Universidade Federal de Uberlândia
Faculdade de Computação
Sistemas Operacionais

Gerenciamento de Memória Física

Prof. Dr. Marcelo Zanchetta do Nascimento

Roteiro

- Introdução
- Hardware
- Unidade de Gerenciamento da Memória (MMU)
- Formas de Alocação de Memória
- Segmentação
- Paginação
- Exemplo na Arquitetura Intel 32-bits ou 64 bits
- Leituras Sugeridas

Introdução

- O propósito geral de um sistema computacional é executar programas:
  - ***Programa (código + dados) deve estar na memória;***
- No uso da CPU, há vários programas presentes na memória (Multi-programação);
  - Necessidade de uma política de **gerenciamento da memória**;
- Diferentes estratégias são aplicadas de acordo com requisitos, algoritmos e suporte de hardware para gerenciamento desse recurso.

Introdução

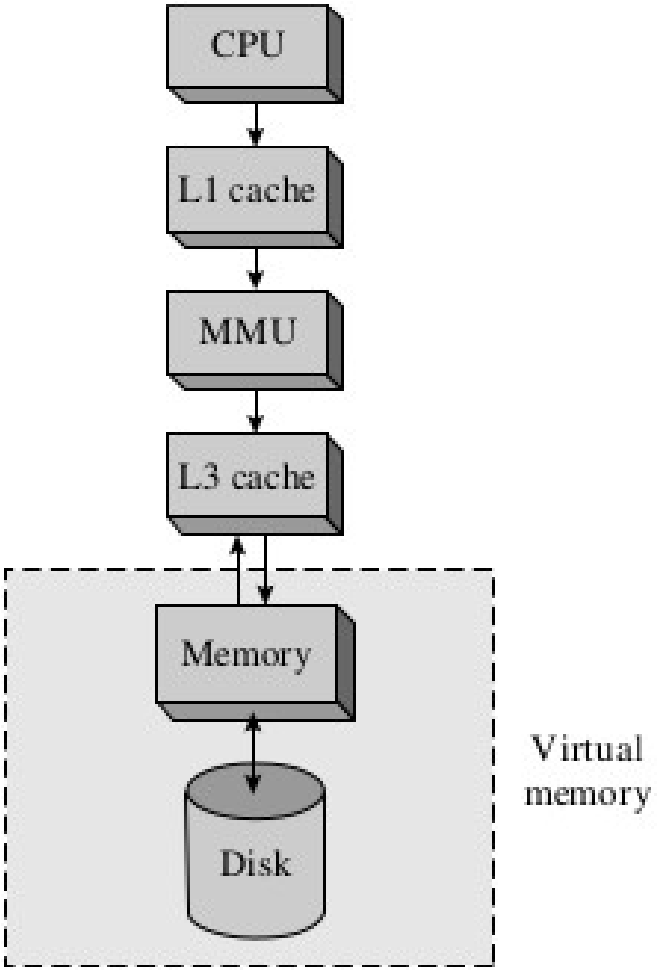


Figura 1. Gerenciamento da Memória

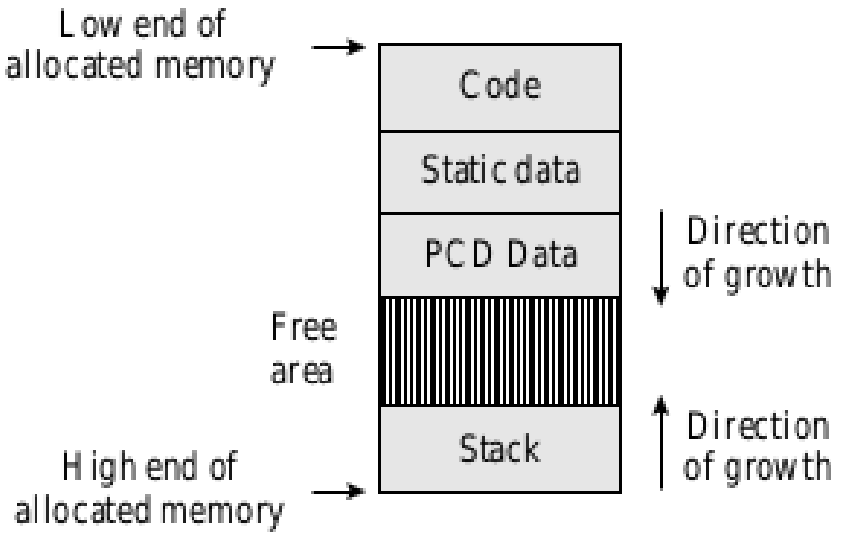


Figura 2. Alocação do Processo

Introdução - Vinculação

Vinculação de endereço pode acontecer em:

Tempo de compilação:

- Se a localização da memória é conhecida a priori, o código absoluto pode ser gerado;
- Deve recompilar o código se iniciar as alterações de local.

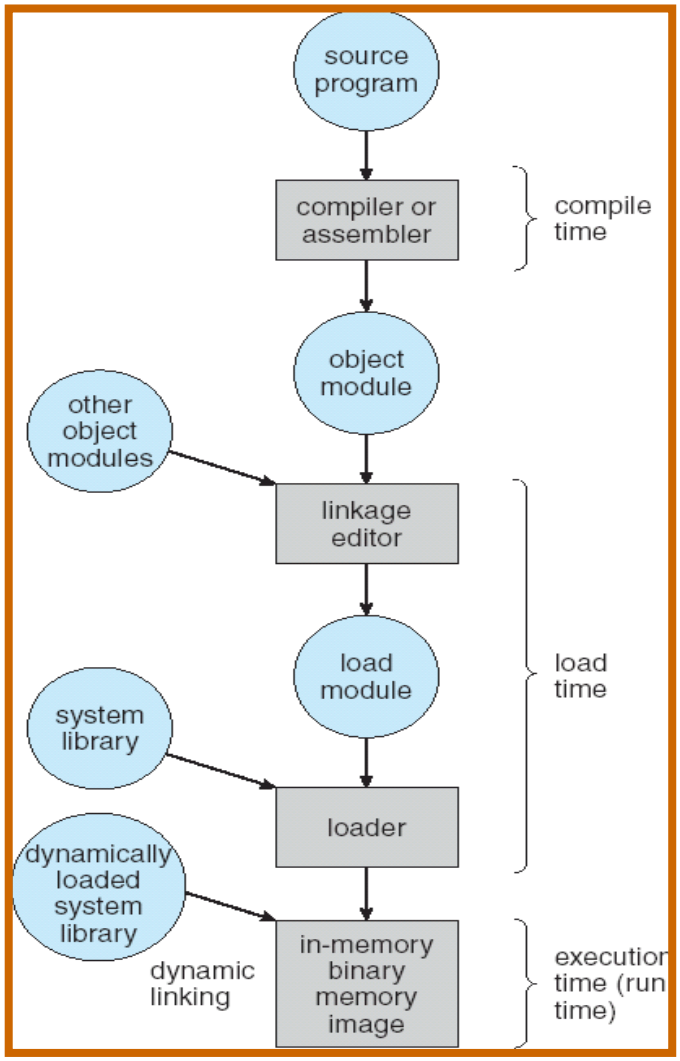
Tempo de carga:

- Deve gerar código realocável se a localização da memória não for conhecida em tempo de compilação;

Tempo de execução:

- Processo pode ser movido durante a sua execução de um segmento de memória para outro;
- Precisa de suporte de hardware para mapas de endereços;
- Mais comuns.

Introdução - Vinculação



- Os endereços de um programa fonte são geralmente simbólicos (tais como uma variável “count”).
- O compilador tipicamente vincula esses endereços simbólicos para endereços realocados,

Figura 3. Processamento de um programa de usuário em vários passos.

Introdução

Principais funções em Linguagem C para alocação de memória:

- void *malloc (size_t size);
- void *calloc (size, size_t);
- void free (void *ptr);
- void *realloc (void *ptr, size_t size);

Endereçamento na memória

- **Memória física:**
  - Memória “real” do sistema implementada com “CIs”;
  - Possui áreas reservadas (ex. vetor de interrupções);
  - Endereço físico:
    - Acessa posições da memória física.

- **Memória lógica (virtual):**
  - Memória que um processo pode acessar;
  - Utiliza os endereços lógicos;
  - Necessita tradução dos endereços lógicos para físicos.

Endereçamento (Lógico versus Físico)

O mapeamento entre endereço lógico e físico ocorre:

- Emprega o Hardware: Unidade de Gerenciamento de Memória (MMU – memory-management unit) para mapear endereço lógico para endereço físico.

Endereço Lógico:

- Gerado pela CPU;
- Também referenciado como endereço virtual;
- Programas de usuário trabalham com endereço lógico, não manipulam os endereços físicos.

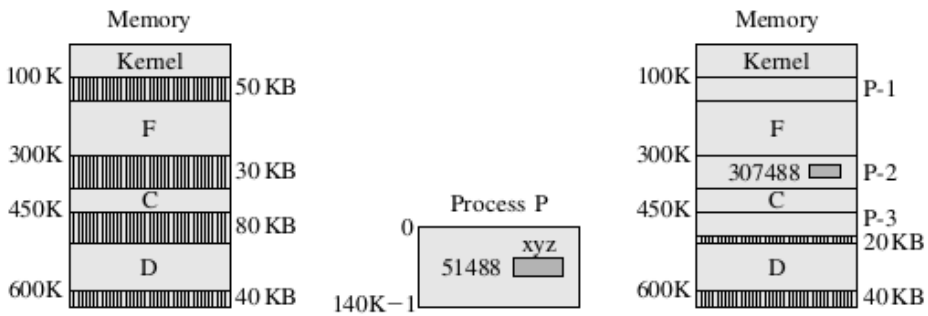
Endereço Físico:

- Endereço que a unidade de memória trabalha.

MMU – memory-management unit

Exemplo:

- O espaço de endereço do processo P é 0 até 140K − 1.
- Os dados da variável "**xyz**" no processo tem o endereço lógico 51.488, a qual é situada no componente da página P-2.
- A MMU organiza a tradução.



