Para determinar a qual página um programa está se referindo, uma parte dos *bits* que compõem o endereço é interpretada como um número de página, e a outra parte, como o número do *byte* dentro da página. Por exemplo, se o espaço de endereçamento possui 24 *bits*, então a memória virtual é de 2^{24} *bytes*; se o tamanho da página é de 512 *bytes* (2^9), então 9 *bits* são utilizados para representar o número do *byte* dentro da página e os 15 *bits* restantes são utilizados para representar o número da página.

O mapeamento de endereços a partir do espaço de endereçamento (número da página mais número do *byte*) para o espaço de memória (localização física da memória) é realizado por meio de uma Tabela de Páginas, cuja p-ésima entrada contém a localização $p\prime$ da Moldura de Página contendo a página número p, desde que esteja na memória principal (a possibilidade de que p não esteja na memória principal será tratada mais à frente). Logo, o mapeamento de endereços é:

$$f(e) = f(p, b) = p\prime + b,$$

em que o endereço de programa e (número da página p e número do *byte* b) pode ser visto na Figura 6.1.

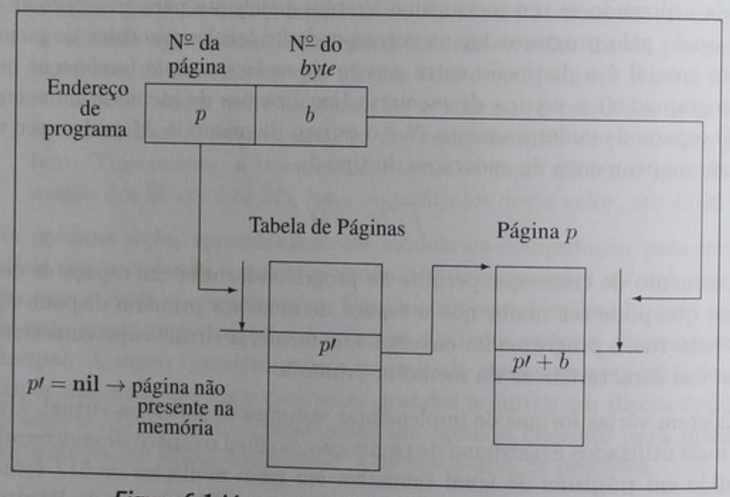

Figura 6.1 *Mapeamento de endereços para paginação.*

A Tabela de Páginas pode ser um arranjo do tamanho do número de páginas possíveis. Quando acontecer de o programa endereçar um número de página que não esteja na memória principal, a entrada correspondente estará vazia ($p\prime$ = nil) na Tabela de Páginas e a página correspondente terá de ser trazida da memória secundária para a memória primária, atualizando a Tabela de Páginas.

Se não existir uma Moldura de Página vazia no momento de trazer uma nova página do disco, então alguma outra página tem de ser removida da memória

principal para abrir espaço para a nova página. O ideal é remover a página que não será referenciada pelo período de tempo mais longo no futuro. Entretanto, não há meios de prever o futuro. O que normalmente é feito é tentar inferir o futuro a partir do comportamento passado. Existem vários algoritmos propostos na literatura para a escolha da página a ser removida. Os mais comuns são:

- Menos Recentemente Utilizada (LRU). Um dos algoritmos mais utilizados é o LRU (*Least Recently Used*), o qual remove a página menos recentemente utilizada, partindo do princípio de que o comportamento futuro deve seguir o passado recente. Nesse caso, temos de registrar a sequência de acesso a todas as páginas.

  Uma forma possível de implementar a política LRU para sistemas paginados é pelo uso de uma fila de Molduras de Páginas, conforme ilustrado na Figura 6.2. Toda vez que uma página é utilizada (para leitura apenas, para leitura e escrita ou para escrita apenas), ela é removida para o **fim** da fila (o que implica a alteração de cinco apontadores). A página que está na moldura do **início** da fila é a página LRU. Quando uma nova página tem de ser trazida da memória secundária, ela deve ser colocada na moldura que contém a página LRU.

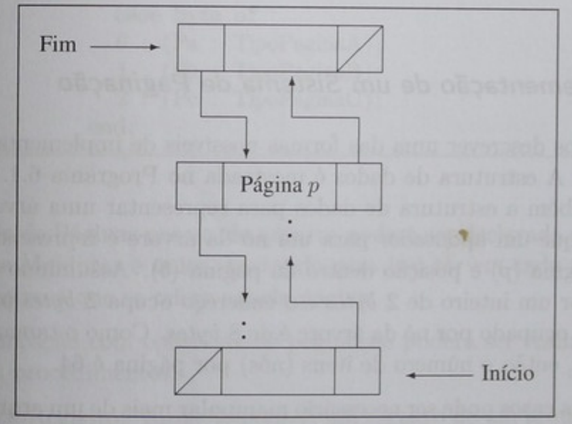

Figura 6.2 *Fila de Molduras de Páginas.*

- Menos Frequentemente Utilizada (LFU). O algoritmo LFU (*Least Frequently Used*) remove a página menos frequentemente utilizada. A justificativa é semelhante ao caso anterior, e o custo é o de registrar o número de acessos a todas as páginas. Um inconveniente é que uma página recentemente trazida da memória secundária tem um baixo número de acessos registrados e, por isso, pode ser removida.
- Ordem de Chegada (FIFO). O algoritmo FIFO (*First In First Out*) remove a página que está residente há mais tempo. Esse algoritmo é o mais