

Ministério da Educação

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

UNED Nova Friburgo

Bacharelado em Sistemas da Informação

Gerência de Memória (1)

Sistemas Operacionais



Prof. Bruno Policarpo Toledo Freitas
bruno.freitas@cefet-rj.br



Objetivos

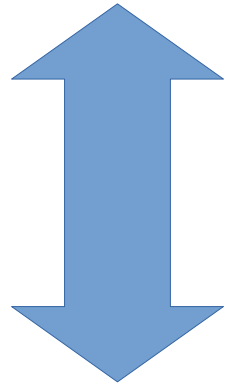
- **Compreender como a hierarquia de memória afeta o desempenho do computador**
- **Definir as principais propriedades do espaço de endereçamento de um processo**
- **Apresentar algoritmos para gerência de memória livre e compreender suas vantagens e desvantagens**
- **Apresentar o conceito de memória virtual e o funcionamento da paginação**

Gerência de Memória

- **Em um mundo ideal, programas querem:**
 - Privada
 - Infinitamente grande
 - Infinitamente rápida
 - Não-volátil
- **O que realmente existe:**
 - Cache + memória primária + memória secundária
- **Abstrair esse conjunto de forma que a memória pareça *um recurso único* é tarefa do gerenciador de memória do sistema operacional.**

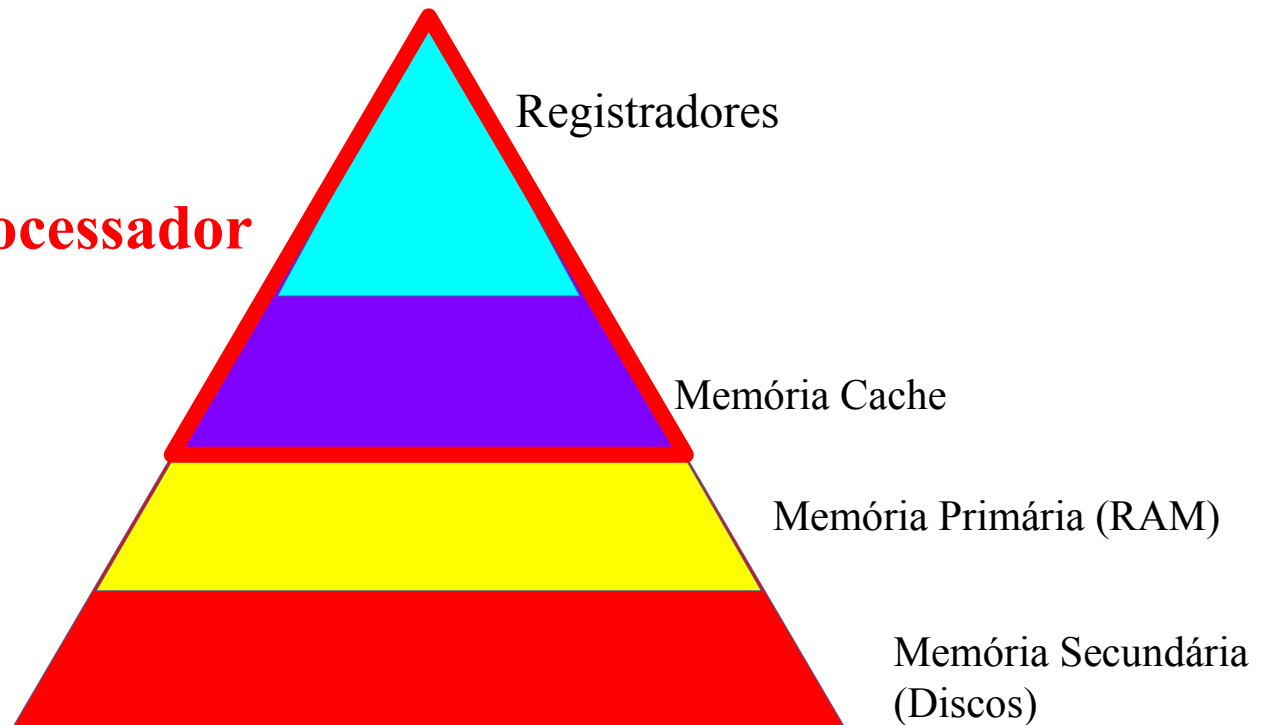
Hierarquia de Memória

Custo maior
Velocidade maior
Capacidade menor



Processador

Custo menor
Velocidade menor
Capacidade maior



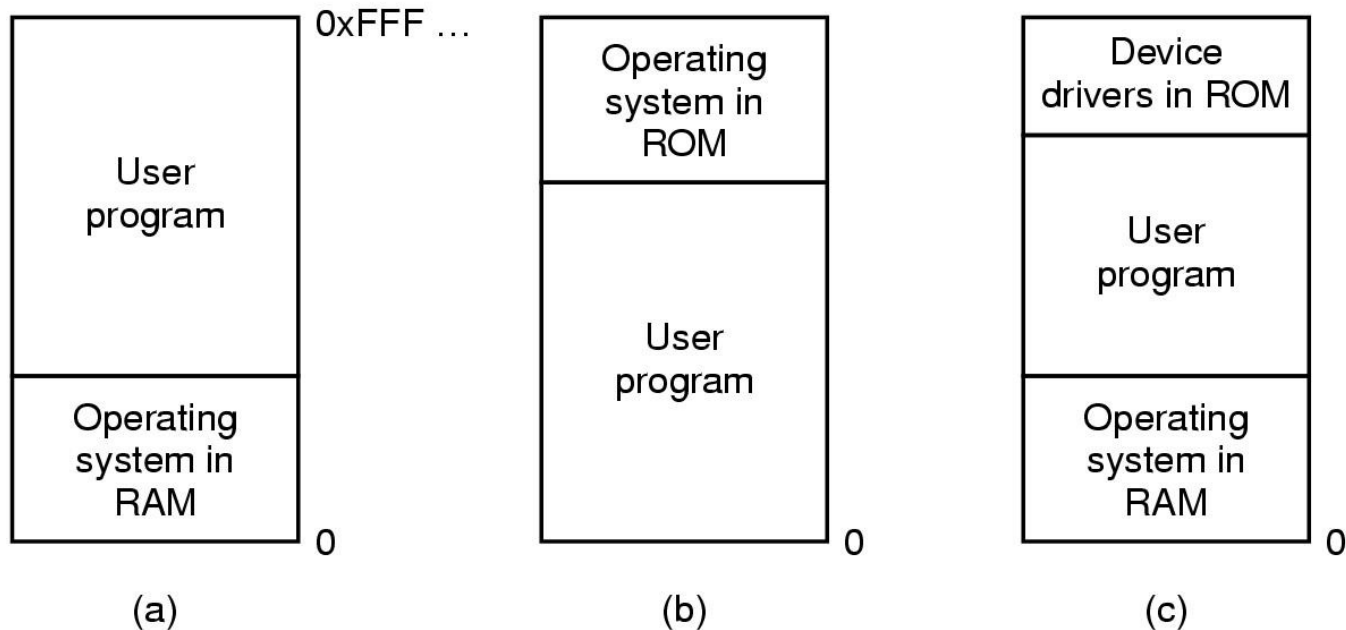
Sem abstração de memória

- **Sem abstração de memória, programas acessariam diretamente endereços:**

MOV REGISTER1, 1000

- **Multiprogramação na memória não é possível**
 - Interferência entre programas
- **Porém, o SO ainda consegue executar mais de um programa**
 - Swapping

