



Laboratório de Pesquisa em Redes e Multimídia

Gerência de Memória



Universidade Federal do Espírito Santo
Departamento de Informática

Introdução

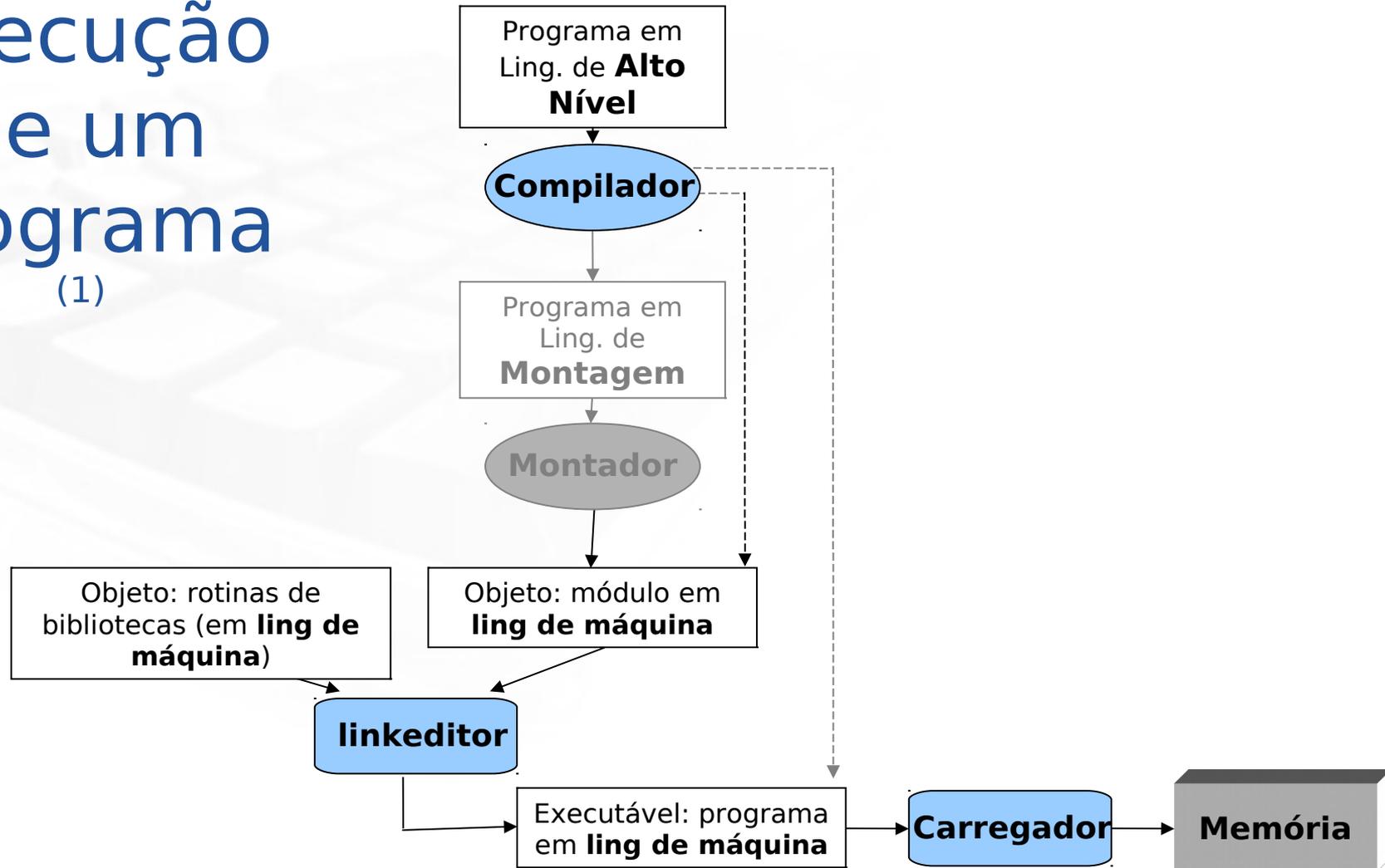
- Considerações:
 - Recurso **caro** e **escasso**;
 - Programas só executam se estiverem na memória principal;
 - Quanto mais processos **residentes** na memória principal, melhor será o **compartilhamento** do processador;
 - Necessidade de uso otimizado;
 - O S.O. não deve ocupar muita memória;
 - “É um dos fatores mais importantes em um projeto de S.O.”.

Introdução

- 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 **Gerenciador 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.

Execução de um Programa

(1)



Execução de um Programa (2)

