INTRODUÇÃO À SISTEMAS DE INFORMAÇÃO

AULA 9: CONCEITOS E GERENCIAMENTO DE MEMÓRIA

PROFª: LEONARA BRAZ

LEONARABRAZ@GEMAIL.COM

GERÊNCIA DE MEMÓRIA

- O Gerente de Memória é um componente do Sistema Operacional que aloca memória principal para os processos, e gerencia a hierarquia de memória
  - Caches, RAM, e Disco
- Quais as tarefas do Gerente de memória?

GERÊNCIA DE MEMÓRIA

- **Tarefas do gerente de memória:**
  - Garante isolamento mútuo entre processos (proteção)
  - Mantém o registro das áreas de memória em uso (e memória livre)
  - Aloca memória RAM para novos processos
  - Faz o swapping transparente entre memória principal e disco
  - Mantém o mapeamento de memória virtual para memoria física
  - Implementa a política de alocação de memória para os processos

GERÊNCIA DE MEMÓRIA

- O ideal seria ter memória infinitamente grande, com acesso infinitamente rápido, não-volátil e de baixo custo

GERENCIAMENTO DE MEMÓRIA

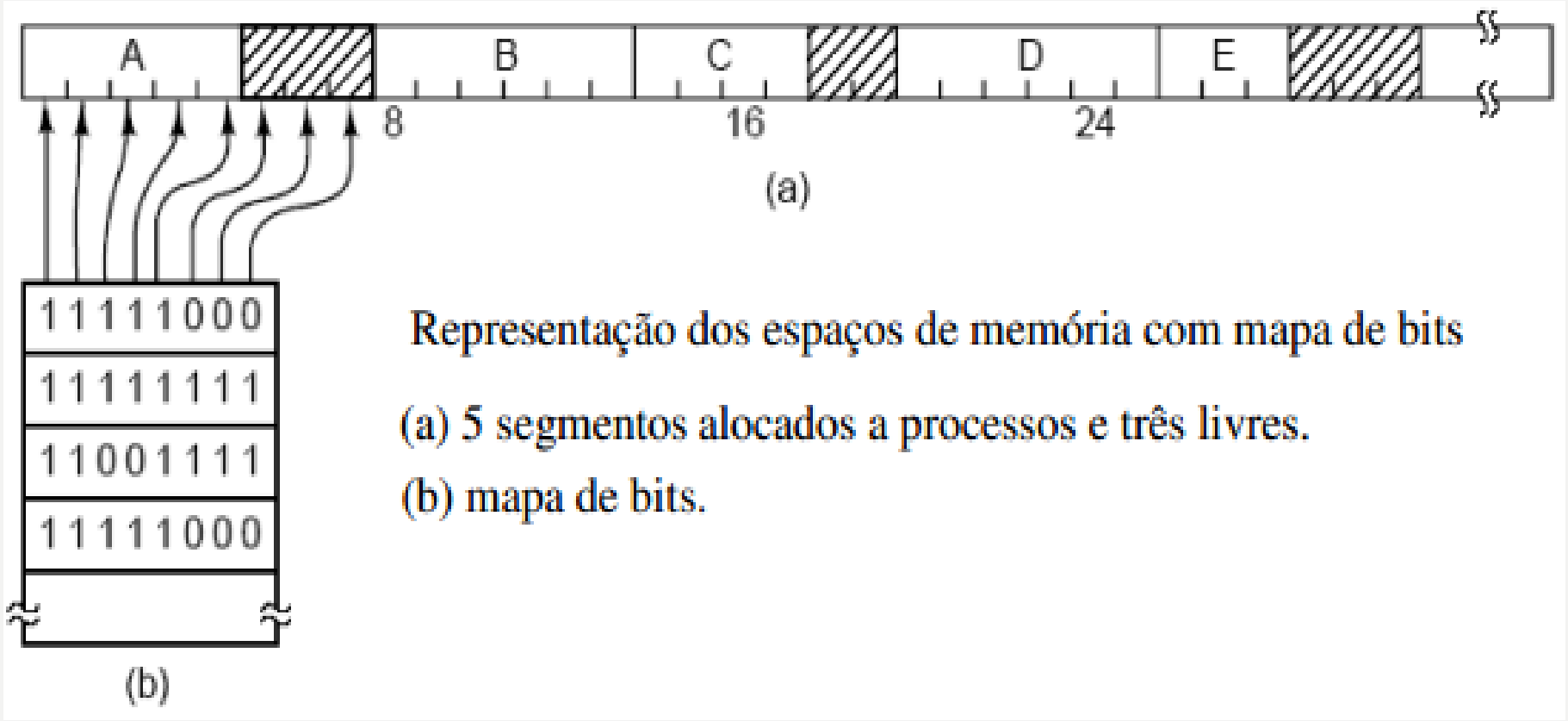
- Quando a memória é atribuída dinamicamente, o sistema operacional deve gerenciá-la
  - De modo geral, há dois tipos de verificar a utilização da memória:
    - **Gerenciamento de memória com mapa de bits**
    - **Gerenciamento de memória com listas encadeadas**