Sem abstração de memória



- **Utilizado em sistemas embarcados pequenos**
 - “S.O” é apenas uma biblioteca
 - Geralmente chamado de *Barebone*

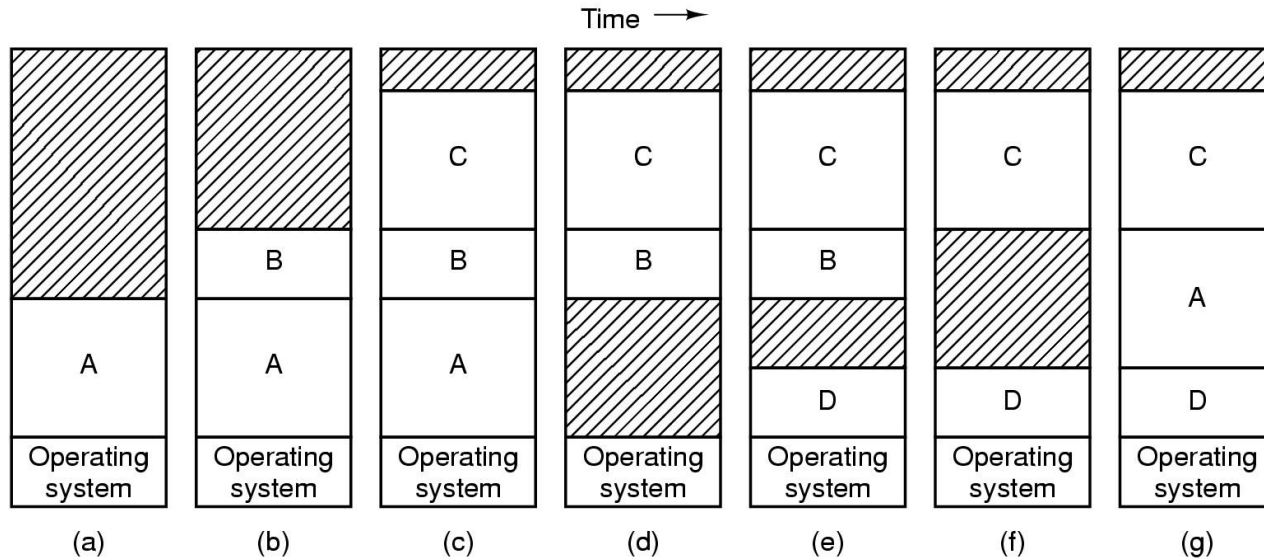
Espaço de Endereçamento

- **Expor toda a memória física aos programas é perigoso**
 - Proteção
- **Quando um programa é carregado, não se sabe onde ele será colocado na memória**
 - Relocação
- **O espaço de endereçamento é a região de memória que um programa pode endereçar.**

Sobrecarga da memória

- **Na prática, sempre haverá mais processos do que memória física disponível**
 - Processos de boot
 - Processos em background
 - ... antes de iniciar o uso do computador
- **Duas estratégias principais:**
 - Swapping
 - Memória virtual

Swapping puro



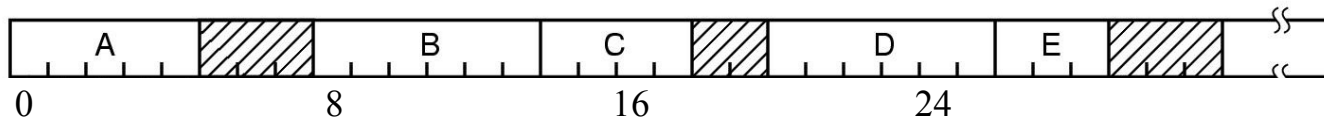
- **Se for necessário memória, retira um dos programas da mesma.**
- **Problema: espaços vazios entre programa**
 - *Fragmentação externa*
 - Solução: compactação
 - Problema: tempo

Gerência de espaço livre

- **Sistema operacional precisa controlar espaços livres e utilizados da memória de diversas maneiras**
 - Memória dinâmica (malloc/free em C ou new/delete C++)
 - Sistemas de arquivos
- **Dois métodos principais:**
 - Bitmap
 - Lista encadeada

Gerência de espaço livre

Bitmap

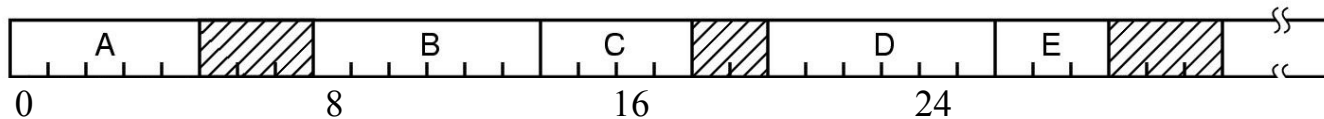


Byte 0	1	1	1	1	1	0	0	0
Byte 1	1	1	1	1	1	1	1	1
Byte 2	1	1	0	0	1	1	1	1
Byte 3	1	1	1	1	1	0	0	0

- **Vantagens:**
 - Simples
 - Tamanho necessário
- **Desvantagens:**
 - Custo da procura

Gerência de espaço livre

Bitmap



Byte 0	1	1	1	1	1	0	0	0
Byte 1	1	1	1	1	1	1	1	1
Byte 2	1	1	0	0	1	1	1	1
Byte 3	1	1	1	1	1	0	0	0

- **Vantagens:**
 - Simples
 - Tamanho necessário
- **Desvantagens:**
 - Custo da procura

Exemplo 1: Qual é o tamanho total de uma tabela de bitmaps para 128 MiB de memória e tamanho de bloco de 32 B?