Executável: programa em **ling de máquina**

Espaço de Endereçamento Lógico

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

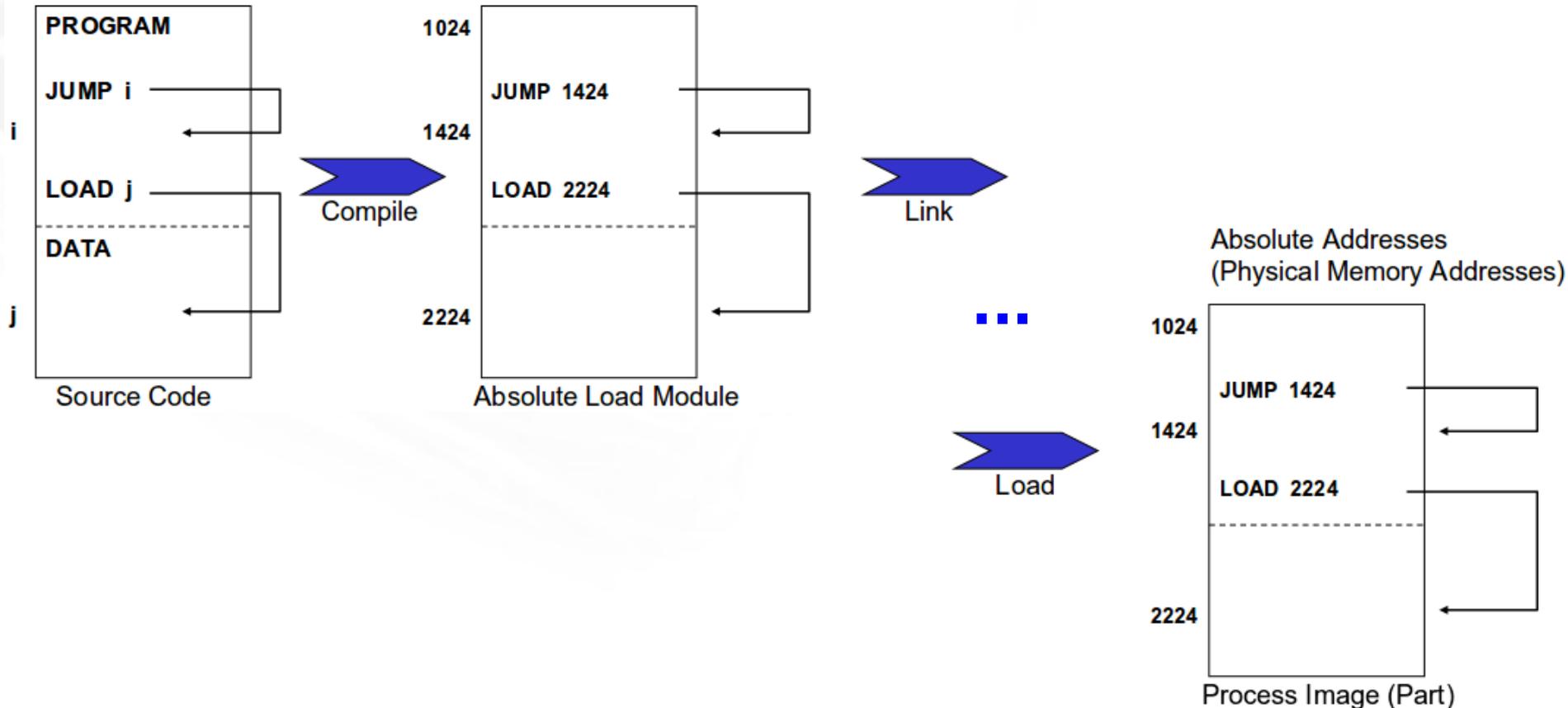
Código relocável

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

Código Absoluto

Symbolic
Addresses

Absolute Addresses
(Physical Memory Addresses)



Código Relocável (1)

Executável:
programa
em **ling de
máquina**

**Espaço de
Endereçamento
Lógico**

Tradução



**Espaço de
Endereçamento
Físico**

- Conjunto de endereços
reais

Código Relocável (2)

- Relocação de Endereços **Estática**

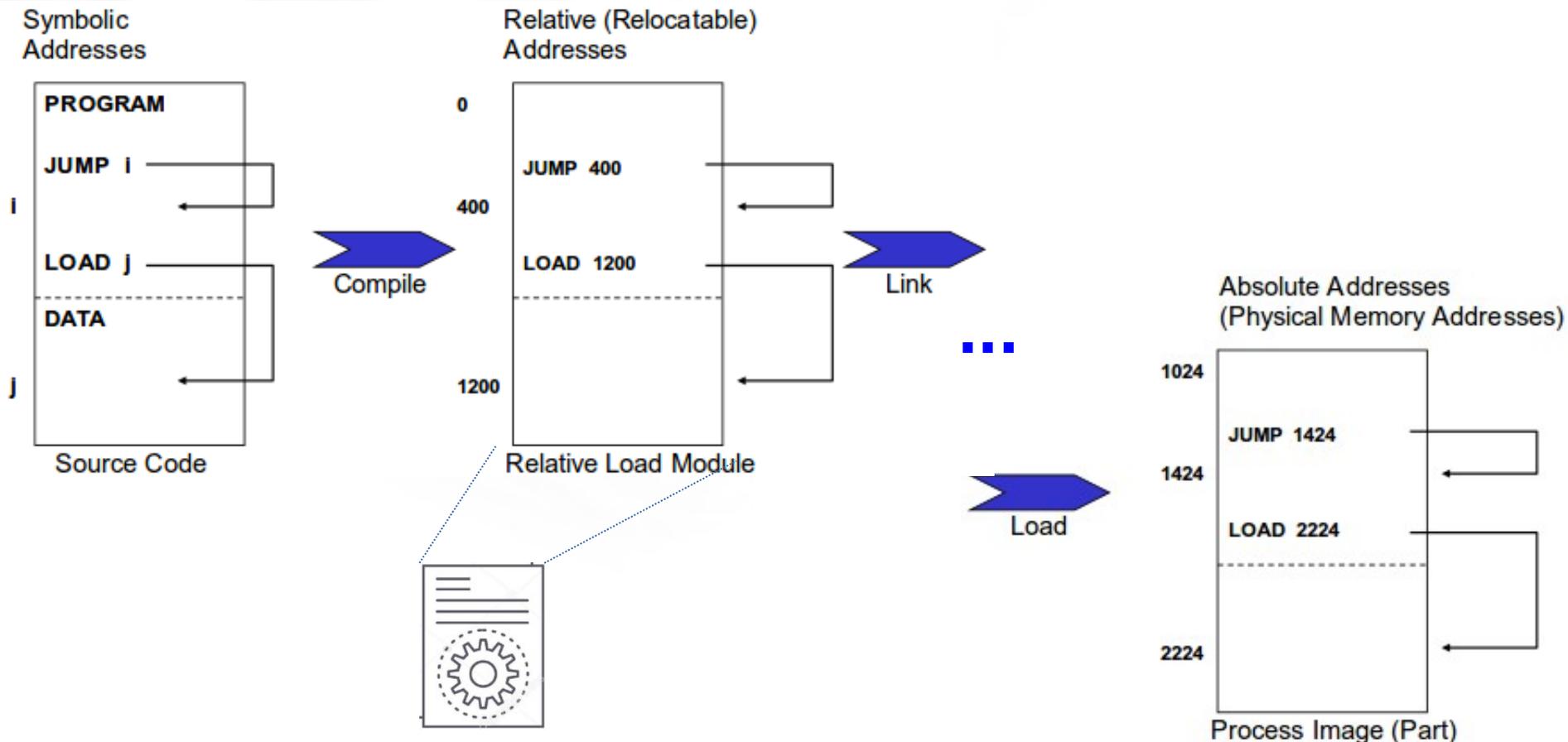
O Loader (em tempo de carga) reloca os endereços das instruções relocáveis (ex: JMP endx)

