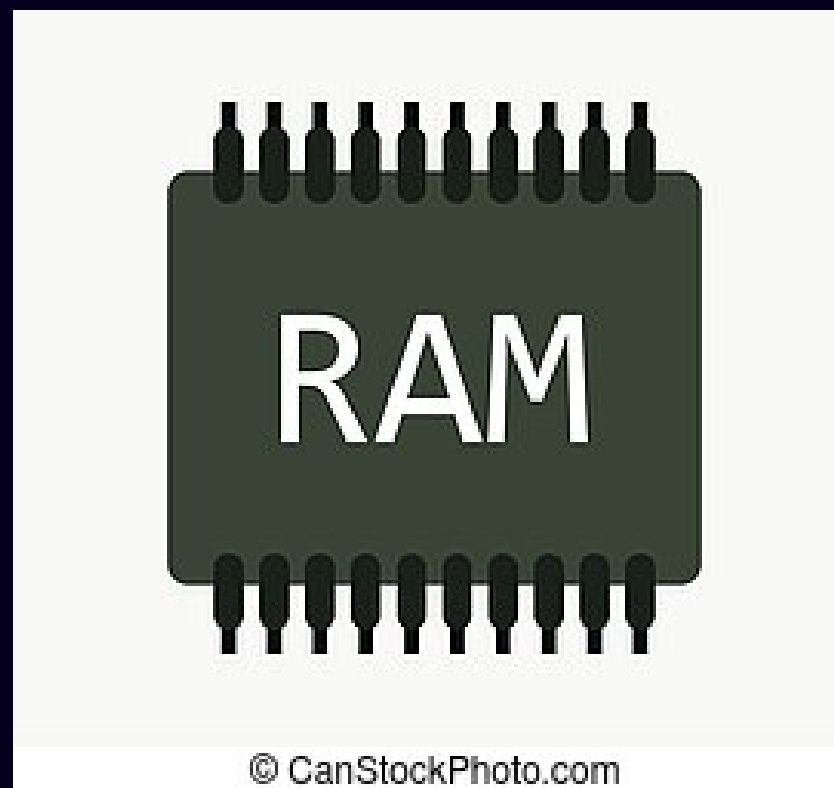


Simulador de Memória Virtual e Física

Marcia Gabrielle Oliveira, Shelly Leal, Wandressa Reis

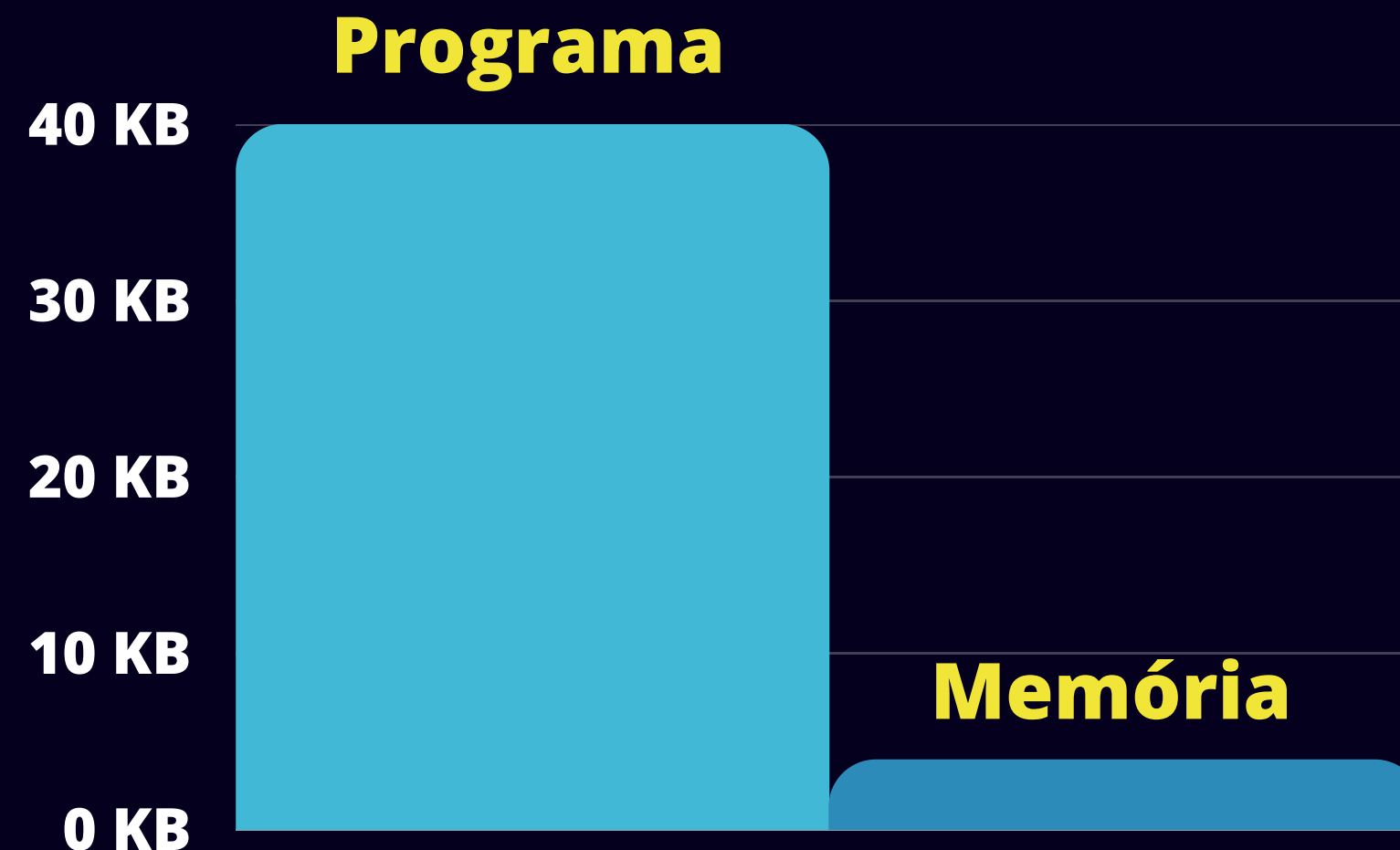
O que é Memória Física?

