Organização e Arquitetura de Computadores

Gerência de Memória

Alexandre Amory

Edson Moreno

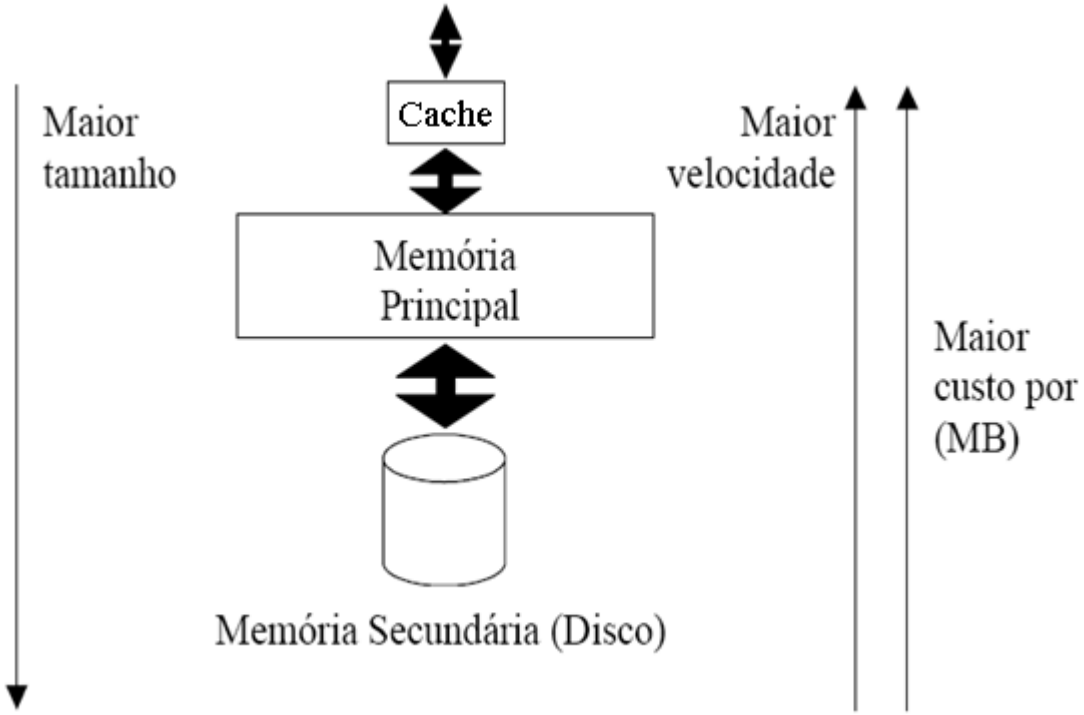
Índice

Introdução

- Sistema monoprogramado
  - Memória principal dividida em duas partes
    - Uma para SO
    - Outra para programa em execução
- Sistema multiprogramado
  - Memória principal dividida entre vários processos
- SO divide memória dinamicamente
  - Gerência de memória
- Gerência de memória eficiente é vital para sistemas multiprogramados
  - Motivo
    - Se poucos processos estiverem na memória, processador ficará parado esperando por operações de E/S
  - Solução
    - Técnica de gerência aumenta número de processos na memória, aumentando utilização do processador

Hierarquia

- Memória principal (MP) é mais um nível da hierarquia de memória
  - Princípio da gerência igual a outros níveis de memória
    - Dados mais usados são trazidos para MP para diminuir tempo médio de acesso ao nível mais baixo, neste caso o disco



Histórico e Técnicas de Gerência

- Memórias RAM
  - Eram empregada como área temporária para acelerar o acesso aos dados
- Recurso caro na época
- Tamanho reduzido (poucos Kbytes)
  - Muitos programas não cabiam na memória junto com seu ambiente de execução (interpretador, bibliotecas, etc.)
- Gerência de MP não utiliza mesmas técnicas que gerência de *caches* por 2 motivos
  - Evolução Histórica
    - Idéia de memória como área de armazenamento temporário de dados é anterior ao conceito de cache
  - Diferentes Características
    - Tamanho e tempo de acesso da MP são muito maiores que das *caches*
    - Ao contrário das *caches* parte da gerência pode ser feita em SW
    - Unidades de gerência possuem a identificação do processo dono

Gerência de Memória (Monoprogr.)

- Objetivo
  - Permitir que programas maiores que a memória pudessem executar
- Primeiras estratégias baseadas em *overlays* (sobreposição)
  - Responsabilidade total do programador
  - Programador dividia programa em partes que podiam executar autonomamente na memória (*overlay*)
- No final dessas partes era colocado código responsável pela carga da próxima parte que iria sobrepor
  - Endereços antigos não são mais necessários
- Programador tinha controle total da memória
  - Responsável pela troca das partes
- Programa escrito de forma a evitar quebras em muitas partes
  - A troca tinha um alto custo

Gerência de Memória (Multiprogr.)

- Multiprogramação trouxe dificuldades
  - Gerência de overlays
    - Deve possibilitar vários programas executando concorrentemente
    - Execução mesmo quando a soma do programa não coubessem na memória
  - Gerência de overlays de cada programa
    - Não podia interferir na dos outros programas
    - Necessário que agente externo seja responsável pela gerência
    - Responsabilidade de gerenciar memória passou do usuário para monitor residente, mas tarde chamado de SO

Índice

Endereçamento

- Problema
  - Como endereços de um programa são usados para acessar posições da MP?
- Solução
  - Modos de endereçamento
- Endereçamento Contíguo
  - Programa é carregado inteiro em uma única área de memória contígua
- Endereçamento Não-Contíguo
  - Quebra programa em pedaços carregados em áreas distintas de memória
  - Vantagens
    - Não há necessidade de respeitar qualquer ordem
    - Melhora aproveitamento da memória → menor fragmentação externa pelo aproveitamento de lacunas
  - Desvantagens
    - Gerência de memória fica mais trabalhosa

