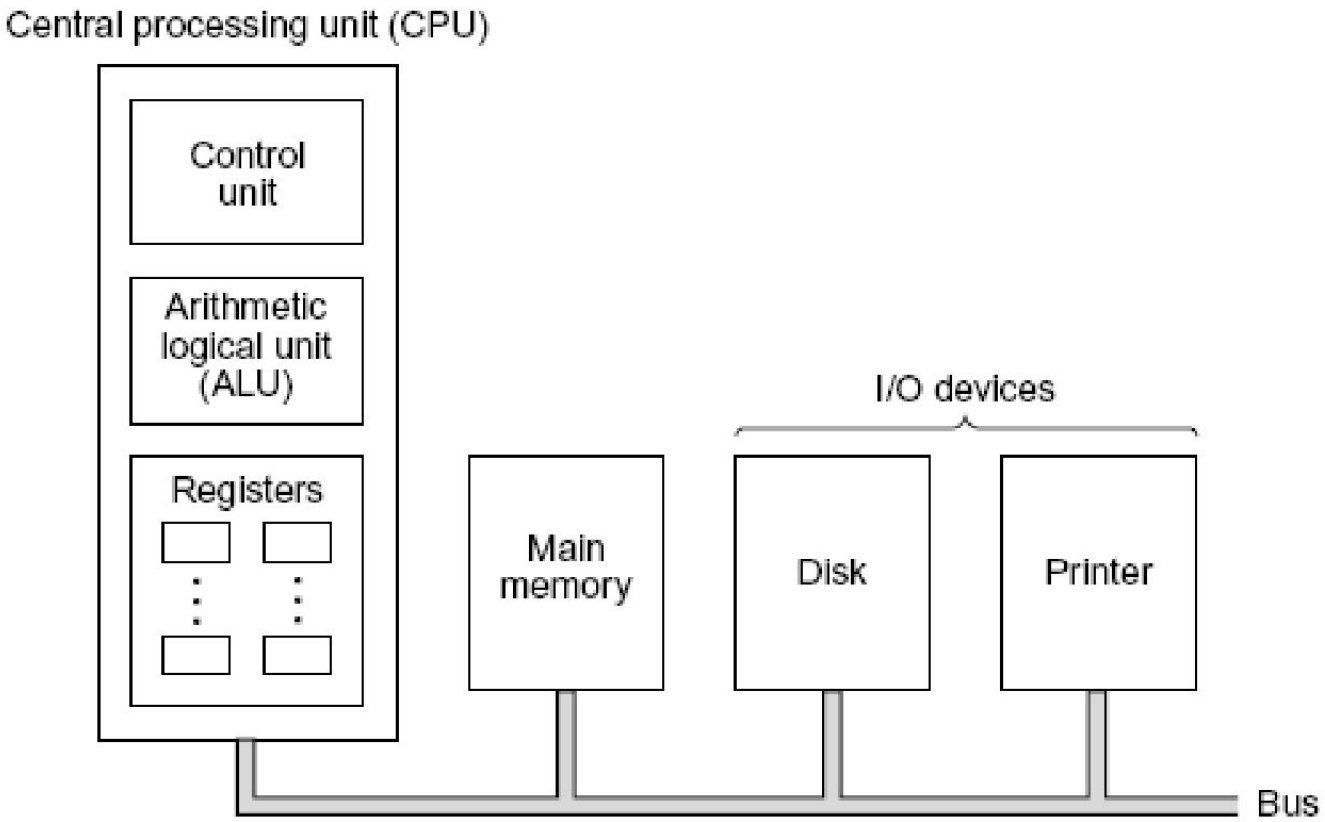
Laboratório de Pesquisa em Redes e Multimídia

Sistemas Operacionais

Gerência de Memória - conceitos básicos

Universidade Federal do Espírito Santo
Departamento de Informática

Memória Principal



Memória Principal (cont.)

- Componente essencial dentro da arquitetura de "programa armazenado", de John von Neumann.
- Quanto mais processos residentes na memória principal, melhor será o compartilhamento da CPU, que é o principal recurso do sistema e computação.
- A memória é um recurso caro e escasso, o que significa dizer que o S.O. deve implementar alguma estratégia de gerência deste este recurso.
- O próprio S.O., da sua parte, não deve usar muita memória, de modo a "liberá-la" para os programas de usuário.
- A Gerência de Memória, isto é, o seu uso eficiente e racional, é, portanto, uma das tarefas primordiais dos S.O.

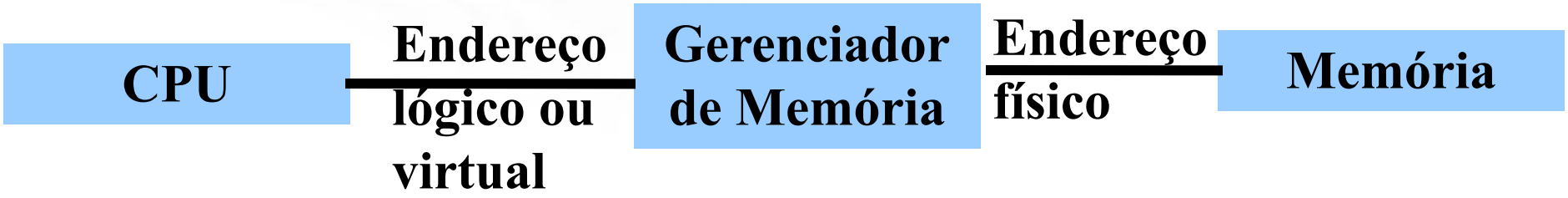
Memória Principal (cont.)

- Sistema operacional deve
  - controlar quais regiões de memória são utilizadas e por qual processo
  - decidir qual processo deve ser carregado para a memória, quando houver espaço disponível
  - alocar e desalocar espaço de memória
- Algumas funções do **Gerente de Memória**:
  - Controlar quais as unidades de memória estão ou não estão em uso, para que sejam alocadas quando necessário;
  - Liberar as unidades de memória que foram desocupadas por um processo que finalizou;
  - Tratar do Swapping entre memória principal e memória secundária.
    - Transferência temporária de processos residentes na memória principal para memória secundária, atendendo demanda do gerenciamento de processos.

