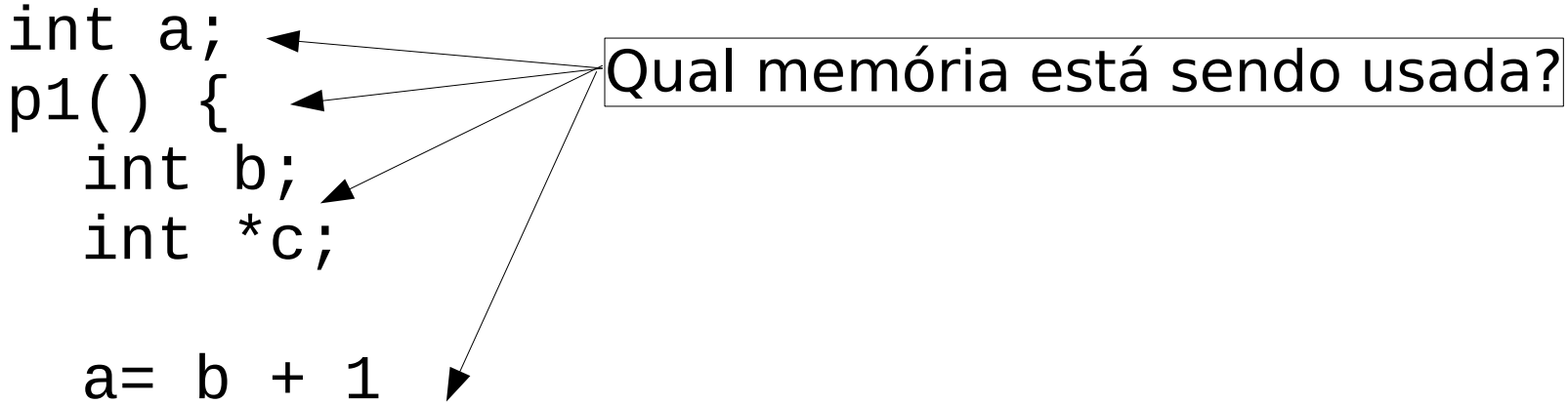
Aula 11

Gerência de Memória

1.1 Porquê?
1.2 Tipos de memória
1.3 Ligação (binding) estática e dinâmica
1.4 Fragmentação
1.5 Referências: Capítulo 9 (9.1 a 9.3)

Memória? Que Memória? (1)

Veja o seguinte trecho de programa:

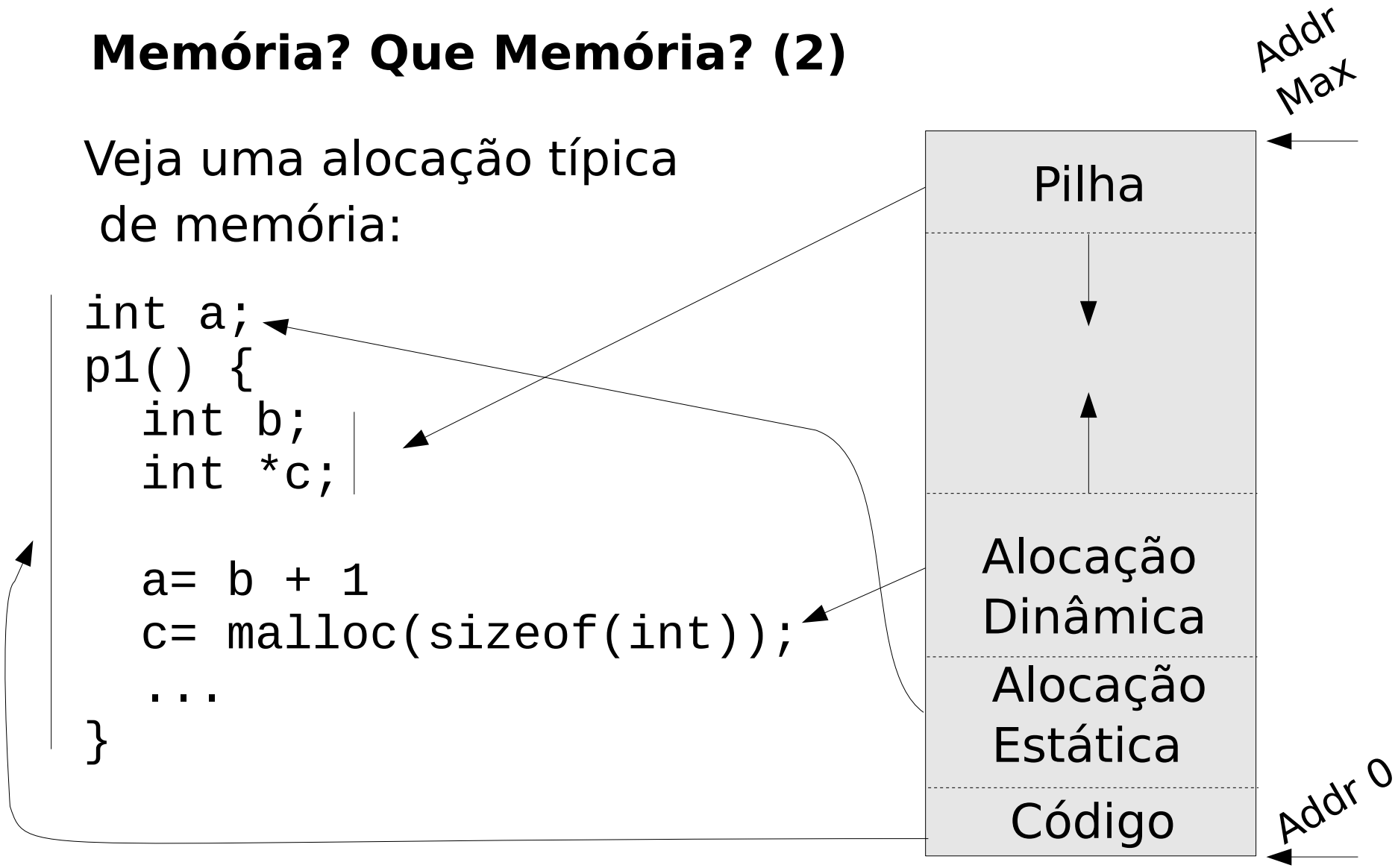


Memória? Que Memória? (2)

