# Sistemas Operacionais


## Gerenciamento de memória

Prof. José Roberto Bezerra

# Agenda

- Lei de Parkinson
- Gerenciamento de memória
- Relocação e proteção
- Grau de multiprogramação
- *Swapping*
- Memória virtual
- *Memory Management Unit*
- Paginação de memória
- Algoritmos de substituição de página

# Lei de Parkinson

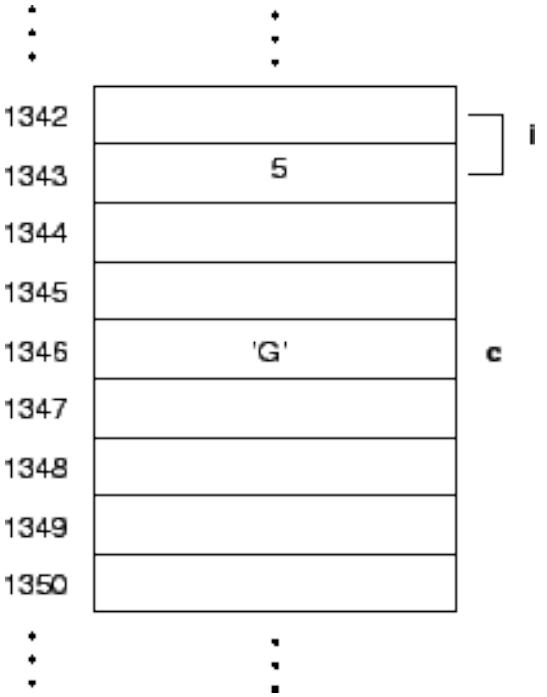
Programas tendem a se expandir ocupando toda a memória disponível

# Gerenciamento de memória

Manter o controle das partes da memória em uso e em desuso, alocando memória aos processos quando necessário e liberando memória quando finalizam, além de gerenciar a troca de processos (*swapping*) entre a memória e o disco quando a memória principal não é suficiente

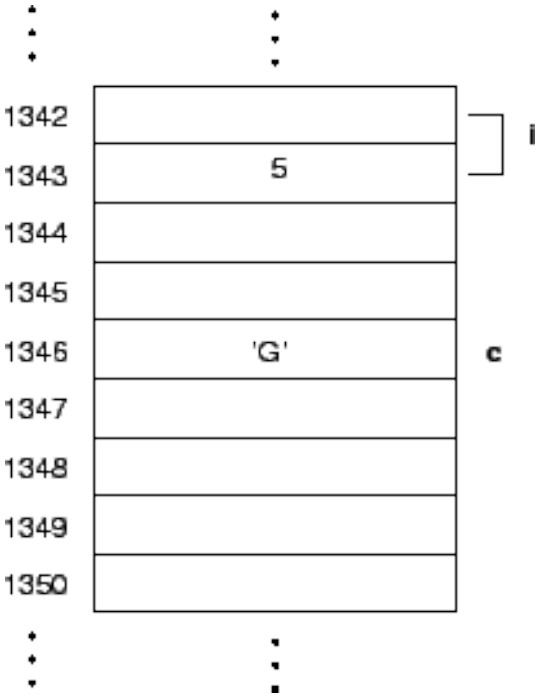
# Abstração de memória

- Os primeiros computadores não possuíam nenhum tipo de abstração de memória
- Apenas a memória física era considerada
- Era composta por um conjunto de endereços numerados



# Abstração de memória

- Desta maneira apenas um programa por vez poderia ser executado
- Toda a memória presente no sistema era disponibilizada para tal programa