Gerência de Memória

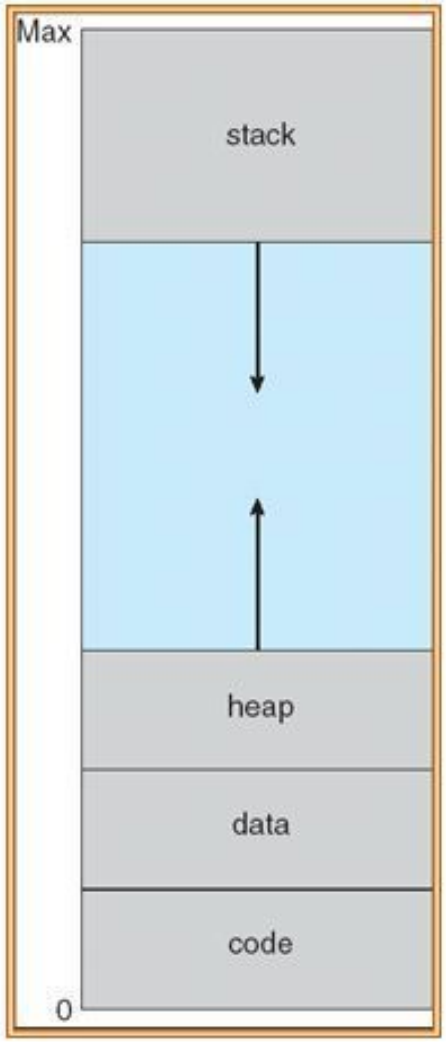
Memória Lógica / Virtual - é aquela que o processo enxerga, o processo é capaz de acessar.

Memória Física - é aquela implementada pelos circuitos integrados de memória, pela eletrônica do computador (memória real, RAM)

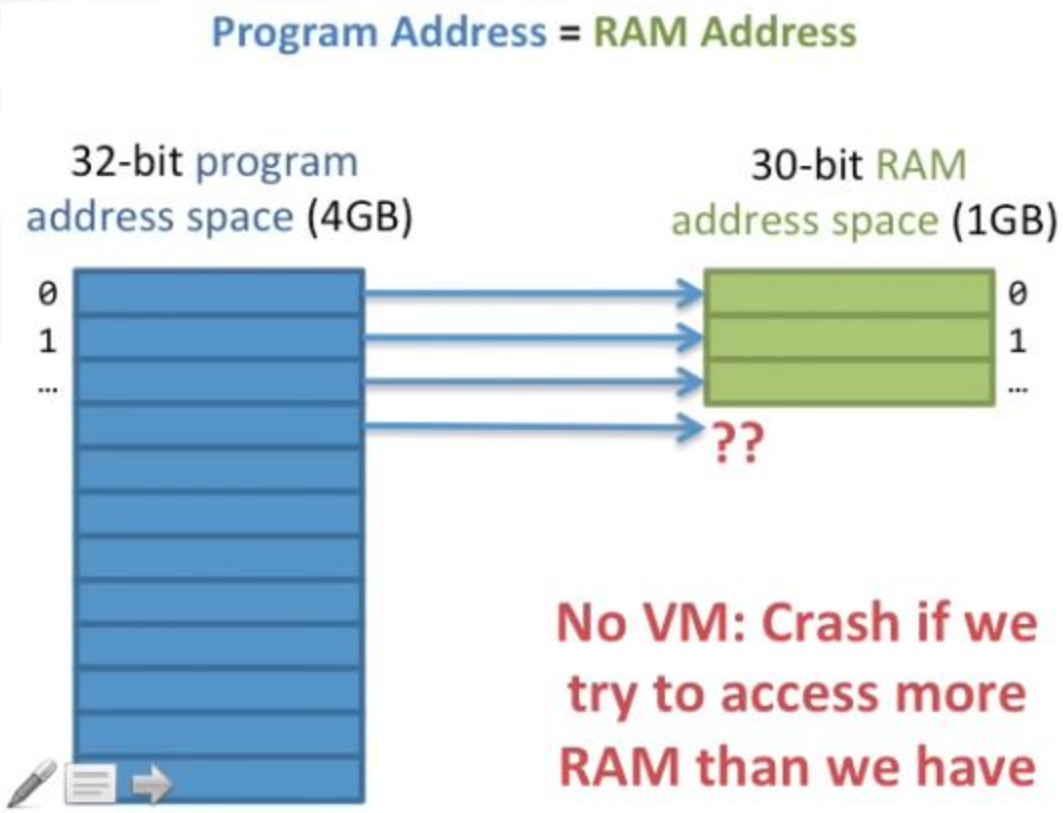


Espaço de Endereçamento Virtual de um Processo

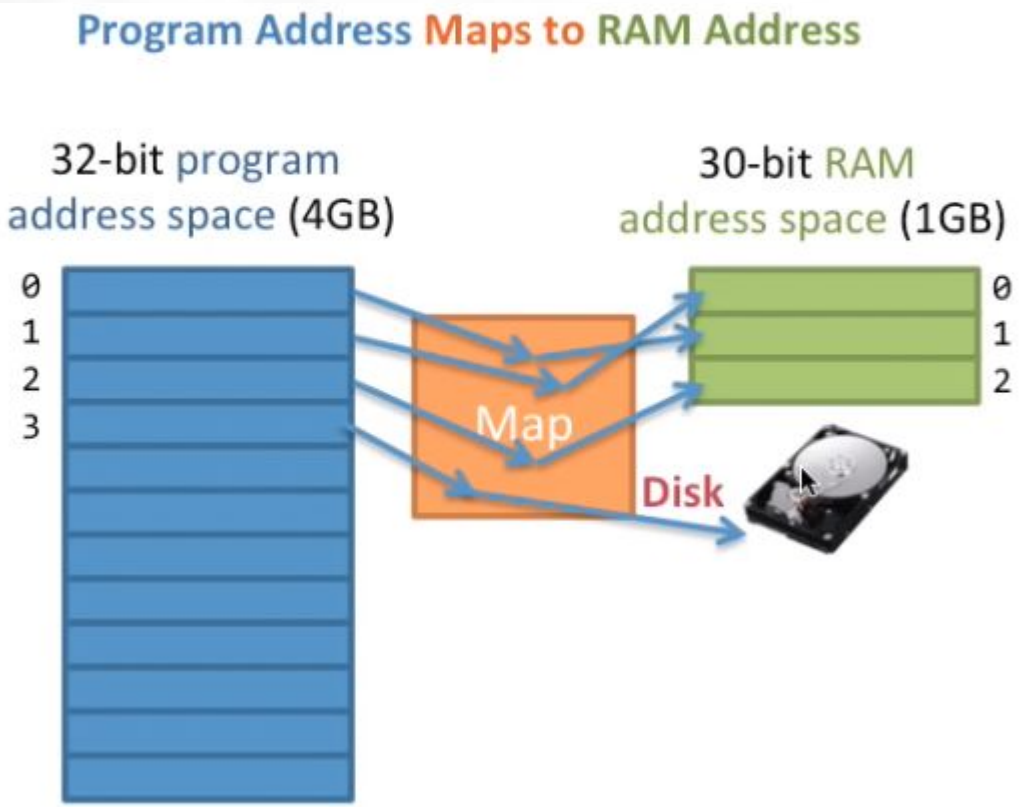
- Cada processo possui o seu próprio espaço de endereçamento, que é o Espaço de Endereçamento Virtual
- Esse espaço é contínuo, começando em 0 , indo até o limite definido pela arquitetura da CPU (ex: em maq. de 32bits, o espaço de endereçamento é 0 a “4G-1” (são 4G endereços ao total)
- Normalmente os processos não utilizam todos os endereços do seu espaço de endereçamento (procure agora no seu PC algum processo com 4Gbytes de memória!!) . A figura ao lado ilustra como isso é feito:
  - O processo é dividido em “trechos” ou segmentos (código, dados, pilhas,...) que são posicionados nesse espaço de endereçamento
  - Os demais endereços são endereços NÃO utilizados. Se o processo tentar acessar algum desses endereços o sistema acusará um erro.
    - Com isso, em um maq. de 32bits… quando um processo é alocado na memória física, ele não vai ocupar 4G de RAM… seu tamanho é calculado considerando-se apenas os endereços virtuais que são de fato utilizados
  - À medida que o processo vai alocando memória dinamicamente (ex: malloc, shmat) o SO escolhe endereços virtuais livres e vai associando aos novos trechos de memória. Essa mem. dinâmica compõe a *Heap* do processo.



