Sistemas Operacionais

Gerência de Memória (cont)

Andreza leite
andreza.leite@univasf.edu.br

O Grande Problema
Alocação Contínua: Fragmentação

Memória

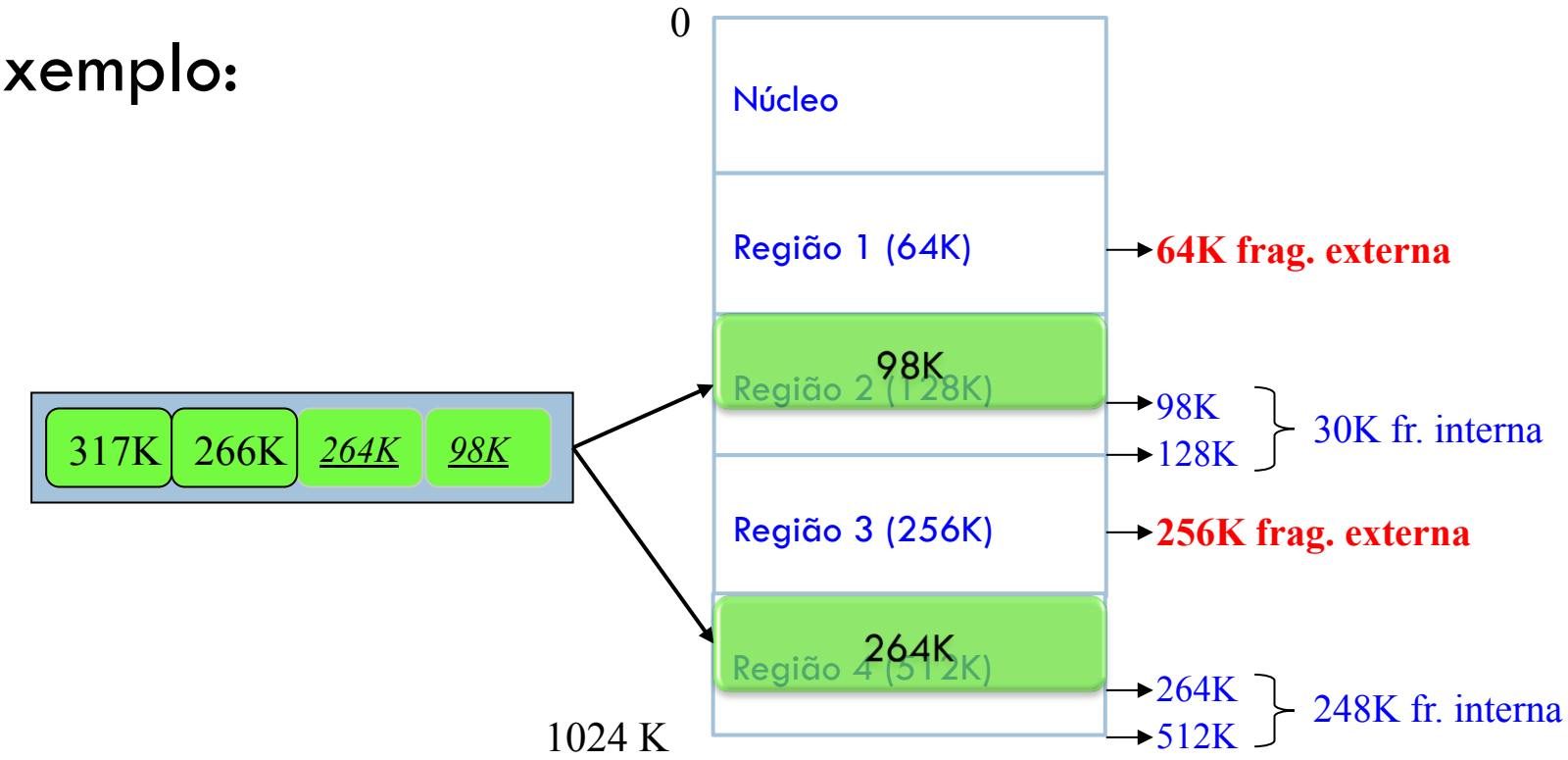
- Fragmentação de Memória
  - A gestão de memória mediante partições fixas provoca o aparecimento de áreas de memória **não utilizadas.**
  - Este efeito se denomina fragmentação da memória e pode ser de dois tipos:
    - Fragmentação interna

      Porção de memória de uma determinada partição que não é utilizada devido aos requisitos reduzidos dos processos.
    - Fragmentação externa

      Ocorre quando existe espaço suficiente de memória livre para satisfazer uma requisição, porém **não é continuo** e portanto não pode ser utilizado.

Memória

- Exemplo:



Fragmentação **Interna** = 30K (Reg2) + 248K (Reg4) = 278K

Fragmentação **Externa** = 64K (Reg1) + 256K (Reg3) = 320K

Memória

- Partições de tamanho fixo
  - Inconvenientes
    - Nivel de multiprogramação do sistema limitado
      - N°. de partições especificadas no tempo de geração do sistema
      - Limita o n°. de processos ativos dentro do sistema
    - Fragmentação interna.
      - Tamanho das partições são setadas no tempo de geração do sistema
    - Fragmentação externa.

Memória

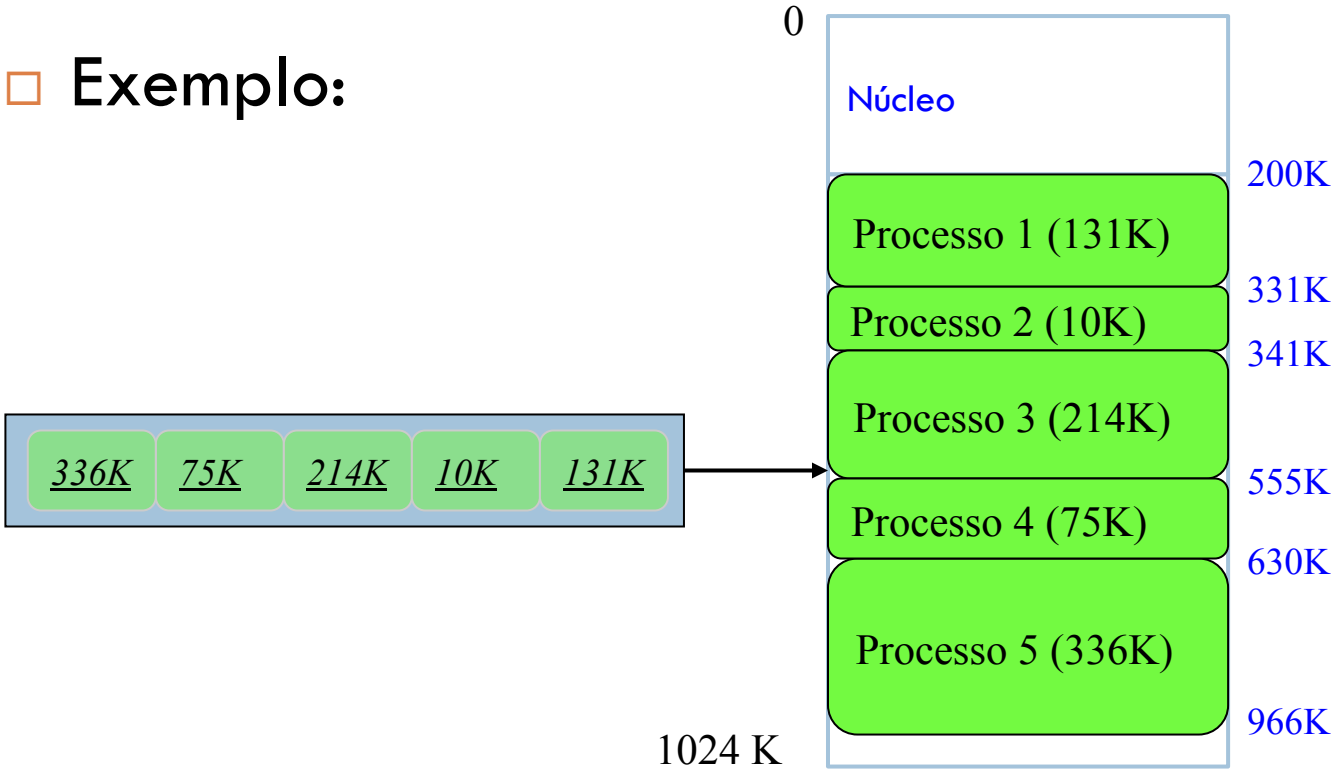
- Partições de tamanho variável (Dinâmicas)
  - O Principal **problema** das partições de tamanho fixo é **determinar o melhor tamanho** para as regiões para reduzir a fragmentação interna e externa.
  - A solução para este problema consiste em permitir que os tamanhos das regiões possam variar em função das necessidades das tarefas, portanto, utilização de partições variáveis.

