

Sistemas Operacionais

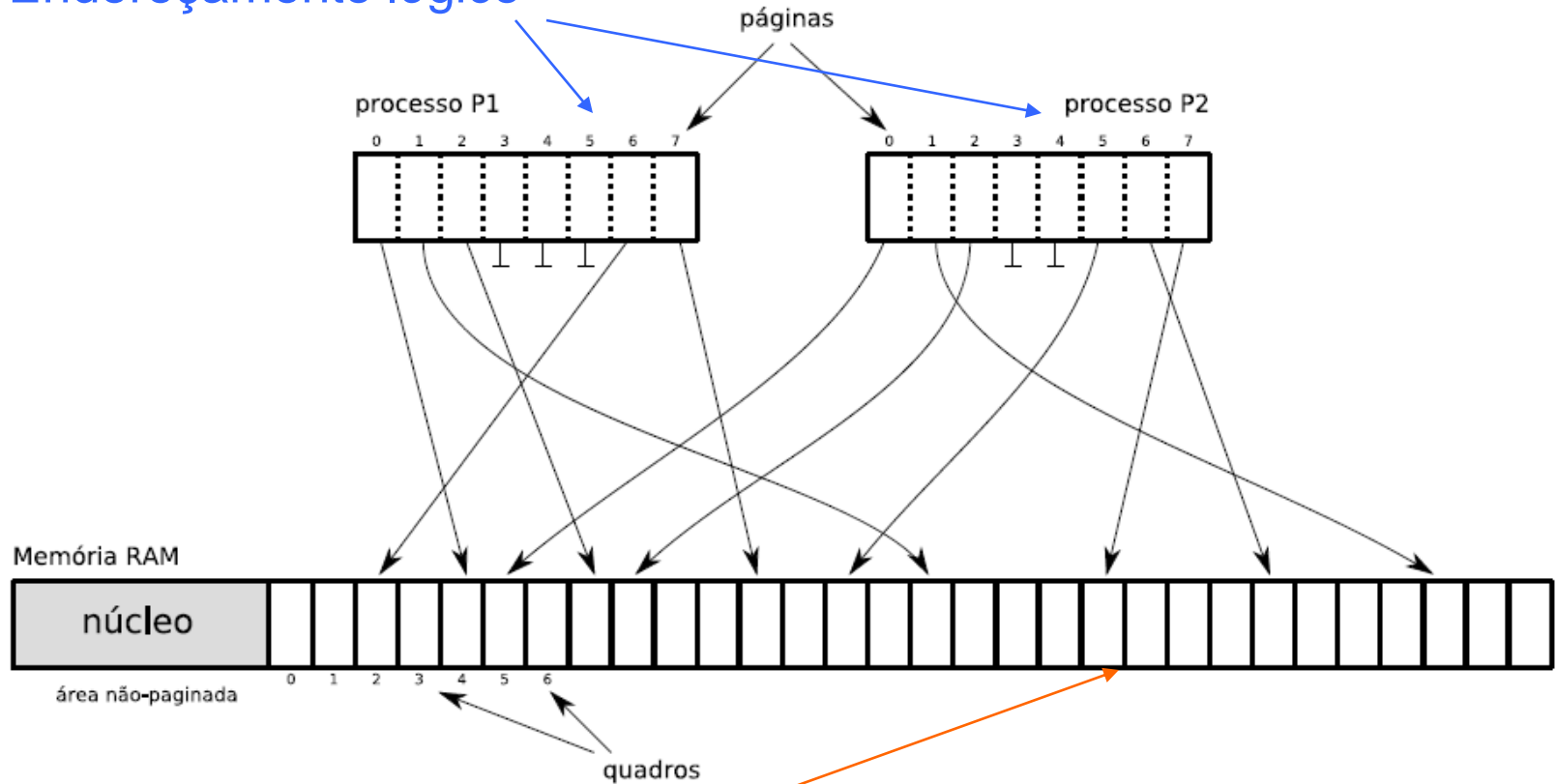
Gerência de Memória

Objetivo

- Alocação paginada
 - Definições
 - Resolução de endereços
 - Tabelas de alocação multi-nível
 - TLB

Alocação Paginada

Endereçamento lógico



Endereçamento físico

Motivação

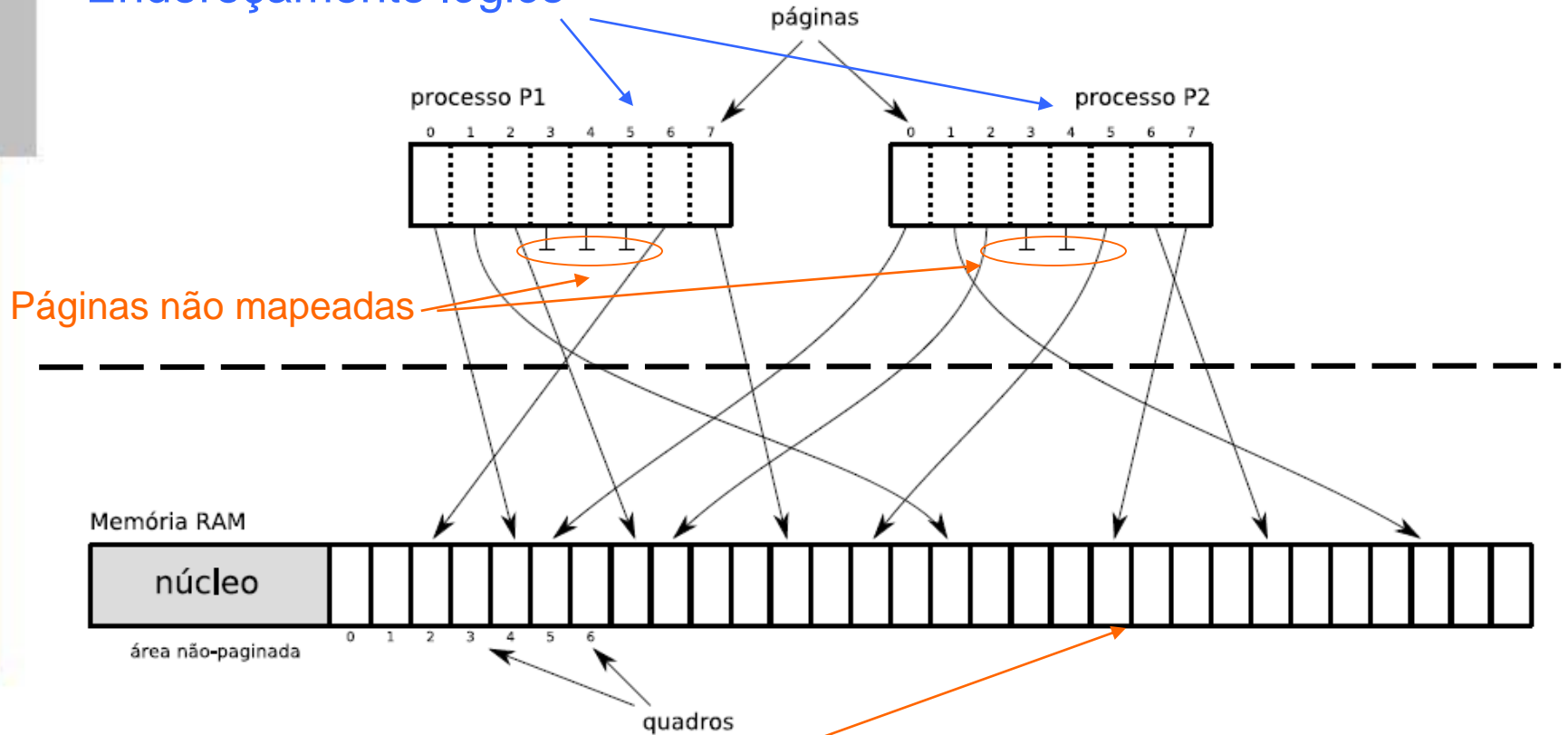
- Páginas de um processo podem estar em qualquer posição da memória física disponível aos processos:
 - grande flexibilidade de alocação

Motivação

- Páginas de um processo podem estar em qualquer posição da memória física disponível aos processos:
 - grande flexibilidade de alocação
- Páginas não usadas pelo processo não precisam estar mapeadas:
 - eficiência no uso da memória física

Alocação Paginada

Endereçamento lógico



Endereçamento físico

Tamanho da página?

Nas arquiteturas atuais, as páginas geralmente têm **4 kbytes** (4.096 bytes), mas podem ser encontradas arquiteturas com páginas de outros tamanhos.

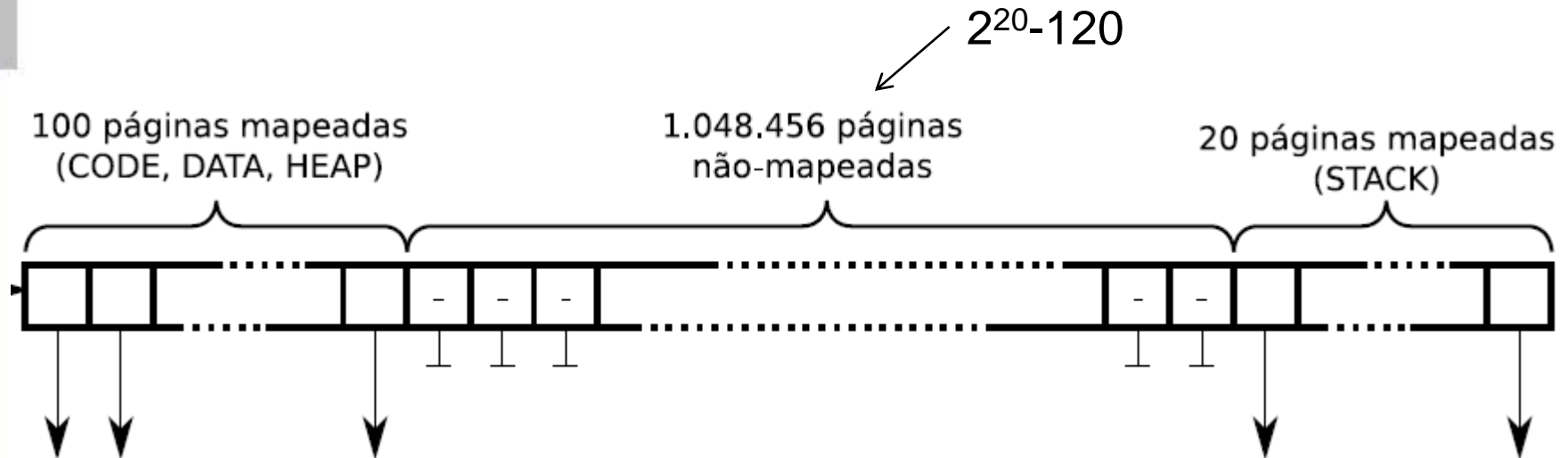


Tamanho da página?