- Relocação de Endereços **Dinâmica**

- O Loader (em tempo de carga) reloca os endereços das instruções relocáveis (ex: JMP endx)
- Em tempo de execução
- O processo pode ser movimentado dentro da memória física
- Um hardware especial deve estar disponível para que funcione (MMU)

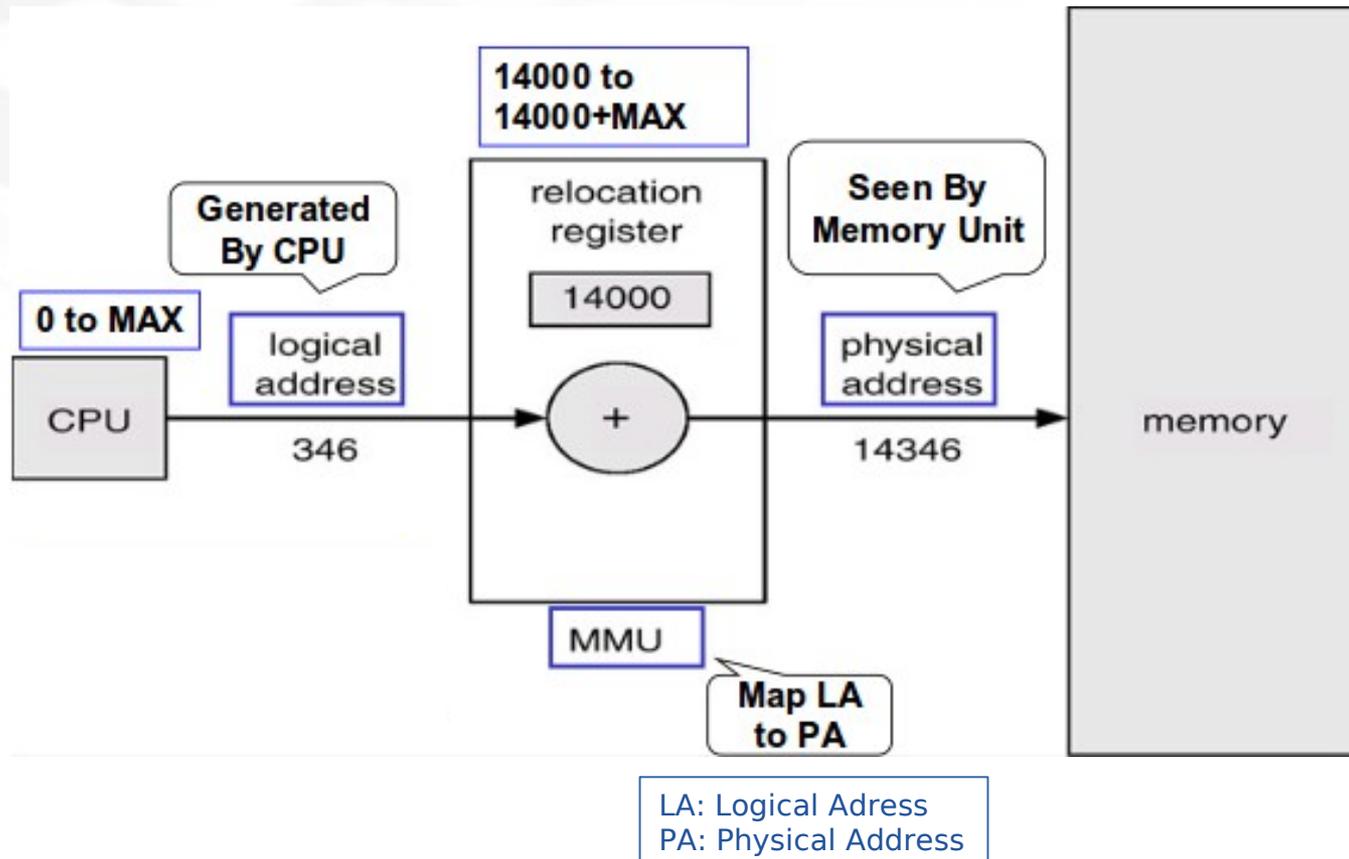
Código Relocável (3)

- Relocação Estática



Código Relocável (4)

- Relocação Dinâmica



Gerência de Memória

Memória Lógica - 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)