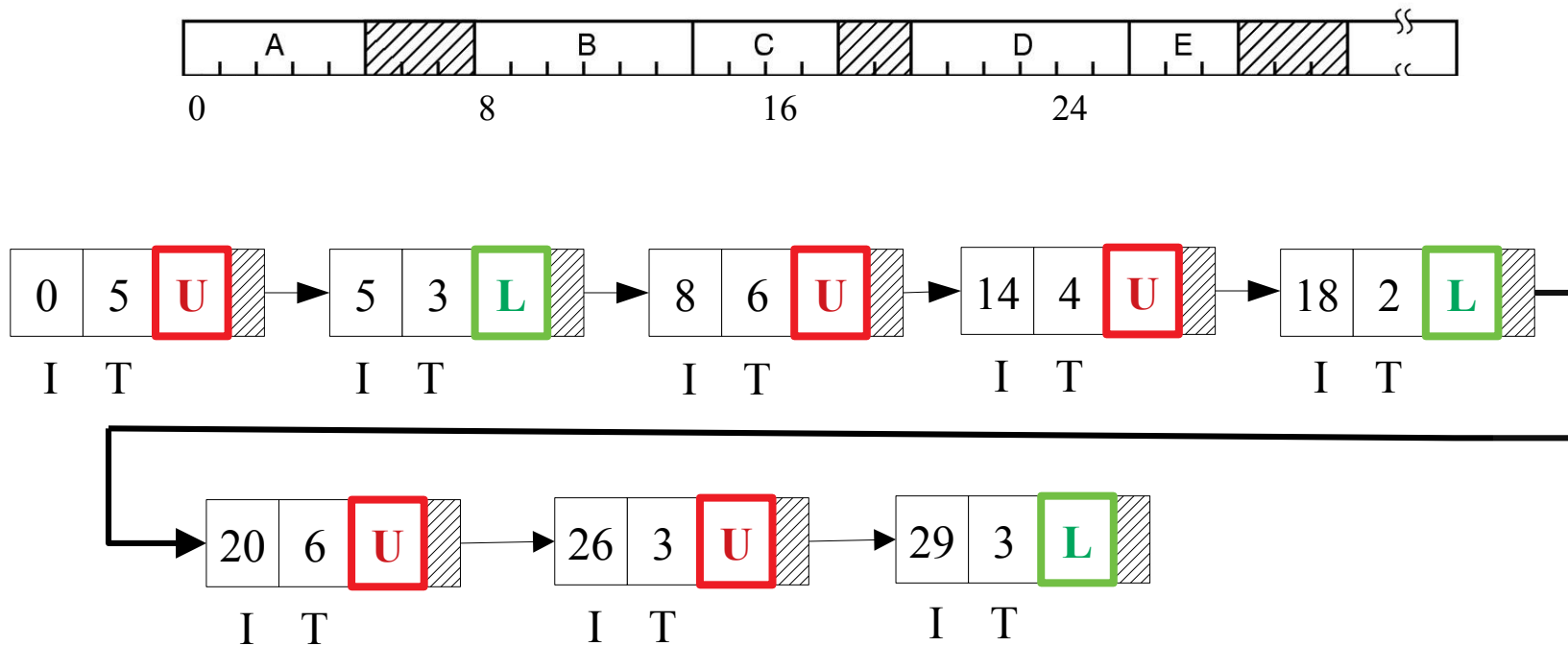
Gerência de espaço livre

Lista Encadeada

- **Mantém lista de blocos utilizados (U) e livres (L)**
 - Necessário: Início e Tamanho do Bloco
- **Vantagens:**
 - Procura mais rápida
 - Algoritmos de alocação de memória: *first-fit*, *next-fit*, *best-fit*, *worst-fit*, *quick-fit*
- **Desvantagens:**
 - Mais complexo

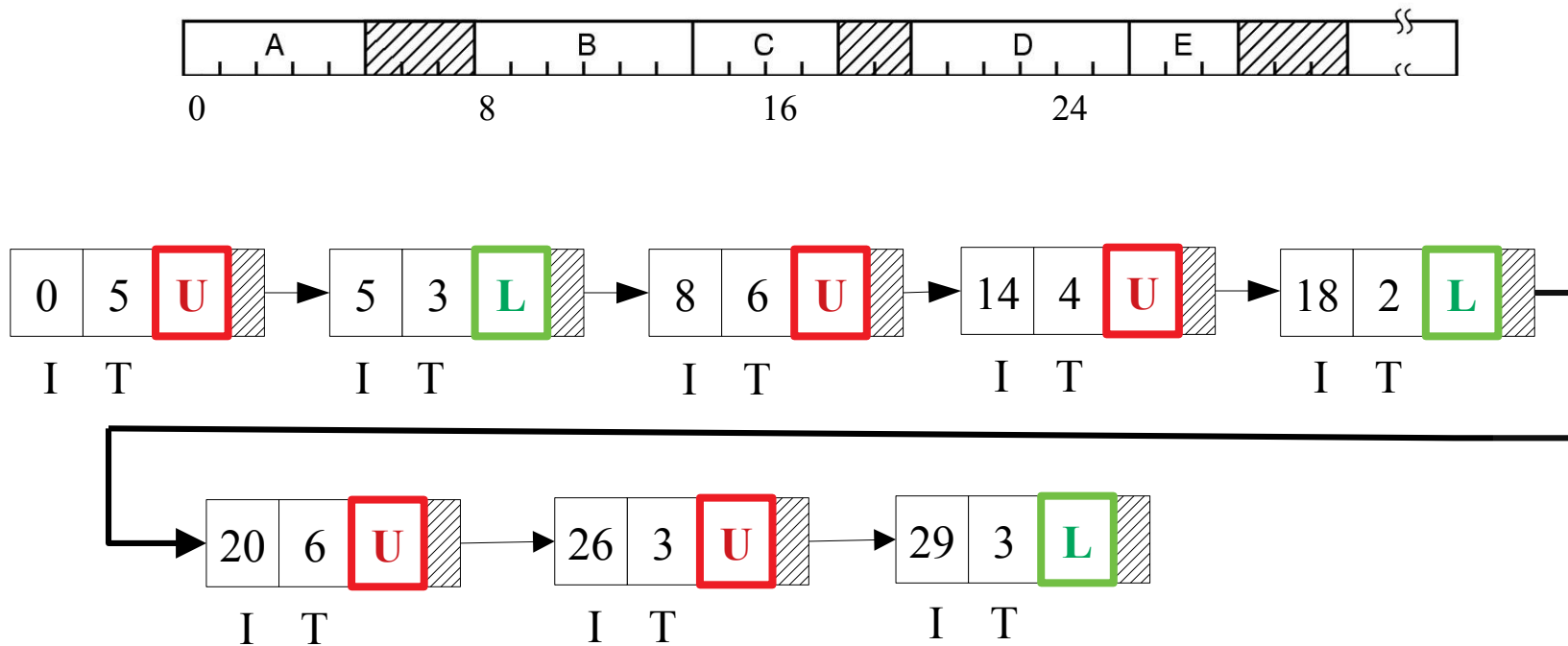
Gerência de espaço livre

Lista Encadeada



Gerência de espaço livre

Lista Encadeada



EXEMPLO 2

Considere a seguinte lista de memória livre (em MiB) : 28 – 30 – 5 – 6 – 10 – 12

Se for solicitado 4 MiB de memória, qual será o bloco de memória escolhido para cada um dos algoritmos abaixo?