- Arquiteturas mais recentes suportam diversos tamanhos de páginas, inclusive páginas **muito grandes**
 - super-páginas (*hugepages, superpages ou largepages*)
 - entre 1 e 16 MBytes, ou mesmo acima disso
- Em conjunto com as páginas normais permite obter mais **desempenho** no acesso à memória
- Torna os mecanismos de gerência de memória bem mais **complexos**

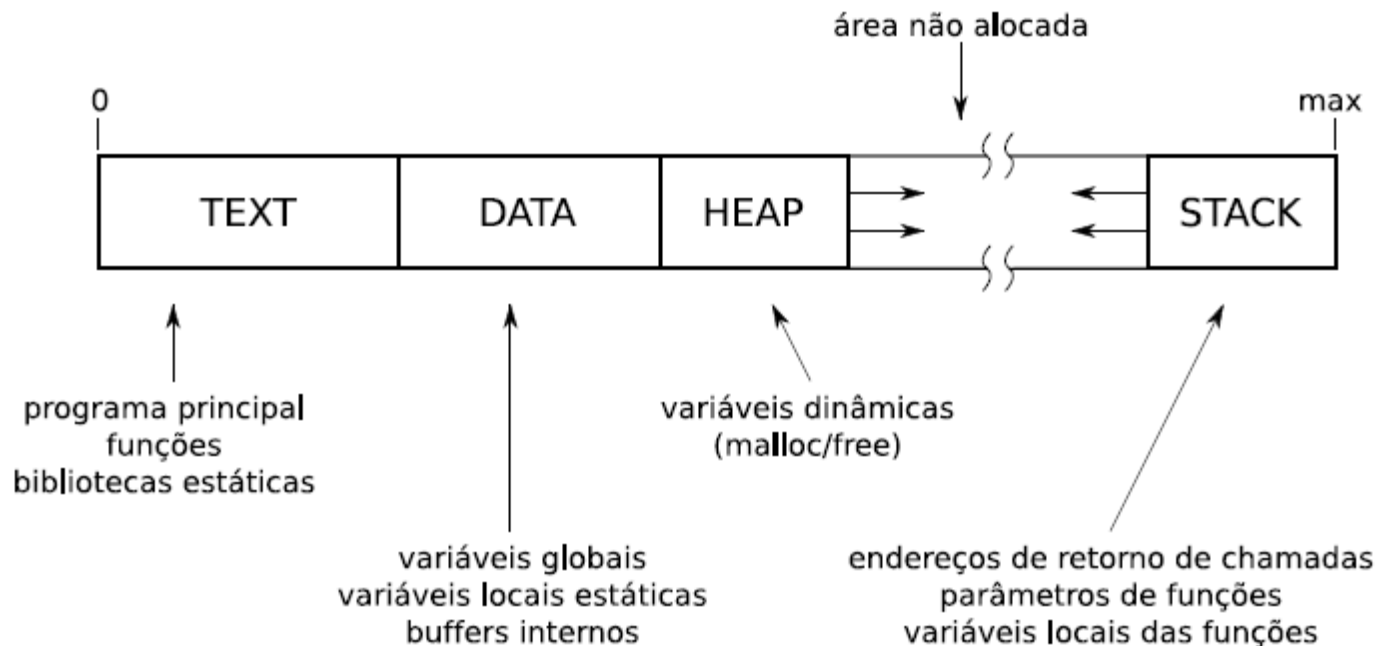
Espaço de endereçamento

Ex.: Processo de 480 kB, páginas de 4 kB e sistema de 32 bits



Modelo de memória dos processos

Cada processo é visto pelo **sistema operacional** como uma **área de memória** exclusiva que só ele e o núcleo do sistema podem acessar.



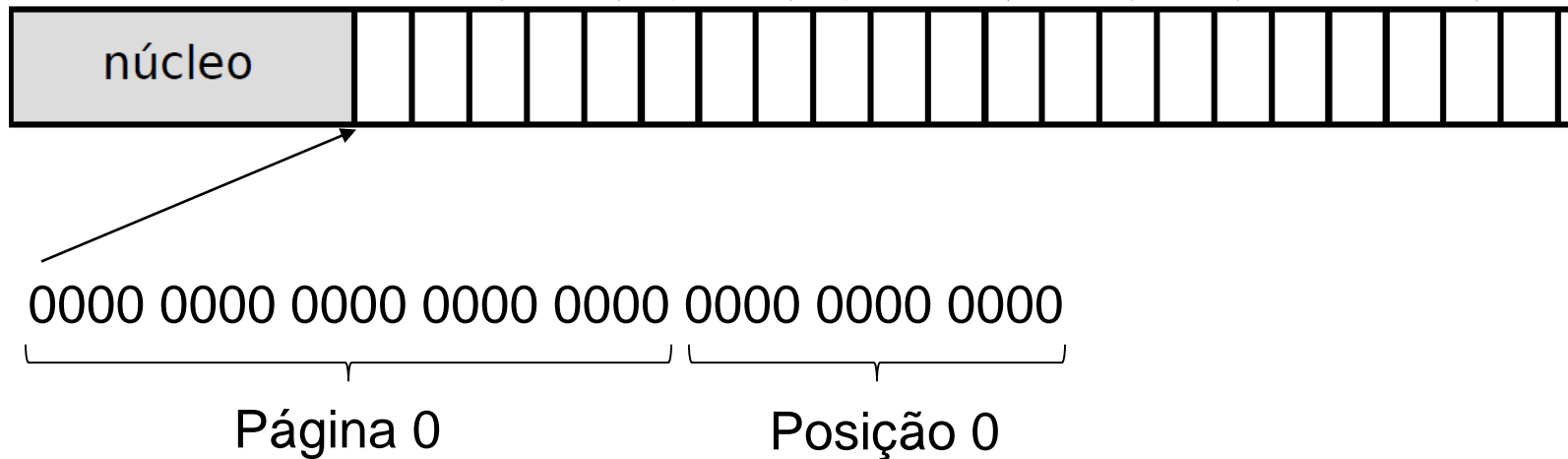
Espaço de endereçamento

Ex.: Pentium – Endereços lógicos de 32 bits e páginas de 4kbytes

01805E9A_H → 0000 0001 1000 0000 0101 1110 1001 1010₂
→ $\underbrace{0000\ 0001\ 1000\ 0000\ 0101}_{{20\ \text{bits}}}_2$ e $\underbrace{1110\ 1001\ 1010}_{{12\ \text{bits}}}_2$
→ $\underbrace{01805_H}_{\text{página}}$ e $\underbrace{E9A_H}_{\text{offset}}$
→ página 01805_H e offset E9A_H