Índice

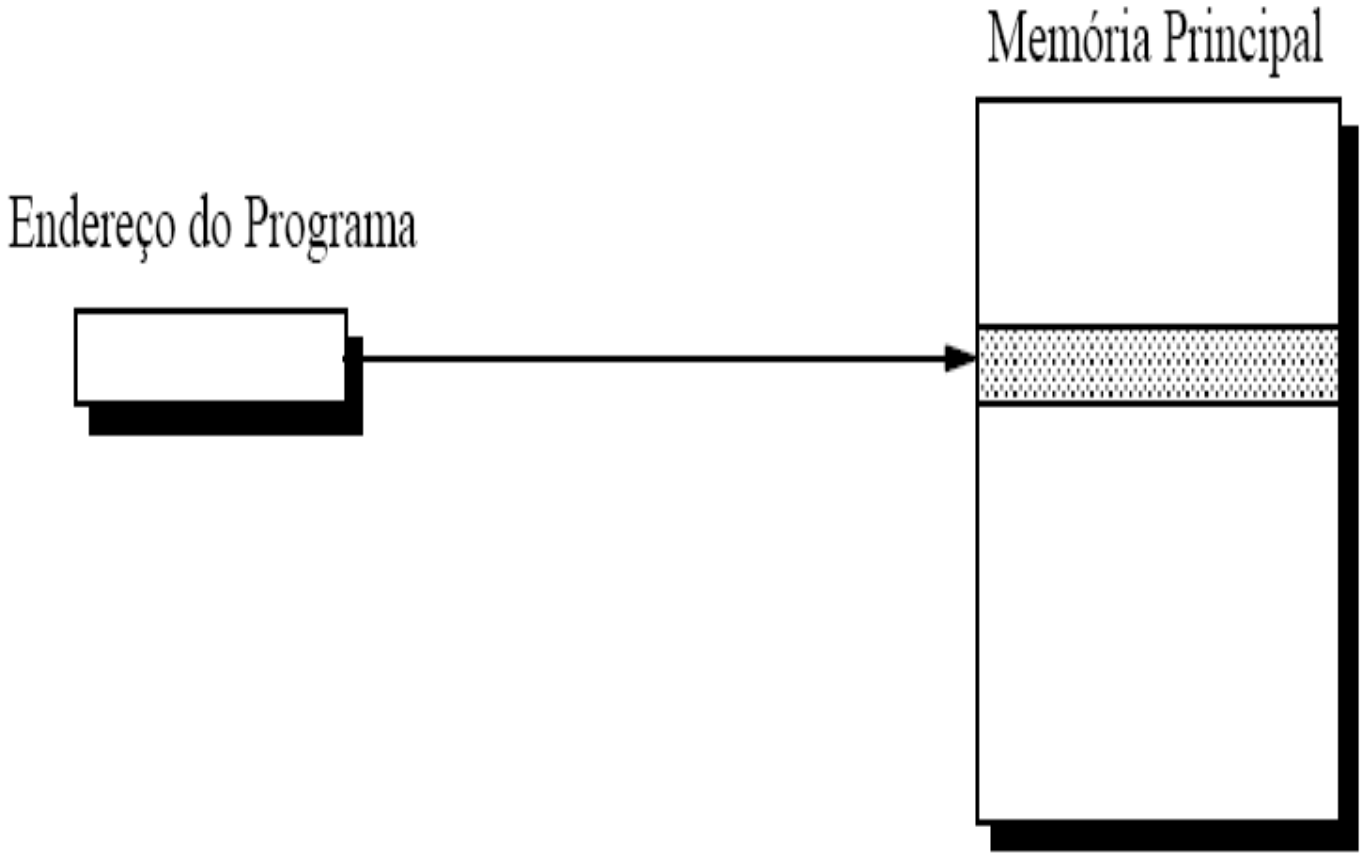
Endereçamento Contíguo

- Existem duas formas de endereçamento
  - Endereçamento direto
  - Endereçamento relativo
- Endereçamento Direto
  - Endereços do programa são usados diretamente no acesso à memória principal
  - Endereços são definidos durante a compilação/ligação ou carga
- Endereçamento Relativo
  - Endereços do programa são relativos, devendo ser definidos em tempo de execução
  - Endereços são definidos na hora do acesso

Índice

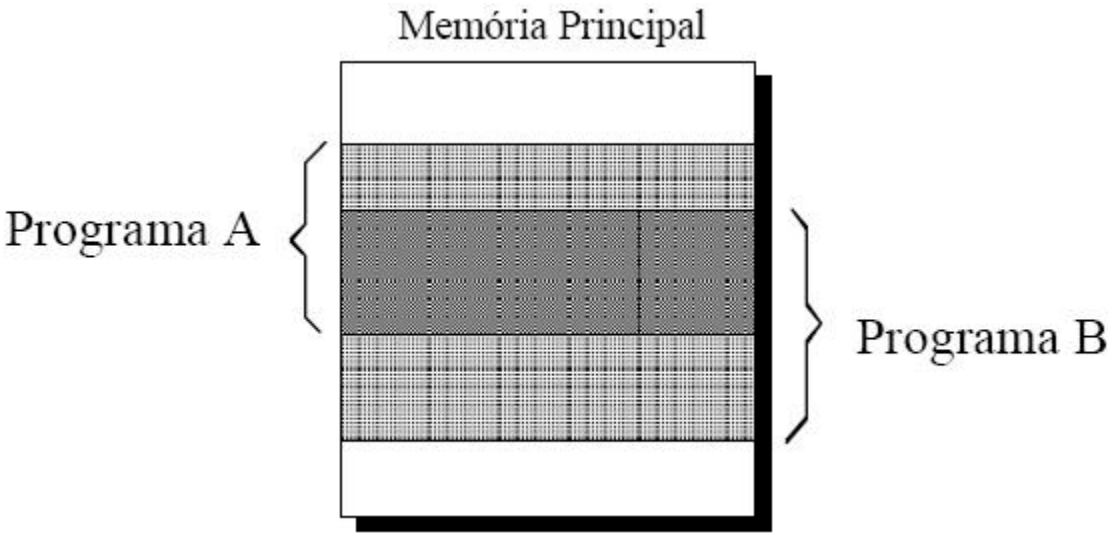
Endereçamento Contíguo Direto

- Endereço aponta para posição na memória que está definida e não pode ser alterada