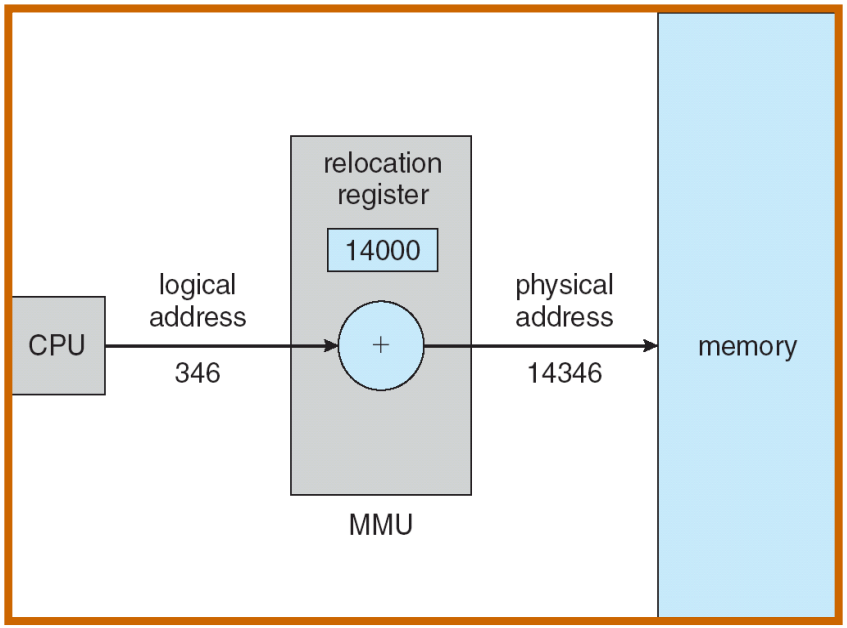


Figura 4. Realocação dinâmica usando um registrador

Hardware

- Instruções trabalham com os endereços de memória (argumento), mas não os endereços do disco;
- Para garantir que cada processo tenha um espaço de memória, o ***registrador de alocação*** contém o endereço base para realizar a tradução.

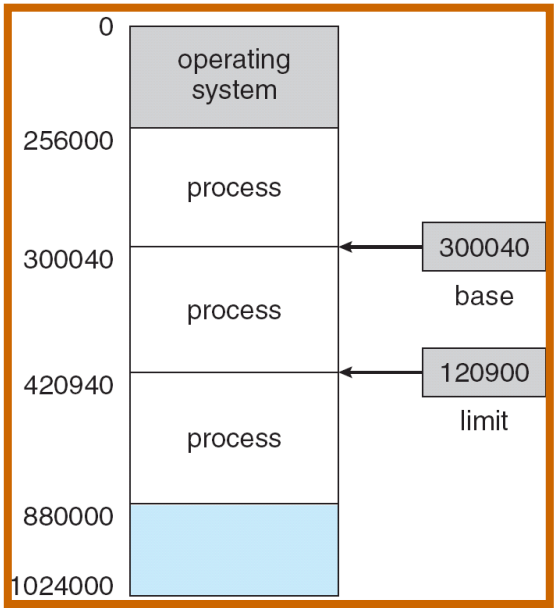


Figura 5. Definição do espaço de endereçamento

Hardware

- A **proteção** ocorre quando o **hardware da CPU compara** os endereços gerados no modo usuário com os registradores;
- Qualquer tentativa de violar a região, uma ***trap*** é enviada para o monitor (supervisor);
- O SO que carrega os **registradores base** e **limite ao processo**.

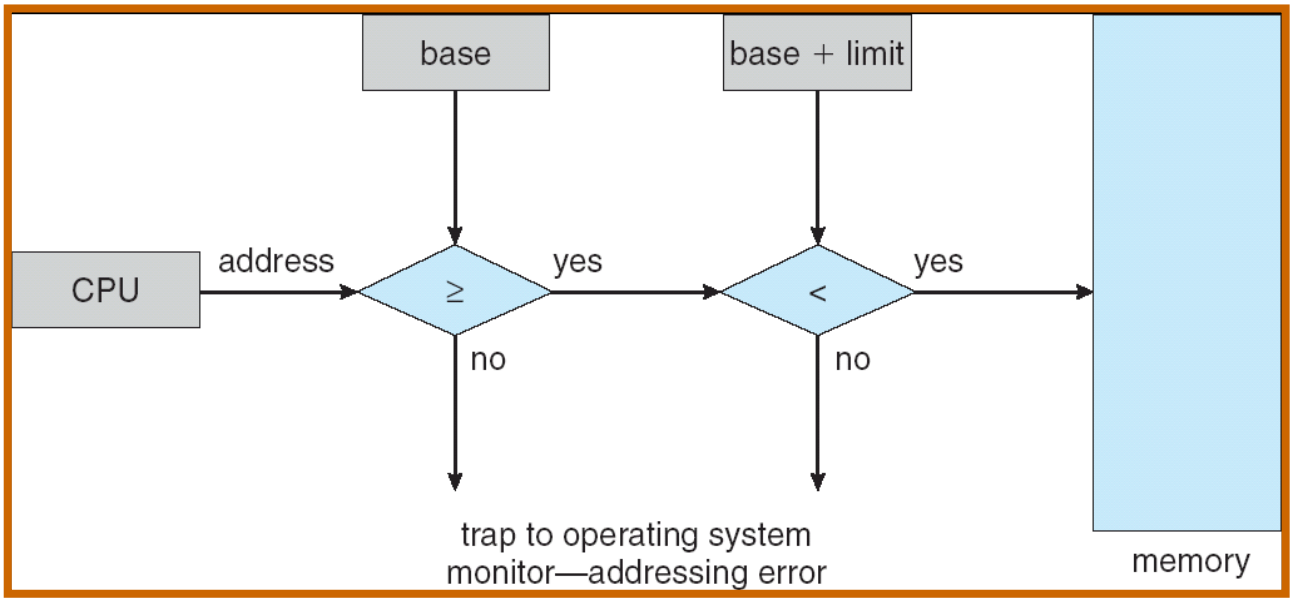


Figura 6. Proteção de endereços de hardware

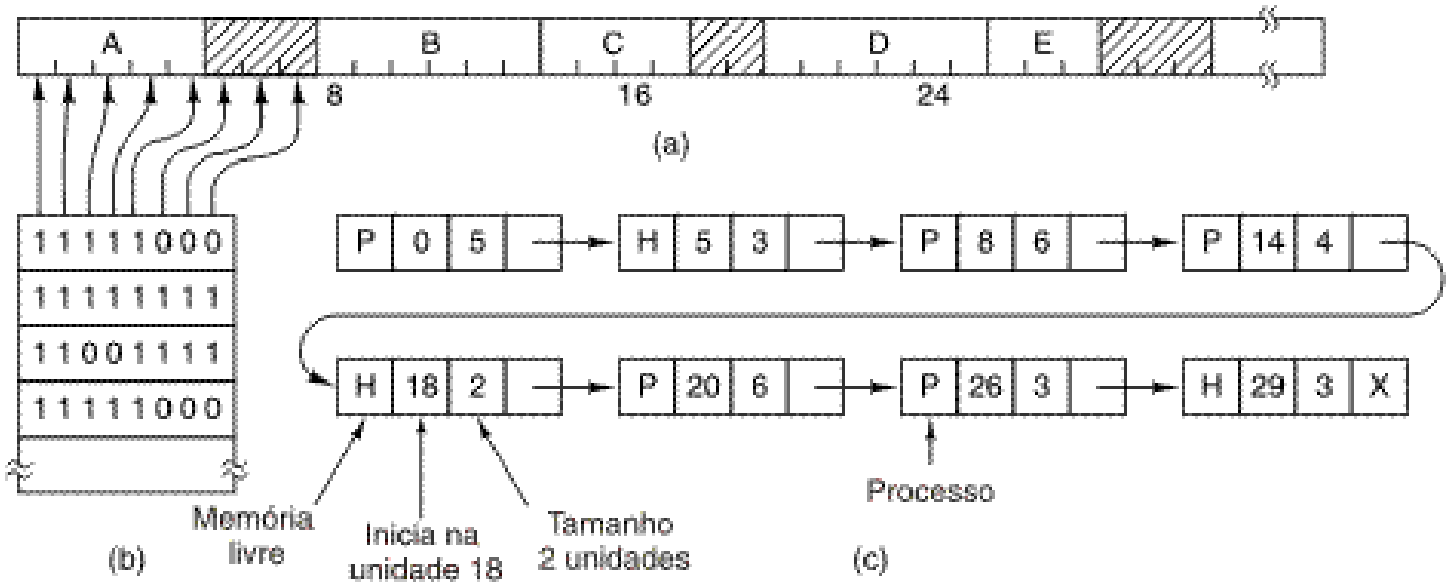
Gerenciamento de espaço na memória

- O S.O. deve gerenciar o espaço de memória atribuída de forma dinâmica aos processos;
- As principais técnicas empregadas para o gerenciamento:
  - Técnica baseada em lista encadeada;
  - Técnica baseada em mapa de *bits:*
    - Memória é dividida em unidades de alocação: kbytes;
    - Cada unidade corresponde a um *bit* no mapa:

      0 - livre

      1 – ocupado

Gerenciamento de espaço na memória



a) Memória: região da memória com 5 segmentos de processos e 3 segmentos de memória livre

- pequenos riscos denotam as unidades de alocação
- regiões sombreadas denotam segmentos livres

b) Técnica de mapa de bits

c) Técnica com a lista encadeada

Swapping

Swapping é uma técnica para resolver o **problema da insuficiência de memória:**

Antes: O programa ficava na memória até o fim da sua execução, enquanto os outros esperavam por memória livre.

swapping: O sistema retira temporariamente um programa da memória, coloca-o no disco (swapp out), para a entrada de outro.

Exemplo: Algoritmo Round Robin

- Se o quantum terminou, o gerenciador de memória começará a descarregar o processo que finalizou e carregará outro processo para o espaço liberado.

Swapping

Exemplo: Escalonamento por prioridade:

- Processos com alta prioridade alocam memória e processos com baixa são eliminados da área de memória principal;

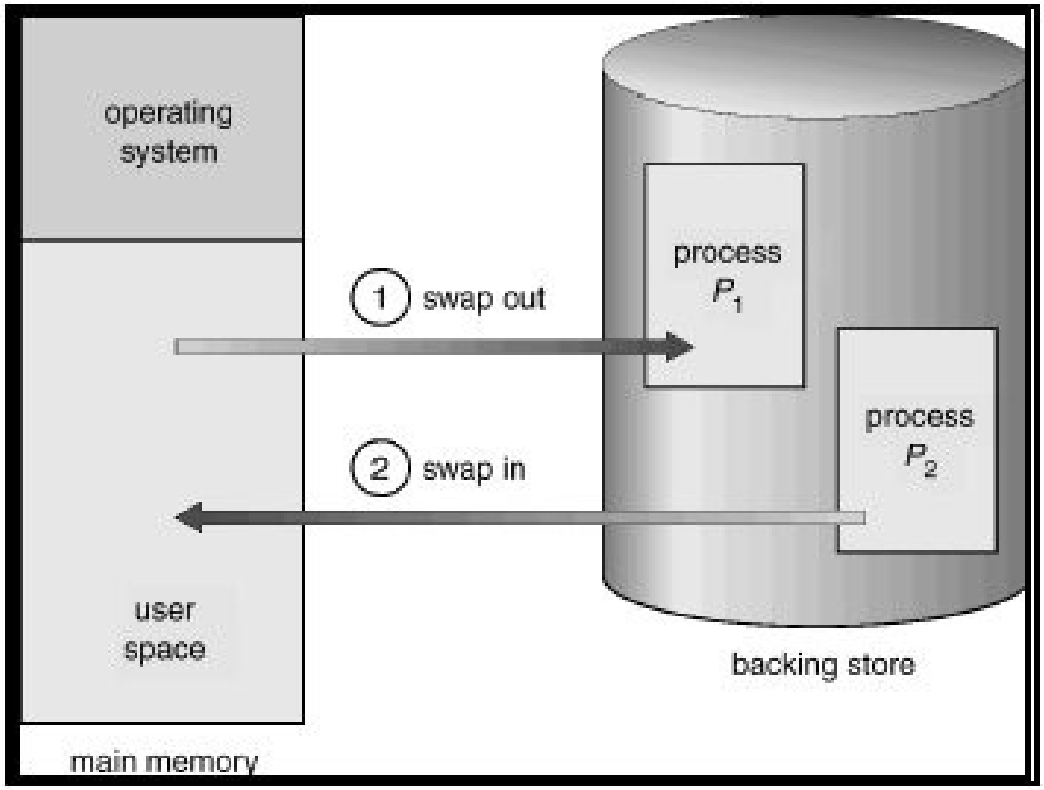


Figura 7. Permuta entre 2 processos usando um disco

16

Alocação de espaço na Memória

- **Alocação de espaço na memória**:
  - **Contígua:** Se existe memória principal suficiente não há necessidade de ter divisões;
    - Não há chaveamento entre processos;
  - **Alocação não contígua:** com chaveamento (Multiprogramação):
    - Processos são movidos entre a memória principal e o disco;
    - Artifício usado para resolver o problema da falta de memória;

