

GERÊNCIA DE MEMÓRIA

SLIDE EM CONSTRUÇÃO

Raí Emanuel T. de Freitas
raiemmanuel50.50@gmail.com

MEMÓRIA

MEMÓRIA

- O que é memória?
 - A memória consiste em um longo array de bytes a qual cada byte tem seu endereço próprio

MEMÓRIA

- O armazenamento dos programas precisa ser realizado em uma memória não volátil
 - Memórias magnéticas ou óticas (Não voláteis)
 - Memórias de silício (Voláteis)
- HD e SSD são uma boa escolha (memória secundária)
 - Grandes, baratos e não voláteis
- O processador não consegue executar diretamente as instruções que estão na memória secundária
 - O SO transfere as instruções do programa da memória secundária para a principal antes de serem executadas
 - HD/SSD -> RAM
- O SO visa diminuir os acessos à memória secundária por questões de desempenho

MEMÓRIA

- Historicamente, a memória principal sempre foi vista como um recurso escasso e caro
 - Até hoje, o gerenciamento de memória é um dos fatores mais importantes no projeto de sistemas operacionais
 - Hierarquia de memória torna a gerência de memória mais eficiente
 - Acesso a dados com mais rapidez
 - Redução da latência na busca de dados
 - Aumenta a eficiência do sistema como um todo

HIERARQUIA DE MEMÓRIA

HIERARQUIA DE MEMÓRIA

- Desde o início da computação, os programadores anseiam por uma memória infinita, rápida, barata e não volátil
 - Ainda não descoberta até hoje
- Ilusão de memória ilimitada, rápida, barata e não volátil
 - Hierarquia de memória
 - O sistema operacional abstrai essa hierarquia
 - Gerenciador de memória
 - Controla quais partes estão sendo usadas
 - Aloca o espaço para novos processos
 - Libera o espaço dos processos terminados

HIERARQUIA DE MEMÓRIA

- Preliminary Discussion of the Logical Design of an Electronic Computing Instrument, 1946

4.0. The Memory Organ

4.1. Ideally one would desire an indefinitely large memory capacity such that any particular 40 binary digit number or word would be immediately - i.e., in the order of 1 to 100 μ s - available and that words could be replaced with new words at about the same rate. It does not seem possible physically to achieve such a capacity. We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.

HIERARQUIA DE MEMÓRIA

- Princípio da localidade

- Programas acessam uma pequena porção dos endereços em um determinado instante de tempo
 - Espacial
 - Se um item é referenciado, itens próximos tendem a ser referenciados também
 - Temporal
 - Se um item é referenciado, ele tende a ser referenciado novamente em breve
- As instruções dos programas são acessadas sequencialmente
 - Qual localidade é usada?
- Acesso a um array ou componentes de um struct C++
 - Qual localidade é usada?
- Atualização da variável `i`, contadora de iteração, em um loop
 - Qual a localidade usada?

EXERCÍCIOS DE FIXAÇÃO

Na hierarquia de memórias, funciona o princípio da localidade, segundo o qual o programa acessa uma porção relativamente pequena do espaço de endereçamento em qualquer instante de tempo.

- ☐ Certo
- ☐ Errado

Questão 1

Analise as afirmativas a seguir sobre o sistema de memória de computadores e as suas características:

I. O princípio da localidade diz que os acessos à memória realizados em um programa tendem a usar uma parte relativamente pequena do seu espaço de endereçamento. II. No projeto de um computador utilizamos apenas um tipo de memória com grande capacidade de armazenamento e com uma grande velocidade de acesso. III. À medida que se desce na hierarquia de memória, o tempo de acesso aumenta e o custo da memória diminui. IV. Discos magnéticos são exemplos de memória primária.

Assinale

- ☐ A se somente as afirmativas I e II estiverem corretas.
- ☐ B se somente as afirmativas II e III estiverem corretas.
- ☐ C se somente as afirmativas III e IV estiverem corretas.
- ☐ D se somente as afirmativas II e IV estiverem corretas.
- ☐ E se somente as afirmativas I e III estiverem corretas.

Questão 2

Assinale a única alternativa que preenche corretamente as lacunas em branco.

A solução para a alocação de segmentos maiores do que o espaço disponível na memória (ou até mesmo maior que ela toda) veio com um dos conceitos mais importantes de otimização de programas e sistemas, que é o _____. Este princípio diz que os endereços de memória não têm probabilidade igual de acesso, sendo mais provável que após executar uma instrução da página x , que acesse um dado da página y , é muito mais provável que a próxima instrução também esteja na página x e também acesse dados na página y .

- ☐ A princípio da elasticidade.
- ☐ B princípio da localidade.
- ☐ C princípio da alocação.
- ☐ D princípio do endereçamento.
- ☐ E princípio dos blocos.

Questão 3

A observação de que os acessos à memória realizados em qualquer intervalo de tempo curto tendem a usar somente uma pequena fração da memória total é denominada

- ☐ A princípio da dualidade.
- ☐ B observância temporal.
- ☐ C dualidade de cache.
- ☐ D observância de acesso.
- ☐ E princípio da localidade.

Questão 4

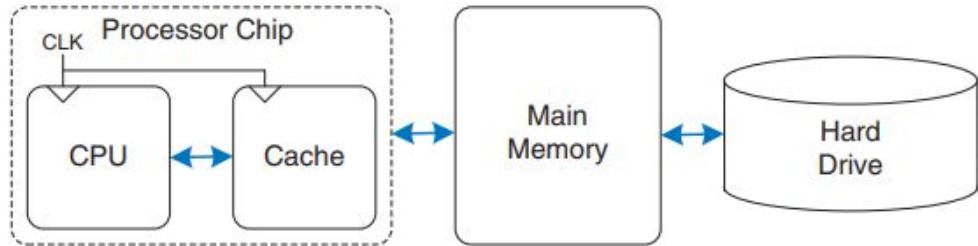
HIERARQUIA DE MEMÓRIA

- Aproveitando-se do princípio da localidade, a memória de um computador é implementada por meio da hierarquia de memória
- A hierarquia de memória consiste em múltiplos níveis com memórias de diferentes tamanhos e velocidades
- As memórias mais rápidas são mais caras por bit, enquanto que as mais lentas são mais baratas por bit
 - Registrador - R\$ 25,00/bit
 - HD - R\$ 0,25/bit
 - Valor fictício

HIERARQUIA DE MEMÓRIA

- Hoje em dia, há três (3) principais tecnologias usadas para a construção da hierarquia da memória
 - SRAM, DRAM e Disco magnético

Figure 8.3 A typical memory hierarchy



HIERARQUIA DE MEMÓRIA

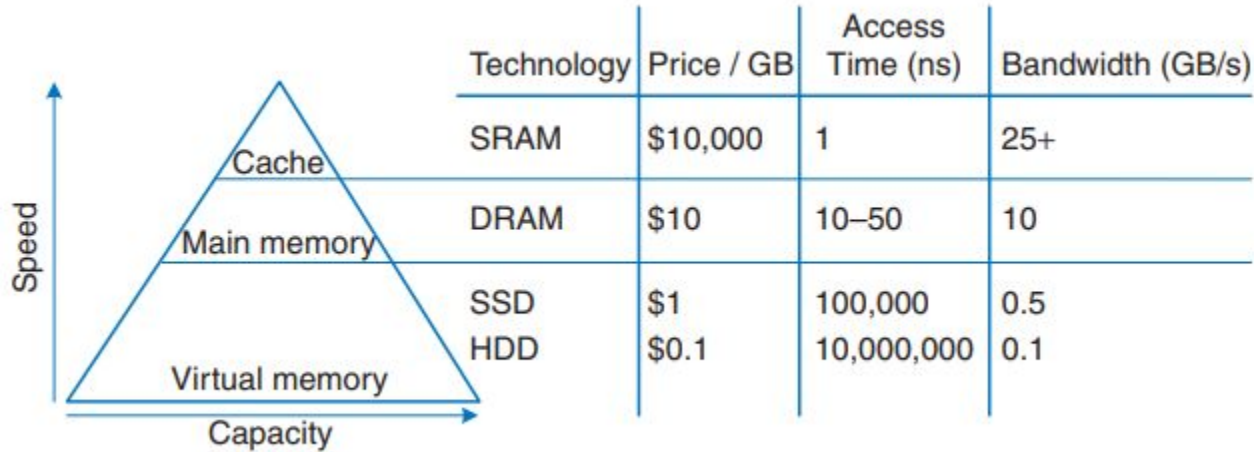


Figure 8.4 Memory hierarchy components, with typical characteristics in 2012

Em computadores digitais, a estrutura de armazenamento pode ser constituída por: Memória Cache (MC), Registradores (RE), Memória Principal (MP) e Disco Magnético (DM). Esses dispositivos podem ser organizados em uma hierarquia de acordo com a velocidade e o custo. A classificação correta dos componentes acima citados, a partir do que proporciona acesso mais veloz é:

- ☐ A MC/RE/MP/DM.
- ☐ B MC/MP/DM/RE.
- ☐ C RE/MC/MP/DM.
- ☐ D RE/MC/DM/MP.

Questão 5

Qual é o item que compõe a hierarquia de um sistema de memória, que fornece maior velocidade?

- ☐ A Memória principal.
- ☐ B Cache.
- ☐ C Disco Magnético.
- ☐ D Fita Magnética.
- ☐ E Registradores.

Questão 6

As memórias de um microcomputador são organizadas por níveis hierárquicos representados por uma pirâmide. Em relação às características das memórias do microcomputador, pode-se afirmar que

- ☐ A a memória principal apresenta baixa velocidade, maior capacidade e menor custo quando comparada com a memória secundária.
- ☐ B a memória cache possui menor tempo de acesso, elevada capacidade de armazenamento e alto custo.
- ☐ C dentro do sistema de computação, a memória secundária é caracterizada por sua elevada capacidade de armazenamento e maior custo.
- ☐ D a memória primária possui baixa velocidade, pequena capacidade de armazenamento e elevado custo dentro do sistema.
- ☐ E os registradores possuem maior velocidade de transferência dentro do sistema de computação, menor capacidade de armazenamento e maior custo.

Questão 7

A memória rápida também tem um valor financeiro maior. Para amenizar essa questão, há a hierarquia de memória, organizada em vários níveis, em que a menor é mais rápida e com maior valor por byte. Sobre hierarquia de memória, podemos afirmar que:

- A** Registradores, Cache L1, Cache L2, Cache L3, Memória RAM, Disco, sendo o registrador mais lento e o disco mais rápido.
- B** Pode ser representada da seguinte forma, sendo da mais lenta para a mais rápida: Cache L1, Cache L2, Cache L3, Memória RAM, Disco.
- C** O registrador não entra na hierarquia de memória, sendo apenas uma instrução do processador.
- D** Da mais veloz para a mais lenta: Registradores, Cache L1, Cache L2, Cache L3, Memória RAM, Disco.
- E** A hierarquia de memória pode ser representada da seguinte forma, sendo da mais lenta para a mais rápida: Cache L1, Registradores, Cache L2, Cache L3, Memória RAM, Disco.

Questão 8

GERÊNCIA DE MEMÓRIA

GERÊNCIA DE MEMÓRIA

- A gerência de memória é simples nos sistemas monoprogramados
 - Toda a memória é disponível para um único programa
- A gerência de memória é complexa nos sistemas multiprogramados
 - Otimizada para suportar o máximo de usuários e programas compartilhando eficientemente a memória principal

GERÊNCIA DE MEMÓRIA

- A gerência de memória tenta manter na RAM o máximo possível de processos simultaneamente
 - Maximiza o compartilhamento do processador e outros recursos
- Caso não haja espaço livre para um novo processo? o que fazer?
 - O SO deve permitir que novos processos sejam aceitos e processados
 - SWAP
 - Permite a ilusão de uma RAM maior que o tamanho real
 - Transferência temporária da RAM para o HD/SSD
- Caso o programa seja maior que a RAM disponível? o que fazer?
 - Particiona-o para executar suas partes

GERÊNCIA DE MEMÓRIA

- Em sistemas multiprogramados, o SO deve proteger a área de memória ocupada por cada processo
 - Além de proteger a área de memória do próprio SO
 - Caso um programa tente acessar uma área de memória indevida, o SO deve impedir tal ação
 - Segurança
 - Isolamento
 - Apesar da proteção de memória, deve existir mecanismos de compartilhamento seguro de dados para que diferentes processos possam trocar dados de forma protegida
- Por que gerenciar a memória?
 - Isolamento de processos
 - Segurança
 - Uso eficiente da memória

AUSÊNCIA DE ABSTRAÇÃO DE MEMÓRIA

- A abstração de memória mais simples é não ter abstração de memória
 - Programas usavam os endereços físicos da memória RAM
 - Primeiros computadores de grande porte (antes de 60)
 - Primeiros computadores pessoais (antes de 80)
 - `Mov $r1, 1000`
 - Conteúdo do endereço 1000 da RAM física era movido para o registrador `$r1`
 - O modelo de memória apresentada ao programador era apenas a física
 - 0 até algum máximo
 - Cada célula endereçável possui 8 bits
 - Não era viável ter dois programas ao mesmo tempo na memória
 - Um processo sobrescreveria os dados armazenados por outro processo naquela posição de memória

AUSÊNCIA DE ABSTRAÇÃO DE MEMÓRIA

- É possível executar vários programas ao mesmo tempo sem uma abstração de memória?
 - O SO salva o conteúdo inteiro da memória em um arquivo do disco
 - Traz e executa o programa seguinte
 - Apenas um programa por vez estará na memória
 - Técnica de Swapping
- E com abstração de memória? Como os sistemas operacionais abstraem a memória e as gerenciam?

ABSTRAÇÃO DE
MEMÓRIA: ESPAÇO
DE ENDEREÇAMENTO

ESPAÇO DE ENDEREÇAMENTO

- Expor a memória física para os programas possui suas desvantagens
 - 1 - Processos do usuário podem endereçar cada byte da memória
 - Podem comprometer a segurança ao “bagunçar” os dados (intencionalmente ou não)
 - Necessário mecanismos de segurança e proteção
 - 2 - Díficil revezar múltiplos programas ao mesmo tempo
 - Desempenho comprometido
- Daí surge a ideia de um espaço de endereçamento
 - Cria uma memória abstrata para os programas usarem
- Espaço de endereçamento é um conjunto de endereços que um processo pode usar para endereçar a memória
 - Cada processo terá o seu (diferentes entre si)

ESPAÇO DE ENDEREÇAMENTO

- Números de telefones e placas de transportes usam espaço de endereçamento
 - 9.xxxx-xxxx e yyyxxxx
 - x são números e y são letras
 - 9.0000-0000 a 9.9999-9999
 - AAA0000 a ZZZ9999
- O endereço 28 em um programa deve referenciar uma posição de memória diferente do endereço 28 em outro programa
 - Registrador-base e registrador-limite

ESPAÇO DE ENDEREÇAMENTO

- Registrador-base e registrador-limite
 - Usa realocação dinâmica
 - Mapeia o espaço de endereçamento de cada processo em uma parte diferente de memória
 - Cada processo terá os dois registradores
 - Base e limite
 - Delimita a área de atuação do processo
 - Os processos são carregados consecutivamente na memória ->
 - São atribuídos os valores dos registradores base e limite ->2x
 - A - base = 0 e limite = 16.384
 - B - base = 16.384 e limite = 16.384
 - C - base = 32.768 e limite = 16.384

ESPAÇO DE ENDEREÇAMENTO

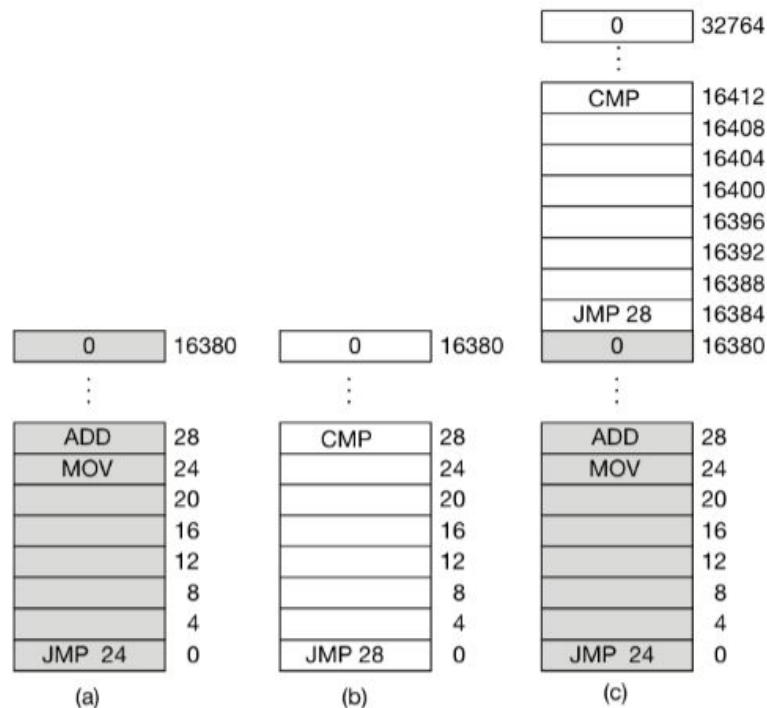


Figura 3.2 Exemplo do problema de realocação. (a) Um programa de 16 KB. (b) Outro programa de 16 KB. (c) Os dois programas carregados consecutivamente na memória.