É um componente fundamental em um sistema de computador. Ela é responsável por armazenar dados e instruções que são acessados diretamente pelo processador.



Memória Virtual

Desde os primórdios da computação, é comum a existência de programas maiores que o tamanho de memória disponível;



O que é Memória Virtual?

- É uma técnica usada para o gerenciamento da memória
- Permite que o sistema operacional aloque espaço em disco para complementar a memória física
- Cria uma ilusão de que o sistema possui mais memória do que realmente tem.

Gerenciamento de Memória

Controla e organiza os recursos de memória, que inclui:

- **alocação**
- **desalocação**
- **compartilhamento de memória entre os processos em execução.**

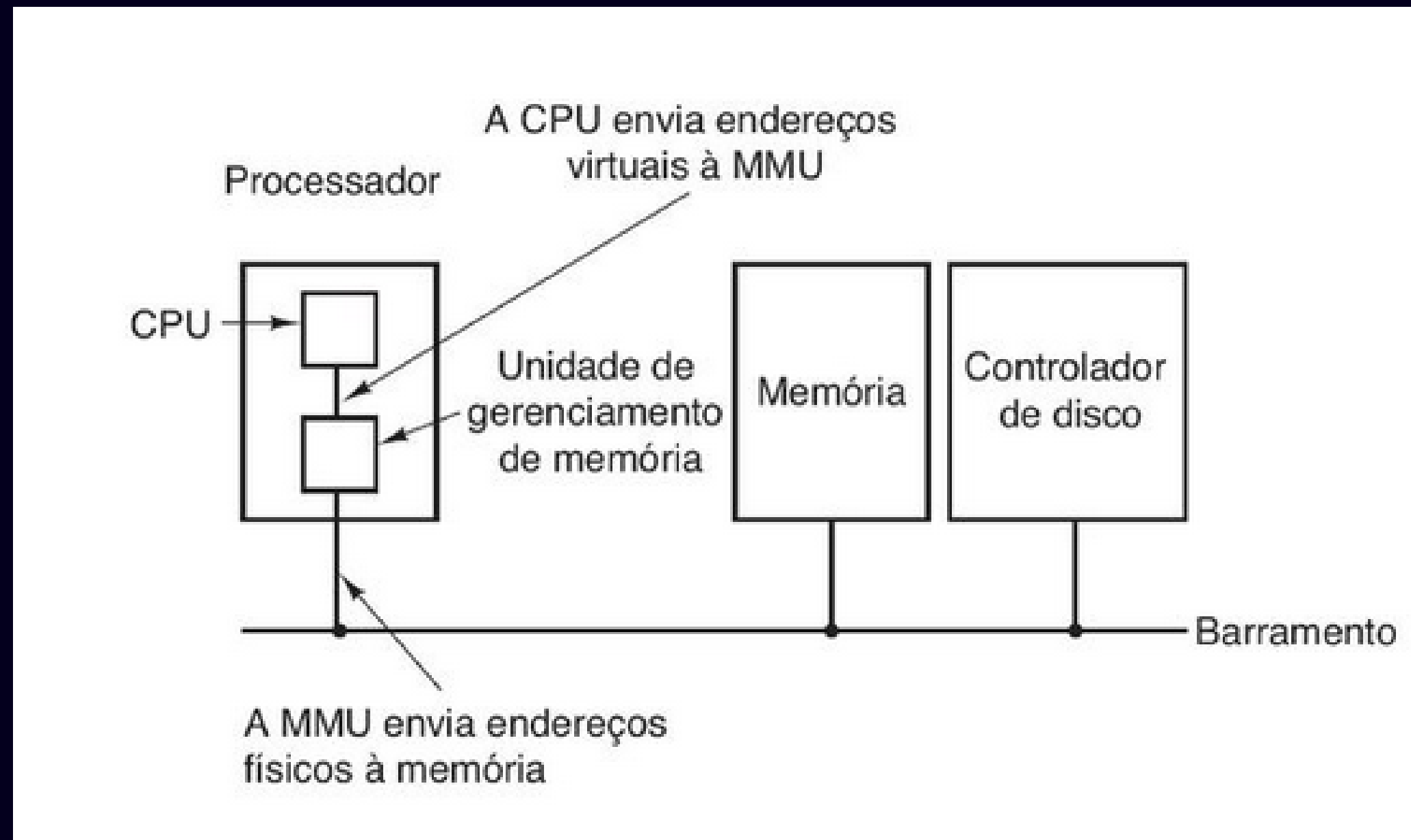
Paginação

- Sistemas operacionais modernos utilizam o conceito de paginação para implementar a memória virtual;
- Quando um programa executa a instrução:
 - **mov REG, 500**
- Ele deseja copiar o conteúdo do endereço de memória 500 para o registrador REG.

Paginação

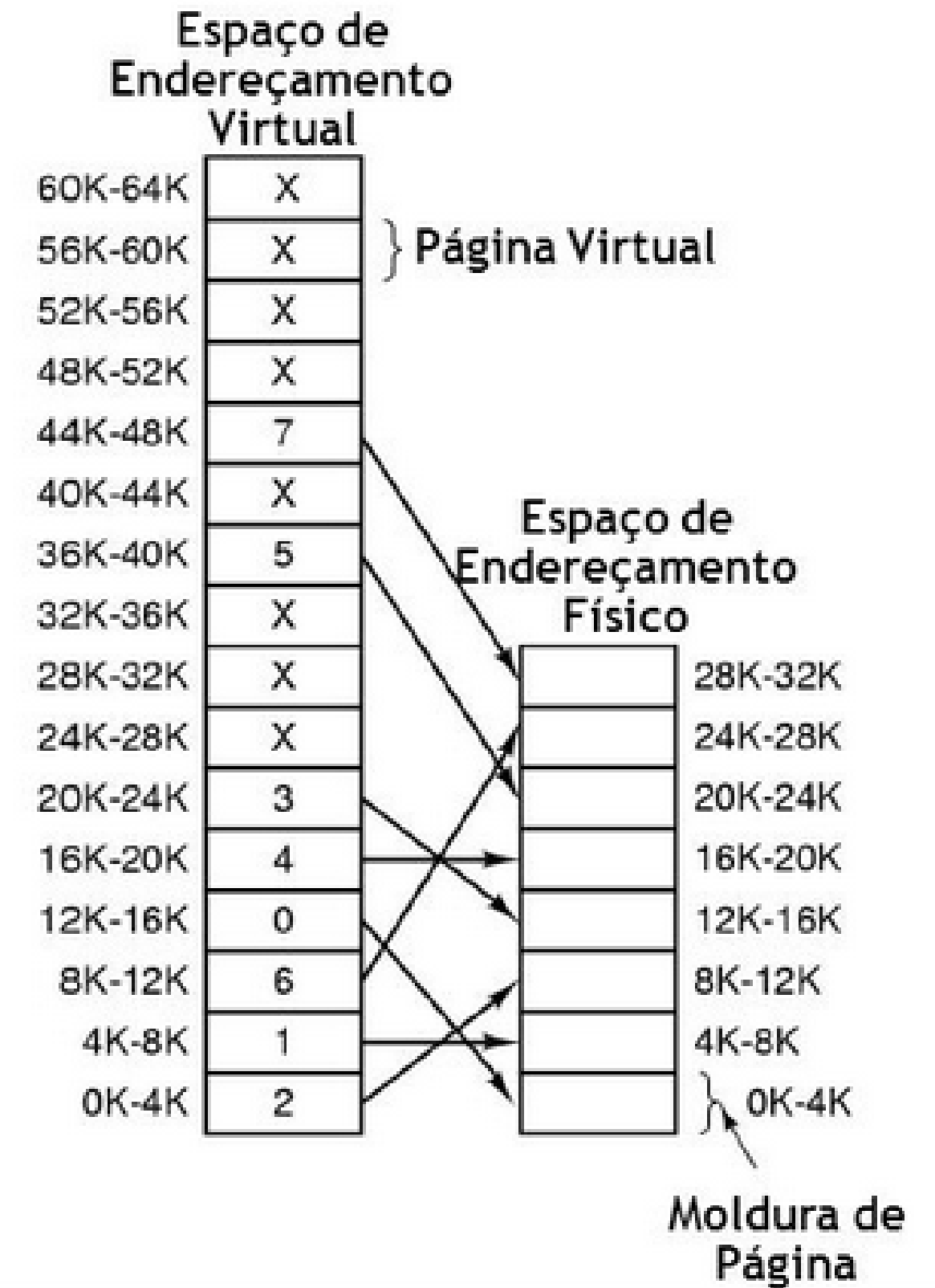
- Em computadores **sem memória virtual**, o endereço virtual é **idêntico ao endereço físico**;
- Em computadores **com memória virtual** é necessário um sistema **especial** para efetuar as conversões de endereço necessárias;
- Este módulo é denominado **MMU** (memory management unit) - **unidade de gerenciamento de memória**.