Memória

- Partições de tamanho variável (Dinâmicas)
  - Partições são criadas **dinamicamente**, de forma que cada processo é alocado dentro da partição do mesmo tamanho do processo
    - Para cada processo é atribuido exatamente a porção de memória que necessita:
      - **Melhor aproveitamento da memória.**
      - **Incremento do número de tarefas em memória.**
  - Para implementar este mecanismo, o S.O. deve manter uma **tabela** com a informação correspondente as regiões definidas em cada instante de tempo.

Memória

- Exemplo:



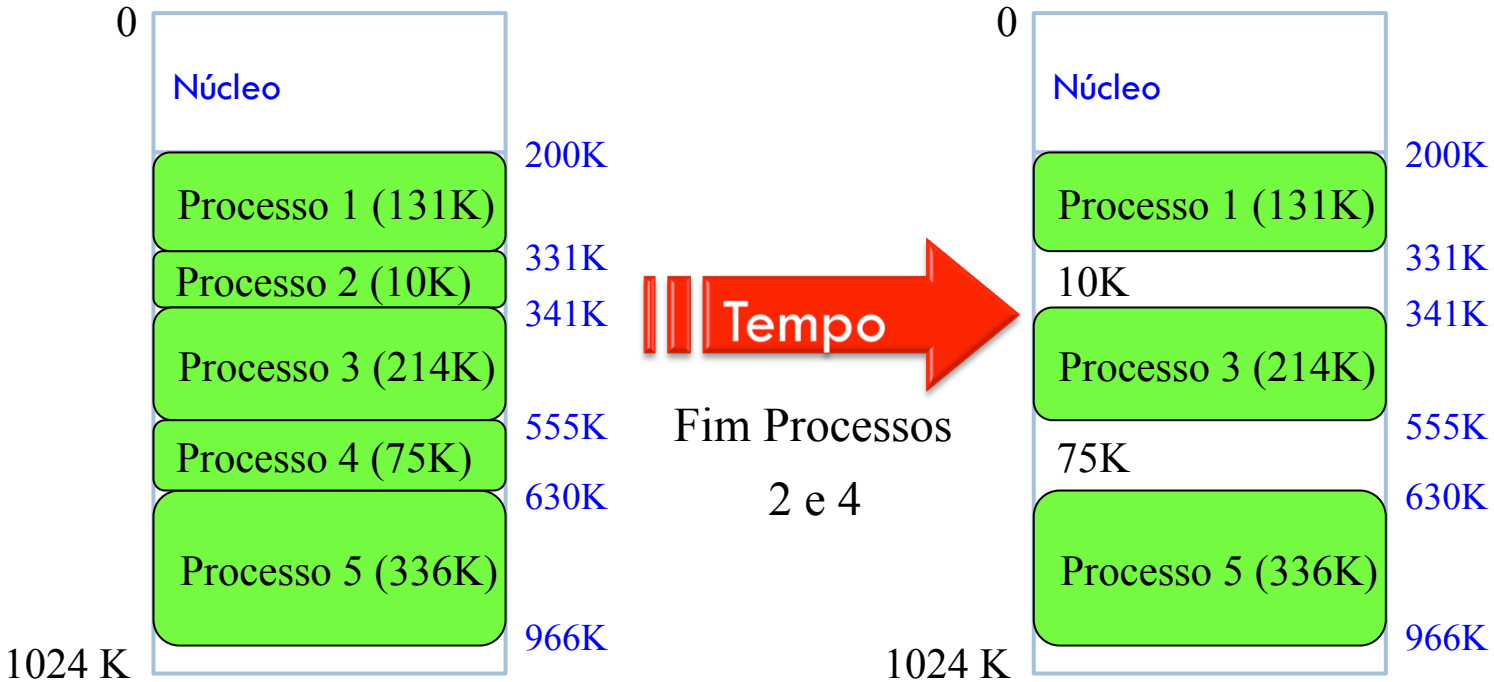
- Este sistema consegue uma melhor utilização da memória, já que a **fragmentação interna é nula**.

Memória

- Partições de tamanho variável (Dinâmicas)
  - Problema:
    - Aparição da fragmentação externa a medida que os processos vão iniciando e finalizando.

Memória

- Exemplo:

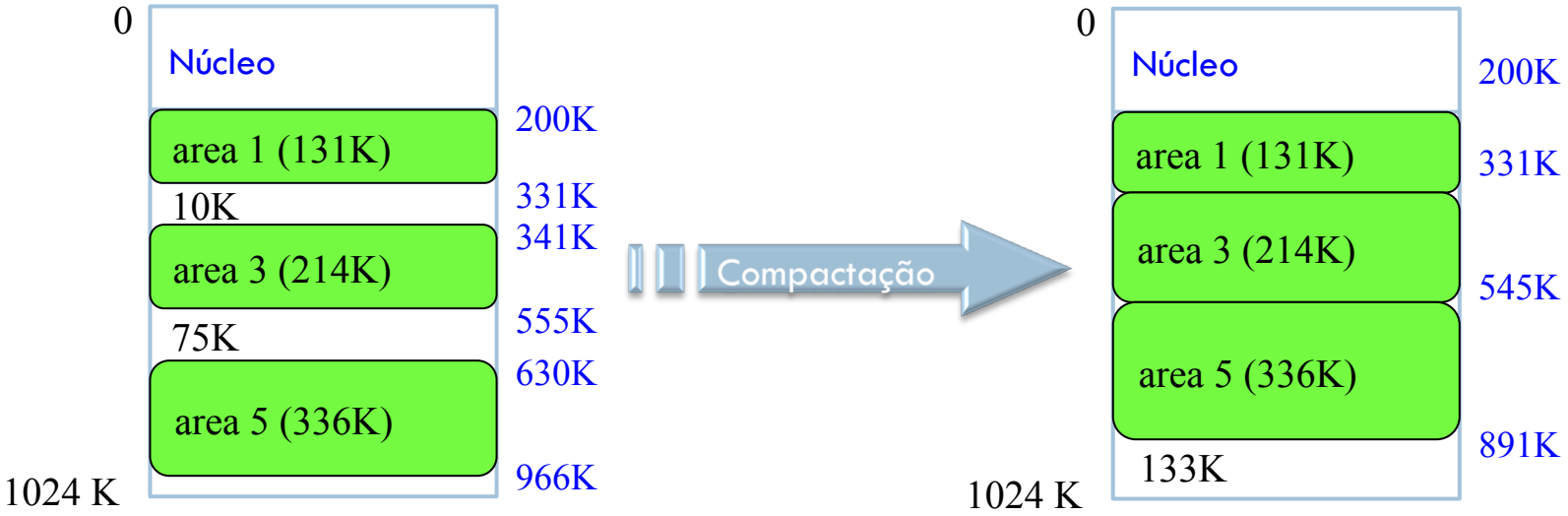


Memória

- Partições de tamanho Variável (Dinâmicas)
  - A solução à fragmentação externa é agrupar os diferentes fragmentos de memória livre para formar um único bloco maior.
  - Técnicas:
    - **Condensação**:
      - Agrupação de fragmentos continuos (trivial).
    - **Compactação**:
      - Agrupação de fragmentos separados por regiões em uso. Requer a reorganização das tarefas para que todo o espaço livre seja alocado em um único fragmento.

