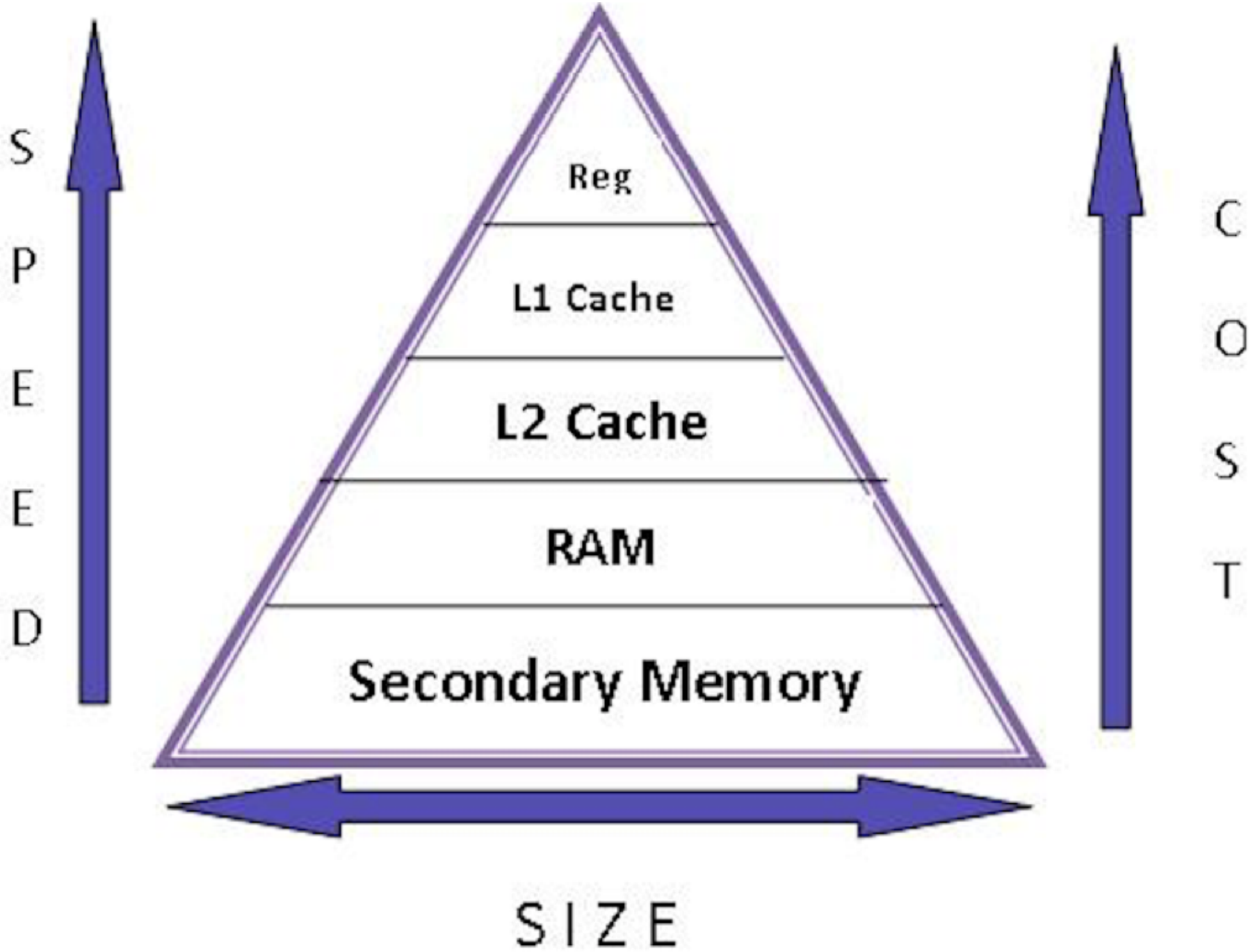
Gerenciamento de Memória

Gerenciamento de Memória

- Idealmente os programadores querem uma memória que seja:
  - Grande
  - Rápida
  - Não Volátil
  - Baixo custo
- Infelizmente a tecnologia atual não comporta tais memórias
- A maioria dos computadores utiliza Hierarquia de Memórias que combina:
  - Uma pequena quantidade de memória cache, volátil, muito rápida e de alto custo
  - Uma grande memória principal (RAM), volátil, com centenas de MB ou poucos GB, de velocidade e custo médios
  - Uma memória secundária, não volátil em disco, com gigabytes (ou terabytes), velocidade e custo baixos


SPEED
Reg
L1 Cache
L2 Cache
RAM
Secondary Memory
COST
SIZE

Gerência de Memória

- Cabe ao Gerenciador de Memória gerenciar a hierarquia de memória

- Controla as partes da memória que estão em uso e quais não estão, de forma a:

- **alocar memória aos processos, quando estes precisarem;**
- **liberar memória quando um processo termina; e**
- **tratar do problema do swapping (quando a memória é insuficiente).**

Gerenciamento Básico de Memória

- Sistemas de Gerenciamento de Memória, podem ser divididos em 2 classes:
  - Sistemas que, durante a execução levam e trazem processos entre a memória principal e o disco
  - Sistemas mais simples, que não fazem troca de processos

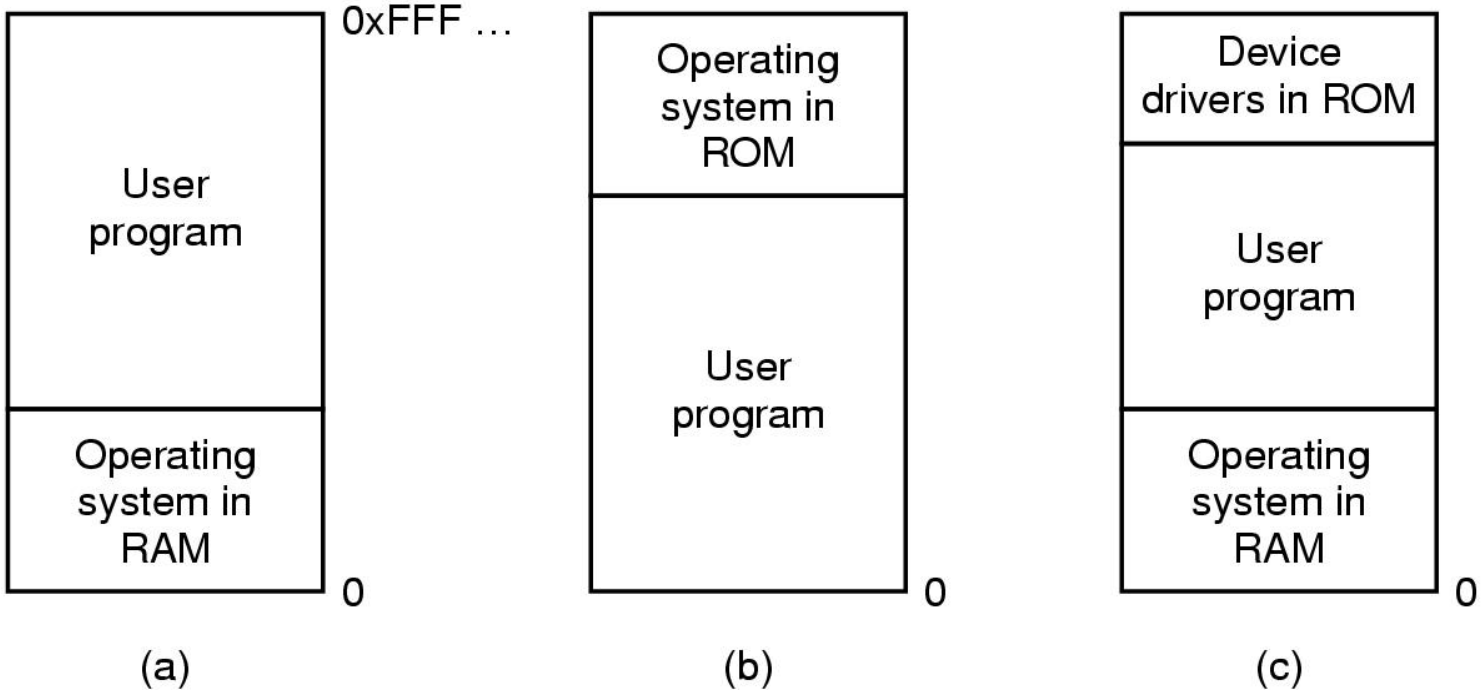
Monoprogramação sem trocas de processos

- Sistemas Mono-usuários: gerência de memória é bem simples, pois toda a memória é alocada à próxima tarefa, incluindo a área do S.O.

- Erros de execução podem vir a danificar o S.O.

- Neste caso, a destruição do S.O. é um pequeno inconveniente, resolvido pelo recarregamento do mesmo.