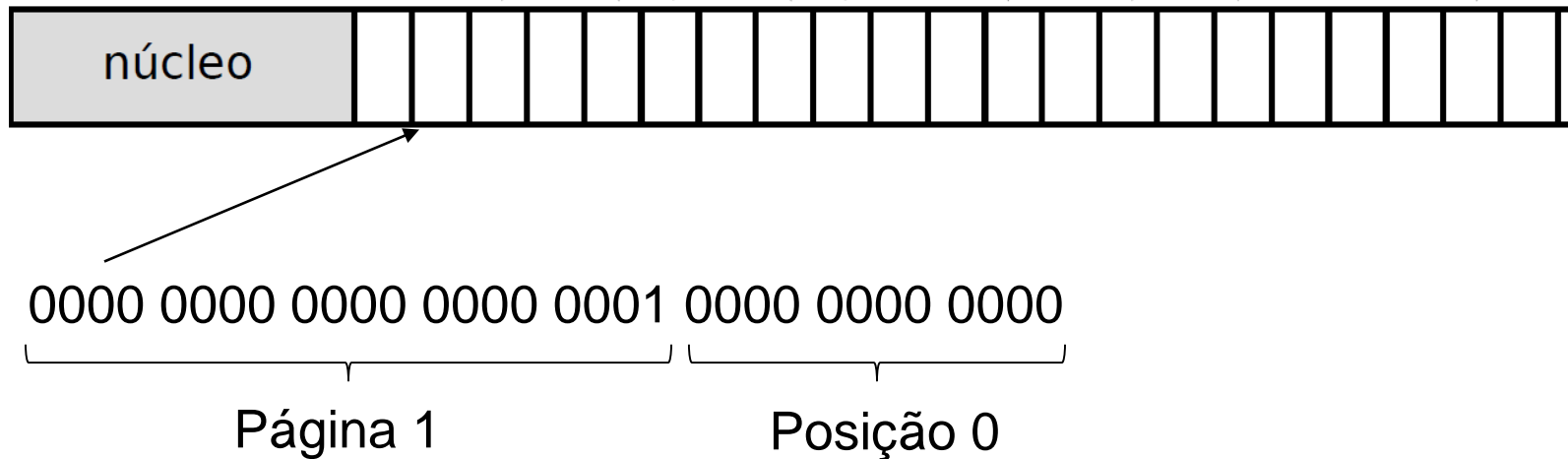
Espaço de endereçamento

Ex.: Pentium – Endereços lógicos de 32 bits e páginas de 4kbytes



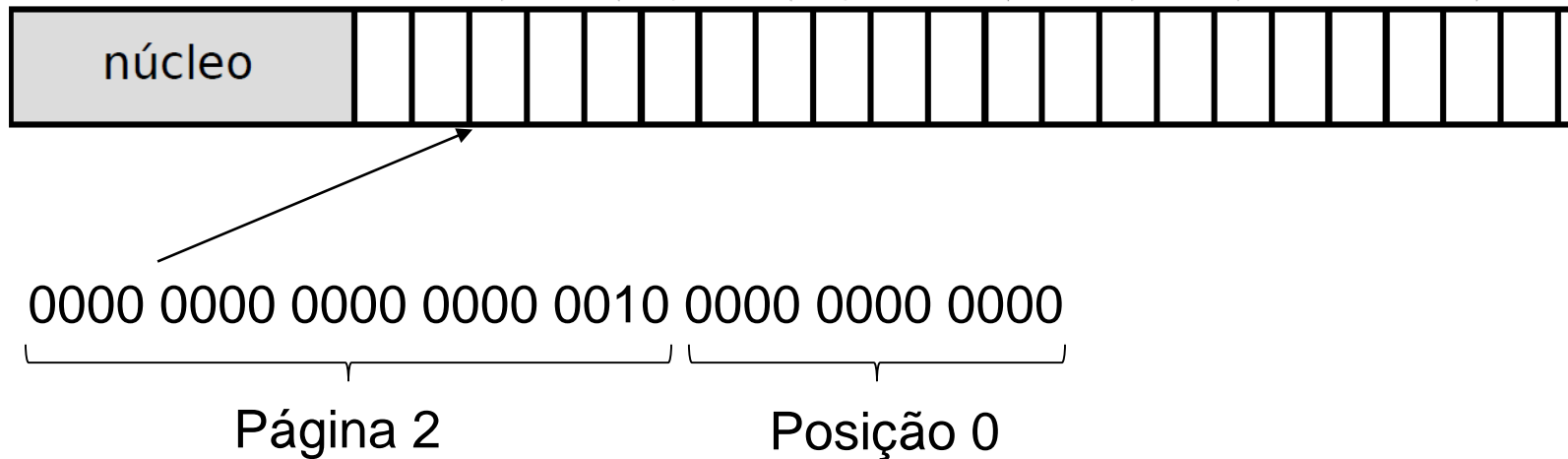
Espaço de endereçamento

Ex.: Pentium – Endereços lógicos de 32 bits e páginas de 4kbytes

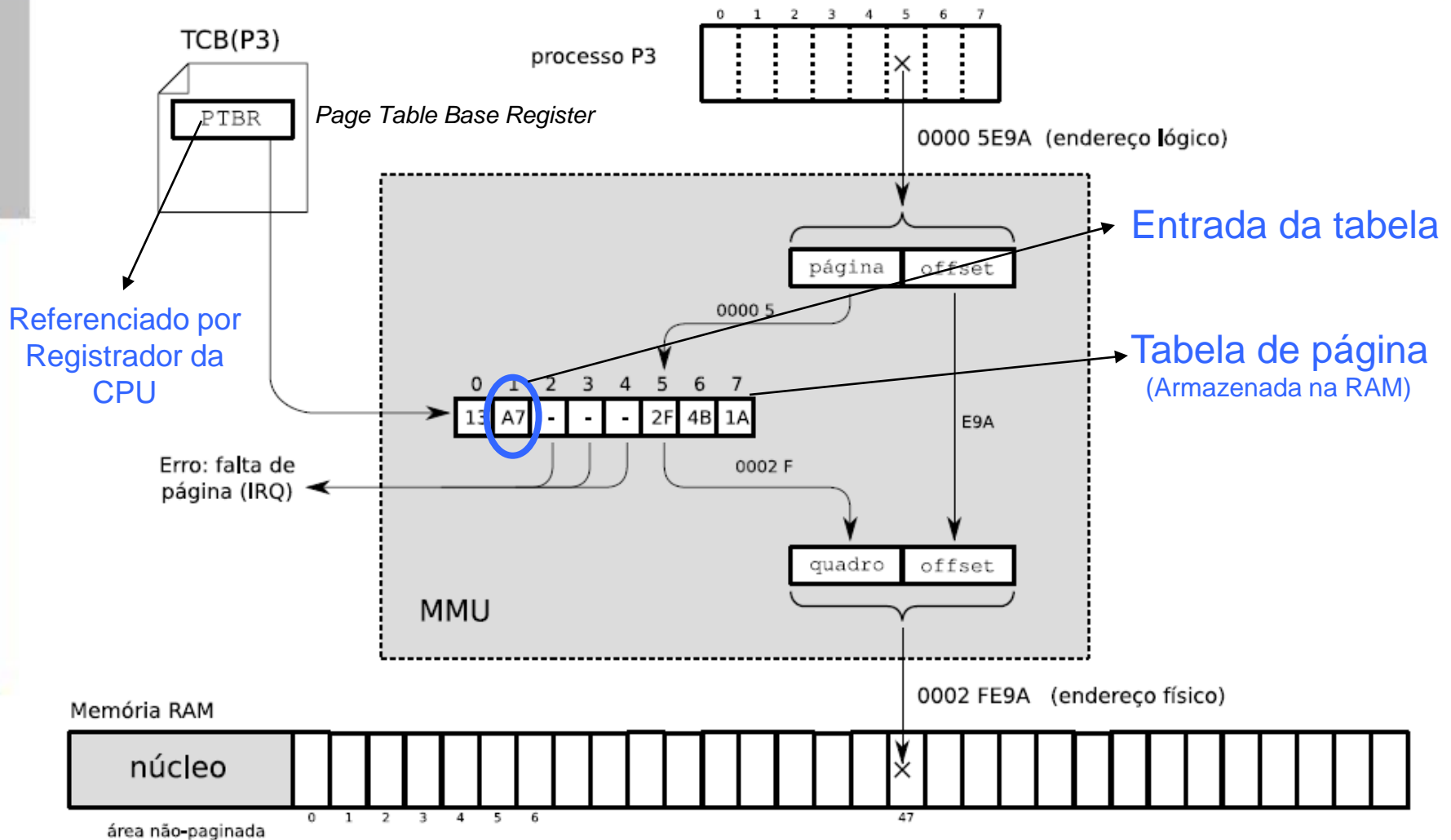


Espaço de endereçamento

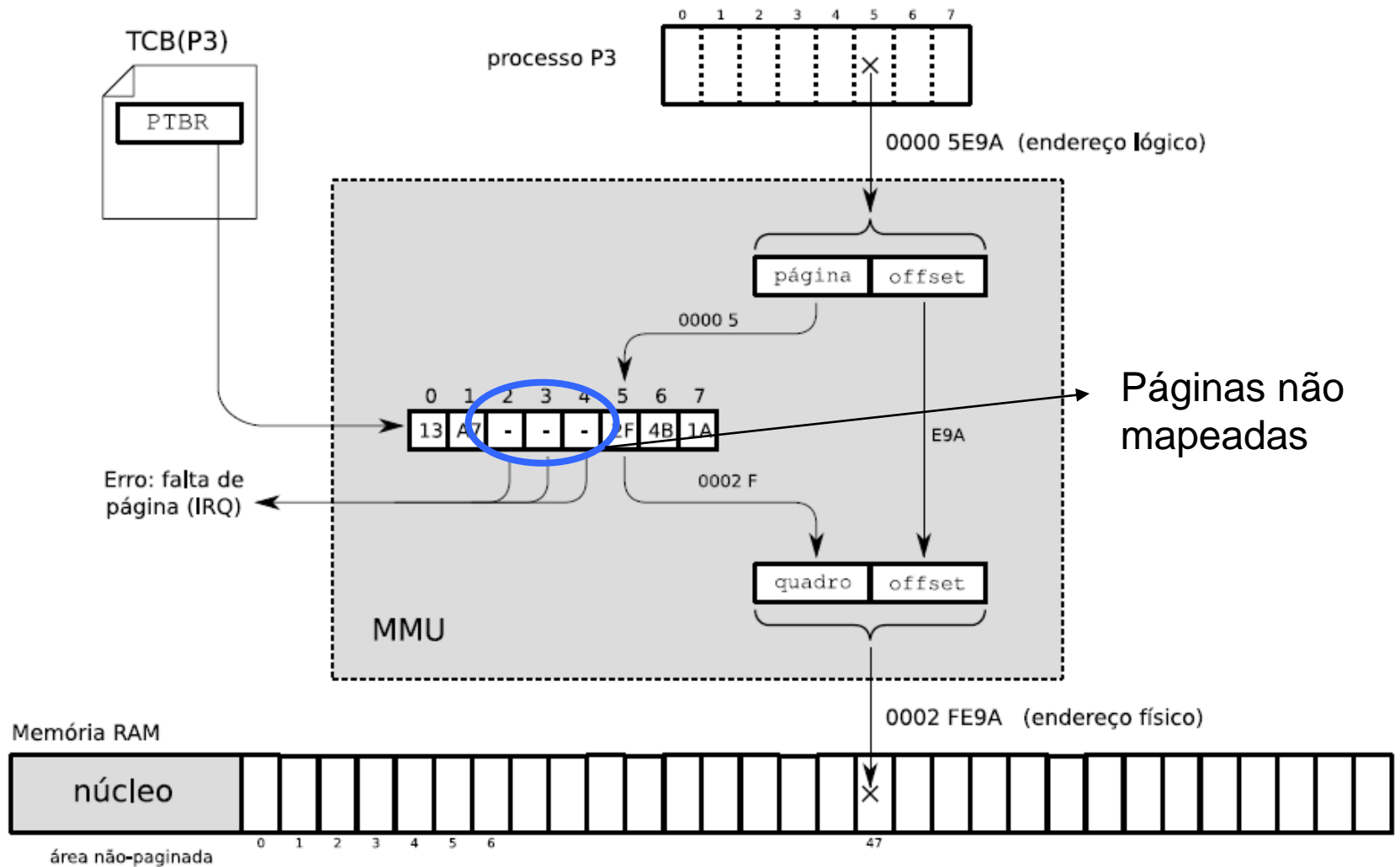
Ex.: Pentium – Endereços lógicos de 32 bits e páginas de 4kbytes



Tradução de endereços



Falta de página



Entradas da tabela

Flags de Controle

Cada entrada da **tabela de páginas** de um processo contém o **número do quadro** correspondente e um conjunto de **flags ou bits** de controle (não mostrados na figura anterior), com diversas finalidades:

- Presença;
- Proteção;
- Referência;
- Modificação;
- etc.

31														12		11		9		8		7		6		5		4		3		2		1		0	
Page Base Address														Avail		G		P A T		D		A		P C D		P W T		U / S		R / W		P					

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

Present _____

Available for system programmer's use _____

Global Page _____

Page Table Attribute Index _____

Dirty _____

Accessed _____

Cache Disabled _____

Write-Through _____

User/Supervisor _____

Read/Write _____

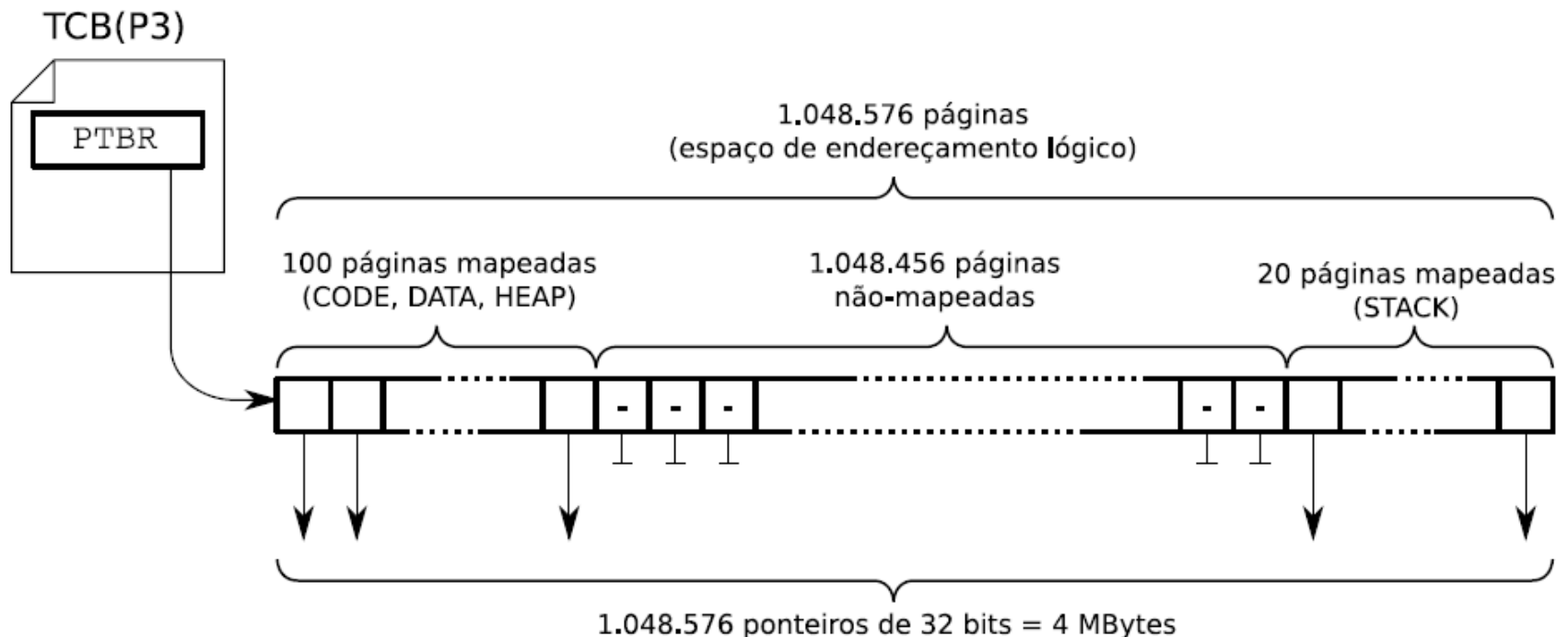
Present _____

A decorative graphic on the left side of the slide. It consists of a vertical light blue bar, a vertical orange bar, and a grey rectangle. A horizontal dark blue bar extends from the grey rectangle across the top of the slide.