Gerência de Memória

Lista encadeada

- **Tamanho da região: 128 MiB**
- **Tamanho de bloco: 32 B**

EXEMPLO 3

Considere o mesmo sistema de alocação de memória dinâmica do exercício 1 mas utilizando lista encadeada de blocos livres.

Quantos bits deve possuir o ponteiro de próximo nó da lista?

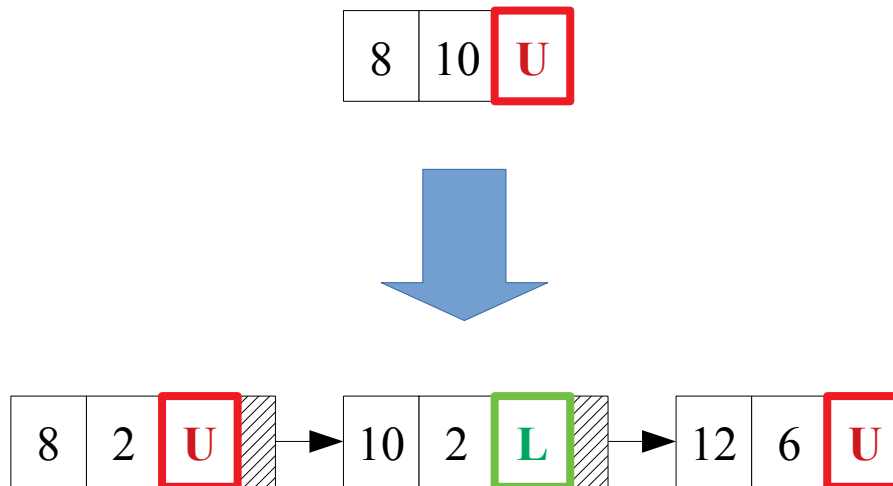
Exercícios resolvidos

- **Considere que um dado sistema de alocação de memória dinâmica utilize blocos de 128 B e que cada processo tenha reservado 4 MiB de memória para alocação dinâmica**
 - (1) Quanto espaço (em bits) será necessário para armazenar uma tabela em bitmaps?
 - (2) Qual deve ser o tamanho mínimo do ponteiro da lista de nodos caso se utilize uma lista encadeada?
- **Considere que em dado instante exista a sequência de blocos livres (em KiB): 128 - 32 -64 -8 -12 -20**
 - (3) Qual será o bloco alocado caso um processo solicite 20 KiB de memória, considerando os algoritmos first-fit, next-fit, worst-fit e best-fit?

Gerência de espaço livre

Lista Encadeada

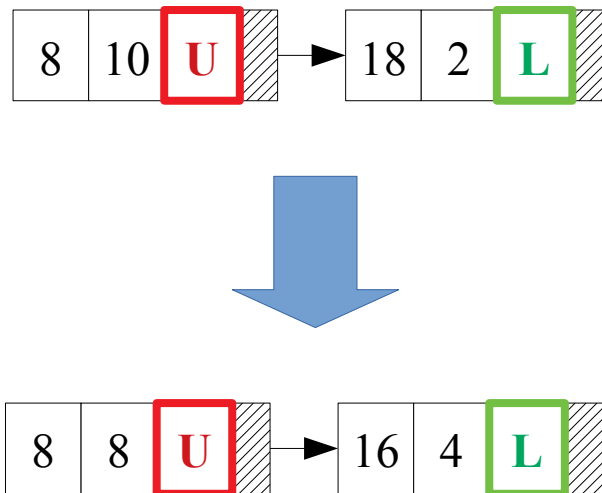
- **Caso 1: liberação de bloco ocupado no meio**
 - Exemplo: liberado 2 blocos a partir do bloco 10



Gerência de espaço livre

Lista Encadeada

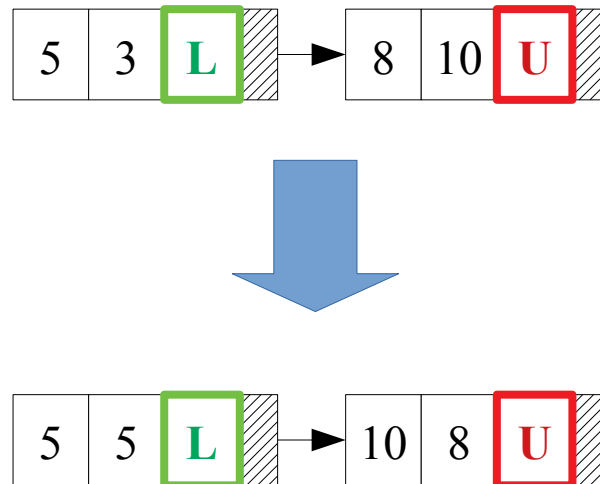
- **Caso 2: liberação de bloco ocupado no fim**
 - Exemplo: liberado 2 blocos a partir do bloco 16



Gerência de espaço livre

Lista Encadeada

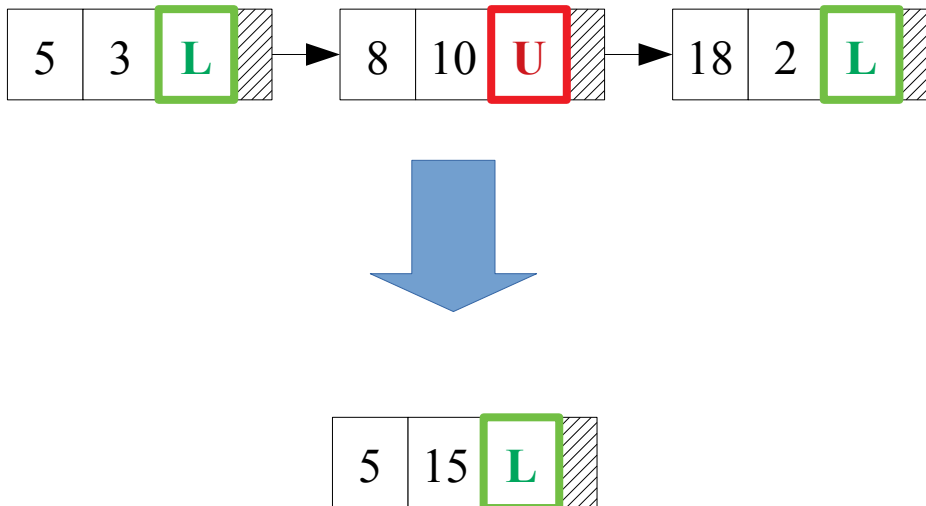
- **Caso 3: liberação de bloco ocupado no início**
 - Exemplo: liberado 2 blocos a partir do bloco 8



Gerência de espaço livre

Lista Encadeada

- **Caso 4: liberação de bloco ocupado no fim**
 - Exemplo: liberado o bloco inteiro