GERENCIAMENTO DE MEMÓRIA – COM MAPA DE BITS

- Com um mapa de bits, a memória é dividida em unidades de alocação, desde um pequeno número de palavras até muitos Kbytes.

- Para cada unidade de alocação existe um bit no mapa de bits, que é 0 se a unidade estiver livre e 1 caso esteja ocupada (ou vice-versa).

GERENCIAMENTO DE MEMÓRIA – COM MAPA DE BITS



Representação dos espaços de memória com mapa de bits

(a) 5 segmentos alocados a processos e três livres.

(b) mapa de bits.

GERENCIAMENTO DE MEMÓRIA – COM MAPA DE BITS

- O tamanho de cada unidade de alocação é uma importante característica de projeto.
  - Quanto menor for a unidade de alocação, maior será o mapa de bits
  - Se a unidade de alocação for grande, o mapa de bits será pequeno
    - Mas memória considerável pode ser desperdiçada se o tamanho do processo não for um múltiplo exato da unidade de alocação.

GERENCIAMENTO DE MEMÓRIA – COM MAPA DE BITS

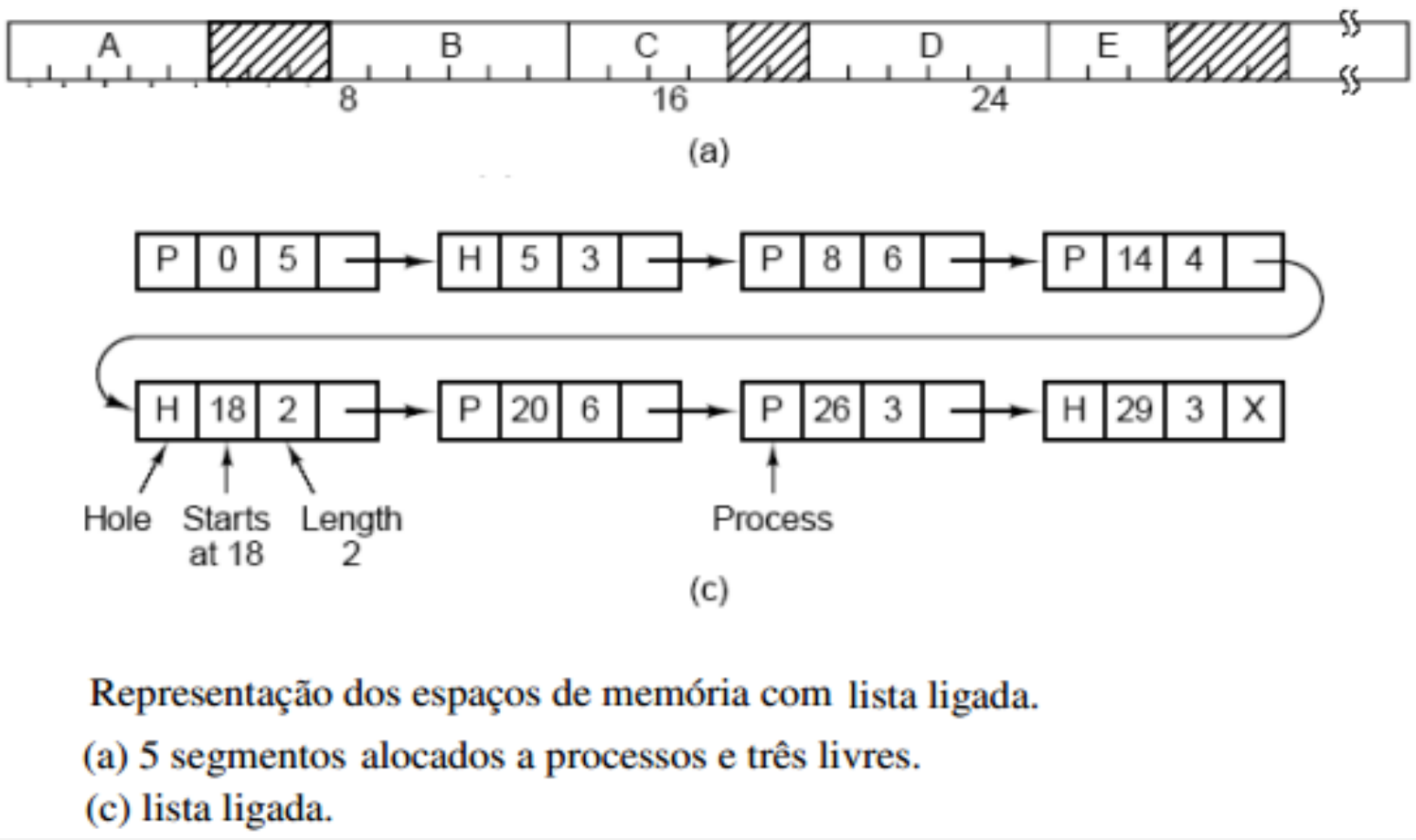
- Um mapa de bits provê uma maneira simples de gerenciar memória, uma vez que o tamanho do mapa de bits depende somente do tamanho da memória e do tamanho da unidade de alocação.
- O maior problema com os mapas de bits é que procurar uma lacuna (sequência de 0s) suficientemente grande para um determinado processo
- A desvantagens se mostra quando um processo necessita de k unidades de alocação, pois o gerenciador de memória deve encontrar uma sequência de k bits 0, o que se constitui um processo lento.