Tabelas multi-níveis

O problema

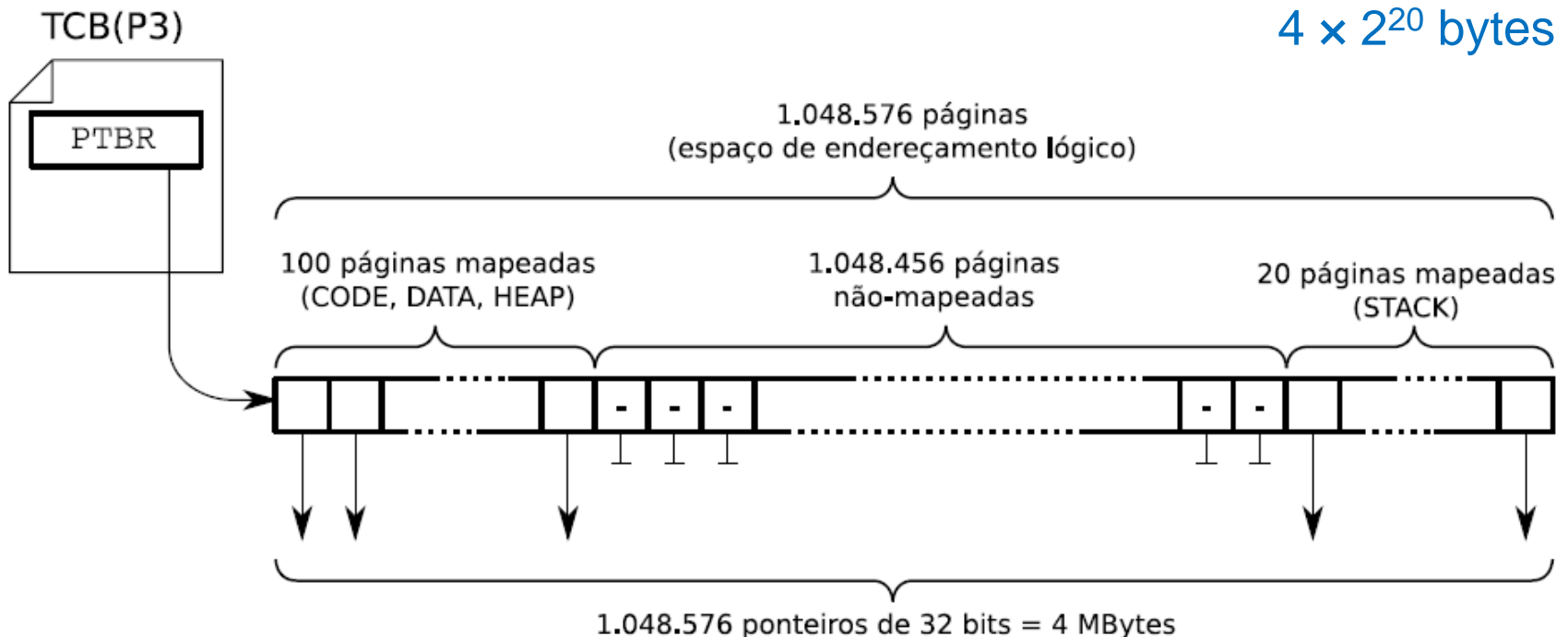
Ex.: Se em uma **arquitetura** de **32 bits** com páginas de **4 kbytes**, cada **entrada** na tabela de páginas ocupa **32 bits = 4 bytes** (**20 bits** para o número do quadro e **12 bits** para *flags*). Como cada tabela de páginas tem **2^{20} entradas**, cada tabela ocupará **4 Mbytes** de memória (4×2^{20} bytes).



O problema

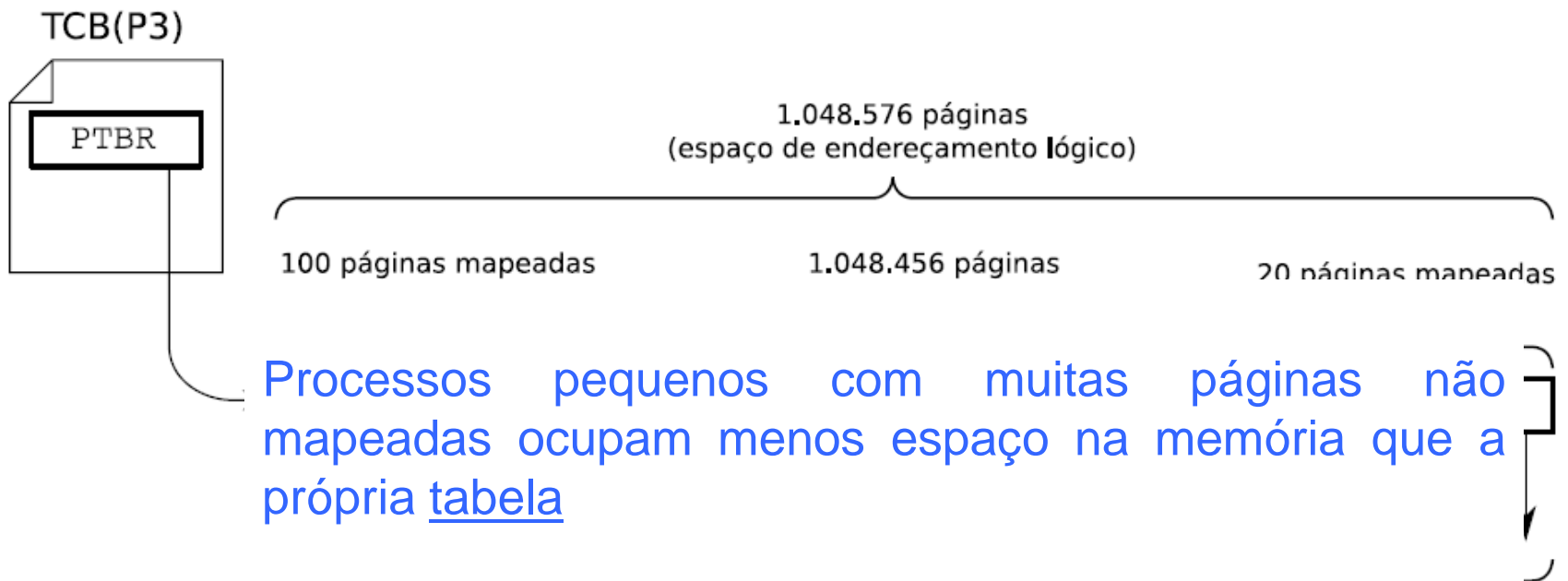
Ex.: Se em uma arquitetura de 32 bits com páginas de 4 kbytes, cada **entrada** na tabela de páginas ocupa 32 bits = 4 bytes (20 bits para o número do quadro e 12 bits para *flags*). Como cada tabela de páginas tem **2²⁰ entradas**, cada tabela ocupará 4 Mbytes de memória (4 × 2²⁰ bytes).

4 × 2²⁰ bytes



O problema

Ex.: Se em uma **arquitetura** de **32 bits** com páginas de **4 kbytes**, cada **entrada** na tabela de páginas ocupa **32 bits = 4 bytes** (**20 bits** para o número do quadro e **12 bits** para *flags*). Como cada tabela de páginas tem **2²⁰ entradas**, cada tabela ocupará **4 Mbytes** de memória (4×2^{20} bytes).



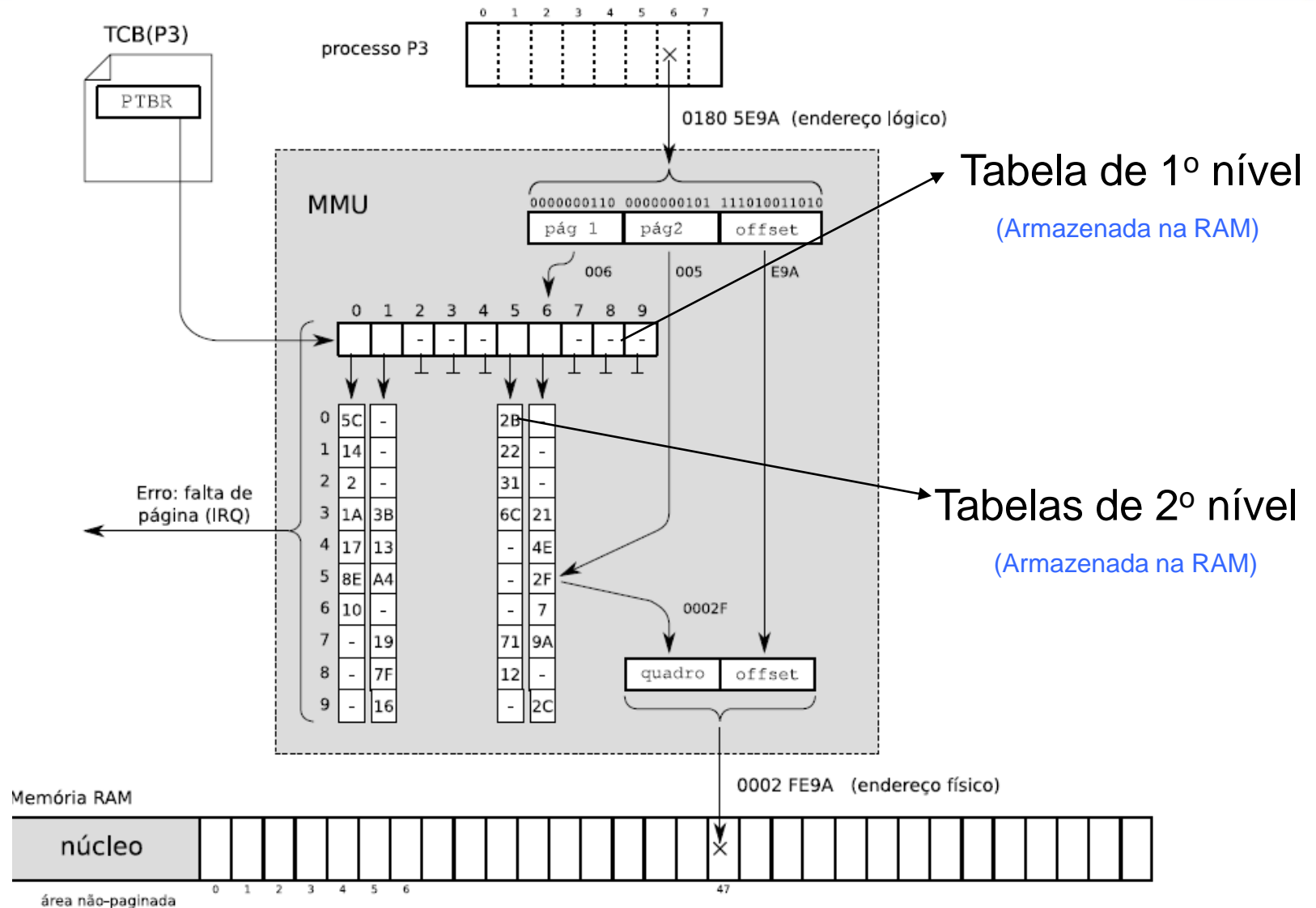
1.048.576 ponteiros de 32 bits = 4 MBytes

A solução

Tabelas multi-níveis:

São estruturadas na forma de árvore: uma *tabela de páginas de primeiro nível* contém ponteiros para *tabelas de páginas de segundo nível*, e assim por diante...