ESPAÇO DE ENDEREÇAMENTO

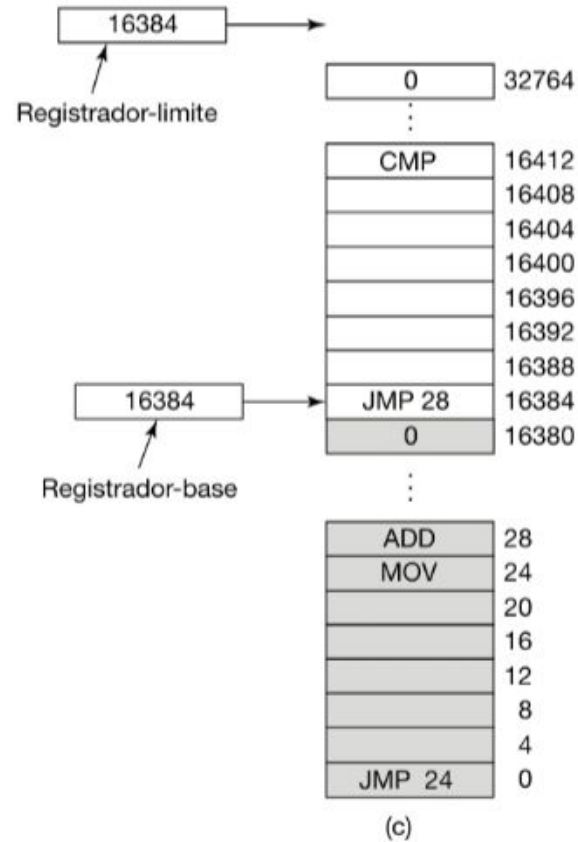


Figura 3.3 Registradores-base e registradores-limite podem ser usados para dar a cada processo um espaço de endereçamento em separado.

ESPAÇO DE ENDEREÇAMENTO

- O que acontece quando um processo referencia a memória?
 - Soma-se o endereço base + memória referenciada antes de enviar pelo barramento da RAM
 - Verifica-se se a soma não ultrapassa o limite
 - A desvantagem dessa abordagem é uma soma e uma comparação a mais a cada referência à memória

ALOCAÇÃO DE MEMÓRIA

ALOCÇÃO DE MEMÓRIA

- Existem três tipos principais de técnicas de alocação de memória
 - 1 - Alocação contígua simples
 - 2 - Overlay
 - 3 - Alocação particionada
 - Estática
 - Absoluta
 - Relocável
 - Dinâmica

1 - ALOCAÇÃO CONTÍGUA SIMPLES

1 - ALOCAÇÃO CONTÍGUA SIMPLES

- Método implementado nos primeiros sistemas operacionais
 - Ainda presente em alguns sistemas monoprogramáveis
- A memória principal é subdividida em duas partes: ->
 - 1 - Sistema operacional
 - 2 - Programa
- O usuário tem controle sobre toda a memória
 - Pode acessar qualquer posição
 - Inclusive a área do SO
 - Intencional ou não
 - Registradores são usados para proteção de memória ->
 - Sempre que um programa faz referência a uma posição de memória, o SO verifica se o endereço está dentro do limite seguro/permitido
 - Caso não esteja, o programa é fechado e gerada uma mensagem de erro

1 - ALOCAÇÃO CONTÍGUA SIMPLES



Fig. 9.1 Alocação contígua simples.

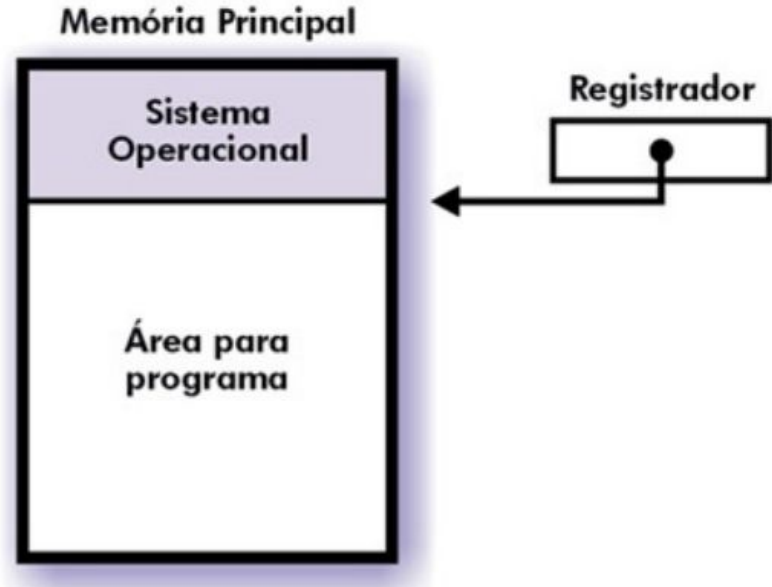


Fig. 9.2 Proteção na alocação contígua simples.

1 - ALOCAÇÃO CONTÍGUA SIMPLES

- Apesar de simples, esse método não permite a eficiência pois apenas um usuário/programa pode fazer uso da memória por vez
 - Uma grande porção da memória pode estar ociosa

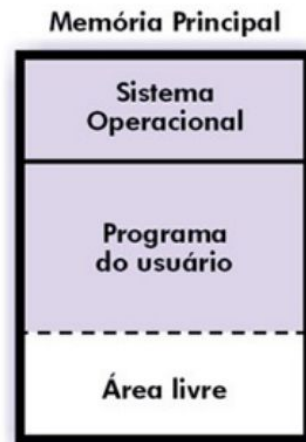


Fig. 9.3 Subutilização da memória principal.

2 - OVERLAY

2 - OVERLAY

- Na alocação contígua simples, os programas são limitados ao tamanho da área de memória disponível
 - E se um programa for maior que a memória livre?
- Uma solução é dividir o programa em módulos
 - Cada módulo pode ser executado independentemente utilizando a mesma área de memória
 - Overlay
- Considere um programa com 3 módulos
 - Principal, cadastramento e impressão
 - Cadastramento e impressão independentes
 - Não precisam coexistir ao mesmo tempo na memória
 - O módulo principal é comum aos outros dois
 - Precisa estar sempre na memória
 -

Em relação à gerencia de memória, marque a alternativa **CORRETA**:

- ☐ A a técnica de *overlay* divide o programa em módulos para ser carregado em memória.
- ☐ B a alocação continua simples é utilizada em sistemas multiprogramáveis.
- ☐ C a alocação particionada estática é utilizada em sistemas monoprogramados.
- ☐ D a técnica de memória virtual combina as memórias cachê e memória principal.
- ☐ E a técnica de swapping é utilizada para resolver o problema de velocidade na memória secundária.

No gerenciamento de memória, a técnica de *overlay* permite

- ☐ A que se sinalize para o sistema operacional que a memória RAM está cheia.
- ☐ B que se organizem as tarefas, criando uma fila de programas a serem executados pelo processador.
- ☐ C que se aloque, de forma contígua e simples, o espaço na memória principal.
- ☐ D que se gerencie o espaço dos registradores de memória.
- ☐ E que se divida um programa em módulos, de tal maneira que seja possível a execução independente de cada módulo, utilizando uma mesma área da memória principal.

- A memória é insuficiente para armazenar o programa
 - Memória - 8kb
 - Programa - 11kb
- Considere que o módulo principal referenciou um dos dois módulos
 - 1 - Caso o módulo referenciado já esteja na overlay, então nada será feito
 - 2 - Caso o módulo referenciado não esteja na overlay, então será carregado da memória secundária para a principal

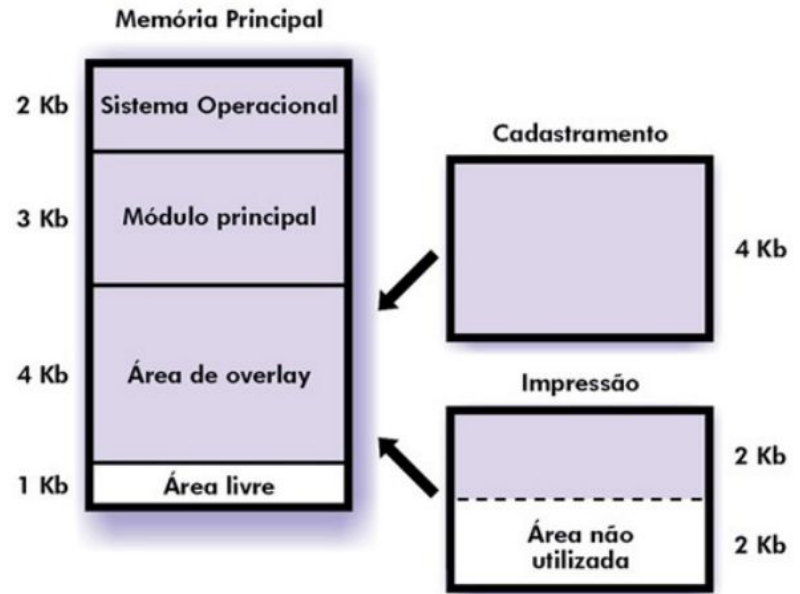


Fig. 9.4 Técnica de overlay.

2 - OVERLAY

- O tamanho da área de overlay é definida pelo tamanho do maior módulo
 - Cadastramento - 4kb
 - Área overlay - 4kb
- Quando um overlay termina a execução, abre espaço para outro ser executado na mesma posição de memória
- Necessita de muito cuidado
 - Prejudica manutenção e desempenho
 - A divisão do programa em overlays não é simples e deve ser realizada pelo programador - Manutenção
 - Transferências excessivas entre memórias - Desempenho

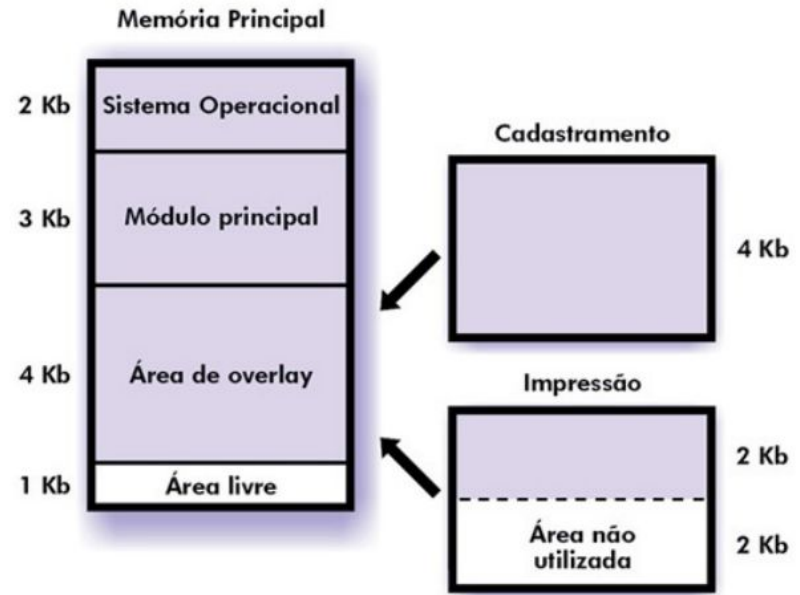


Fig. 9.4 Técnica de overlay.

2 - OVERLAY

3 - ALOCAÇÃO PARTICIONADA

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA

- Nos primórdios da multiprogramação, a memória era dividida em pedaços de blocos de tamanho fixo
 - Partição ->
- O tamanho da partição era definido na inicialização do sistema
 - Definido em função do tamanho dos programas que iriam ser executados
- E se o tamanho da partição precisasse mudar o tamanho para acomodar os novos programas?
 - Sistema desativado e reinicializado com a nova configuração
 - Alocação particionada estática

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA

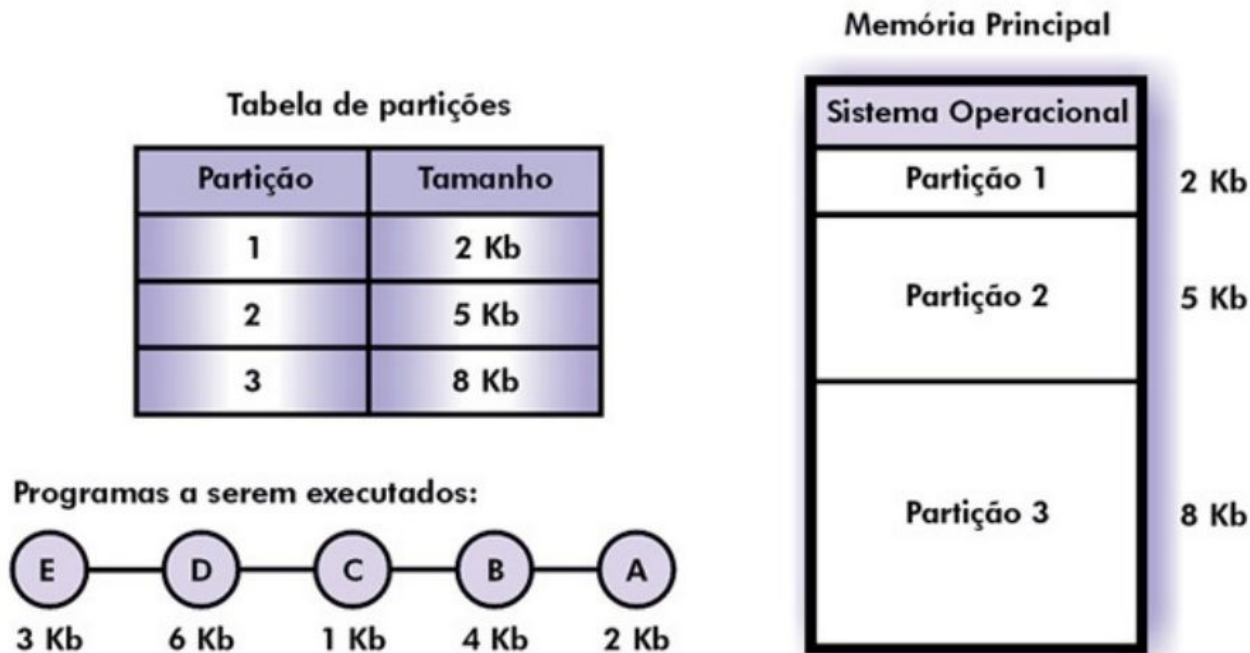


Fig. 9.5 Alocação particionada estática.

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA ABSOLUTA

- Inicialmente, os programas só podiam ser alocados em partes específicas da memória, mesmo que houvesse outros espaços livres
 - Compiladores e montadores apenas geravam código absoluto
 - Todas as referências a posições de memória são posições físicas da RAM
 - O programa necessitava ser carregado em uma posição específica definida no código
 - Imagine que A e B estão sendo executados nas partições 1 e 2, respectivamente. A partição 3 está livre
 - C e E não poderia ser executados imediatamente porque dependem da alocação absoluta na partição 1 e 2, respectivamente
 - Alocação particionada estática absoluta ->

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA ABSOLUTA

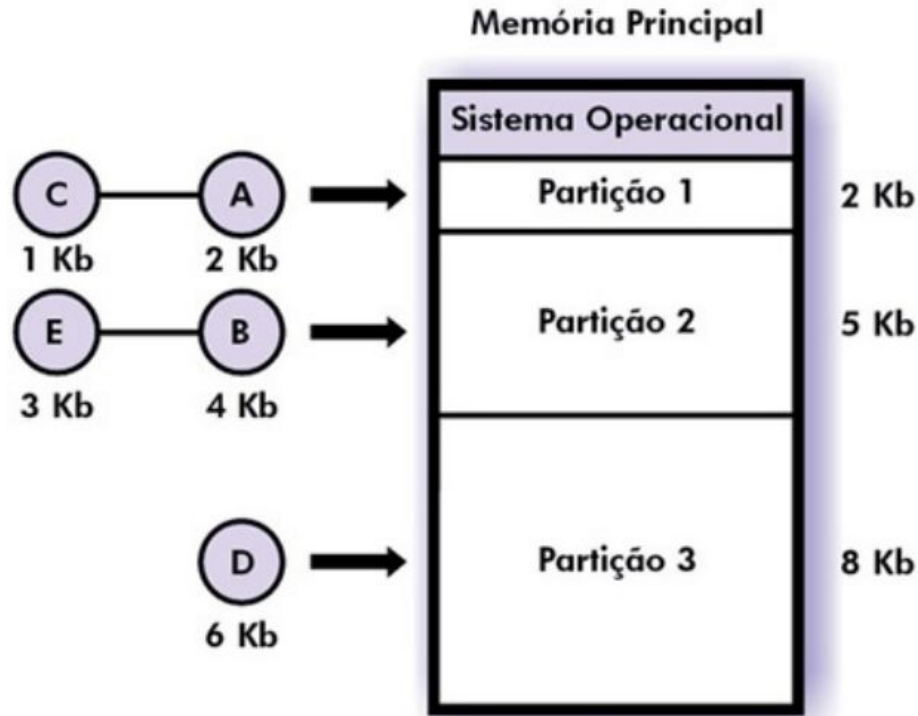


Fig. 9.6 Alocação particionada estática absoluta.

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA RELOCÁVEL

- Compiladores, montadores, linkers e loader evoluíram
 - Deixaram de ser absolutos e passaram a ser relocáveis
- No código relocável, todas as referências a endereços são em relação ao início do programa em vez dos endereços de memória física
 - Os programas passaram a ser executados em qualquer partição
 - O loader calcula todos os endereços a partir da posição que o programa foi alocado
- Caso A ou B terminem, E pode ser alocado em qualquer uma das partições ->
 - Alocação particionada estática relocável

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA RELOCÁVEL

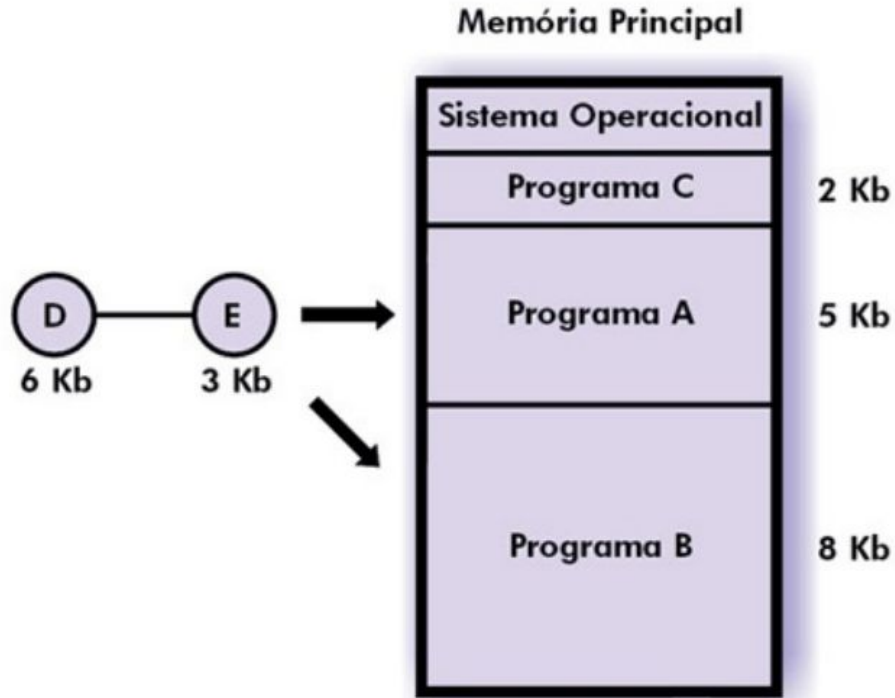


Fig. 9.7 Alocação particionada estática relocável.