Veja uma alocação típica de memória:

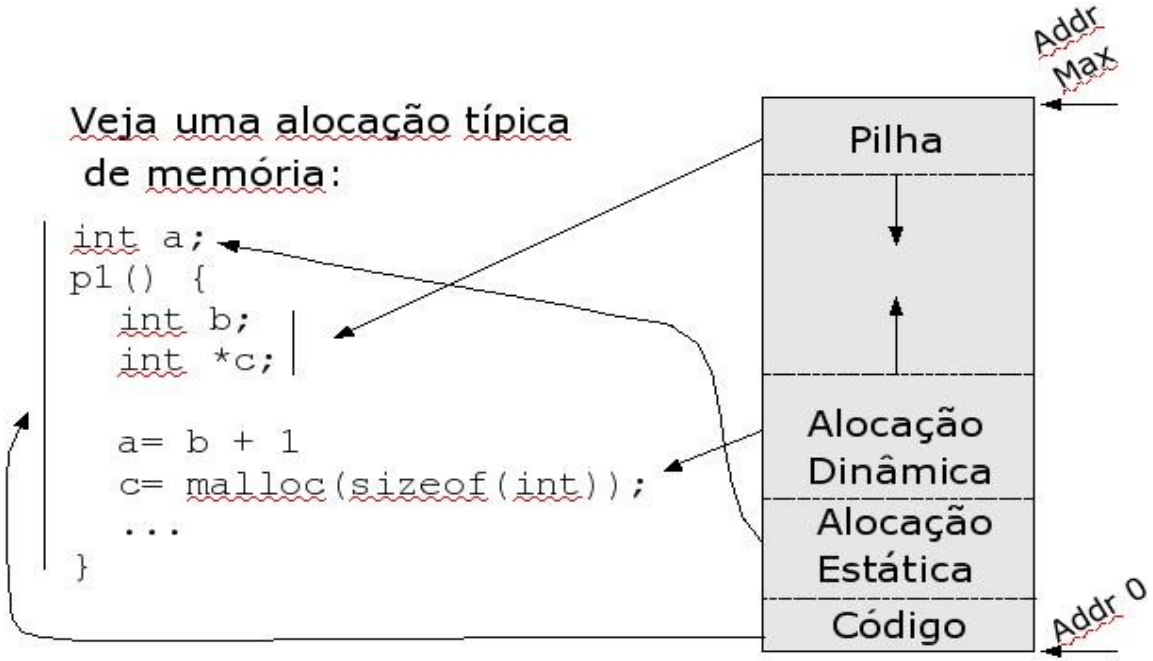
```
int a;
p1() {
  int b;
  int *c;

  a= b + 1
  c= malloc(sizeof(int));
  ...
}
```



Memória? Que Memória? (3)

Fazer isto tudo funcionar de forma organizada...



... é a função da **Gerência de Memória**

Tipos de memória (1)

Pense... o que fica em memória?

- Variáveis?
- Constantes?
- Programas?

... e como são usados?

Tipos de memória (2)

São usados assim:

- Leitura e escrita: variáveis
- Leitura apenas: constantes
- Execução: o código dos programas

E como podem usados?
Quem pode acessar o quê?

Tipos de memória (3)

Proteção é necessária (processos + S.O.):

- Proteger o quê?
- Proteger contra o quê/contra quem?

Vejamos, então, *um cenário* de proteção (outros são possíveis):

- Variáveis: somente o programa lê/escreve
- Constantes: somente o S.O. escreve, somente o programa lê
- Código: somente o SO escreve, somente programas executam

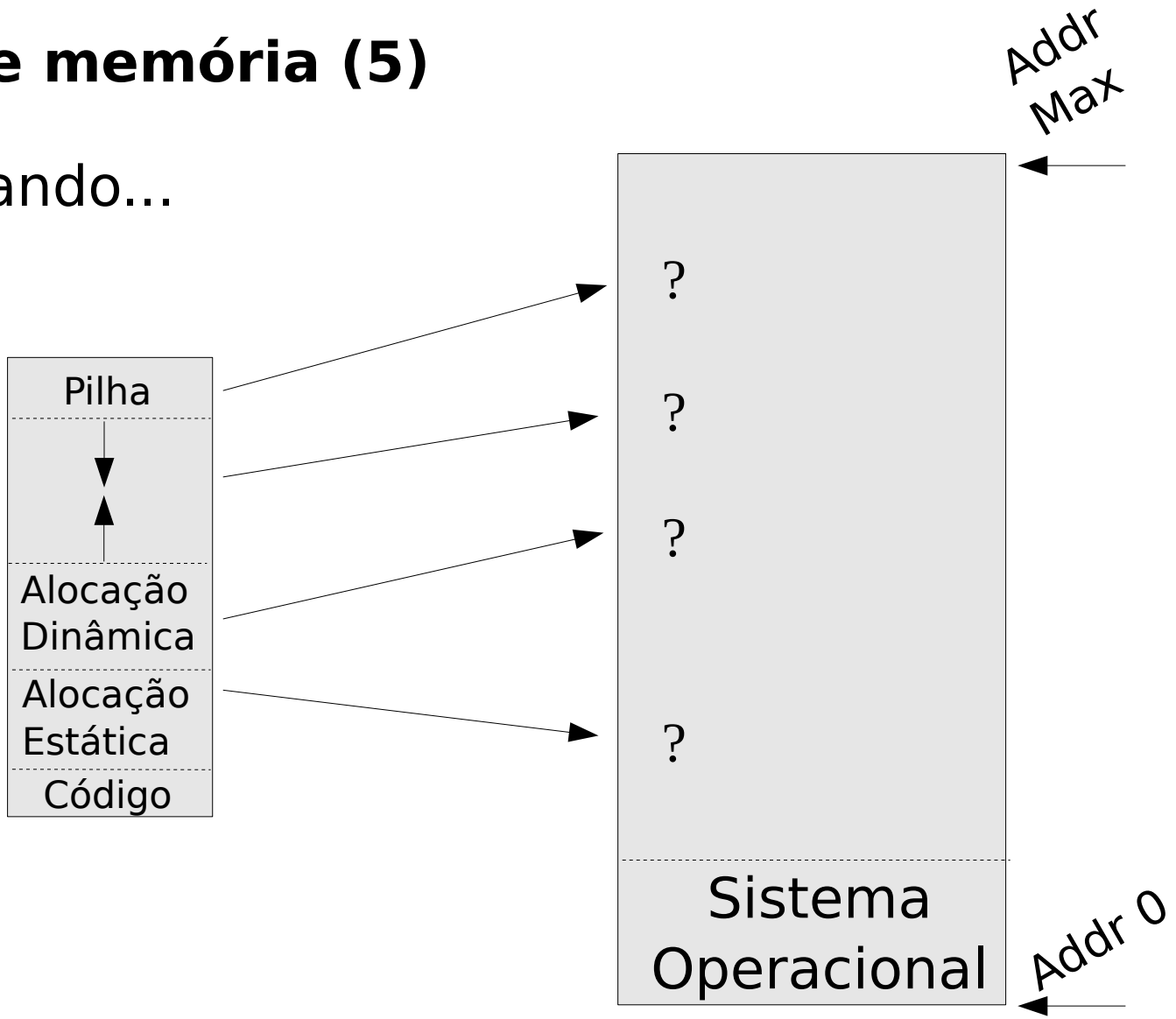
Tipos de memória (4)