GERENCIAMENTO DE MEMÓRIA – COM LISTAS ENCADEADAS

- Outra maneira de gerenciar a memória é manter uma lista de alocações e segmentos de memória livre
  - A lista mantêm, em cada entrada, o endereço em que inicia, o seu comprimento e, evidentemente, o ponteiro para a próxima entrada.

- A principal vantagem de utilizar uma lista encadeada classificada por endereço é que sua atualização é simples e direta.

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA



Representação dos espaços de memória com lista ligada.
(a) 5 segmentos alocados a processos e três livres.
(c) lista ligada.

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- Vários algoritmos podem ser usados para alocar memória, a fim de criar ou permutar processos.

- Tais algoritmos são evocados quando o gerenciador de memória necessita um segmento de memória
  - **First-fit (Primeiro ajuste)**
  - **Next-fit (Próximo ajuste)**
  - **Best-fit (melhor ajuste)**
  - **Wors-fit (pior ajuste)**
  - **Quick-fit (Ajuste rápido)**

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- **First-fit**
  - É o algoritmo mais simples.
  - O algoritmo procura ao longo da lista de segmentos até encontrar um espaço livre de tamanho maior ou igual a M.
  - Caso o espaço livre tenha tamanho superior a M (por exemplo, tamanho N), o espaço livre é quebrado em dois segmentos:
    - Um para o processo (de tamanho M)
    - E o outro para a memória não usada (de tamanho N - M).
  - É um algoritmo rápido pois finaliza a busca o mais cedo possível.