Paginação



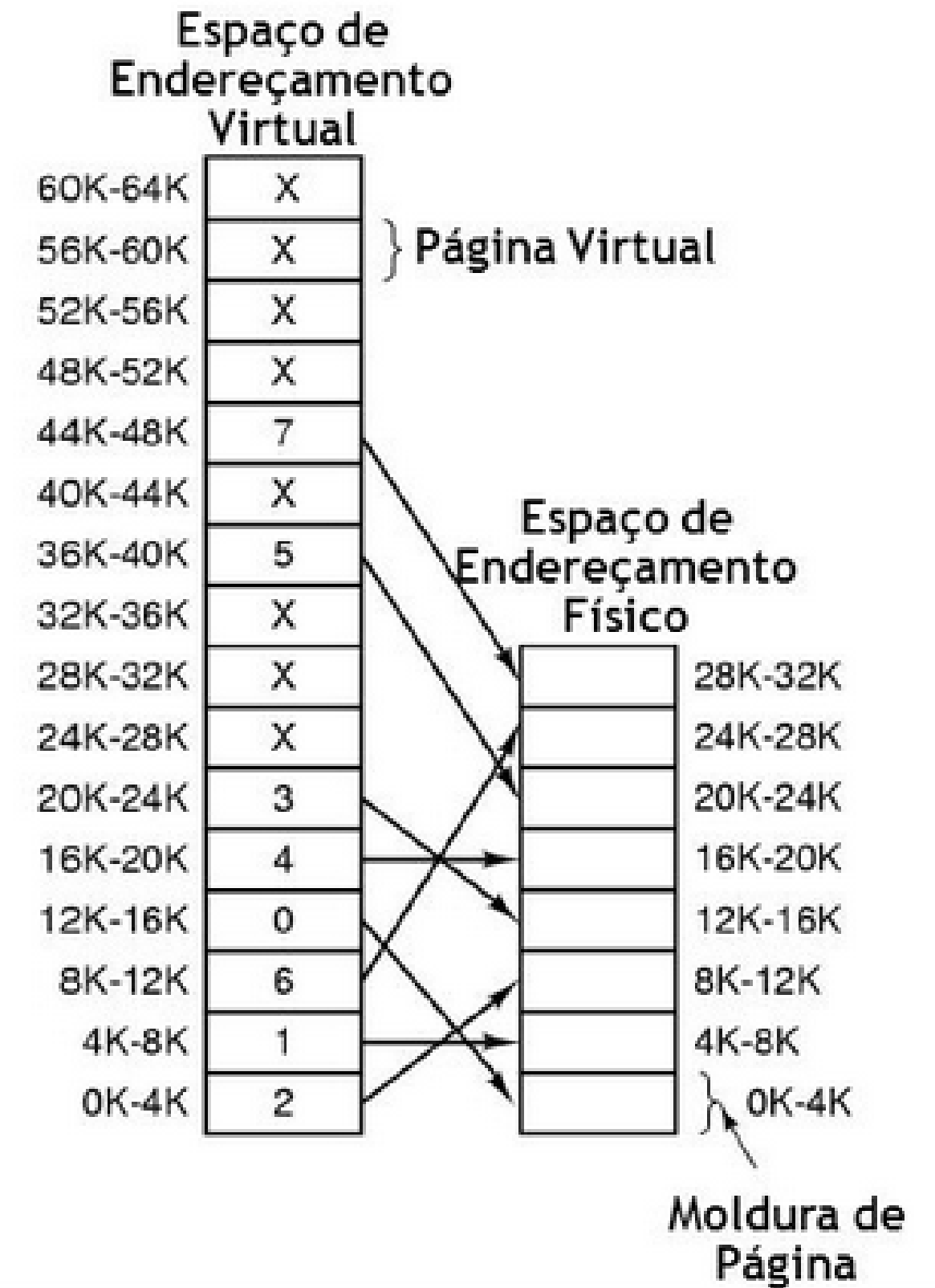
Conversão efetuada pela MMU

- A instrução **mov REG, 0** é convertida pela MMU para **mov REG, 8192;**
- O endereço 0 (virtual) pertence a página 2 da memória real.
 - Tamanho da página = 4k;
 - Endereço = $(1024 * 2 * 4)$