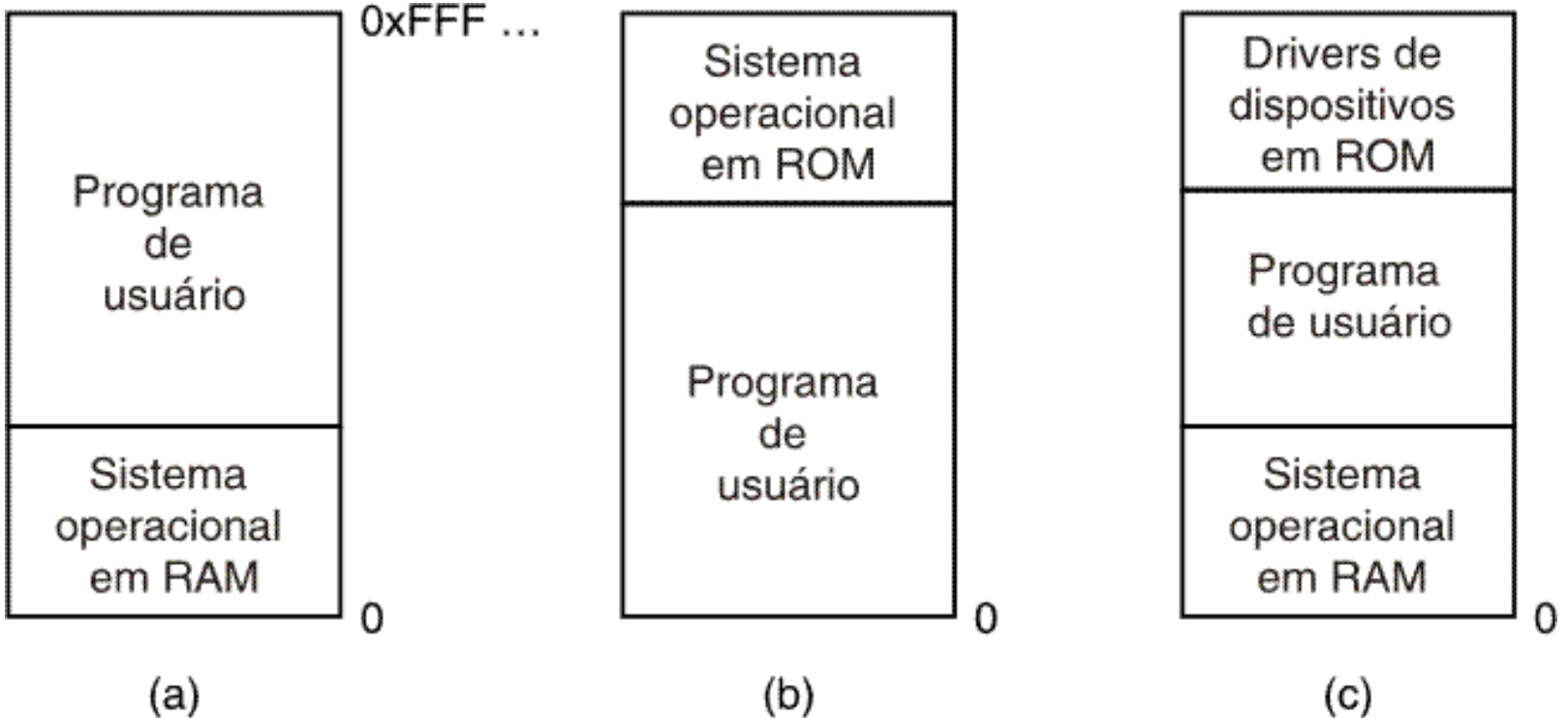


# Gerenciamento básico

- Monoprogramação sem troca de processos
- Multiprogramação com partição fixas

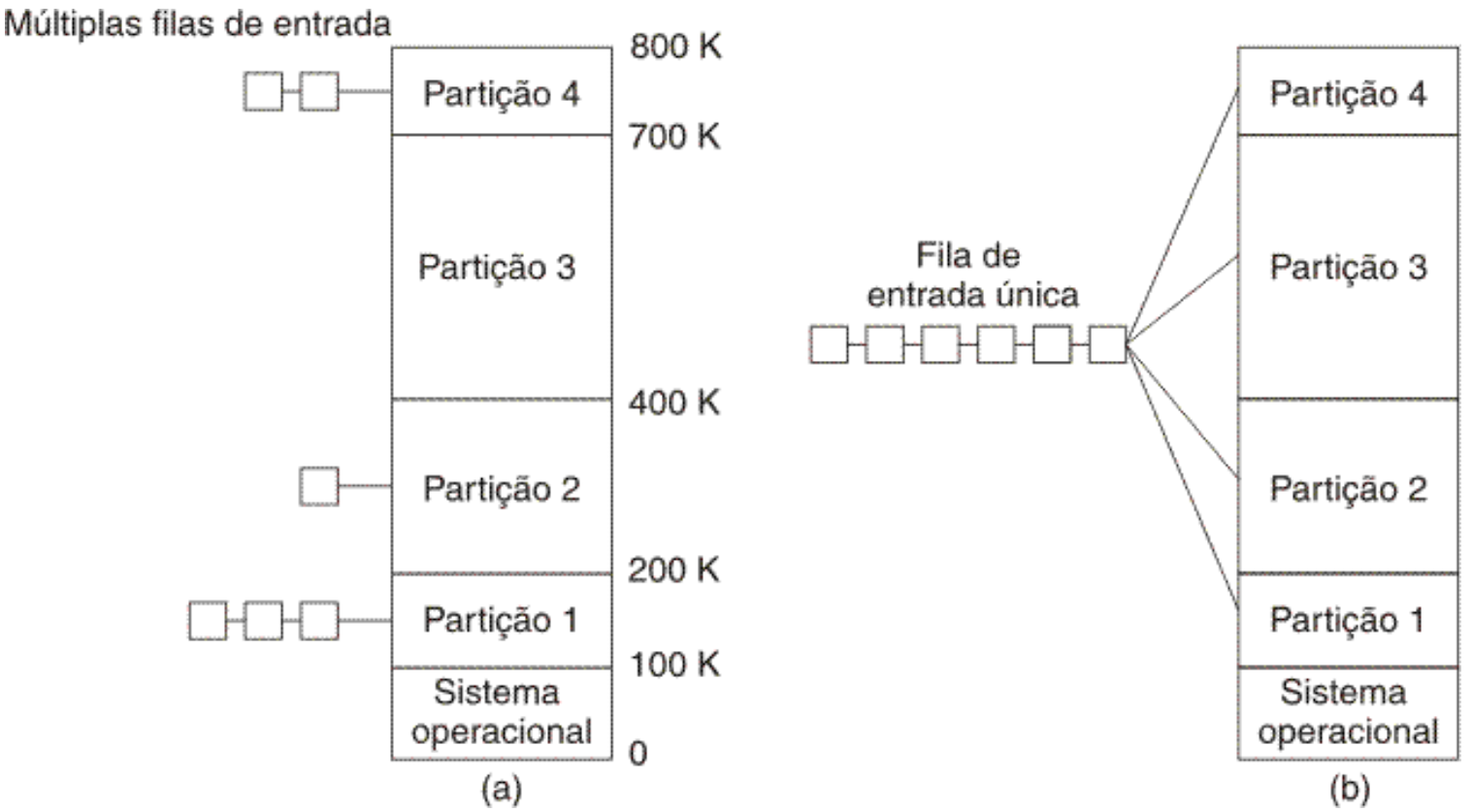
# Monoprogramação sem troca de processos

- Sistemas de baixa complexidade
- Compartilhamento entre programas e SO
- Não há alternância entre processos, apenas um programa é executado por vez



