Sistemas Operacionais

Gerenciamento de Memória

Norton Trevisan Roman
Marcelo Morandini
Jó Ueyama

Apostila baseada nos trabalhos de Kalinka Castelo Branco, Antônio Carlos Sementille, Paula Prata e nas transparências fornecidas no site de compra do livro "Sistemas Operacionais Modernos"

Gerenciamento de Memória

- Recurso importante;
- Tendência atual do software
  - Lei de Parkinson: “Os programas se expandem para preencher a memória disponível para eles” (adaptação);
- Hierarquia de memória:
  - Cache;
  - Principal;
  - Disco;

Gerenciamento de Memória

- Idealmente os programadores querem uma memória que seja:
  - Grande
  - Rápida
  - Não Volátil
  - De baixo custo
- Infelizmente a tecnologia atual não comporta tais memórias

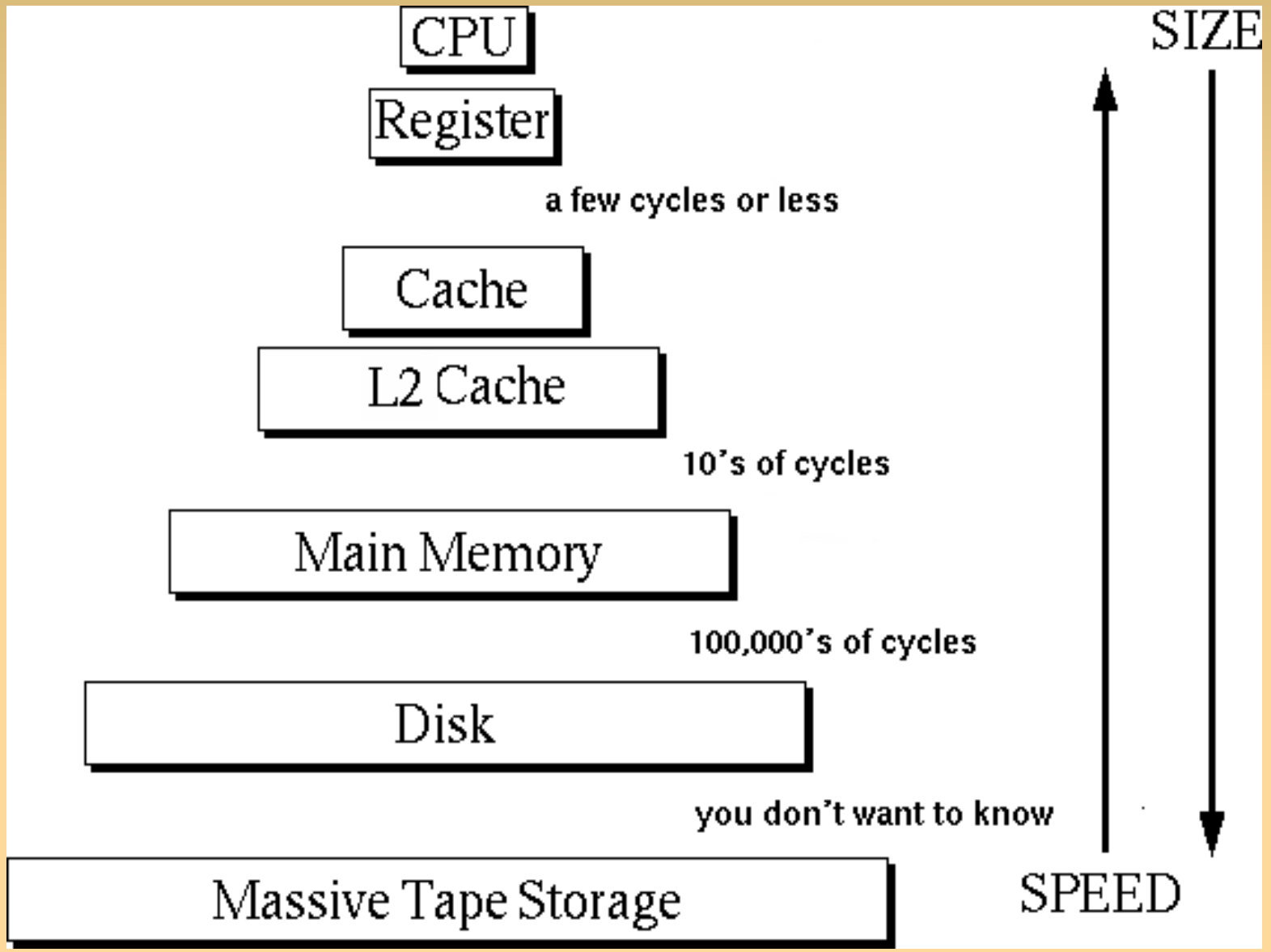
Gerenciamento de Memória

- 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

Gerenciamento de Memória

- Cache
  - Pequena quantidade
    - k bytes
  - Alto custo por byte
  - Muito rápida
  - Volátil
- Memória Principal
  - Quantidade intermediária
    - M bytes
  - Custo médio por byte
  - Velocidade média
  - Volátil
- Disco
  - Grande quantidade
    - G bytes
  - Baixo custo por byte
  - Lenta
  - Não volátil

Gerenciamento de Memória



Gerenciamento de Memória

- Cabe ao Gerenciador de Memória:
  - Gerenciar a hierarquia de memória
    - Para cada tipo de memória:
      - Gerenciar espaços livres/ocupados;
      - Alocar processos/dados na memória;
      - Localizar dado;
  - Controlar 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