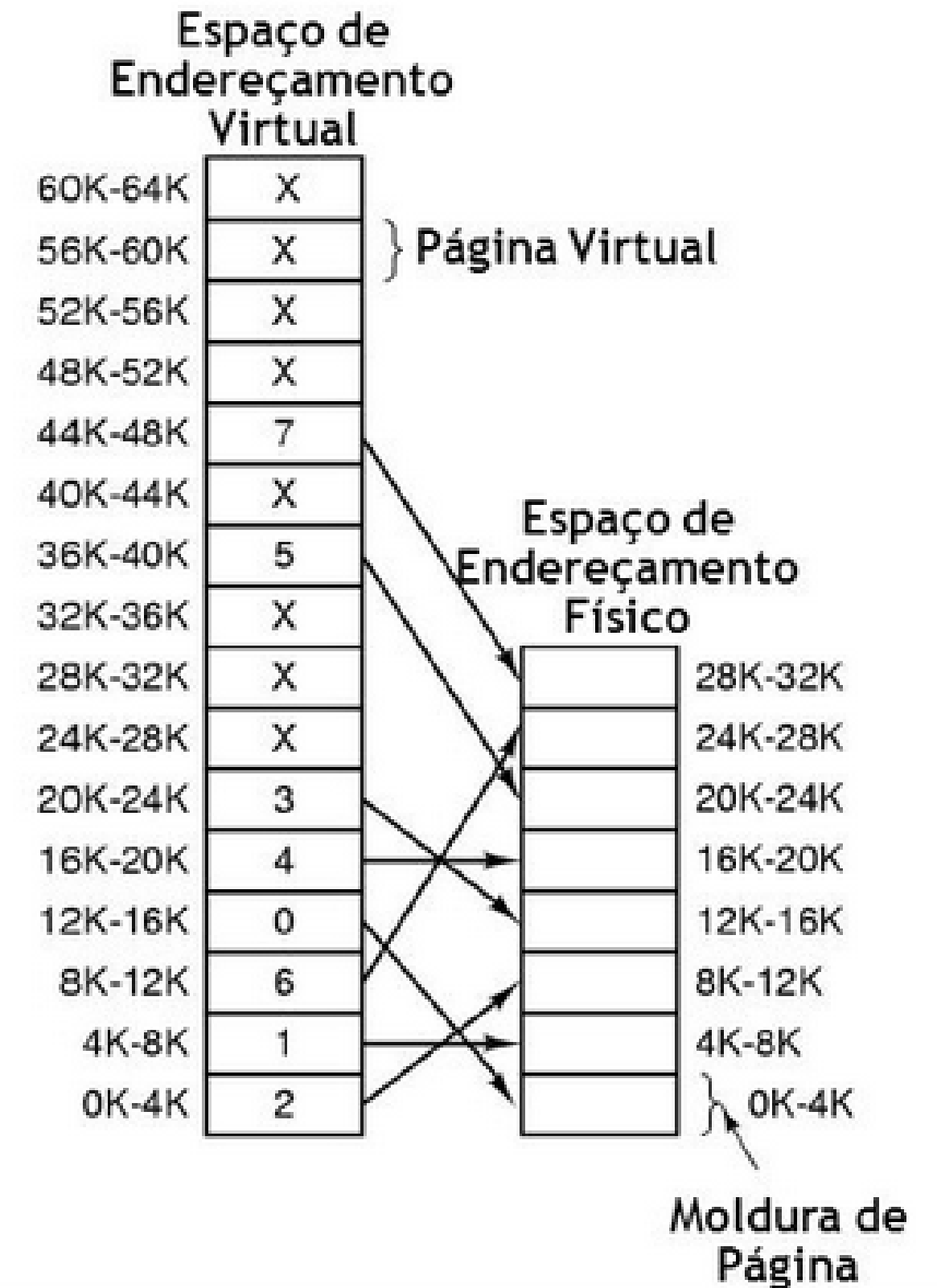
Conversão efetuada pela MMU

- **Limitação**
- **A memória virtual é maior do que a memória física primária;**
- **O que acontece ao invocarmos**
 - **mov REG, 32780?**
- **A MMU encontra um indicador "X". Isso significa que o conteúdo não está na memória principal. Está no disco.**