Endereçamento Contíguo Direto

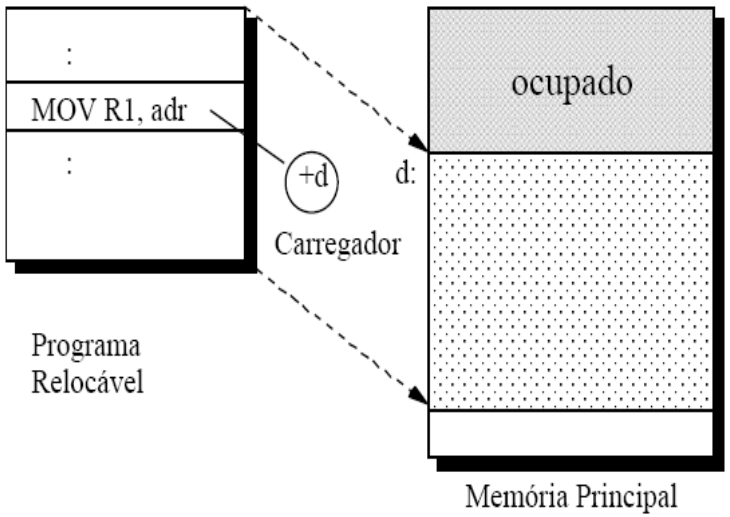
- Na multiprogramação
  - Endereçamento direto pode gerar conflitos no acesso à memória
  - Posição do programa na memória definida sem conhecimento do que estava alocado



- Programas só podem ser carregados na memória se seus espaços de endereçamento são disjuntos (um espaço específico para cada programa)

Endereçamento Contíguo Direto

- Multiprogramação - geração do endereço na carga
  - Espaço de endereçamento de um programa inicia sempre em 0
    - Facilmente realocável na momento da carga

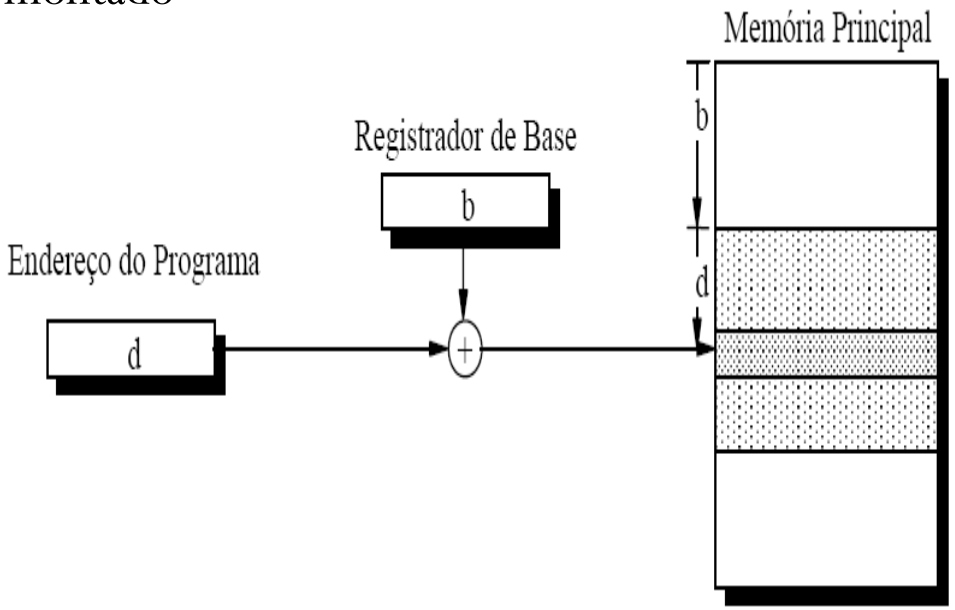


  - Problemas
    - A conversão dos endereços pode atrasar consideravelmente a operação de carga
    - O programa não pode ser mudado de lugar durante a sua execução

Índice

Endereçamento Contíguo Relativo

- Alternativa mais flexível
  - Endereço montado somente no acesso
  - Espaço de endereçamento lógico inicia em 0
  - Processador possui registradores de endereçamento base
    - Contém base do endereço físico dos programas
  - Endereço real a ser acessado é montado
    - com a base somada ao endereço relativo do programa em cada acesso à memória
  - Programa pode ser trocado de lugar na memória em tempo de execução



Índice

Partições fixas

- Memória dividida em partições de tamanhos fixos
  - Tamanhos podem ser os mesmos ou não
- Quando programa é carregado
  - Entra em uma fila de processos para utilizar uma partição livre
- O número máximo de processos concorrentes é baseado no número de partições
- Exemplo:
  - Particionamento de uma máquina com 32 KBytes de memória
    - 4KBytes: processos pequenos
    - 6KBytes: processos médios
    - 12KBytes: processos grandes
    - 10KBytes: kernel