Gerenciamento Básico de Memória
Monoprogramação sem trocas de processos



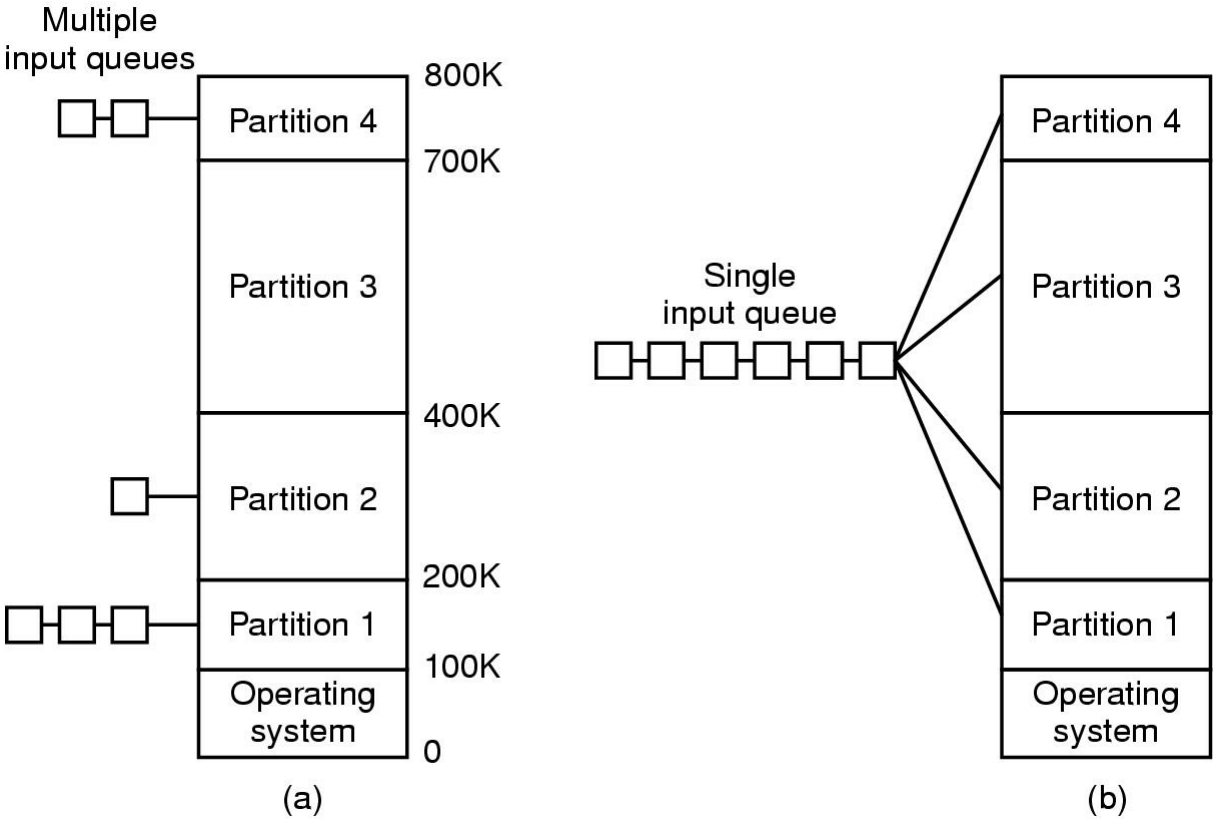
Três esquemas simples de organização de memória

- Um sistema operacional com um processo de usuário

Multiprogramação com partições Fixas

- **Sistemas Monoprogramados:** raramente usados atualmente.
- **Sistemas modernos:** permitem multiprogramação
- A maneira mais comum de realizar a multiprogramação é dividir simplesmente a memória em **n partições** (provalmente de tamanhos diferentes).
- Esta divisão pode ser feita de maneira manual, quando o sistema é inicializado
- Ao chegar um job, pode ser colocado em uma fila de entrada associada à menor partição, grande o suficiente para armazená-lo

Multiprogramação com partições Fixas



- Partições de memória fixa
  - fila separada para cada partição
  - uma única fila de entrada