SEM Memória Virtual

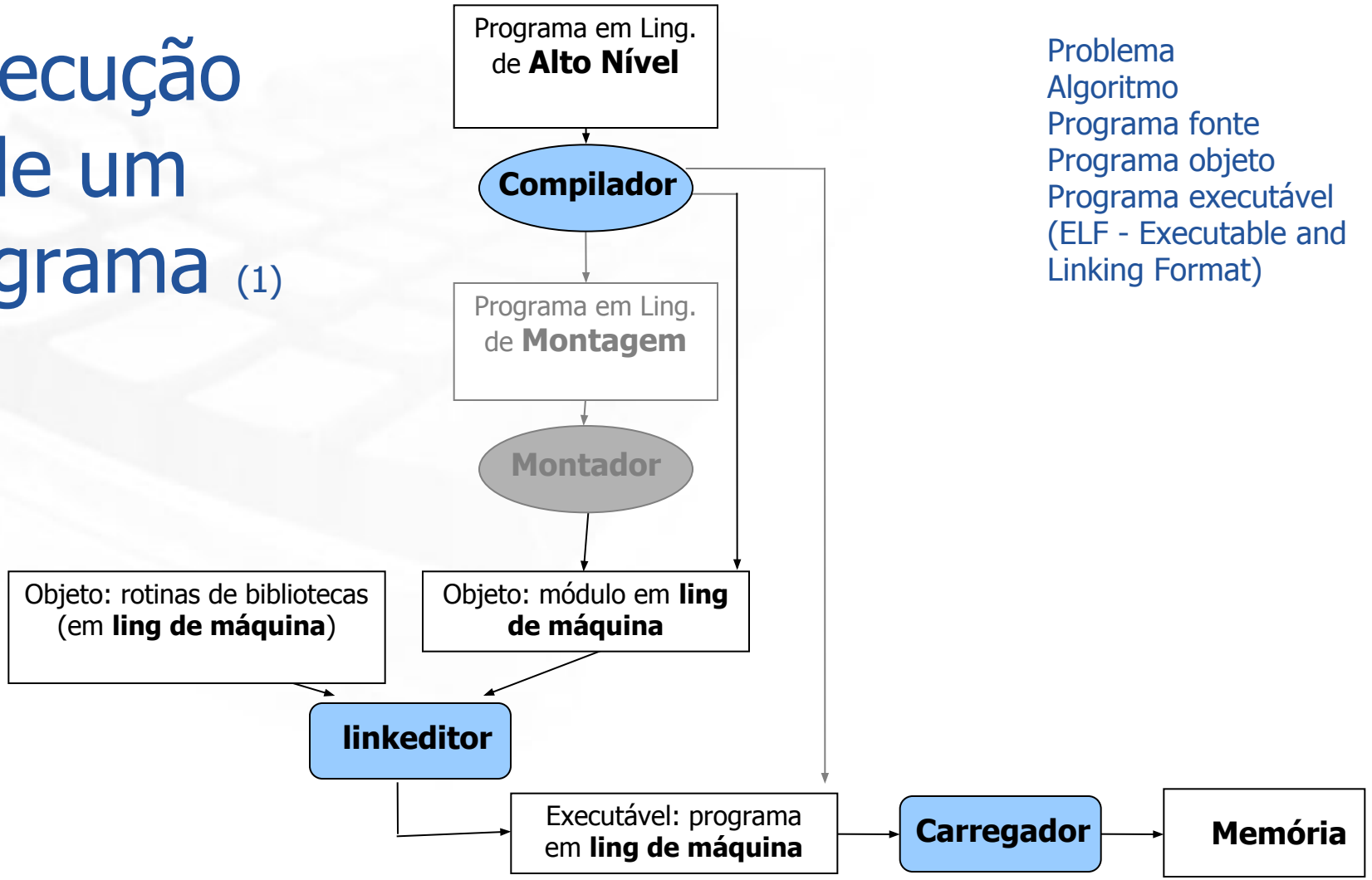


COM Memória Virtual



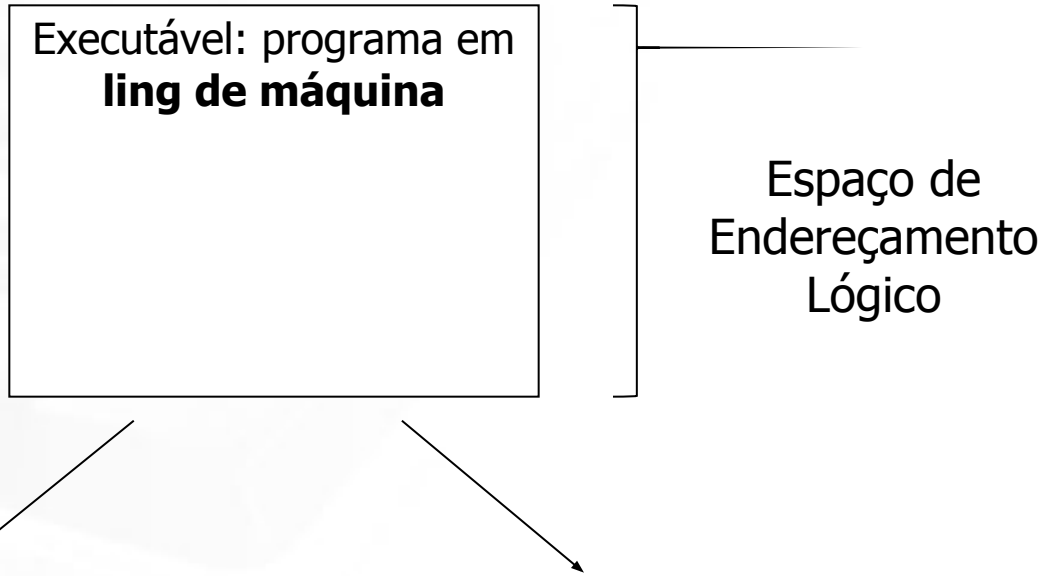
https://www.youtube.com/watch?v=qlH4-oHnBb8