Memória virtual

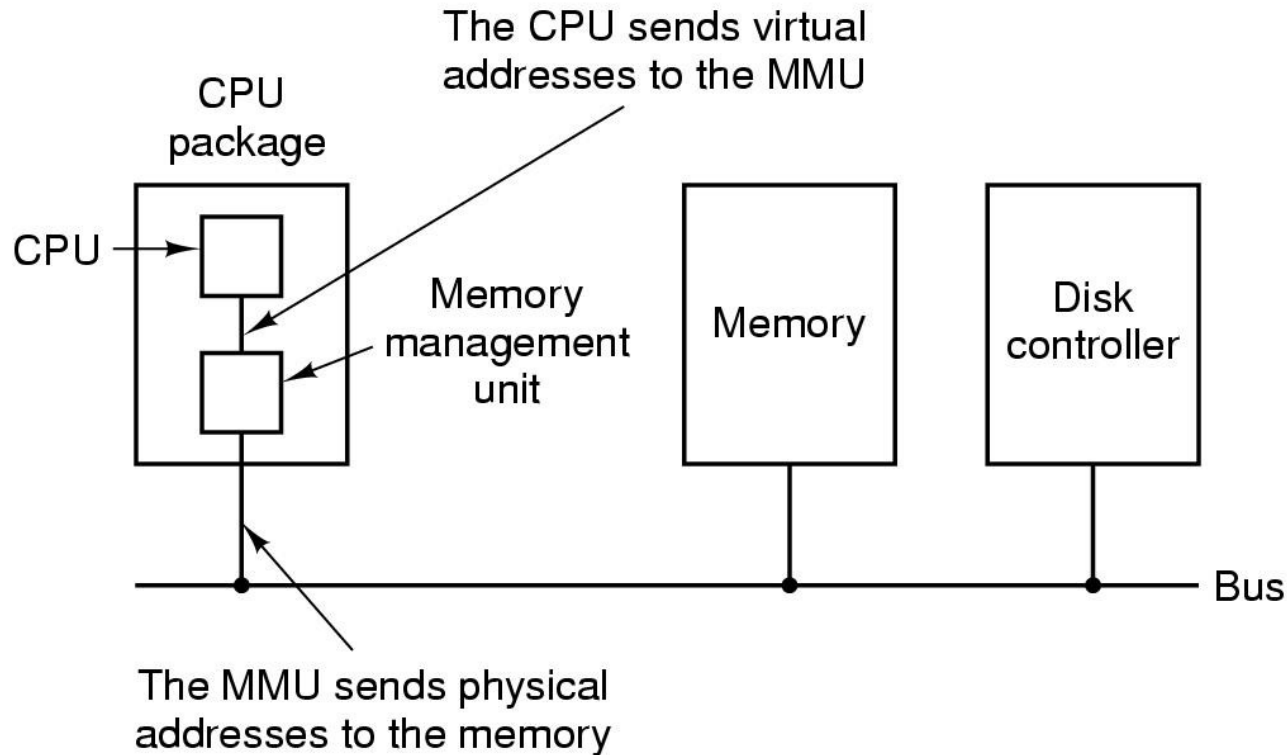
- Desde o início dos tempos, programas necessitavam de memória maior do que o disponível
- **Solução: memória virtual**
 - Programas enxergam mais memória do que realmente existe
 - Porém, apenas partes ativas do programa são mantidas na memória
 - Partes inativas residem na memória secundária via *swapping*
- O método de implementação de memória virtual mais comum é a paginação

Paginação

- **Princípio da paginação é dividir a memória em molduras de página (*frames*)**
- **Uma unidade de gerenciamento de memória (Memory Management Unit , MMU) transforma endereços virtuais em endereços físicos**
 - Endereços virtuais são maiores do que os endereços físicos, permitindo que processos “enxerguem” mais memória
 - Ambos endereçam páginas de mesmo tamanho
- **Programas mantêm na memória apenas páginas que estão utilizando**