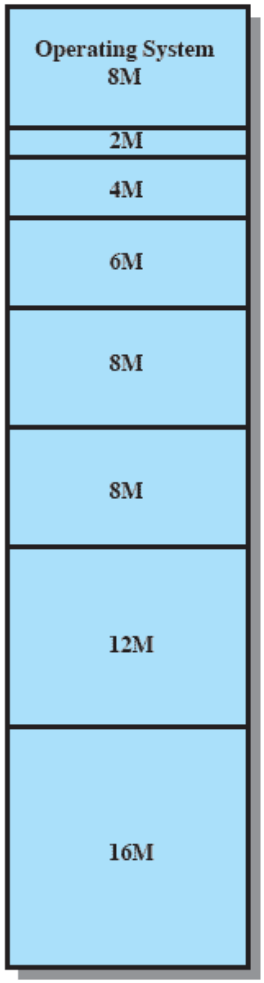
(a) Equal-size partitions

Partições fixas

- Problema:
  - Tamanho do processo a ser carregado e otamanho da partição não são equivalentes
- Consequência
  - ***Fragmentação interna:***
    - Ocorre quando o tamanho da partição é maior que o tamanho do processo
    - Espaço que sobra dentro da partição quando processo é alocado na partição

Partições fixas

- Proposta de solução para fragmentações
  - Emprego de partições de tamanhos distintos, mas fixos
- Diminui ambos problemas
  - Mas não resolve completamente
- Exemplo da figura
  - Processos de até 16M podem ser acomodados
  - Processos menores podem ser acomodados nas partições menores, reduzindo a fragmentação interna

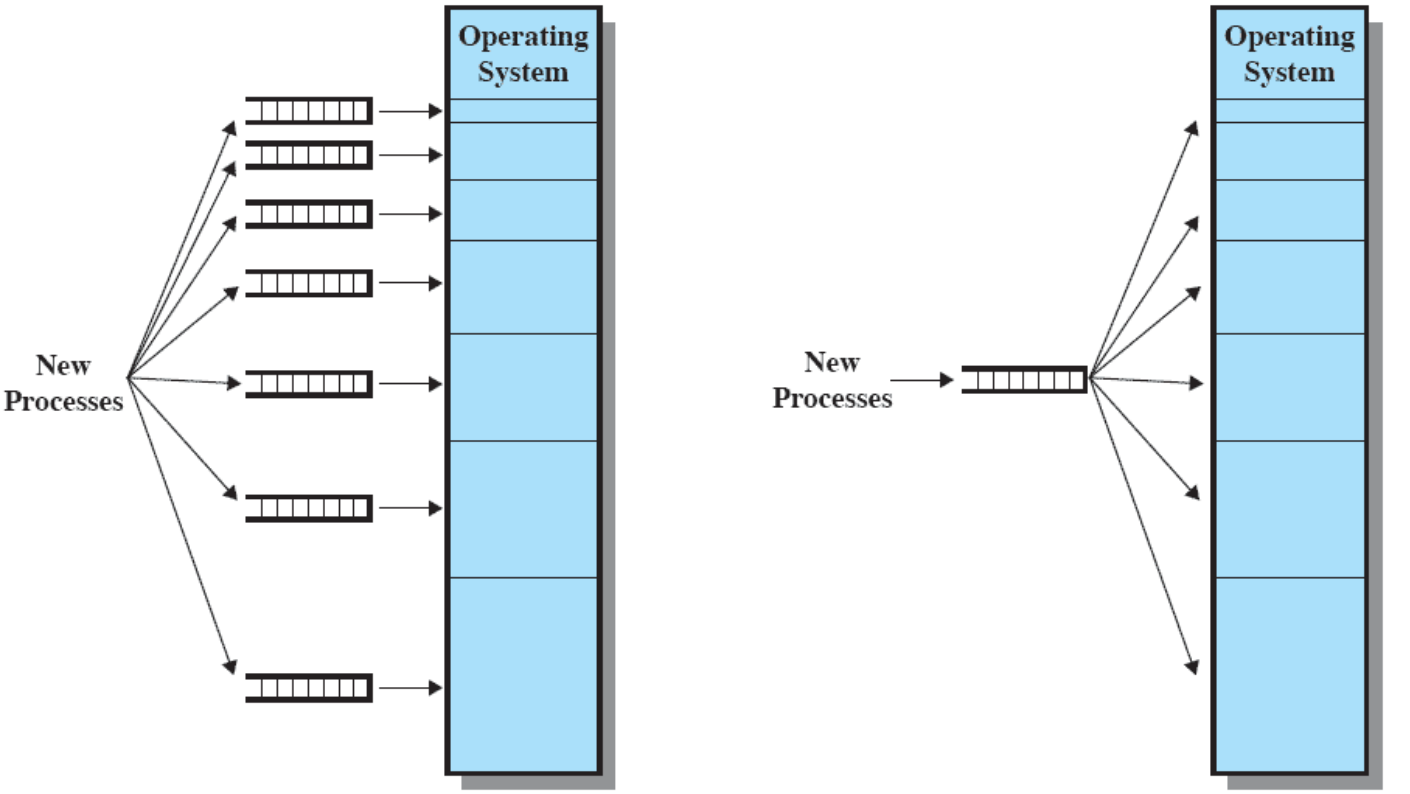


(b) Unequal-size partitions

Partições fixas

- Algoritmo de alocação
  - Partições de tamanho equivalentes
    - Trivial: Havendo espaço disponível, aloca-se para o processo em carga
  - Partições de tamanho distintos
    - Pode associar cada processo com a menor partição na qual ele cabe
      - Processos devem ser alocados de tal maneira que minimizem o desperdício de memória em cada partição
    - Pode-se empregar uma fila para cada tamanho de partição
      - Garante a redução/eliminação da ocorrência de fragmentação interna
      - Pode causar mal uso de partições
        - Partições maiores não utilizadas poderiam ser alocadas para processos menores

