Universidade Federal da Bahia

Sistemas Operacionais

MATA58

Prof. Maycon Leone M. Peixoto

mayconleone@dcc.ufba.br

Programa

- Introdução aos Sistemas Operacionais
- Processos
  - Deadlocks – Final!!!
- Gerência de Memória
- Sistemas de Arquivos
- Entrada/saída
- Segurança
- Exemplos de Sistemas Operacionais

Gerênciamento de Memória

- Hierarquia de Memória
- Alocação particionada estática e dinâmica
- Gerenciamento dos espaços
- Swapping
- Memória virtual
  - Paginação e segmentação

Gerenciamento de Memória

- Memória - recurso muito importante;
- Tendência atual do software
  - Lei de *Parkinson*: “Os programas se expandem para preencher a memória disponível para eles” (adaptação);
- Requisitos:
  - Muito grande;
  - Rápida;
  - Não volátil;
  - Baixo custo.

Gerenciamento de Memória

Hierarquia de Memória:

- Cache – vários sub-níveis
- Memória Principal
- Disco

Gerenciamento de Memória

Hierarquia de Memória:

- Cache
  - Pequena quantidade – k/M bytes
  - Alto custo por byte
  - Muito rápida
  - Volátil
- Memória Principal
- Disco

Gerenciamento de Memória

Hierarquia de Memória:

- Cache
- Memória Principal
  - Quantidade intermediária – M/G bytes
  - Custo médio por byte
  - Velocidade média
  - Volátil
- Disco

Gerenciamento de Memória

Hierarquia de Memória:

- Cache
- Memória Principal
- Disco
  - Grande quantidade – G/T bytes
  - Baixo custo por byte
  - Lenta
  - Não volátil

Gerenciamento de Memória

Hierarquia de Memória:

- Para cada tipo de memória:
  - gerenciar espaços livres/ocupados
  - Alocar processos/dados na memória
  - Localizar dado
- Entre os níveis de memória:
  - Gerenciar trocas

Hierarquia de Memórias

- Ponto chave:
  - Organizar a hierarquia de memórias para que a porcentagem de acesso a um determinado nível seja sucessivamente menor do que a porcentagem de acesso a um nível imediatamente superior.

Revisão sobre tipos de Memórias

<table>
<tr><th>Tipo de Memória</th><th>Categoria</th><th>Mecanismo de apagamento</th><th>Mecanismo de escrita</th><th>Volatilidade</th></tr>
<tr><td>Memória de acesso aleatório (RAM)</td><td>Memória de leitura e de escrita</td><td>Eletricamente, em nível de bytes</td><td>Eletricamente</td><td>Volátil</td></tr>
<tr><td>Memória apenas de leitura (ROM)</td><td rowspan="2">Memória apenas de leitura</td><td rowspan="2">Não é possível</td><td>Máscaras</td><td rowspan="5">Não-volátil</td></tr>
<tr><td>ROM programável (PROM)</td><td rowspan="4">Eletricamente</td></tr>
<tr><td>PROM apagável (EPROM)</td><td rowspan="3">Memória principalmente de leitura</td><td>Luz UV, em nível de pastilha</td></tr>
<tr><td>Memória flash</td><td>Eletricamente, em nível de blocos</td></tr>
<tr><td>PROM eletricamente apagável (EEPROM)</td><td>Eletricamente, em nível de bytes</td></tr>
</table>