Gerenciamento de Memória

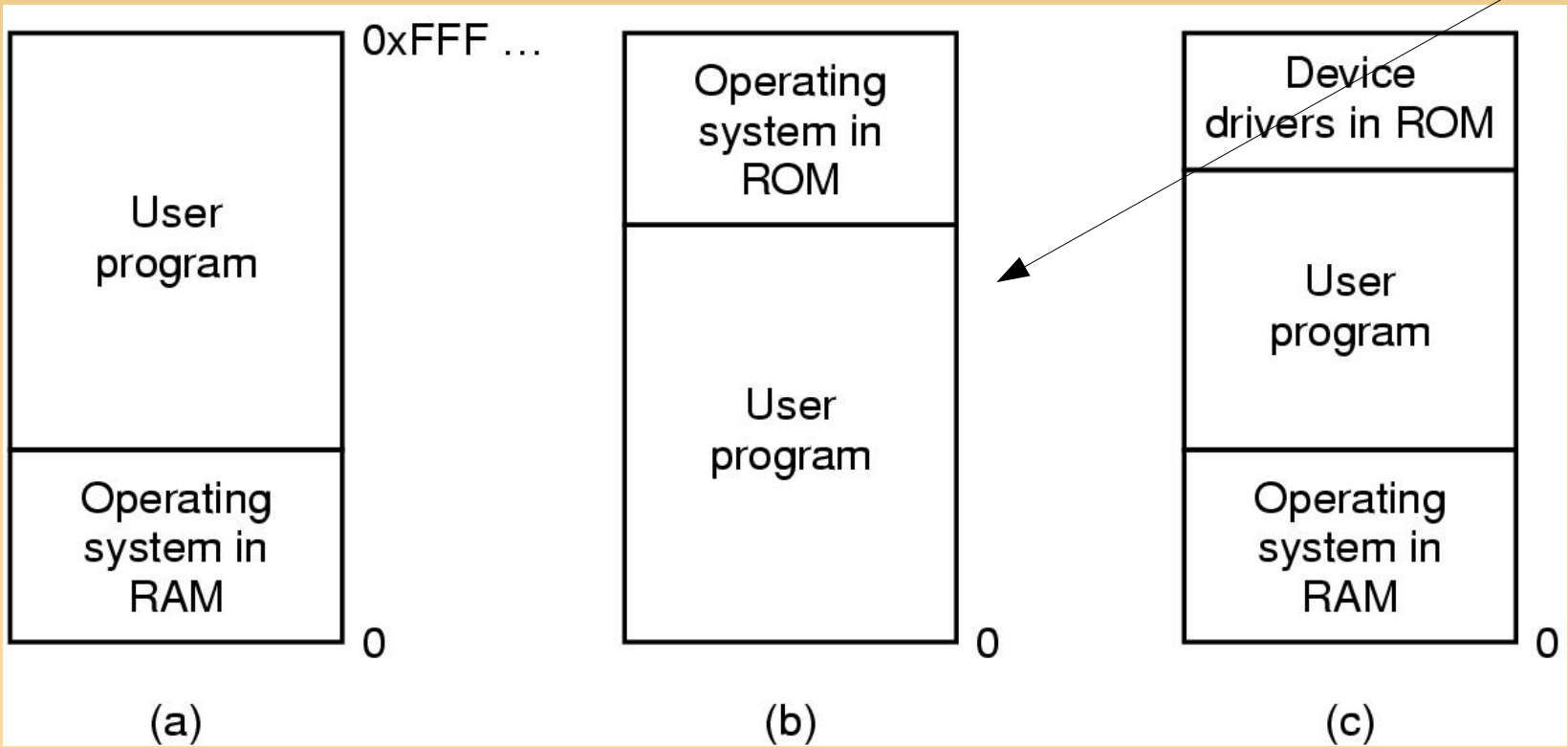
- Cabe ao Gerenciador de Memória:
  - Controlar as partes da memória que estão em uso e quais não estão, de forma a:
    - Tratar do problema do swapping (quando a memória é insuficiente).
      - Responsável por gerenciar chaveamento entre a memória principal e o disco, e memória principal e memória cache;
    - Swap?
      - Veremos mais adiante

Abstrações de Memória

- Como a memória é “vista” pelos programas?
  - Sem abstração
  - Espaços de endereços
- Sem abstração:
  - A mais simples – programas simplesmente vêem toda a memória física
    - Acessam-na diretamente, pelo seu endereço
  - Ex: MOV R1, 1000
    - Mova o conteúdo do endereço de memória 1000 ao registrador R1

Abstrações de Memória

- Sem abstração:
  - Algumas variações em sua organização:



Usado antigamente em mainframes

Usado em alguns handhelds

Primeiros computadores pessoais

Abstrações de Memória

- Sem abstração:
  - Sistemas assim geralmente são monousuários, rodando apenas um processo por vez
    - Toda a memória é alocada à próxima tarefa, incluindo a área do S.O
    - Sistemas do usuário podem danificar o S.O., que deve ser recarregado
  - Poderíamos ter multiprogramação assim?

Abstrações de Memória

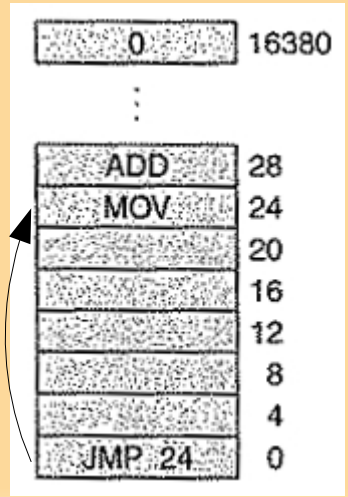
- Sem abstração:
  - Sistemas assim geralmente são monousuários, rodando apenas um processo por vez
    - Toda a memória é alocada à próxima tarefa, incluindo a área do S.O
    - Sistemas do usuário podem danificar o S.O., que deve ser recarregado
  - Poderíamos ter multiprogramação assim?
    - Se o S.O. salvar todo o conteúdo da memória em disco, e então rodar o próximo programa, sim.
      - Basta que haja apenas um na memória por vez.
      - Este conceito se chama swapping (mais adiante)

Abstrações de Memória

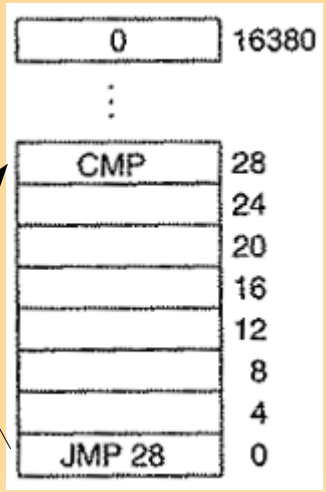
- Sem abstração:
  - Mas, sendo os programas pequenos, não poderíamos tê-los na memória ao mesmo tempo?