Modelagem da Multiprogramação

O uso da Multiprogramação pode melhorar a utilização da UCP

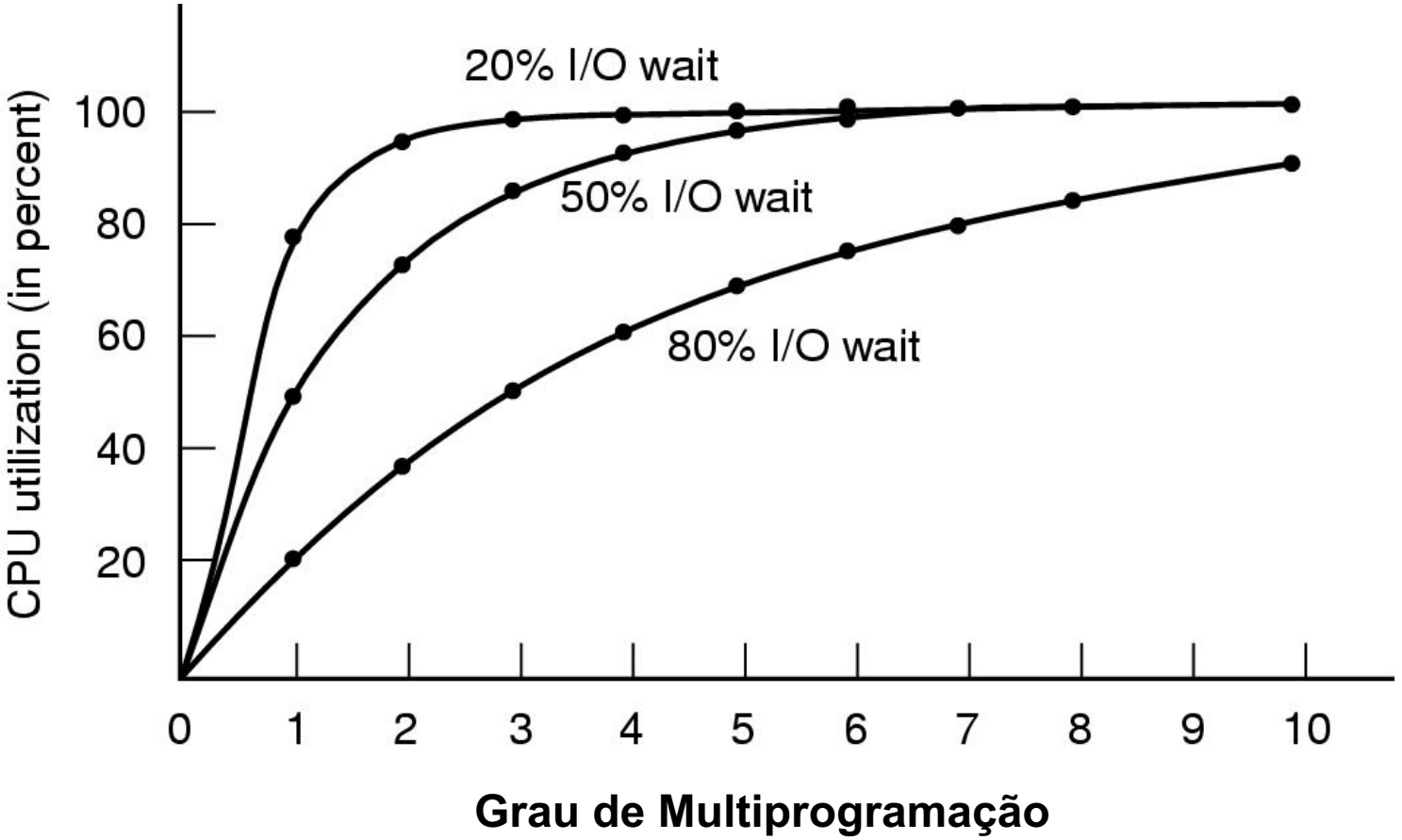
Considerando a utilização da UCP de modo probabilístico:

- Suponha que um processo gaste uma fração p de seu tempo esperando pela finalização de sua solicitação de E/S
- Com n processos simultâneos na memória, a probabilidade de todos os n processos estarem esperando por E/S (situação em que a UCP está ociosa) é p^n

- A utilização da UCP é dada pela fórmula:

$$\text{utilização da UCP} = 1 - p^n$$

Modelagem da Multiprogramação



A utilização da UCP como uma função do número de processos na memória

Relocação e Proteção

- Pode não ter certeza de onde o programa será carregado na memória
  - As localizações de endereços de localização das variáveis e do código das rotinas não podem ser absolutos
  - Deve-se manter um programa fora das partições de outros processos
- Uso de valores de base e limite
  - Os endereços das localizações são somados a um valor de base para mapear um endereço físico
  - Valores de localizações maiores que um valor limite é considerado um erro

Troca de Processos

Com um sistema em lote, é simples e eficiente organizar a memória em partições fixas.

Em sistemas de tempo compartilhado:

- Pode não existir memória suficiente para conter todos os processos ativos
- Os processos excedentes são mantidos em disco e trazidos dinamicamente para a memória a fim se serem executados