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA RELOCÁVEL

- É necessário controlar quais partições estão alocadas
 - Tabela contém
 - Endereço base
 - Tamanho
 - Se está em uso
- A tabela é percorrida para encontrar uma partição que acomode o novo processo

Partição	Tamanho	Livre
1	2 Kb	Não
2	5 Kb	Sim
3	8 Kb	Não

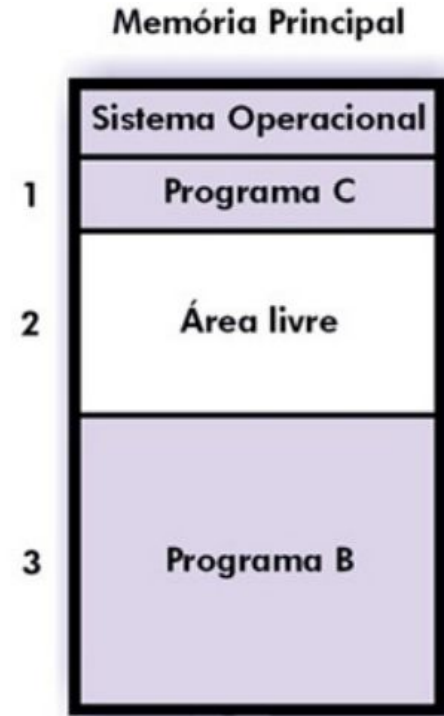


Fig. 9.8 Tabela de alocação de partições.

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA RELOCÁVEL

- A **proteção** nesse modo é **implementada** a partir de **dois (2) registradores**
 - **Definem o limite inferior e superior da partição** que o **processo** está **executando**
- E se o processo tentar acessar uma posição de memória fora dos limites?
 - Processo interrompido (kill)
 - Mensagem de violação de acesso é gerada

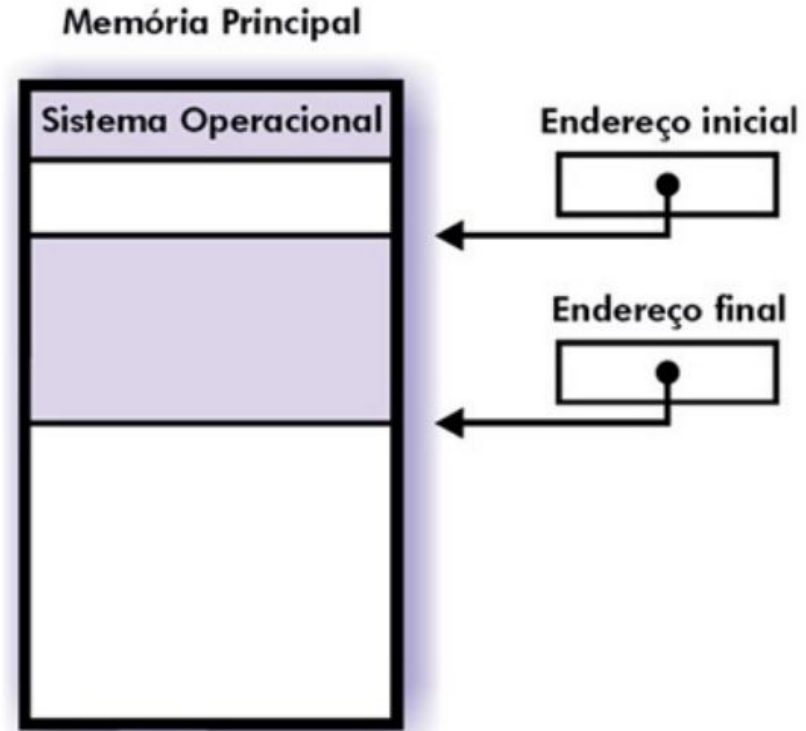


Fig. 9.9 Proteção na alocação particionada.

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA ABSOLUTA E RELOCÁVEL

- Há um **problema inerente** ao modo de **alocação particionada estática absoluta e relocável**
- Os programas, geralmente, não preenchem completamente a **partição a qual foi concedida**
 - C, A e E não ocupam integralmente o espaço da partição a qual estão alocados
 - 1kb, 3kb e 5kb ficaram livres e subutilizados
- Esse problema decorrente da alocação de partições fixas é chamado de **fragmentação interna**

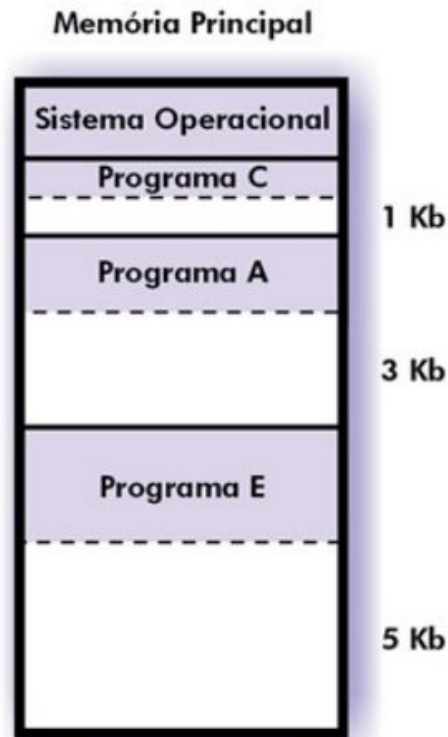
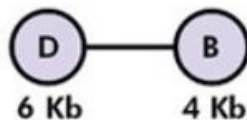


Fig. 9.10 Fragmentação interna.

3 - ALOCAÇÃO PARTICIONADA ESTÁTICA RELOCÁVEL

- Qual sistema operacional implementou essa alocação particionada?
 - OS/MFT (Multiprogramming with a Fixed Number of Tasks) da IBM

3 - ALOCAÇÃO PARTICIONADA DINÂMICA

- Como resolver a fragmentação interna?
 - Ajustando o tamanho das partições para se adequarem dinamicamente ao tamanho dos programas
 - Elimina a fragmentação interna
 - Maximiza o número de processos compartilhando a memória
- A Alocação particionada dinâmica elimina o uso de partições de tamanho fixo
- Cada programa utiliza apenas a memória necessária ->
 - Torna-se a sua partição
- Surge o problema da fragmentação externa
 - Os processos vão terminando e deixando espaços cada vez menores disponíveis entre as partições
 - Na teoria, processos podem ser alocados aos espaços livres (fragmentados)
 - Na prática, não permite o ingresso de novos processos

3 - ALOCAÇÃO PARTICIONADA DINÂMICA

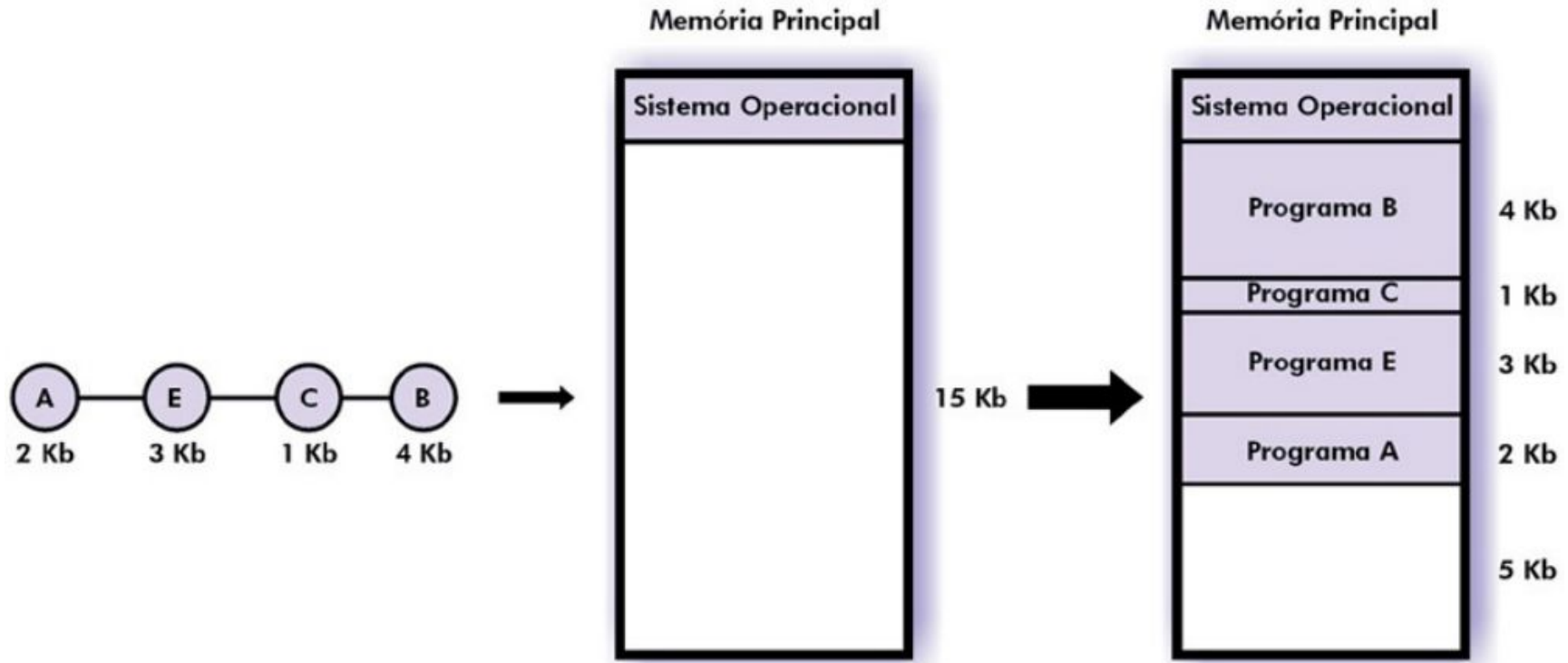


Fig. 9.11 Alocação particionada dinâmica.

3 - ALOCAÇÃO PARTICIONADA DINÂMICA

- Problema da fragmentação externa
 - Processos podem não ser alocados mesmo que exista espaço livre
 - 12kb livres
 - D necessita de 6kb
 - Não é possível a alocação porque o espaço não é contíguo

D
6 Kb

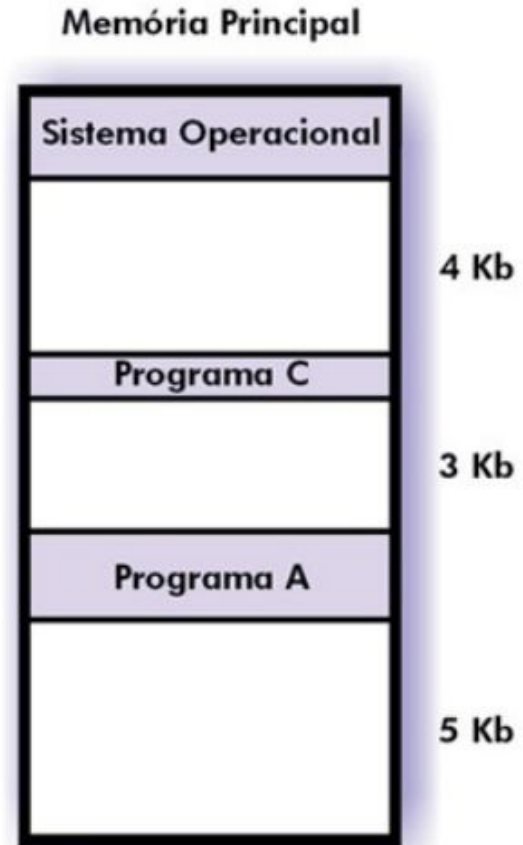


Fig. 9.12 Fragmentação externa.

3 - ALOCAÇÃO PARTICIONADA DINÂMICA

- Existem duas (2) soluções para o problema
 - 1 - Conforme os processos terminam, os espaços livres adjacentes são agrupados para formarem áreas livres maiores ->
 - 2 - Realiza a relocação de todas as partições ocupadas, eliminando todos os espaços entre elas e criando uma única área livre contígua ->2x
 - O sistema precisa ter a capacidade de mover os programas na memória principal
 - Relocação dinâmica
 - Alocação particionada dinâmica com relocação?
 - Ajuda a reduzir a fragmentação externa
 - Complexidade e consumo de recursos do sistema pode torná-lo inviável
 - OS/MVT (Multiprogramming with a Variable Number of Tasks) da IBM

3 - ALOCAÇÃO PARTICIONADA DINÂMICA

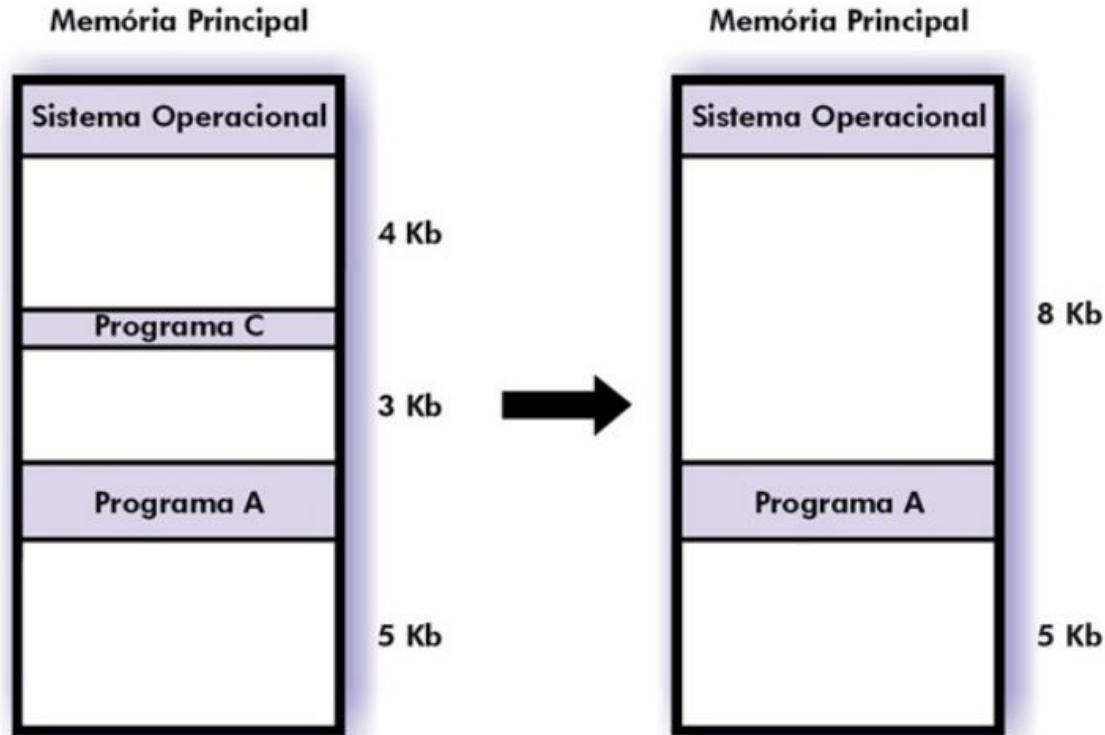


Fig. 9.13 Solução para a fragmentação externa (a).

3 - ALOCAÇÃO PARTICIONADA DINÂMICA

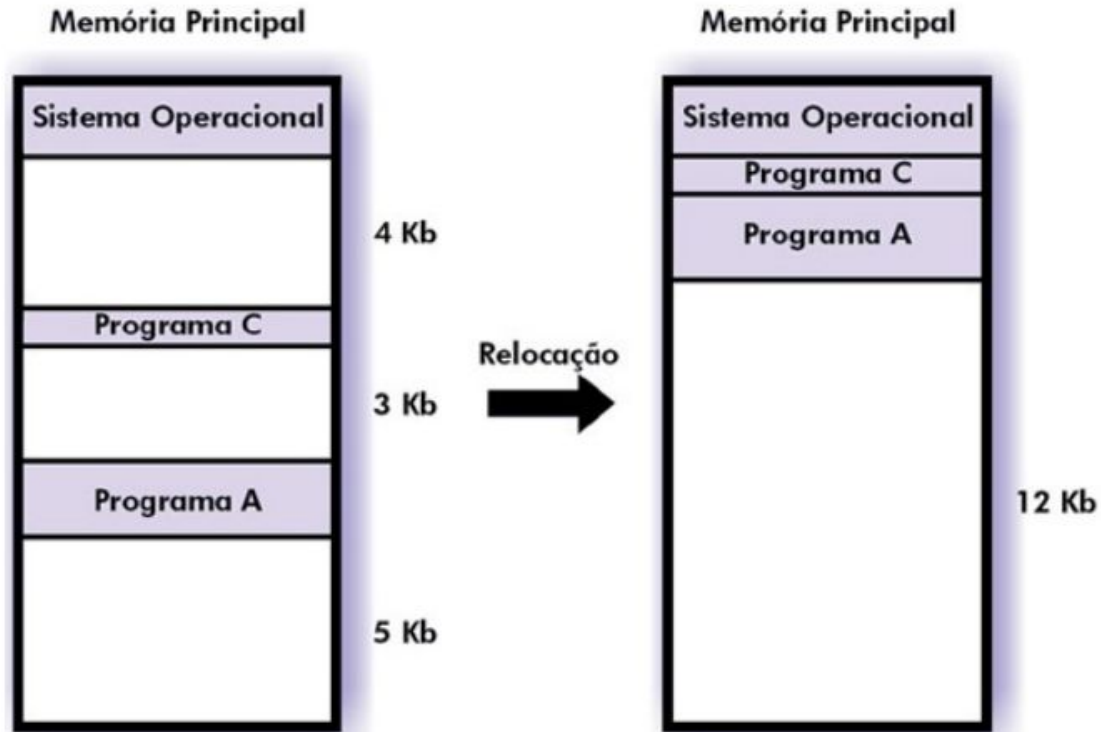


Fig. 9.14 Solução para a fragmentação externa (b).