Paginação

Unidade de Gerenciamento de Memória

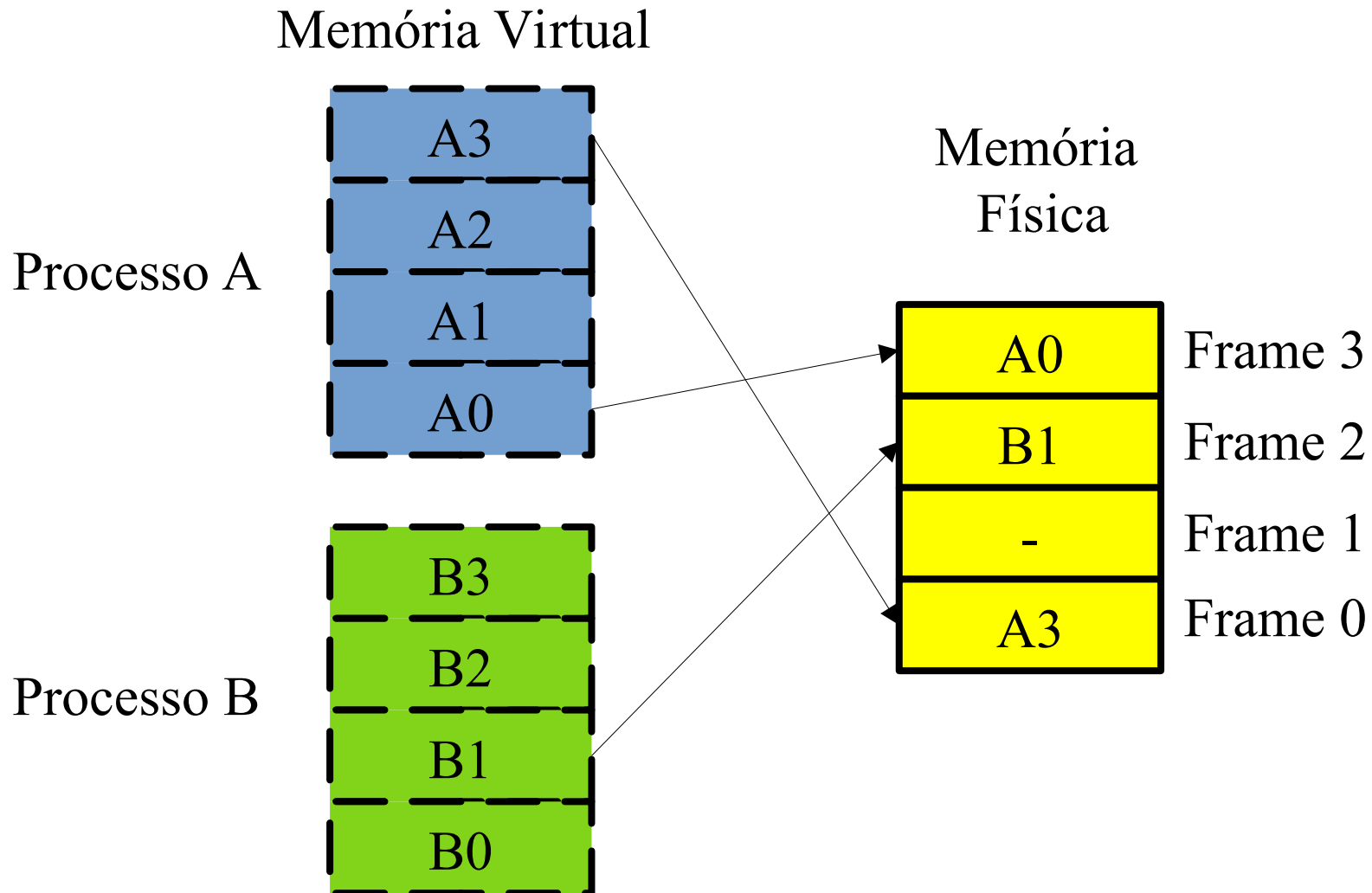


Posição da MMU

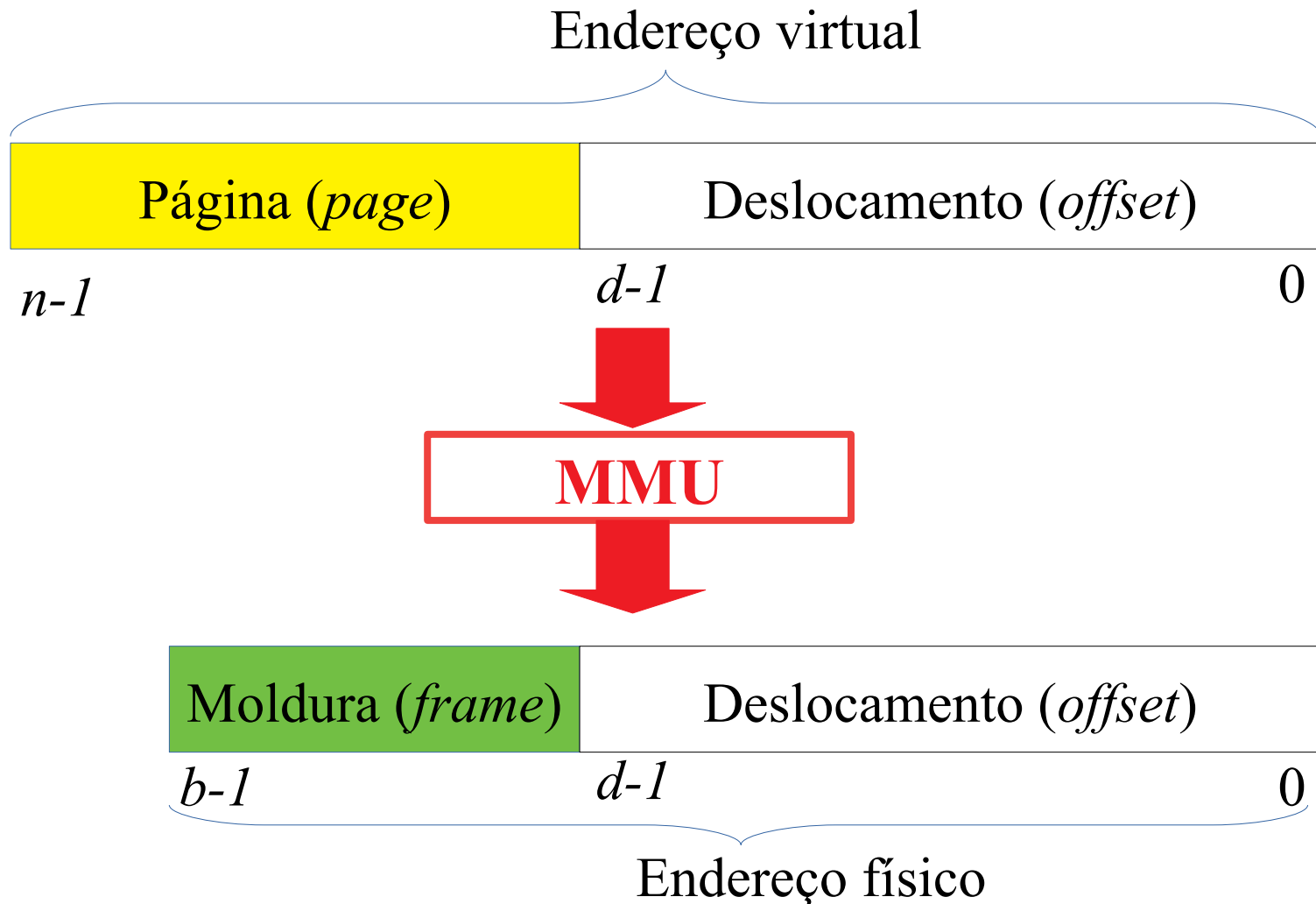
Paginação

- **Sistemas de paginação necessitam dos seguintes parâmetros:**
 - Tamanho do endereço virtual (em bits)
 - Tamanho do endereço físico (em bits)
 - Determina o máximo de memória física do computador
 - Na prática, é o tamanho da memória realmente instalada
 - Quantidade de molduras OU tamanho da moldura (em bits) OU tamanho da página
- **A tradução envolve substituir a página (*page*) do programa pela moldura (*frame*) em que ele se encontra, mantendo o mesmo deslocamento (*offset*)**

Paginação



Paginação



Exemplo de paginação

- **Considere o seguinte sistema:**

- Largura (em bits) do endereço virtual: 9 bits
- Largura (em bits) do endereço físico: 8 bits
- Quantidade de molduras: 4

- **A partir disso, deve-se obter:**

1. Quantidade de bits para endereçar uma *moldura* $\log_2(4)=2$
2. Quantidade de bits para endereçar o *deslocamento* $8-2=6 \text{ bits}$
3. Quantidade de bits para endereçar uma *página* $9-6=3 \text{ bits}$

8 páginas

Exemplo de paginação

- **Considere o seguinte sistema:**

- Largura (em bits) do endereço virtual:
 - 9 bits
- Largura (em bits) do endereço físico:
 - 8 bits
- Largura (em bits) da moldura:
 - 2 bits

- **A partir disso, deve-se obter:**

1. Quantidade de bits para endereçar o *deslocamento*

$$8 - 2 = 6 \text{ bits}$$

2. Quantidade de bits para endereçar uma *página*

$$9 - 6 = 3 \text{ bits}$$

8 páginas

Exemplo de Paginação

Endereços Virtuais (512 bytes)

7		447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3		192 - 255
2		128 - 191
1		64 - 127
0		0 - 63

Endereços Físicos (256 bytes)