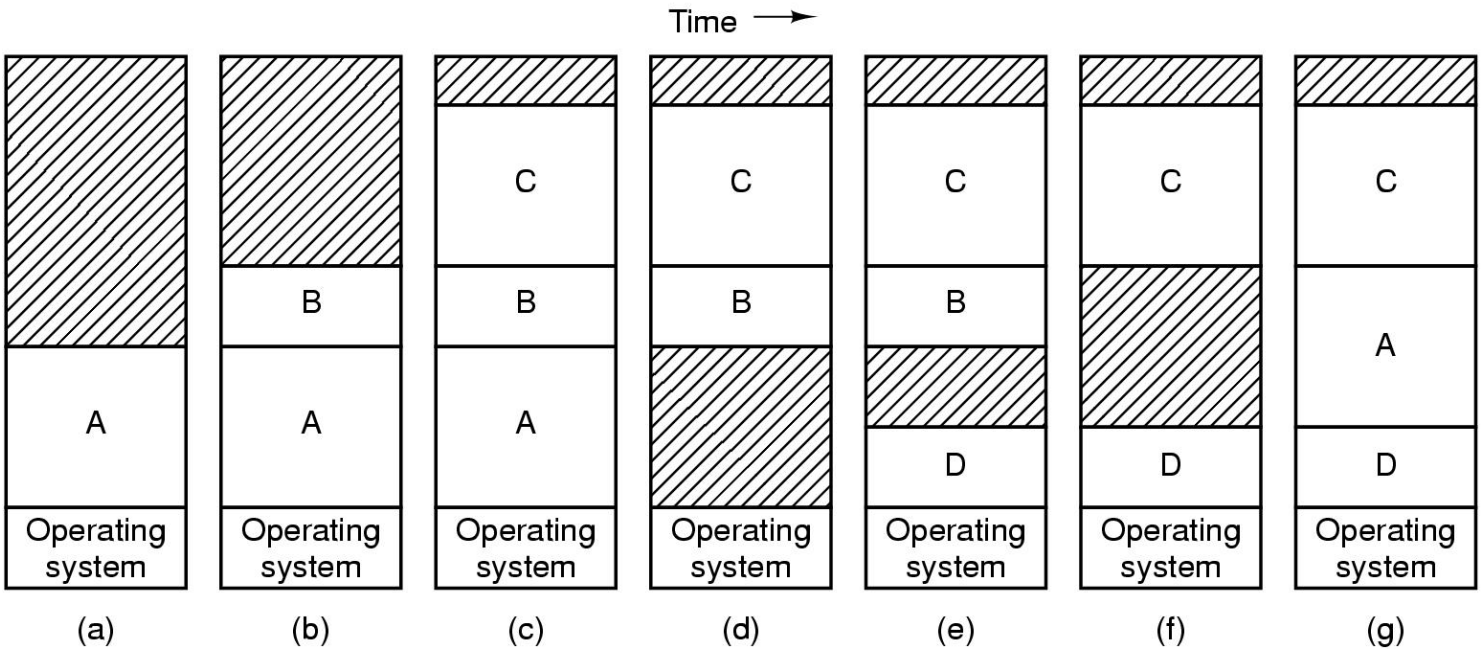
Troca de Processos

Existem 2 processos gerais que podem ser usados:

- A troca de processos (*swapping*): forma mais simples, consiste em trazer totalmente cada processo para a memória, executá-lo durante um tempo e, então, devolvê-lo ao disco

- Memória Virtual: permite que programas possam ser executados mesmo que estejam parcialmente carregados na memória principal.