Proteção:

- A cada acesso o endereço deve ser checado para impedir acessos indevidos
- A checagem é feita por hardware, senão fica muito lento
- Primeiro processador *x86* para adultos: 386 porque os anteriores não têm hardware para checar acessos à memória

Tipos de memória (5)

Relembrando...



Ligação – binding (1)

Qual endereço alocar para cada variável?

- Alocação estática
- Alocação dinâmica

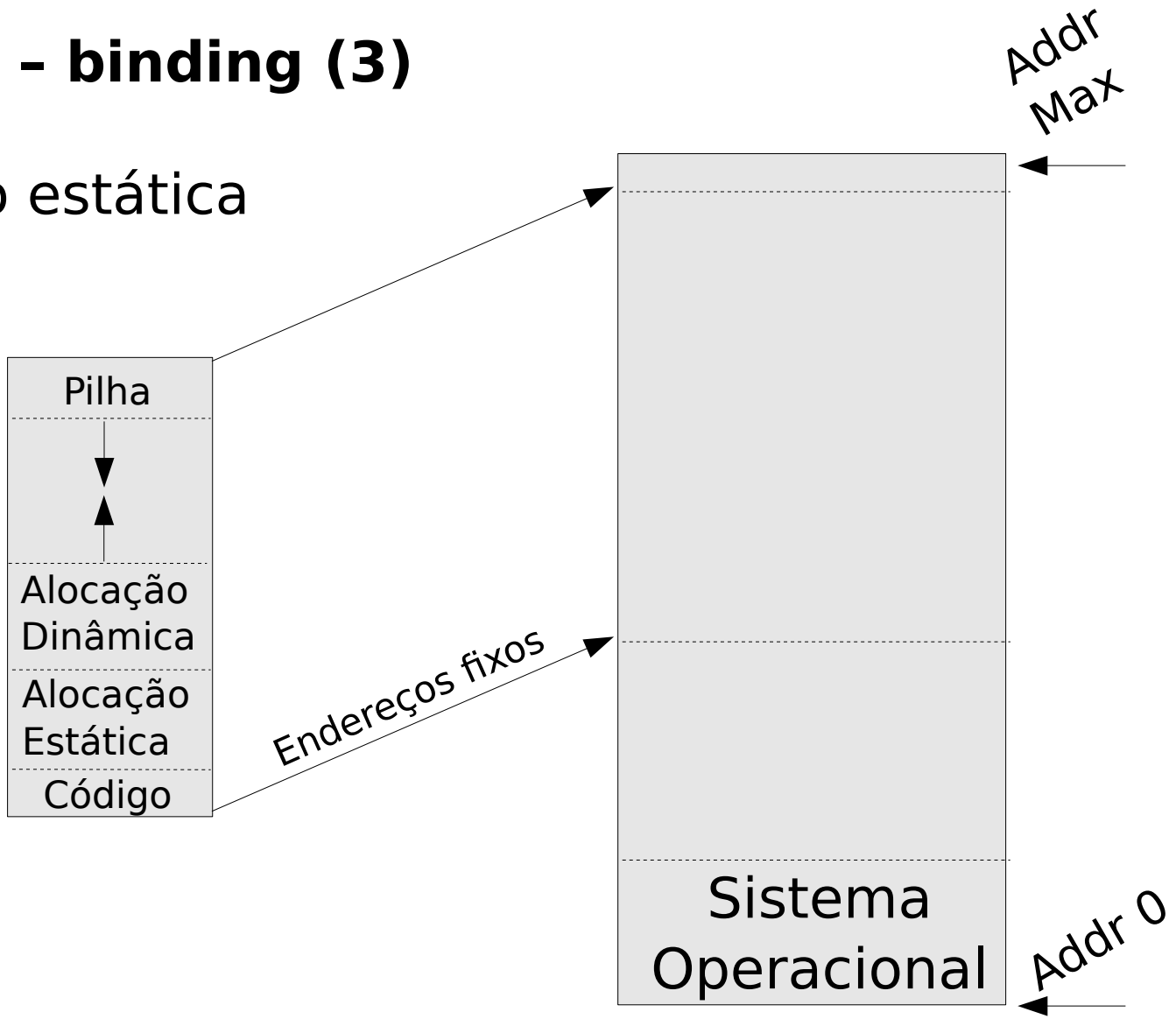
Ligação – binding (2)

Alocação estática - Decisão tomada quando o programa é compilado

- Ex: &a= 0x4C037F01
- Vantagem: SIMPLES!
- Desvantagem: Rígido. O programa tem que ser executado sempre no mesmo lugar na memória

Ligação – binding (3)