Alocação de espaço na Memória

Alocação Contígua

- A alocação pode ter partições de tamanho fixo que contém um espaço para um processo;
- Existe um esquema de partições de tamanho variável, a qual indica a parte ocupada e parte livre na memória;
- As técnicas denominadas first-fit, best-fit e worst-fit são as mais comuns:
  - São usadas para selecionar um espaço livre do conjunto de espaço disponível na memória RAM.

Alocação de espaço na Memória

Alocação Contígua: *o* IBM mainframe com OS/MFT (Multiprogramming with a Fixed Number of Tasks) com partições fíxas.

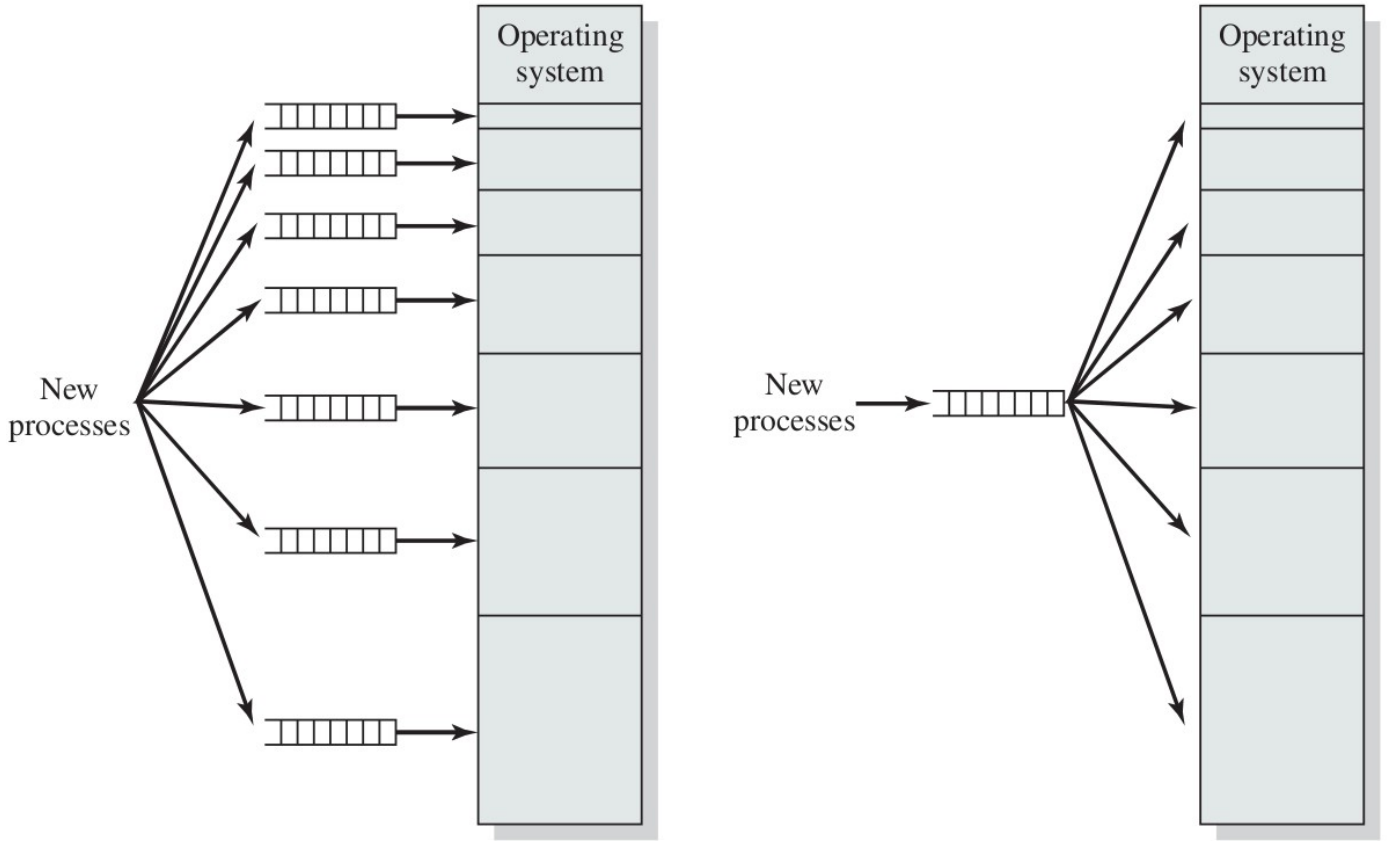


Figura 8. Memória com partições fixas

Alocação de Memória

Best fit: Escolhe o melhor espaço, ou seja, aquela em que o programa deixa o menor espaço sem utilização;

A tendência é que a memória fique cada vez mais com pequenas áreas livres não contíguas

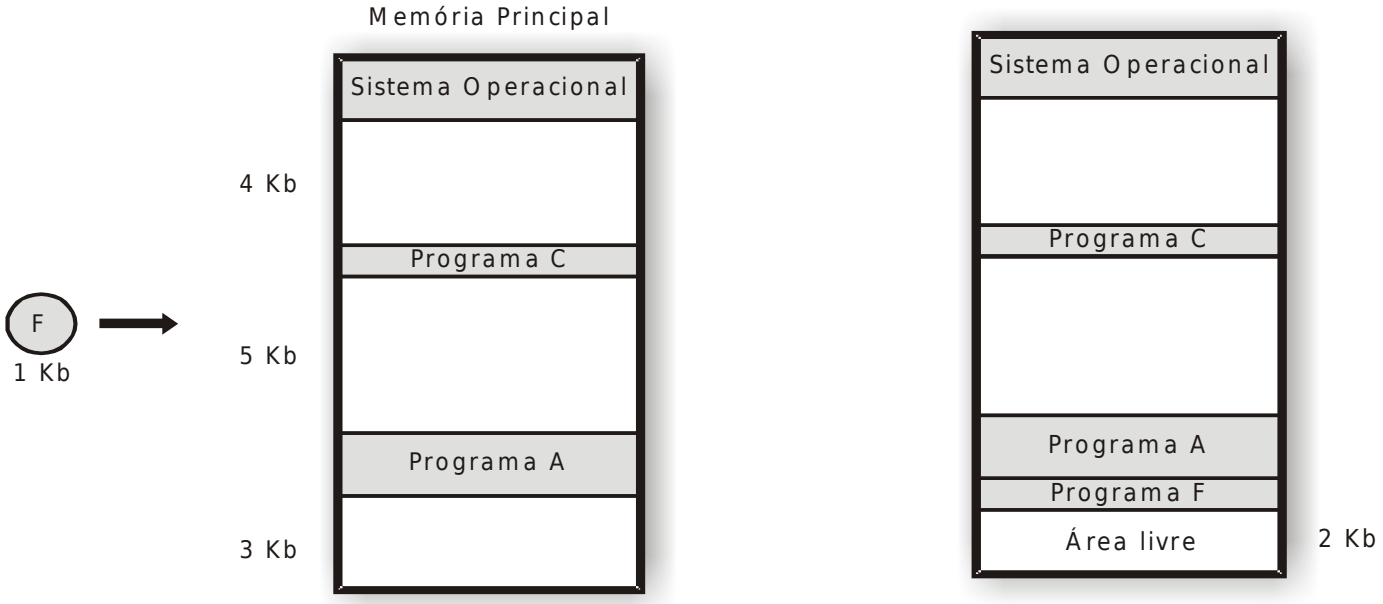


Figura 9. Exemplo de alocação com best-fit

Alocação de Memória

Worst fit: Escolhe o pior segmento, ou seja, aquela em que o programa deixa o maior espaço sem utilização;

Deixando espaços maiores, a tendência é permitir que um maior número de programas utilize a memória, diminuindo o problema da fragmentação.

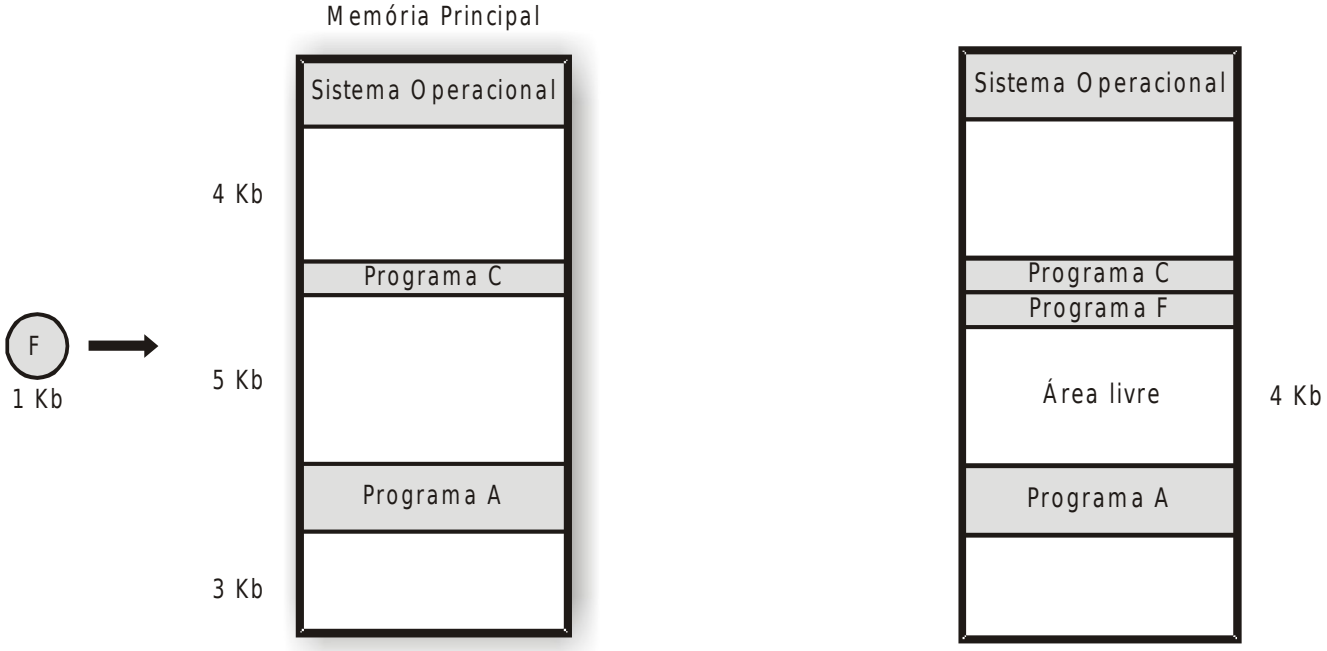


Figura 10. Exemplo de alocação com worst-fit

Alocação de Memória

First fit: escolhe o primeiro segmento livre que seja suficiente para carregar o programa.