Abstrações de Memória

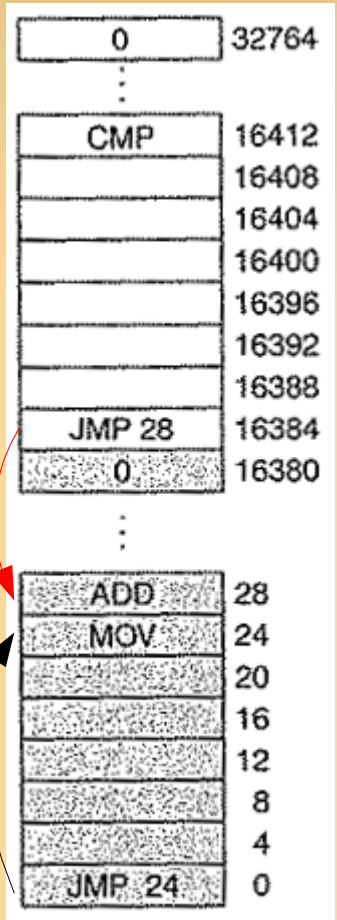
- Sem abstração:
  - Mas, sendo os programas pequenos, não poderíamos tê-los na memória ao mesmo tempo?
    - Lembre que os programas vêem a memória toda, e não sabem que outros programas também estão rodando:



P1



P2



P1 e P2, alocados em seqüência

Abstrações de Memória

- Sem abstração:
  - Uma solução seria adicionar um valor-base a cada endereço usado no segundo processo
    - Caro, do ponto de vista computacional – podemos ter que fazer muitas substituições
    - Exige distinguir entre o que é endereço e o que é valor numérico, nas diferentes instruções
  - Tipo de esquema ainda usado
    - Smart cards
    - Eletrodomésticos

O usuário não controla que programas serão carregados

Abstrações de Memória

- Multiprogramação revista:
  - Melhora o uso da CPU – sem novidade
  - Em quanto?

Abstrações de Memória

- Multiprogramação revista:
  - Melhora o uso da CPU – sem novidade
  - Em quanto?
    - Depende do quanto de E/S é feito
    - Suponha que os processos fiquem em execução efetiva apenas 20% do tempo em que reside na memória
      - Quantos processos deveriam estar na memória para ocupar a CPU 100%?

Abstrações de Memória

- Multiprogramação revista:
  - Melhora o uso da CPU – sem novidade
  - Em quanto?
    - Depende do quanto de E/S é feito
    - Suponha que os processos fiquem em execução efetiva apenas 20% do tempo em que reside na memória
      - Quantos processos deveriam estar na memória para ocupar a CPU 100%?
      - Em tese, 5
      - E qual o problema com essa visão?

Abstrações de Memória

- Multiprogramação revista:
  - Melhora o uso da CPU – sem novidade
  - Em quanto?
    - Depende do quanto de E/S é feito
    - Suponha que os processos fiquem em execução efetiva apenas 20% do tempo em que reside na memória
      - Quantos processos deveriam estar na memória para ocupar a CPU 100%?
      - Em tese, 5
      - E qual o problema com essa visão?
        - Presume que dois processos não estarão esperando, simultaneamente, por E/S

Abstrações de Memória

- Multiprogramação revista:
  - Em quanto?
    - Olhemos de um ponto de vista probabilístico
    - Suponha que um processo gaste uma fração $0 \leq p \leq 1$ 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:

Abstrações de Memória

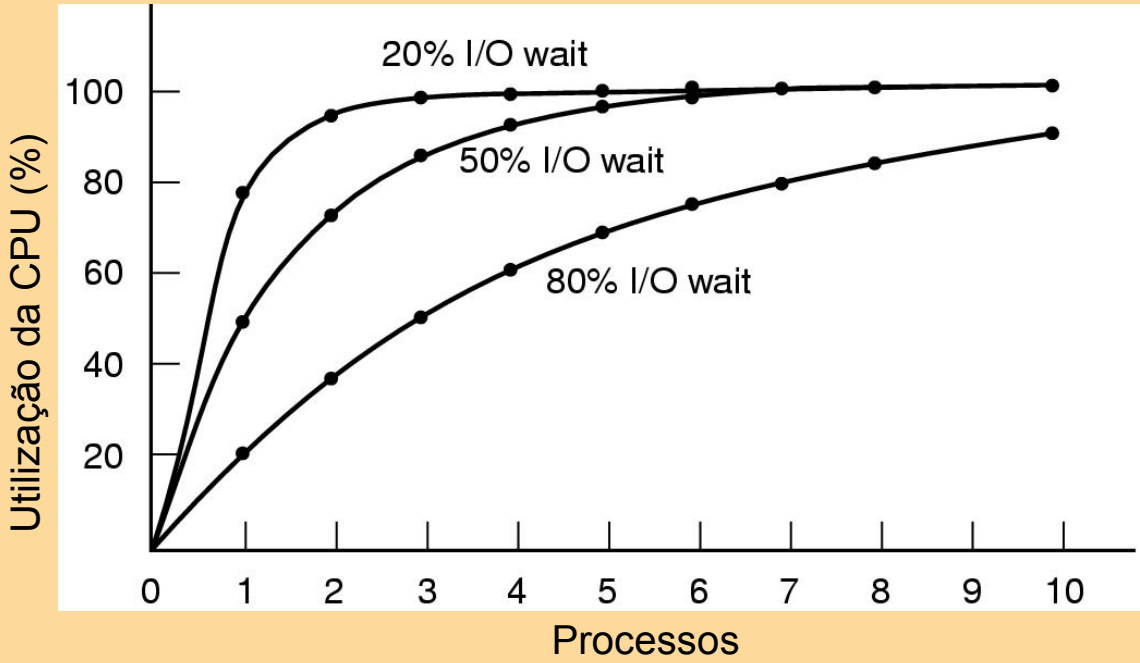
- Multiprogramação revista:
  - Em quanto?
    - Olhemos de um ponto de vista probabilístico
    - Suponha que um processo gaste uma fração $0 \leq p \leq 1$ de seu tempo esperando pela finalização de sua solicitação de E/S
    - Com n processos simultâneos na memória, a probabilidade de todos os n processos estarem esperando por E/S (situação em que a UCP está ociosa) é:
      - P^n