Memória

- Partições de tamanho variável (Dinâmicas)



Memória

- Partições de tamanho variável (Dinâmicas)

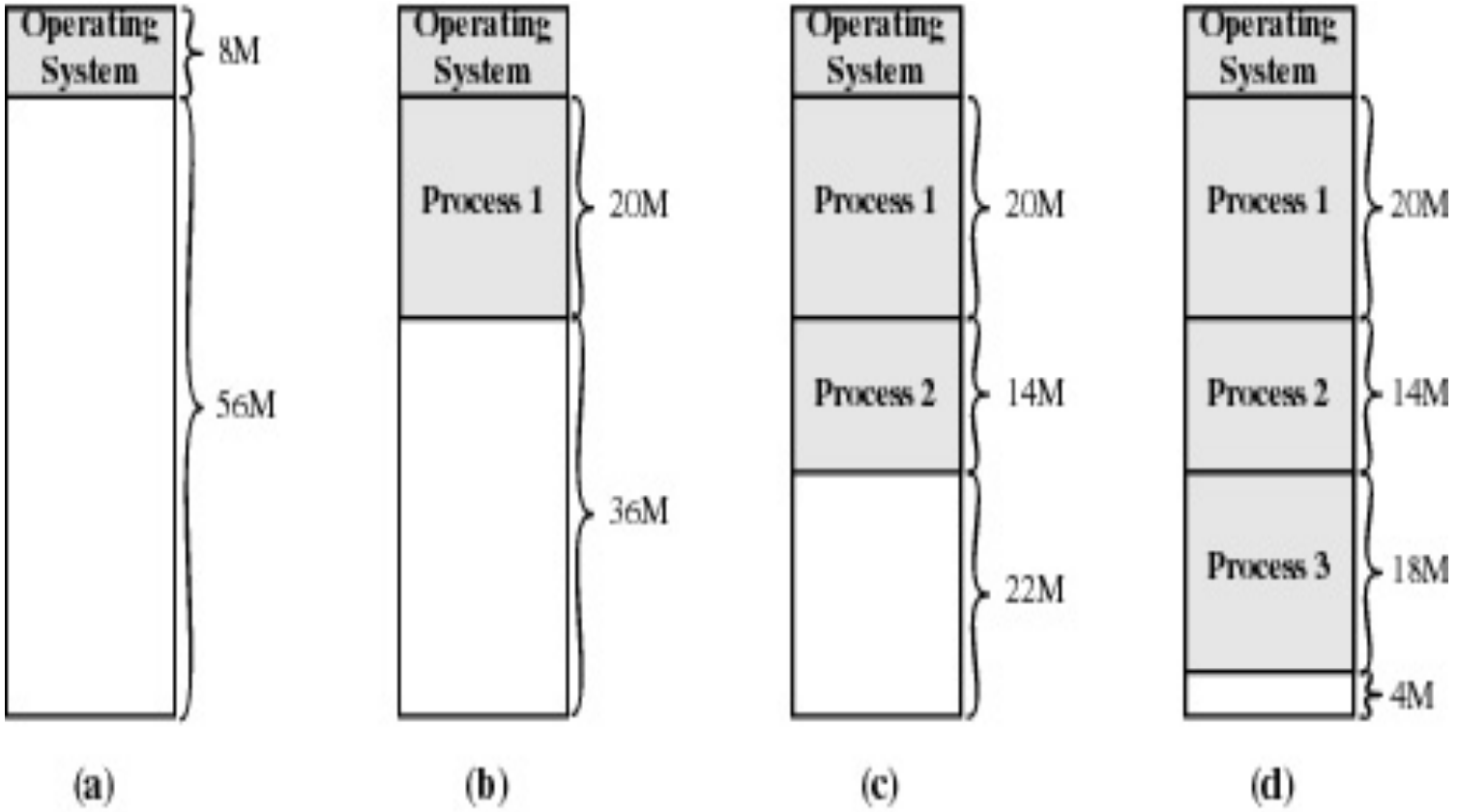


Figure 7.4 The Effect of Dynamic Partitioning

Memória

- Partições de tamanho variável (Dinâmicas)

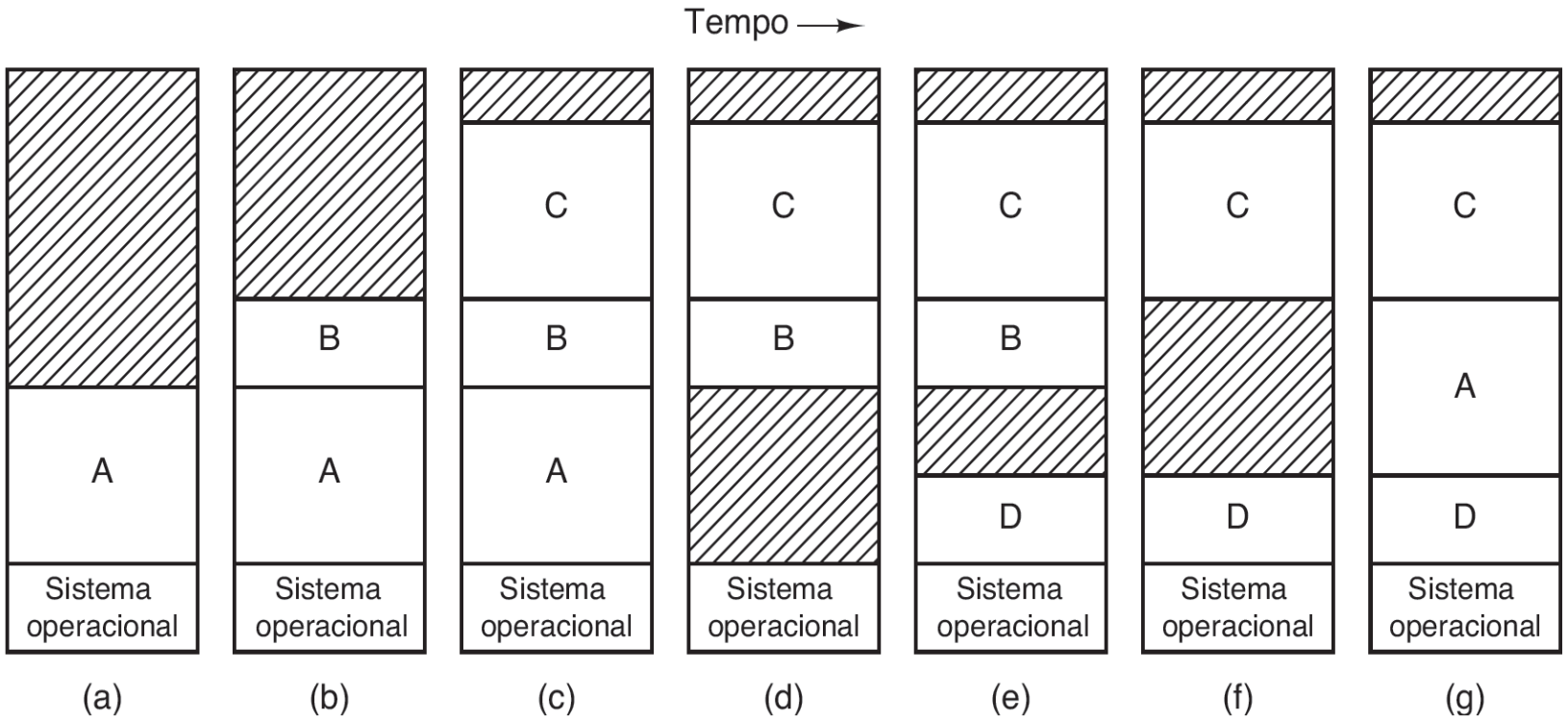


Figura 3.4 Alterações na alocação de memória à medida que processos entram e saem dela. As regiões sombreadas correspondem a regiões da memória não utilizadas naquele instante.

Memória

- Partições de tamanho variável (Dinâmicas)
  - Gerenciando buracos e processos
    - Mapas de bits
      - Problema: Busca de k zeros consecutivos para alocação de k unidades. Raramente é utilizado atualmente. É muito lenta
    - Listas encadeadas
      - Lista ordenada por endereço permite vários algoritmos de alocação
  - Algoritmos de alocação
    - O SO deve decidir qual bloco livre (buraco) a ser alocado ao processo de forma eficiente.