Execução de um Programa (1)



Problema
Algoritmo
Programa fonte
Programa objeto
Programa executável
(ELF - Executable and Linking Format)

Espaço de Endereços de um Programa



Código absoluto:

- Endereços relativos ao início da memória (endereços reais)
- Gerado quando a localização do processo na memória é conhecida **a Priori**
- Ex: arquivos .COM do DOS

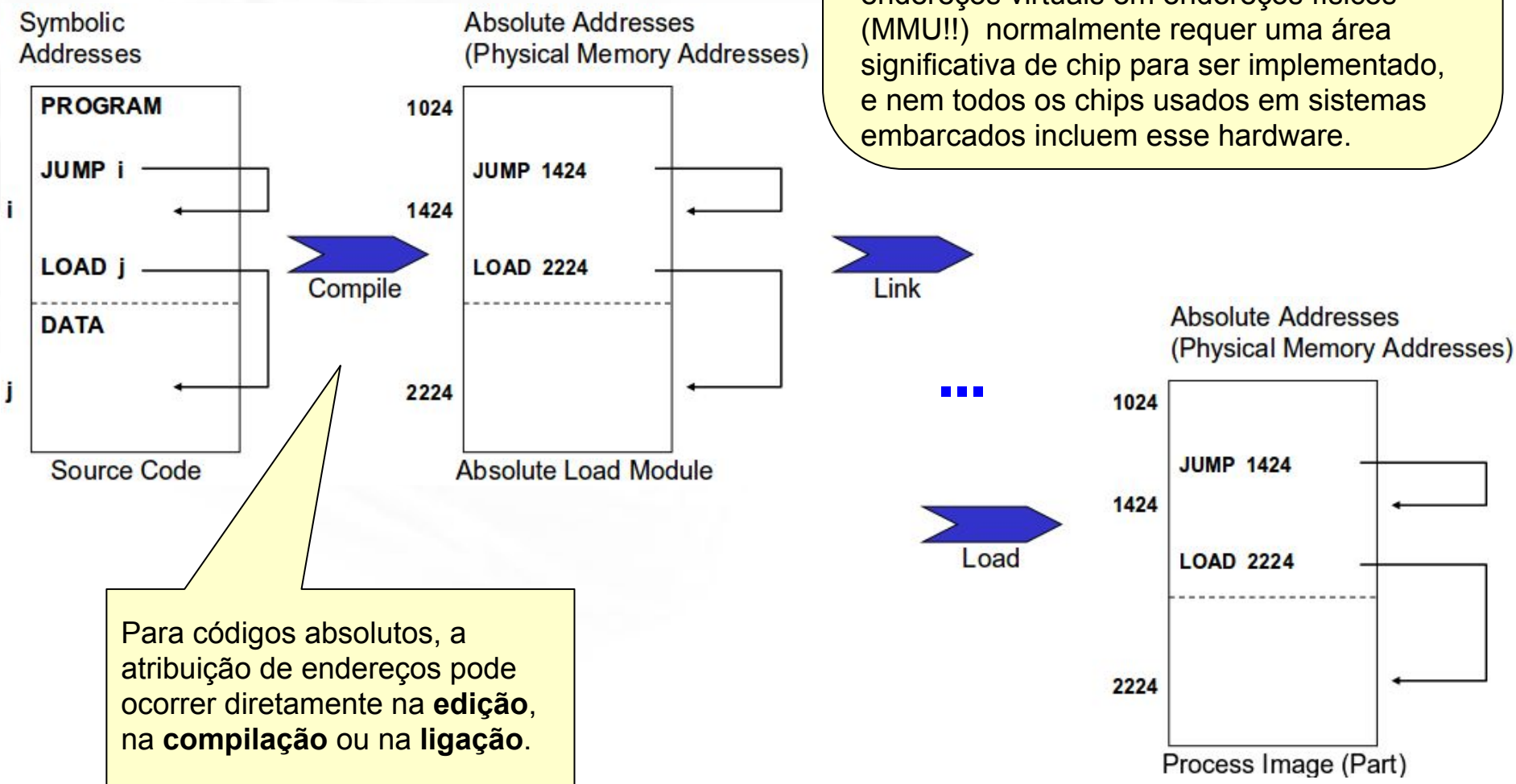
Código relocável

- O programa pode ser carregado em qualquer posição da memória.
- Deve haver uma **tradução** de endereços (ou relocação de endereços)

Código Absoluto

Aqui não há memória virtual!!

Sistemas embarcados que exigem tempos de resposta muito rápidos e/ou muito consistentes podem optar por não usar memória virtual (sistemas de memória virtual podem causar alguns indeterminismos…). Além disso, o hardware para converter endereços virtuais em endereços físicos (MMU!!) normalmente requer uma área significativa de chip para ser implementado, e nem todos os chips usados em sistemas embarcados incluem esse hardware.

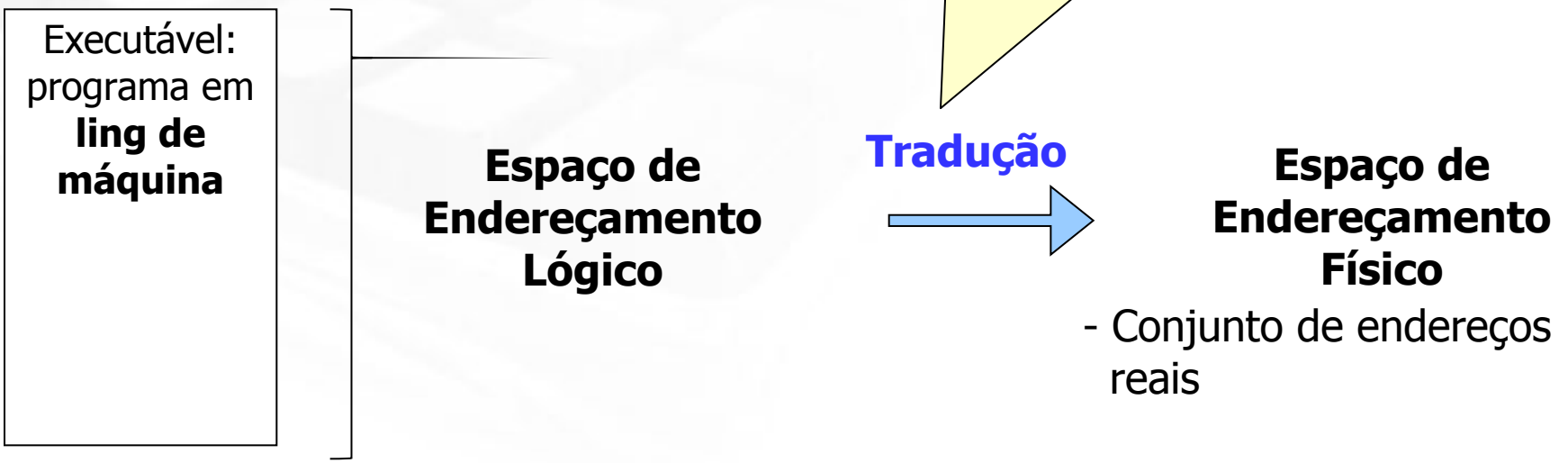


Para códigos absolutos, a atribuição de endereços pode ocorrer diretamente na **edição**, na **compilação** ou na **ligação**.