Troca de Processos

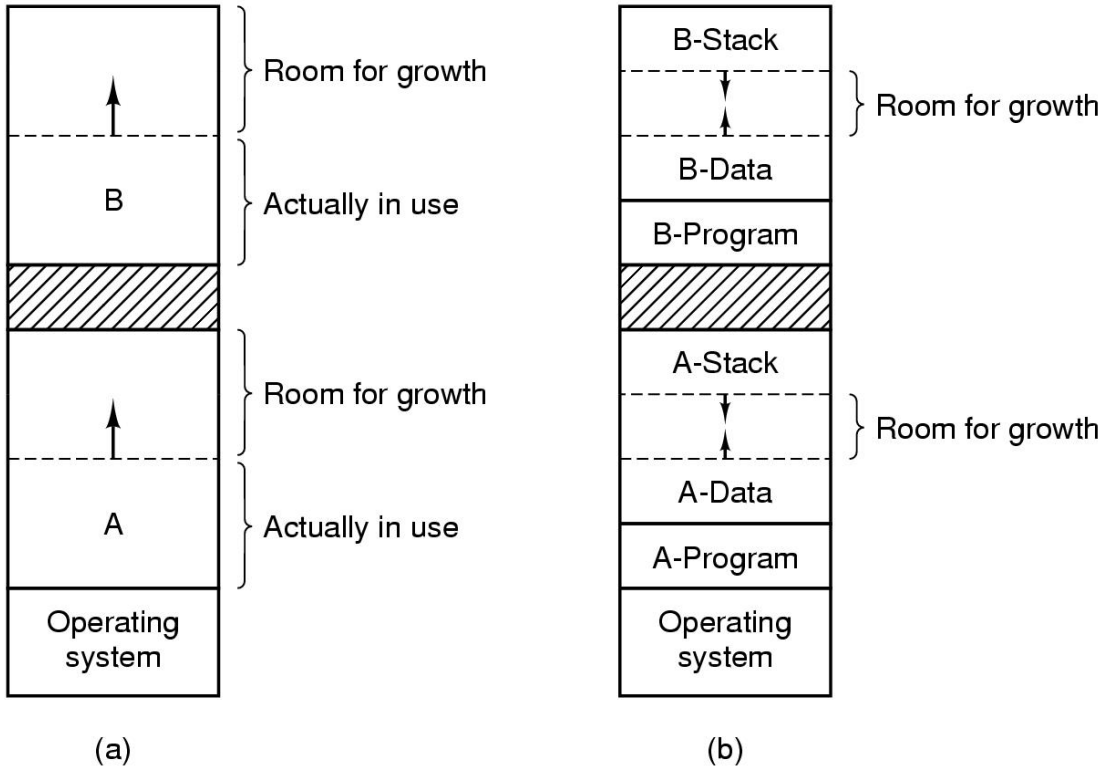


A Alocação de memória muda a medida que

- Os processos chegam à memória
- Os processos deixam a memória

As regiões sombreadas representam a memória não usada

Troca de Processos



- Alocando espaço para um segmento de dados em expansão
- Alocando espaço para segmentos de pilha e dados em expansão

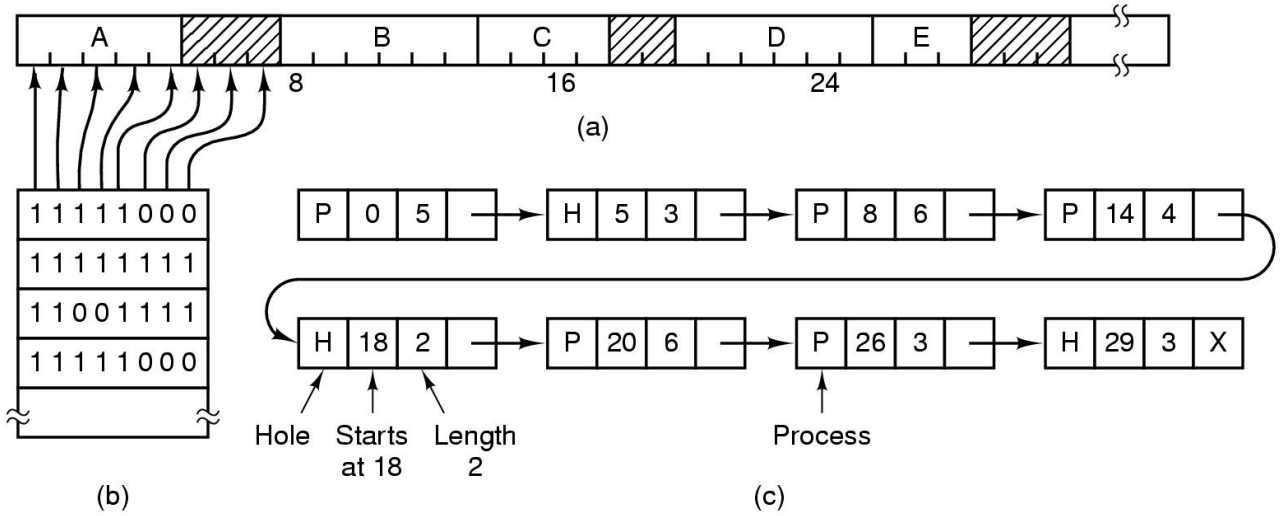
Gerenciamento de Memória

- Quando a memória é alocada dinamicamente, o S.O. deve gerenciá-la
- Existem 2 formas de fazer isto:
  - Mapa de Bits (Bitmap)
  - Lista de Disponíveis

Gerenciamento de Memória com Mapas de Bits

Mapa de Bits (Bitmap): a memória é dividida em unidades de alocação (de poucas palavras ou vários KB). A cada unidade de alocação é associado um bit no mapa de bits, o qual vale 0 (se a unidade está disponível) e 1 (se estiver ocupada).

Gerenciamento de memória com Mapas de Bits



(a) Uma parte da memória com 5 processos e 3 buracos

- As regiões em branco (1 no bitmap) marcam as unidades já alocadas
- As regiões sombreadas (0 no bitmap) marcam unidades desocupadas