CPU

Endereço
lógico ou
virtual

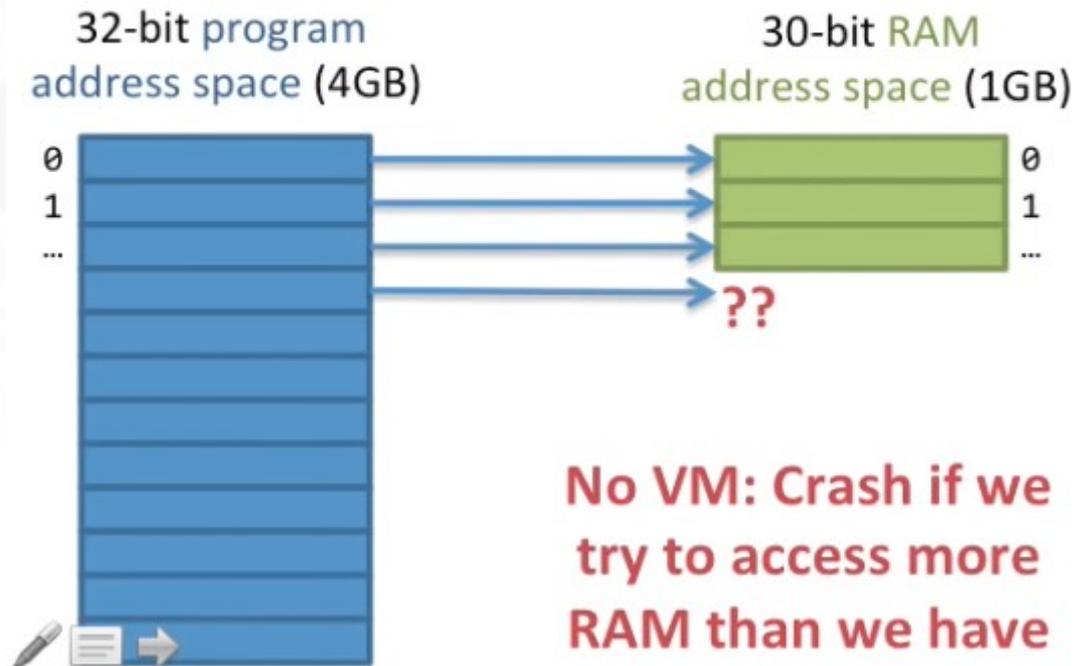
Gerenciador
de Memória

Endereço
físico

Memória

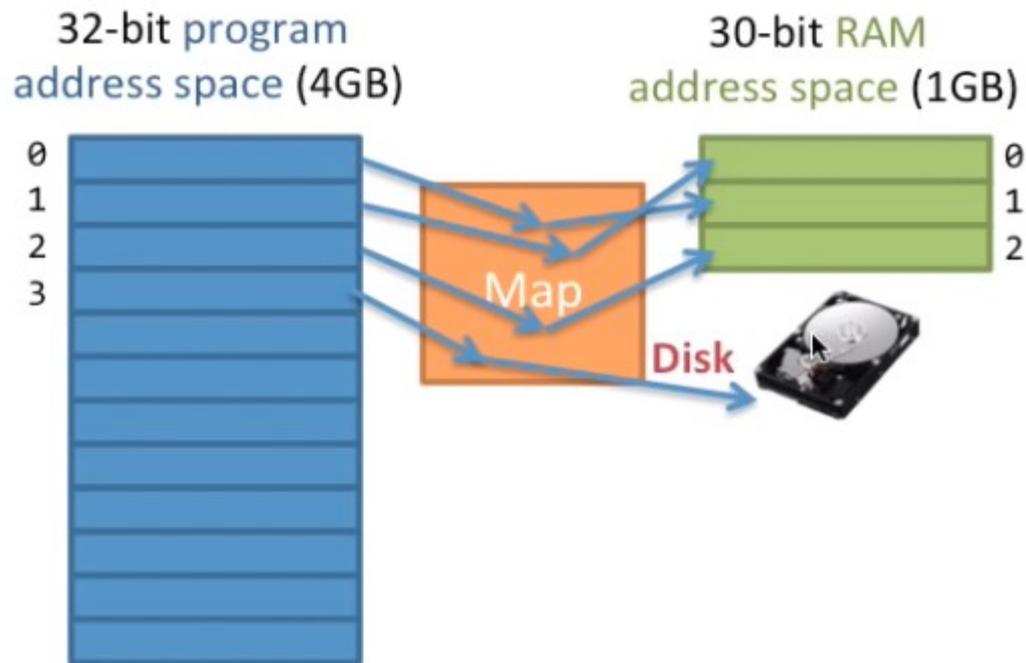
SEM Memória Virtual

Program Address = RAM Address



COM Memória Virtual

Program Address **Maps** to RAM Address



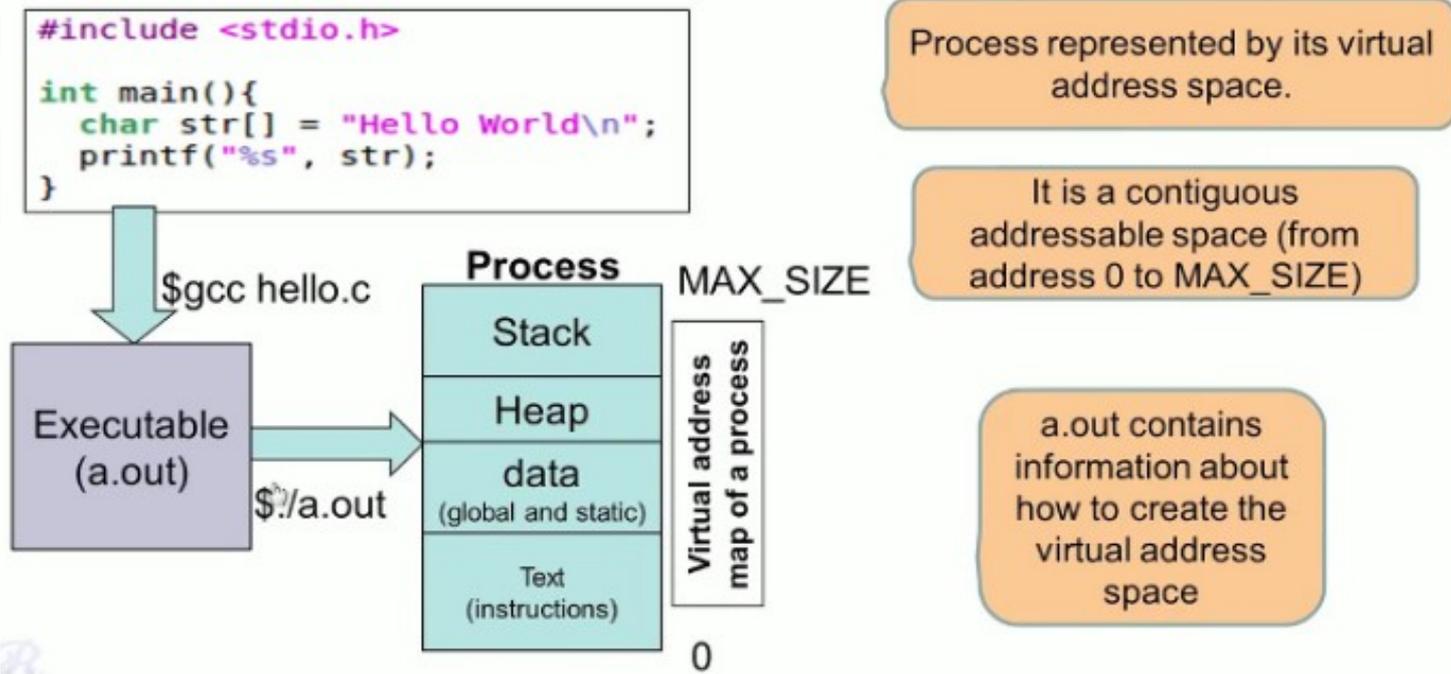
<https://www.youtube.com/watch?v=qIH4-oHnBb8>





- Espaço de Endereçamento do Processo
- Técnicas de Gerência de Memória Real
- Gerenciamento de espaços livres
- Pagninação/Segmentação

Espaço de Endereçamento Virtual de um Processo