Código Relocável (1)

Relocação de Endereços Estática
A **tradução** dos endereços é feita toda de uma vez (em todo o código) quando este é **carregado** em memória (i.e. quando o processo é criado)

Relocação de Endereços Dinâmica
O código é mantido em memória contendo os endereços todos "lógicos". A **tradução** dos endereços é feita no momento em que um endereço é referenciado: ele é traduzido **em tempo real** (pela MMU) para um endereço físico.

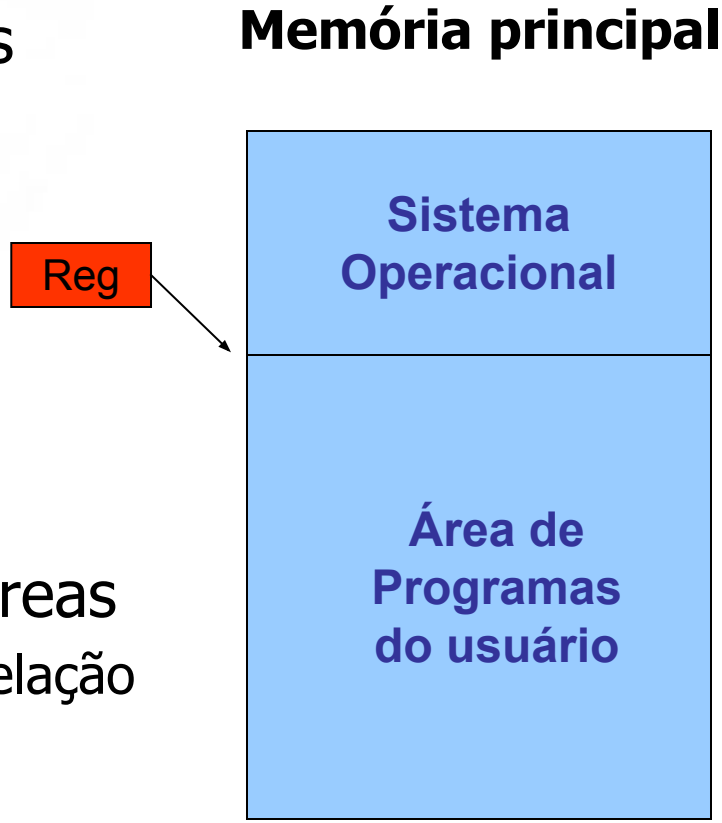


Técnicas de Gerência de Memória Real

- Alocação Contígua Simples
- Alocação Particionada
  - Partições Fixas
    - Alocação Particionada Estática;
  - Partições Variáveis
    - Alocação Particionada Dinâmica.

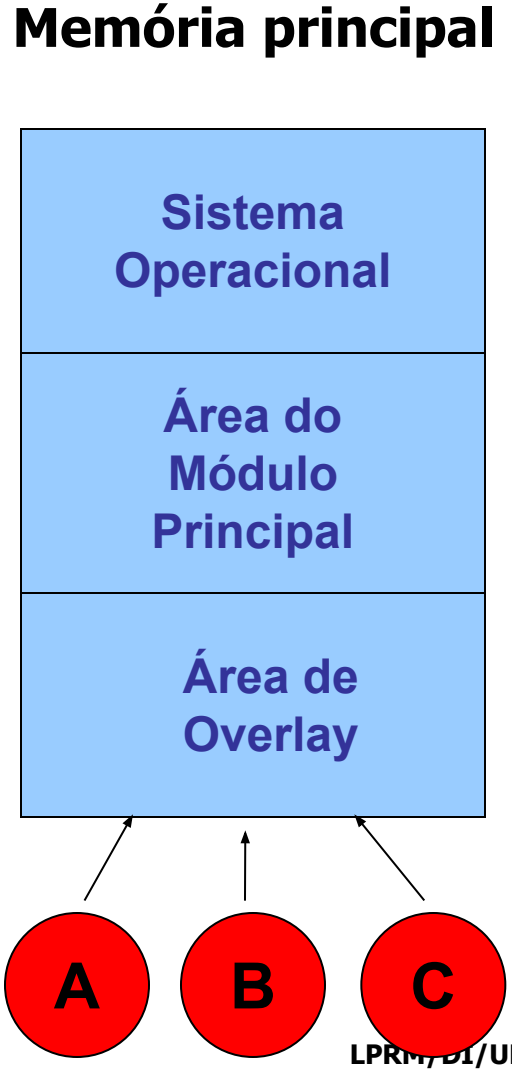
Alocação Contígua Simples (1)

- Implementada nos primeiros sistemas
  - Ainda usada nos monoprogramáveis
- Memória é dividida em duas áreas:
  - Área do Sistema Operacional
  - Área do Usuário
- Um usuário não pode usar uma área maior do que a disponível
- Registrador de proteção delimita as áreas
  - Sistema verifica acessos à memória em relação ao valor do registrador;
- Simples, mas não permitia utilização eficiente de processador/memória



Alocação Contígua Simples (2)

- Limitados pelo tamanho da memória principal disponível...
- Solução: Overlay
  - Dividir o programa em módulos;
  - Permitir execução independente de cada módulo, usando a mesma área de memória;
- Área de Overlay
  - Área de memória comum onde módulos compartilham mesmo espaço.



Alocação Particionada

- Multiprogramação.
  - Necessidade do uso da memória por vários usuários simultaneamente.
- Ocupação mais eficiente do processador;
- **Alocação Particionada Estática=>Partições fixas**
  - Memória dividida em pedaços de tamanho fixo chamados partições;
- O tamanho de cada partição era estabelecido na inicialização do sistema;
- Para alteração do particionamento, era necessário uma nova inicialização com uma nova configuração.