Tabelas multi-níveis



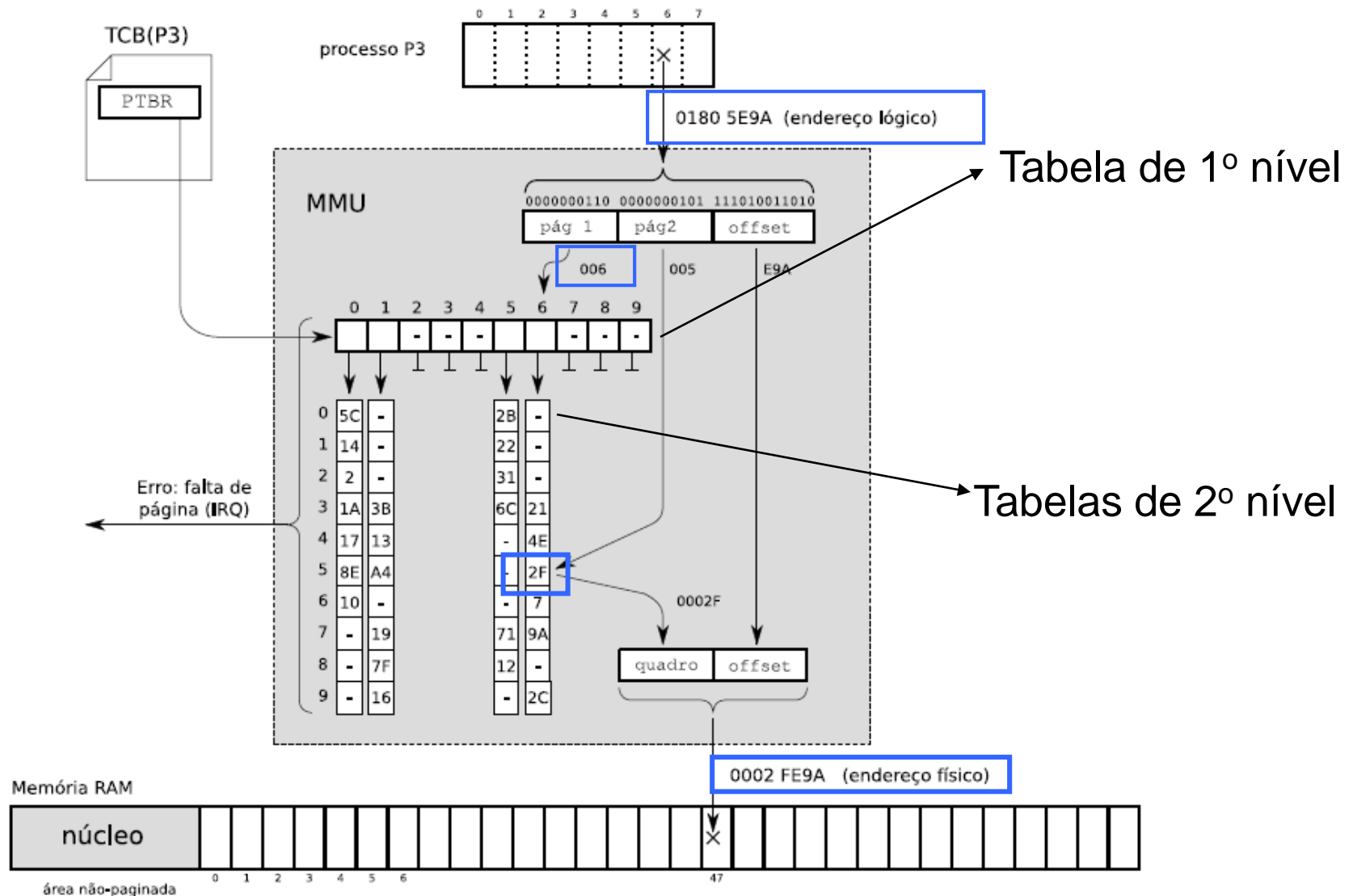
Tabelas multi-níveis

Ex.: Considere uma arquitetura de 32 bits com páginas de 4 kbytes. 20 bits são usados para acessar a tabela de páginas.

Esses 20 bits podem ser **divididos** em dois grupos de 10 bits que são usados como índices em uma tabela de páginas com dois níveis.

01805E9A_H → 0000 0001 1000 0000 0101 1110 1001 1010₂
→ $\underbrace{0000\ 0001\ 10_2}_{10\text{ bits}}\ e\ \underbrace{00\ 0000\ 0101_2}_{10\text{ bits}}\ e\ \underbrace{1110\ 1001\ 1010_2}_{12\text{ bits}}$
→ 0006_H e 0005_H e E9A_H
→ p₁ 0006_H, p₂ 0005_H e offset E9A_H

Tabelas multi-níveis

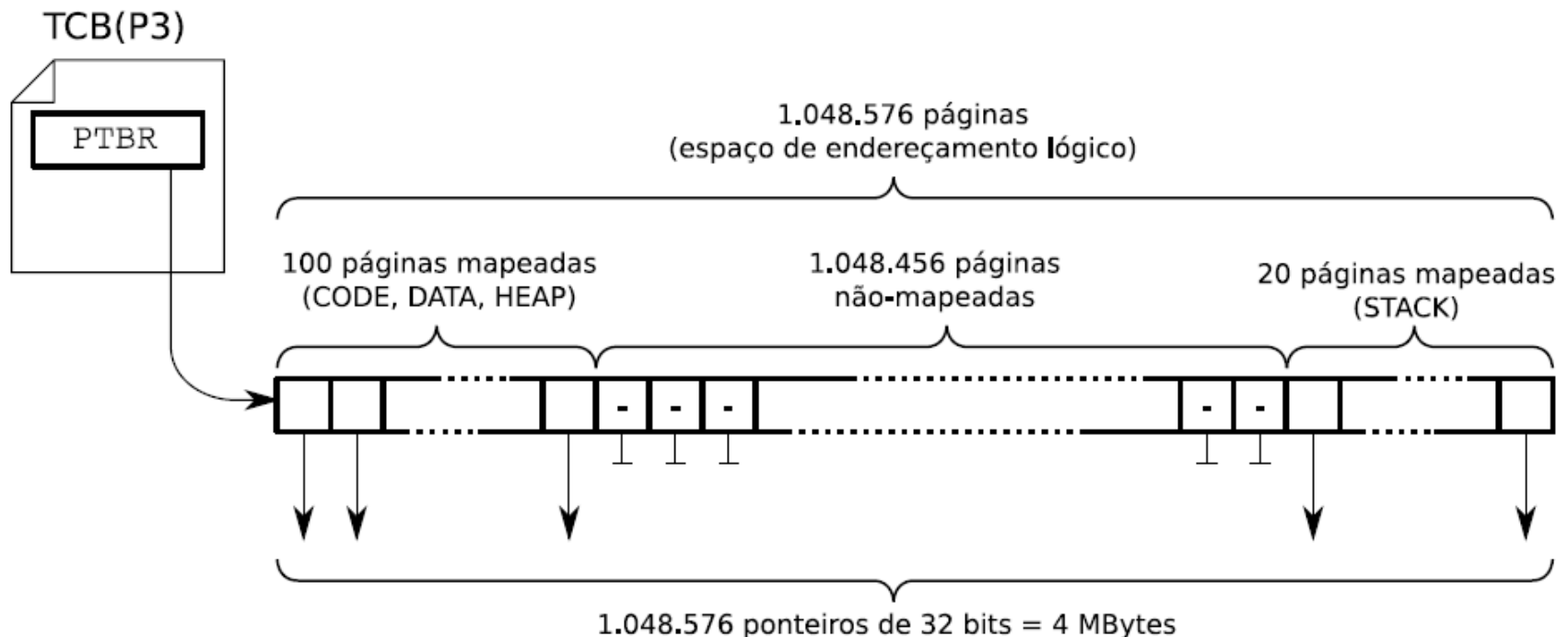


A decorative graphic on the left side of the slide. It consists of a vertical light blue bar, a vertical orange bar, and a grey rectangle. A horizontal dark blue bar extends from the grey rectangle across the top of the slide.

Reduzindo consumo de memória

Relembrando...o problema

Ex.: Se em uma **arquitetura** de **32 bits** com páginas de **4 kbytes**, cada **entrada** na tabela de páginas ocupa **32 bits = 4 bytes** (**20 bits** para o número do quadro e **12 bits** para *flags*). Como cada tabela de páginas tem **2^{20} entradas**, cada tabela ocupará **4 Mbytes** de memória (4×2^{20} bytes).



