

## Memória virtual: Paginação

- A Memória é dividida em Molduras (ou Frames) de mesmo tamanho  
Tamanho das Páginas = tamanho das Molduras (e.g., 4K):
- Fragmentação interna pequena
- Processo não precisa ocupar área contígua em memória
- Elimina fragmentação externa
- Processo não precisa estar completamente na MP
- SO mantém uma tabela de páginas por processo
- Endereços são gerados dinamicamente em tempo de execução


## Paginação: Como funciona?

Endereço virtual = (no da página , deslocamento)


**MOV REG, 20500**

1-  $20500/4096$  (Tamanho da página 4k =  $4 \times 1024$ ) = página 5 (moldura 3)

2- deslocamento =  $20500 - 20K + 12K$

**Endereço Virtual 20500**  
010 1000 0000 0000

Laboratório de Pesquisa em Redes e Multimídia



## Paginação: Como funciona? (2)

Virtual address space			Physical memory address
60K-64K	X	} Virtual page	28K-32K
56K-60K	X		24K-28K
52K-56K	X		20K-24K
48K-52K	X		16K-20K
44K-48K	7		12K-16K
40K-44K	X		8K-12K
36K-40K	5		4K-8K
32K-36K	X		0K-4K
28K-32K	X		
24K-28K	X		
20K-24K	3		
16K-20K	4		
12K-16K	0		
8K-12K	6		
4K-8K	1		
0K-4K	2		

Page frame 10

- **MOV REG, 20500**
  - **Qual é a página?**  
Pag. 5, que contém os endereços de 20k (20480) até 24k-1 (24575)
  - **Esta página está em qual moldura?**  
Na moldura 3, que contém end. físicos de 12k (12288) a 16k-1 (16384)
  - **Qual o deslocamento do endereço 20500 dentro da página?**  
Desl. = End. virtual – End. virtual do 1º byte da página  
=  $20500 - 20480 = 20$
  - **Qual será o endereço físico correspondendo ao end. virt. 20500?**  
= End. do 1º byte da moldura + desloca.  
=  $12288 + 20 = 12308$

Sistemas Operacionais

Ex prático:

### **Espaço Virtual X Tamanho da Página**

<b>Espaço de Endereçamento Virtual</b>	<b>Tamanho da página</b>	<b>Número de páginas</b>	<b>Número de entradas nas tabela de páginas</b>
<b><math>2^{32}</math> endereços</b>	<b>512 bytes</b>	<b><math>2^w</math></b>	<b><math>2^w</math></b>
<b><math>2^{32}</math> endereços</b>	<b>4 kbytes</b>	<b><math>2^x</math></b>	<b><math>2^x</math></b>
<b><math>2^{48}</math> endereços</b>	<b>4 kbytes</b>	<b><math>2^y</math></b>	<b><math>2^y</math></b>
<b><math>2^{48}</math> endereços</b>	<b>64 kbytes</b>	<b><math>2^z</math></b>	<b><math>2^z</math></b>

Qual é o número w ?

- 1) Temos  $2^{32}$  endereços
- 2) Temos 512 bytes =  $2^9$
- 3) Número de páginas =  $32 - 9 = 23$

Qual é o número x?

- 1) Temos  $2^{32}$  endereços
- 2) Temos 4K bytes =  $4096 = 2^{12}$
- 3) Número de páginas =  $32 - 12 = 20$

Qual é o número y ?

- 1) Temos  $2^{48}$  endereços
- 2) Temos 4K bytes =  $4096 = 2^{12}$
- 3) Número de páginas =  $48 - 12 = 36$

Qual é o número z ?

- 1) Temos  $2^{48}$  endereços
- 2) Temos 64K bytes =  $2^{16}$
- 3) Número de páginas =  $48 - 16 = 32$

Considere um sistema com endereços lógicos de 32 bits, páginas de 1KB e o tamanho de uma entrada na tabela igual a 4 bytes cada. O tamanho total da tabela, em megabytes é ..