Abstrações de Memória

- Multiprogramação revista:
  - Em quanto?
    - Considerando a Utilização da CPU como a fração do tempo dos programas em que não estão esperando E/S ao mesmo tempo, temos

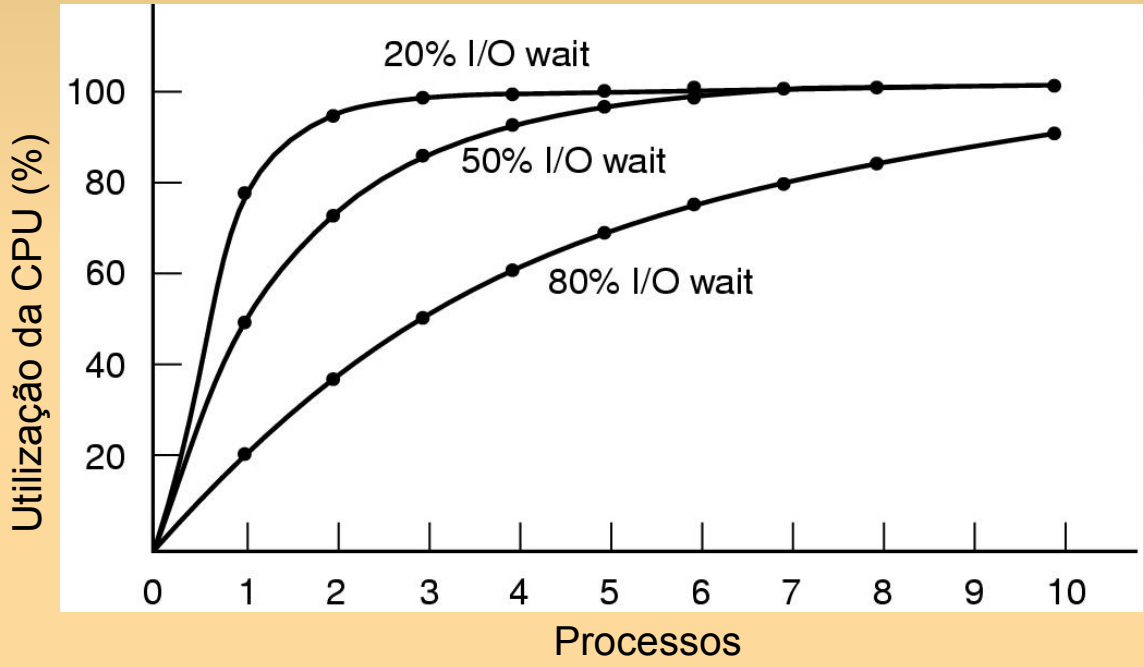
Utilização da UCP = $1 - p^n$

Pressupõe:
• Pior caso (todo ao mesmo tempo)
• Independência dos processos



Abstrações de Memória

- Multiprogramação revista:
  - Se processos gastarem 80% em E/S, precisaremos de no mínimo 10 processos na memória para que a ociosidade da CPU seja mantida inferior a 10%



    - 80% são comuns, quando se espera pelo usuário
  - Mais memória → melhor aproveitamento da CPU

Abstrações de Memória

- Multiprogramação e memória:
  - Como fazer para armazenar n processos na memória?

Abstrações de Memória

- Multiprogramação e memória:
  - Como fazer para armazenar n processos na memória?
    - Divida a memória em n partições, de tamanho fixo, não necessariamente iguais (solução mais simples)
      - Pode ser feita de modo manual, quando o sistema é iniciado
    - Ao chegar um job, coloque-o na fila de entrada associada à menor partição capaz de armazená-lo
      - O espaço que sobrar, não será utilizado

Abstrações de Memória

- Multiprogramação e memória:
  - Para permitir que múltiplas aplicações estejam na memória, precisamos resolver dois problemas: Proteção e Realocação
  - Proteção:
    - Com várias partições e programas ocupando diferentes espaços da memória é possível acontecer um acesso indevido;
      - Como proteger os processos uns dos outros e o kernel de todos os processos?
      - Deve-se manter um programa fora das partições de outros processos

Abstrações de Memória

- Multiprogramação e memória:
  - Realocação:
    - Como carregar processos em regiões da memória diferentes das explicitamente definidas em seu código?
    - Quando um programa é linkado (programa principal + rotinas do usuário + rotinas da biblioteca → executável) o linker deve saber em que endereço o programa irá iniciar na memória;
    - Nesse caso, para que o linker não escreva em um local indevido, como por exemplo na área do SO (100 primeiros endereços), é preciso de realocação:
      - #100 + Δ → que depende da partição!!!