Partições fixas



(a) One process queue per partition

(b) Single queue

Figure 7.3 Memory Assignment for Fixed Partitioning

Índice

Partições variáveis

- ***Fragmentação Externa***
  - Memória externa a todos os processos é fragmentada
  - Pode ser resolvido com **compactação**
    - OS move os processos
    - Permite juntar áreas livres
  - Consome tempo e gasta tempo de CPU

Partições variáveis

- Alocação de partições
  - Sistema operacional deve decidir qual bloco livre será associado a um processo
  - Algoritmos de alocação
    - Best-fit
    - First-fit
    - Next-fit
    - Worst-fit

Partições variáveis

- Best-fit
  - Escolhe-se a partição onde o processo deixa o menor espaço sem utilização
    - Escolhe o bloco cujo tamanho é o mais próximo do requisitado
  - Nesse algoritmo
    - Desempenho ruim quando todos os blocos tem de ser avaliados
      - Objetivo é garantir a melhor escolha de partição livre
    - Otimização
      - A lista de áreas livres está ordenada por tamanho
      - Diminuindo o tempo de busca por uma área desocupada.
  - Desvantagem do algoritmo
    - Escolha da partição mais aproximada resulta em pequenas partições livres
    - Tendência é ter grande quantidade de pequenas áreas livres não-contíguas
    - Aumentando o problema da fragmentação.
  - Solução pode ser o emprego de Compactação de memória
    - Pode ser necessária mais frequentemente

Partições variáveis

- Worst-fit
  - Escolhe-se a partição onde o processo deixa o maior espaço sem utilização
    - Escolhe o maior espaço livre na memória
  - Nesse algoritmo
    - A lista de áreas livres deve estar ordenada por tamanho para otimizar busca
  - Comparado ao best-fit
    - Reduz (não elimina) o problema da fragmentação

Partições variáveis

- First-fit
  - Busca por espaço livre
    - Varre a memória do início
    - Escolhe o primeiro bloco disponível que seja grande o suficiente
  - Método tenta primeiro utilizar as áreas livres de endereços mais baixos
    - Boa chance de se obter uma grande partição livre nos endereços mais altos
  - Algoritmo mais rápido dos três (best / worst / first)
    - A lista de áreas livres está ordenada por endereços crescentemente
    - Consome menos recursos para a busca
- Next-fit
  - Similar ao First-fit
    - Diferença está na busca, que ocorre a partir do endereço da última posição alocada

Fragmentação de Memória

- Problema diretamente relacionado com gerência de MP
  - Fragmentação resulta em desperdício de memória
- Tipos de fragmentação
  - Fragmentação Interna
    - Quando ocorre
      - Quando usada unidade de gerência de tamanho fixo (Ex: página)
    - Como ocorre
      - Requisição não exatamente divisível pela unidade é arredondada para cima
        - Unidade alocada mas não completamente ocupada
  - Fragmentação Externa
    - Quando ocorre
      - Quando usada unidade de gerência de tamanho variável (Ex: segmento)
    - Como ocorre
      - Seqüência de alocações e liberações
  - Conseqüência
    - Requisição de usuário pode ser negada apesar de existir memória livre

Índice

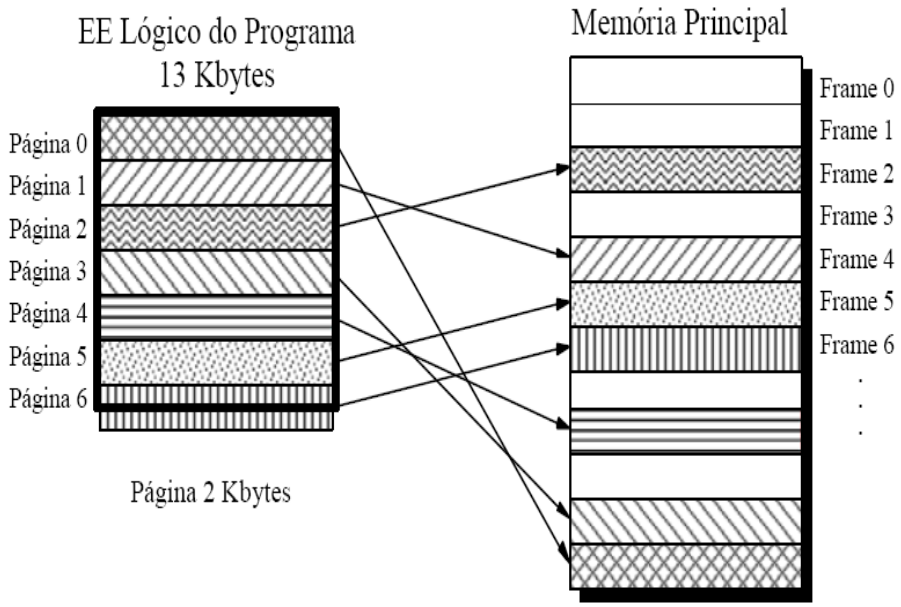
Endereçamento Não-Contíguo

- Programa é dividido em pedaços
  - Carregados em áreas distintas de memória
- Conversão dinâmica de endereços
  - Ao acessar a memória, endereços lógicos são convertidos em físicos
  - Implementada em HW (mais rápido). Exemplo: MMU (*Memory Manager Unit*)
- Três formas de endereçamento
  - Paginação
    - Unidade de gerência de memória física quebrada em *frames* de tamanho fixo
  - Segmentação
    - Unidade de gerência de memória física quebrada em segmentos de tamanho variável
  - Segmento-paginação
    - Combinação dos endereçamentos acima