Memória

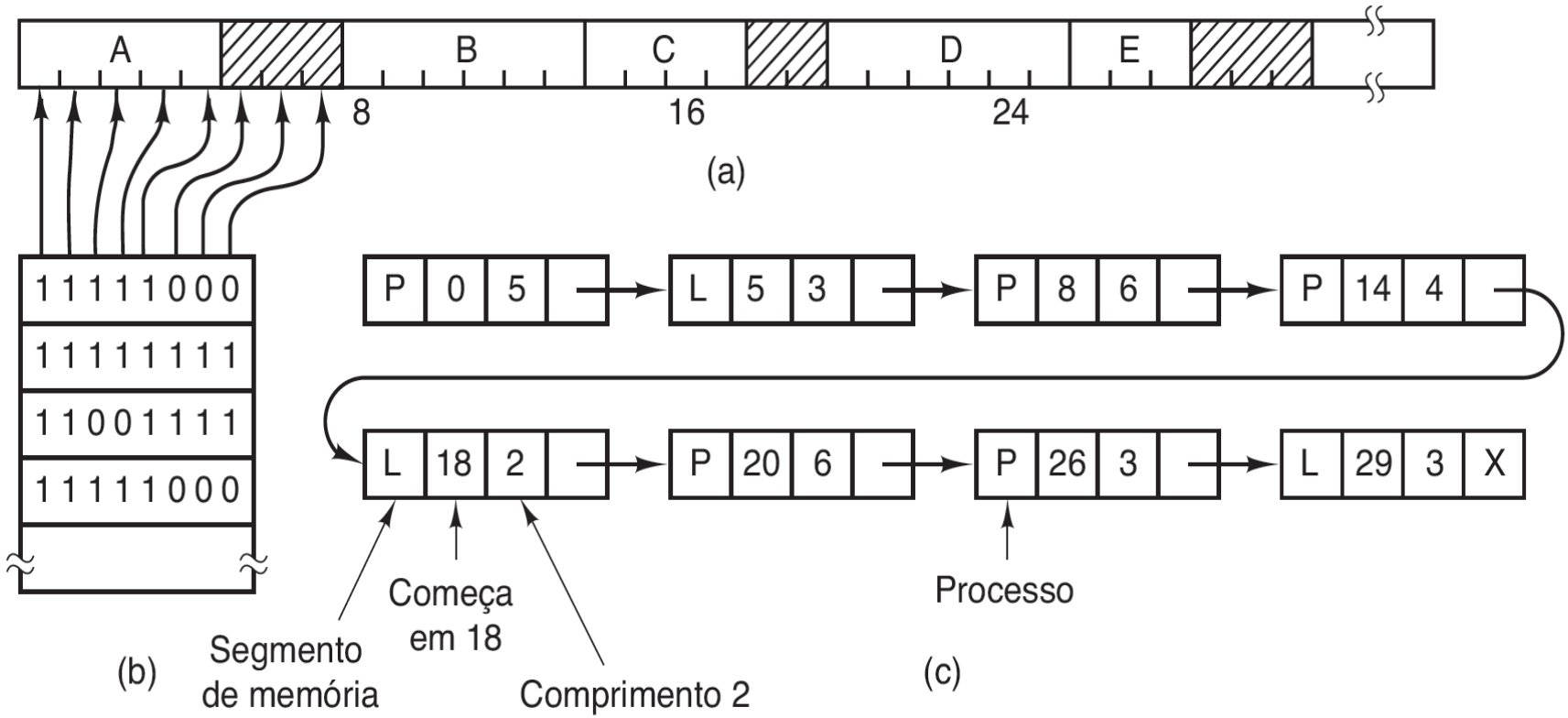


Figura 3.6 (a) Parte da memória com cinco processos e três segmentos de memória. As marcas mostram as unidades de alocação de memória. As regiões sombreadas (0 no mapa de bits) estão livres. (b) O mapa de bits correspondente. (c) A mesma informação como lista.

Memória

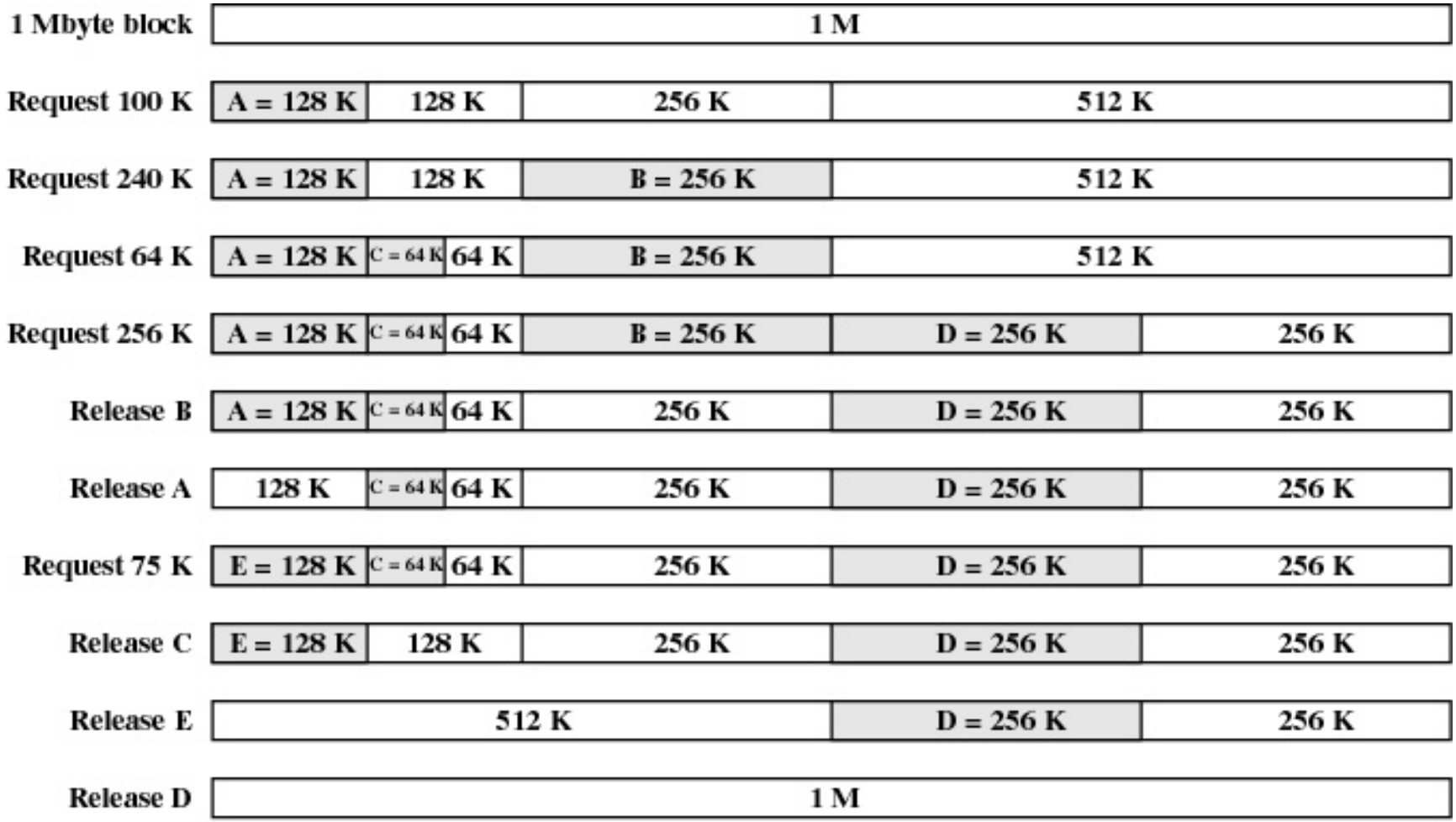
- Sistema Buddy
  - O espaço disponível total da memória é tratado como um único bloco de 2^U (bloco de maior tamanho que pode ser alocado)
  - Se um pedido de tamanho s é tal que

    $$2^{U-1} < s \leq 2^U,$$

    - Então: o bloco completo 2^U é alocado
    - Senão: O bloco é divido em 2 "buddies" iguais. A divisão continua até que o bloco encontrado seja > ou = ao s requerido.
  - Uma forma modificada deste sistema é usada para alocação de memória no kernel do UNIX. Também tem sido usado em aplicações de sistemas paralelos (alocação e liberação de programas paralelos).