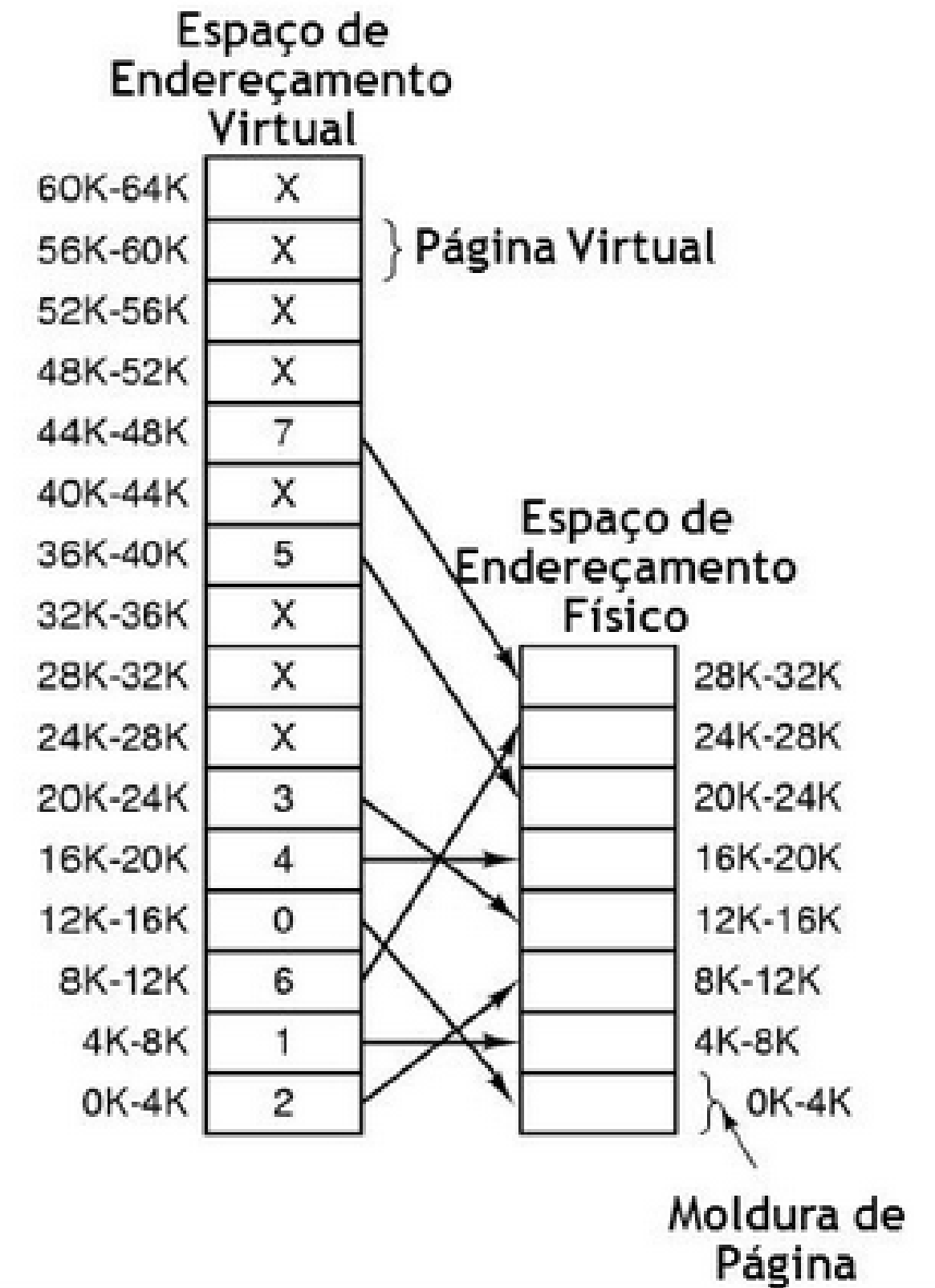
Page Fault

- O que a MMU deve fazer?
- A MMU deve trazer a página do disco para a memória primária;
- Mas todas as páginas da memória primária estão sendo referenciadas pela memória virtual;
- O que fazer?



Page Fault

- Neste caso, MMU deve executar um algoritmo para a **substituição de página**;
- Elimina uma das páginas da memória física (mas a armazena no disco) e traz a página requisitada para a posição liberada.
- **Problema:** Qual página vai ser eliminada da memória?



Troca de Páginas

Ocorre quando uma página precisa ser movida entre a memória física e o armazenamento em disco. Isso acontece quando a memória física está cheia e é necessário liberar espaço para carregar páginas adicionais necessárias.