3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO

- Os SO's usam algumas estratégias para escolher em qual espaço livre irá alocar determinado processo
 - Best-fit, First-fit, Worst-fit, Next-fit, Quick-fit
 - Objetivam diminuir o problema da fragmentação externa
- A estratégia escolhida depende de alguns fatores
 - Tamanho dos programas processados no ambiente
 - Frequência de solicitação de memória
 - Quantidade de relocações
- Independentemente da estratégia usada, o sistema tem uma tabela com as áreas livres ->
 - Endereço
 - Tamanho

3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO

Áreas livres	Tamanho
1	4 Kb
2	5 Kb
3	3 Kb

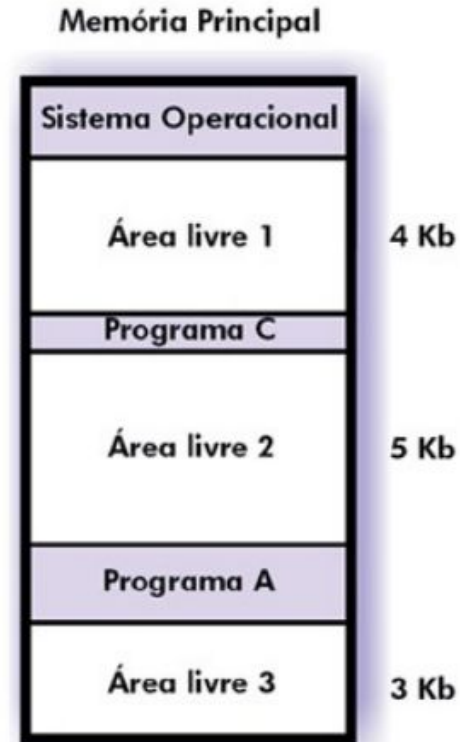
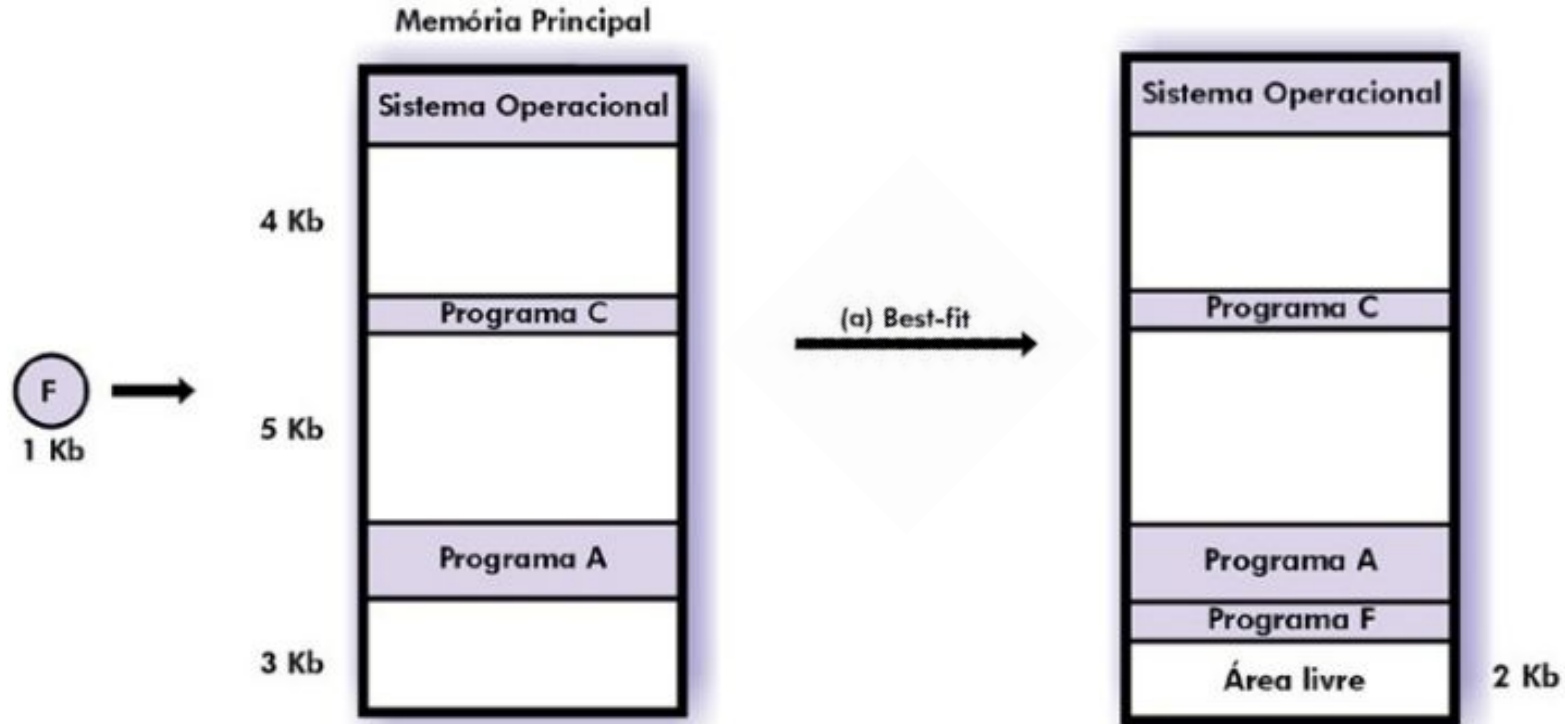


Fig. 9.15 Lista de áreas livres.

3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO - BEST-FIT

- A melhor partição é escolhida ->
 - É escolhida a área livre que resta o menor espaço subutilizado após alocação do processo
- A lista de áreas livres é ordenada crescentemente pelo tamanho para melhorar o tempo de busca por uma área melhor ajustada
 - (a3, 3kb), (a1, 4kb) e (a2, 5kb)
- Culmina no aumento da fragmentação externa
 - Como é escolhida a partição que sobra o menor espaço livre, a tendência é que restem na memória pequenas áreas não contíguas
 - Pequenas o suficiente para não conseguirem acomodar novos processos
 - Necessária a relocação

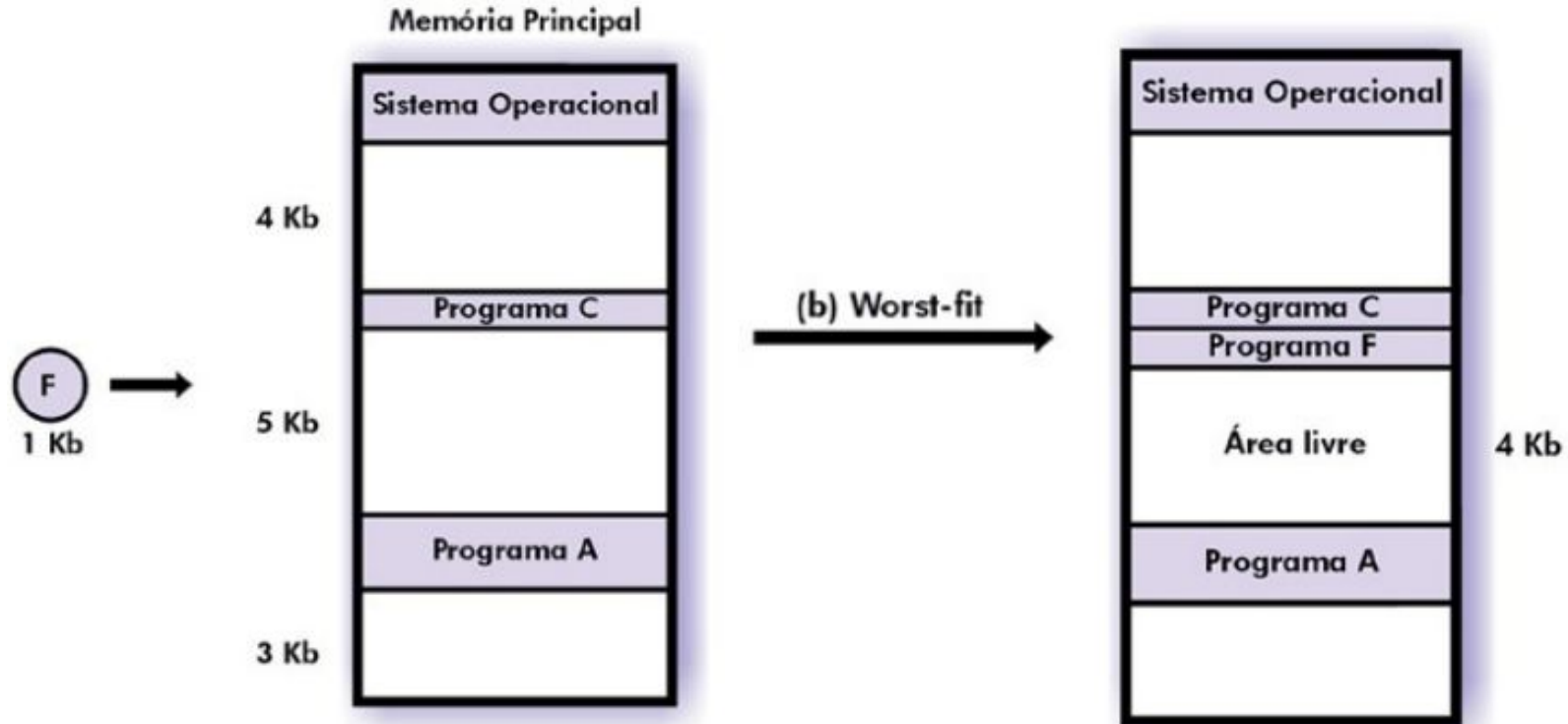
3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO - BEST-FIT



3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO - WORST-FIT

- A pior partição é escolhida ->
 - É escolhida a área livre que resta o maior espaço subutilizado após alocação do processo
- A lista de áreas livres é ordenada decrescentemente pelo tamanho para melhorar o tempo de busca por uma área pior ajustada
 - (a2, 5kb), (a1, 4kb) e (a3, 3kb)
- Culmina na diminuição da fragmentação externa
 - Como é escolhida a partição que sobra o maior espaço livre, a tendência é que restem na memória grandes áreas não contíguas
 - Grande o suficiente para conseguirem acomodar novos processos
 - Tende a não necessitar de relocação

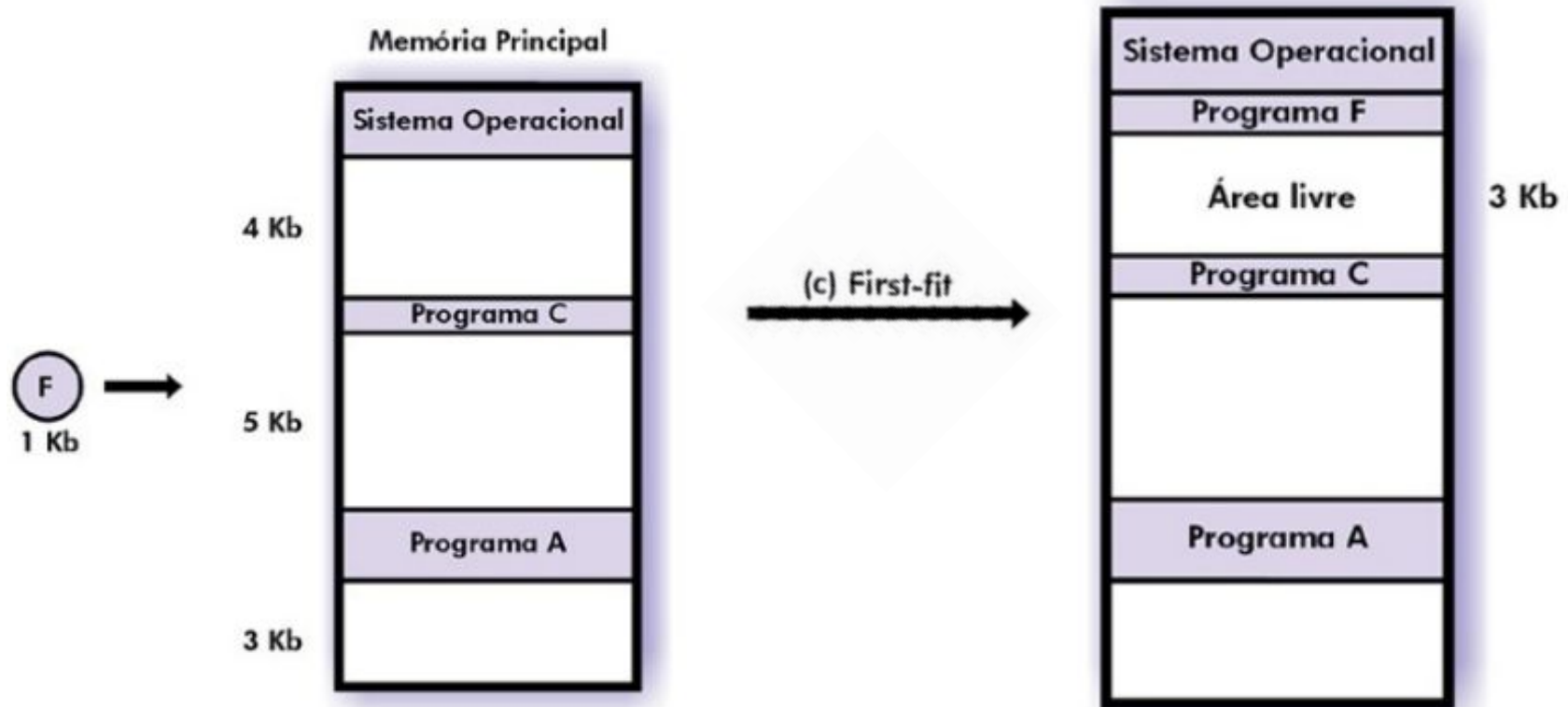
3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO - WORST-FIT



3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO - FIRST-FIT

- A primeira partição é escolhida ->
 - É escolhida a primeira área livre que consiga acomodar o processo
- A lista de áreas livres é ordenada crescentemente pelo endereço para melhorar o tempo de busca pela primeira área
 - (a1, 4kb), (a2, 5kb), e (a3, 3kb)
- Incerto quanto aumento ou diminuição da fragmentação externa
- Como tende a alocar nos endereços mais baixos (primeiros), os maiores endereços oferecem grandes áreas livres contíguas
- Estratégia mais rápida entre as três
 - Retorna a primeira partição viável encontrada

3 - ESTRATÉGIAS DE ALOCAÇÃO DE PARTIÇÃO - FIRST-FIT

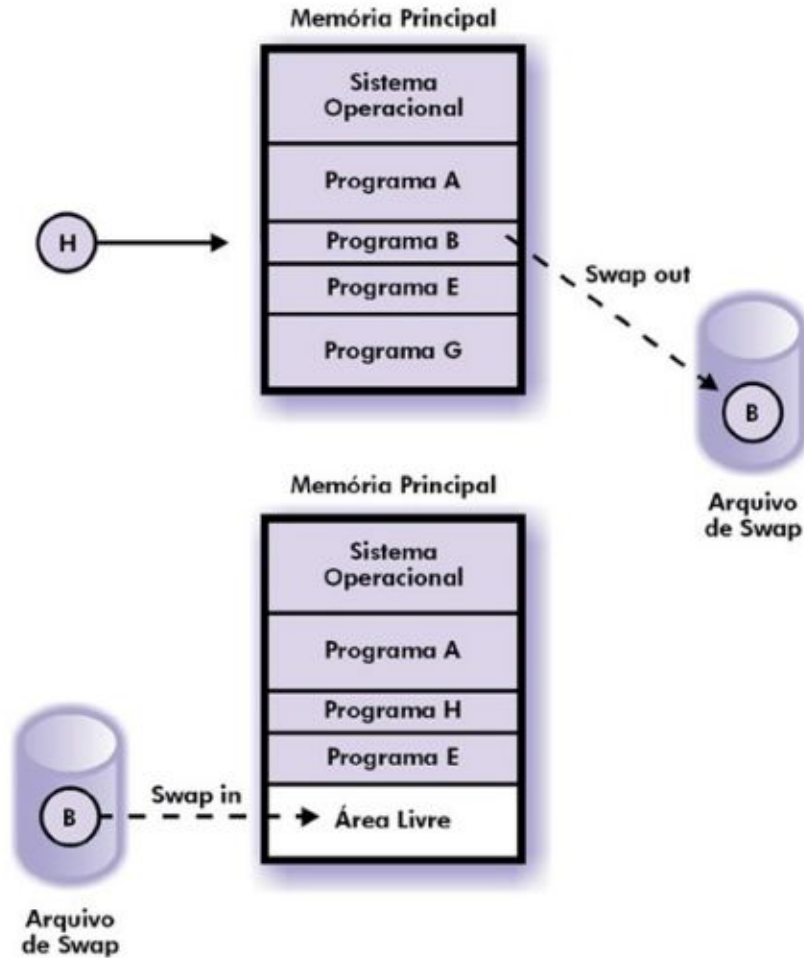


SWAPPING

SWAPPING

- Até aqui, o processo permanece na memória em todo o ciclo de vida
 - Espera por I/O mantém processo ocioso
- Algum processo pode ainda não ser executado por falta de memória RAM física. Não há espaço livre suficientemente
 - Swapping tenta contornar esse problema
 - Lembram da função 1 da gerência de memória?
- O swapping é uma técnica aplicada à gerência de memória
 - >
 - O SO escolhe um processo residente na RAM e o transfere para a memória secundária (swap out)
 - Futuramente, esse processo pode ser transferido de volta da memória secundária para a primária (swap in) e continuar normalmente