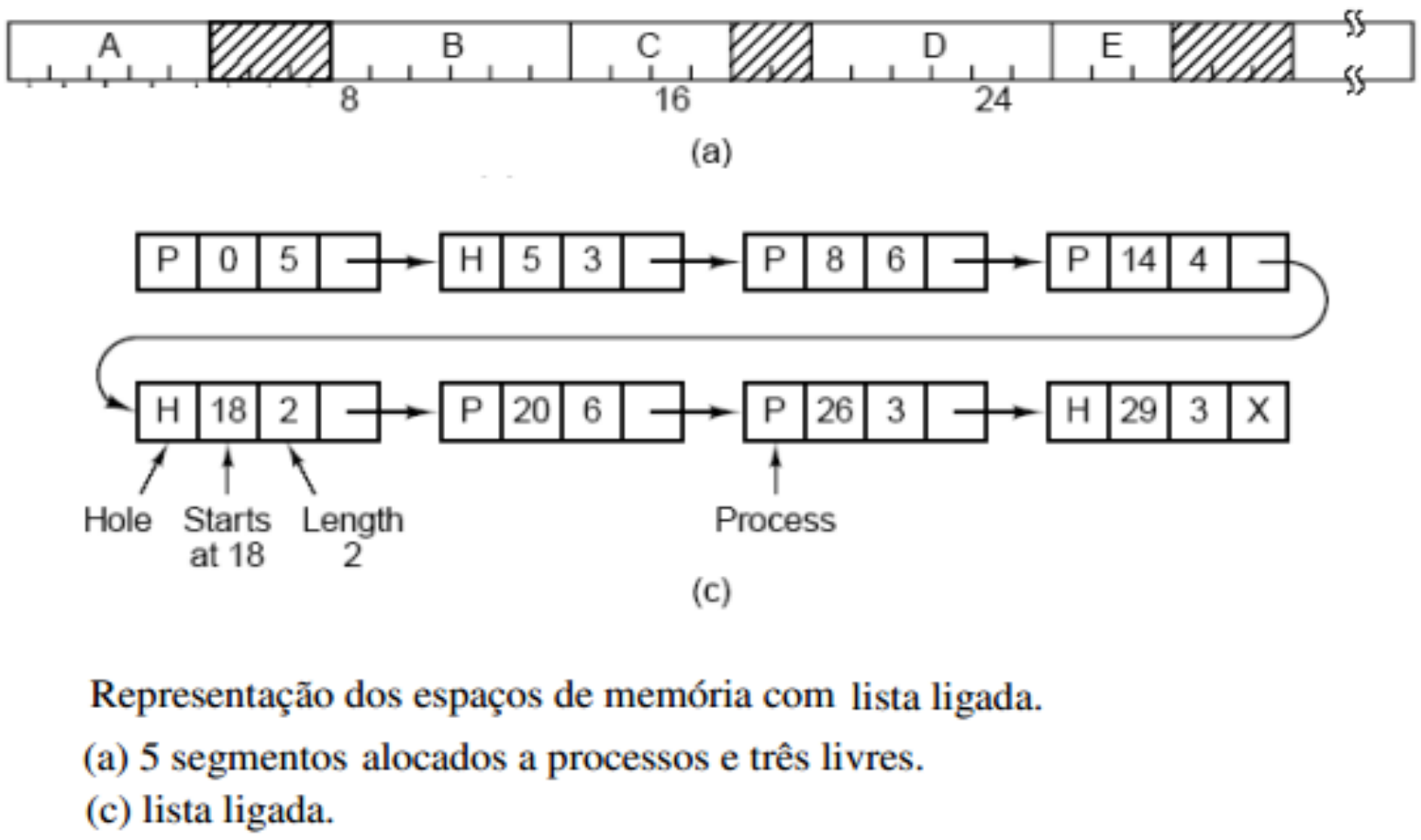
GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- **Next-fit**
  - Este algoritmo opera da mesma forma que o first-fit
    - Exceto que guarda a posição da lista onde o último espaço livre foi alocado.
    - Da próxima vez que é chamado, o algoritmo começa a procurar a partir deste ponto.