Índice

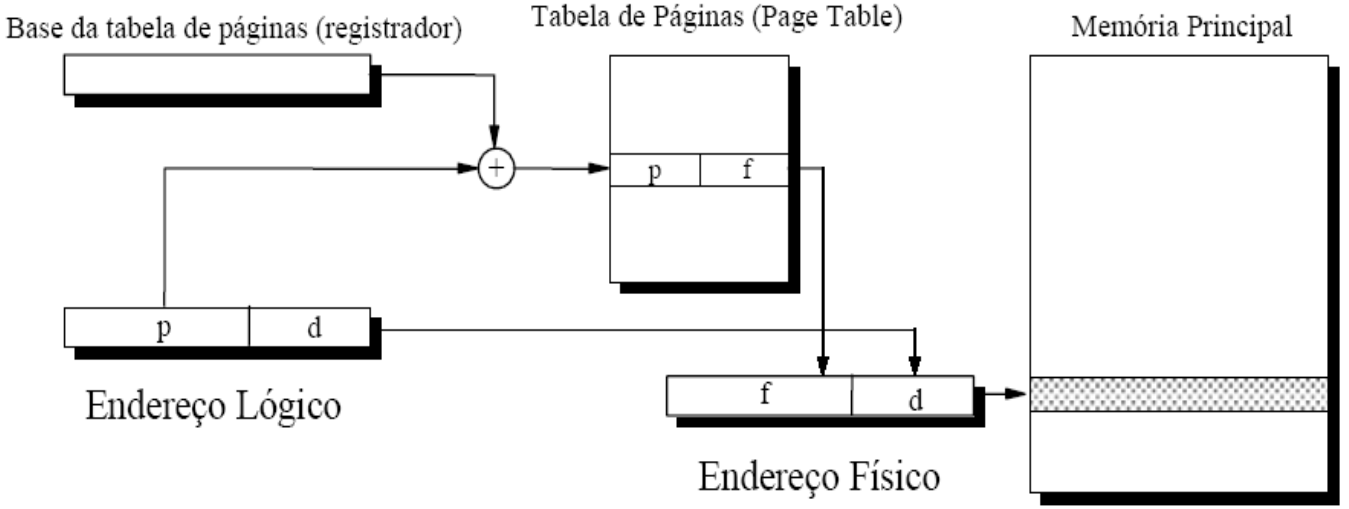
Endereçamento Não-Contíguo Paginado

- *Frames* com tamanho fixo (2k, 4k, ou 8k)
- Endereçamento lógico do programa
  - Quebrado em páginas do mesmo tamanho dos *frames*
- Quando processo é executado
  - Páginas são carregadas para *frames* livres da memória
  - Qualquer página em qualquer frame livre



Endereçamento Não-Contíguo Paginado

- Conversão de endereços
  - Bits usados para ***d*** determinam tamanho dos frames
  - Bits usados para ***p*** determinam número máximo de páginas de um processo
  - Bits de ***f*** determinam total de frames da MP
    - ***f*** é obtido com a divisão da MP pela página
    - Exemplo:
      - Para uma MP de 4 Gbyte (2^{32}) e páginas de 1 Kbyte(2^{10} → p=10 bits)
      - Resultam 4 Mega frames na MP ((2^{32})/(2^{10}))= (2^{22}). ***--22 bits para f --***



Endereçamento Não-Contíguo Paginado

- Conversão de endereços
  - Unidade de gerência de tamanho fixo
    - Gera fragmentação interna
  - Tabela de páginas
    - Converte páginas em frames (1 tabela por processo)
  - Tabela de páginas pode ser armazenada em
    - Registradores
      - Rápido mas limita tamanho da tabela
    - Memória principal (área do sistema)
      - lento, são necessários dois acessos (1 tabela + 1 dado)
      - Pode ter tamanho ilimitado
    - TLB (Translation Lookaside Buffer)
      - Área de memória associativa usada como cache para as conversões mais efetuadas
      - Situa-se entre processador e cache

Endereçamento Não-Contíguo Paginado

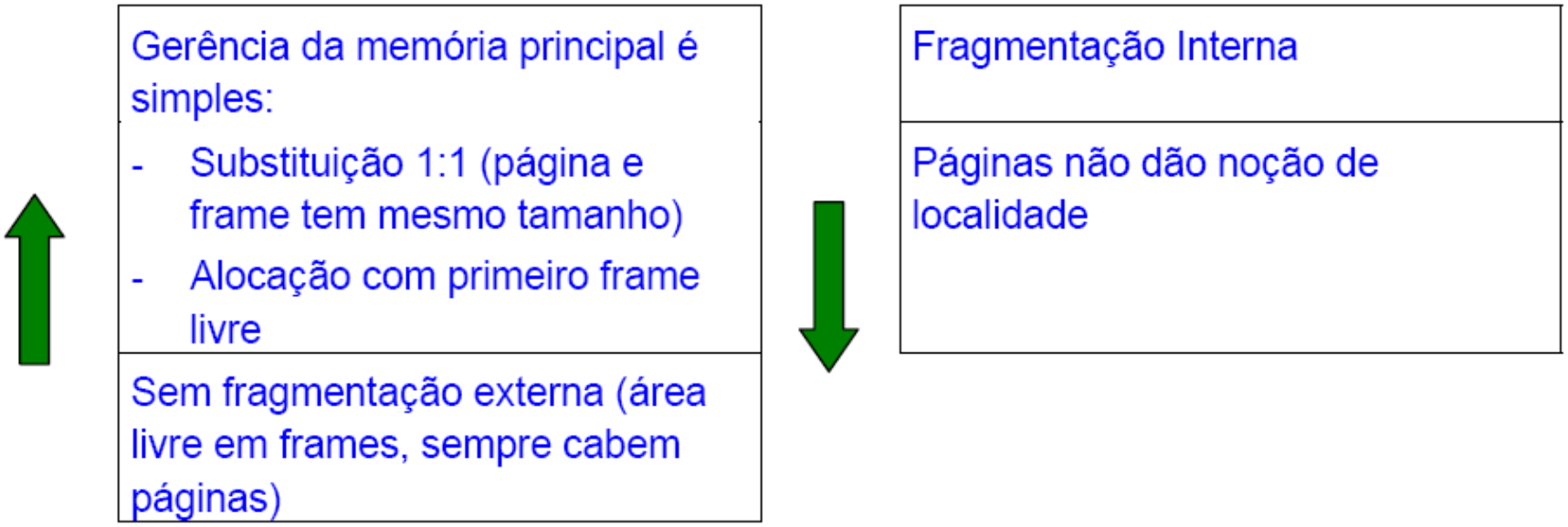
- Conversão de endereços
  - Tamanho da página
    - Pequena
      - Menor fragmentação interna, tabela de páginas fica maior
    - Grande
      - Maior fragmentação interna, tabela de páginas fica menor
  - Tabela de Frames (*Frame Table*)
    - Controle dos frames livres (para alocação e substituição de páginas)