Memória

- Sistema Buddy



Memória

- Sistema Buddy

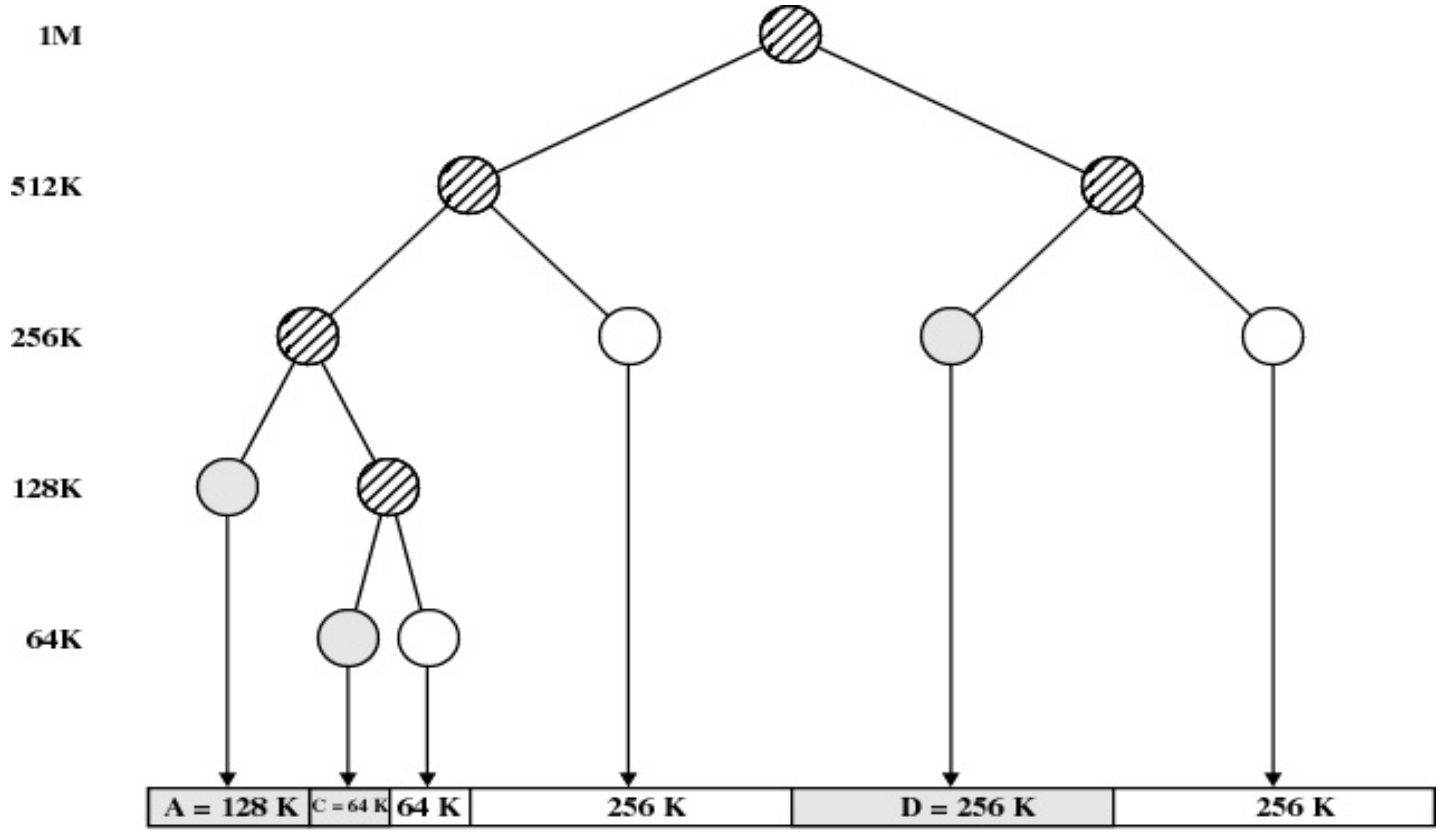


Figure 7.7 Tree Representation of Buddy System

Memória

- Partições de tamanho Variável (Dinâmicas)
  - Três algoritmos podem ser usados:
    - **First-fit** : Procura dentro da memória a partir o inicio e escolhe o primeiro bloco que é grande o suficiente para o processo a ser alocado;
    - **Next-fit:** Inicia a procura a partir do local onde foi feito a ultima locação e escolhe o bloco que é grande o suficiente para o processo a ser alocado
    - **Best-fit:** Escolhe o bloco que mais se aproxima do tamanho requerido;
    - **Worst-Fit:** Escolhe a lacuna que resultar na maior sobra/ Maior espaço possível;

Memória

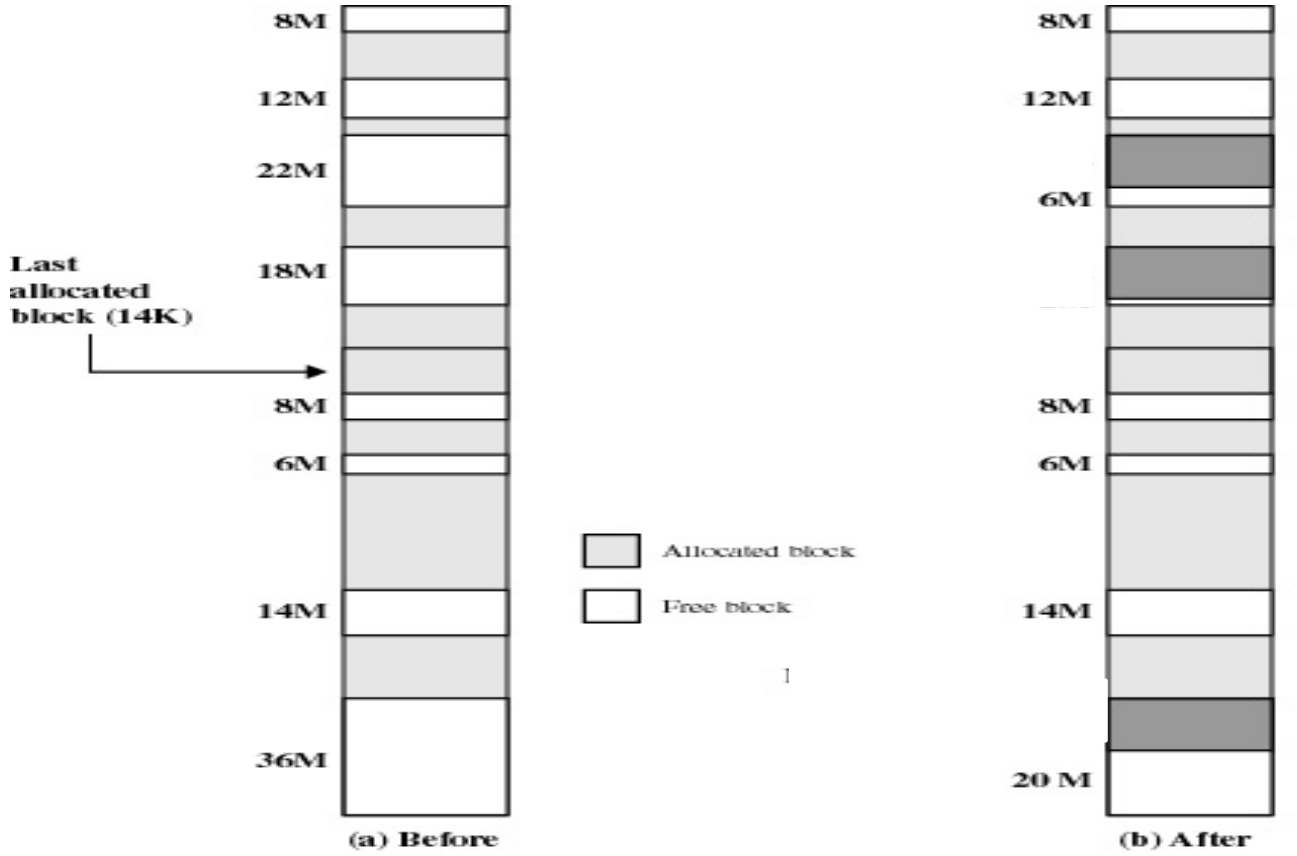


Figure 7.5 Example Memory Configuration Before and After Allocation of 16 Mbyte Block

Memória

- Qual deles é o melhor?
  - **First-fit :**
    - Mais rápido e o melhor
    - Ele gera no inicio da memória pequenas partições livres que necessitam ser pesquisadas a cada passo subseqüente do first-fit
  - **Next-fit:**
    - Freqüentemente aloca blocos livres no fim da memória
    - Como resultado, blocos grandes de memória são quebrados rapidamente em blocos menores
    - Compactação mais freqüente é requerida para obter grandes blocos no fim da memória
    - Performance inferior ao Fist-Fit.