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- **Best-fit**
  - Este algoritmo procura pela lista inteira e toma o espaço livre de tamanho mais próximo de M.
  - É um algoritmo lento e cria na memória espaços livres pequenos que dificilmente serão alocados.
  - Entretanto, para M grande, best-fit aumenta as chances de se encontrar na lista um espaço livre de tamanho adequado, posto que minimiza o uso espaços livres grandes para atender requisições pequenas.

BEST-FIT X FIRST-FIT



Representação dos espaços de memória com lista ligada.
(a) 5 segmentos alocados a processos e três livres.
(c) lista ligada.

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- **Wors-fit**
  - Oposto do Best-fit
    - Varre a lista completamente e aloca no espaço que gerar a maior lacuna de memória disponível.
    - Quando dividido, o segmento de memória disponível restante seja suficientemente grande para ser útil depois

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- **Quick-fit**
  - Este algoritmo mantêm listas separadas para tamanhos comumente requeridos.
  - Por exemplo, seja uma tabela com n entradas, na qual a primeira é um ponteiro para a cabeça da lista de espaços livres de tamanho 4K, a segunda é um ponteiro para a cabeça da lista de espaços livres de tamanho 8K, a terceira de tamanho 12K, e assim sucessivamente
  - Com o quick-fit, acha-se um espaço livre de tamanho requerido muito rapidamente, mas com a desvantagem de todos os esquemas de classificar os espaços livres por tamanho, a saber, quando um processo termina ou é permutado para disco

GERENCIAMENTO DE MEMÓRIA – COM LISTA ENCADEADA

- Todos os algoritmos podem aumentar seus respectivos desempenhos mantendo-se em separado listas para processos e espaços livres.
- Neste caso, todos devotam suas energias para inspeção de espaços livres, não de processos.
- O preço pago por esse aumento de velocidade na alocação é uma complexidade adicional e diminuição de velocidade quando se trata de liberar memória
  - Uma vez que um segmento livre tem de ser removido da lista de processos e inserido na lista de espaços livres.

SEÇÃO 2

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- Quando ocorre uma falta de página, o sistema operacional precisa escolher uma página a ser removida da memória, a fim de liberar espaço para uma nova página a ser trazida para a memória

- Se a página a ser removida tiver sido modificada enquanto estava na memória, ela deverá ser reescrita no disco com o propósito de atualizar a cópia lá existente

- Se a página não tiver sido modificada, a cópia em disco já está atualizada, não sendo necessário reescrevê-la

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **PERGUNTA!**
  - Como escolher uma página para descartar, quando ocorre falta de página?

- Embora seja possível escolher aleatoriamente uma página a ser descartada a cada falta de página, o desempenho do sistema será muito melhor se a página escolhida for uma que não estiver sendo muito usada
  - Se uma página intensamente usada for removida, é provável que logo ela precise ser trazida de volta

ALGORITMO ÓTIMO

- O melhor algoritmo de substituição de página é fácil de descrever, mas impossível de implementar

- Como o algoritmo ótimo funciona?
  - No momento que ocorre uma falta de página, existe um determinado conjunto de páginas na memória
  - Cada página pode ser rotulada com o número de instruções que serão executadas antes de aquela página ser referenciada pela primeira vez
  - O algoritmo ótimo diz que se deve remover a página com o maior rótulo