# Multiprogramação com partições fixas

- Divide-se a memória em n partições
- Cada tarefa é colocada em uma fila a uma partição que lhe caiba
- Se o processo não utiliza todo espaço da partição o excedente é perdido
- Uma abordagem alternativa utiliza uma fila única para todas as partições
- Exemplo: IBM/OS360


Múltiplas filas de entrada
800 K
Partição 4
700 K
Partição 3
400 K
Partição 2
200 K
Partição 1
100 K
Sistema operacional
0
(a)
Fila de entrada única
Partição 4
Partição 3
Partição 2
Partição 1
Sistema operacional
(b)

# Espaço de endereçamento

Conjunto de endereços de memória que um processo pode utilizar para armazenar informações. Cada processo tem seu próprio espaço de endereçamento independente de outros processos

# Espaço de endereçamento

- Em resumo, o EE tem o objetivo de fazer com que o endereço 28 de um programa seja apontado para um endereço físico diferente do endereço 28 de outro programa
- O mapeamento do EE de um processo em uma parte diferente da memória física é a chamada realocação dinâmica
- A RD é feita através de registradores

# Realocação e proteção

- Diferentes processos são executados em diferentes áreas de memória
- *Linker*
- Registrador base e limite
  - É carregado com o endereço do início da partição
  - RL é carregado com o tamanho da partição
  - Cada troca estes registradores são atualizados

# Grau de multiprogramação

- Supondo que um processo gaste uma fração p de seu tempo com aguardando E/S
- Com *n* processos na memória a probabilidade de todos estarem esperando por E/S é $p^n$
- A utilização de CPU é dada pela fórmula:
  - $U = 1 - p^n$
- *n* é chamado grau de multiprogramação

# Grau de multiprogramação

- Se processos gastam 80% do tempo com E/S
- Pelo menos 10 processos devem estar na memória para um percentual de utilização da CPU de 90%

# Exemplo

- Supondo um sistema com 32MB de memória
  - 16Mb para o SO
  - Cada programa do usuário ocupa 4MB
  - Considerando que cada processo gasta 80% do tempo esperando E/S, de quanto será a utilização da CPU?

# Exemplo

- Acrescentando-se mais 16MB permite a utilização de 8 programas na memória (grau de multiprogramação 8)
  - De quanto será a utilização da CPU?
- Acrescentando-se mais 16MB de quanto será a utilização da CPU?

# Conclusão

O modelo permite avaliar de maneira simplificada se a ampliação de memória apresentará uma significativa melhora no desempenho do sistema ou não

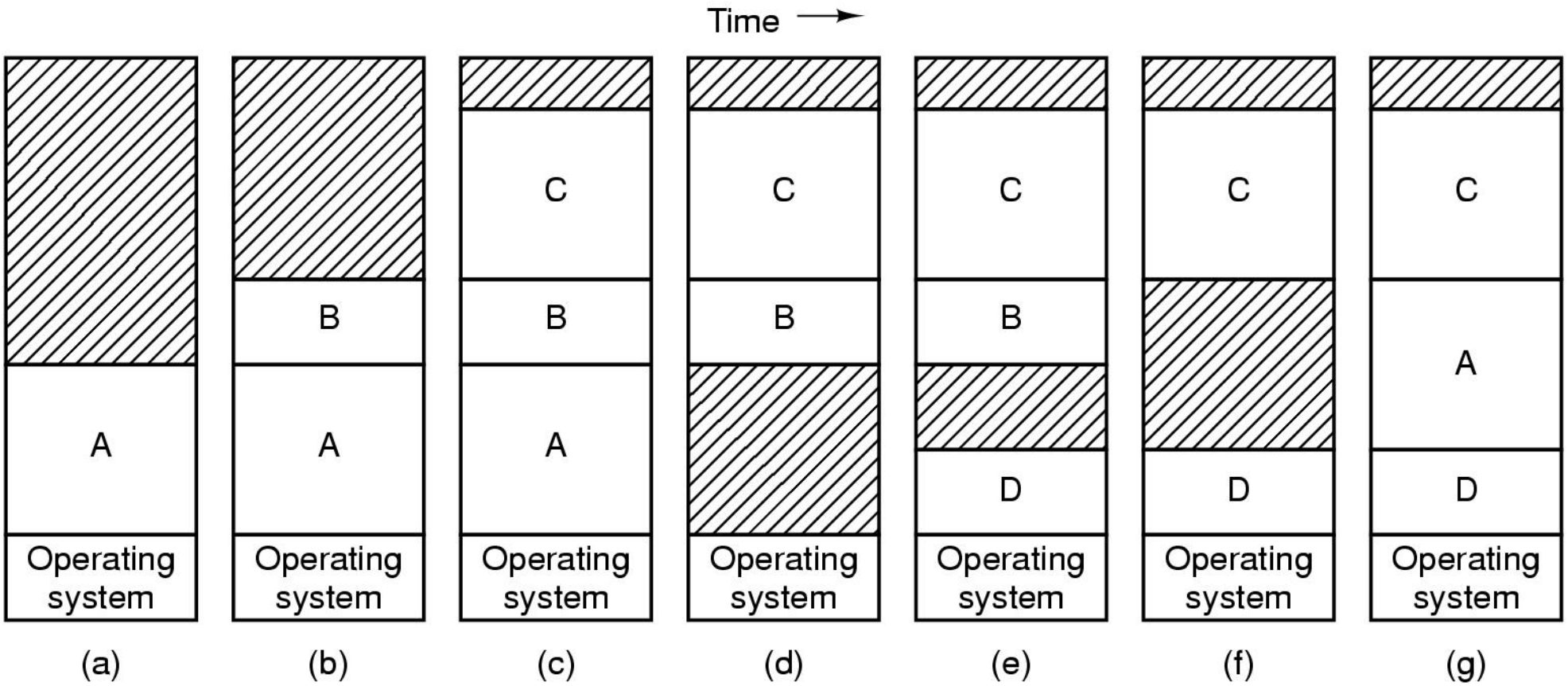
# Disco como memória complementar

- Em geral, não há memória principal suficiente para manter todos os processos ao mesmo tempo
- Sistemas de tempo compartilhado e Interfaces Gráficas
- Assim, utiliza-se o disco como memória complementar a memória principal
  - *Swapping*
  - Memória Virtual