Relocação e proteção

- Multiprogramação fica difícil com endereçamento direto
  - Podemos não ter certeza de onde o programa será carregado na memória
    - As localizações de endereços das variáveis e do código das rotinas não podem ser absolutos
- Solução para ambos os problemas:
  - Uma abstração – o espaço de endereços
  - Conjunto de endereços que um processo pode usar para endereçar memória
    - Cada processo tem seu próprio espaço de endereços

Relocação e proteção

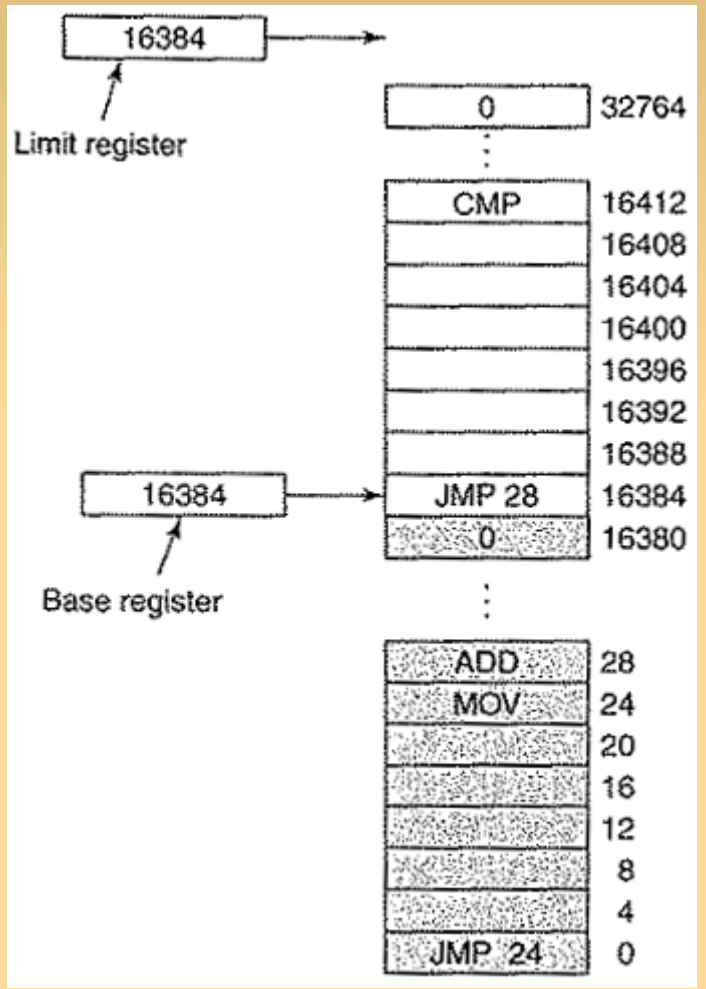
- Solução para ambos os problemas:
  - Como dar a cada programa seu próprio espaço de endereços, de modo a que o endereço 28 em um seja diferente, na memória física, do 28 em outro?
    - 2 registradores → base e limite
      - Protegidos pelo hardware contra modificações pelos usuários
    - Quando um processo é escalonado o registrador-base é carregado com o endereço de início da partição alocada ao processo, e o registrador-limite com o tamanho da partição;
      - Toda vez que um processo referencia a memória, a CPU automaticamente adiciona o valor base ao endereço
      - Ao mesmo tempo, checa se o endereço referenciado é maior ou igual que o limite

Relocação e proteção

- Solução para ambos os problemas:
  - Como dar a cada programa seu próprio espaço de endereços, de modo a que o endereço 28 em um seja diferente, na memória física, do 28 em outro?
    - O registrador-base torna impossível a um processo uma remissão a qualquer parte de memória abaixo de si mesmo
      - Previne acessos ao espaço de endereço de outro programa
  - Técnica bastante simples, mas que caiu em desuso
    - Deu lugar a mais complicadas

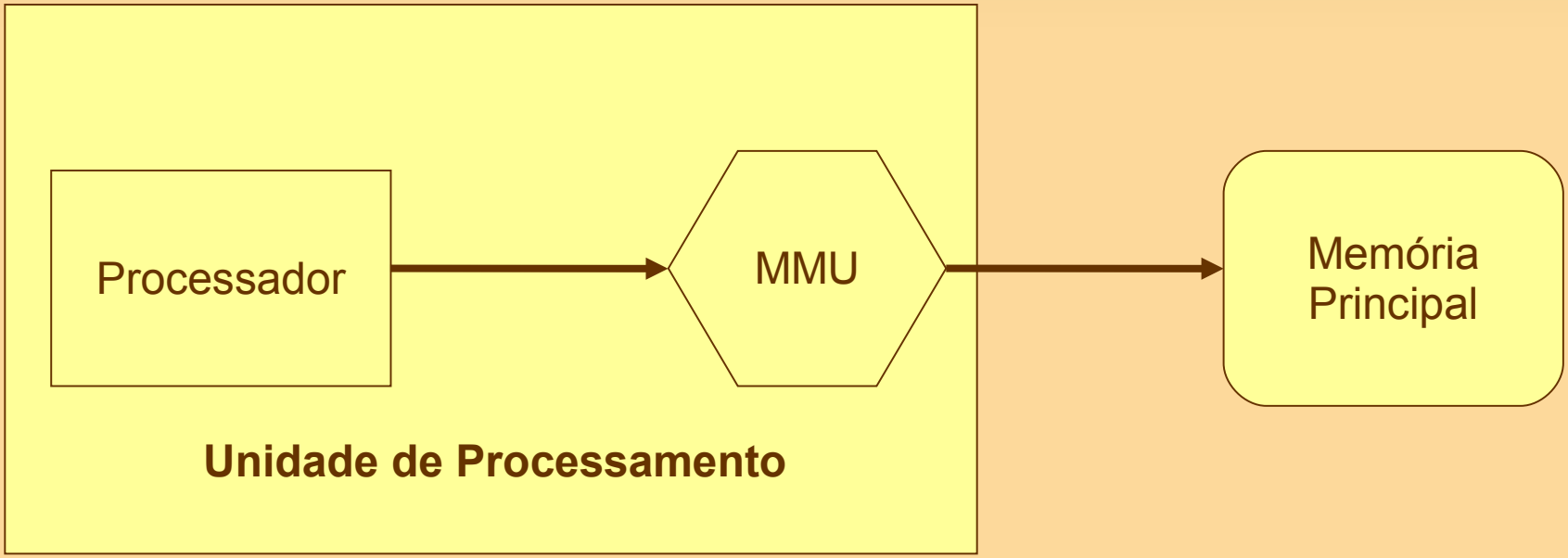
Relocação e proteção

- Registrador 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 são considerados erro
  - Desvantagem:
    - Faz uma soma e comparação a cada referência à memória