ALGORITMO ÓTIMO

- **Exemplo:**
  - Se determinada página só for usada após oito milhões de instruções e outra página só for usada após seis milhões, a primeira deve ser removida antes da segunda

- Dessa maneira, o algoritmo ótimo de substituição adia a ocorrência da próxima falta de página o máximo possível

ALGORITMO ÓTIMO

- **PROBLEMA:**
  - Este algoritmo é irrealizável!
  - Na ocorrência de uma falta de página, o sistema operacional não tem como saber quando cada uma das páginas será referenciada novamente

ALGORITMO NÃO USADA RECENTEMENTE

- A maioria dos computadores com memória virtual tem dois bits de status: o bit ***referenciado (R)*** e o bit ***modificado (M)***
  - Estes bits estão associados a cada página virtual, permitindo ao sistema operacional saber quais páginas físicas estão sendo usadas e quais não estão
- O bit R é colocado em 1 sempre que a página é referenciada (lida ou escrita)
- O bit M é colocada em 1 sempre que se modifica a página (escrita)

ALGORITMO NÃO USADA RECENTEMENTE

- Tais bits devem ser atualizados em todas as referências à memória.

- Os bits R e M podem ser usados para construir um algoritmo de paginação simples
  - Quando um processo é inicializado, os dois bits, são colocados em 0 pelo sistema operacional
  - Periodicamente, o bit R é limpo, de modo que diferencie as páginas que não foram referenciadas recentemente daquelas que foram

ALGORITMO NÃO USADA RECENTEMENTE

- Quando acontece uma falta de página, o sistema operacional inspeciona todas as páginas e as separa em quatro categorias, com base nos valores atuais dos bits R e M
  - **Classe 0:** não referenciada, não modificada
  - **Classe 1:** não referenciada, modificada
  - **Classe 2:** referenciada, não modificada
  - **Classe 3:** referenciada, modificada

ALGORITMO NÃO USADA RECENTEMENTE

- O algoritmo NRU (*Not Recently Used*) remove aleatoriamente uma classe de ordem mais baixa que não esteja vazia
  - Neste algoritmo, é melhor remover uma página modificada mas não referenciada do que uma página não modificada que está sendo **instantaneamente referenciada**

- A principal vantagem deste algoritmo é sua fácil compreensão e implementação
  - Adicionalmente, este algoritmo fornece um desempenho adequado

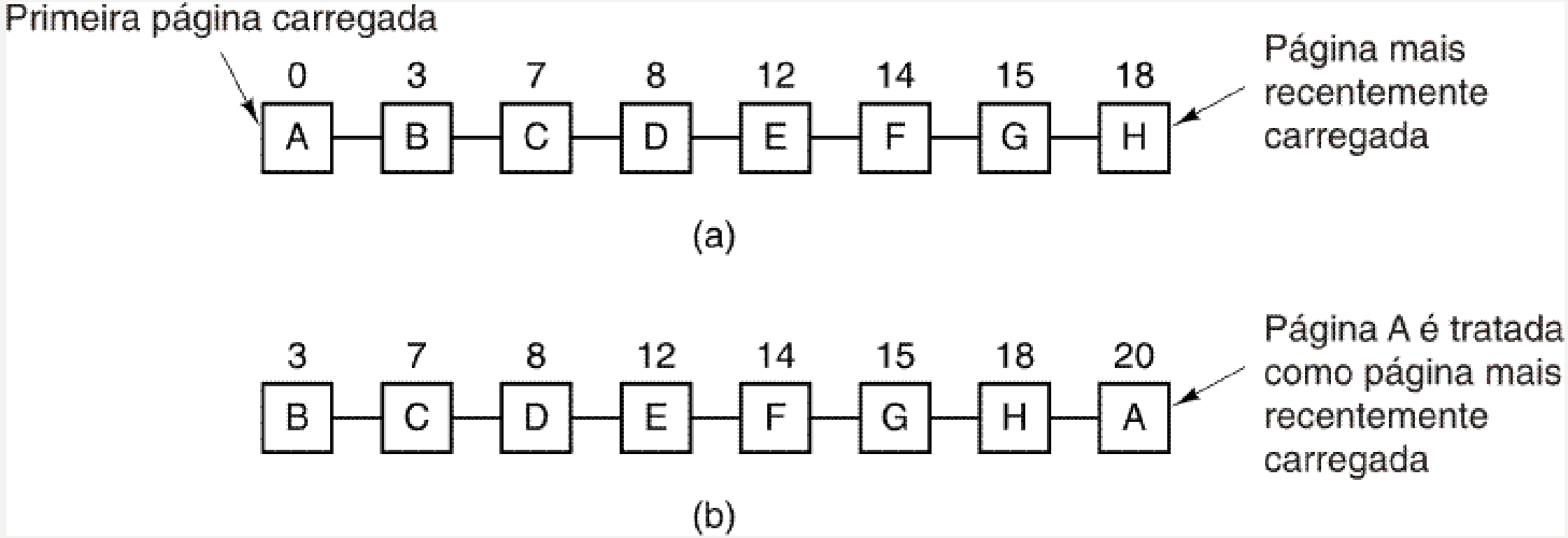
ALGORITMO PRIMEIRO A ENTRAR – PRIMEIRO A SAIR