Endereçamento Não-Contíguo Paginado

- Tabela de frames para todo sistema
  - Tamanho fixo
    - número de frames da MP é conhecido
    - tamanho da memória / tamanho da página
  - Procura por first-"found" (qualquer lacuna serve para qualquer página)
  - Possui campos adicionais para controle da política de troca de páginas (LRU, LFU)
- Quando processos terminam
  - Páginas na tabela de frames são marcadas como livres
- É possível compartilhar frames entre vários processos (leitura)
  - Fácil de implementar com várias tabelas de página apontando para mesmo frame
  - Controle adicional
    - Verificar quando frame compartilhado pode ser desalocado

Endereçamento Não-Contíguo Paginado

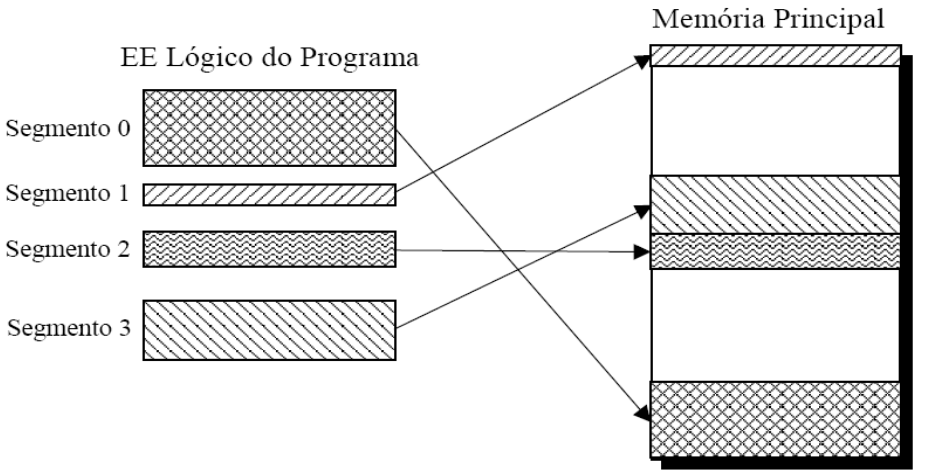
- Vantagens / Desvantagens



Índice

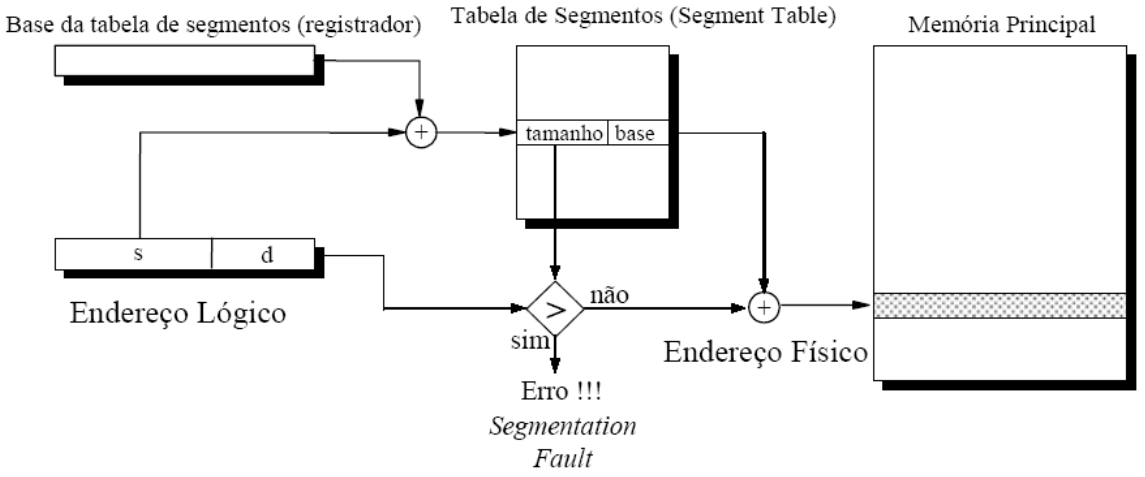
Endereçamento Não-Contíguo Segmentado

- Não divide memória física
  - Pode alocar unidade em qualquer posição
- Unidade de gerência com tamanho variável denominada segmento
  - Segmento definido pelo usuário ou compilador
    - Endereços de um mesmo segmento estão relacionados
    - Gerência de memória mais preocupada com a visão do usuário
    - Exemplo de segmentos: dados, código, pilha
- Quando processo é executado
  - Todos seus segmentos são carregados em memória
  - Ocupam qualquer posição livre



Endereçamento Não-Contíguo Segmentado (Conversão de endereços)