Gerenciamento de Memória

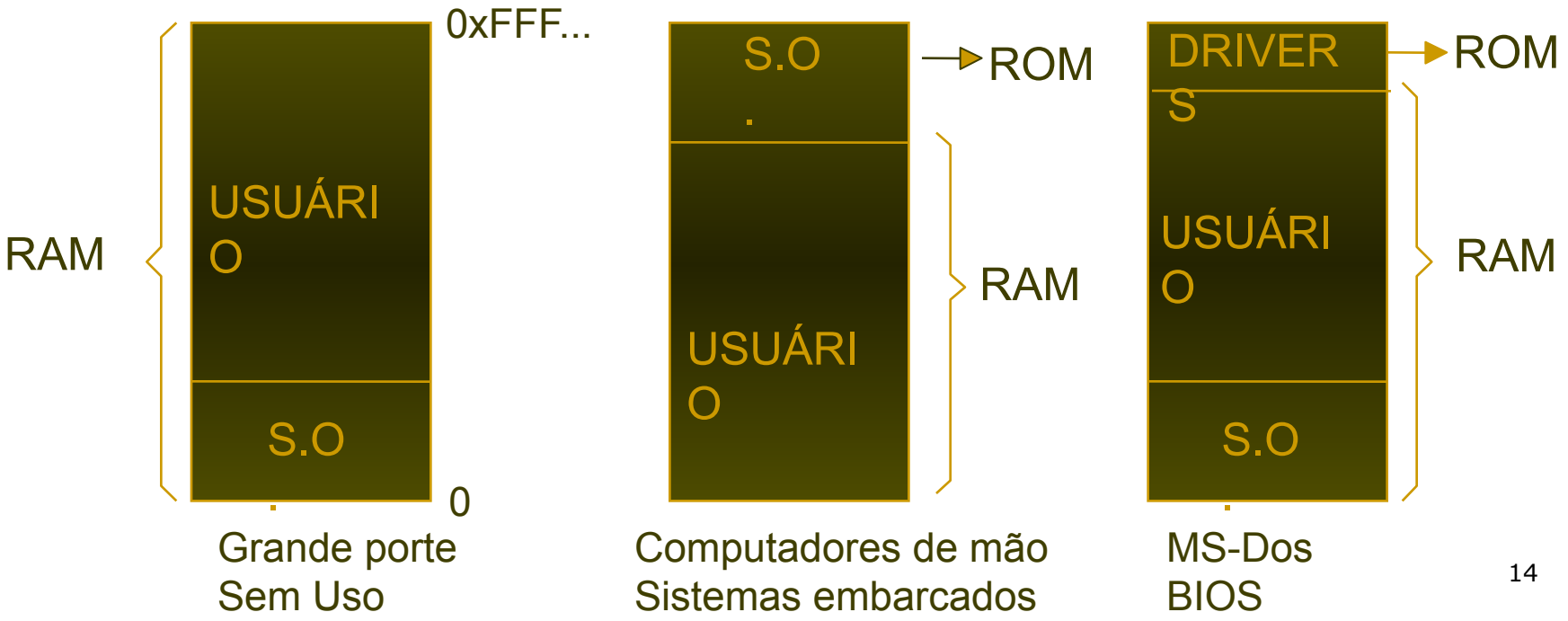
- Gerenciador de memória: responsável por alocar e liberar espaços na memória para os processos em execução; também responsável por gerenciar chaveamento entre os níveis de memória: principal e disco; principal e cache.

Gerenciamento de Memória

- Tipos básicos de gerenciamento:
  - Com paginação (chaveamento): Processos são movidos entre a memória principal e o disco; artifício usado para resolver o problema da falta de memória;
    - Se existe MP suficiente não há necessidade de se ter paginação;
  - Sem paginação: não há chaveamento;

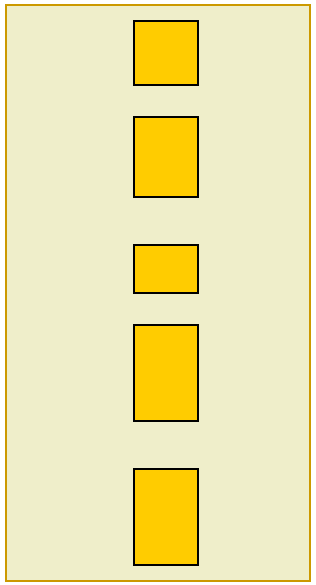
Gerenciamento de Memória

- Monoprogramação:
  - Sem paginação: gerenciamento mais simples;
- Apenas um processo na memória;



Gerenciamento de Memória

- Modelo de Multiprogramação:
  - Múltiplos processos sendo executados;
  - Eficiência da CPU;



Processo

Memória Principal - RAM

Necessidade de Particionamento da Memória Principal

Gerenciamento de Memória

- **Multiprogramação:**
  - vários processos na memória:
  - como proteger os processos uns dos outros?
  - e *kernel* de todos os processos?
  - como tratar a realocação?

- Todas as soluções envolvem equipar a CPU com um hardware especial:
  - **MMU (*memory management unit*);**

Gerenciamento de Memória

- **Realocação:**
  - Quando um programa é montado (*link),* i.e. programa principal + rotinas do usuário + rotinas da biblioteca □ executável, o montador (*linker)* deve saber ***em que endereço*** o programa irá iniciar na memória;
  - Nesse caso, para que o montador não escreva em um local indevido (por exemplo na área do SO), é preciso de realocação:

    #100 + Δ □ que depende da partição!!!