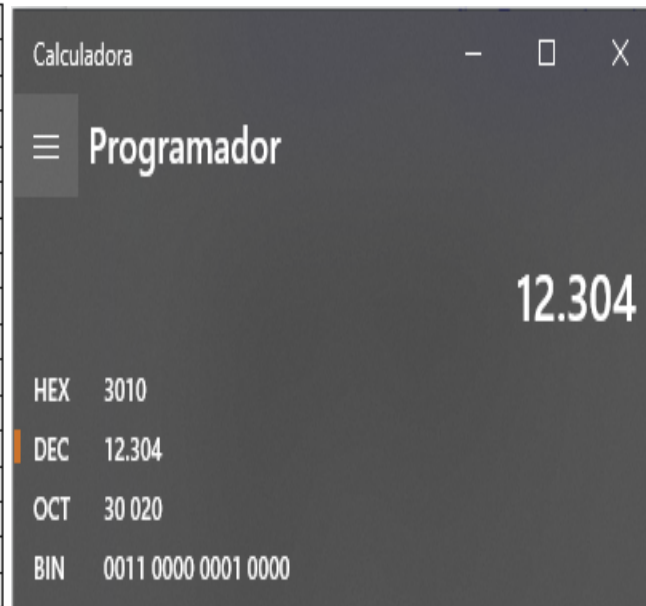
- Tamanho da página = 1KB =  $2^{10}$  (10 bit de deslocamento)
- Endereço lógico  $2^{32}$  = 32 bits
- Tamanho de entrada da tabela = 4 B = 4bytes (1 MB= $2^{20}$  B)
- Cálculo -> *Número de páginas \* tamanho de cada entrada*  
=  $(2^{32} - 2^{10}) \times 4B$   
=  $2^{22} \times 2^2B$   
=  $2^{24}B$   
=  $2^4 \times 2^{20}$   
=  $2^4 \times 1MB$   
= 16 MB

Considere uma máquina com 64MB de memória física e endereços virtuais de 32 bits. Se o tamanho da página é 4KB, considerando que cada entrada na tabela ocupa 32 bits, qual o tamanho da tabela. Considere que o tamanho da entrada na tabela (em bits) deve ser múltiplo de 8 por questões de alinhamento de memória.

- Memória Física = 64MB =  $2^{26}B$  (26 bits endereços físicos)
- Endereço Virtual =  $2^{32}$  bits
- Tamanho da Entrada = 32 bits
- Número de Páginas =  $2^{32} - 2^{12} = 2^{20}$
- Tamanho da Página = 4KB =  $2^{12}B$
- Cálculo ->  
=Número de páginas x tamanho da entrada  
=  $(2^{20}) \times 32$  bits  
=  $(2^{20}) \times 4$ Bytes  
=  $2^{20} \times 2^2$ Bytes  
= 4M

Considere um sistema com páginas de 4K, endereçamento lógico de 16 páginas, e endereçamento físico de 8 frames. Considere a seguinte tabela de página do processo em execução. Em qual endereço físico a MMU traduz a seguinte referência à memória feita pelo processo corrente: endereço 12304

	bit validade	moldura
0	1	2
1	1	1
2	1	6
3	1	0
4	1	4
5	1	3
6	0	-
7	0	-
8	0	-
9	1	5
10	0	-
11	1	7
12	0	-
13	0	-
14	0	-
15	0	-



Respondendo:

"0011 0000 0001 0000"

Número de páginas = 0011 = página 3

Deslocamentos = 0000 0001 0000

Ná página 3, possui a moldura igual a 0, então esse é o endereço correto:  
0000 0000 0001 0000.

### Tabela Invertida

Tabelas com o tamanho da quantidade de molduras (memória real) e não da quantidade de páginas (memória virtual).

Temos uma entrada para cada frame da memória física, e o número do frame é um índice nessa tabela.

Além disso, cada entrada deve indicar , além do número da página, qual é o PID do processo que está ocupando aquele frame.

Por fim, temos o Bit de presença (ou validade) que indica se o frame está de fato ocupado por alguma página de algum processo ou se ele está vago.

	bit validade	moldura
0	1	2
1	1	1
2	1	6
3	1	0
4	1	4
5	1	3
6	0	-
7	0	-
8	0	-
9	1	5
10	0	-
11	1	7
12	0	-
13	0	-
14	0	-
15	0	-

Tabela de Página Invertida			
	Bit Presença	PID	No. da página
frame 0	1	pid-x	3
frame 1	1	pid-x	1
frame 2	1	pid-x	0
frame 3	1	pid-x	5
frame 4	1	pid-x	4
frame 5	1	pid-x	9
frame 6	1	pid-x	2
frame 7	1	pid-x	11

Como nesta segunda imagem, representa a tabela feita a partir da moldura Moldura (as que levam número na parte da moldura) / Bit Presença (0 ou 1)/ PiD/ Número de página (apontada pela aquela moldura).