Vantagens da Memória Virtual

- Permite que programas executem mesmo quando a quantidade de memória física é insuficiente, possibilitando a execução de aplicativos maiores do que a capacidade da RAM.
- Permite o compartilhamento de recursos entre processos, isolando-os uns dos outros.

Algoritmos de Substituição de Página

Código

```
7  // ESTRUTURA DA PÁGINA
8  typedef struct Page
9  {
10     char address[9];
11     struct Page *next;
12 } Page;
```

- A estrutura Page representa uma página da memória. Ela contém uma string address para armazenar o endereço da página e um ponteiro next para apontar para a próxima página na lista.

- Essa função é responsável por lidar com uma operação de escrita na memória. Ela verifica se há espaço disponível na memória para adicionar uma nova página ou se é necessário substituir uma página existente.

```
223 // FUNÇÃO WriteAddres
224 void WriteAddress(char value[9])
225 {
226     if (usedPages < numPages)
227     {
228         AddNewPage(tmpAddress);
229     }
230     else
231     {
232         faults++;
233         ReplacePage(tmpAddress);
234     }
235 }
```


LRU - Least Recently Used

A página que foi acessada há mais tempo é a menos provável de ser acessada novamente no futuro próximo. Portanto, a página que está há mais tempo na memória sem ter sido acessada será escolhida para ser substituída.

LRU - Least Recently Used

Page fault: Tentativa de acessar uma página que não está na Memória Física

LRU realiza as seguintes etapas:


- Verifica a página
- Escolha da página a ser substituída
- Remove a página escolhida
- Registro do histórico de Acesso

LRU

```
141 // Algoritmo LRU (Least Recently Used)
142 void LRU(char value[9])
143 {
144     // Adiciona a nova página
145     AddNewPage(value);
146     if (usedPages == numPages)
147         first = first->next;
148 }
```

```
173 // Procura por uma página na memória
174 bool Find(char value[9])
175 {
176     Page *tmp = first, *prev = NULL;
177     while (tmp != NULL)
178     {
179         if (strcmp(tmp->address, value) == 0)
180         {
181             if (strcmp(alg, "lru") == 0)
182             {
183                 if (prev != NULL)
184                 {
185                     if (tmp->next != NULL)
186                         prev->next = tmp->next;
187                 }
188                 else
189                 {
190                     first = first->next;
191                 }
192                 last->next = tmp;
193                 last = tmp;
194                 tmp->next = NULL;
195             }
196             return true;
197         }
198         prev = tmp;
199         tmp = tmp->next;
200     }
201     return false;
202 }
```

First In - First Out (FIFO)

	2	3	1	b	1	2	
PF 1	2						
PF 2							
PF 3							
PF 4							
PLS	2						
T	1	2	3	4	5	6	
							

A primeira que chega - a primeira que sai

Trata-se de uma fila que prioriza a substituição de uma página que está há mais tempo em memória.

Estratégia

Implementa uma lista encadeada, sendo que as páginas recém alocadas são inseridas no final e a página escolhida para a substituição é a do topo da lista.

Exemplo

PF (Página Física), PLS (Página Lógica a ser substituída)
T (Tempo).

First In - First Out (FIFO)

	2	3	1	b	1	2	
PF 1	2	2					
PF 2		3					
PF 3							
PF 4							
PLS	2	2					
T	1	2	3	4	5	6	
	×	×					

A primeira que chega - a primeira que sai

Trata-se de uma fila que prioriza a substituição de uma página que está há mais tempo em memória.

Estratégia

Implementa uma lista encadeada, sendo que as páginas recém alocadas são inseridas no final e a página escolhida para a substituição é a do topo da lista.

Exemplo

PF (Página Física), PLS (Página Lógica a ser substituída)
T (Tempo).

First In - First Out (FIFO)

	2	3	1	b	1	2	
PF 1	2	2	2				
PF 2		3	3				
PF 3			1				
PF 4							
PLS	2	2	2				
T	1	2	3	4	5	6	
	×	×	×				

A primeira que chega - a primeira que sai

Trata-se de uma fila que prioriza a substituição de uma página que está há mais tempo em memória.

Estratégia

Implementa uma lista encadeada, sendo que as páginas recém alocadas são inseridas no final e a página escolhida para a substituição é a do topo da lista.

Exemplo

PF (Página Física), PLS (Página Lógica a ser substituída)
T (Tempo).

First In - First Out (FIFO)

	2	3	1	b	1	2	
PF 1	2	2	2	2			
PF 2		3	3	3			
PF 3			1	1			
PF 4				b			
PLS	2	2	2	2			
T	1	2	3	4	5	6	
	×	×	×	×			

A primeira que chega - a primeira que sai

Trata-se de uma fila que prioriza a substituição de uma página que está há mais tempo em memória.