Memória

- Qual deles é o melhor?
  - **Best-fit:**
    - Tem a pior performance
    - Procura pelo menor bloco que satisfaça o requisito
    - Como resultado, a memória principal é rapidamente quebrada em blocos muito pequenos que não podem satisfazer um pedido de alocação, exigindo assim uma compactação freqüente
  - **Worst-Fit:**
    - Escolhe Maior espaço possível;
    - Tempo de busca grande;
    - Não apresenta bons resultados

Alocação Não Contínua

Memória

- Alocação de memoria não continua.
  - Os métodos vistos anteriormente realizam uso ineficiente de memória (**fragmentação interna** e **externa**) devido a exigência de que toda memória alocada a um processo deveria ser continua.
  - **Solução**: Utilizar sistemas de gestão de memória que permitam a alocação de memória física **não** contínua aos processos.

Memória

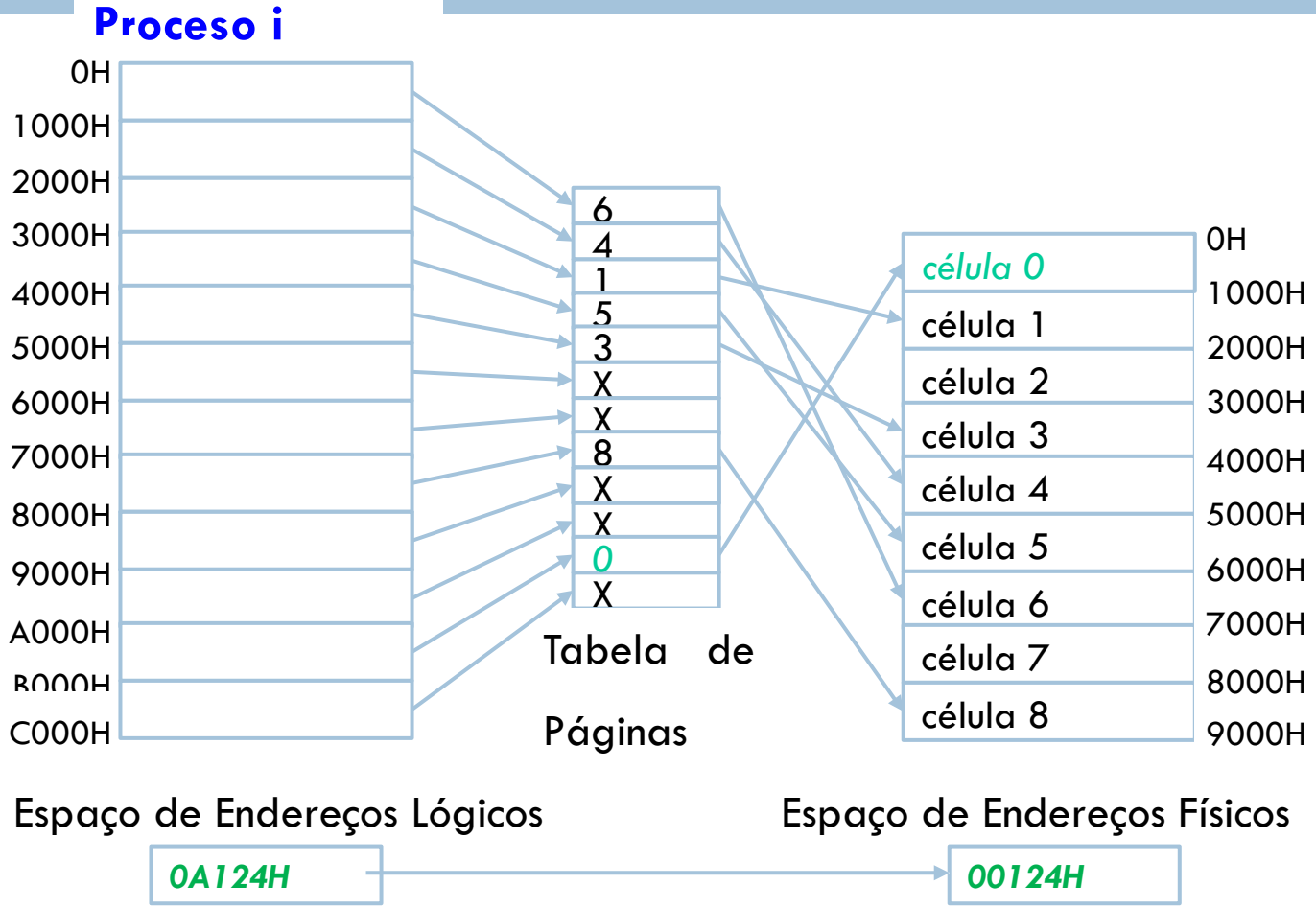
- Alocação de memoria não continua.
  - Existem dois sistemas principais de gestão de memória não continua:
    - Paginação.
    - Segmentação.

Paginação

Gerência Memória

- Alocação de memória não continua.
- Paginação
  - Mediante a paginação da memória física, o espaço de endereçamento dos processos se dividem em blocos de tamanhos fixos denominados páginas.
  - Qualquer página do espaço de endereçamento pode ser mapeada em qualquer página de memoria física (celula/frame/moldura de paginas).
    - O espaço de endereçamento lógico de um processo é dividido em páginas lógicas de tamanho fixo;