# *Swapping* e Memória virtual

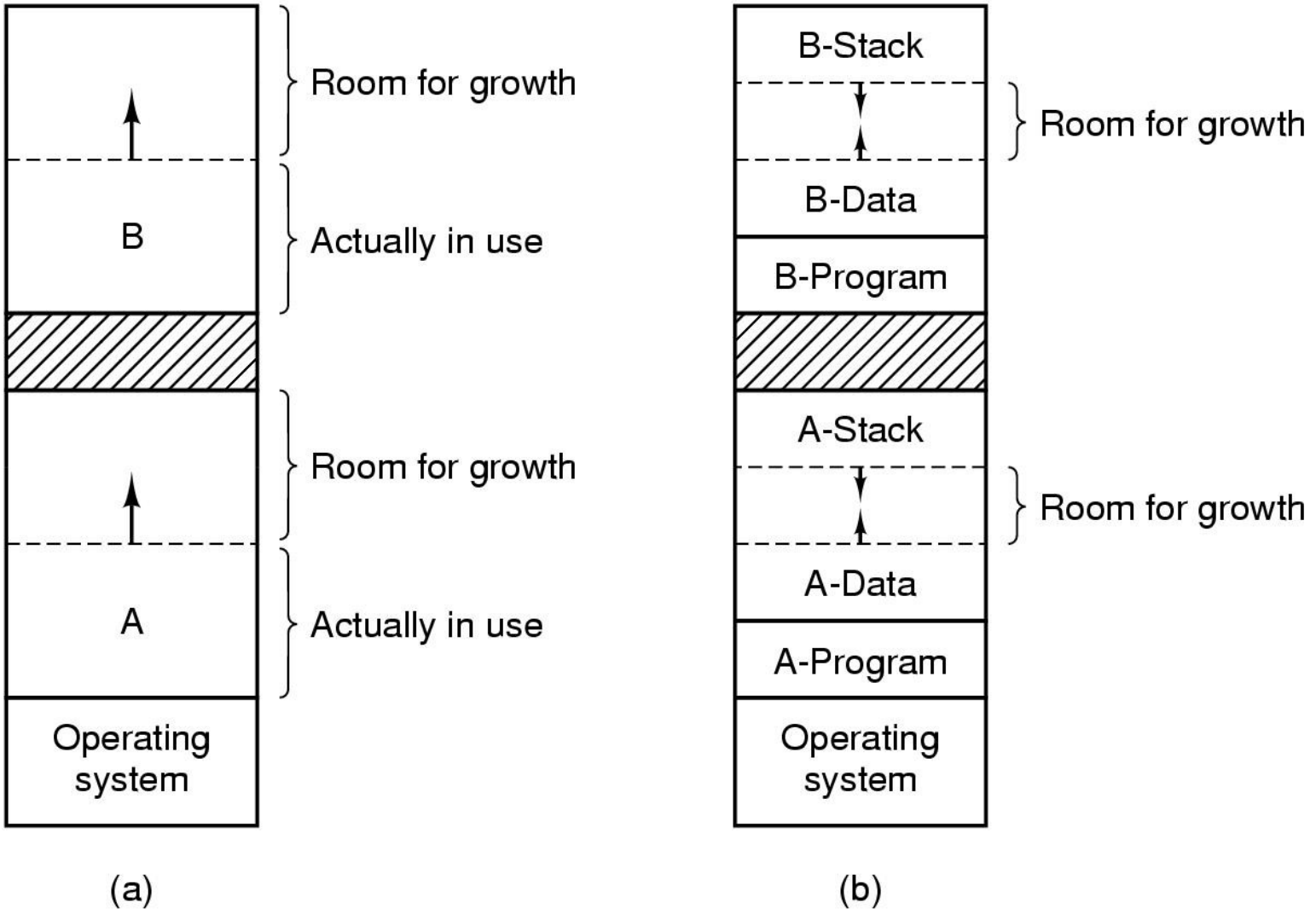
- *Swapping*
  - Consiste em carregar o processo na MP executá-lo durante um período, retirar da MP e em seguida levá-lo para o disco novamente. Quando o processo ganha tempo de CPU novamente, é copiado do disco para MP e o ciclo se repete
- Memória Virtual
  - Permite que programas possam ser executados mesmo que não totalmente carregados na MP, usa o disco como memória complementar

# Swapping



# Questões relacionadas ao *Swapping*

- Qual o espaço que deve ser alocado aos processos na memória?
- O espaço pode aumentar no decorrer da execução?