- O algoritmo Primeiro a Entrar, Primeiro a Sair (first in, first out - FIFO) tem por característica seu baixo custo
  - O sistema operacional mantém uma lista de todas as páginas atualmente na memória, ordenando por ordem de chegada (das mais antigas às mais recentes)
  - Na ocorrência de uma falta de página a primeira página da lista é removida e a nova página é adicionada no final da lista
  - *No entanto, pode estar sendo removida uma página bastante utilizada*

ALGORITMO SEGUNDA CHANCE

- Uma modificação simples no algoritmo FIFO evita o problema de se jogar fora uma página intensamente usada
  - Isso é feito inspecionando o bit R da página mais antiga
  - Se o bit R for 0
    - Essa página, além de ser a mais antiga não está sendo usada, de modo que será substituída imediatamente
  - Se o bit R for 1
    - Ele será colocado em 0
    - A página será posta no final da lista de páginas
    - Seu tempo de carregamento (chegada) será atualizado

ALGORITMO SEGUNDA CHANCE



ALGORITMO SEGUNDA CHANCE

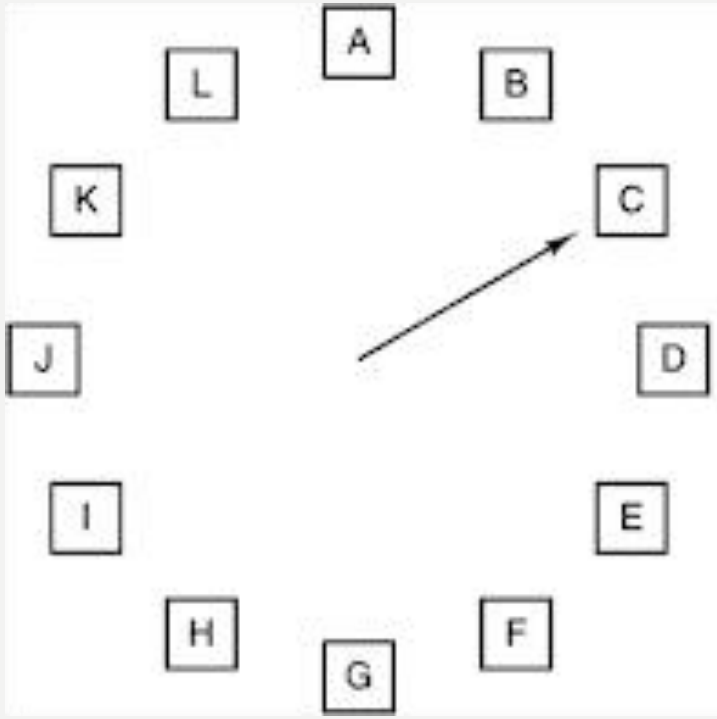
- O que este algoritmo faz é procurar uma página antiga que não tenha sido referenciada no intervalo de relógio anterior