Algoritmo mais rápido das três abordagens (best / worst / first).

Consome menos recursos para a busca

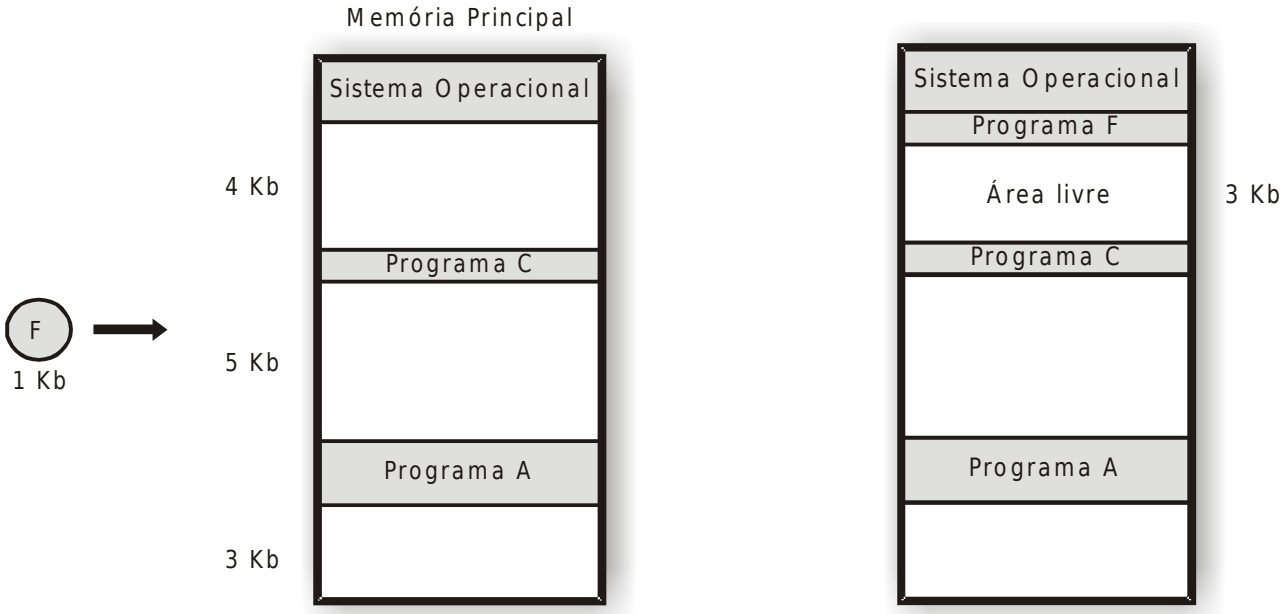


Figura 11. Exemplo de alocação com first-fit

Alocação de Memória

Fragmentação:

Interna: programa é carregado em uma partição um pouco maior que o necessário.

Externa: não há espaço disponível para atender um processo

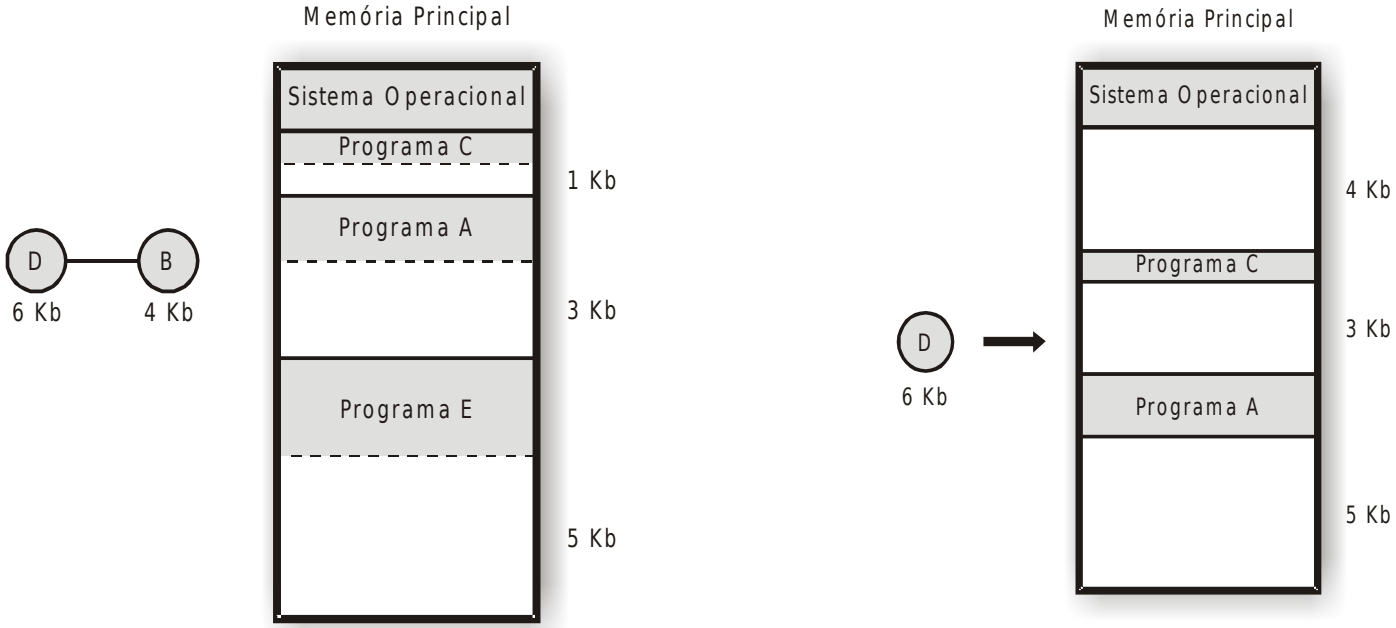


Figura 11. Fragmentação interna e externa

Alocação Não Contígua: Segmentação

- Sistemas modernos **não trabalham** com alocação contígua;
- **Segmentação:** esquema de gerenciamento de memória que suporta visão do usuário na memória;
- Aproveita a modularidade do programa em que a memória não é dividida em tamanhos fixos, mas baseado na estruturação do programa;
- Isso permite que os programas sejam divididos logicamente em sub-rotinas e estruturas de dados;

Alocação Não Contígua: Segmentação

- O compilador cria os segmentos:
  - As variáveis globais,
  - Chamadas de procedimento,
  - A parte do código para cada procedimento,
  - As variáveis locais.
- Essa estrutura é empregado na memória.

Segmentação

- **Tabela de segmento** – mapeia o endereço lógico ao endereço físico;
- Cada tabela é composta por:
  - **base** – contém o endereço físico inicial onde o segmento reside na memória;
  - **limite** – especifica o tamanho do segmento.
- ***Segment-table base register*** **(STBR)** aponta para a localização da tabela de segmento na memória;
- ***Segment-table length register*** **(STLR)** indica o número de segmento usado por um processo;
  - O número do segmento ***s*** é permitido se ***s*** < **STLR**

Segmentação

- O hardware de mapeamento verifica o bit de proteção associado com cada entrada na tabela de segmentos para evitar acesso ilegal.