Alocação estática



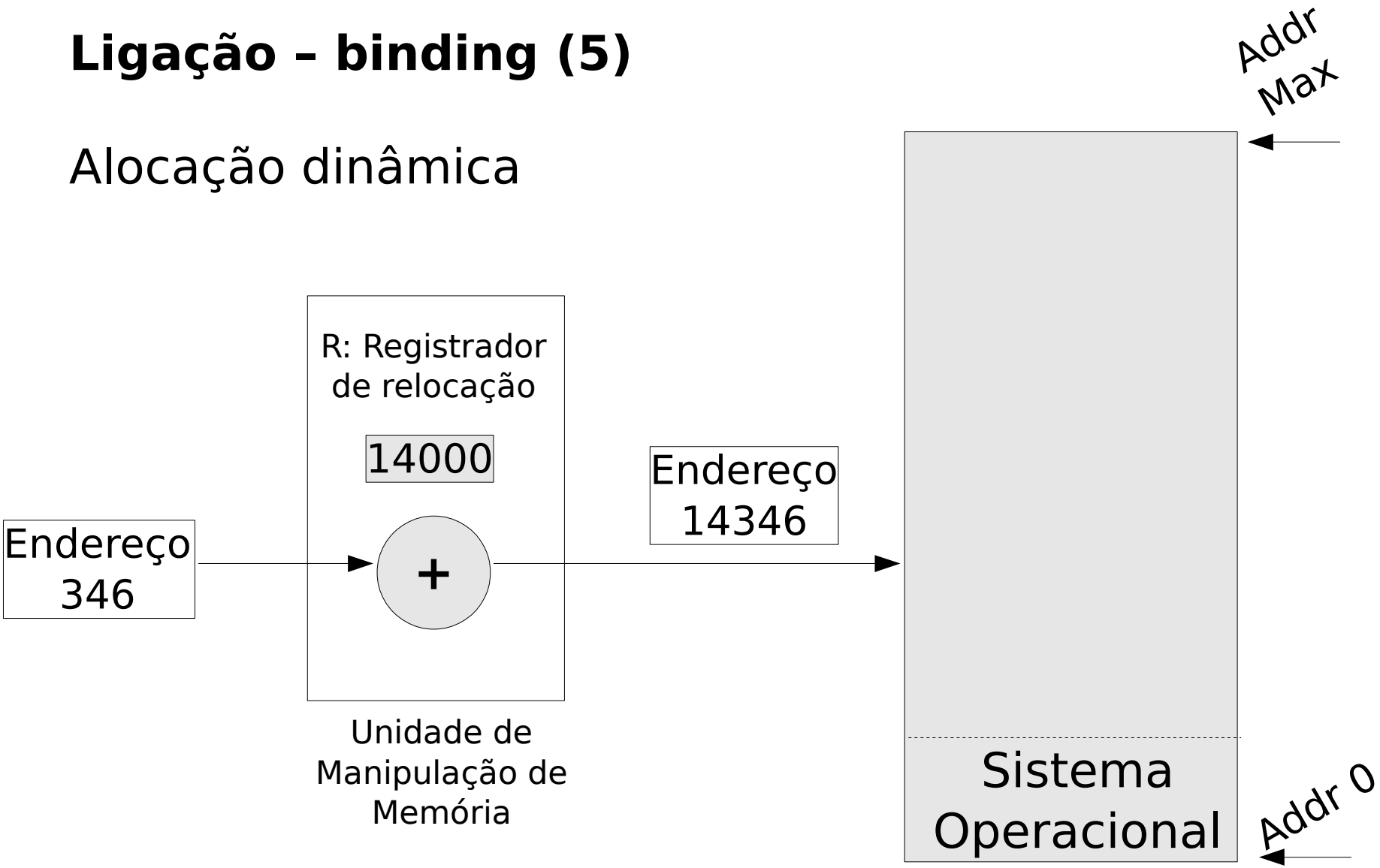
Ligação – binding (4)

Alocação dinâmica - Decisão adiada até a hora da execução

- Endereços são relativos. Ex: &a= R + 0x00FC
- Vantagem: Flexível
  - O programa pode executar em qualquer lugar da memória
  - O valor de R é definido somente na hora da execução
- Exige hardware específico

Ligação – binding (5)

Alocação dinâmica



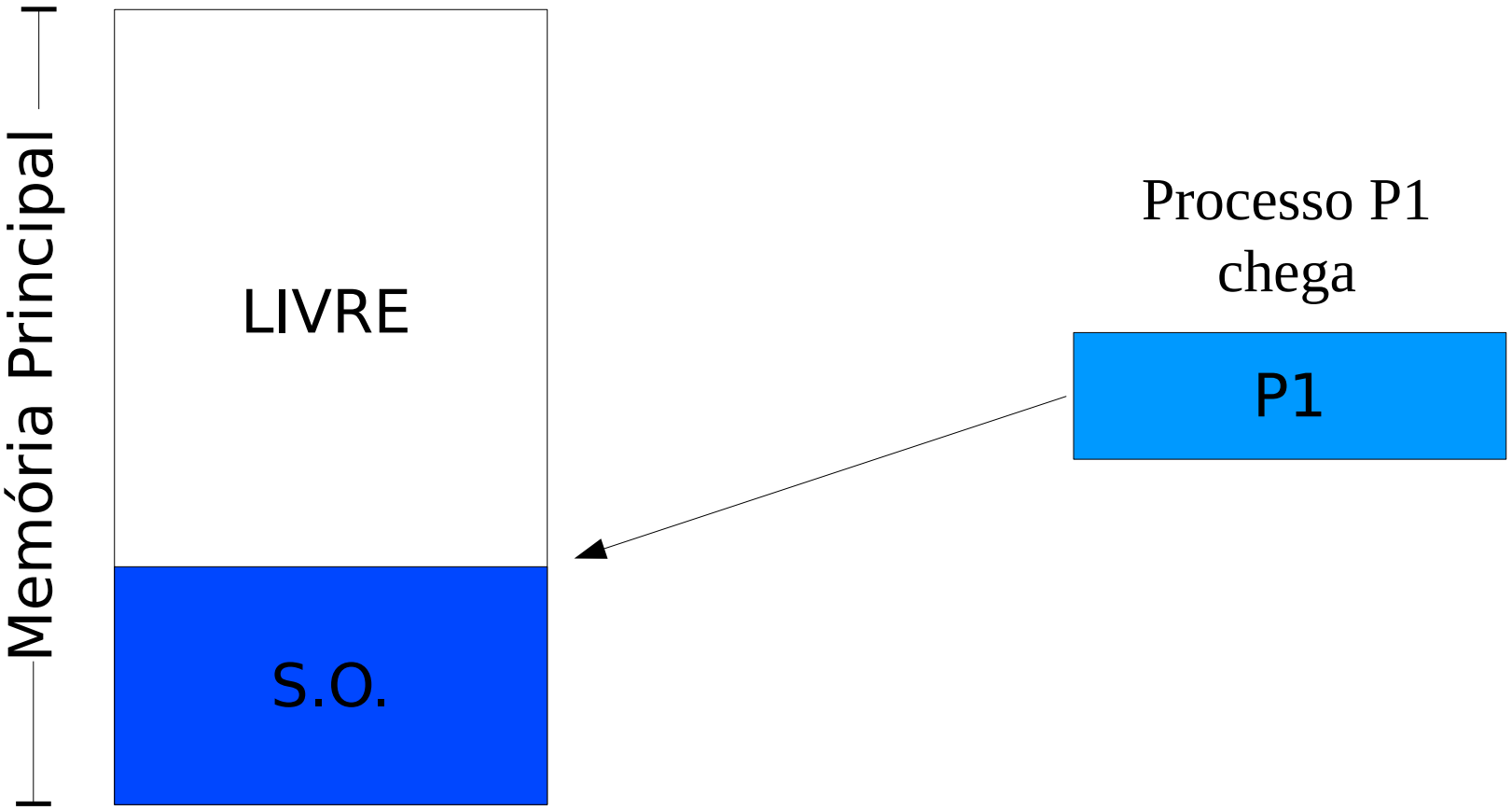
Alocação Dinâmica (1)