Gerenciamento de Memória

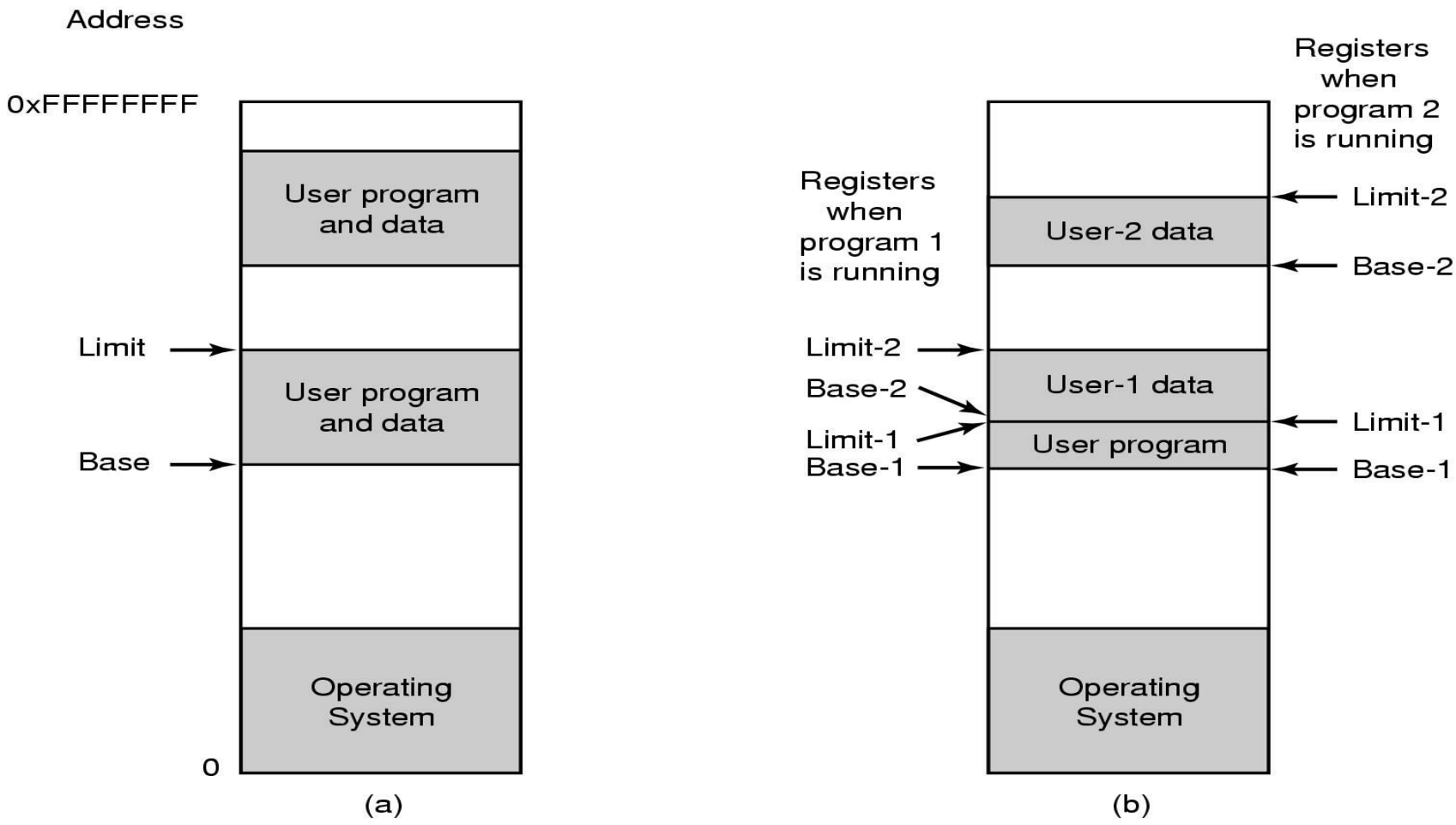
- **Proteção:**
  - Com várias partições e programas ocupando diferentes espaços da memória é possível acontecer um acesso indevido;
- **Solução para ambos os problemas:**
  - 2 registradores ☐ base e limite
    - Quando um processo é escalonado o ***registrador-base*** é carregado com o endereço de início da partição e o ***registrador-limite*** com o tamanho da partição;
    - O registrador-base torna impossível a um processo uma remissão a qualquer parte de memória abaixo de si mesmo.