Gerência Memória



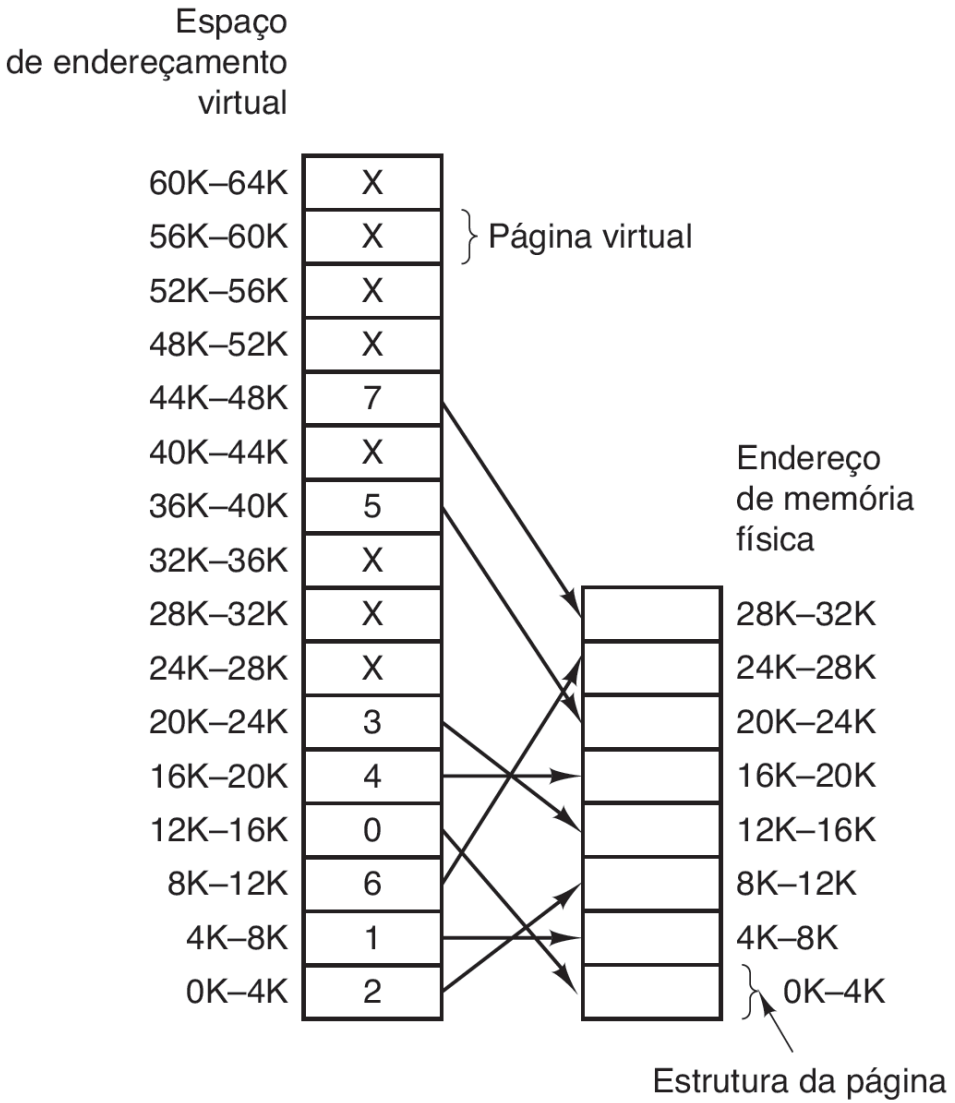


Figura 3.9 A relação entre endereços virtuais e endereços de memória física é dada pela tabela de páginas. Cada página começa com um múltiplo de 4096 e termina 4095 endereços acima; assim, 4K–8K na verdade significa 4096–8191 e 8K–12K significa 8192–12287.

Gerência Memória

- Alocação de memoria não continua.
  - Os Endereços lógicos são decompostos em duas partes:

Página	Deslocamento

    - Número da página lógica (End_log **DIV** Tam_pág)
    - Deslocamento dentro da página (End_log **MOD** Tam_pág)
  - O número de página (p) se utiliza como índice da tabela de páginas do processo para obter o endereço base da página de memória física (célula de memoria física) associada com o endereço lógico, o qual se soma o deslocamento (d).

Gerência Memória

- Alocação de memoria não continua.
  - Os Endereços lógicos são decompostos em duas partes:

Página	Deslocamento

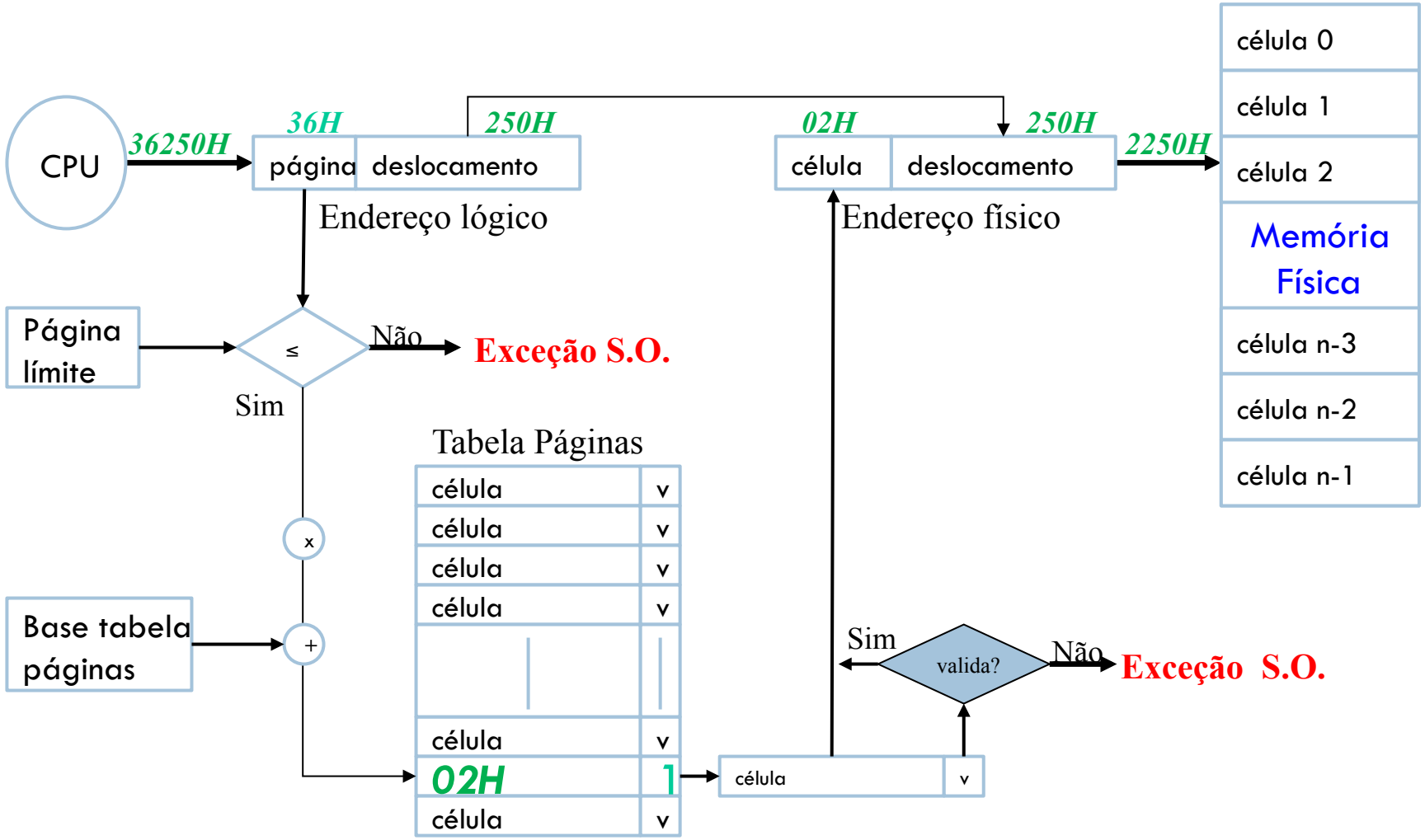
    - Número da página lógica (End_log **DIV** Tam_pág)
    - Deslocamento dentro da página (End_log **MOD** Tam_pág)
  - O número de página (p) se utiliza como índice da tabela de páginas do processo para obter o endereço base da página de memória física (célula de memoria física) associada com o endereço lógico, o qual se soma o deslocamento (d).

End_físico = TabelaPaginas[p].EndBase + d

ou

TabelaPaginas[p].Célula * TamPag + d

Gerência Memória



Gerência Memória

- Estruturas de dados paginação
  - Tabela de páginas.
    - Cada processo tem sua própria tabela de páginas.
    - Entrada tabelas de páginas:
      - Número de célula de memória física onde se situa a página.
      - Um bit que indica se a página é válida ou inválida.
      - Bits de proteção.

Gerência de Memória

Válida	Página virtual	Modificada	Proteção	Moldura da página
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

Gerência Memória

- Estruturas dados paginação
  - Tabela de células de memória física.
    - Contém informação de controle para gerenciar as células de memória física:
      - Se a célula está livre ou alocada.
      - A que processo e a que página está alocada a célula.

Gerência Memória

- Proteção das páginas
  - A proteção da memória em um ambiente com paginação é conseguido mediante utilização de bits de proteção ligados com cada uma das entradas da tabela de páginas.
  - Formato entrada tabela páginas:

Célula página física	V	R	W	X

    - V → Entrada válida.
    - R → Permissão de leitura
    - W → Permissão de escrita
    - X → Permissão execução
      - Mediante estes bits é possível uma proteção individual para cada uma das páginas.

Gerência Memória

- Proteção das páginas
  - A proteção da memória em um [illegible] é conseguido mediante utilização de bits de proteção ligados com cada uma das entradas da tabela de p[illegible]nas.
  - Formato entrada tabela páginas:

Célula página física	V	R	W	X

    - V → Entrada válida.
    - R → Permissão de leitura
    - W → Permissão de escrita
    - X → Permissão execução
      - Mediante estes bits é possível uma proteção individual para cada uma das páginas.

