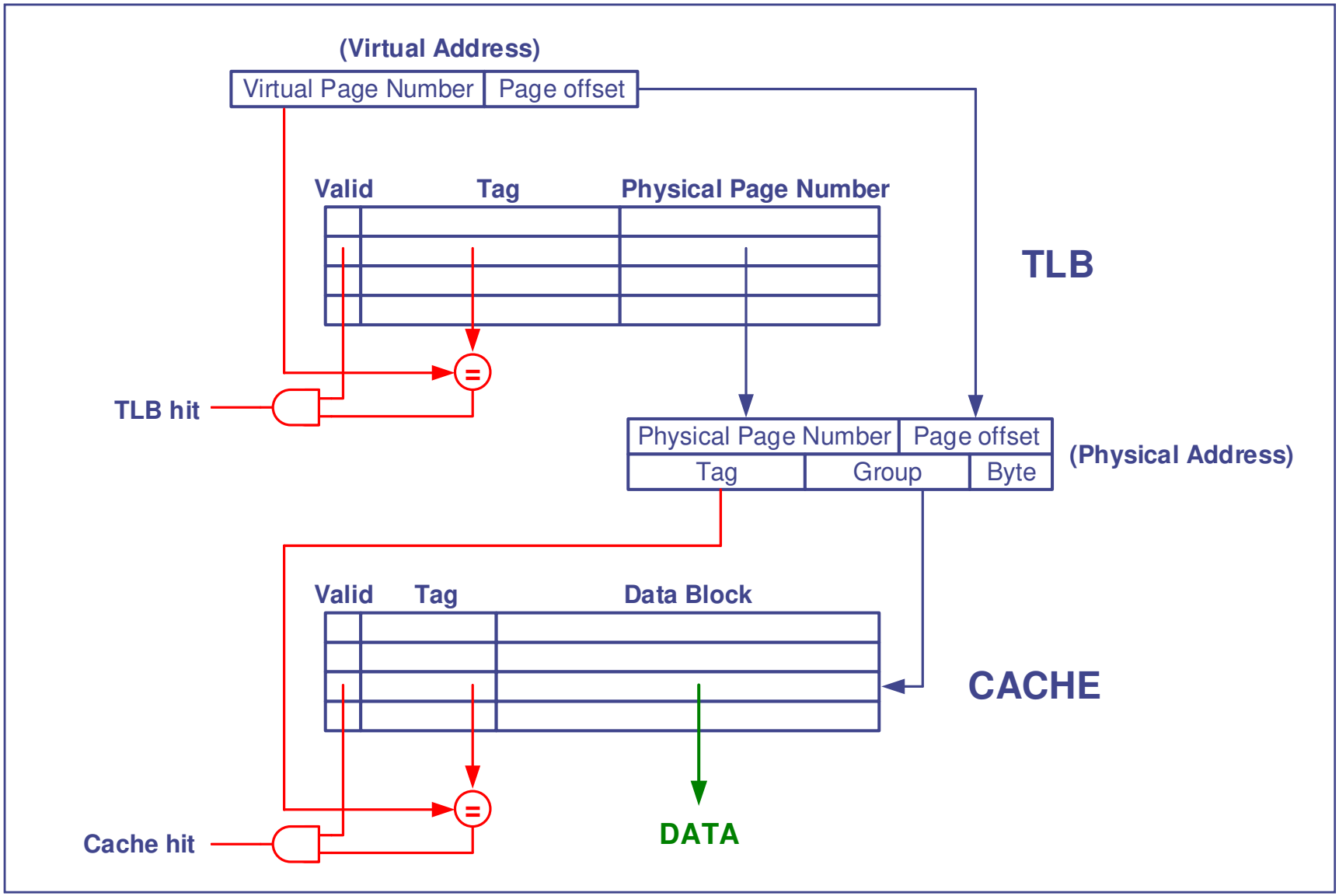


Tradução de endereços com o TLB

- Combinações possíveis dos eventos iniciados por
  - TLB (miss / hit)
  - Page Table (miss / hit, miss ≡ Page Fault)
  - Cache (miss / hit)