SWAPPING



SWAPPING

- A escolha do processo a sair da principal deve priorizar aquele com menor chance de ser escalonado
 - Evita dar swapping em um processo que logo deverá voltar para a RAM
 - Logo terá a atenção do processador
- Geralmente são escolhidos os processos no estado de espera
 - Menos comum um processo no estado de pronto sofrer swap out, apesar de ser possível
 - Chamados de não residente (outswapped)

GERENCIAMENTO DE MEMÓRIA LIVRE

GERENCIAMENTO DE MEMÓRIA LIVRE

- Memória alocada dinamicamente necessita de gerenciamento dos espaços livres
 - Duas formas principais de rastrear o uso da memória
 - Mapa de bits
 - Lista encadeada
 - Rastrear o uso dos recursos não é exclusivo do gerenciamento de memória

GERENCIAMENTO DE MEMÓRIA LIVRE COM MAPA DE BITS

- Cada unidade de alocação é associada a um bit ->
 - 0 - Se unidade livre
 - 1 - Se unidade ocupada
- Qual o tamanho da unidade de alocação ideal?
 - Menor unidade, maior mapa de bits
 - Maior unidade, menor mapa de bits
 - Desperdício, fragmentação
 - Considere a unidade de 4 bytes (32 bits)
 - 32 bits na ram exigirão 1 bit no mapa de bits
 - Uma memória de $32n$ bits usará um mapa de n bits
 - O mapa de bits ocupará $1/32$ da memória
- Mapa de bits é uma maneira simples de controlar as palavras na memória em uma quantidade fixa dela
 - Seu tamanho depende apenas do tamanho da memória disponível (16GB) e do tamanho da unidade de alocação

GERENCIAMENTO DE MEMÓRIA LIVRE COM MAPA DE BITS

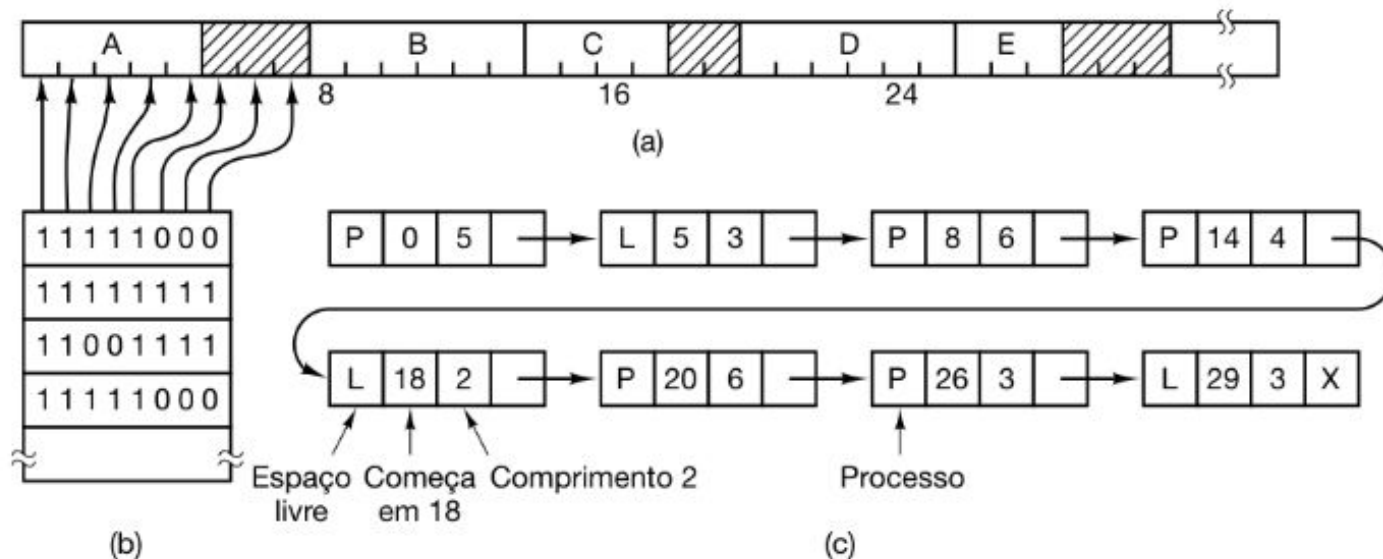


Figura 3.6 (a) Uma parte da memória com cinco processos e três espaços. As marcas indicam as unidades de alocação de memória. As regiões sombreadas (0 no mapa de *bits*) estão livres. (b) Mapa de *bits* correspondente. (c) A mesma informação como uma lista.

GERENCIAMENTO DE MEMÓRIA LIVRE COM MAPA DE BITS

- Um novo processo de tamanho k unidades de alocação deve ser alocado na memória. O que fazer?
 - O mapa de bits deve ser percorrido em busca de, pelo menos, k bits consecutivos com valor 0
 - Busca custosa em termos de tempo

GERENCIAMENTO DE MEMÓRIA LIVRE COM LISTA ENCADEADA

- Uso de uma lista encadeada com os segmentos de memória livres e os alocados ->
 - Cada segmento possui um processo ou um espaço livre entre dois processos
 - Cada entrada da lista especifica se é um espaço livre (L) ou alocado (P), o endereço que inicia o segmento, o comprimento e um ponteiro para o item seguinte
 - A lista de segmentos é ordenada pelos endereços
 - Tem como vantagem a atualização simples da lista quando um processo é terminado ou transferido ->2x

GERENCIAMENTO DE MEMÓRIA LIVRE COM LISTA ENCADEADA

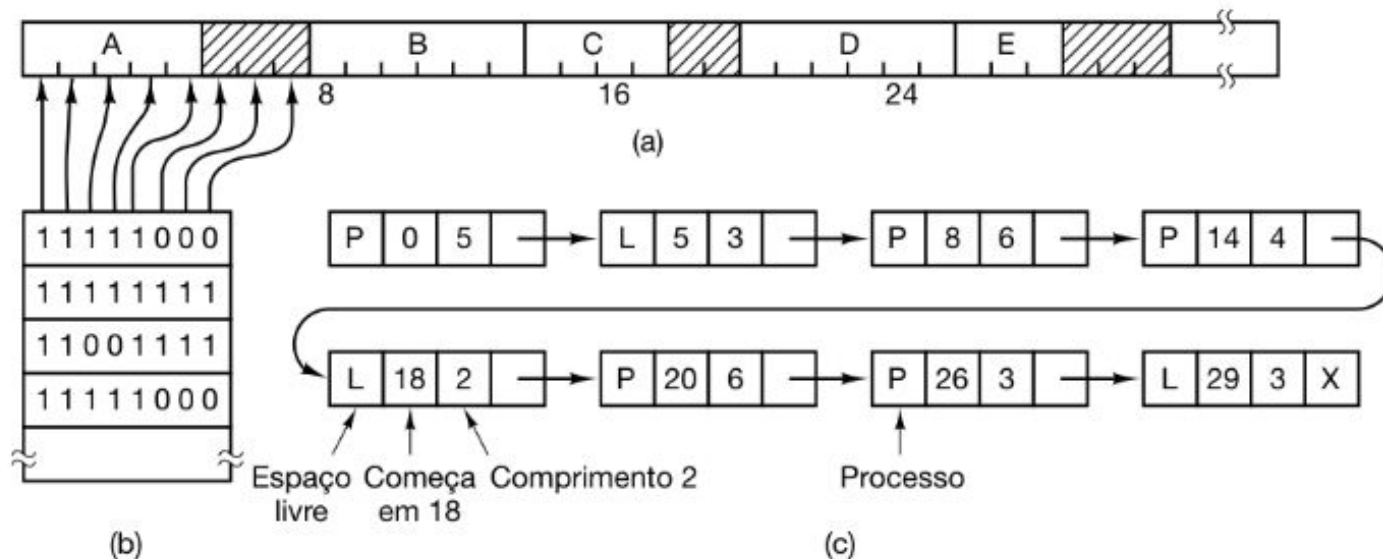


Figura 3.6 (a) Uma parte da memória com cinco processos e três espaços. As marcas indicam as unidades de alocação de memória. As regiões sombreadas (0 no mapa de *bits*) estão livres. (b) Mapa de *bits* correspondente. (c) A mesma informação como uma lista.

GERENCIAMENTO DE MEMÓRIA LIVRE COM LISTA ENCADEADA

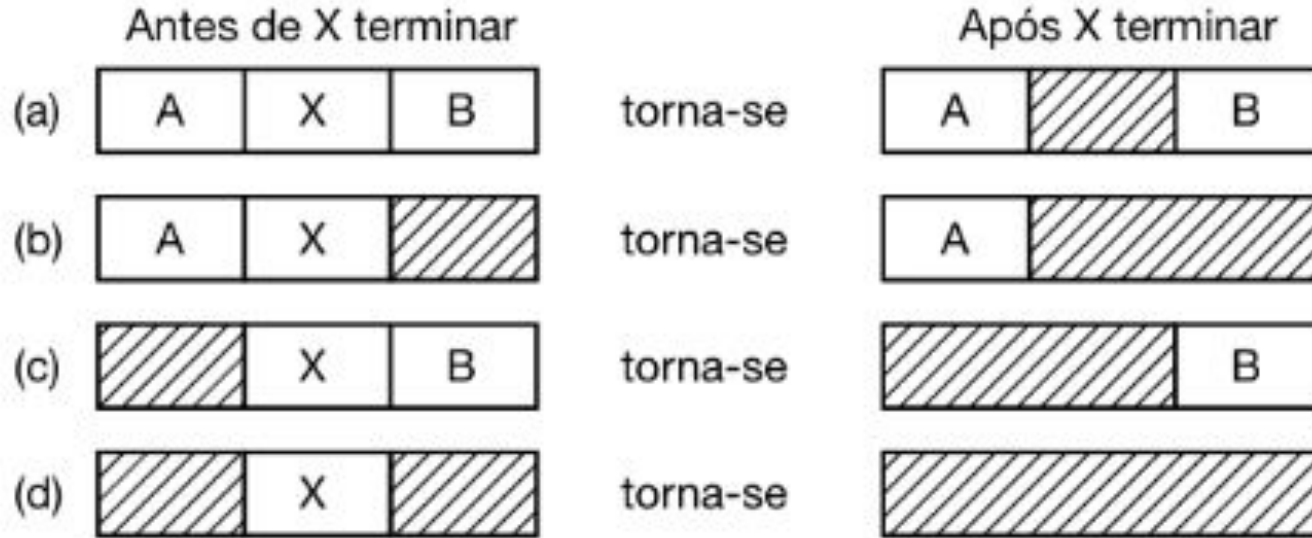


Figura 3.7 Quatro combinações de vizinhos para o processo que termina, X.

MEMÓRIA VIRTUAL

MEMÓRIA VIRTUAL

- Vimos diversas técnicas para apoiar as funções da gerência de memória e reduzir o problema da fragmentação
 - Maximizar o número de processos na memória
 - Acomodar programas maiores que a RAM física disponível
 - Proteger as áreas de memória
 - Reduzir a fragmentação interna e externa
- Técnicas ineficientes para os sistemas operacionais modernos
 - Alocação contígua simples
 - Overlay
 - Alocação particionada
- Surge a memória virtual
 - Técnica sofisticada e moderna para gerência de memória
 - Combina a memória primária e secundária para dar a ilusão de uma RAM física maior que a real

MEMÓRIA VIRTUAL

- A ideia é desassociar os endereços usados pelos programas aos endereços físicos reais da RAM
 - Programas e estruturas de dados deixam de estar limitados à memória primária
 - Aumenta a quantidade de processos residentes na memória virtual porque apenas parte deles podem estar na principal e a outra parte na secundária
 - Minimiza a fragmentação da memória
- A primeira implementação aconteceu na década de 60
 - Sistema Atlas da Universidade de Manchester
 - IBM implementa o conceito no System/370
 - Década de 70
 - Quase todos os sistemas atuais implementam memória virtual, exceto supercomputadores
 - Por quê?

MEMÓRIA VIRTUAL

- Algumas funções da gerência de memória virtual podem ser feitas em hardware para melhorar o desempenho
 - Chip MMU
 - O SO deve levar em consideração o modo de endereçamento do processador
- Três técnicas permitem a criação de memória virtual
 - Paginação
 - Segmentação
 - Paginação com segmentação

ESPAÇO DE ENDEREÇAMENTO VIRTUAL

- O conceito de memória virtual assemelha-se a um vetor das linguagens de alto nível
 - Acessar o `a[0]` não há preocupação sobre a posição de memória real daquele dado
 - Totalmente transparente ao programador

Endereço Físico

500		VET [1]
501		VET [2]
502		VET [3]
503		VET [4]
504		VET [5]
.	.	.
.	.	.
.	.	.
599		VET [100]

Fig. 10.1 Vetor de 100 posições.

ESPAÇO DE ENDEREÇAMENTO VIRTUAL

- Um programa no ambiente de memória virtual utiliza o espaço de endereçamento virtual
 - Não faz referência a espaços físicos de memória
 - O que acontece quando uma instrução da arquitetura vai executar endereço virtual?
 - Necessária a conversão do endereço virtual em endereço real
 - O processador apenas manipula posições de memória física
- A tradução de endereço virtual em real é chamada de mapeamento

ESPAÇO DE ENDEREÇAMENTO VIRTUAL

- O espaço de endereçamento virtual representa o conjunto de endereços virtuais que um processo pode endereçar
 - Processos têm espaço de endereçamento virtual
- O espaço de endereçamento real representa o conjunto de endereços reais que um processador pode endereçar
 - Processadores têm espaço de endereçamento real

ESPAÇO DE ENDEREÇAMENTO VIRTUAL

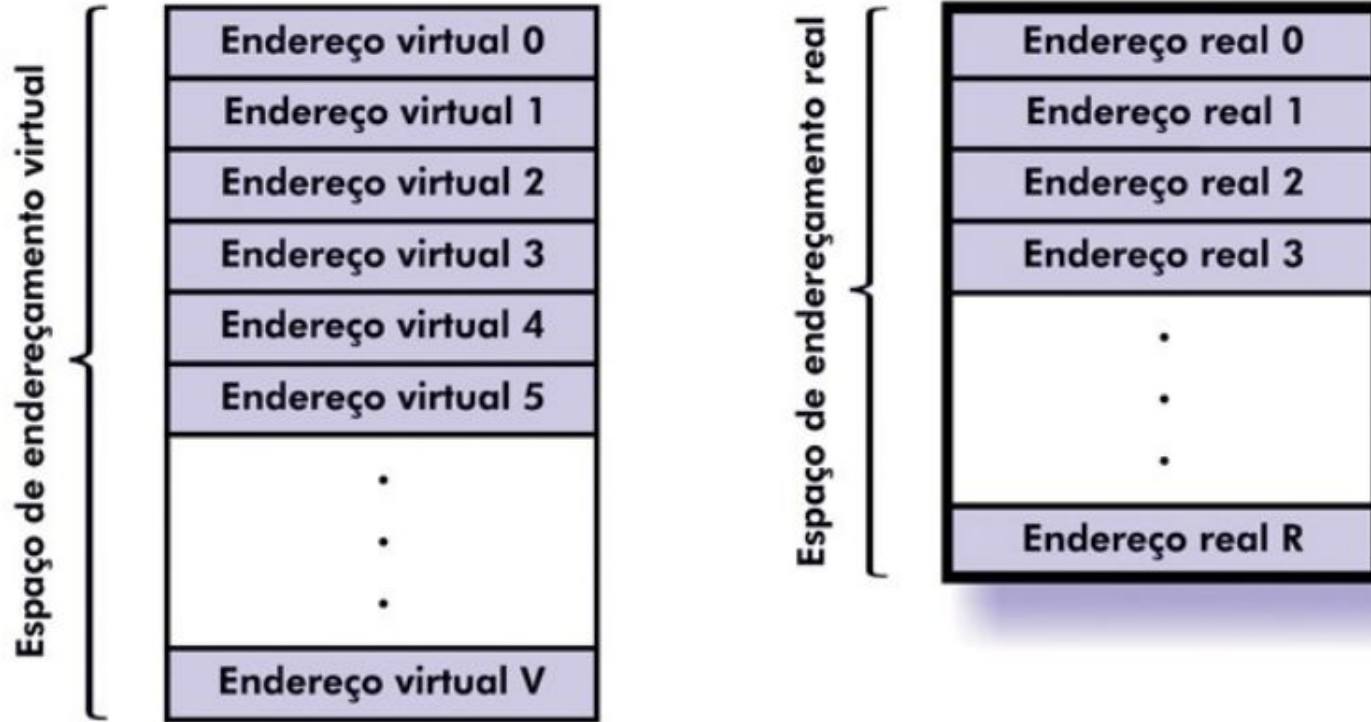


Fig. 10.2 Espaço de endereçamentos virtual e real.