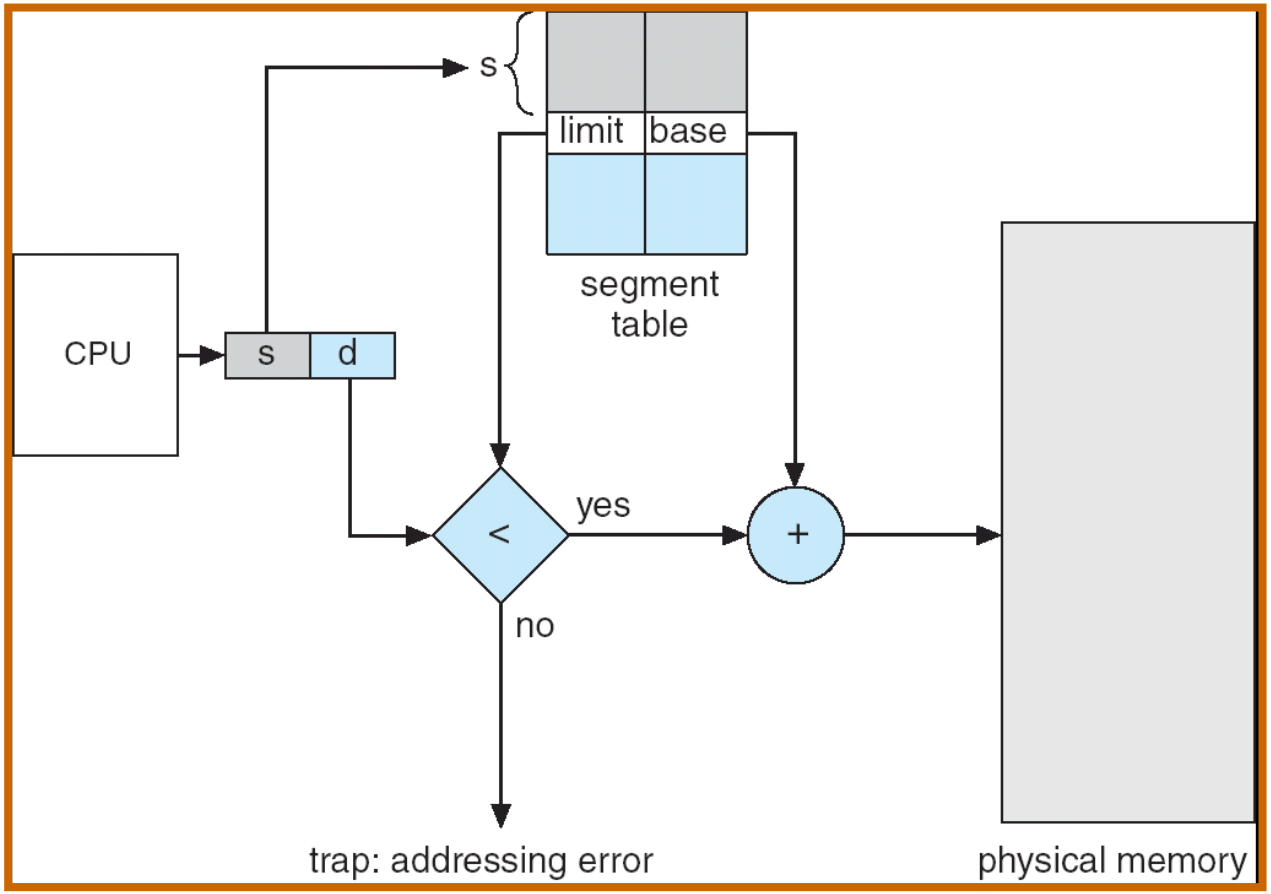


Figura 12. Hardware para mapeamento do segmento

Segmentação

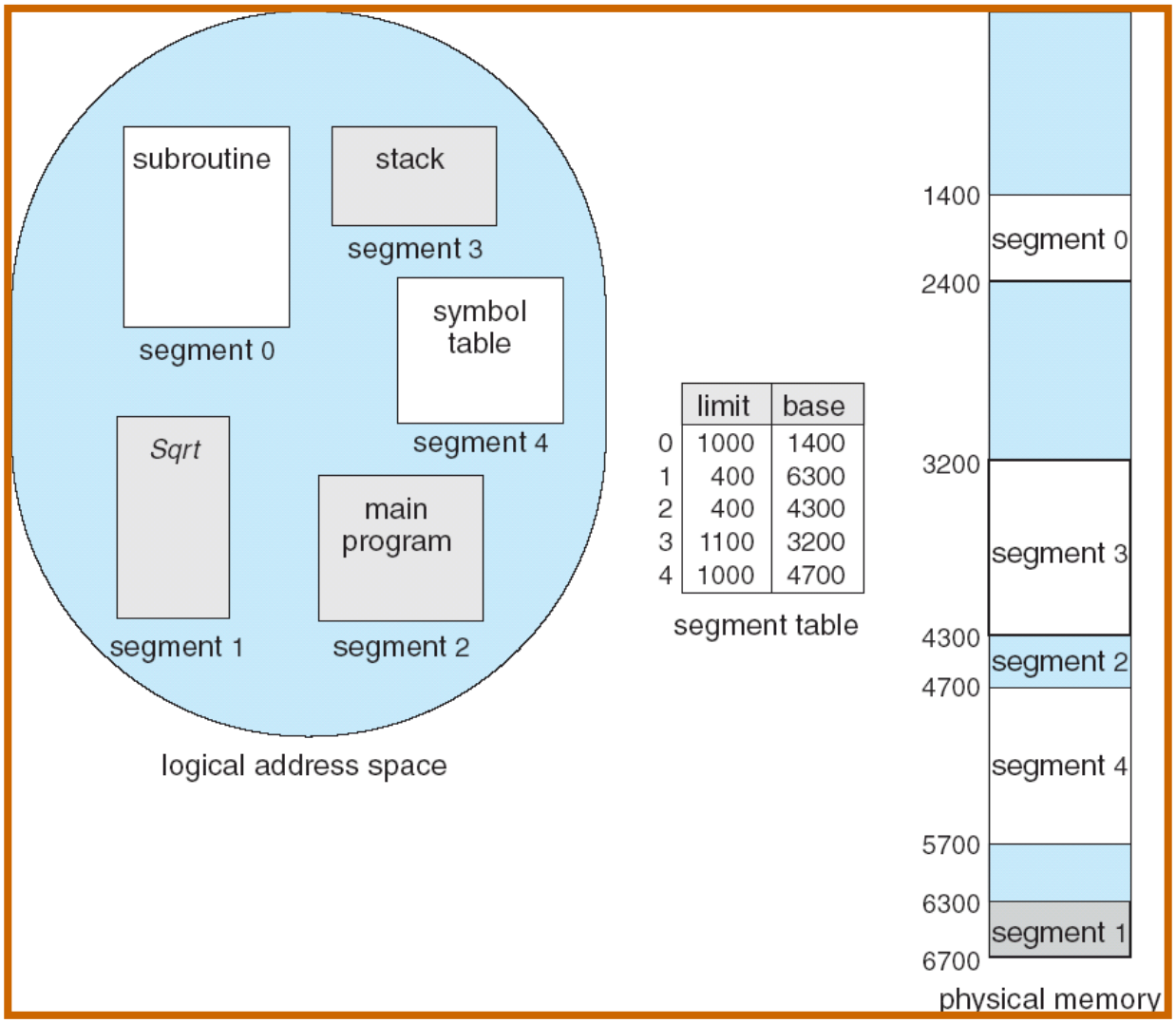


Figura 13. Divisão do programa de usuário na memória

Alocação Não Contígua: Paginação

- O processo é alocado na memória onde há espaço disponível com tamanhos determinados;
- Divide a memória física com tamanho de blocos fixos denominado quadros (definidos pelo hardware).
- Também divide a memória lógica em blocos do mesmo tamanho chamado de página.
- Para executar um programa, com tamanho de n páginas, é necessário encontrar n páginas livre para carregá-lo.
- Configura uma tabela de páginas a cada processo para traduzir o endereço lógico em endereço físico.

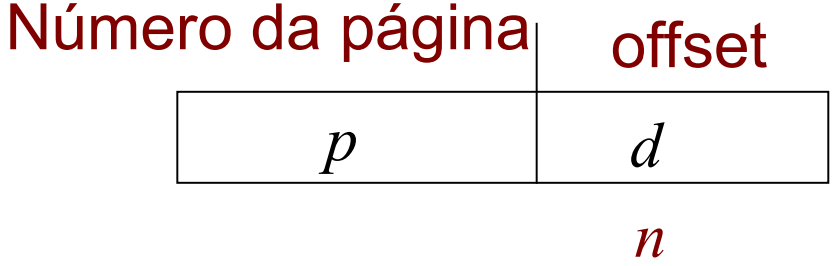
Paginação: Alocação Não Contígua

O endereço gerado pela CPU é dividido em:

Número de Página (*p*) – usado como um indice dentro de uma tabela de páginas, a qual contém o endereço base de cada página na memória física;

Página offset – deslocamento (d) – combinada com endereço base define o endereço de memória física que é enviado para unidade de memória;

As consultas acontecem analisando essas informações:



Paginação: Alocação Não Contígua

O endereço gerado pela CPU

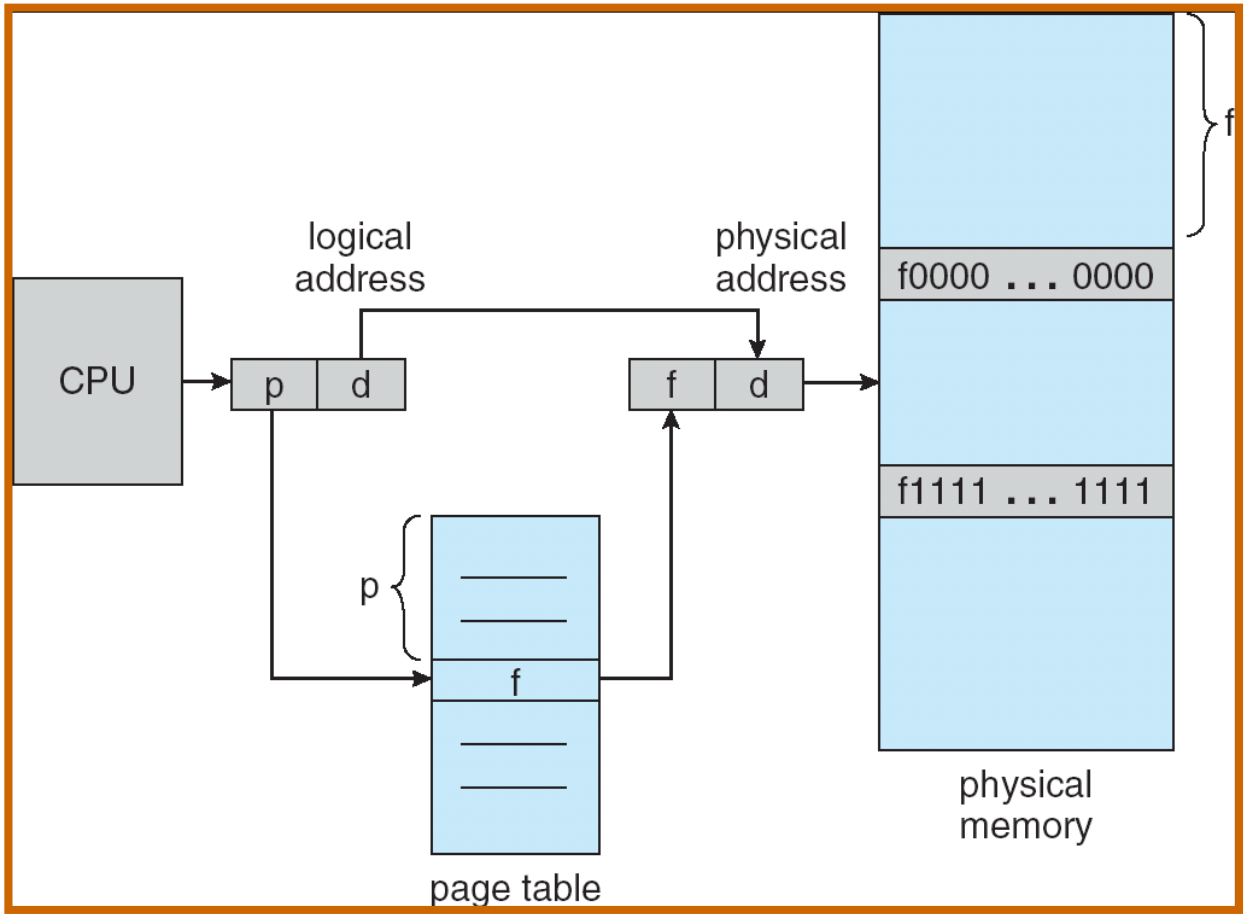


Figura 14. Hardware para o mapeamento da paginação

Paginação: Alocação Não Contígua

O endereço gerado pela CPU deve acessar a tabela de página

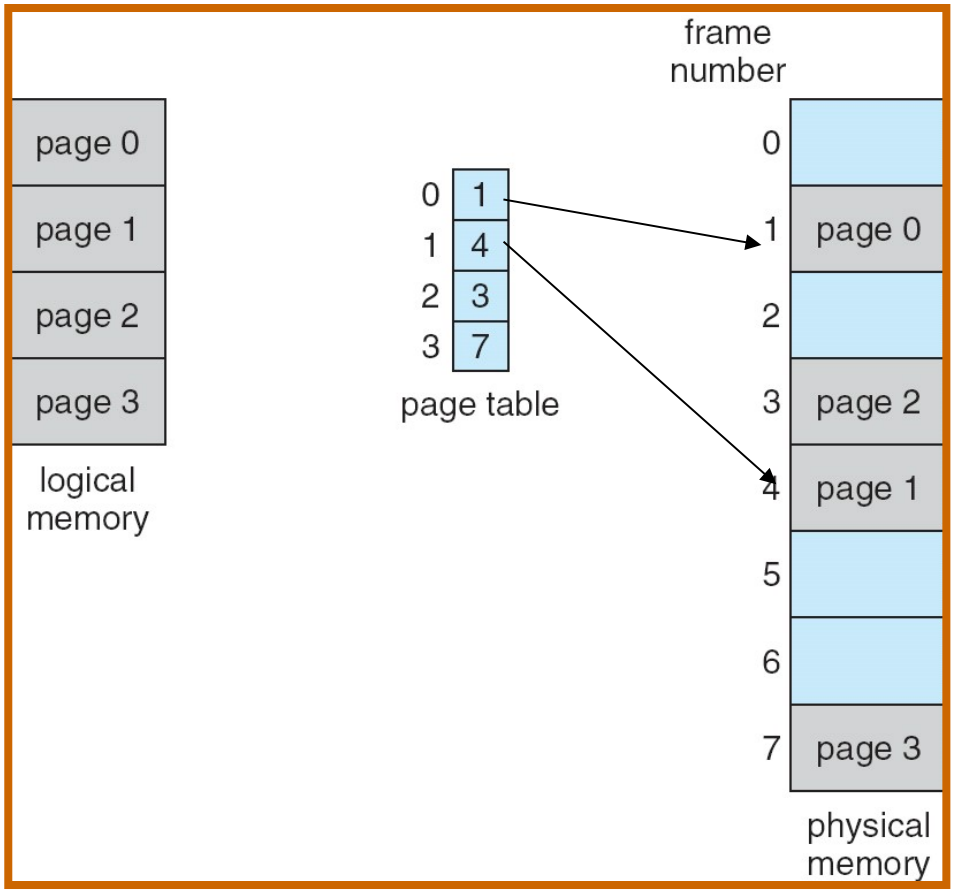


Figura 15. Modelo de paginação de memória lógica e física

Exemplo de Paginação

- **Tamanho da página:** 4 bytes
- **Tamanho da memória Física:**
- 8 páginas = 32 bytes

Acesso ao endereço lógico 0:

- Página = 0 / 4 = 0,
- Offset = 0%4 = 0.
- **Tabela**: quadro 5 e offset 0 -
- endereço físico 20

Acesso ao endereço lógico 13:

- Página = 13 / 4 = 3,
- Offset = 13%4 = 1.
- **Tablela:** quadro 2 + offset 1- endereço físico 9

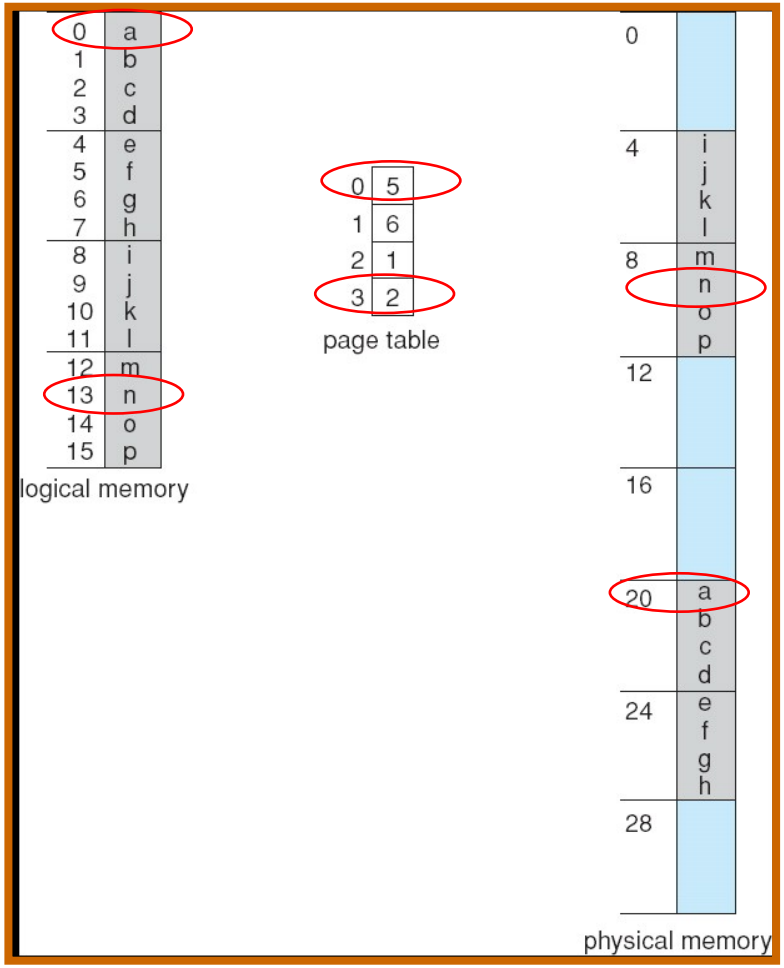


Figura 16. Exemplo de paginação para acesso aos endereços.

Quadros Livres

Cada processo tem sua tabela de páginas

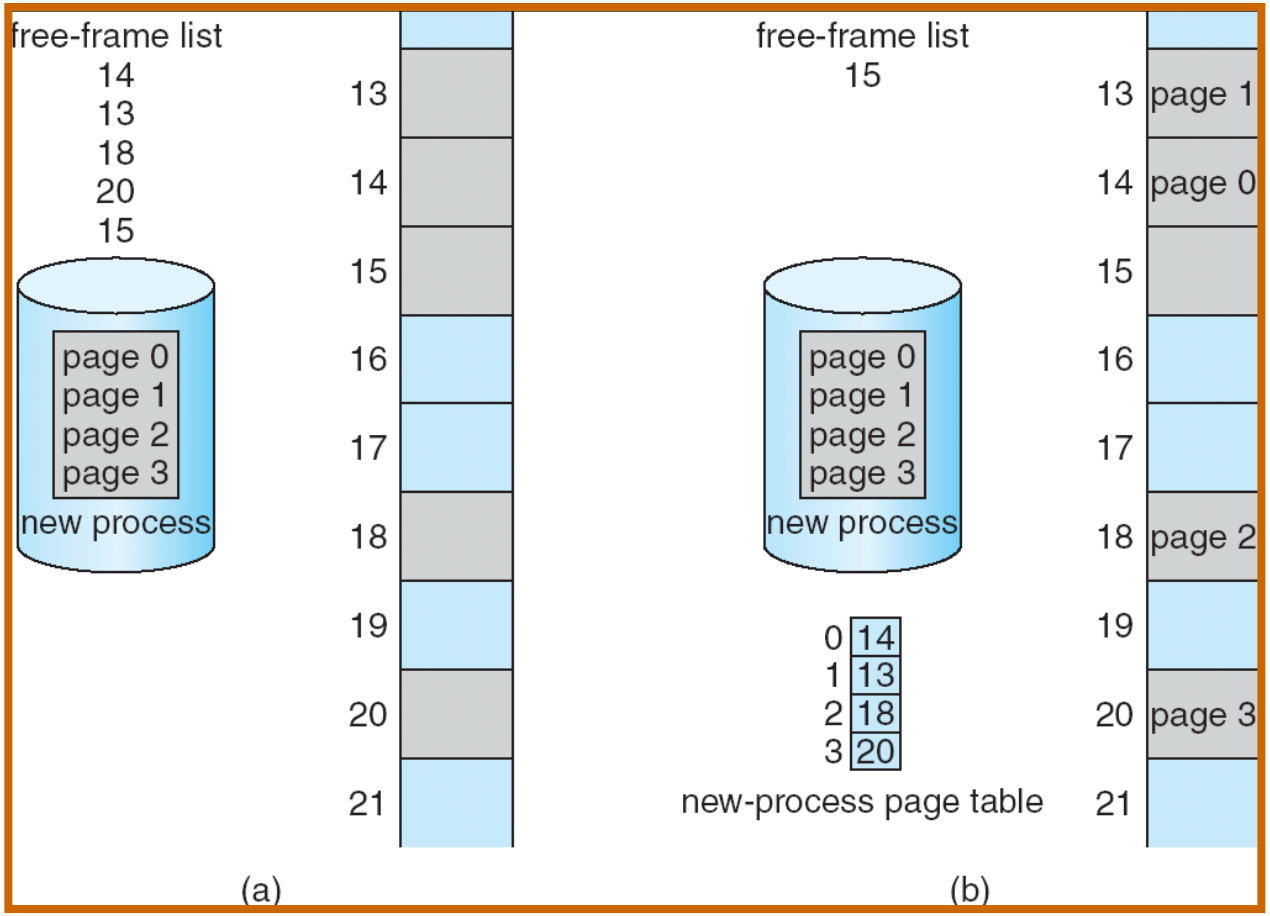


Figura 17. Alocação de um processo baseado em paginação: (a) início; (b) alocado.

Paginação: Proteção

- Proteção de memória é realizada pela proteção dos bits associadas a cada quadro.
- Esses bits costumam ser mantidos na tabela de páginas
- Um bit pode definir uma página como sendo de leitura/escrita ou somente leitura
- Outro bit (válido-inválido) pode ser usado
  - "válido" indica onde a página esta no espaço do endereço do processo, isto é, uma página válida para acesso
  - "inválido" indica que a página não está no espaço de endereço do processo