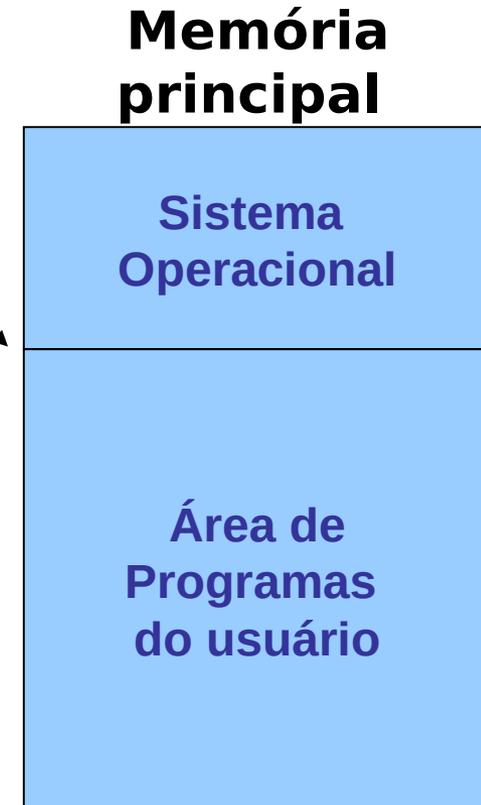


Técnicas de Gerência de Memória Real

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

Alocação Contígua Simples (1)

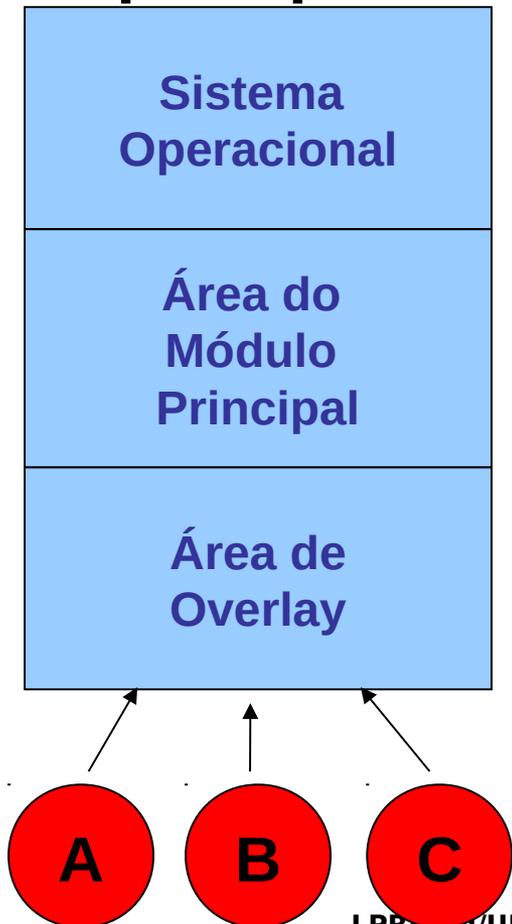
- Implementada nos primeiros sistemas
 - Ainda usada nos monoprogramáveis
- Memória é dividida em duas áreas **Reg**
 - Área do Sistema Operacional
 - Área do Usuário
- Um usuário não pode usar uma área maior do que a disponível
- Registrador de proteção delimita as áreas
 - Sistema verifica acessos 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.