ESPAÇO DE ENDEREÇAMENTO VIRTUAL

- Um programa pode fazer referência a endereços virtuais que estão fora dos limites dos endereços reais
 - Os programas não estão limitados ao tamanho da RAM disponível
 - Uso da memória secundária para estender a memória principal
- Quando um processo é executado, somente parte dele fica residente na RAM
 - O restante fica na secundária até que seja referenciado
 - Isso aumenta o compartilhamento da memória e mitiga a ociosidade da memória

ESPAÇO DE ENDEREÇAMENTO VIRTUAL

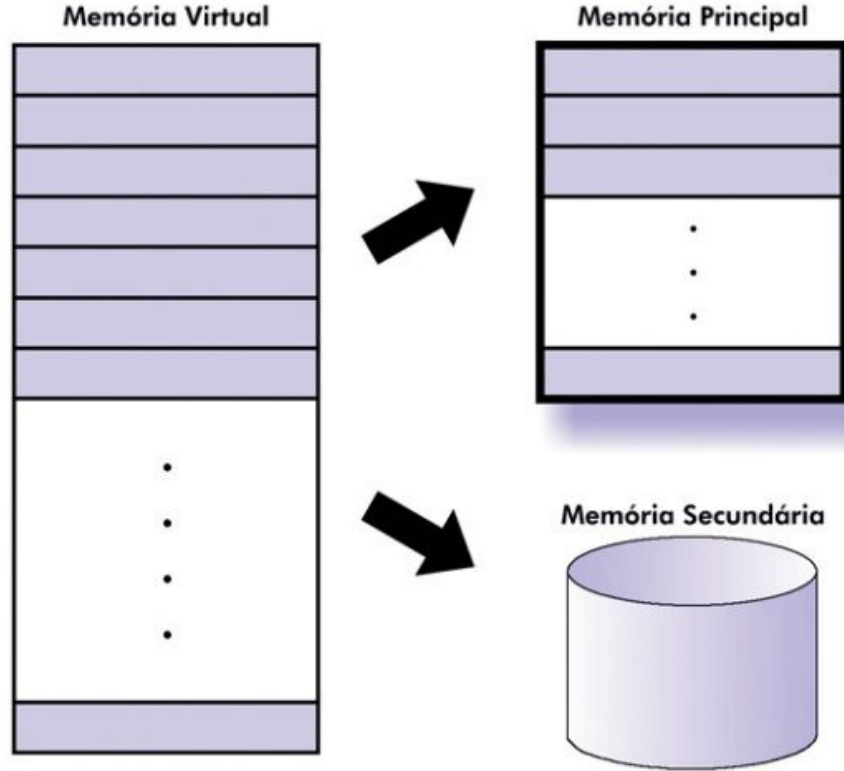


Fig. 10.3 Espaço de endereçamento virtual.

MAPEAMENTO

- O processador apenas executa instruções e referencia dados no espaço de endereçamento real
- Como converter endereços virtuais em reais?
 - Mapeamento ->
- Um programa não necessita estar em espaços contíguos de memória RAM para ser executado
- O mapeamento é realizado em nível de hardware
 - Não comprometer o desempenho
 - Memory Management Unit – MMU
 - Interna ou separada da CPU
 - Após traduzido, o endereço real pode ser utilizado pelo processador
- O processo tem a ilusão de possuir uma memória própria para si
- Cada processo tem uma tabela exclusiva de mapeamento

MAPEAMENTO

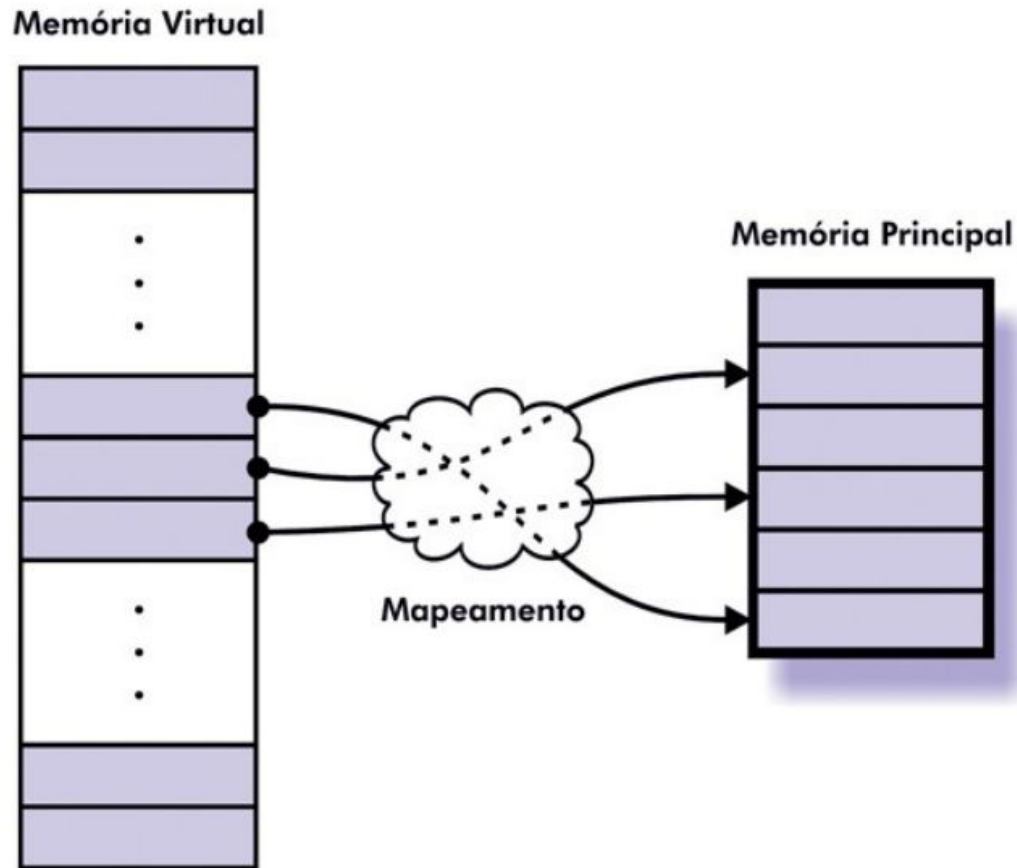
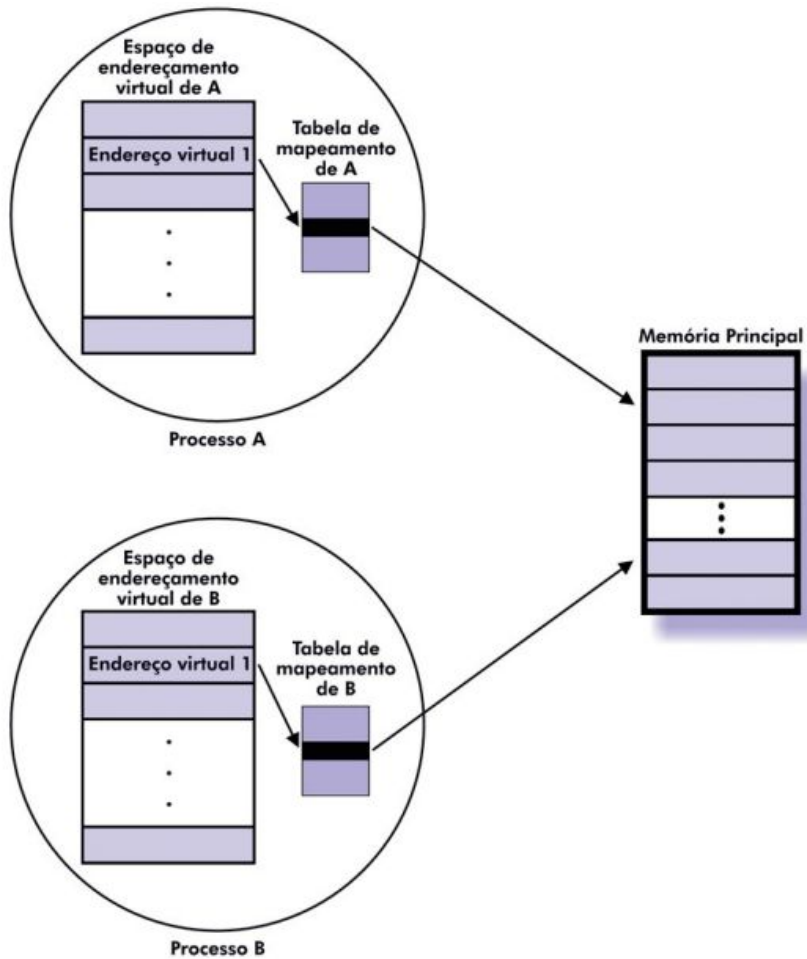


Fig. 10.4 Mapeamento.



ZOOM

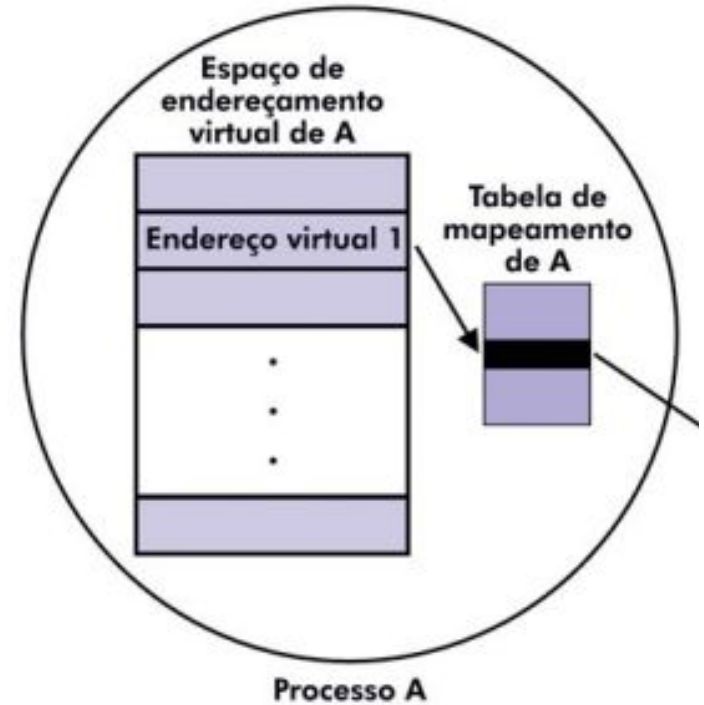


Fig. 10.5 Tabela de mapeamento.

MAPEAMENTO

- O sistema utiliza a tabela de mapeamento correspondente ao processo em execução para realizar a tradução dos endereços virtuais
- Tabelas mapeiam blocos de dados
 - O tamanho do bloco determina a quantidade de entradas na tabela
 - Blocos maiores geram tabelas menores

MAPEAMENTO

Tabela 10.1 Espaço virtual × tamanho do bloco

Espaço de endereçamento virtual	Tamanho do bloco	Número de blocos	Número de entradas na tabela de mapeamento
2^{32} endereços	512 endereços	2^{23}	2^{23}
2^{32} endereços	4 K endereços	2^{20}	2^{20}
2^{64} endereços	4 K endereços	2^{52}	2^{52}
2^{64} endereços	64 K endereços	2^{48}	2^{48}

MEMÓRIA VIRTUAL POR PAGINAÇÃO

- Técnica que divide o espaço de endereçamento virtual e real em blocos de mesmo tamanho
 - Páginas
 - virtuais ou reais (frames)
- O mapeamento de endereço virtual em real é feito utilizando a tabela de páginas
 - Cada processo possui a sua tabela de páginas
 - Cada página virtual do processo tem uma entrada/registro/linha na tabela de páginas com informações que permitem o sistema localizar a página real correspondente ->
- Quando um programa é executado, as páginas saem da memória secundária e vão para a primária nos frames

MEMÓRIA VIRTUAL POR PAGINAÇÃO

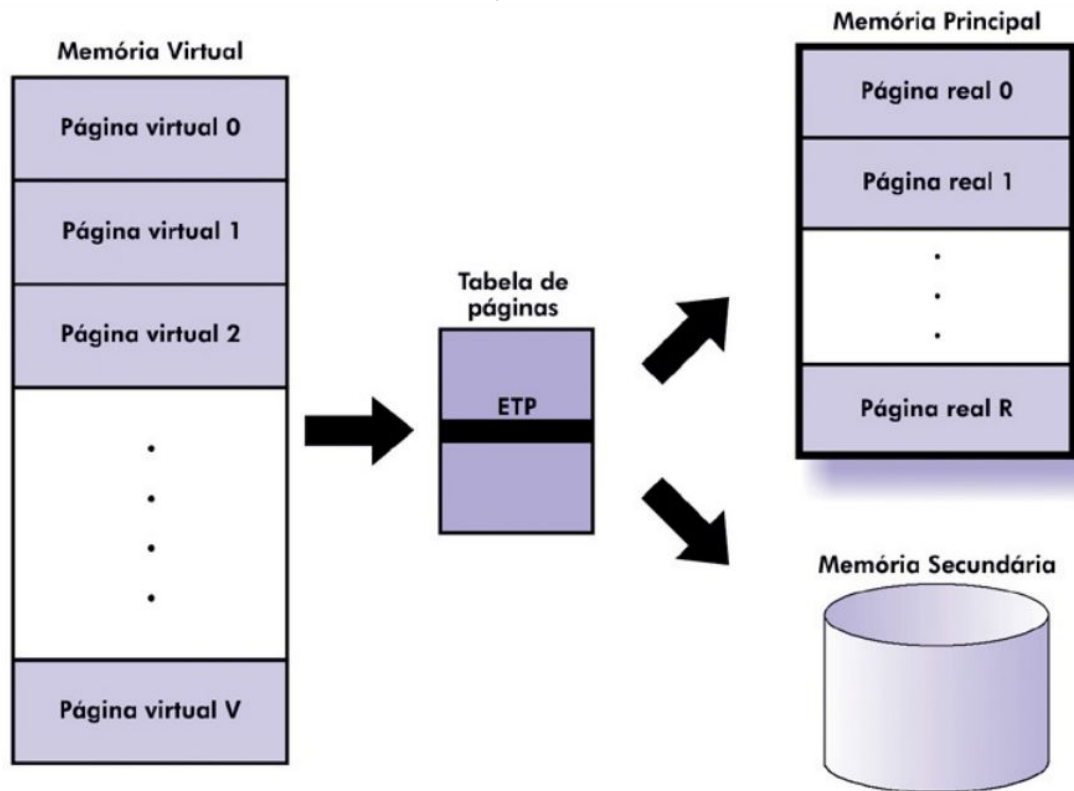


Fig. 10.6 Tabela de páginas.

MEMÓRIA VIRTUAL POR PAGINAÇÃO

- Suponha que um programa faz referência a um endereço virtual
 - O mecanismo de mapeamento localizará na ETP o endereço físico do frame no qual se encontra o endereço real correspondente
- O endereço virtual é formado pelo número da página virtual (NPV) e por um deslocamento
 - $\text{Endereço virtual} = \text{núm. page} + \text{offset}$
 - O NPV identifica unicamente a página virtual que contém o endereço virtual desejado
 - Funciona como índice na tabela de páginas
 - O deslocamento/offset indica a posição do endereço virtual em relação ao início da página na qual se encontra
- O endereço físico é formado pelo endereço do frame e pelo deslocamento já conhecido
 - Endereço do frame é obtido na tabela de páginas
 - $\text{Endereço físico} = \text{end. frame} + \text{offset}$

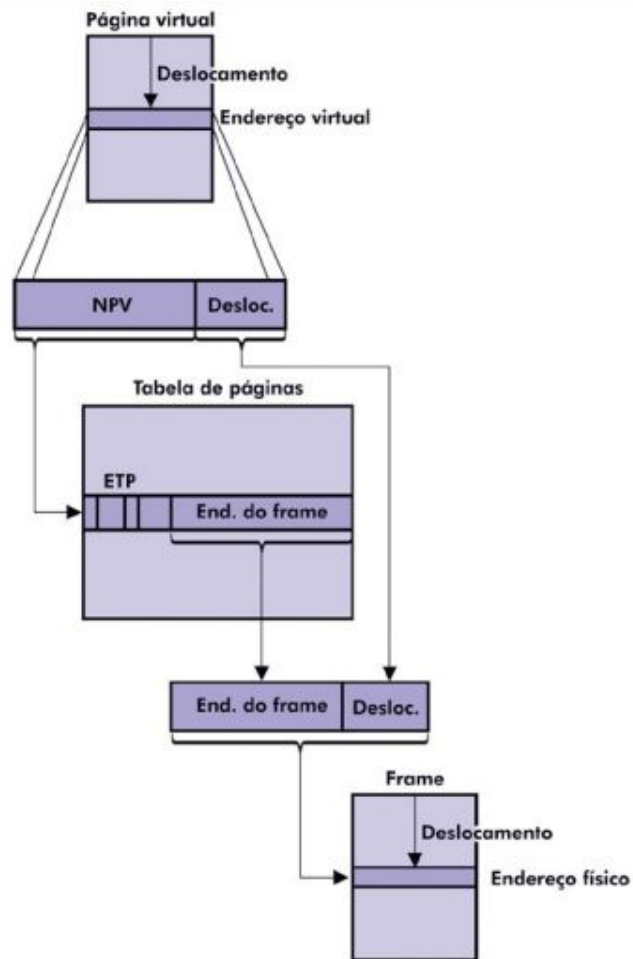


Fig. 10.7 Tradução do endereço virtual.