Paginação: Proteção

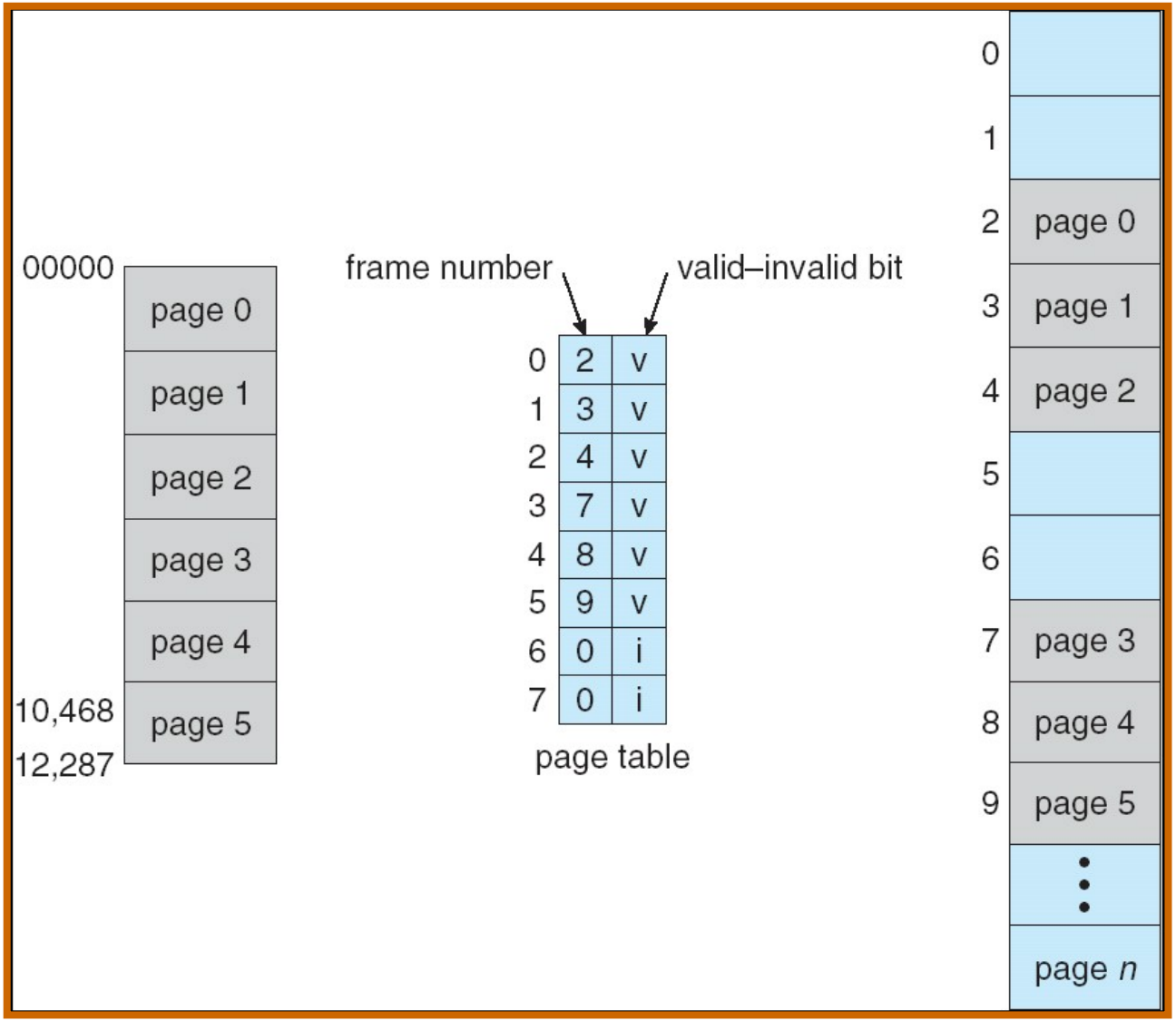


Figura 18. Bit válido ou inválido em uma tabela de páginas

Paginação: Implementação

- Tabela de página é guardada na memória principal
  - Registrador de base da tabela de páginas (page-table length register - PTBR) aponta para a tabela;
  - Page-table length register (PRLR): indica o tamanho da tabela de página.
- Qual a desvantagem de guardar a tabela de páginas na memória?
  - Todo **acesso as instruções** deve ocorrer 2 acessos a memória;
  - Um para tabela de página e outro para as instruções.
- Como isso pode ser melhorado?

Paginação: Implementação

- Solução: Usar uma **cache** especial, menor, de pesquisa rápida chamada de **buffer paralelo de conversão** (Translation Lookaside Buffer -TLB)

- É uma memória associativa de alta velocidade;

- Cada entrada da TLB consiste em duas partes:

  - Uma chave e um valor.

- Quando recebe um item, o item é comparado com todas as chaves:

  - Exemplo: Intel Pentium Core i7 - com 512 entradas

Paginação: Implementação

Solução: Translation Lookaside Buffer (TLB)

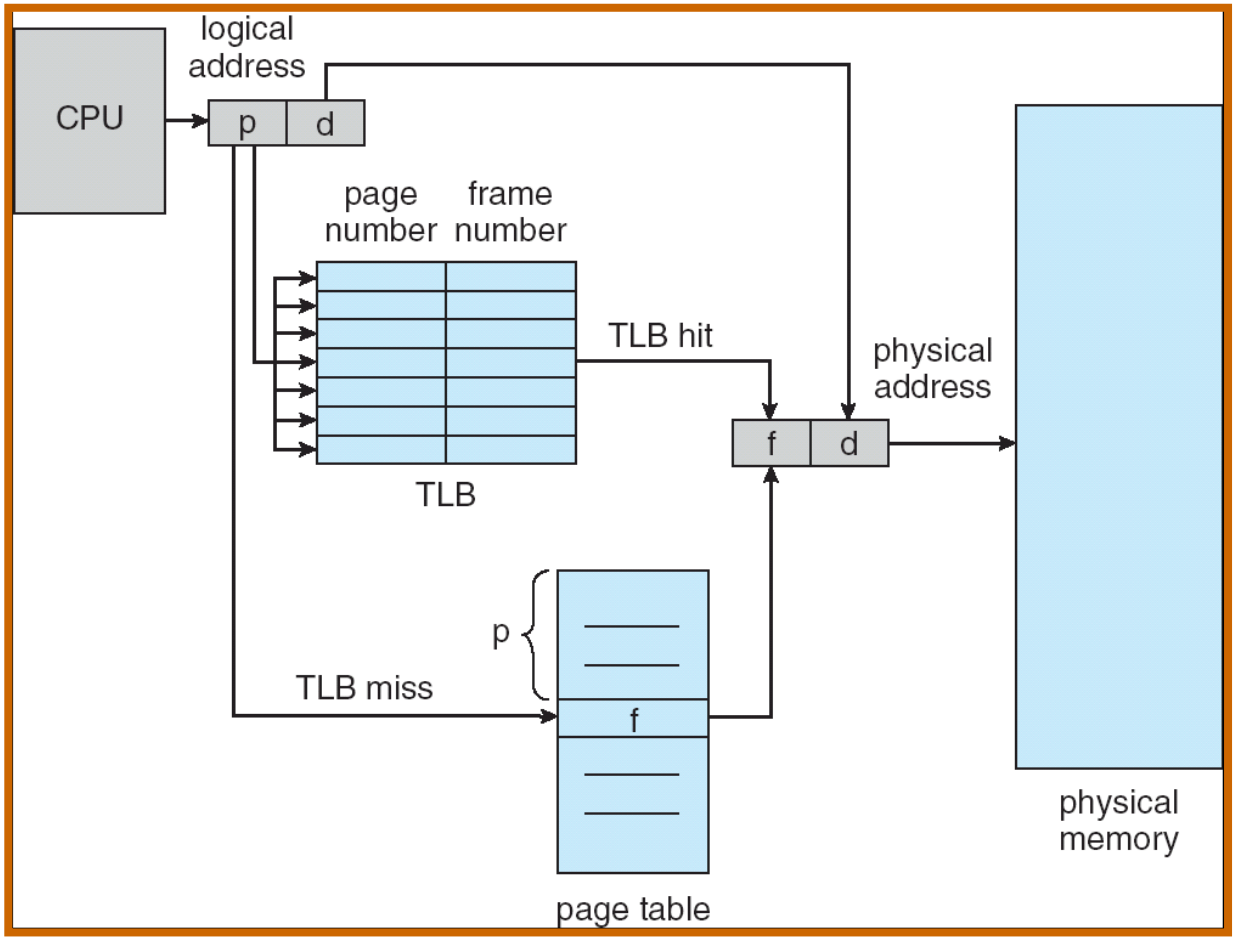


Figura 19. Hardware de paginação com TLB

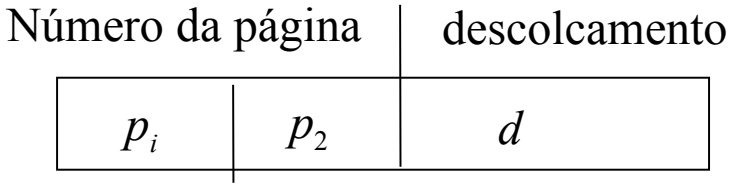
Paginação: Estrutura da tabela de página

A maioria dos sistemas computacionais modernos admite um grande espaço de endereço;

Nesse contexto, a própria tabela de página se torna excessivamente grande para buscas;

Uma solução é usar um algoritmo de paginação com dois níveis, em que a própria tabela de página também é paginada:

- Exemplo: computador com 32 bits: 20 bits ficam para páginas e 12 para deslocamento



Paginação: Estrutura da tabela de página

A maioria dos sistemas computacionais modernos admite um grande espaço de endereço;

Em 32 bits de endereço:

- Páginas de 4 KB
- 20 primeiros bits indicam a página
- 12 últimos bits indicam o deslocamento dentro da página

- Veja o código pagesize.c

Paginação: Estrutura da tabela de página

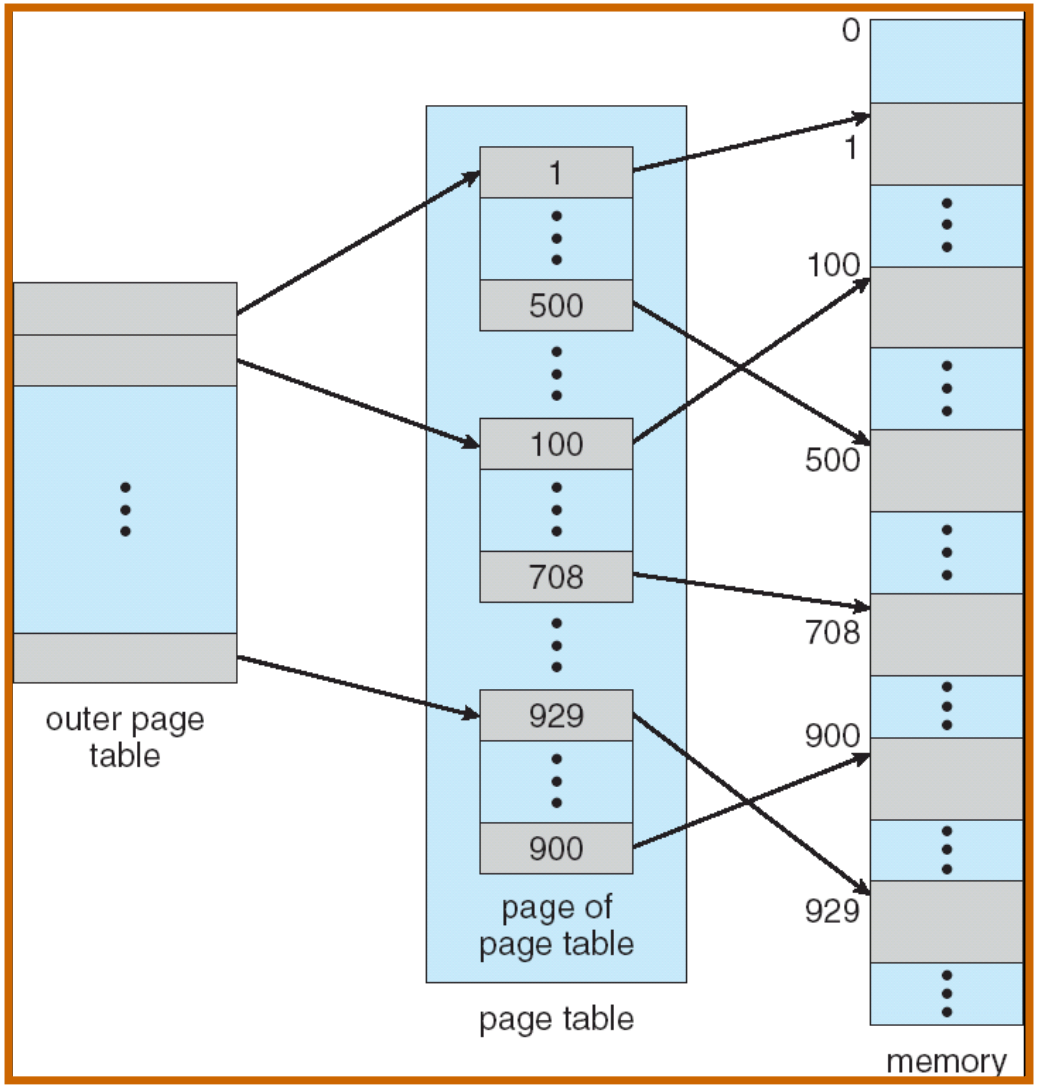


Figura 20. Tabela de páginas em 2 níveis

Paginação: Estrutura da tabela de página

- O S.O. cria uma tabela de página externa e tem-se a página da tabela de página.