Estratégia

Implementa uma lista encadeada, sendo que as páginas recém alocadas são inseridas no final e a página escolhida para a substituição é a do topo da lista.

Exemplo

PF (Página Física), PLS (Página Lógica a ser substituída)
T (Tempo).

First In - First Out (FIFO)

A primeira que chega - a primeira que sai

Trata-se de uma fila que prioriza a substituição de uma página que está há mais tempo em memória.

Estratégia

Implementa uma lista encadeada, sendo que as páginas recém alocadas são inseridas no final e a página escolhida para a substituição é a do topo da lista.

Exemplo

PF (Página Física), PLS (Página Lógica a ser substituída)
T (Tempo).

	2	3	1	b	1	2	
PF 1	2	2	2	2	1		
PF 2		3	3	3	3		
PF 3			1	1	1		
PF 4				b	b		
PLS	2	2	2	2	2		
T	1	2	3	4	5	6	
	✗	✗	✗	✗	✓		

First In - First Out (FIFO)

A primeira que chega - a primeira que sai

Trata-se de uma fila que prioriza a substituição de uma página que está há mais tempo em memória.

Estratégia

Implementa uma lista encadeada, sendo que as páginas recém alocadas são inseridas no final e a página escolhida para a substituição é a do topo da lista.

Exemplo

PF (Página Física), PLS (Página Lógica a ser substituída)
T (Tempo).

	2	3	1	b	1	2	
PF 1	2	2	2	2	1	1	
PF 2		3	3	3	3	2	
PF 3			1	1	1	1	
PF 4				b	b	b	
PLS	2	2	2	2	2	3	
T	1	2	3	4	5	6	
	✗	✗	✗	✗	✓	✓	

FIFO

```
150 // Algoritmo FIFO (First-In-First-Out)
151 void FIFO(char value[9])
152 {
153     // Adiciona a nova página
154     AddNewPage(value);
155     if (usedPages == numPages)
156         first = first->next;
157 }
```

```
173 // Procura por uma página na memória
174 bool Find(char value[9])
175 {
176     Page *tmp = first, *prev = NULL;
177     while (tmp != NULL)
178     {
179         if (strcmp(tmp->address, value) == 0)
180         {
181             if (strcmp(alg, "lru") == 0)
182             {
183                 if (prev != NULL)
184                 {
185                     if (tmp->next != NULL)
186                         prev->next = tmp->next;
187                 }
188                 else
189                 {
190                     first = first->next;
191                 }
192                 last->next = tmp;
193                 last = tmp;
194                 tmp->next = NULL;
195             }
196             return true;
197         }
198         prev = tmp;
199         tmp = tmp->next;
200     }
201     return false;
202 }
```

RANDOM

```
159 // Algoritmo Random (Aleatório)
160 void Random(char value[9])
161 {
162     writes++;
163     srand(time(NULL));
164     int index = rand() % usedPages;
165     Page *tmp = first;
166     for (int i = 0; i < index; i++)
167     {
168         tmp = tmp->next;
169     }
170     strcpy(tmp->address, value);
171 }
```

Situações em que ele pode ser utilizado:

- Simplicidade
- Avaliação inicial



RESULTADOS DOS TESTES

LRU

```
Executando o simulador...  
Arquivo de entrada: arquivo.log  
Tamanho da memoria: 128KB  
Tamanho das paginas: 4KB  
Tecnica de reposicao: lru  
Paginas lidas:685182  
Paginas escritas:685214
```

FIFO

```
Executando o simulador...  
Arquivo de entrada: arquivo.log  
Tamanho da memoria: 128KB  
Tamanho das paginas: 4KB  
Tecnica de reposicao: fifo  
Paginas lidas:689877  
Paginas escritas:689909
```

RESULTADOS DOS TESTES

RANDOM

```
Executando o simulador...  
Arquivo de entrada: arquivo.log  
Tamanho da memoria: 128KB  
Tamanho das paginas: 4KB  
Tecnica de reposicao: random  
Paginas lidas:900867  
Paginas escritas:900899
```

COMPARAÇÃO DOS ALGORITMOS

FIFO

Fácil de implementar e requer menos overhead computacional, mas pode resultar em subutilização da memória quando há páginas frequentemente acessadas.

LRU

Tende a ter um melhor desempenho em geral, pois leva em consideração a atividade recente, mas pode exigir um pouco mais de processamento para manter o registro das páginas acessadas.

RANDOM

Pode acontecer de uma página recentemente acessada ser substituída novamente ou que uma página pouco utilizada seja mantida na memória por um longo período de tempo.

OBRIGADA!