Reduzindo consumo de memória

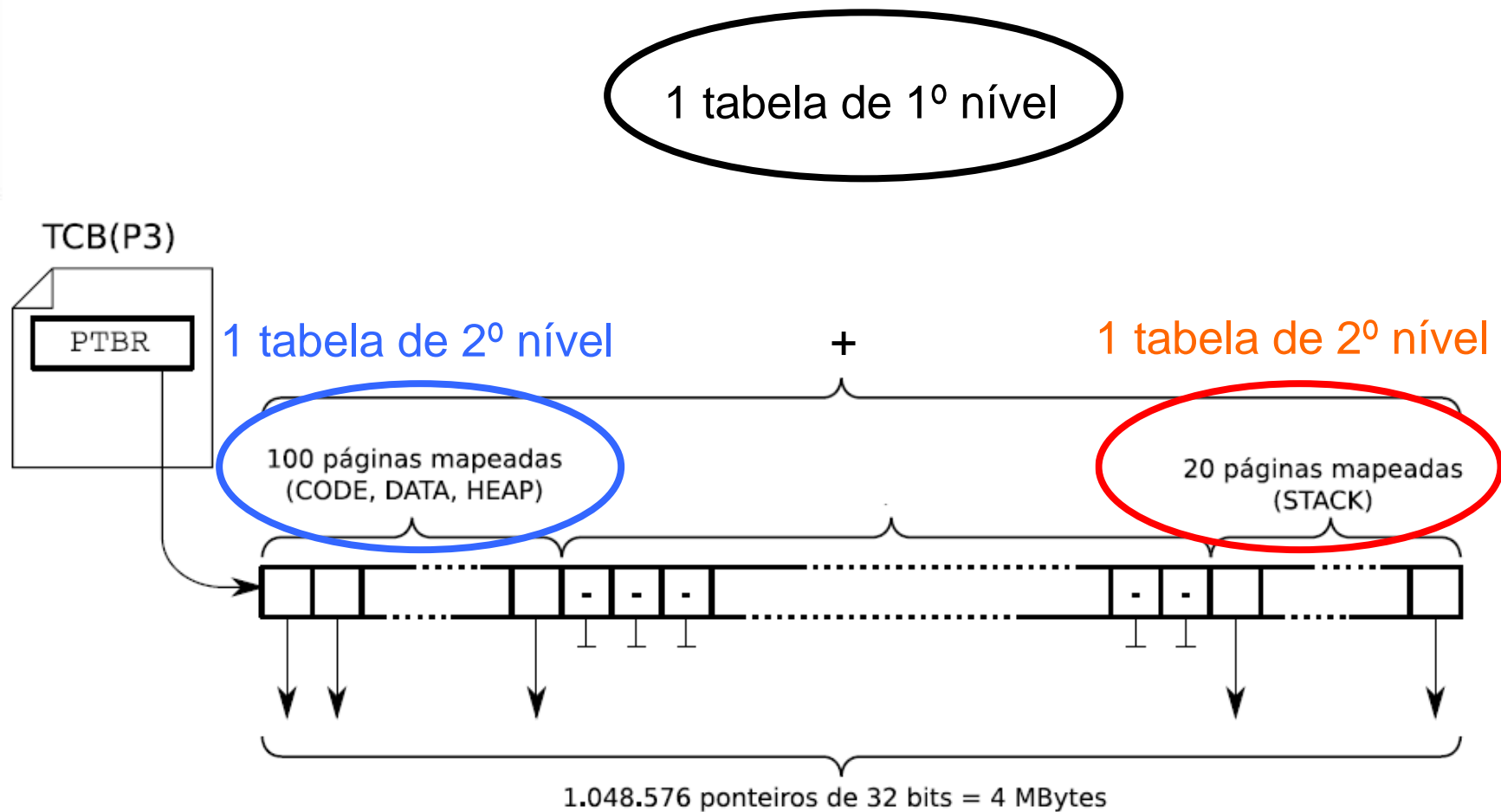
Ex.: Assumindo que cada entrada de tabela ocupa 4 bytes (32 bits), serão necessários somente 12 kB...

Reduzindo consumo de memória

Ex.: Assumindo que cada entrada de tabela ocupa 4 bytes (32 bits), serão necessários somente 12 kB...

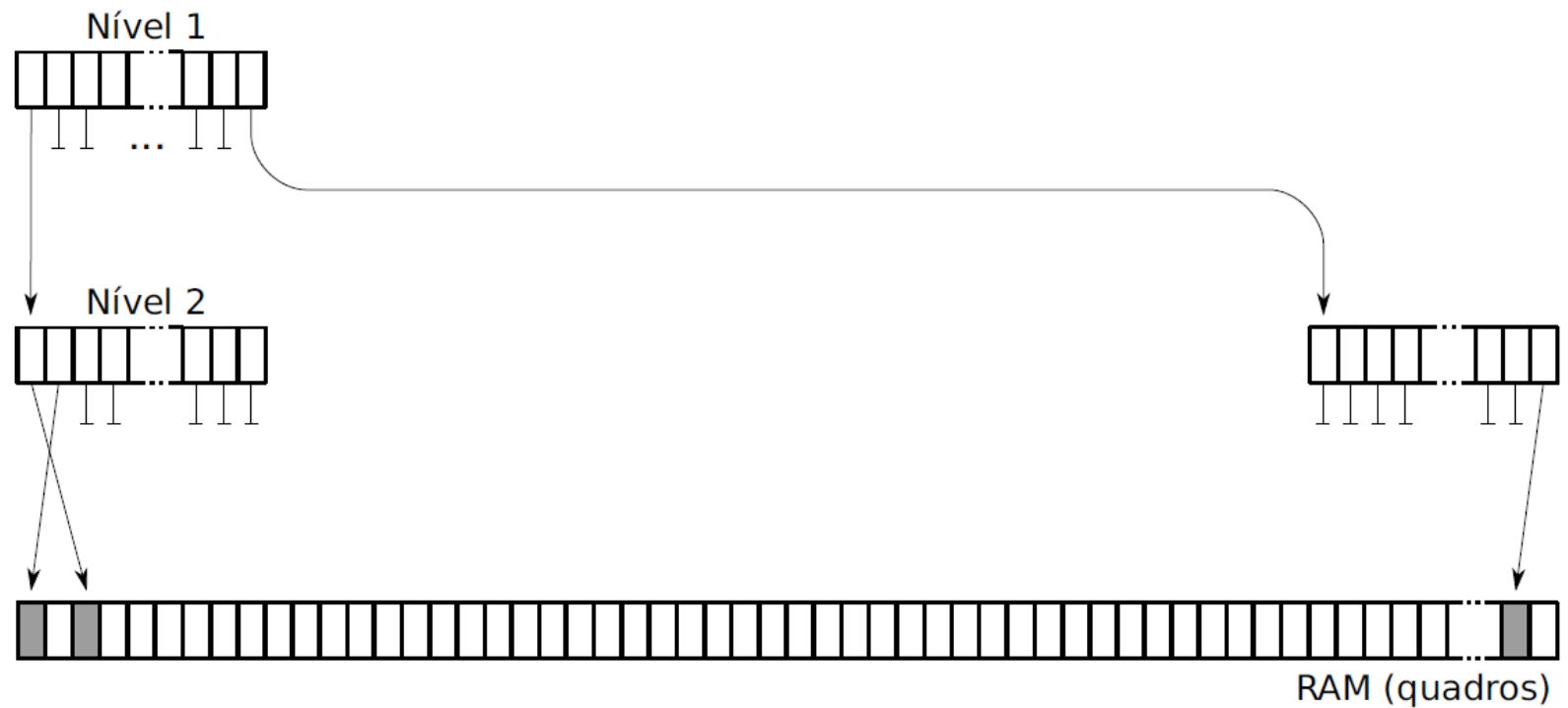
...para armazenar **três** tabelas ($3 \times 4 \times 2^{10}$ bytes) de um processo de 480 kB.

Reduzindo consumo de memória



Reduzindo consumo de memória

Três tabelas:



Reduzindo consumo de memória

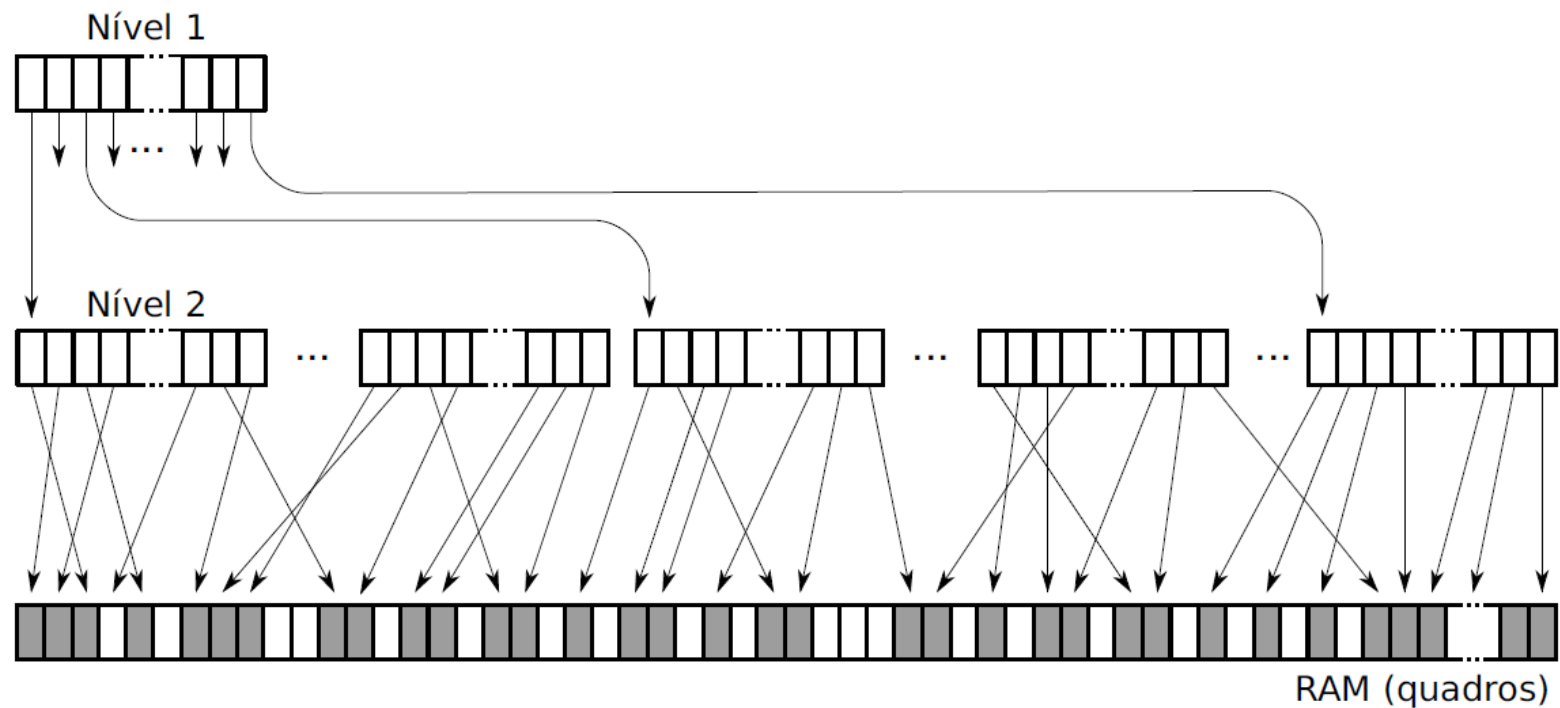
Pior caso

O processo ocupa toda a memória possível:

- 1 tabela de primeiro nível
- 1.024 tabelas de segundo nível
 - Total $4 \times (2^{10} + 2^{10} \times 2^{10})$ bytes = 0,098% a mais que um só nível (4×2^{20} bytes).