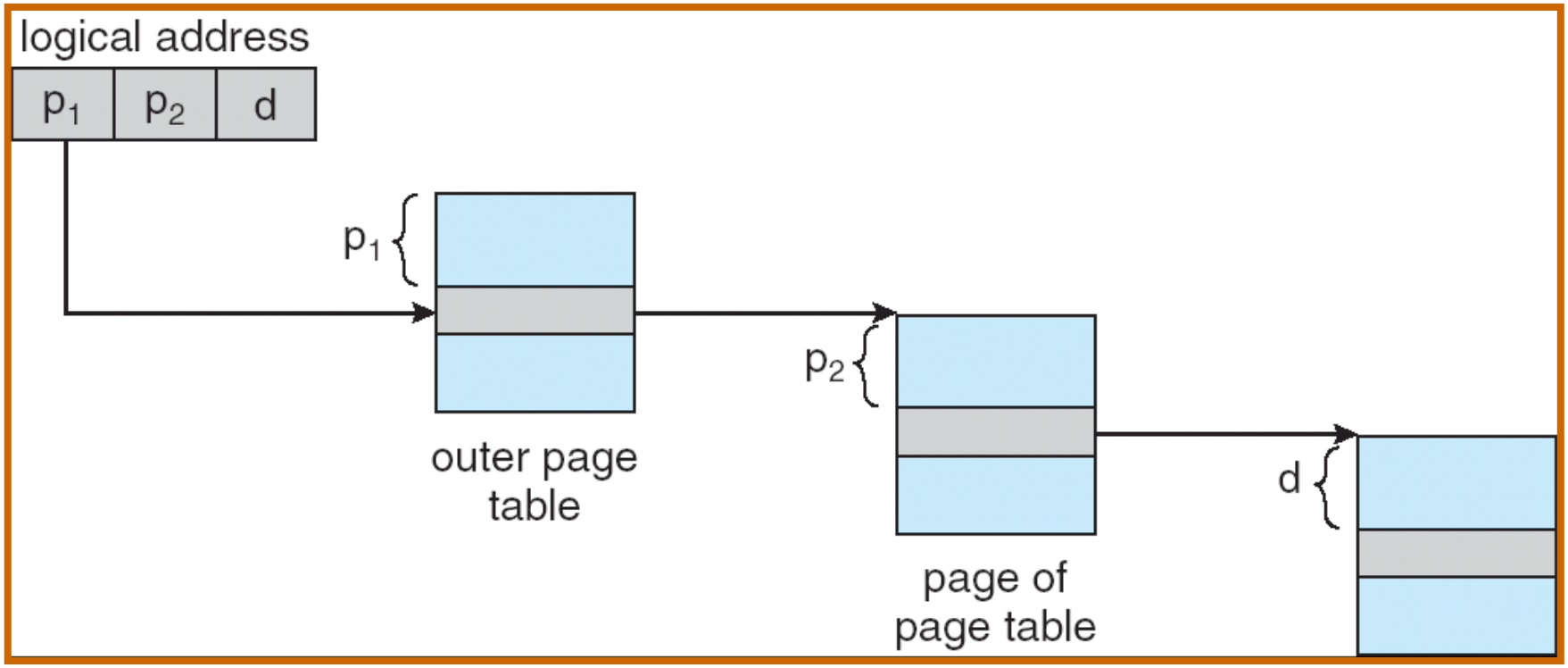


Figura 21. Conversão do endereços de uma arquitetura de 32 bits em 2 níveis

Paginação: Estrutura da tabela de página

- Em um espaço de endereçamento de 64 bits, o esquema de endereço em 2 níveis não é mais adequado.

- Então, emprega-se um esquema de 3 níveis.

outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

Figura 23. Relação entre tabela de páginas de 32 bits e 64 bits.

Paginação: Tabela de Páginas Invertidas

- A entrada consiste no endereço virtual da página armazenado nesse local da memória física com informações sobre o precesso que possui essa página. Composto por tripla:
  - <id do processo, número da página, deslocamento>

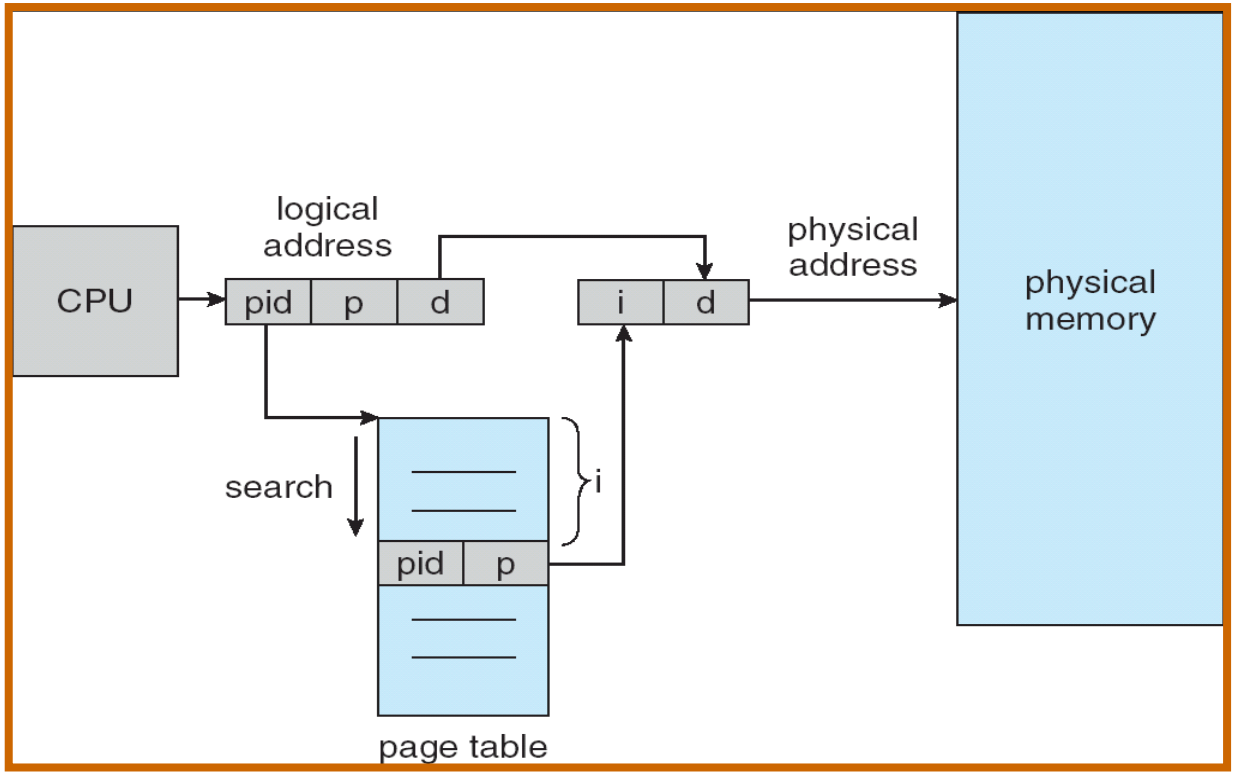


Figura 23. Tabela de páginas invertidas.

45

Exemplo: Intel 32-bits e 64 bits

- A CPU gera os endereços lógicos, que são os dados à unidade de segmentação;
- A unidade produz um endereço linear para cada endereço lógico;
- Então, dado à unidade de paginação, que por sua vez gera o endereço físico na memória;

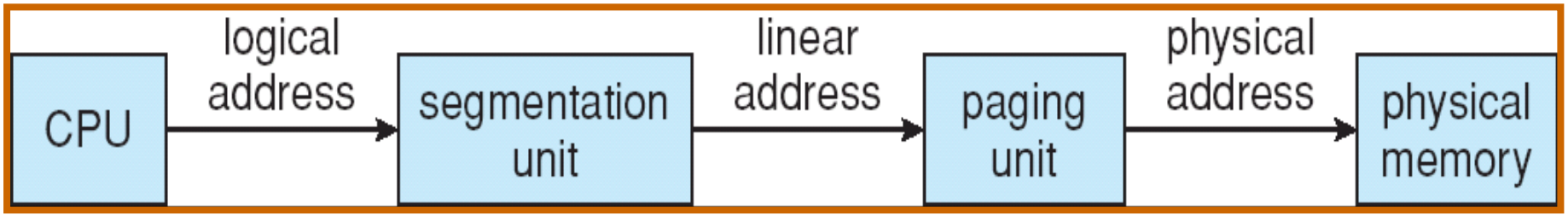
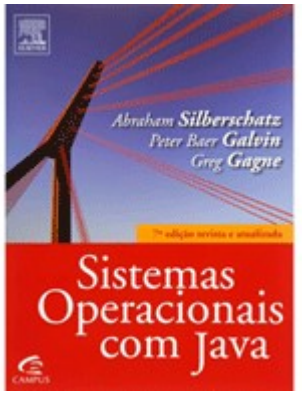


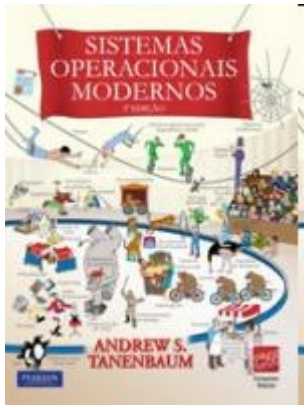
Figura 24. Conversão dos endereços lógicos em físicos no Pentium

Leituras Sugeridas

- Silberschatz, A., Galvin, P. B. Gagne, G. Sistemas Operacionais com Java. 7º edição. Editora Campus, 2008.
  - Capítulo 7.



- TANENBAUM, A. Sistemas Operacionais Modernos. Rio de Janeiro: Pearson, 3 ed. 2010
  - Capítulo 3.



Exercícios

- Lista de Exercícios de Memória no Moodle
- Exercícios de 1 até 21