Alocação Particionada Estática (1)

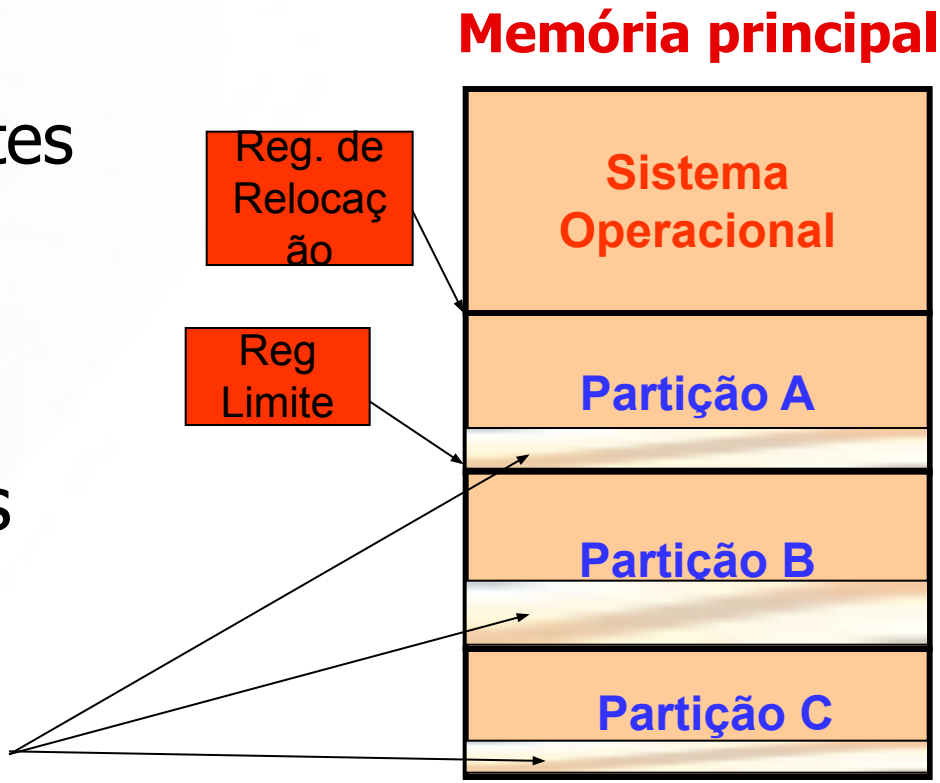
- Partições fixas
  - Tamanho fixo ; número de partições fixo

a) Alocação Particionada Estática Absoluta:
   - Código absoluto;
   - Programas exclusivos para partições específicas.
   - Simples de gerenciar
   - E se todos os processos só pudessem ser executados em uma mesma partição

b) Alocação Particionada Estática Relocável:
   - Código relocável
   - Programas podem rodar em qualquer partição

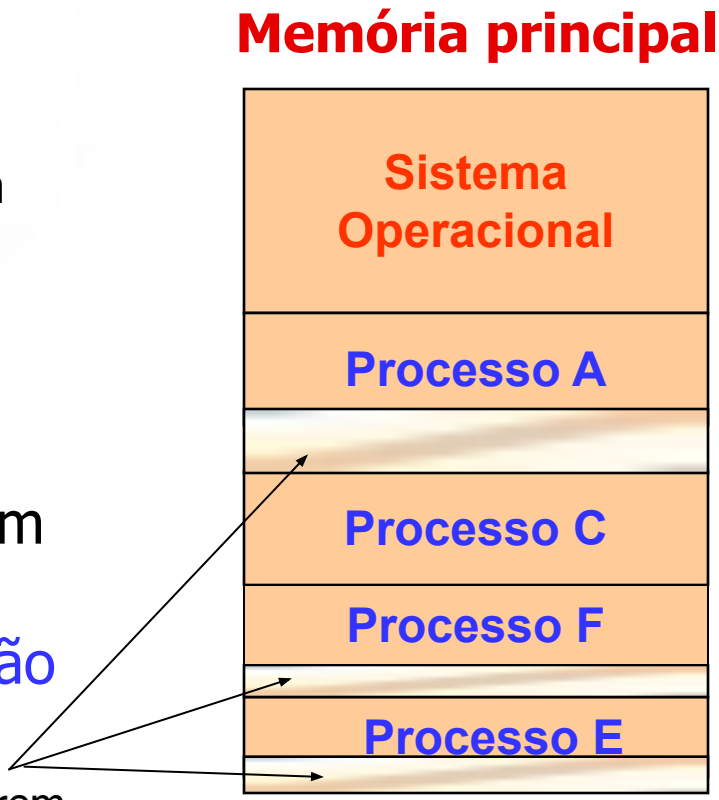
Alocação Particionada Estática (2)

- Proteção:
  - Registradores com limites inferior e superior de memória acessível.
- Programas não ocupam totalmente o espaço das partições, gerando uma fragmentação interna.

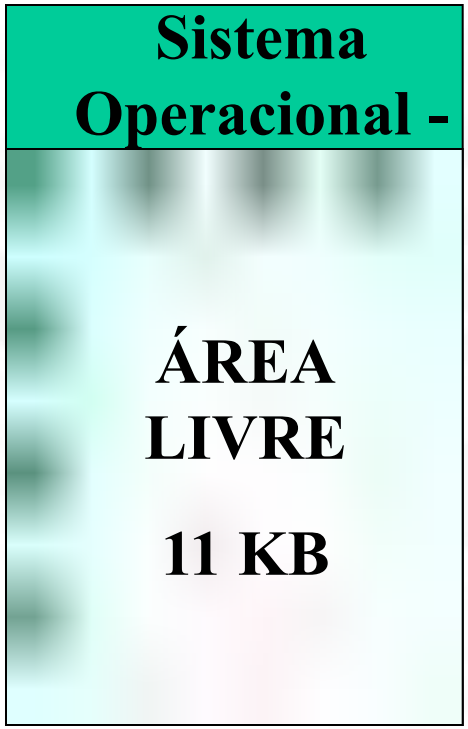


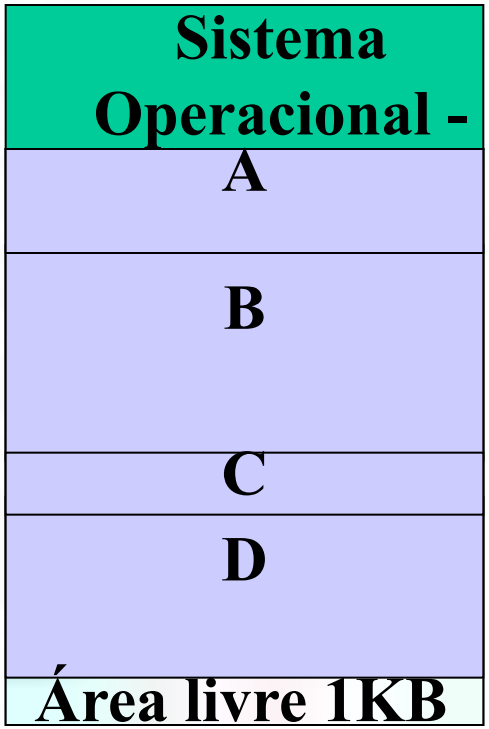
Alocação Particionada Dinâmica (1)