Gerenciamento de Memória

- 2 registradores □ base e limite
  - automaticamente, a **MMU** adiciona o conteúdo do ***registrador-base*** a cada endereço de memória gerado;
  - endereços são comparados com o ***registrador-limite*** para prevenir acessos indevidos;

Gerenciamento de Memória

Registradores base e limite



Gerenciamento de Memória
Partições

- Particionamento da memória pode ser realizado de duas maneiras:
  - Partições fixas (alocação estática);
  - Partições variáveis (alocação dinâmica);

- **Partições Fixas**:
  - Tamanho e número de partições são fixos (estáticos);
  - Não é atrativo, porque partições fixas tendem a desperdiçar memória (Qualquer espaço não utilizado é literalmente perdido)
  - Mais simples;

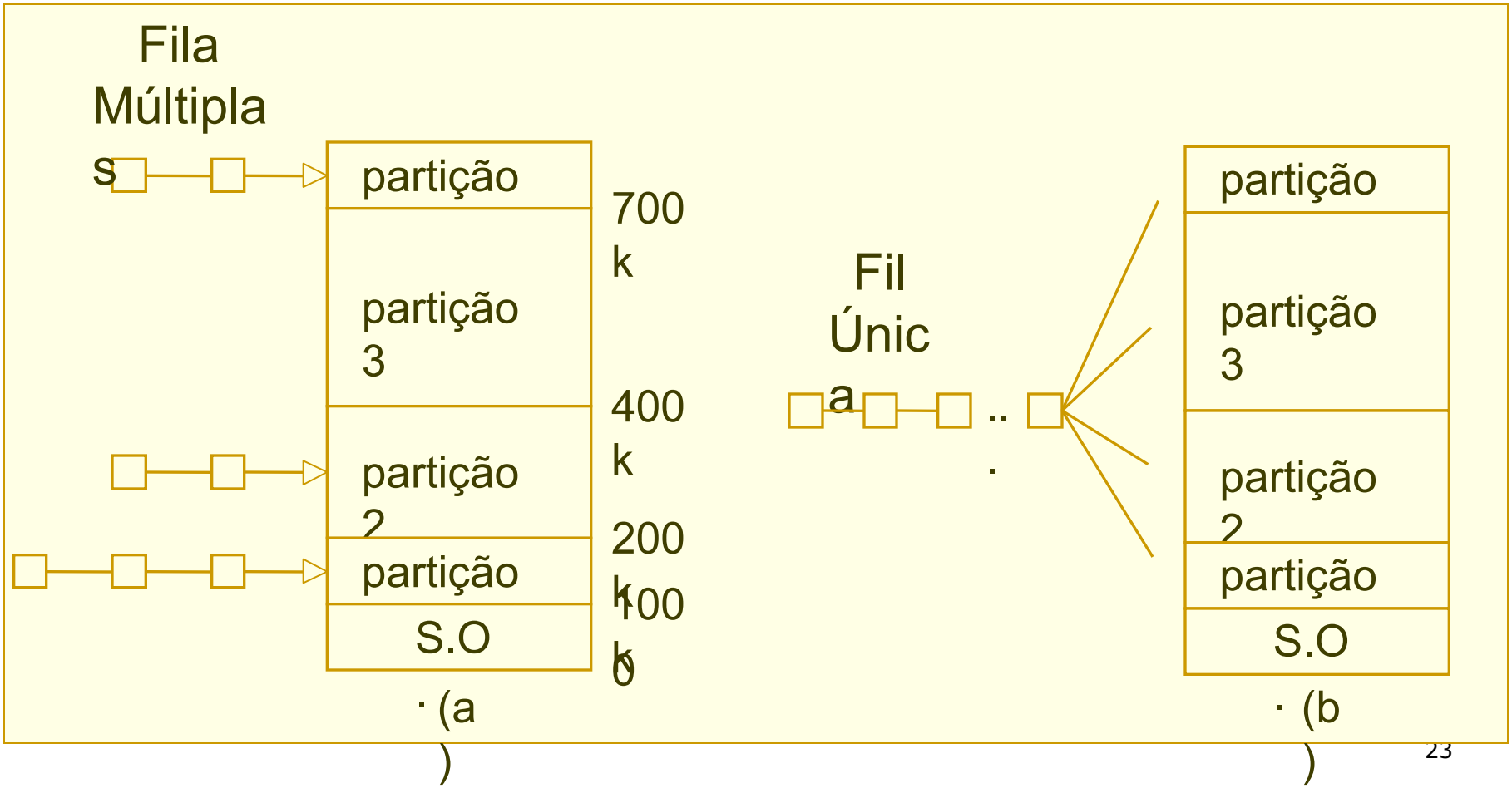
Gerenciamento de Memória
Partições Fixas