TLB	Page Table	Cache	Possível?
miss	miss	miss	Sim! Os dados não estão em memória
miss	miss	hit	Não! A página não está em memória, hit na cache impossível
miss	hit	miss	Sim! A página está memória, dados não disponíveis na cache
miss	hit	hit	Sim! A página está em memória, dados na cache
hit	miss	miss	Não! A página não está referenciada na "Page Table"
hit	miss	hit	Não! A página não está referenciada na "Page Table"
hit	hit	miss	Sim! Página referenciada no TLB, dados não disponíveis na cache
hit	hit	hit	Sim! Página referenciada no TLB, dados disponíveis na cache

Tradução de endereços com o TLB

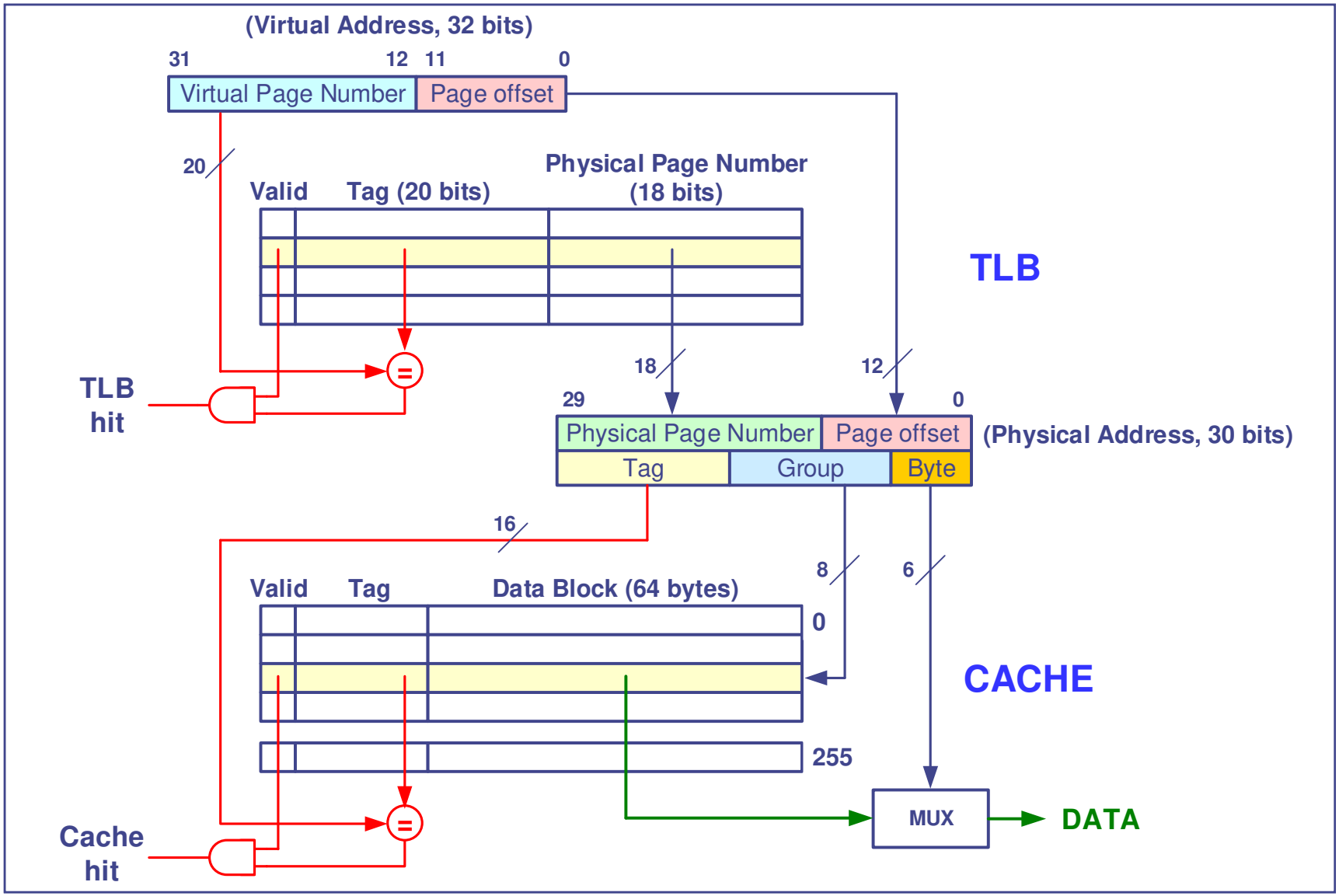


Tradução de endereços com o TLB - Exemplo

S

- Memória virtual:
  - Espaço de endereçamento virtual: 32 bits
  - Espaço de endereçamento físico: 30 bits
  - Dimensão da página: 4 KB (12 bits)
    - VPN – 20 bits
    - PPN – 18 bits
- Cache de 16 kBytes
  - Mapeamento direto
  - Nº de linhas: 256 (8 bits)
  - Dimensão do bloco 64 bytes (6 bits)

Tradução de endereços com o TLB - Exemplo



Em resumo...

- A memória virtual introduz uma indireção entre o endereço virtual gerado pelo CPU e o endereço físico da memória
- Isto permite
  - Mapear memória em disco (memória "ilimitada")
  - Gerir de forma eficiente a memória disponível e evitar a fragmentação
  - Impedir que um processo aceda à memória de outro processo ou escreva em zonas de memória atribuídas ao processo, mas onde não tem permissão de escrita (segurança)
- Cada acesso à memória envolve a tradução do endereço virtual para um endereço físico
  - A memória é organizada em páginas
  - Cada processo tem uma *Page Table*, localizada em memória, que contém o mapeamento entre páginas virtuais e páginas físicas, bem como os respetivos atributos de acesso
  - A tradução de endereços é acelerada através do TLB que contém um subconjunto das entradas da *Page Table*

Exercício

S

- Complete a seguinte tabela, preenchendo as quadrículas em falta e substituindo o ? pelo valor adequado. Utilize as seguintes unidades: K = 2^{10} (Kilo), M = 2^{20} (Mega), G = 2^{30} (Giga), T = 2^{40} (Tera), P = 2^{50} (Peta) ou E = 2^{60} (Exa).