3		192 - 255
2		128 - 191
1		64 - 127
0		0 - 63


Exemplo de Paginação

Endereços Virtuais (512 bytes)

7		447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3		192 - 255
2	0	128 - 191
1		64 - 127
0		0 - 63

Endereços Físicos (256 bytes)

Moldura escolhida
ao acaso

3		192 - 255
2		128 - 191
1		64 - 127
0		0 - 63

Endereço virtual 129

Exemplo de Paginação

Endereços Virtuais (512 bytes)

7	1	447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3		192 - 255
2	0	128 - 191
1		64 - 127
0		0 - 63

Endereços Físicos (256 bytes)

Moldura escolhida
ao acaso

3		192 - 255
2		128 - 191
1	✖	64 - 127
0	✖	0 - 63

Endereço virtual 450

Exemplo de Paginação

Endereços Virtuais (512 bytes)

7	1	447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3	2	192 - 255
2	0	128 - 191
1		64 - 127
0		0 - 63

Endereços Físicos (256 bytes)

Moldura escolhida
ao acaso

3		192 - 255
2	✖	128 - 191
1	✖	64 - 127
0	✖	0 - 63

Endereço virtual 195

Exemplo de Paginação

Endereços Virtuais (512 bytes)

7	1	447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3	2	192 - 255
2	0	128 - 191
1		64 - 127
0		0 - 63

Endereços Físicos (256 bytes)

Moldura escolhida
ao acaso

3		192 - 255
2	✖	128 - 191
1	✖	64 - 127
0	✖	0 - 63

Endereço virtual 150

Exemplo de Paginação

Endereços Virtuais (512 bytes)

7	1	447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3	2	192 - 255
2	0	128 - 191
1		64 - 127
0	3	0 - 63

Endereços Físicos (256 bytes)

Moldura escolhida
ao acaso

3	✖	192 - 255
2	✖	128 - 191
1	✖	64 - 127
0	✖	0 - 63

Endereço virtual 0

Exemplo de Paginação

Endereços Virtuais (512 bytes)

7	1	447 - 511
6		383 - 447
5		320 - 383
4		256 - 319
3	2	192 - 255
2	0	128 - 191
1		64 - 127
0	3	0 - 63

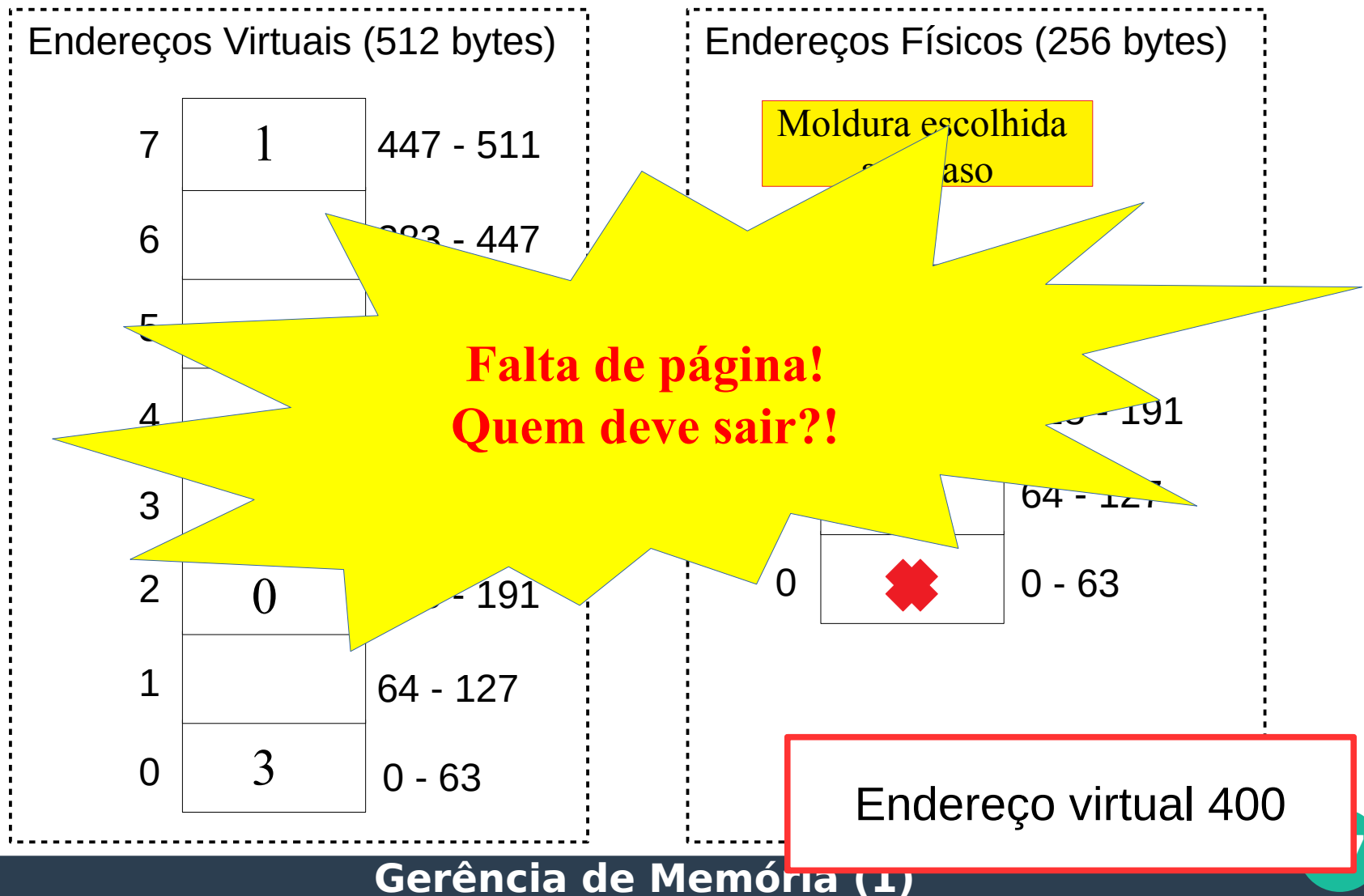
Endereços Físicos (256 bytes)

Moldura escolhida
ao acaso

3	✖	192 - 255
2	✖	128 - 191
1	✖	64 - 127
0	✖	0 - 63

Endereço virtual 400

Exemplo de Paginação



Exercício 4

- **Para um computador com 10 bits para endereços virtuais, 8 bits para endereços físicos, e 8 molduras, responda:**
 - a) Qual é a quantidade de bits necessária para endereçar molduras?
 - b) Quantas páginas cada processo possui nesse sistema?
 - c) Esboce as faixa de endereços virtuais e físicos desse sistema

Tabela de páginas

- Local onde são armazenadas as informações de mapeamento de endereços virtuais para físicos
- A *página* endereça uma posição da tabela e verifica se ela está na memória
 - Se sim: substitui os bits da página pela moldura
 - Se não: carrega a página em uma moldura e atualiza a tabela