- **PERGUNTA!**
  - O que acontece se todas as páginas foram referenciadas?
  - **Resposta:** o algoritmo degenera-se para o FIFO

ALGORITMO DO RELÓGIO

- Embora o algoritmo de segunda chance seja razoável ele é desnecessariamente ineficaz
  - Pois permanece constantemente reinserindo páginas no final da lista
- Uma estratégia melhor é manter todas as páginas em uma lista circular em forma de relógio
  - Um ponteiro aponta para a página mais antiga, sinalizando a "cabeça" da lista

ALGORITMO DO RELÓGIO



Quando ocorre uma falta de página,
a página apontada é examinada.
A atitude a ser tomada depende do bit R:
R = 0: Retira a página,
R = 1: Faz R = 0 e avança o ponteiro.

ALGORITMO DO RELÓGIO

- Quando ocorre uma falta de página, a página indicada pelo ponteiro é examinada
  - Se o bit R for 0
    - A página é removida
    - A nova página é inserida em seu lugar
    - O ponteiro avança uma posição
  - Se o bit R for 1
    - Ele é zerado
    - O ponteiro avança para próxima página

ALGORITMO USADA MENOS RECENTEMENTE

- A ideia é que as páginas que foram intensamente utilizadas nas últimas instruções provavelmente serão utilizadas de forma intensa no futuro próximo

- Seguindo este pensamento, páginas que não estão sendo utilizadas por um longo período de tempo, provavelmente permanecerão inutilizadas por muito tempo

ALGORITMO USADA MENOS RECENTEMENTE

- Deste modo, quando ocorrer uma falta de página, elimina-se a página não utilizada pelo período de tempo mais longo
  - Essa estratégia é chamada de paginação LRU (last recently used)

- PERGUNTA!
  - Qual o maior problema dessa abordagem?

ALGORITMO USADA MENOS RECENTEMENTE

- Embora o LRU seja teoricamente razoável, ele não é barato.
  - Para implementar completamente o LRU é necessário manter uma lista vinculada de todas as páginas na memória,
    - Com a página usada mais recentemente na dianteira
    - E a menos usada na parte de trás
  - A dificuldade é que a lista deve ser atualizada em cada referência à memória
    - Encontrar uma página na lista, deletá-la e posicionar na dianteira é uma operação demorada

ALGORITMO DE SUBSTITUIÇÃO DE PÁGINA

- Outros algoritmos são encontrados na literatura para realização das substituições de páginas, quando ocorre uma falta de página. Dentre esses algoritmos, destacam-se:
  - NFU (*Not Frequently Used* – Não usado frequentemente)
  - Algoritmo de envelhecimento (*aging*)
  - Algoritmo do conjunto de trabalho
  - Algoritmo WSClock

EXERCÍCIO

Um sistema tem 4 quadros na memória principal. No momento de um tratamento de falta de páginas, existem as seguintes páginas alocadas:

PÁGINA	CARREGADA	ÚLTIMO ACESSO	REFERENCIA	MODIFICADA
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

O campo "Página" indica o identificador da página. Os campos "Carregada" e "Último Acesso" indicam, respectivamente, os instantes de tempo em que as páginas foram carregadas em memória principal e acessadas pela última vez. Os dois últimos campos indicam o estado dos bits de referência e modificada de cada página. Nestas condições, determine:

a) Qual página será removida pelo algoritmo NRU?

b) Qual página será removida pelo algoritmo FIFO?

c) Qual página será removida pelo algoritmo LRU?

d) Qual página será removida pelo algoritmo do relógio?