Quando o processo tenta acessar uma página que está marcada como inválida, a MMU gera uma interrupção de proteção, e o sistema operacional é acionado.

Em geral, isso representa um erro de programação, pois o processo está tentando acessar uma página que não faz parte da sua memória lógica.

Gerência Memória

- Proteção das páginas
  - A proteção da memória em
    conseguido mediante utiliza
    com cada uma das entradas d
  - Formato entrada tabela páginas:

Célula página física	V	R	W	X

    - V → Entrada válida.
    - R → Permissão de leitura
    - W → Permissão de escrita
    - X → Permissão execução
      - Mediante estes bits é possível uma proteção individual para cada uma das páginas.

Caso o processo tente acessar a página de uma maneira diferente daquela determinada pelos bits de proteção, a MMU gera uma interrupção de proteção e aciona o sistema operacional.

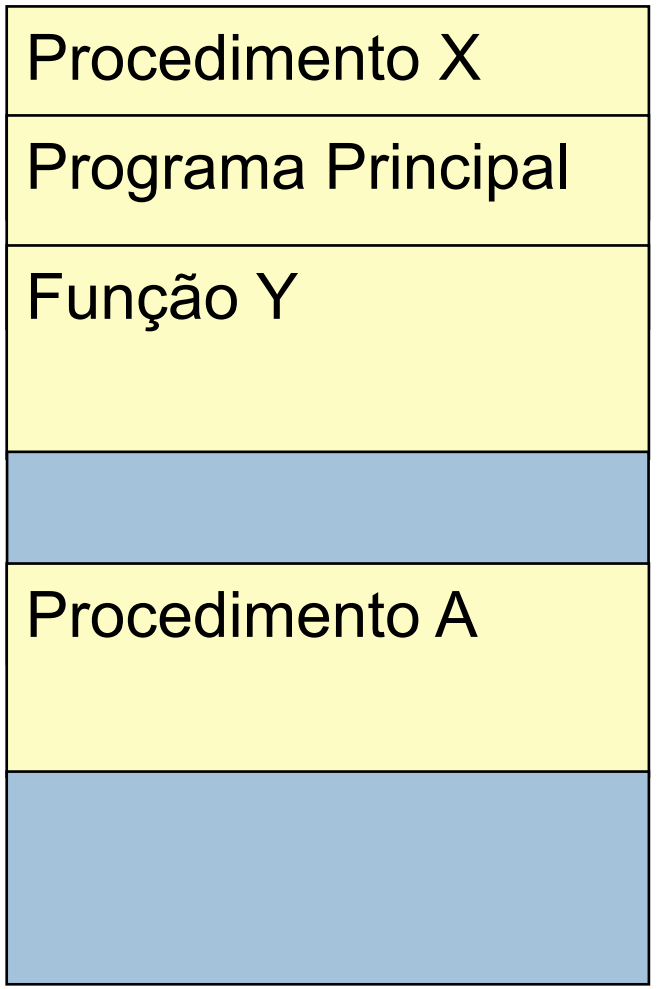
Segmentação

Segmentação

Técnica de gerência de memória onde programas são divididos em segmentos de tamanhos variados cada um com seu próprio espaço de endereçamento.

Segmentação

```
PROGRAM pr01;
   VAR A:ARRAY...
         C....
PROCEDURE X:
END;
FUNCTION Y;
END;
BEGIN
END.
```



Segmentação

- a principal diferença entre a **paginação** e a **segmentação** é a alocação da memória de maneira não fixa, a alocação depende da lógica do programa.
- o **mapeamento** é feito através das tabelas de mapeamento de segmentos.
- os **endereços** são compostos pelo número do segmento e um deslocamento dentro do segmento.
- cada **entrada** na tabela mantém o **endereço físico** do segmento, o tamanho do segmento, se ele está ou não na memória e sua proteção

Segmentação

- o sistema operacional mantém uma tabela com as áreas livres e ocupadas da memória
- somente segmentos referenciados são transferidos para a memória principal
- ocorre fragmentação externa

Paginação x Segmentação

	Paginação	Segmentação
quantos espaços de endereçamento lineares existem?	1	muitos
estes espaços de endereçamento podem exceder o tamanho da memória física?	sim	sim
os procedimentos e dados podem ser separadamente distinguidos e protegidos?	não	sim
tabelas cujos tamanhos sofram alteração podem se acomodadas facilmente?	não	sim
o compartilhamento de procedimentos é facilitado?	não	sim

Relembrando

Paginação

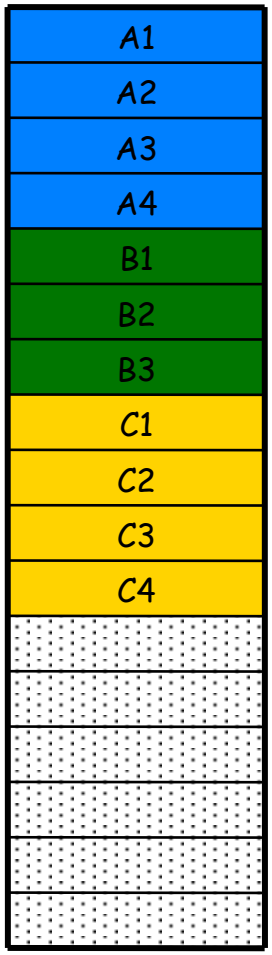
- Problemas tanto em particionamento fixo quanto dinâmico:
  - fixo – fragmentação interna;
  - dinâmico – fragmentação externa e realocação dinâmica;
- Solução:
  - Processo é dividido em páginas (blocos de processos);
  - MP é dividida em quadros de mesmo tamanho;
- Páginas/quadros são de pequeno tamanho (e.g., 1K): fragmentação interna pequena;
- Elimina fragmentação externa;
- SO mantém uma tabela de páginas por processo.

Paginação

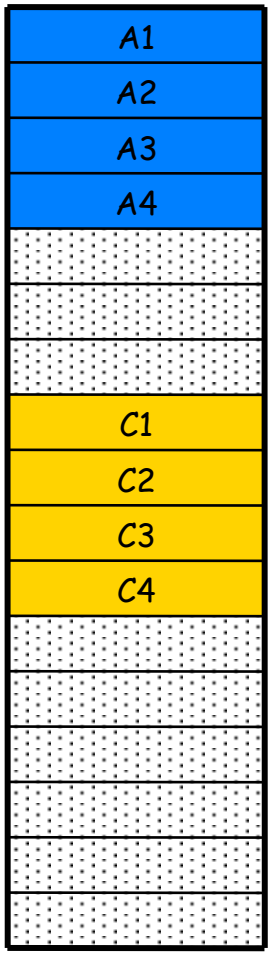
Exemplo: número de páginas por processo

A	4
B	3
C	4
D	5

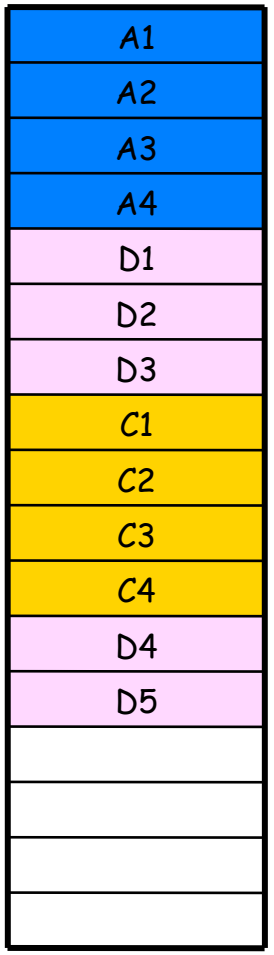
Processos A, B, C estão prontos



B termina



D é submetido



Paginação

- Processo não precisa estar completamente na MP;
- Processo não precisa ocupar área contígua em memória;
- Endereços são gerados dinamicamente em tempo de execução;
- Somente um registrador então, não é suficiente;
- Cada processo tem sua Tabela de Páginas (TP).

Segmentação

- Programas são normalmente separados em módulos: unidade lógica;
- Segmentos de um programa não precisam ser do mesmo tamanho;
- Existe um tamanho máximo para o segmento;
- Usuário tem controle;
- Elimina fragmentação interna (mas pode haver, externa).