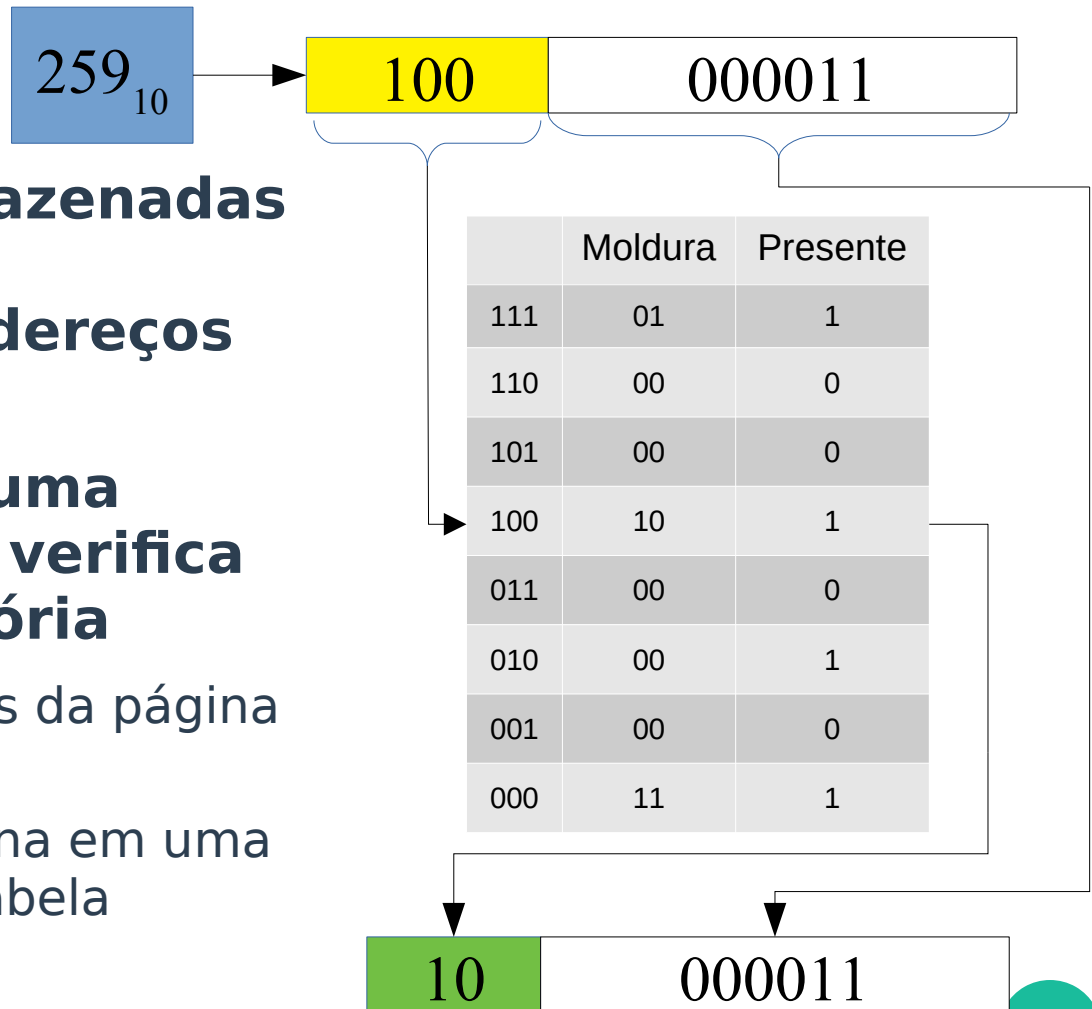
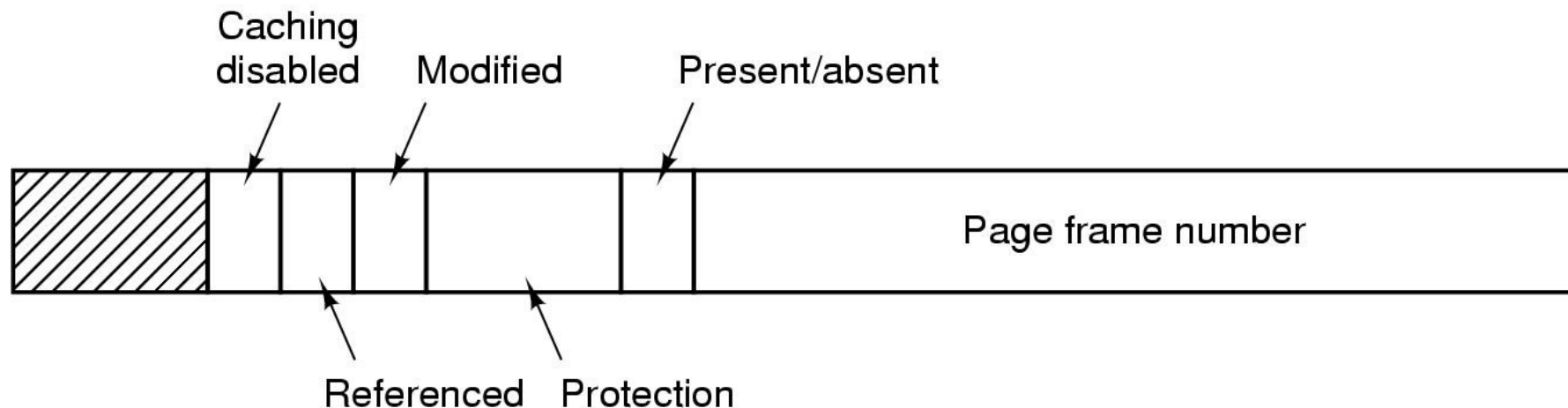


Tabela de páginas

Entrada Típica



- 1) Tamanho da tabela do exemplo de paginação?
- 2) **Tamanho da Tabela do Exercício 3?**

Translation Lookaside Buffers (TLB)

- **Dois problemas devem ser considerados para implementação de paginação:**
 - Tamanho ocupado pela tabela
 - Desempenho
- **Para melhorar o desempenho, utiliza-se TLB's**
 - Dentro da MMU
 - “Memória Cache” para a tabela de páginas
 - Comparação em paralelo

TLBs

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Referências

- **TANENBAUM, A.S. *Sistemas Operacionais Modernos*, 3ª. ed. São Paulo: Pearson Prentice Hall, 2009**
 - Cap. 3: 3.2, 3.3.1 até 3.3.3, 3.4

Exercícios

- **TANENBAUM, A.S. *Sistemas Operacionais Modernos*, 3ª. ed. São Paulo: Pearson Prentice Hall, 2009**
 - Cap. 3: 3, 4, 7, 8, 10, 18, 22, 23, 24, 28, 32