Permite **swapping** – troca:

- Carrega vários processos na memória ao mesmo tempo
- Quando chega um novo processo, e a memória principal está toda ocupada, escolhe um processo, grava-o no disco, e libera memória par ao próximo
- O processo ocupa uma área contígua da memória:
  - Como fica a proteção?
  - Simplifica: basta verificar se o endereço acessado está entre R e o limite máximo do processo
- Funciona???

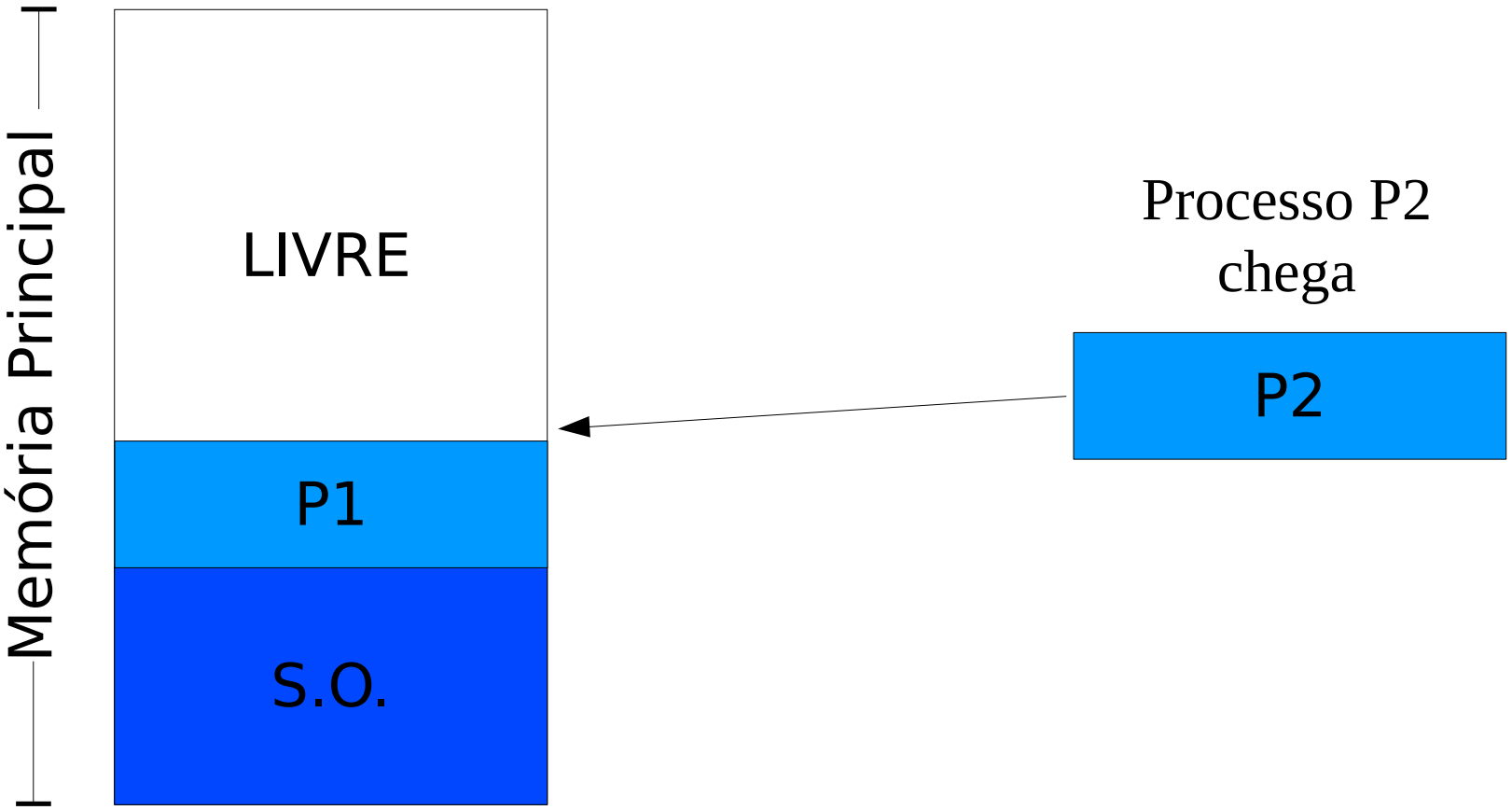
Alocação Dinâmica (2)