- Filas múltiplas:
  - Problema: filas não balanceadas;

- Fila única:
  - Melhor utilização da memória, pois procura o melhor processo para a partição considerada;
  - Problema: processos menores são prejudicados;

Gerenciamento de Memória
Partições Fixas

- Divisão da Memória em Partições Fixas:



Gerenciamento de Memória
Partições Fixas

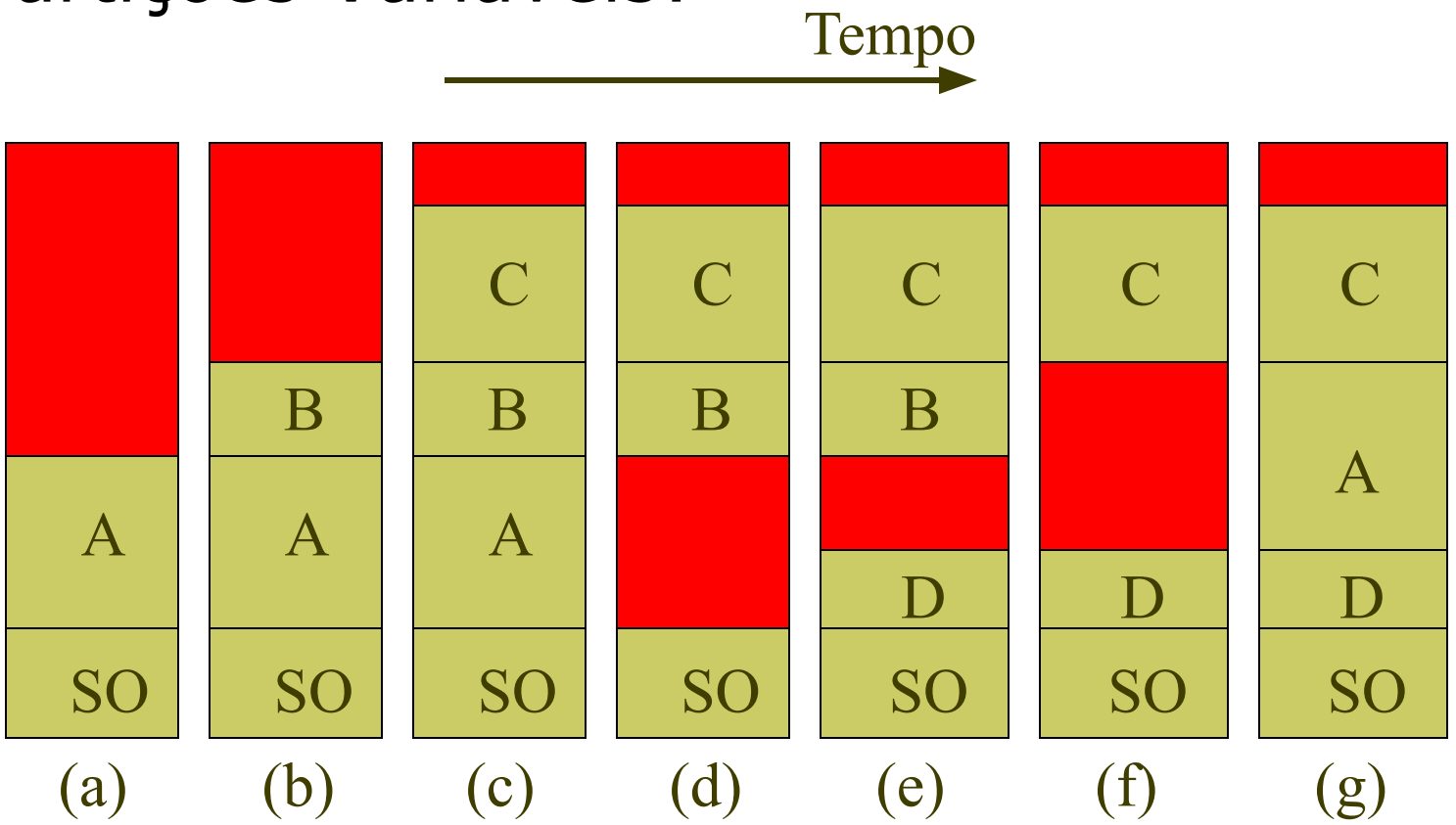
- Partições Fixas: problemas com fragmentação:
  - **Interna**: desperdício dentro da área alocada para um processo;
    - Ex.: processo de tamanho 40K ocupando uma partição de 50k;
  - **Externa**: desperdício fora da área alocada para um processo;
    - Duas partições livres: PL1 com 25k e PL2 com 100k, e um processo de tamanho 110K para ser executado;
    - Livre: 125K, mas o processo não pode ser executado;