(b) O Bitmap correspondente

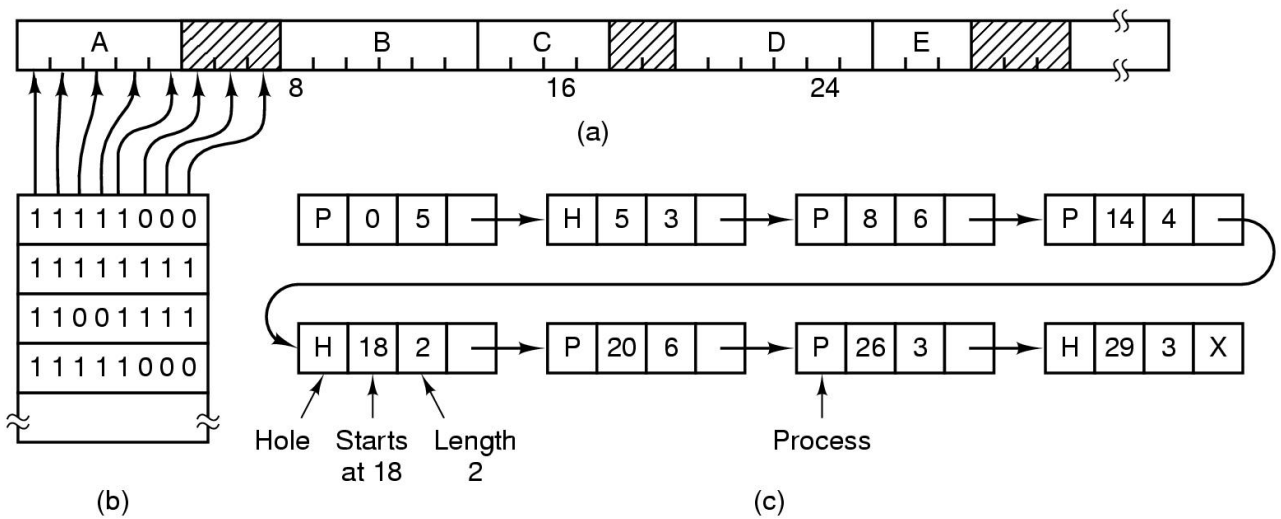
(c) A mesma informação como uma lista encadeada

Gerenciamento de memória com Listas Encadeadas

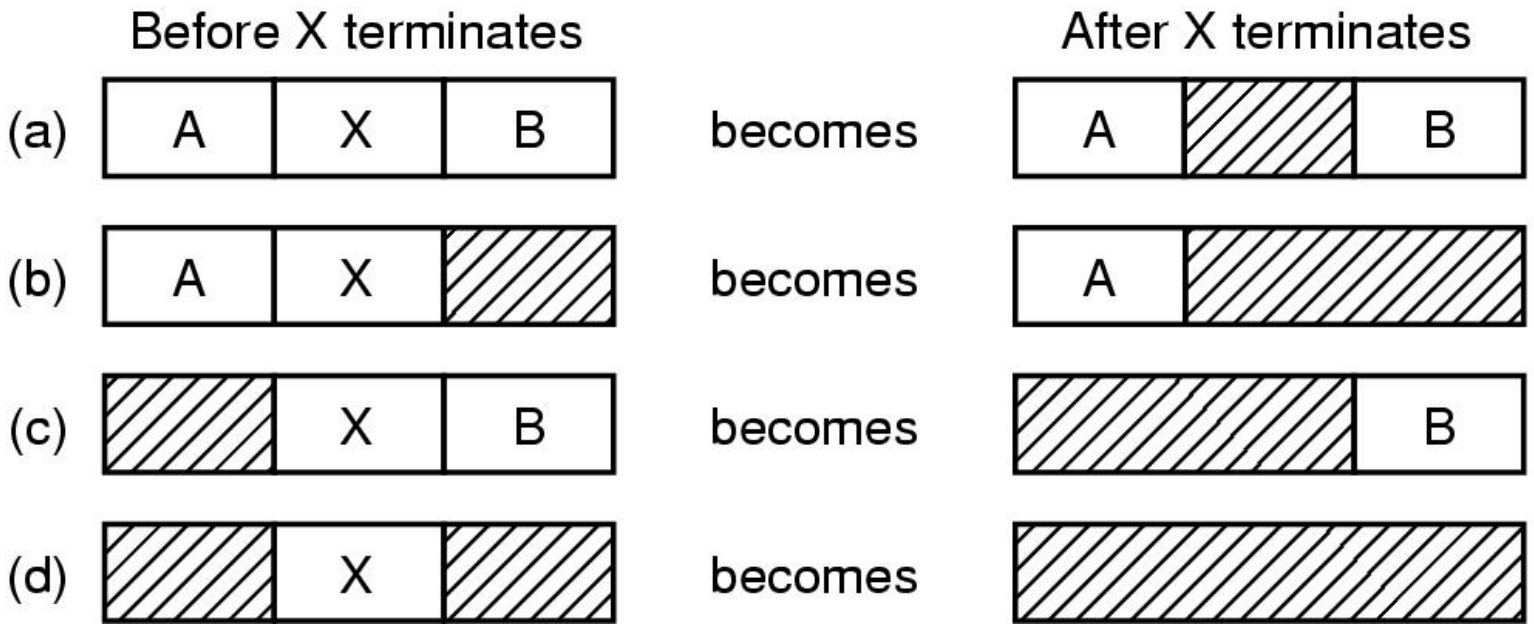
Outra maneira de gerenciar memória é manter uma lista encadeada de segmentos de memória alocados e segmentos disponíveis.