Funciona, mas sofre **fragmentação:**



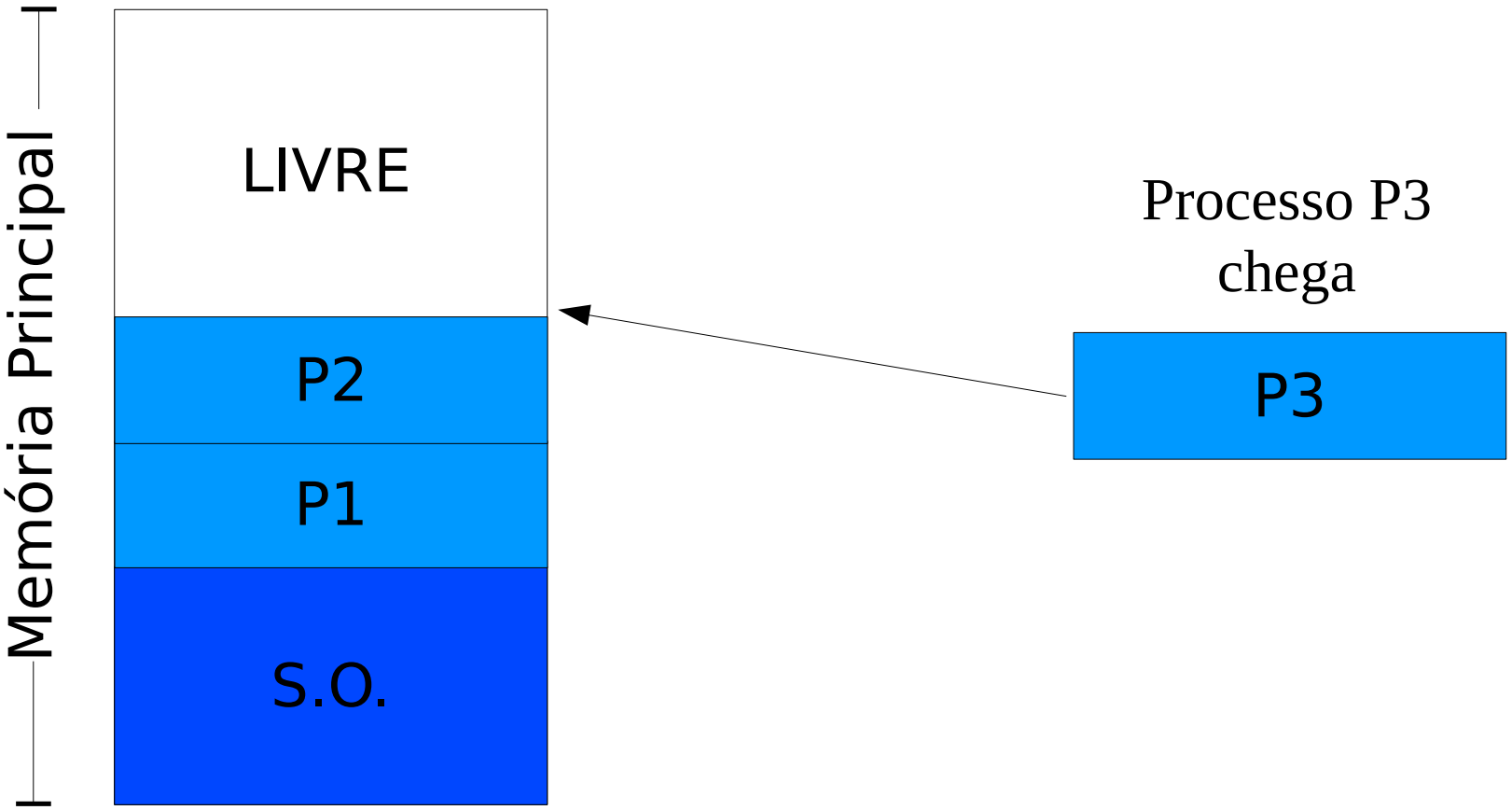
Alocação Dinâmica (3)

Funciona, mas sofre **fragmentação:**



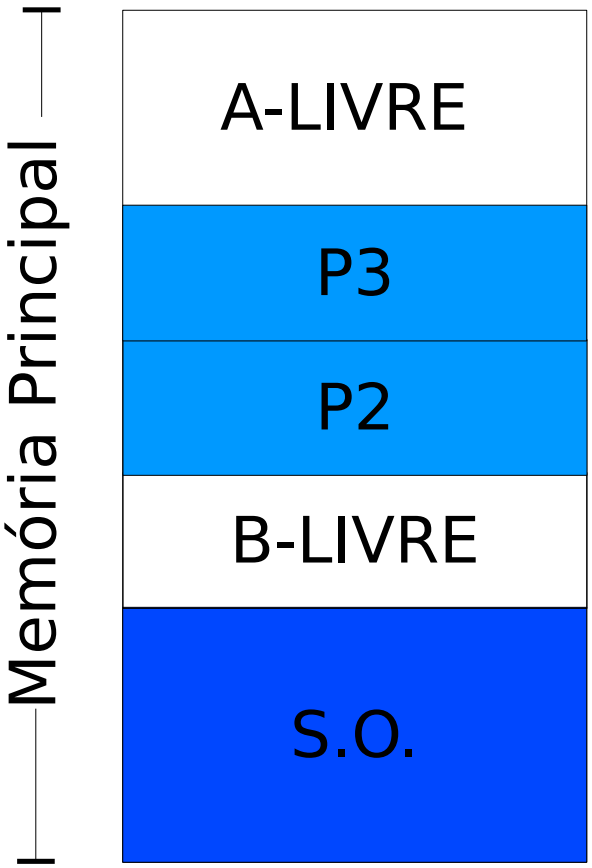
Alocação Dinâmica (4)

Funciona, mas sofre **fragmentação:**



Alocação Dinâmica (4)