Room for growth
B
Actually in use
Room for growth
A
Actually in use
Operating system
(a)
B-Stack
Room for growth
B-Data
B-Program
A-Stack
Room for growth
A-Data
A-Program
Operating system
(b)

# Memória virtual

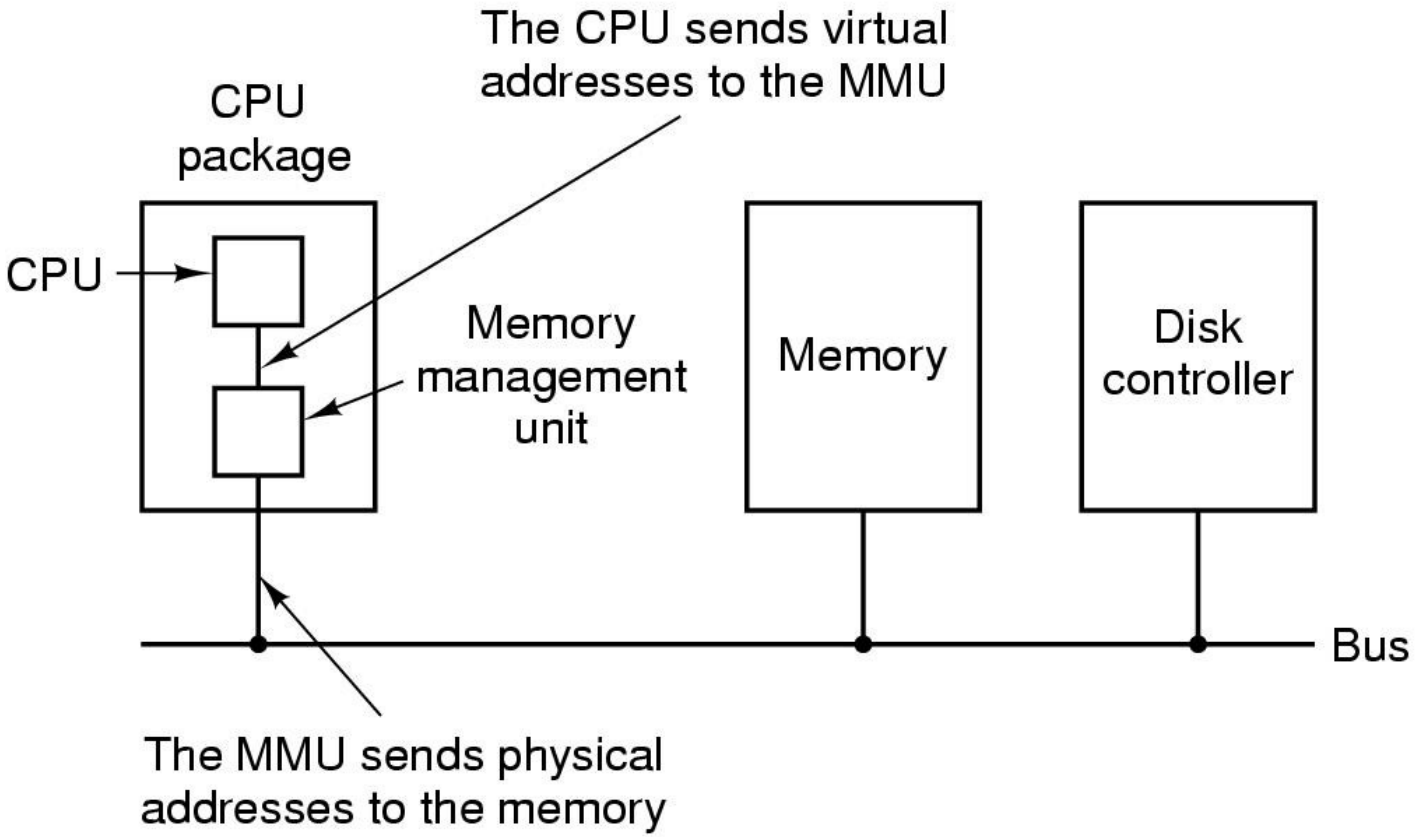
- Relaciona-se com a capacidade de executar programas maiores do que a própria memória disponível
  - Exemplo: programa de 16MB executar em uma máquina com 4MB de memória
- Programas eram subdivididos pelo próprio programador em camadas chamadas, ***overlays***
  - O SO era responsável pela carga de cada *overlay*
- Atualmente o SO faz a divisão e gerenciamento das partes (páginas)

# Endereços virtuais e físicos

- Endereços virtuais
  - São endereços gerados pelos programas e **não** coincidem com os endereços físicos da memória
  - O espaço de endereçamento virtual é **maior** que a capacidade de endereçamento físico
- Endereços físicos
  - São os endereços reais onde são armazenadas as informações
  - Computadores que **não** utilizam memória virtual possuem espaços de endereçamento físico e virtual **idênticos**

# MMU

- Memory Management Unit
- Responsável por fazer o mapeamento entre os endereços virtuais e físicos
-


The CPU sends virtual addresses to the MMU
CPU package
CPU
Memory management unit
Memory
Disk controller
Bus
The MMU sends physical addresses to the memory