Gerenciamento de Memória
Partições Variáveis

- **Partições Variáveis**:
  - Tamanho e número de partições variam;
  - Otimiza a utilização da memória, mas complica a alocação e liberação da memória;
  - Partições são alocadas dinamicamente;
  - SO mantém na memória uma lista com os espaços livres;
  - Menor fragmentação interna e grande fragmentação externa;
    - Solução: Compactação;

Gerenciamento de Memória
Partições Variáveis

- Partições Variáveis:



Gerenciamento de Memória

- Minimizar espaço de memória inutilizados:
  - Compactação: necessária para recuperar os espaços perdidos por fragmentação; no entanto, muito custosa para a CPU;
- Técnicas para alocação dinâmica de memória:
  - *Bitmaps*;
  - Listas Encadeadas;

Gerenciamento de Memória

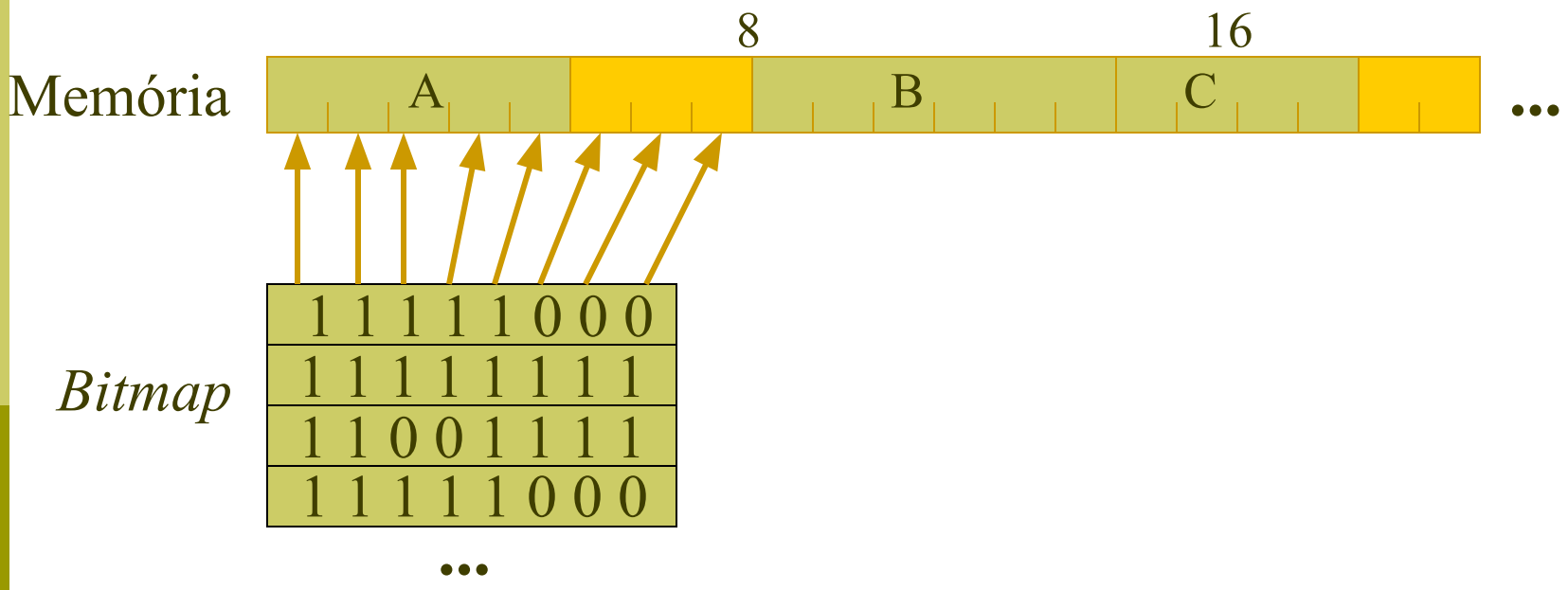
- Técnica com *Bitmaps*:
  - Memória é dividida em unidades de alocação em kbytes;
  - Cada unidade corresponde a um *bit* no *bitmap*:

    0 ☐ livre

    1 ☐ ocupado
  - Tamanho do *bitmap* depende do tamanho da unidade e do tamanho da memória;
  - Ex.:
    - unidades de alocação pequenas ☐ *bitmap* grande;
    - unidades de alocação grandes ☐ perda de espaço;

Gerenciamento de Memória

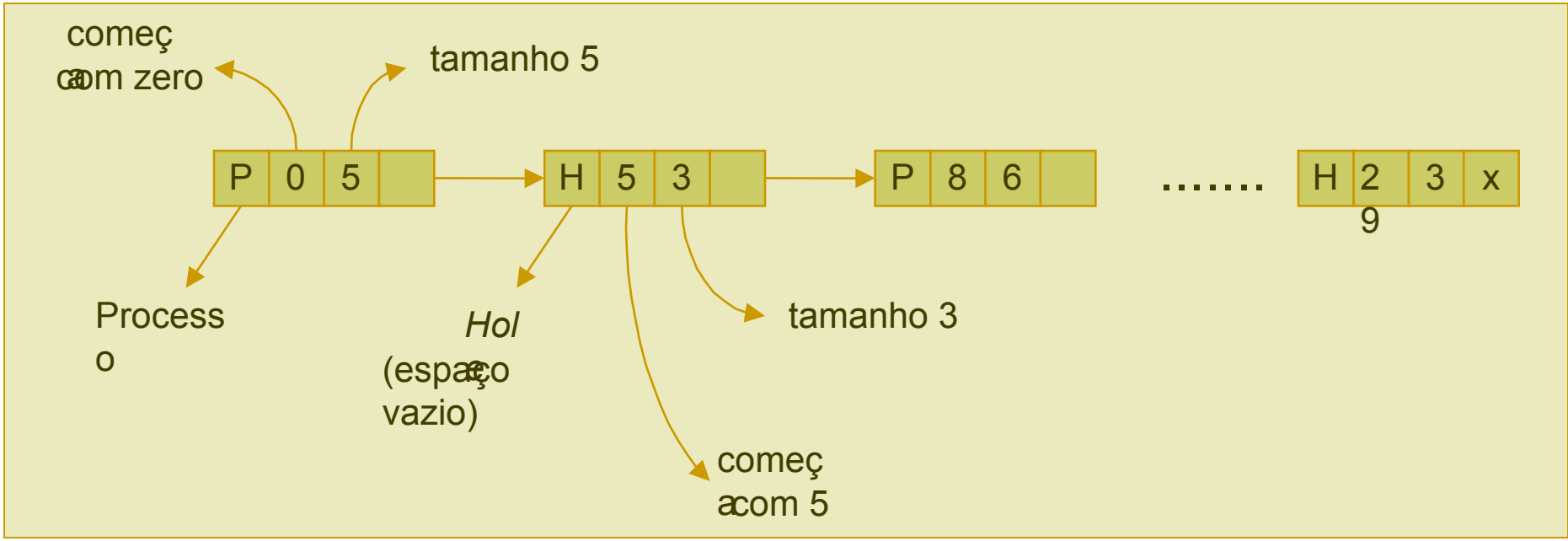
- Técnica com *Bitmaps*:



Gerenciamento de Memória

- Técnica com Listas Encadeadas:
  - Uma lista para os espaços vazios e outra para os espaços cheios, ou uma lista para ambos!

"espaço ≡ segmento"