Reduzindo consumo de memória

Pior caso



Tabelas multi-níveis

Questão:

Qual o mínimo espaço ocupado na memória em um sistema com 2 níveis de tabela? Considerando:

- Sistema com 32 bits e páginas de 4kB
 - Tabelas com 2^{10} entradas de 4 Bytes
 - 10 bits para cada nível
-
- a) 4 MB + 4kB
 - b) 4 MB
 - c) 4kB + 4 kB
 - d) 4 kB

Tabelas multi-níveis

Questão:

Qual o mínimo espaço ocupado na memória em um sistema com 2 níveis de tabela? Considerando:

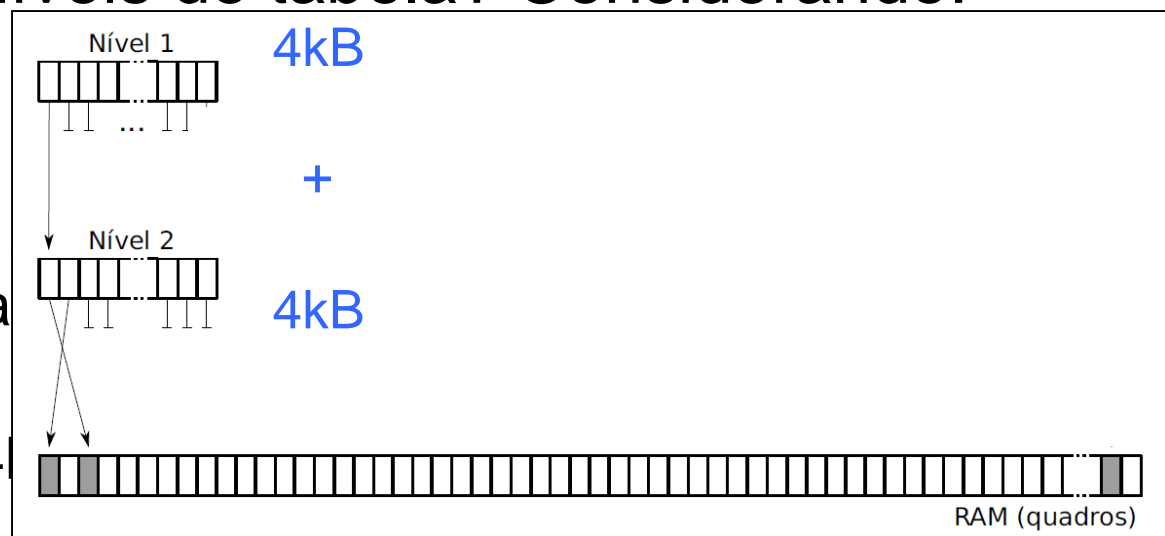
- Sistema com
- Tabelas com
 - 10 bits pa

a) 4 MB + 4

b) 4 MB

c) 4kB + 4 kB

d) 4 kB



Tabelas multi-níveis

Na prática:

- Intel 80.386
 - Tabelas com dois níveis
- Sun Sparc e DEC Alpha
 - Tabelas com 3 níveis
- Intel Itanium
 - Tabelas com 3 ou 4 níveis

Efeito colateral

A estruturação das tabelas de páginas em **vários níveis** resolve o problema do espaço ocupado pelas tabelas de forma muito eficiente...

...mas, tem um efeito colateral muito nocivo:

aumenta drasticamente o tempo de acesso à memória.

Efeito colateral

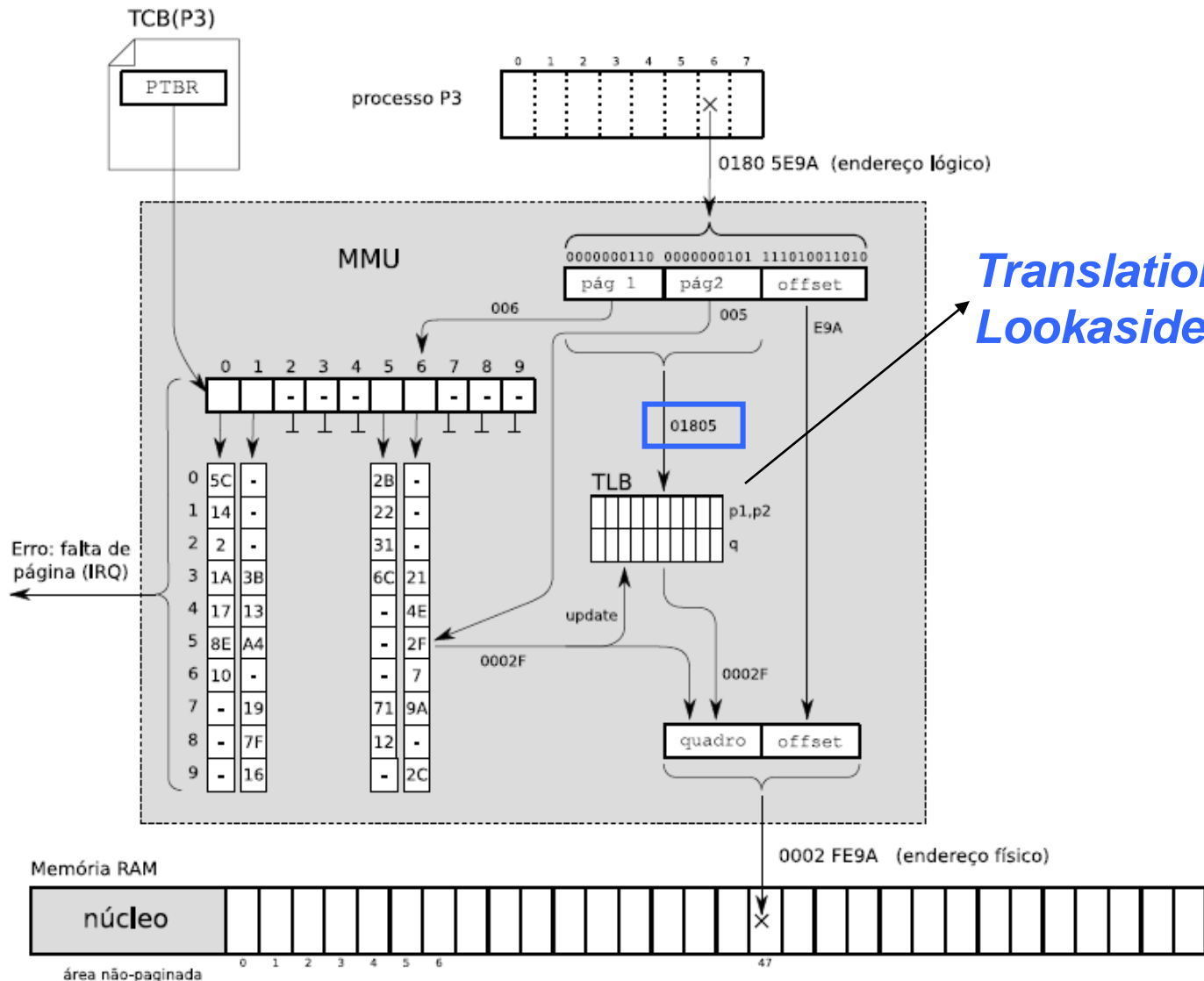
A estruturação das tabelas de páginas em **vários níveis** resolve o problema do espaço ocupado pelas tabelas de forma muito eficiente...

...mas, tem um efeito colateral muito nocivo:

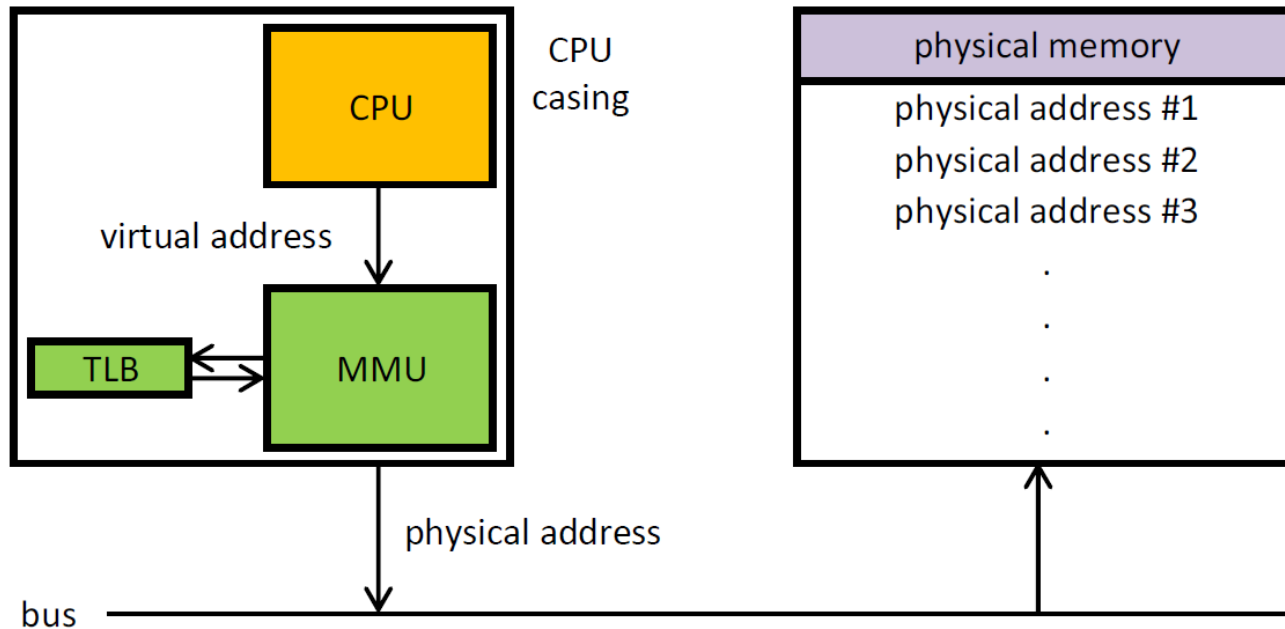
aumenta drasticamente o tempo de acesso à memória.

Solução \Rightarrow Cache da tabela de páginas

Cache da tabela de páginas



Cache da tabela de páginas



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

Tempo de acesso → acerto = 1 ciclo de clock
erro = 10 a 30 ciclos de clock

Cache da tabela de páginas



O **tempo médio** de acesso à memória pode então ser determinado pela média ponderada entre o tempo de acesso com **acerto** de cache e o tempo de acesso no caso de **erro**.

Cache da tabela de páginas

Ex.:

$Ck = 2 \text{ GHz}$, $T = 0,5 \text{ ns}$

Tempo de acesso a RAM = 50 ns

Tabelas de páginas com 3 níveis

Custo TLB: Acerto = 0,5 ns e Erro = 10 ns (20 ciclos de clock)
Taxa de acerto = 95%

Cache da tabela de páginas

Ex.:

$Ck = 2 \text{ GHz}$, $T = 0,5 \text{ ns}$

Tempo de acesso a RAM = 50 ns

Tabelas de páginas com 3 níveis

Custo TLB: Acerto = 0,5 ns e Erro = 10 ns (20 ciclos de clock)
Taxa de acerto = 95%

T_{medio}	= 95% x 0,5ns	//em caso de acerto
	+ 5% x (10ns + 3 x 50ns)	//em caso de erro, consultar
		//as tabelas
	+ 50ns	//acesso ao quadro desejado

= 58,475ns → Aumento de ~ 17 % para acesso em UM nível

A decorative graphic on the left side of the slide. It consists of a vertical light blue bar, a vertical orange bar, and a grey rectangle. A horizontal dark blue bar extends from the grey rectangle across the top of the slide.