Memória principal



Alocação Particionada

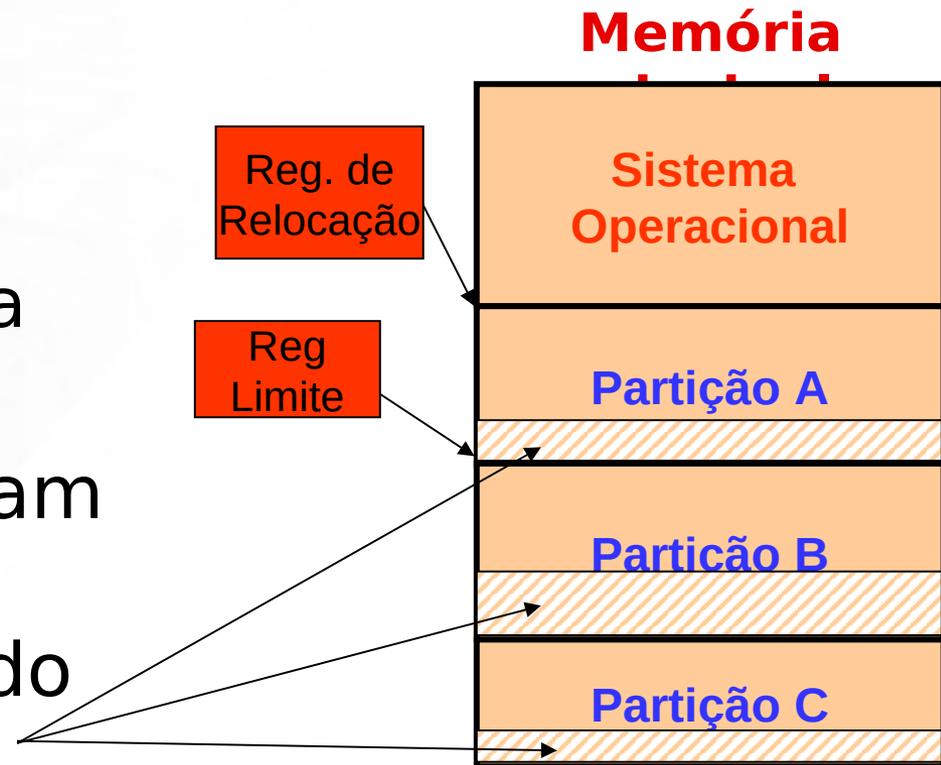
- **Multiprogramação.**
 - Necessidade do uso da memória por vários usuários simultaneamente.
- Ocupação mais eficiente do processador;
- **Alocação Particionada Estática=>Partições fixas**
 - Memória dividida em pedaços de tamanho fixo chamados **partições**;
- O tamanho de cada partição era estabelecido 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

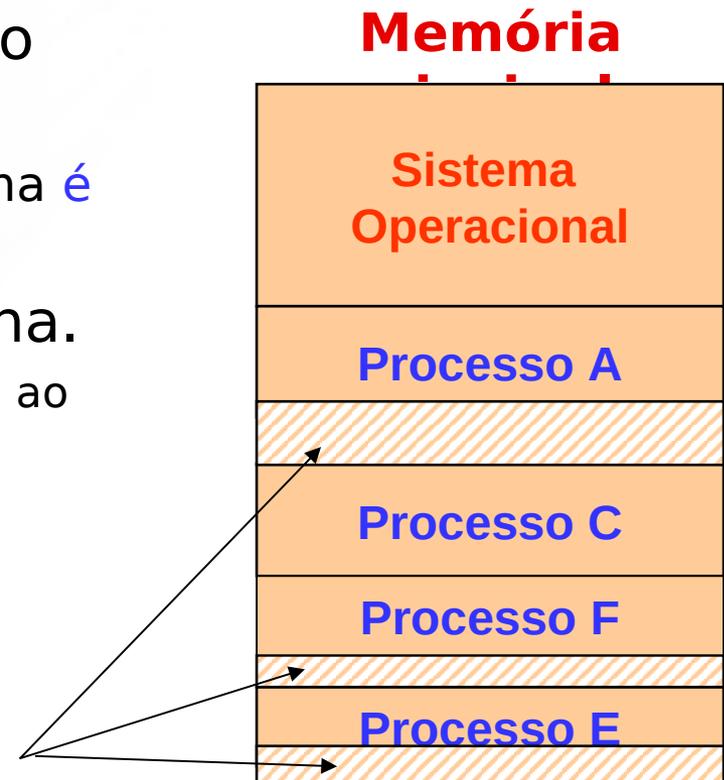
Alocação Particionada Estática (2)