Cada elemento desta lista especifica:

- Um segmento disponível (H), ou alocado a um processo (P)
- O endereço onde se inicia este segmento
- E um ponteiro para o próximo elemento



Gerenciamento de Memória com Listas Encadeadas



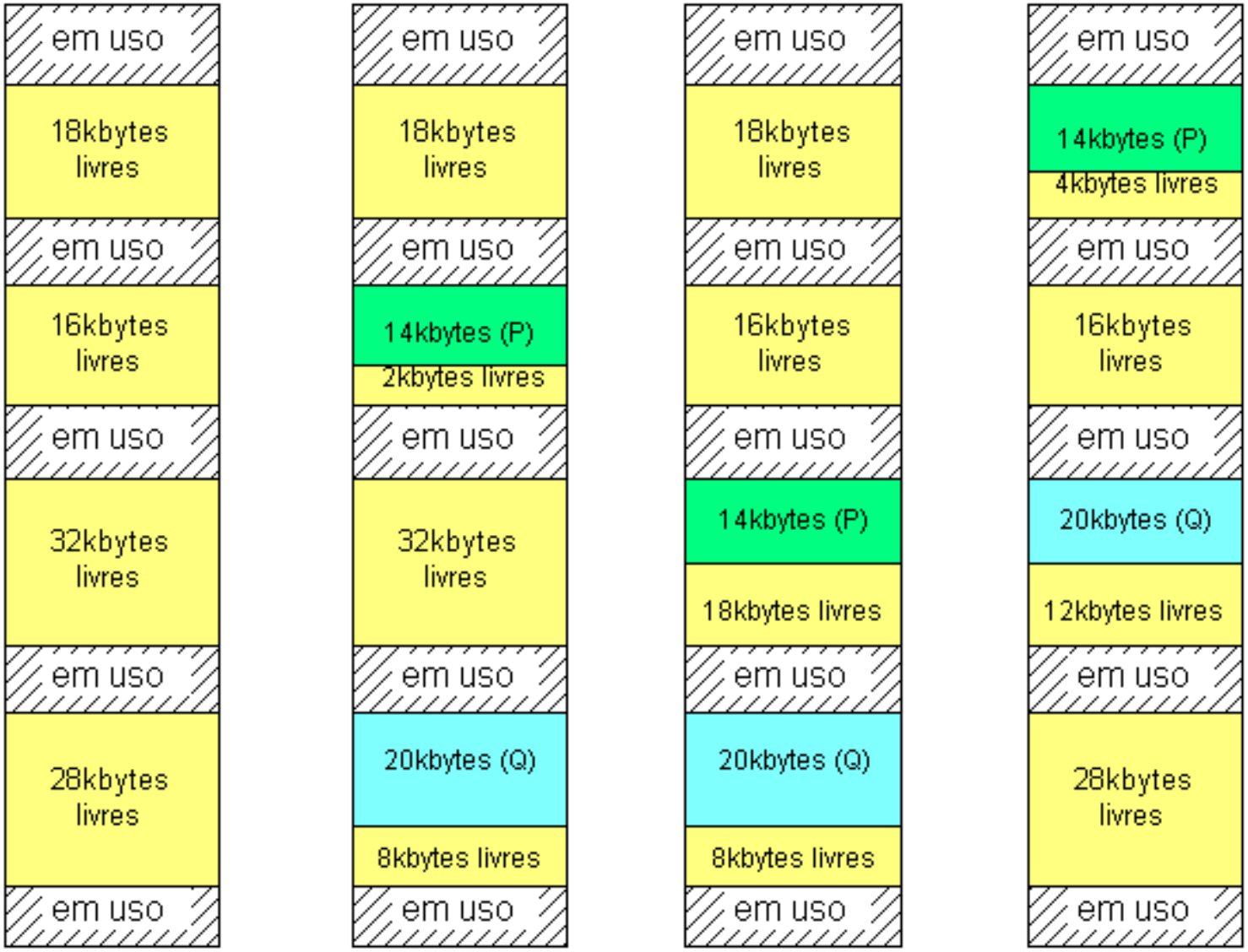
Quatro combinações de vizinhos para um processo X em término

Alocação de segmentos livres

- Existem três métodos que podem ser usados para selecionar uma região para um processo. Os algoritmos de alocação são:

- **Melhor escolha (Best fit): colocar o processo no bloco com o mínimo resto de memória;**
- **Pior escolha (worst fit): usar o bloco com o maior resto de memória;**
- **Primeira escolha (First fit): ir sequencialmente através dos blocos, e tomar o primeiro grande o suficiente.**

Alocação de segmentos livres



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.