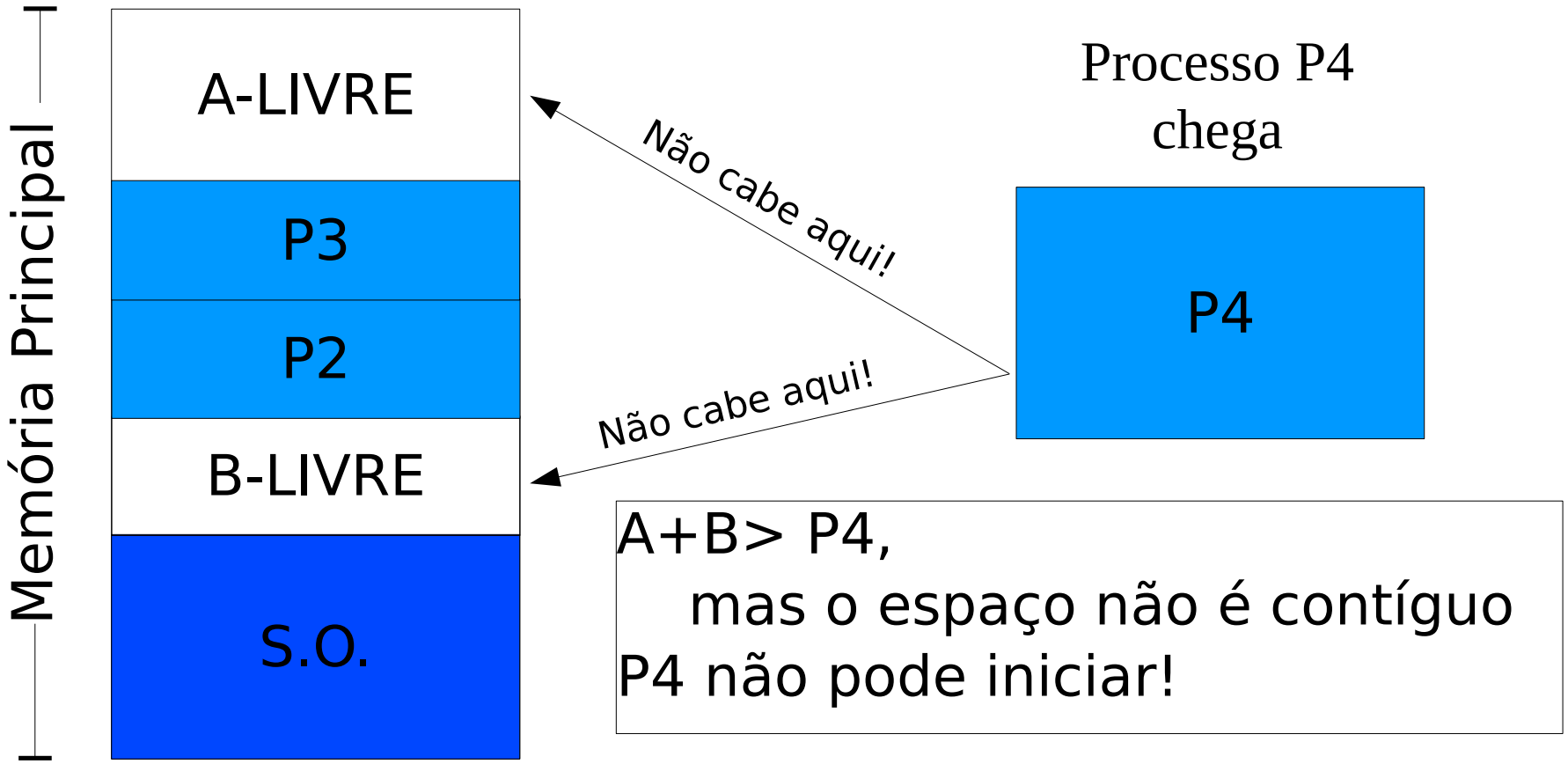
Funciona, mas sofre **fragmentação:**



Processo P1 termina

Alocação Dinâmica (5)

Funciona, mas sofre **fragmentação:**

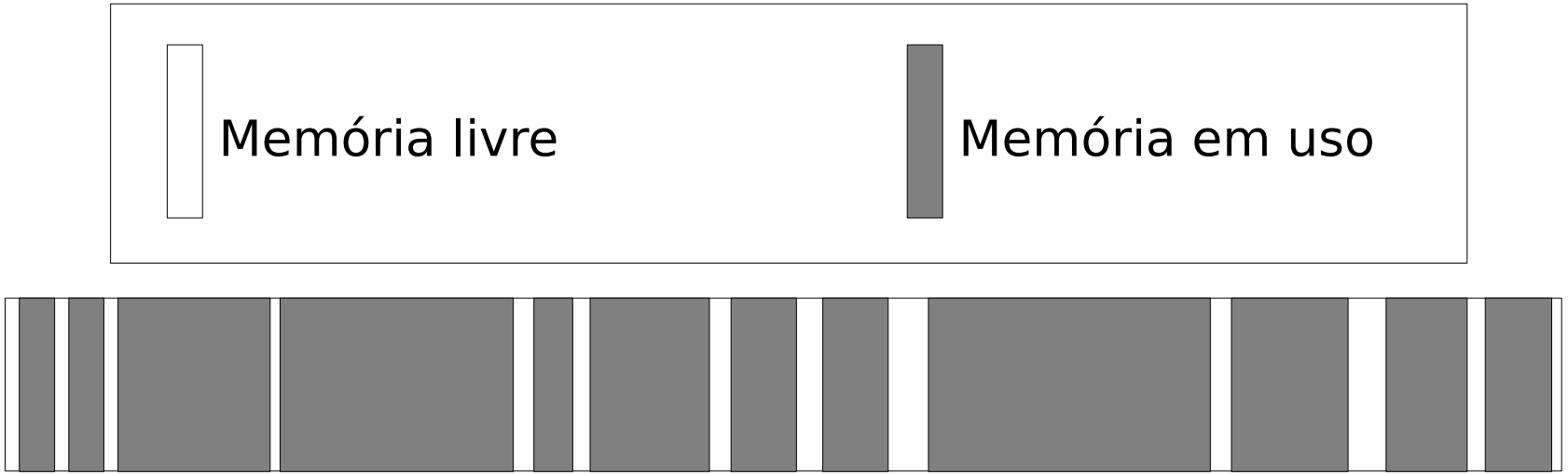


Fragmentação (1)

"Derruba" o desempenho:

- Memória cheia de buracos

- Grande parte da memória fica inutilizada



Fragmentação (2)

Estratégias para atacar o problema:

- *First-fit*
- *Best-fit*
- *Worst-fit*
- Compactação

Fragmentação (3)

Estratégias para atacar o problema:

- *First-fit*
- *Best-fit*
- *Worst-fit*
- Compactação

Escolha o **primeiro** buraco aonde o processo caiba

Fragmentação (3)

Estratégias para atacar o problema:

- *First-fit*
- *Best-fit*
- *Worst-fit*
- Compactação

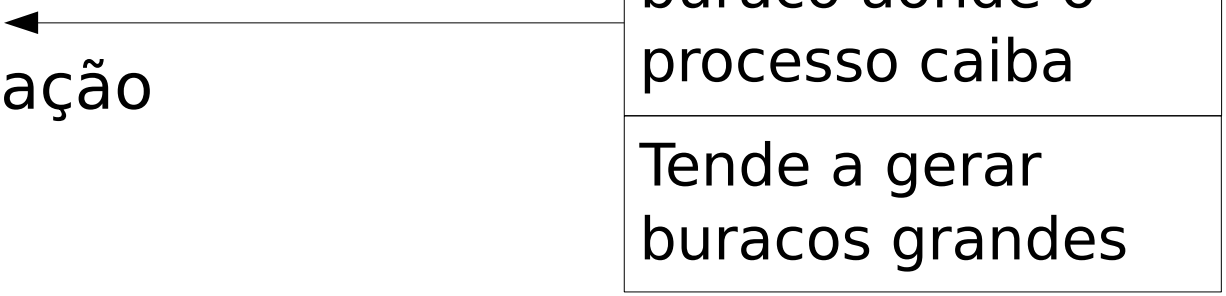
Escolha o **menor** buraco aonde o processo caiba

Tende a gerar buracos pequenos

Fragmentação (3)

Estratégias para atacar o problema:

- *First-fit*
- *Best-fit*
- *Worst-fit*
- Compactação



Fragmentação (3)

Estratégias para atacar o problema:

- *First-fit*
- *Best-fit*
- *Worst-fit*
- Compactação

Agora, pense... Qual a melhor dentre as três?

Não é claro. Em geral, *first-fit* é escolhida por ser mais rápida.

Fragmentação (3)

Estratégias para atacar o problema:

- *First-fit*
- *Best-fit*
- *Worst-fit*
- Compactação

Mover processos para “tapar” buracos

Fragmentação interna x externa (1)

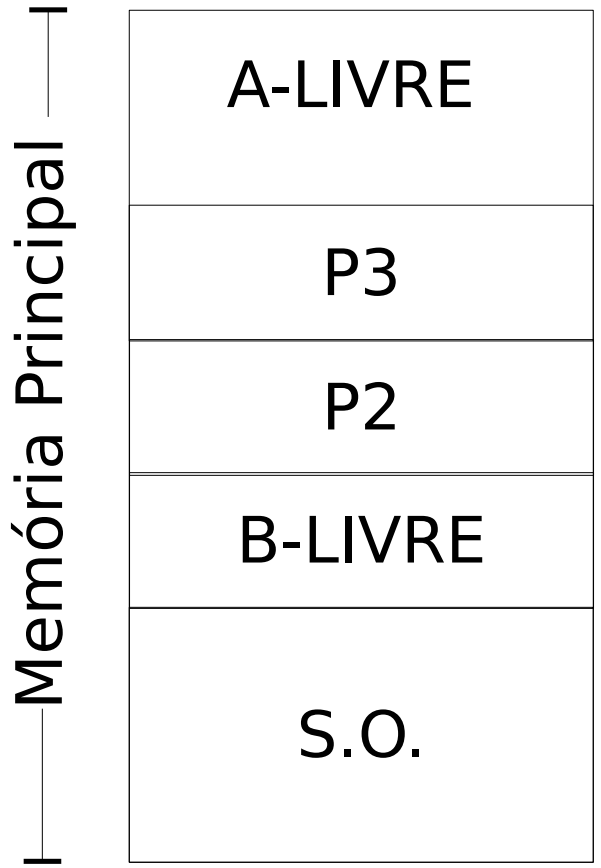
Fragmentação externa: entre processos

Fragmentação interna: dentro de um processo

Fragmentação interna x externa (2)

Fragmentação externa:

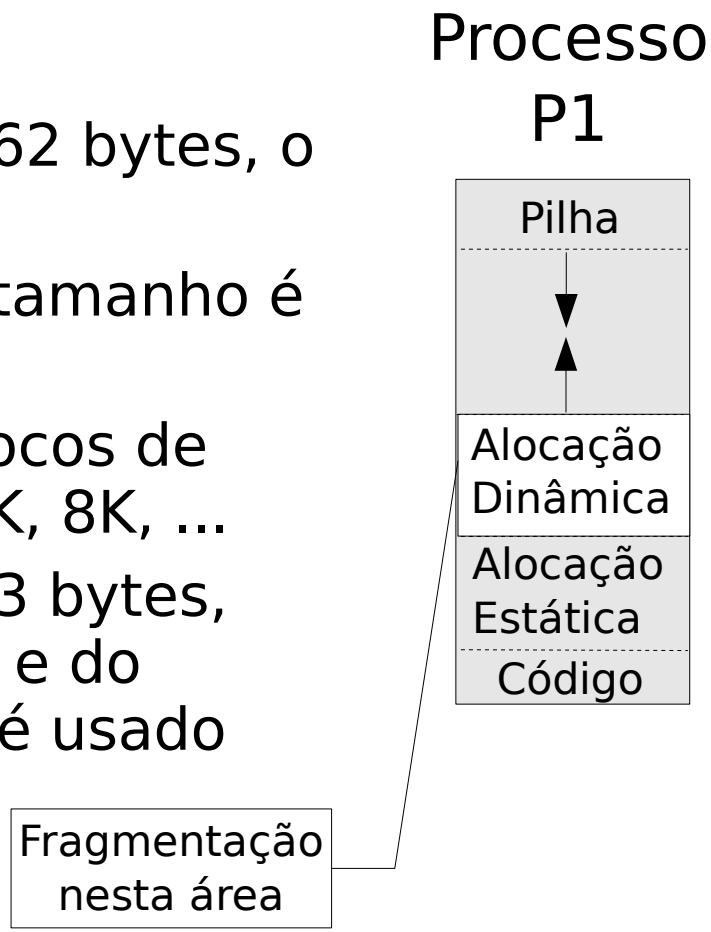
- Entre processos
- Ao nível do S.O.



Fragmentação interna x externa (3)

Fragmentação interna:

- Dentro de um processo
  - Se um processo pede 18462 bytes, o que fazer?
  - Alocar **exatamente** este tamanho é caro de gerenciar
  - Normalmente, aloca-se blocos de tamanho potência de 2: 4K, 8K, ...
  - Se um processo pedir 8193 bytes, alocam-se 2 blocos de 8K, e do segundo, somente 1 byte é usado



Resumo – Gerência de Memória

- **Gerência de Memória:** Alocar e coordenar o uso de um recurso escasso, a memória, para uso por múltiplos programas
- **Usos da memória:** leitura/escrita, leitura, execução
- **Ligação (binding) estática e dinâmica:** registrador de relocação
- **Fragmentação:** o que é e como lidar com ela