Gerenciamento de Memória

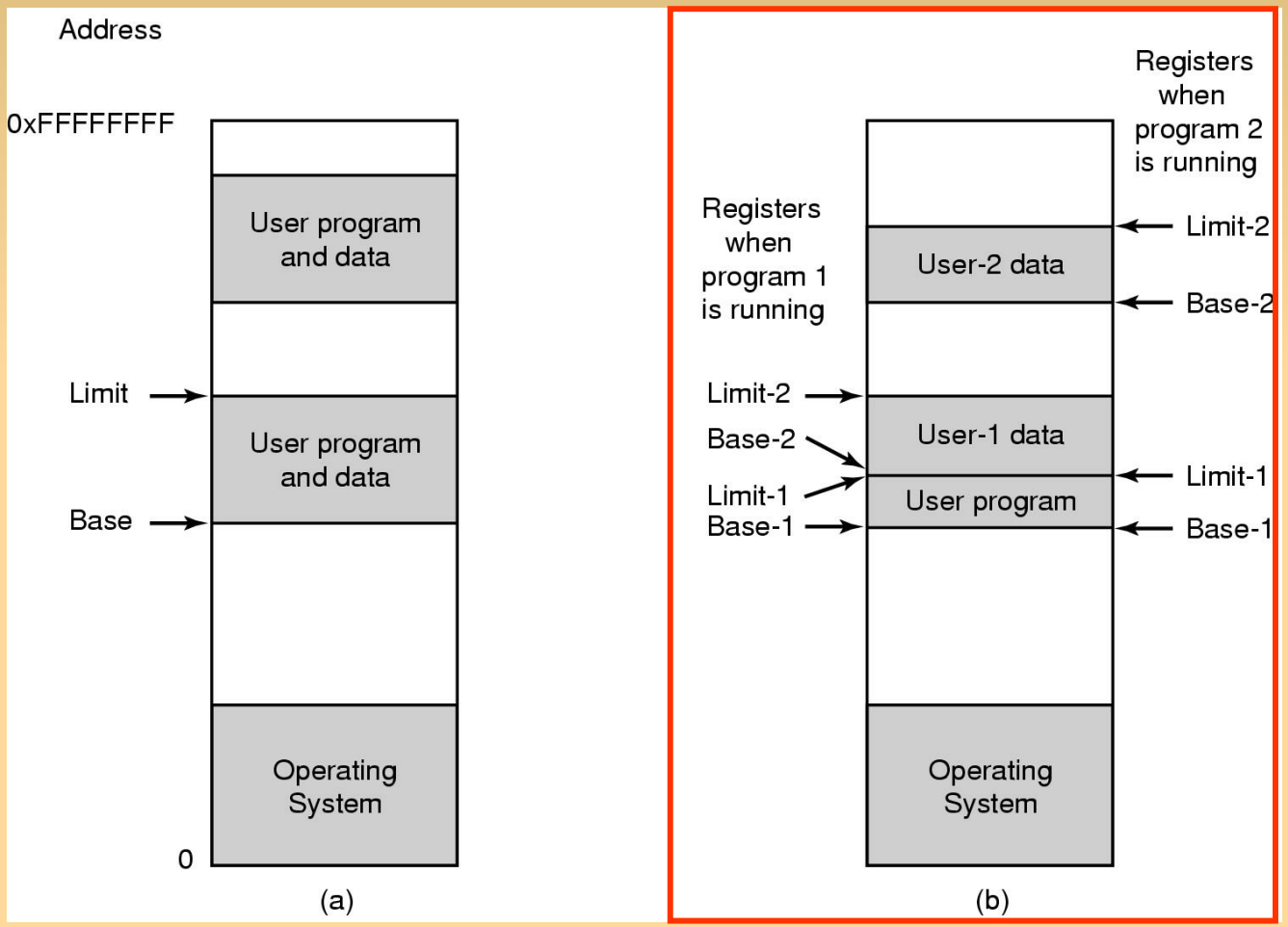
- Todas essas soluções envolvem equipar a CPU com um hardware especial → MMU (memory management unit);



Gerenciamento de Memória

- MMU:
  - Dispositivo de hardware que transforma endereços virtuais em endereços físicos.
  - Na MMU, o valor no registro de realocação é adicionado a todo o endereço lógico gerado por um processo do utilizador na altura de ser enviado para a memória.
    - O programa do utilizador manipula endereços lógicos; ele nunca vê endereços físicos reais.

Gerenciamento de Memória



(b) MMU mais sofisticada → dois pares de registradores: segmento de dados usa um par separado;

- MMUs modernas têm mais pares de registradores.