Gerenciamento de Memória

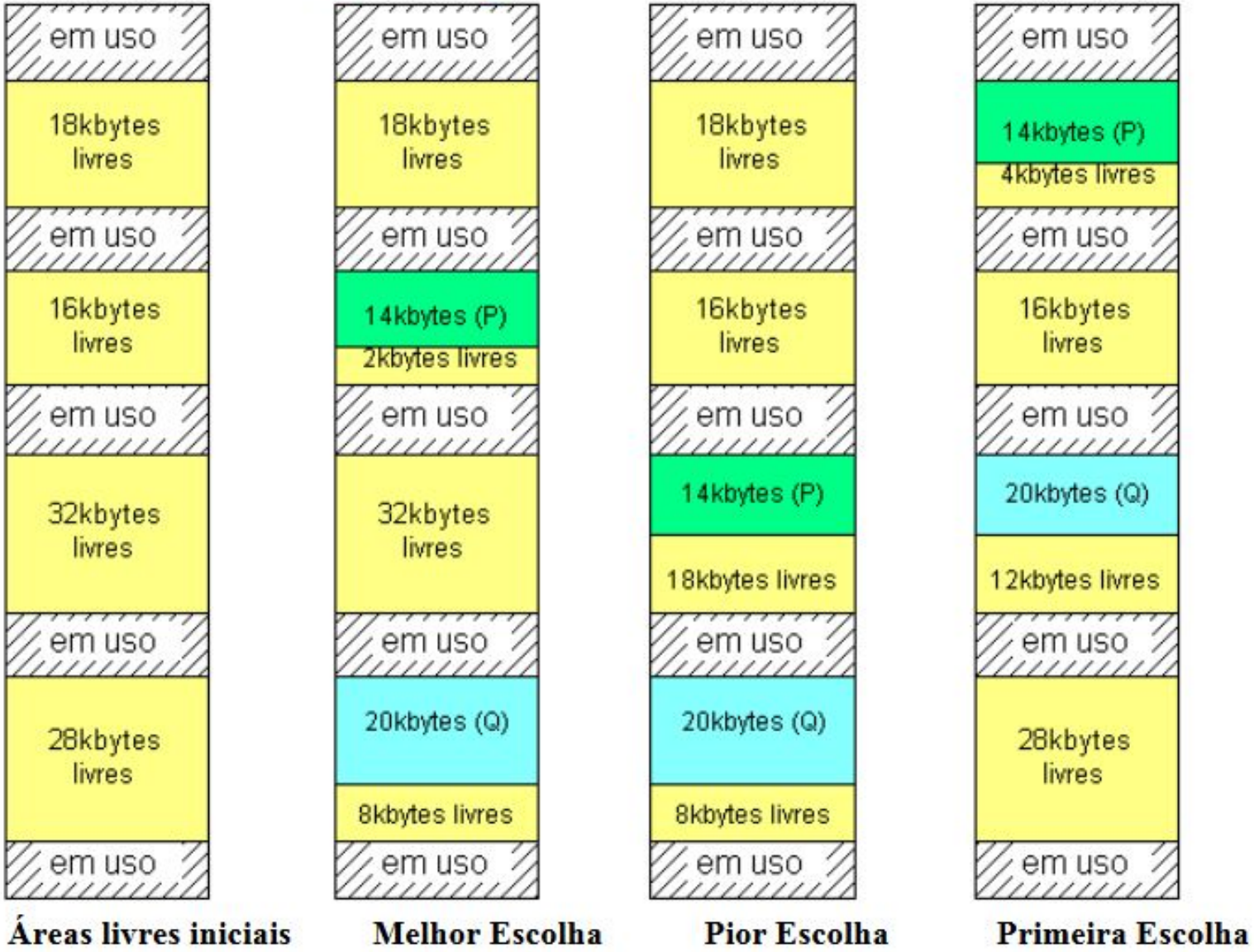
- Algoritmos de Alocação □ quando um novo processo é criado:
  - *FIRST FIT*
    - 1º segmento é usado;
    - Rápido, mas pode desperdiçar memória por fragmentação;
  - *NEXT FIT*
    - 1º segmento é usado;
    - Mas na próxima alocação inicia busca do ponto que parou anteriormente;
    - Possui desempenho inferior;

Gerenciamento de Memória

- *BEST FIT*
  - Procura na lista toda e aloca o espaço que mais convém;
  - Menor fragmentação;
  - Mais lento;
- *WORST FIT*
  - Aloca o maior espaço disponível;
- *QUICK FIT*
  - Mantém listas separadas para os espaços mais requisitados;

Gerenciamento de Memória

Alocação de Segmentos Livres



Gerenciamento de Memória
Alocação de Segmentos Livres

Principais Consequências

- **A melhor escolha:** deixa o menor resto, porém após um longo processamento poderá deixar "buracos" muito pequenos para serem úteis.

- **A pior escolha:** deixa o maior espaço após cada alocação, mas tende a espalhar as porções não utilizadas sobre áreas não contínuas de memória e, portanto, pode tornar difícil alocar grandes jobs.

- **A primeira escolha:** tende a ser um meio termo entre a melhor e a pior escolha, com a característica adicional de fazer com que os espaços vazios migrem para o final da memória.

Gerenciamento de Memória

- Cada algoritmo pode manter listas separadas para processos e para espaços livres:
  - Vantagem:
    - Aumenta desempenho;
  - Desvantagens:
    - Aumenta complexidade quando espaço de memória é liberado – gerenciamento das listas;
    - Fragmentação;