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

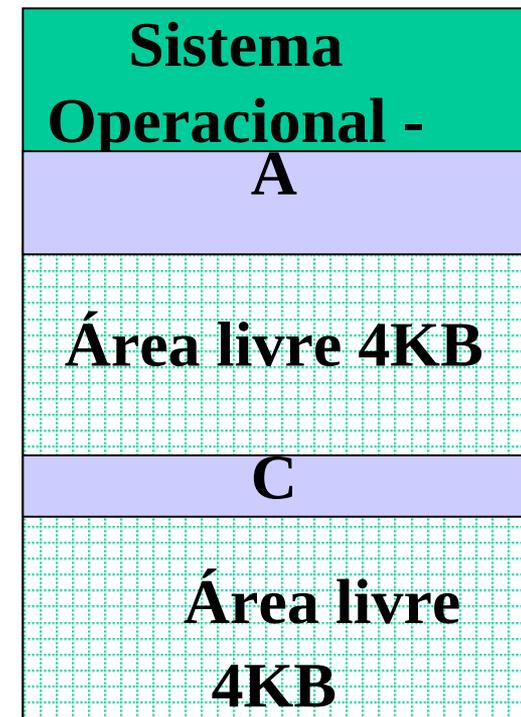
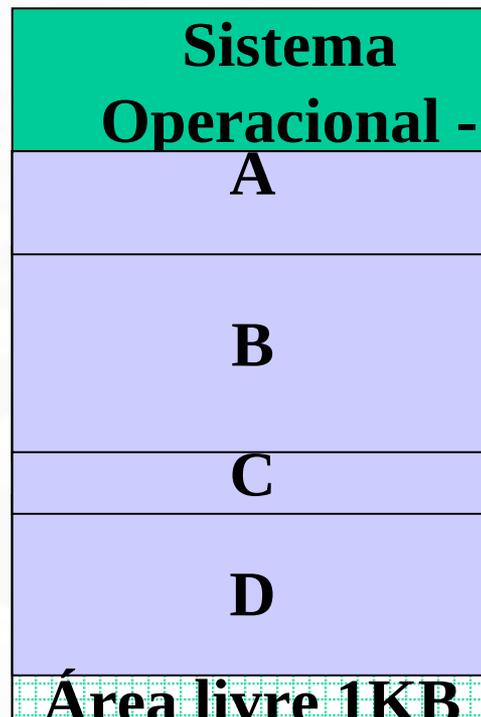
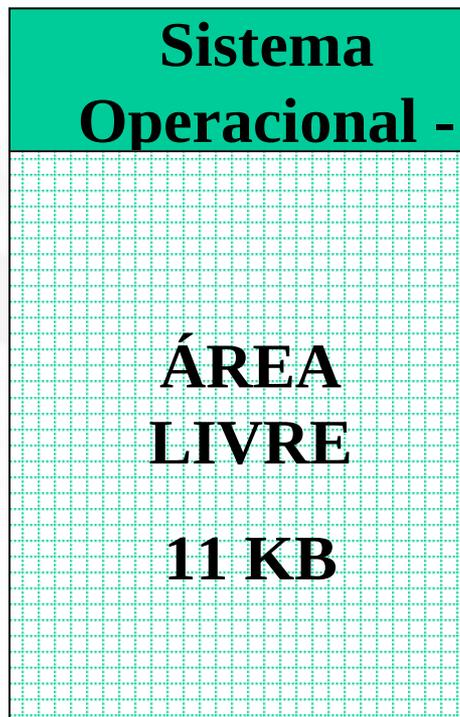


Alocação Particionada Dinâmica (1)

- 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)



A - 2 kB

B - 4 kB

C - 1 kB

D - 3 kB

E - 6 kB

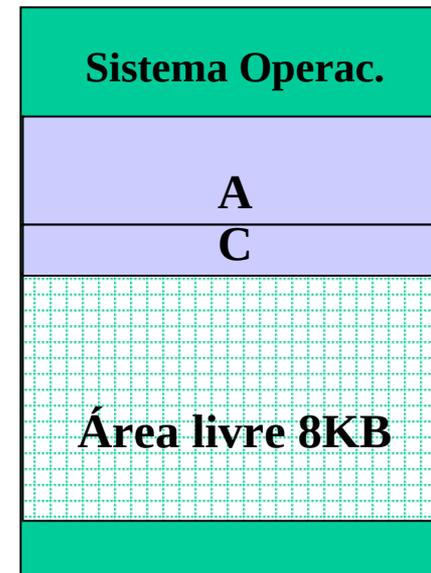
?

Fragmentação externa!

Alocação Particionada Dinâmica (3)

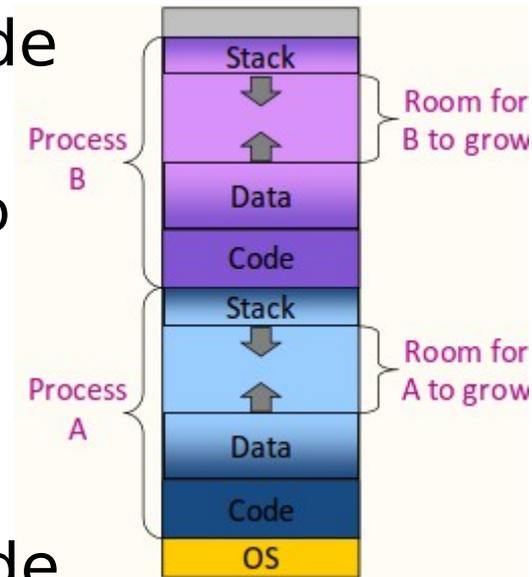
■ Soluções:

- **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



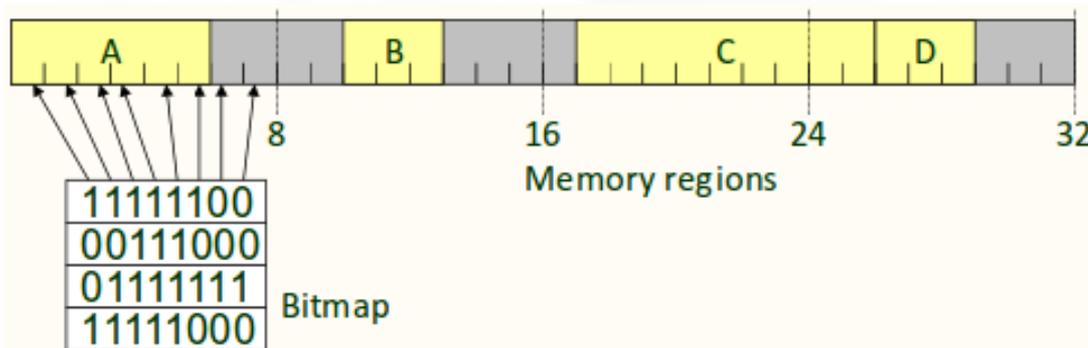
Alocação Particionada Dinâmica (4)

- 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 ?
 - Mapeamento de bits
 - Mapeamento da Memória com listas encadeadas



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

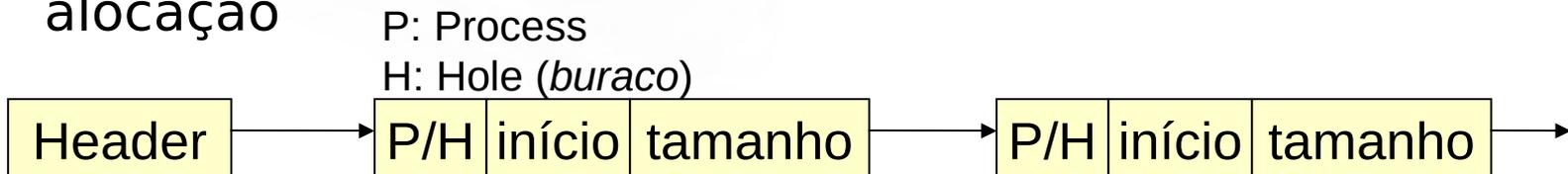


Principal problema:

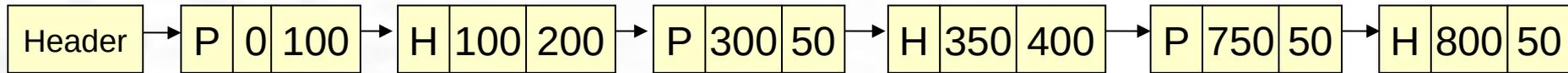
- Busca de k zeros consecutivos para alocação de k unidades
- Raramente é utilizado atualmente (Muito lenta!)

Mapeamento da Memória com lista encadeada (1)

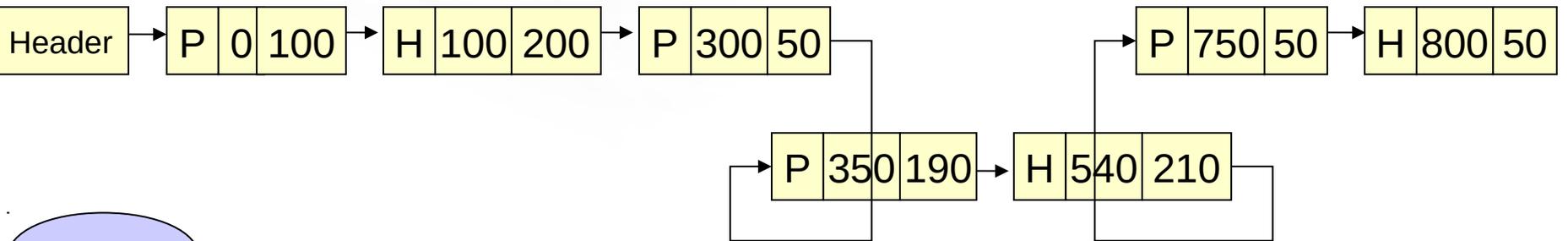
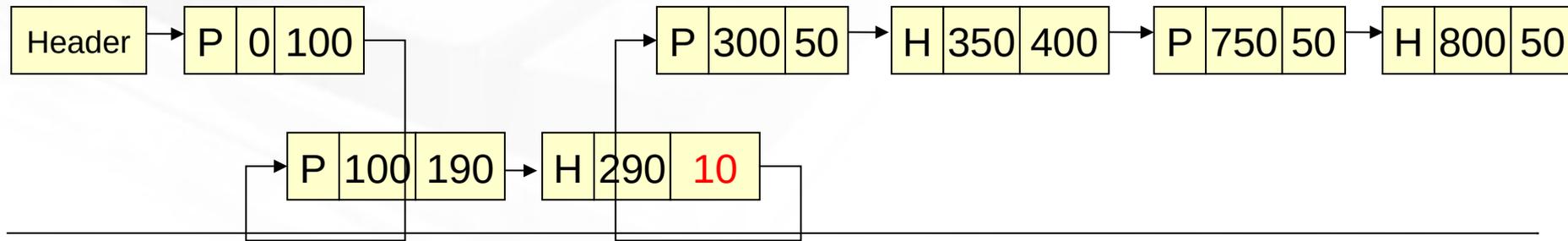
- Lista ligada de segmentos alocados ou livres
- Um segmento é uma área de memória alocada ou livre
- Cada elemento da lista indica
 - Estado do segmento (P) Alocado por um processo ou (H) *Buraco* livre
 - Unidade em que inicia
 - Tamanho em unidades
- Lista duplamente encadeada facilita de concatenação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



Mapeamento da Memória com lista encadeada (2)



A: 190



B: 250

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.