Gerenciamento de Memória

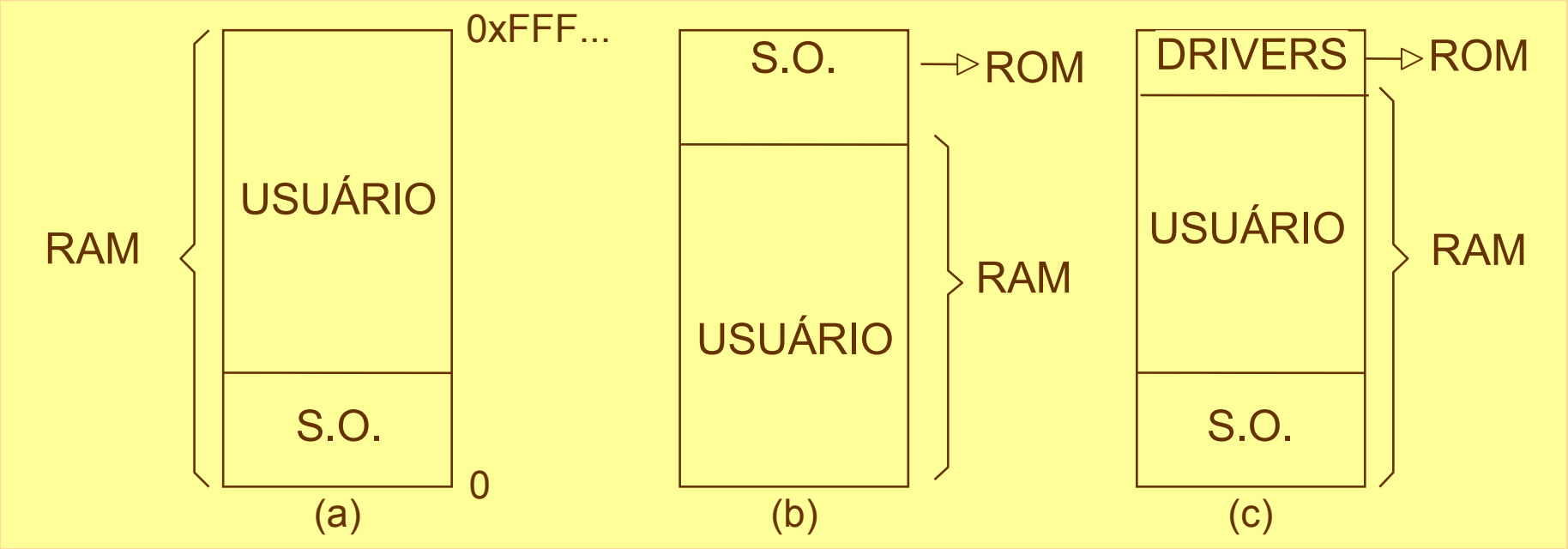
- A MMU divide o espaço de endereçamento virtual (endereços utilizados pelo processador) em páginas
  - Tamanho de 2^n, tipicamente poucos kilobytes.
- A MMU normalmente traduz o número de páginas virtuais para um número de páginas físicas
  - Usa uma cache associada chamada Translation Lookaside Buffer (TLB). Quando o TLB falha em uma tradução, um mecanismos mais lento envolvendo um hardware específico de dados estruturados ou um software auxiliar é usado.

Gerenciamento de Memória

- Tipos básicos de gerenciamento:
  - Com paginação (chaveamento):
    - Processos são movidos, durante a execução, entre a memória principal e o disco
    - Usada para resolver o problema da falta de memória;
    - Se existe memória suficiente, não há necessidade de se ter paginação;
  - Sem paginação:
    - Não há chaveamento;

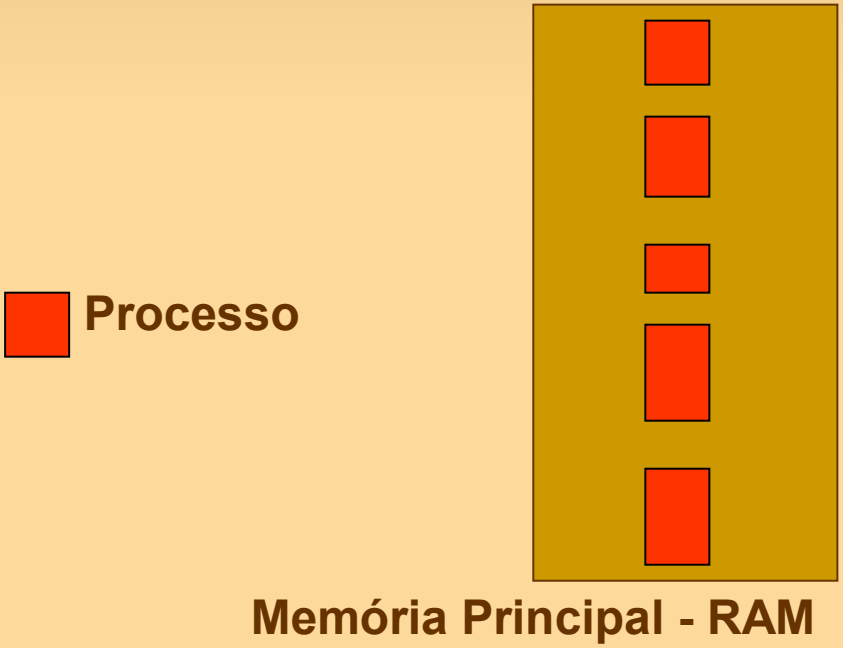
Gerenciamento de Memória

- Monoprogramação:
  - Sem paginação: gerenciamento mais simples;
  - Apenas um processo na memória;



Gerenciamento de Memória

- Modelo de Multiprogramação:
  - Múltiplos processos sendo executados;
  - Eficiência da CPU;
  - Paginação