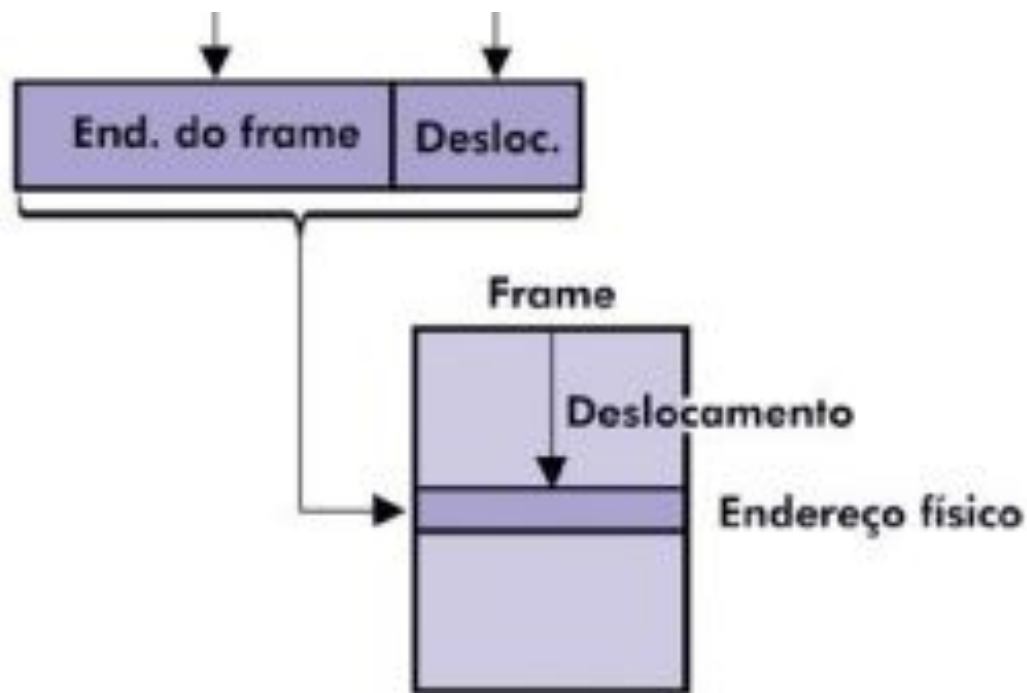
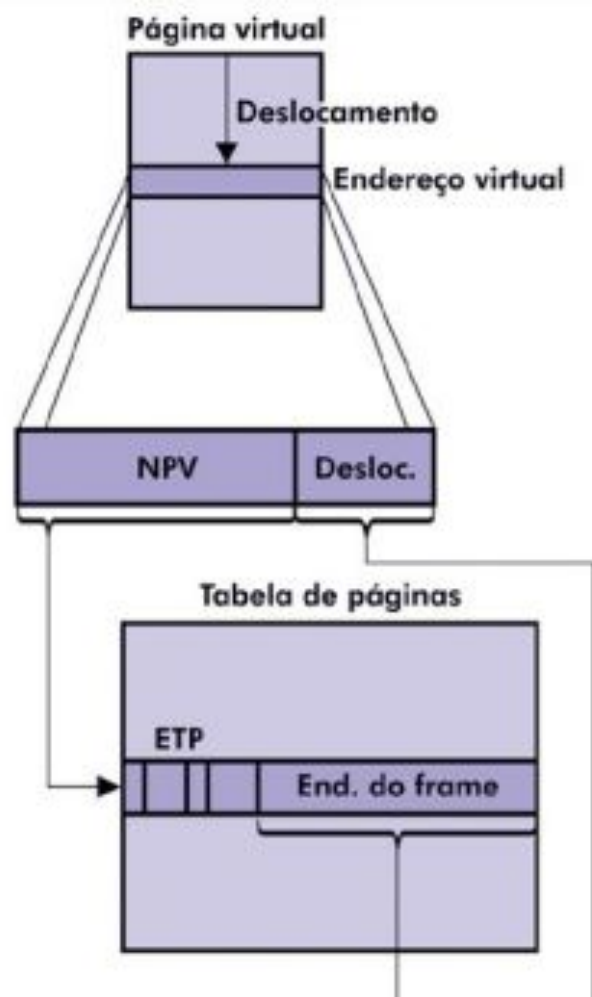


Fig. 10.7 Tradução do endereço virtual.

POSCOMP 2014 - QUESTÃO 45

Considere uma memória paginada, com espaço de endereçamento lógico de 8 páginas, cada uma com 4096 endereços. Nesse caso, a memória física possui 64 quadros. Com relação ao tamanho dos endereços lógicos e físicos, assinale a alternativa correta.

- a) Endereço Lógico possui 15 bits e Endereço Físico possui 18 bits.**
- b) Endereço Lógico possui 15 bits e Endereço Físico possui 12 bits.
- c) Endereço Lógico possui 13 bits e Endereço Físico possui 18 bits.
- d) Endereço Lógico possui 12 bits e Endereço Físico possui 18 bits.
- e) Endereço Lógico possui 12 bits e Endereço Físico possui 12 bits.

POSCOMP 2013 - QUESTÃO 46

Apesar de a alocação de memória em blocos implicar em um mecanismo mais complexo para a conversão entre endereços virtuais e endereços físicos, é a partir do seu conceito que o gerenciamento de memória evoluiu para o que se tem hoje, com o uso de memória cache e memória virtual. Com base nessas informações, considere as afirmativas a seguir.

- I. O endereçamento é facilitado por hardware especializado.
- II. O uso de páginas de tamanho igual a potência de 2 permite um melhor gerenciamento.
- III. O uso de memória cache elimina a necessidade de endereçamento, pois trata as informações como linhas de cache.
- IV. Endereços virtuais não são necessários se não se usar memória virtual. Assinale a alternativa correta.

POSCOMP 2013 - QUESTÃO 46

- a) **Somente as afirmativas I e II são corretas.**
- b) Somente as afirmativas I e IV são corretas.
- c) Somente as afirmativas III e IV são corretas.
- d) Somente as afirmativas I, II e III são corretas.
- e) Somente as afirmativas II, III e IV são corretas

MEMÓRIA VIRTUAL POR PAGINAÇÃO

- A ETP possui não só o endereço do frame correspondente
 - bit de validade
 - Indica se a página está na memória RAM
 - 0 - A página não está na memória RAM
 - 1 - A página está na memória RAM
 - Considere que o processo referenciou uma página
 - O bit de validade é verificado para saber se a página que possui esse endereço está na memória
 - A falha de página ocorre quando o bit de validade é 0
 - Ocorre a paginação para trazer a página referenciada da memória secundária para a principal ->

PAGE FAULT

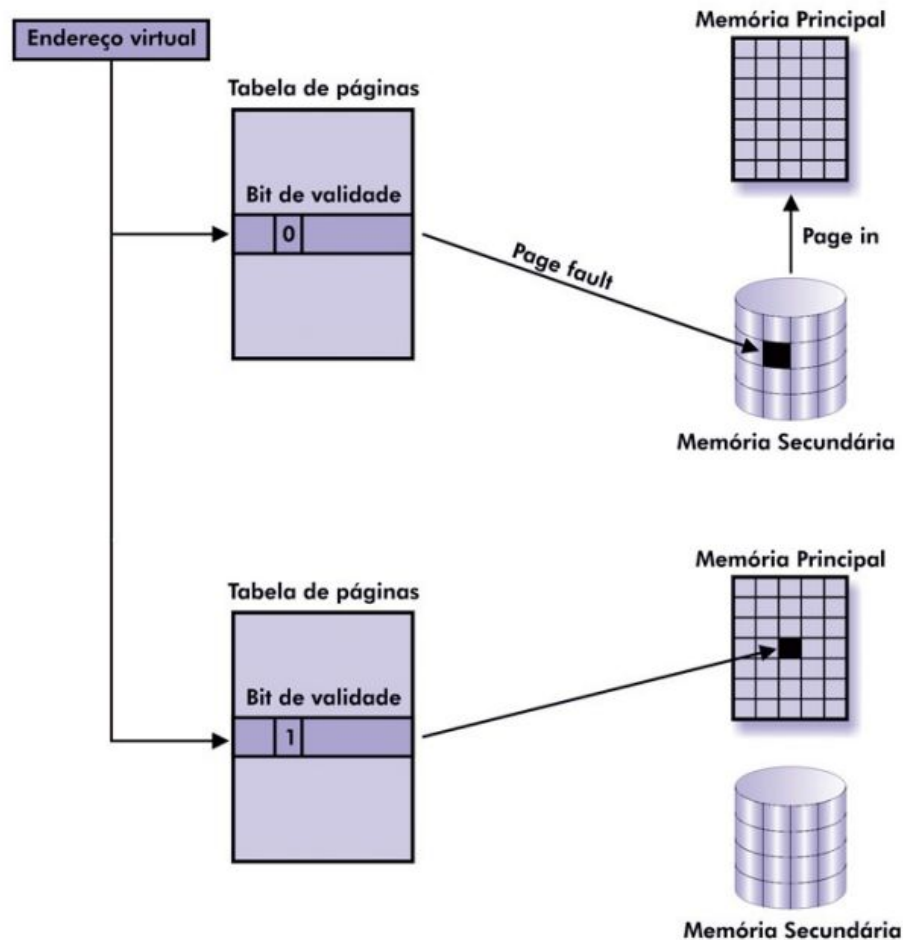


Fig. 10.8 Mecanismo de tradução.

MEMÓRIA VIRTUAL POR PAGINAÇÃO

- O número de page fault gerados por um processo é chamado de taxa de paginação do processo
 - Alta taxa de paginação gera sobrecarga no sistema
 - Excesso de E/S pelo HD
- Na ocorrência de um page fault. O que acontece com o estado do processo?
 - Fica bloqueado até que o swap in seja concluído ->
 - Espera a página sair do disco e ir para a RAM
 - Após concluída a E/S, o processo entra na fila de processos prontos (ready)

MEMÓRIA VIRTUAL POR PAGINAÇÃO

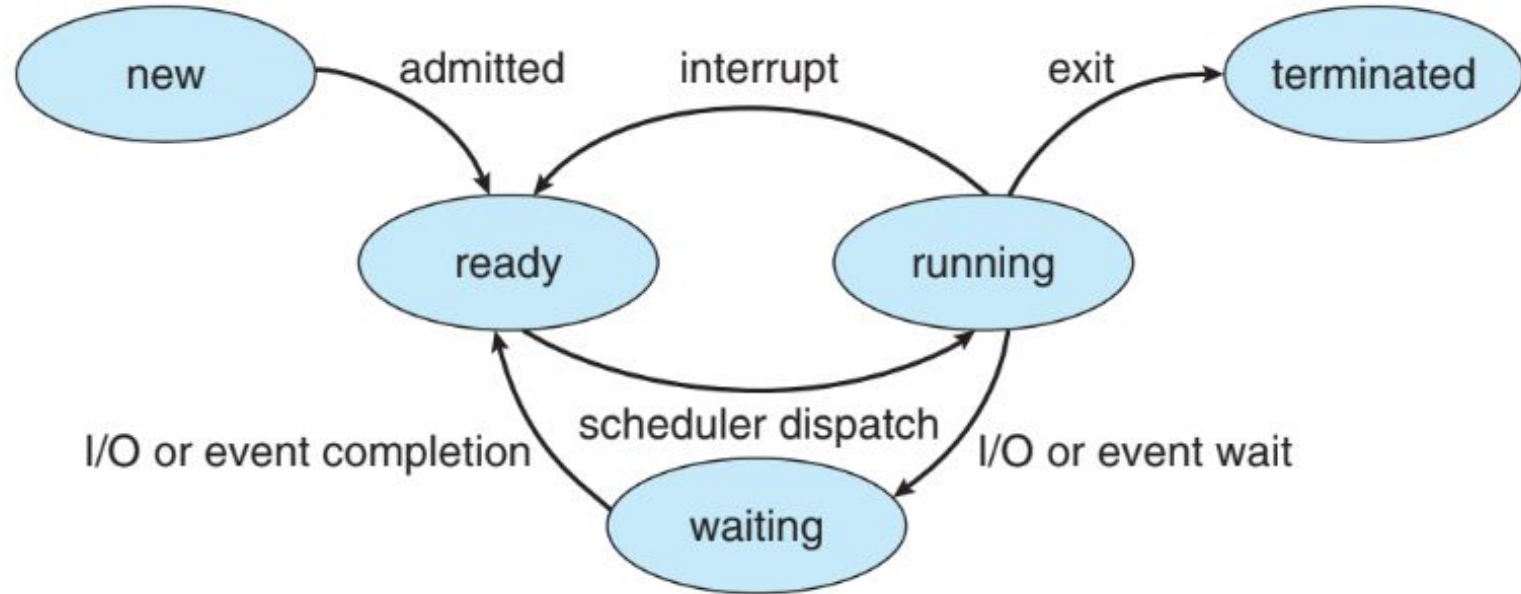


Figure 3.2 Diagram of process state.

POLÍTICAS DE BUSCA DE PÁGINAS

- A política de busca de páginas determina quando uma página deve ser buscada
 - Paginação por demanda
 - Paginação antecipada
- Paginação por demanda
 - As páginas são transferidas do disco para a RAM apenas conforme a necessidade
 - Traz apenas as páginas necessárias naquele momento
 - Talvez partes pouco usadas dos programas nunca sejam carregadas na RAM
- Paginação antecipada
 - Traz páginas além da solicitada naquele momento
 - Essas páginas extras podem ou não serem necessárias
 - É apenas uma expectativa de uso futuro
 - Considere um programa armazenado seguidamente no disco
 - Faz sentido levar várias instruções próximas de uma vez
 - Uma busca traz todas instruções em vez de buscá-las uma a uma
 - Caso não precise, o programa perdeu tempo e espaço

POLÍTICAS DE ALOCAÇÃO DE PÁGINAS

- Determina quantos frames cada processo pode manter na memória RAM
 - Alocação fixa
 - Alocação variável
- Alocação fixa
 - Cada processo tem um número máximo de frames que pode utilizar durante sua execução
 - Deve descartar um página para trazer uma nova
 - O limite pode ser personalizado
 - Necessidades de memória são diferentes
 - Definido no momento da criação do processo
 - Dois problemas
 - E se o limite de páginas for pequeno?
 - Excesso de falha de página
 - E se o limite for muito grande?
 - Toma espaço de outros processos e reduz a multiprogramação

POLÍTICAS DE ALOCAÇÃO DE PÁGINAS

- Alocação variável
 - Cada processo tem um número máximo variável de frames que pode utilizar durante sua execução
 - Sensível a taxa de paginação e ocupação na memória
 - O processo pode aumentar o limite máximo para diminuir a taxa de paginação
 - Processos com baixa taxa de paginação podem ter frames “tomados à força”
- Exige monitoramento constante por parte do S.O.
 - Overhead

POLÍTICAS DE SUBSTITUIÇÃO DE PÁGINAS

- O processo atingiu o seu limite de frames. O que fazer?
 - O S.O deve escolher qual frame deve ser liberado
 - Chamado de política de substituição de páginas
- Uma página real liberada pode ser usada por outro processo
- A substituição de páginas deve considerar se a página foi modificada enquanto estava na RAM, antes que possa ser liberada
 - Dados podem ser perdidos
 - Código executável não sofre desse problema ->
 - Seção apenas leitura (idealmente)
 - Tem backup das instruções no executável do HD
 - Páginas com dados sofrem desse problema
 - Dados e estruturas podem sofrer modificações
 - Necessário salvar as alterações no disco antes do descarte da página -> 2x

LAYOUT DE MEMÓRIA EM C

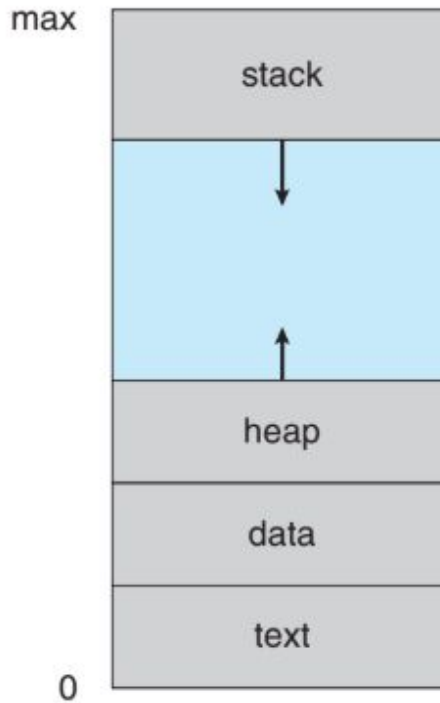
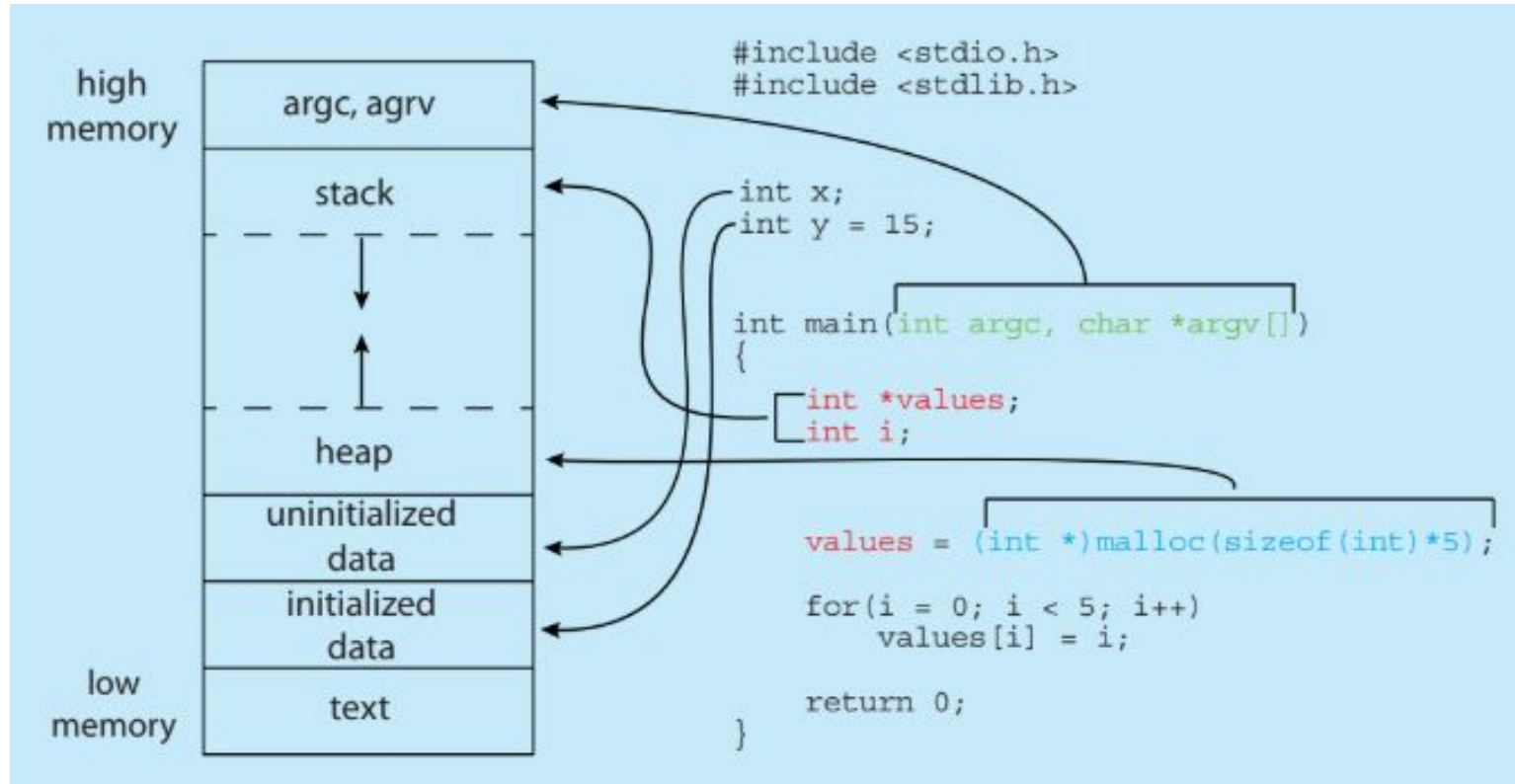
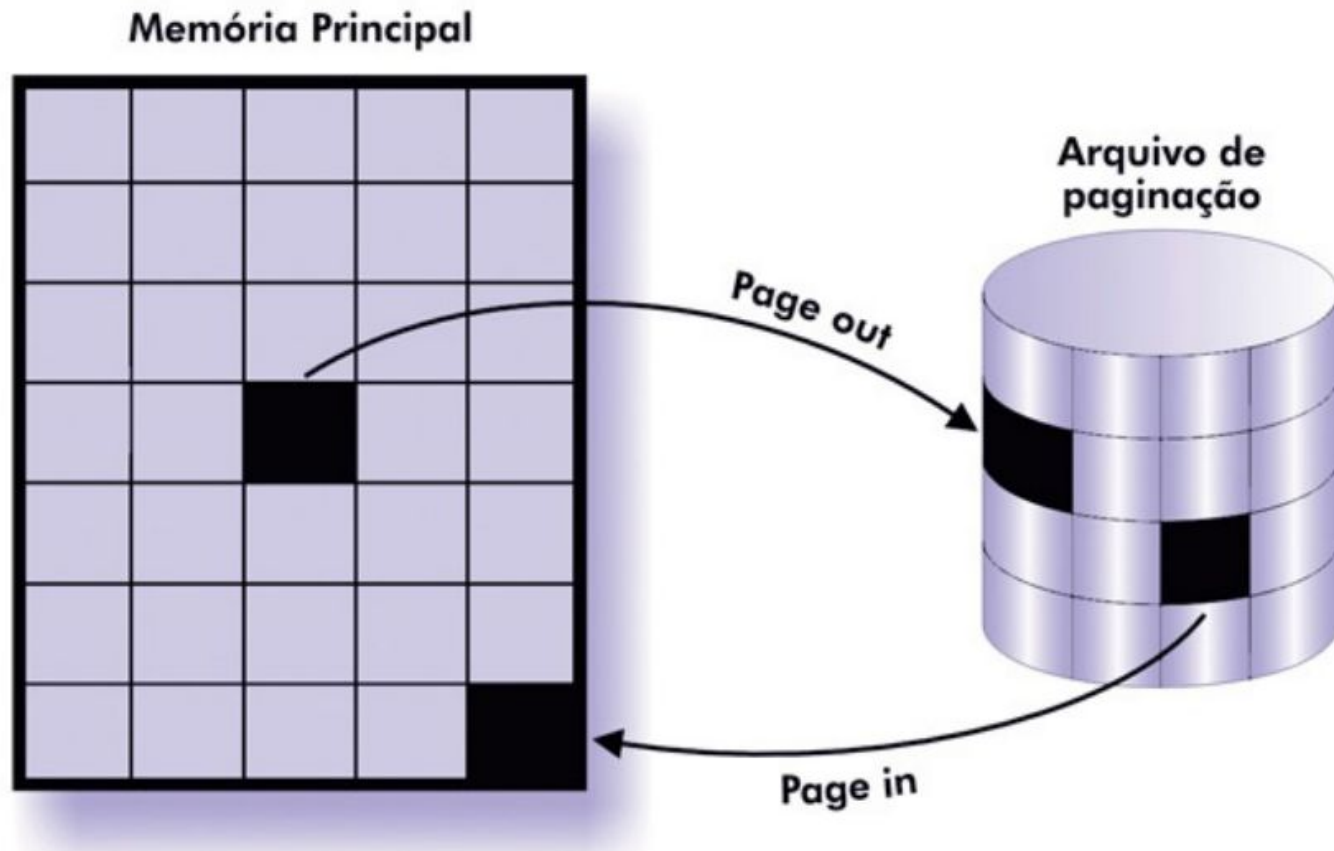