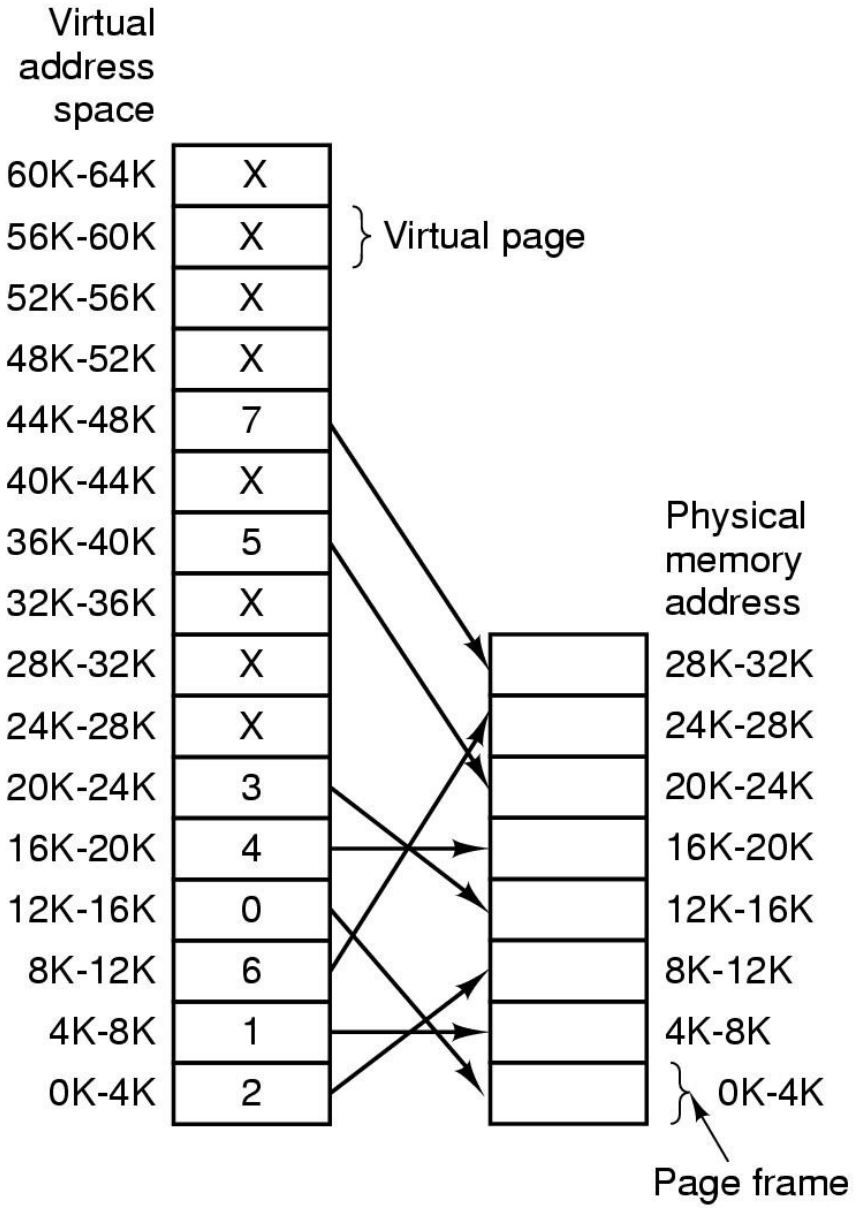
# Mapeamento

- Exemplo
  - Computador que gera endereços virtuais de 16bits (0 a 64K)
  - Memória física de 32K
  - Embora seja possível alocar programas de 64K, não é possível carregá-los totalmente na memória física
  - A cópia completa do programa deve permanecer no disco e carregada dinamicamente (**paginação**)

# Paginação

- Divide-se o espaço de endereçamento virtual em **Páginas** (*pages*)
- Cada página é associada com uma outra correspondente no espaço de endereçamento físico chamadas **Molduras de Página** (*page frames*)
- Páginas e Molduras de Página são sempre do mesmo tamanho