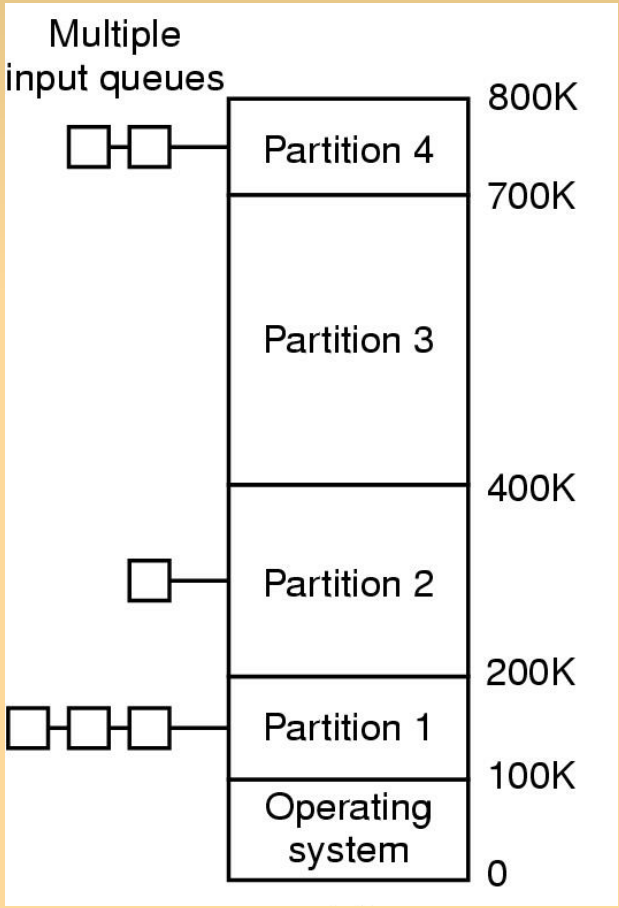


Gerenciamento de Memória Partições/Alocação

- Particionamento da memória pode ser realizado de duas maneiras:
  - Partições fixas (ou alocação estática);
  - Partições variáveis (ou alocação dinâmica);
- Partições Fixas:
  - Tamanho e número de partições são fixos (estáticos);
  - Não é atrativo, porque partições fixas tendem a desperdiçar memória (Qualquer espaço não utilizado é literalmente perdido)
  - Mais simples;

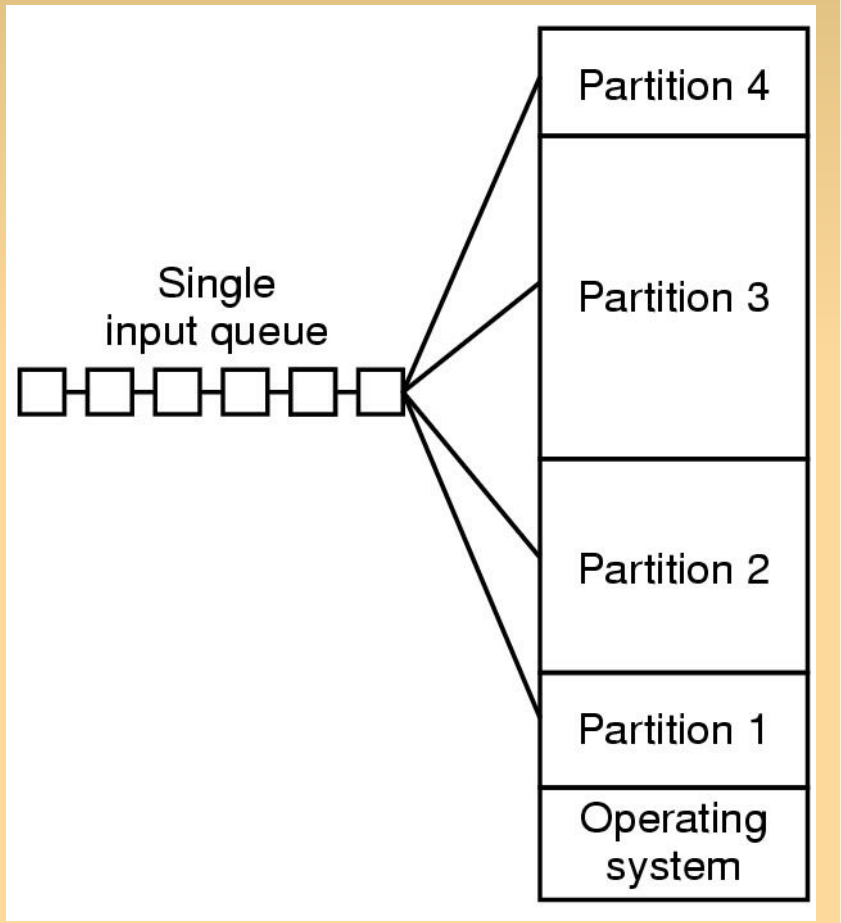
Multiprogramação com partições Fixas

- Partições de memória fixas
  - fila separada para cada partição
    - Coloque o programa na menor partição em que caiba
    - Problema:
      - Filas não balanceadas
      - Ex: a fila em uma grande partição está vazia (3), enquanto que a de uma pequena partição está cheia (1)
      - Processos pequenos terão que esperar, mesmo havendo memória disponível suficiente para acomodá-los



Multiprogramação com partições Fixas

- Partições de memória fixas
  - Uma única fila de entrada
    - Sempre que uma partição se torna disponível, o job mais próximo ao início da fila, e que caiba na partição, é carregado
    - Implementação com Lista:
      - Melhor utilização da memória, pois procura o melhor processo para a partição considerada;
    - Problema:
      - Processos menores são prejudicados;



Gerenciamento de Memória Partições Fixas

- Problemas com fragmentação:
  - Interna: desperdício dentro da área alocada para um processo;
    - Ex.: processo de tamanho 40K ocupando uma partição de 50k;
  - Externa: desperdício fora da área alocada para um processo;
    - Duas partições livres: PL1 com 25k e PL2 com 100k, e um processo de tamanho 110K para ser executado;
    - Livre: 125K, mas o processo não pode ser executado;

Gerenciamento de Memória
Partições Variáveis

- Partições de memória variáveis:
  - Tamanho e número de partições variam;
  - Otimiza a utilização da memória, mas complica a alocação e liberação da memória;
  - Partições são alocadas dinamicamente;
  - SO mantém na memória uma lista com os espaços livres;
  - Menor fragmentação interna e grande fragmentação externa;

Gerenciamento de Memória
Partições Variáveis