Figure 3.1 Layout of a process in memory.

LAYOUT DE MEMÓRIA EM C



PAGE OUT



POLÍTICAS DE SUBSTITUIÇÃO DE PÁGINAS

- O sistema mantém um arquivo de paginação
 - Chamado de page file
 - Todas as páginas modificadas e descartadas são armazenadas
 - A página sofre page in ao ser referenciada
 - Carregada na memória a partir do arquivo de paginação
- O sistema identifica as páginas modificadas a partir do bit de modificação
 - O valor do bit é modificado quando a página sofre alteração
 - O bit fica na entrada/linha correspondente da tabela de páginas

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- A maior dificuldade não é escolher qual página carregar, mas qual página liberar
 - A escolha errada pode ter consequências negativas no desempenho
- Um processo necessita de um frame, mas não há disponível
 - O sistema deve escolher entre todas as páginas na memória qual vai ser liberada
- Os algoritmos de substituição objetivam remover páginas que não serão referenciadas tão logo
 - O page out de uma página que em seguida será referenciada fará um acesso ao disco desnecessariamente
- O algoritmo perfeito seria aquele que adivinharia qual página não será mais referenciada ou demorará a ser chamada

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- Algoritmos mais sofisticados causam overhead no sistema
- Algoritmo ótimo
 - Seleciona a página a qual não será referenciada no futuro ou demorará muito tempo
 - Garantiria menos taxas de paginação
 - Utopia
 - Não é possível adivinhar o futuro
 - Aplicação apenas teórica
- Algoritmo aleatório
 - “joga um dado” e remove alguma página
 - Não há critério de seleção
 - Todas as páginas são equiprováveis de sofrerem page out
 - Páginas de alta frequência de uso estão vulneráveis também
 - Consome pouco recurso do sistema
 - “pensa pouco”
 - Pouco implementação devido a sua baixa eficiência

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- Algoritmos mais sofisticados causam overhead no sistema
- Algoritmo FIFO (FIRST-IN, FIRST-OUT)
 - A primeira página a entrar na memória será a primeira a sair
 - Seleciona-se a página que está a mais tempo na RAM
 - Implementação
 - Estrutura de dados fila
 - Array com timestamp associado a cada página
 - Arrays.sort(), Collections.sort(), ...
 - As páginas mais antigas estão no início e as mais novas estão no fim
->
- Não necessariamente escolher o mais antigo é uma boa escolha
 - Página com dados frequentemente acessados pode tornar-se antiga
- É interessante levar em conta não apenas o fator tempo

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

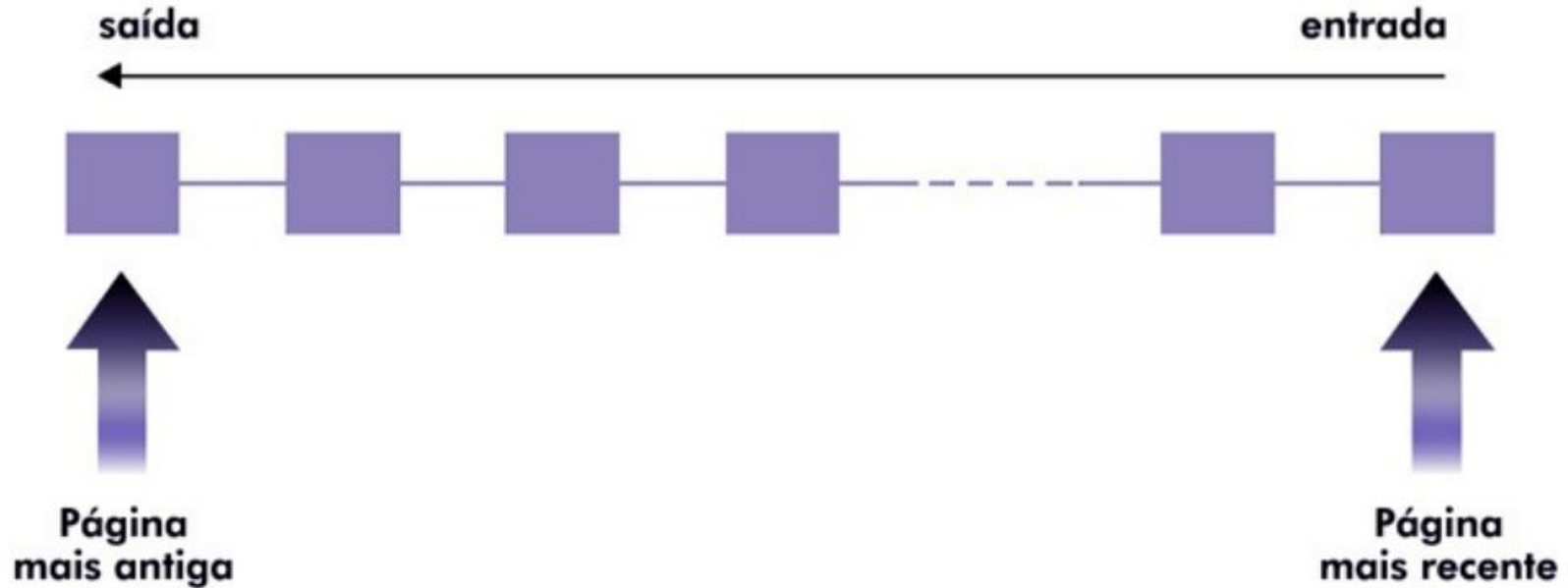


Fig. 10.14 FIFO.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- LFU (Least-Frequently-Used, menos frequentemente usado)
 - Seleciona-se a página menos referenciada
 - O frame menos utilizado
 - É mantido um contador para cada página que está na memória principal
 - O contador conta quantas vezes a página foi referenciada
 - A página com o menor contador vai ser eliminada
 - O algoritmo evita selecionar páginas que são muito referenciadas
 - Um problema surge por existir páginas que acabaram de sofrer page in
 - Página que acabaram de entrar na RAM estão em desvantagem porque o seu contador é recente
 - Uma página muito referenciada no passado pode ter deixado de receber tanta referência
 - Seu contador é grande devido ao passado
 - Leva vantagem em relação às páginas novas

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- LRU (Least-Recently-Used, menos recentemente usada)
 - Seleciona a página que está a mais tempo sem ser referenciada
 - Localidade temporal
 - Uma página que não foi referenciada recentemente tende a não ser referenciada em breve
 - Precisa registrar o momento da última referência
 - Timestamp
 - Cada referência ao frame atualiza o timestamp
 - O sistema busca o frame a mais tempo sem ser referenciado quando necessário fazer uma substituição de página
 - Busca linear em ordem desordenada de timestamp
 - Busca binária em ordem de timestamp
 - heap-min com prioridade no timestamp
 - Lista encadeada ordenadas pelo timestamp
 - Custo de implementação

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- FIFO circular (clock)
 - Estrutura de lista circular
 - Um ponteiro aponta para a página mais antiga da lista
 - Cada página tem um bit de referência
 - Indica se a página foi referenciada recentemente
 - O sistema verifica se o bit de referência $BR = 0$ quando precisa substituir alguém
 - Se $BR = 0$ da página apontada, significa que é a página mais antiga e que não foi referenciada recentemente
 - Descarte
 - Se $BR = 1$ da página apontada, significa que é antiga mas ainda é bem referenciada
 - Desliga-se o BR e o ponteiro aponta para o próximo elemento mais antigo
 - Repete até encontrar alguma página com $BR = 0$
 - Se todos $BR = 1$?
 - O uso do BR concede uma segunda chance à página

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

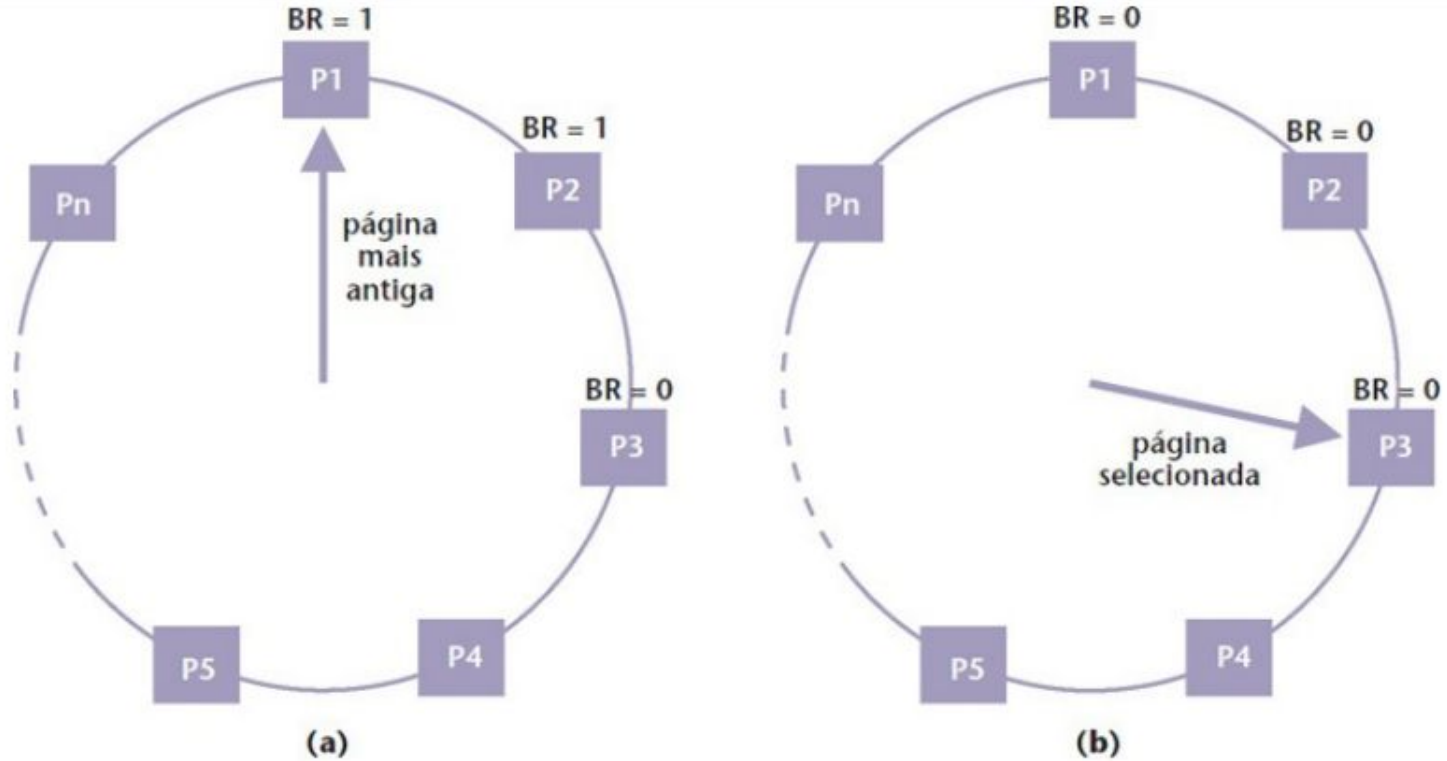


Fig. 10.16 FIFO circular.

ENADE 2005 - QUESTÃO 22

Com relação ao gerenciamento de memória com paginação em sistemas operacionais, assinale a opção correta.

- A) As páginas utilizadas por um processo, sejam de código ou de dados, devem ser obrigatoriamente armazenadas na partição de swap do disco, quando o processo não estiver sendo executado
- B) Todas as páginas de um processo em execução devem ser mantidas na memória física enquanto o processo não tiver terminado
- C) Um processo somente pode ser iniciado se o sistema operacional conseguir alocar um bloco contíguo de páginas do tamanho da memória necessária para execução do processo
- D) O espaço de endereçamento virtual disponível para os processos pode ser maior que a memória física disponível**
- E) Um processo somente pode ser iniciado se o sistema operacional conseguir alocar um bloco contíguo de páginas do tamanho da memória necessária para execução do processo

POSCOMP 2022 - QUESTÃO 62

Qual é o conceito no qual o sistema operacional permite que o computador execute diversos programas – ou processos – ao mesmo tempo e, se houver apenas uma unidade central de processamento (CPU), o sistema operacional executa alguns comandos de um processo, depois suspendem esse processo e executam alguns comandos do próximo processo, e assim por diante?

- A) Sincronização
- B) Multiprogramação**
- C) Difusão de mensagens
- D) Comunicação entre processos
- E) Tolerância a falhas


POSCOMP 2023 - QUESTÃO 42

Em um computador com suporte à memória virtual e paginação, quando ocorre um page fault, o sistema operacional, às vezes, precisa escolher uma página da memória principal (page frame) para dar lugar à página virtual que será carregada do disco como resultado do page fault. Dependendo do tipo de conteúdo presente na página selecionada para substituição, esse conteúdo precisa ser salvo no disco (page out) antes da substituição. Assinale a alternativa que indica uma região de memória típica de um processo, cujo conteúdo não exige salvamento prévio em casos de troca de páginas (page replacement).


- A) Dados alocados dinamicamente (ex.: HEAP).
- B) Dados não inicializados (ex.: BSS).
- C) Dados inicializados (ex.: DATA).
- D) Código (ex.: TEXT).**
- E) Pilha (ex.: STACK).

POSCOMP 2023 - QUESTÃO 44

Considere que o programa abaixo, escrito em Linguagem C, execute em um computador com suporte à memória virtual e segmentação paginada, tal como em sistemas baseados em Intel x86-64.



```
1 #include <stdio.h>
2 main(){
3     int w;
4     printf("%p",&w);
5 }
```



```
1 #include <stdio.h>
2 main(){
3     int w;
4     printf("%p",&w);
5 }
```

A) Endereço físico representando o segmento, a página e o offset onde se localiza a variável w.

B) Endereço virtual associado ao endereço físico onde se localiza a variável w.

C) Resultado do processamento realizado pela MMU (Memory Management Unit).

D) Endereço da variável w no espaço de endereçamento físico do processo.

E) Endereço da variável w no working set do processo.