# Paginação



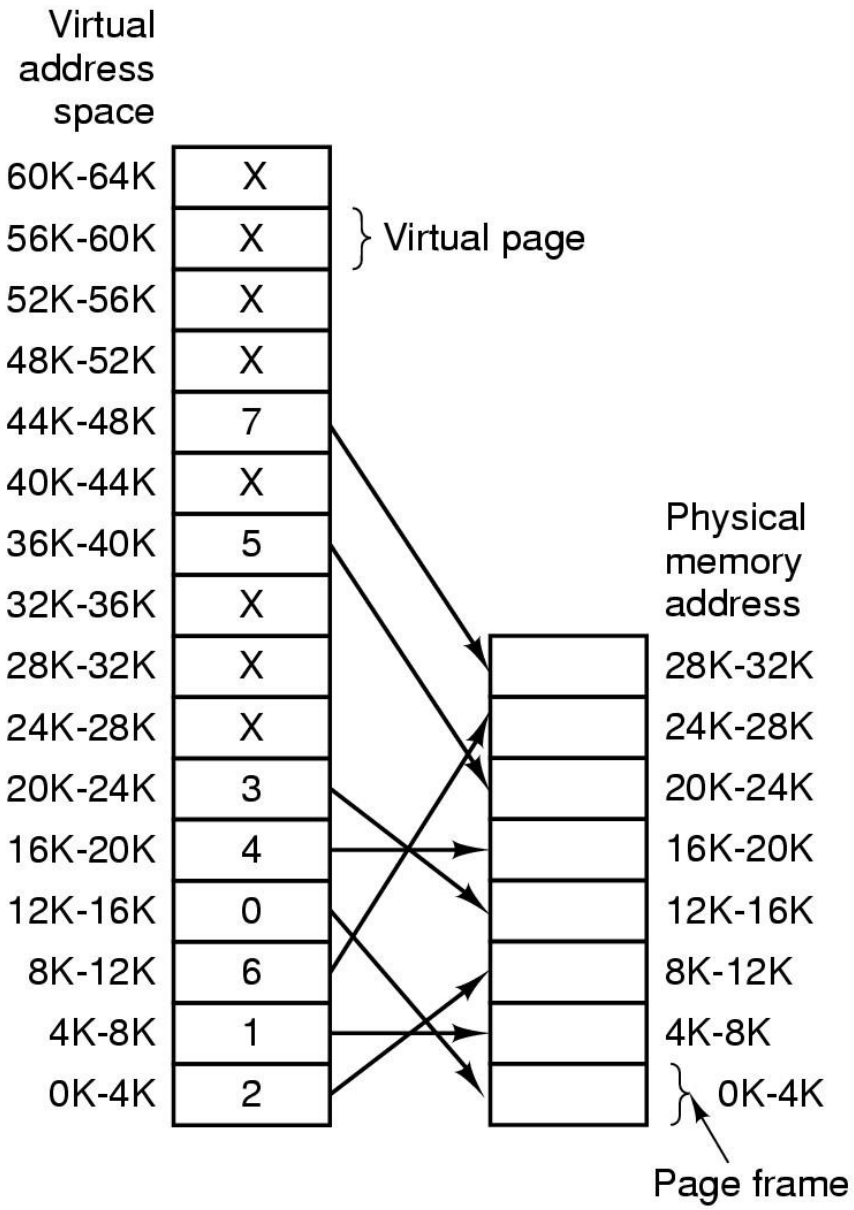

- Comandos na CPU
  - `MOV REG,0`
  - `MOV REG,8192`
  - Que endereços reais estão sendo acessados?
- O que acontce no comando
  - `MOV REG,24676`
- **Page fault**
  - Interrupção ***trap***

# ***Trap***

- A interrupção ***trap*** força um desvio para o SO
- O SO escolhe uma moldura de página pouco utilizada e a salva em disco
- Em seguida, carrega a página referenciada pela instrução na moldura de página recém liberada
- Atualiza o mapeamento da tabela de páginas e reinicia a instrução causadora da interrupção *trap*

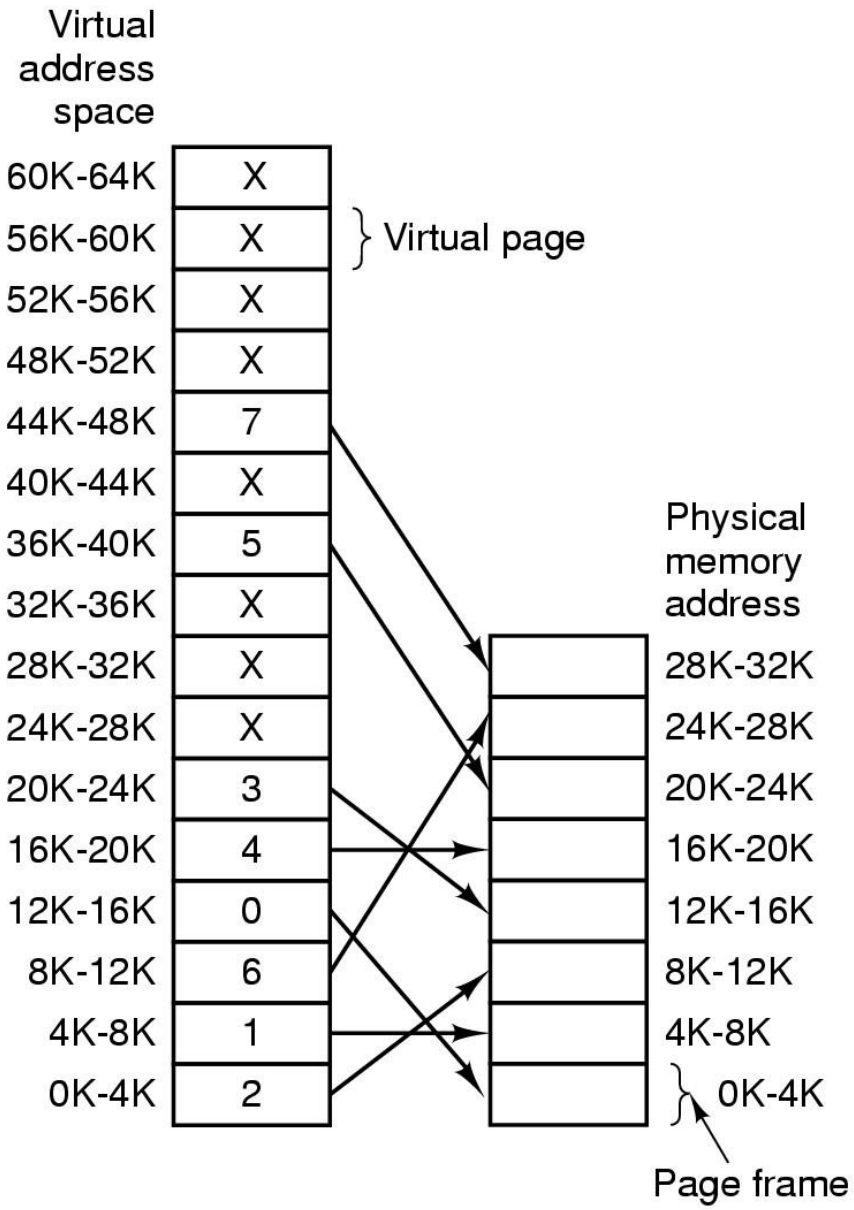
# Exemplo de trap

- No caso da moldura de página 1 ser escolhida para substituição
- A página 8 deverá ser carregada a partir do endereço físico 4K
-



# Exemplo de trap

- A página virtual 1 passará a não mapeada, acessos futuros causarão uma interrupção *trap*
- A página virtual 8 passará de X para 1



# Algoritmos de substituição de páginas

- Ótimo
- NRU (Não Usada Recentemente)
- FIFO (*First In First Out*)
- Segunda Chance (SC – *Second Chance*)
- Menos Recentemente Usada
- *Working Set* (WS)

# Regras gerais

- Faltas de Página geram necessidade de reposição
- Páginas modificadas devem ser salvas (no disco) antes que sejam substituídas
- O ideal é não escolher páginas frequentemente utilizadas, pois a probabilidade desta ser requisitada novamente é alta