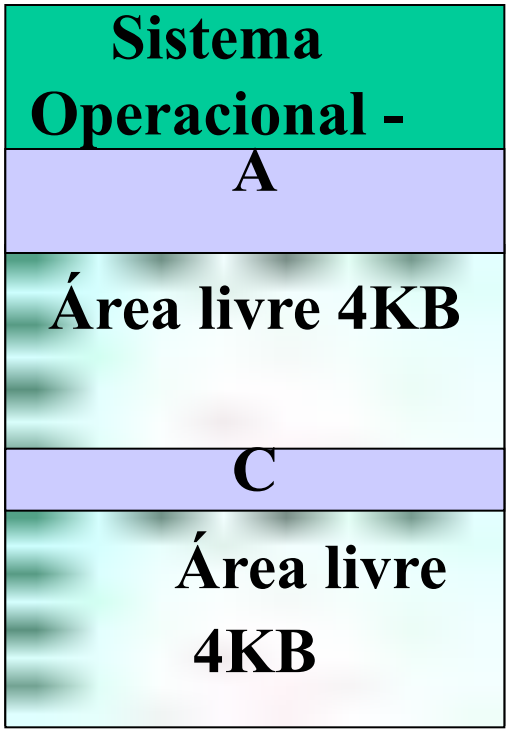
- Não existe realmente o conceito de partição dinâmica.
  - O espaço utilizado por um programa é a sua partição.
- Não ocorre fragmentação interna.
  - o tamanho da memória alocada é igual ao tamanho do programa
- Ao terminarem, os programas deixam espalhados espaços pequenos de memória, provocando a fragmentação externa.
  - os fragmentos são pequenos demais para serem reaproveitados

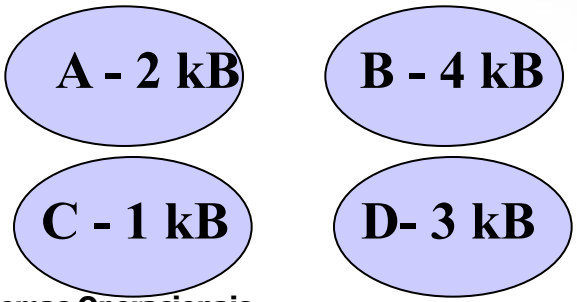


Alocação Particionada Dinâmica (2)







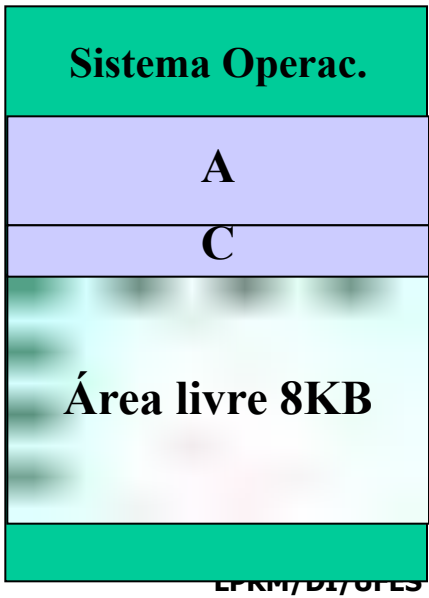


E - 6 kB ?

Fragmentação externa!

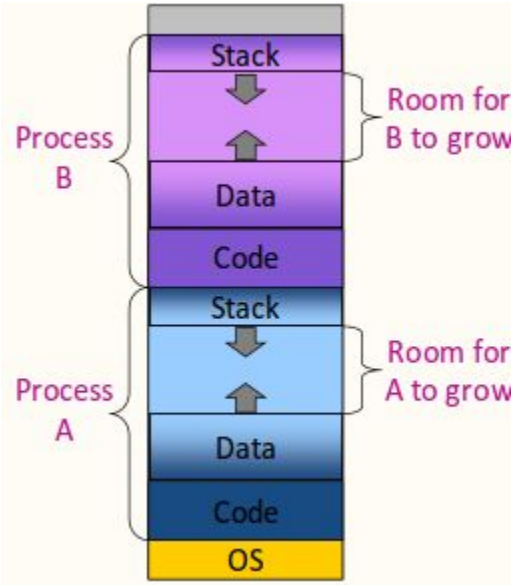
Alocação Particionada Dinâmica (3)

- Soluções:
  - Reunião dos espaços contíguos.
  - Realocar todas as partições ocupadas eliminando espaços entre elas e criando uma única área livre contígua-> Relocação Dinâmica de endereços:
    - Movimentação dos programas pela memória principal.
    - Resolve o problema da fragmentação
    - Consome recursos do sistema
      - Processador, disco, etc.
    - Proteção
      - Não correção ou correção errada implica em acesso a outra partição



Alocação Particionada Dinâmica (4)

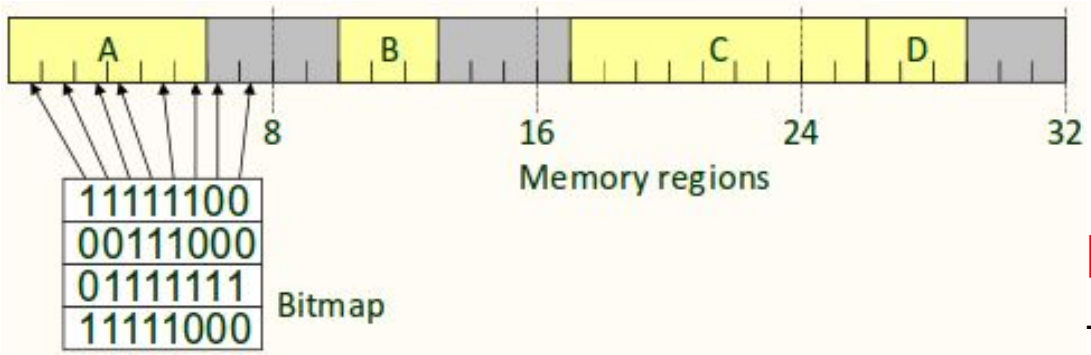
- Definição do tamanho das partições pode ser difícil
  - Processos crescem quando em execução
  - É bom definir áreas extra para dados e pilhas



- Como gerenciar as partições alocáveis de memória ?
  - Mapamento de bits
  - Mapeamento da Memória com listas encadeadas