Virtual address size (n)	# Virtual addresses (N)	Maior endereço virtual (hexadecimal)
8	$2^8 = 256$	0xFF
	$2^? = 64$ K	
		0xFFFFFFFF
	$2^? = 256$ T	
64		

Exercício

S

- Determine o número de entradas da *Page Table* (PTE) para as seguintes combinações de número de bits do espaço de endereçamento virtual (n) e dimensão da página (P):

n	P	# PTEs
16	4 KB	
16	8 KB	
32	4 KB	
32	8 KB	
48	4 KB	

Exercício

S

- Suponha um espaço de endereçamento virtual de 32 bits e um espaço de endereçamento físico de 24 bits:
  - determine o número de bits dos campos: VPN (virtual page number), VPO (virtual page offset), PPN (physical page number), PPO (physical page offset) para as dimensões de página P:

P	VPN	VPO	PPN	PPO	# virtual pages	# physical pages
1 KB						
2 KB						
4 KB						
8 KB						

  - para cada dimensão de página, determine o número de páginas virtuais e físicas

Exercício

S

- Considere um sistema de memória virtual com um espaço de endereçamento virtual de 26 bits, páginas de 512 bytes, em que cada entrada da "Page Table" está alinhada em endereços múltiplos de 2, tem 16 bits de dimensão, e está organizada do seguinte modo:

  Valid bit, **Dirty** bit, **Read** flag, **Write** flag, **PPN**

  - em quantas páginas está organizado o espaço de endereçamento virtual? Quantas entradas tem a "Page Table"?
  - qual a dimensão do espaço de endereçamento físico?
  - em quantas páginas está organizado o espaço de endereçamento físico?
- Suponha que o "Page Table register" do processo em execução tem o valor 0x1A0 e que no endereço 0x252 (e 0x253) está armazenado o valor 0xA26C
  - quais os atributos da página física referenciada por essa entrada da tabela? Onde reside a página física?
  - qual é o VPN representado nessa entrada da "Page Table"?
  - qual o endereço inicial e final da página física?
  - qual a gama de endereços virtuais que indexa esta entrada da "Page Table"?