# Regras gerais

- Bits de status: R (referenciada) e M (modificada)
  - R é colocado em 1 sempre que a página for referenciada (lida ou escrita)
  - M é colocado em 1 sempre que a página for modificada

# Algoritmo ótimo

- Consiste em rotular as páginas com a quantidade de instruções que serão executadas antes da página ser referenciada (10, 20, 100, 200 instruções)
  - Quando ocorrer falta de página aquela que possuir o maior rótulo deve ser removida
- Algoritmo Ótimo é **irrealizável**, pois não é possível prever quantas instruções serão executadas antes de uma página ser referenciada

# Algoritmo ótimo

- Usado como referência para desempenho de outros algoritmos
  - Executando inicialmente em um simulador e guardando as referências às páginas pode-se implementar o algoritmo ótimo em uma segunda execução do mesmo programa com os mesmos dados iniciais
  - Em uma segunda execução o AO poderia ser implementado e seu desempenho considerado o mais eficiente
  - Assim todos os outros algoritmos poderiam ser comparados tendo este resultado como referência

# Não recentemente usado (NRU)

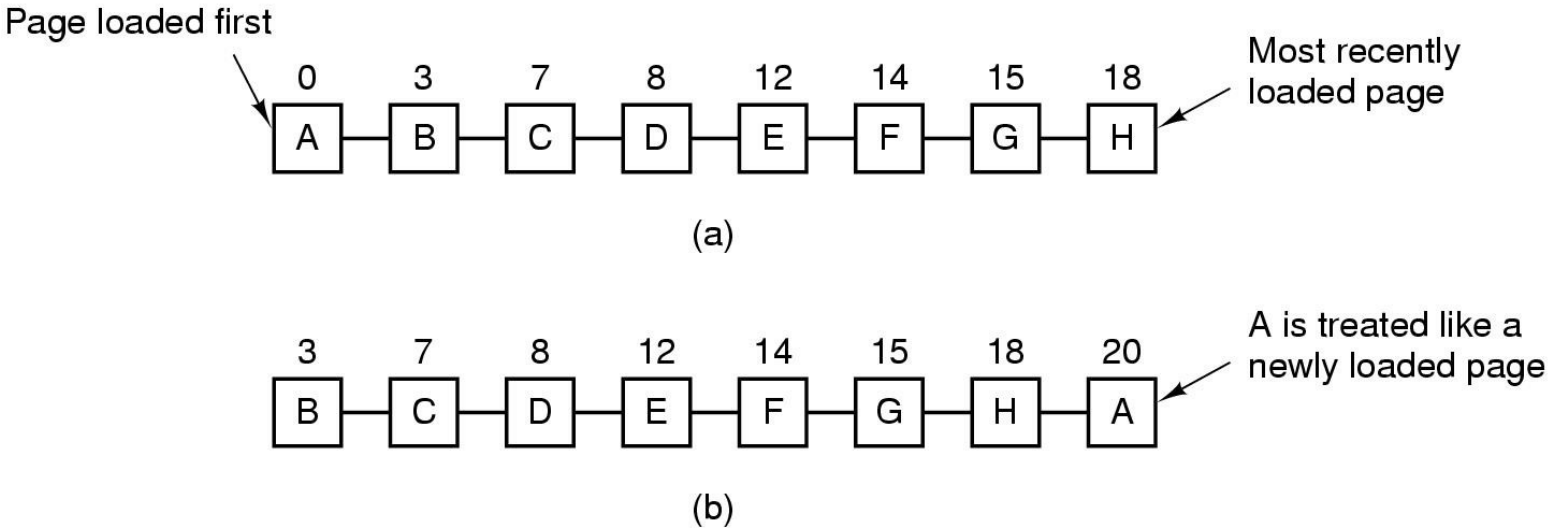
- Quando um processo é iniciado os bits R e M de suas páginas recebem 0 dado pelo SO
  - A cada ciclo de *clock* R é limpo para diferenciar as páginas utilizadas recentemente
- A cada falta de página o SO inspeciona as páginas e classifica em:
  - Classe 0 não referenciada, não modificada
  - Classe 1 não referenciada, modificada
  - Classe 2 referenciada, não modificada
  - Classe 3 referenciada, modificada
- O algoritmo NRU remove aleatoriamente

# FIFO

- Simples e de baixo custo computacional
- O SO mantém uma lista de todas as páginas ordenada por ordem de utilização
  - As menos usadas no início, as mais recentemente utilizadas no final da lista
- Quando uma falta de página ocorre, a primeira da lista é substituída

# Segunda chance

- Semelhante ao FIFO com uma pequena modificação:
  - Elimina a mais antiga com R=0, caso não exista:
    - Verifica na lista a mais antiga, se R=1
      - Faz R=0 e checa a próxima da lista
      - Faz isto até encontrar uma página com R=0



# Menos recentemente usada

- Cria um *buffer* para cada página com uma determinada quantidade de bits
- A cada ciclo de clock, indica se cada página foi referenciada ou não
  - Caso tenha sido o *buffer* é incrementado
- Quando houver necessidade de troca, utiliza-se a página com menor valor no *buffer*, a menos utilizada

# *Working set*

- Cria conjuntos de páginas de um processo carregado
- Verifica qual o WS é o mais importante baseado na utilização de suas páginas
- O algoritmo consiste em trazer para memória o WS de um processo antes que ele seja executado, evitando as faltas de página

# Perguntas / Dúvidas

# Dúvidas/Perguntas

# Referências


- TANENBAUM, A.S. - Sistemas Operacionais Modernos (Capítulo 4)

# FIM