Gerenciamento de Espaço Livre - Mapa de bits

- Usado para o gerenciamento com alocação dinâmica
- Memória é dividida em unidades de alocação
  - De algumas palavras a vários kilobytes
    - Qto menor → maior o mapa de bits
    - Qto maior → desperdiço na última unidade
- A cada unidade é associado um bit que descreve a disponibilidade da unidade



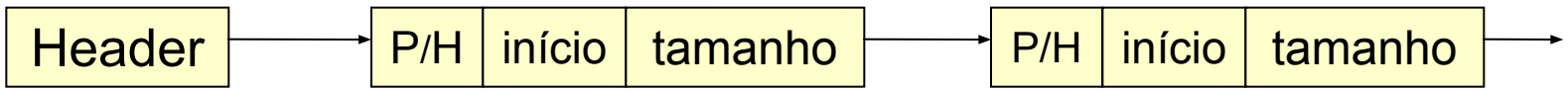
Principal problema:

- Busca de k zeros consecutivos para alocação de k unidades
- Raramente é utilizado atualmente (Muito lenta!)

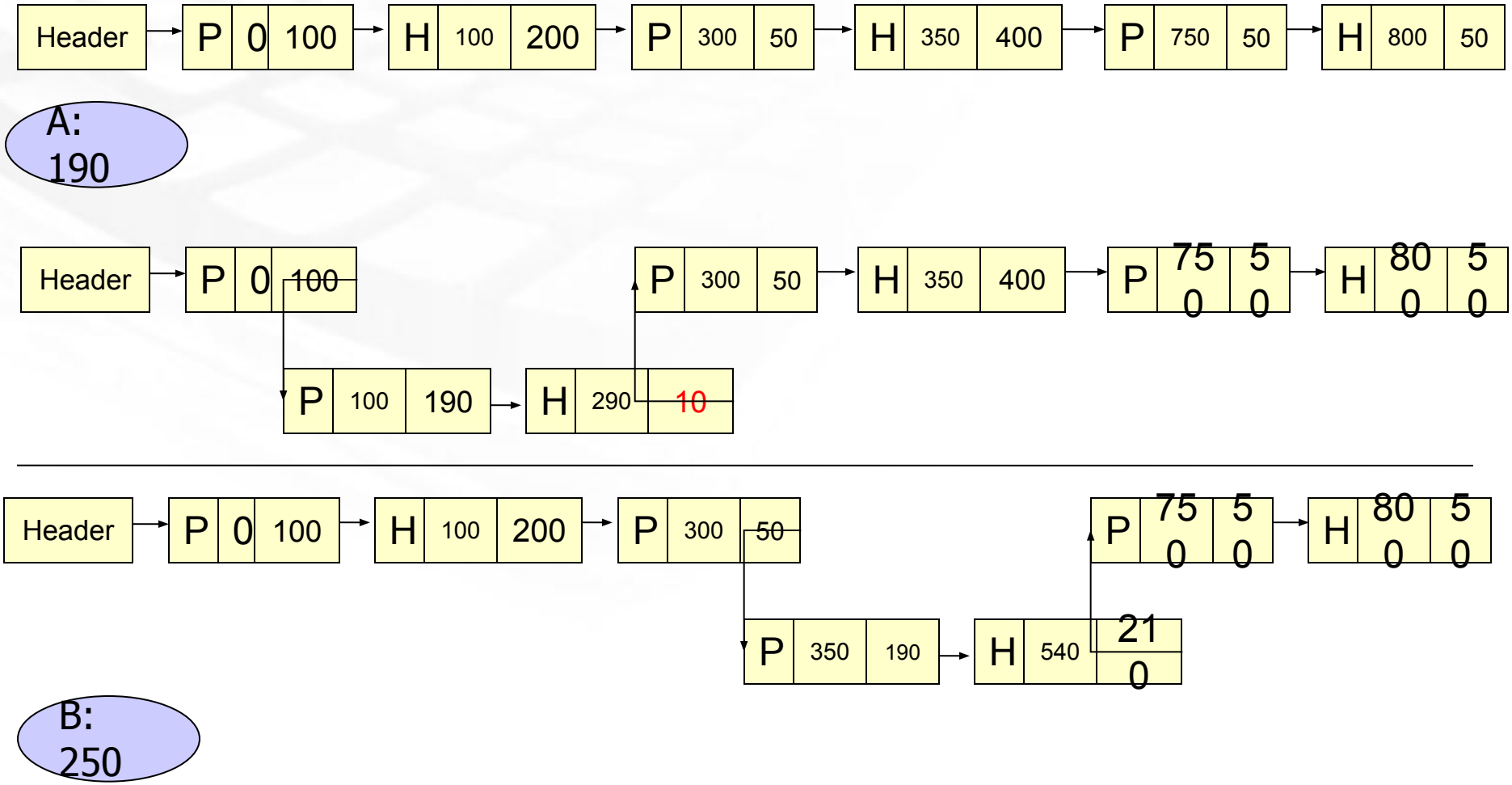
Gerenciamento de Espaço Livre - Mapeamento da Memória com lista encadeada (1)

- Lista ligada de segmentos alocados ou livres
- Um segmento é uma área de memória alocada ou livre
- Cada elemento da lista indica
  - Estado do segmento (P) Alocado por um processo ou (H) *Buraco* livre
  - Unidade em que inicia
  - Tamanho em unidades
- Lista duplamente encadeada facilita de concatenação de segmentos
- Lista ordenada por endereço permite vários algoritmos de alocação

P: Process
H: Hole (*buraco*)



Gerenciamento de Espaço Livre - Mapeamento da Memória com lista encadeada (2)



A escolha da partição ideal (1)

- Existem 4 maneiras de percorrer a lista de espaços livre atrás de uma lacuna de tamanho suficiente, são eles:
  - Best-fit (utiliza a lacuna que resultar a menor sobra)
    - Espaço mais próximo do tamanho do processo;
    - Tempo de busca grande;
    - Provoca fragmentação.
  - Worst-Fit (utiliza a lacuna que resultar na maior sobra):
    - Escolhe o maior espaço possível;
    - Tempo de busca grande;
    - Não apresenta bons resultados.

A escolha da partição ideal (2)

- First-Fit (primeira alocação):
  - utiliza a primeira lacuna que encontrar com tamanho suficiente
  - Melhor performance.
- Circular-fit ou Next-Fit (próxima alocação):
  - como first-fit mas inicia a procura na lacuna seguinte a última sobra
  - Performance inferior ao First-Fit.

A escolha da partição ideal (3)

- Considerações sobre Mapeamento da Memória com listas ligadas :
  - Todos melhoram em performance se existirem listas distintas para processos e espaços, embora o algoritmo fique mais complexo.
  - Listas ordenadas por tamanho de espaço melhoram a performance.