O bit de presença pode ser zero, entretanto isso indica que ele pode ter sido no passado, usado por uma página no processo ele não está mais sendo ocupado.

Na parte de pid, ele pode representar diferentes processos.

## Tabela Multinível

O objetivo é evitar manter toda a tabela de páginas na memória durante todo o tempo.

- Tabela de dois níveis

Uso de dois apontadores e um deslocamento

O endereço de 32 bits de endereço dividido em 3 campos

**PT1 [10 bits]** : indexa o primeiro nível da tabela

**PT2 [10 bits]** : indexa o segundo nível da tabela

**Deslocamento [12 bits]**: => páginas de 4 KB

Considere o end. virtual 0x00403004 (4206596d)

Lprm Laboratório de Pesquisa em Redes e Multimídia

### Tabela de Página Multinível (7)

PT1	PT2	Deslocamento
0000000001	0000000011	0000 0000 0100

- PT1: Entrada 1 da tabela do 1º nível
  - 2º bloco de 4M (4M a 8M de memória virtual)
- PT2: Entrada 3 da tabela do 2º nível
  - Esta entrada indica em qual moldura encontra-se esta página
  - O endereço físico do primeiro byte dessa moldura é somado ao deslocamento
    - Supondo a página encontra-se na moldura 1 (4k a 8k-1), o endereço físico correspondente será  $4096 + 4 = 4100$
- OU:

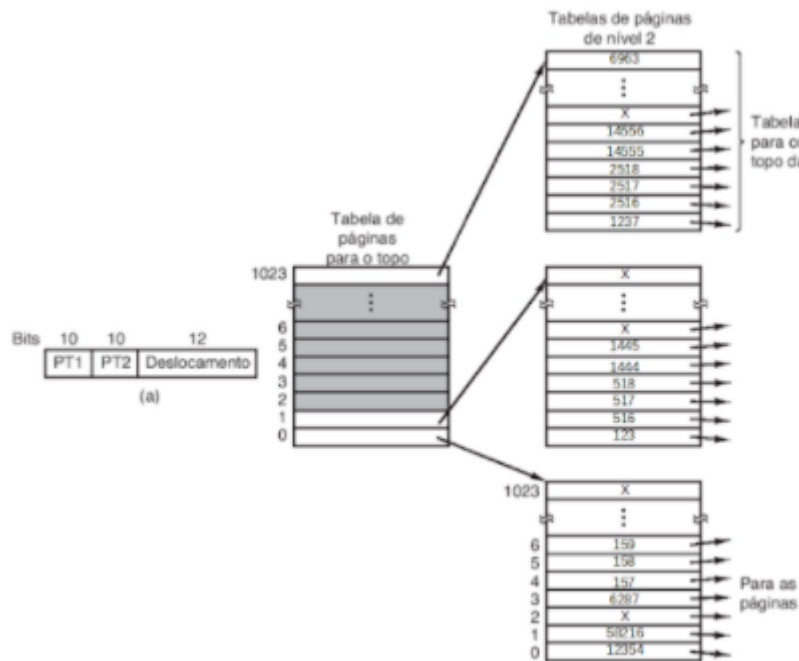
Nº da moldura	Deslocamento
0... 00001	0000 0000 0100

= 4100<sub>d</sub>

LPRM/DI/UFES 32 Sistemas Operacionais

Exemplo prático:

Considerando a tabela multinível abaixo, dado o endereço virtual 0XFFC00F33 , em qual endereço físico ele seria traduzido. Considere que o endereço físico tem o mesmo tamanho do endereço virtual, e que nas tabelas de página de nível 2 na figura temos, em cada entrada, o número do frame no qual se encontra cada página (X indica que a página não se encontra em memória).



Convertendo o número 0XFFC00F33 para binário:

binário: 1111 1111 1100 0000 0000 1111 0011

PT1 = 10 / PT2 = 10 / DESLOCAMENTO = 12 -> são índices

PT1 = 1111 1111 11 (Última entrada na tabela de primeiro nível, o 1023)

PT2 = 00 0000 0000 (Se refere ao primeiro campo na tabela de nível 2, o 1237)

DESLOCAMENTO = 1111 0011

O frame 1237 em binário é: 100 1101 0101

O endereço físico é: deslocamento concatenado com o endereço de moldura

= 100 1101 0101 1111 0011

1237

deslocamento