Alocação híbrida

Alocação híbrida

- Alocação contígua → Simplicidade
- Alocação segmentada → Flexibilidade
- Alocação por páginas → Endereço linear, baixa fragmentação

⇒ **Alocação segmentada-paginada**

Alocação híbrida

- Maioria dos sistemas operacionais **não** usa;
- Sistemas da família Windows NT (2000, XP, Vista) e da família UNIX (Linux, FreeBSD) usam **alocação por páginas**;
- O antigo DOS e o Windows 3.x usavam a **alocação por segmentos**;
- O OS/2 da IBM - um dos poucos a fazer uso da **alocação segmentada-paginada**.

Exercícios

1. Explique a diferença entre endereços *lógicos* e endereços *físicos* e as razões que justificam seu uso.
2. Explique em que consiste a resolução de endereços nos seguintes momentos:

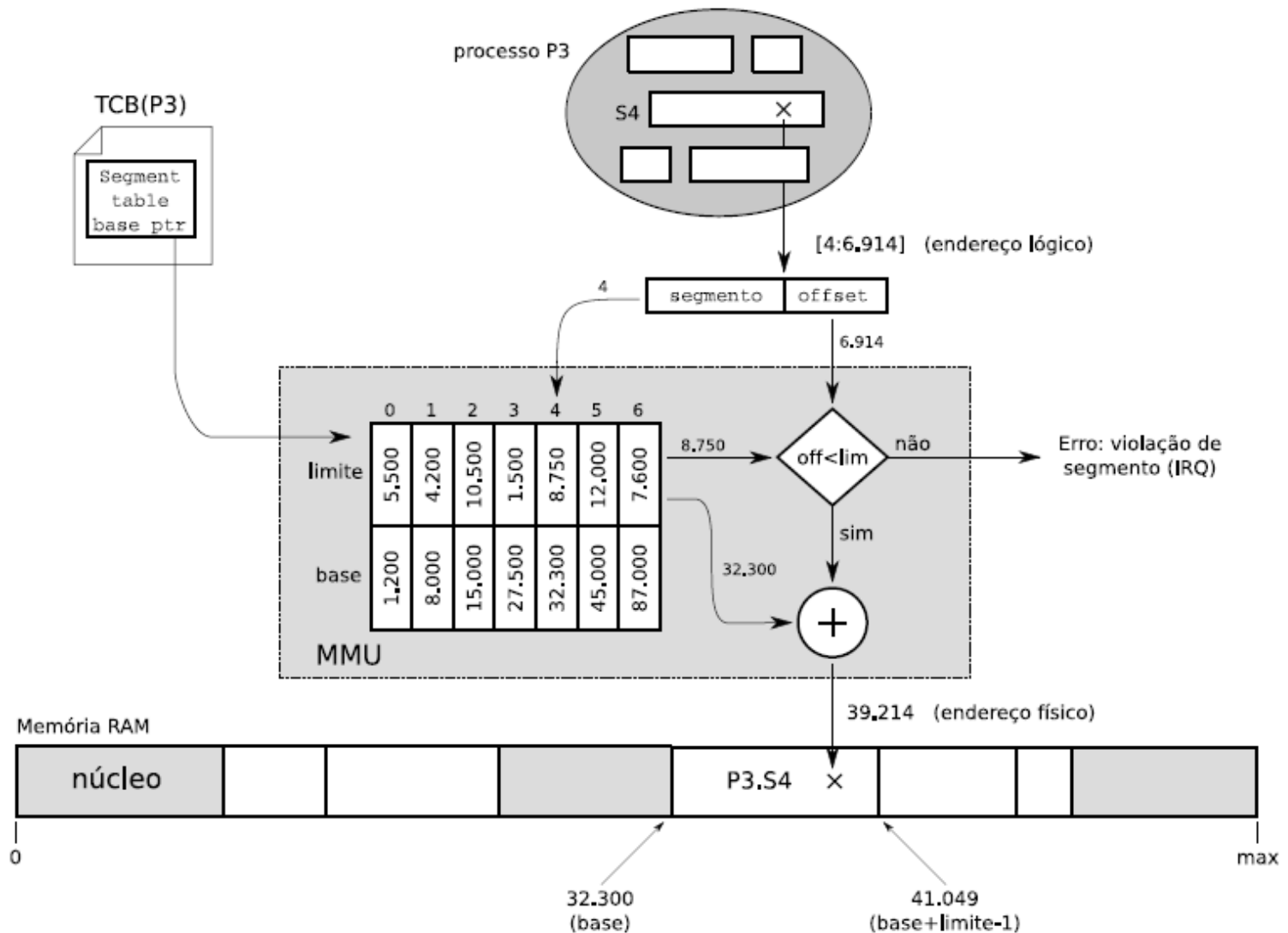
codificação, compilação, ligação, carga e execução.
3. Como é organizado o espaço de memória de um processo?
4. O que é uma MMU – *Memory Management Unit*?
5. Seria possível e/ou viável implementar as conversões de endereços realizadas pela MMU em software, em vez de usar um hardware dedicado? Por que?

Exercícios

Considerando a tabela de segmentos abaixo (com valores em decimal), calcule os endereços físicos correspondentes aos endereços lógicos 0:45, 1:100, 2:90, 3:1.900 e 4:200.

Segmento	0	1	2	3	4
Base	44	200	0	2.000	1.200
Limite	810	200	1.000	1.000	410

Exercícios



Exercícios

Considerando a tabela de segmentos abaixo (com valores em decimal), calcule os endereços físicos correspondentes aos endereços lógicos 0:45, 1:100, 2:90, 3:1.900 e 4:200.

Segmento	0	1	2	3	4
Base	44	200	0	2.000	1.200
Limite	810	200	1.000	1.000	410

Exercícios

Considerando a tabela de segmentos abaixo (com valores em decimal), calcule os endereços físicos correspondentes aos endereços lógicos 0:45, 1:100, 2:90, 3:1.900 e 4:200.

Segmento	0	1	2	3	4
Base	44	200	0	2.000	1.200
Limite	810	200	1.000	1.000	410

$$0:45 \rightarrow 44 + 45 = 89$$

Exercícios

Supondo um tamanho de página de 1kB, quais são os números de página e deslocamento (offset) para as referências de endereço a seguir:

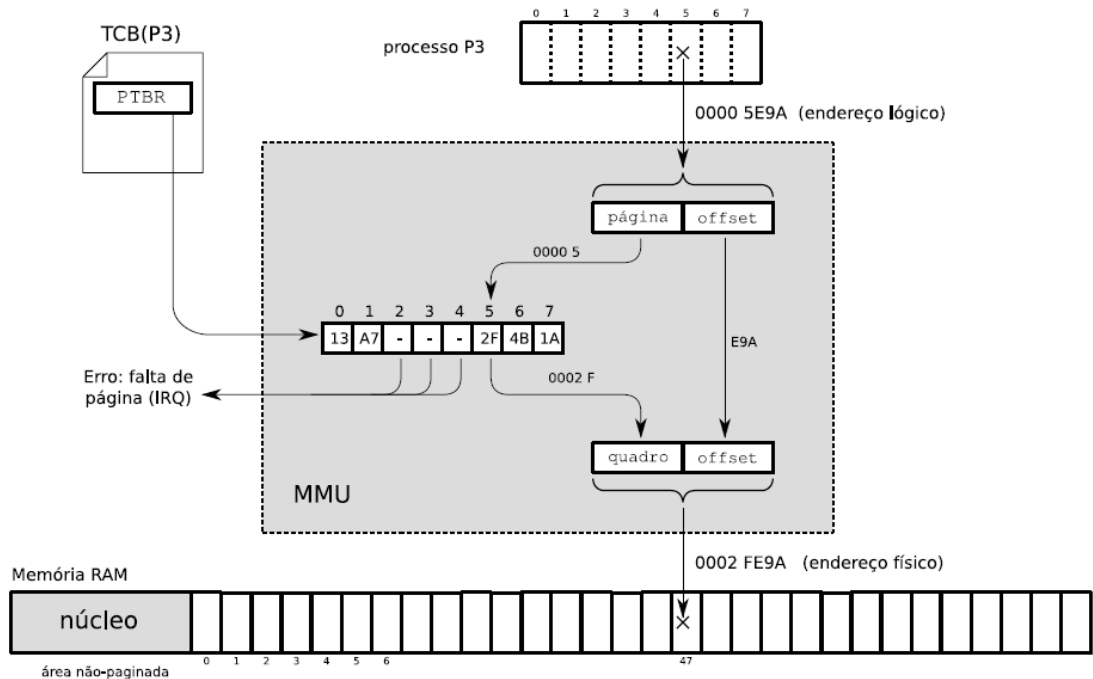
1. 0x0947
2. 0x4BA6
3. 0x7530
4. 0x0256

Exercícios

1. $0x0947 = 0000\ 1001\ 0100\ 0111$

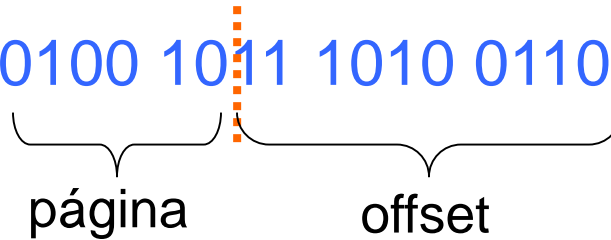
página offset

Offset = 327
Página = 2



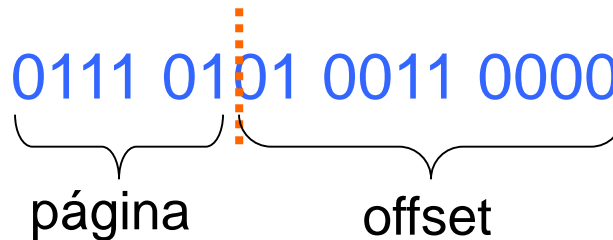
Exercícios

2. $0x4BA6 = 0100\ 1011\ 1010\ 0110$


The binary sequence 0100 1011 1010 0110 is divided by a vertical dashed line. The left side, 0100 1011, is labeled 'página' with a bracket underneath. The right side, 1010 0110, is labeled 'offset' with a bracket underneath.

Offset = 934
Página = 18

3. $0x7530 = 0111\ 0101\ 0011\ 0000$


The binary sequence 0111 0101 0011 0000 is divided by a vertical dashed line. The left side, 0111 0101, is labeled 'página' with a bracket underneath. The right side, 0011 0000, is labeled 'offset' with a bracket underneath.

Offset = 304
Página = 29