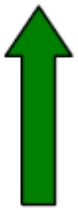
- Número de bits de d
  - Determina tamanho máximo dos segmentos
- Número de bits de s
  - Determina número máximo de segmentos de um processo
- Deslocamento do endereço lógico é comparado com tamanho do segmento
  - Tabela de segmentos → Evita invasões de segmentos vizinhos
  - Este teste não era necessário no endereçamento paginado pela limitação do número de bits



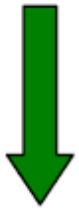
Endereçamento Não-Contíguo Segmentado (Conversão de endereços)

- Unidade de gerência variável
  - Gera fragmentação externa
  - Compactação da memória necessária devido alta fragmentação externa
- É possível compartilhar segmentos entre vários processos (leitura)
  - Fácil implementar com várias tabelas de segmento apontando para mesma área de memória
- Tabela de Segmentos
  - Aponta base do segmento na MP e verifica se deslocamento está dentro do limite do segmento
  - Tabela de segmentos pode ser armazenada igual a paginação
    - Registradores, MP, TLB
- Tabela de Alocação
  - Controla as lacunas livres
  - Uma tabela de alocação para todo o sistema de tamanho variável
    - Número de áreas da MP é inicialmente 1 (toda a memória)
    - Ao longo da gerência pode variar com a inclusão de áreas ocupadas
    - Liberação de segmentos pode resultar na junção de vários segmentos e na diminuição do número de entradas da tabela

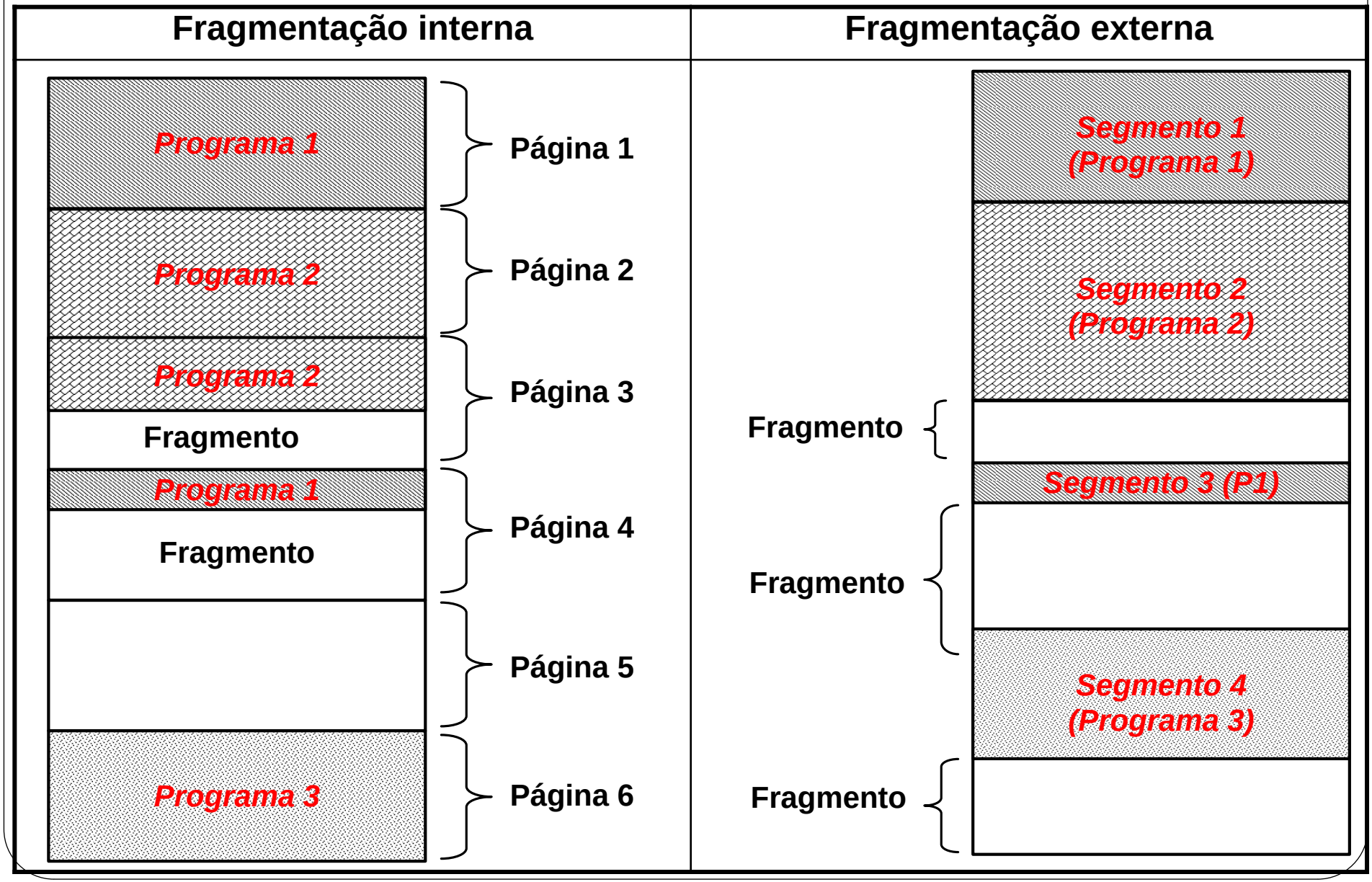
Endereçamento Não-Contíguo Segmentado (Vantagens e desvantagens)



Maior localidade na unidade de gerência (dados dentro do segmento estão relacionados)	Fragmentação externa
	Gerência da memória bem mais complexa que na paginação por causa da unidade variável – Gerência da tabela de alocação – Alocação e liberação de páginas – Verificação de *segmentation fault*
Sem fragmentação interna (unidade de tamanho variável)	Compactação da memória pode ser necessária (alto custo)



Fragmentação de Memória



Índice

Endereçamento Não-Contíguo Segmento-Paginado

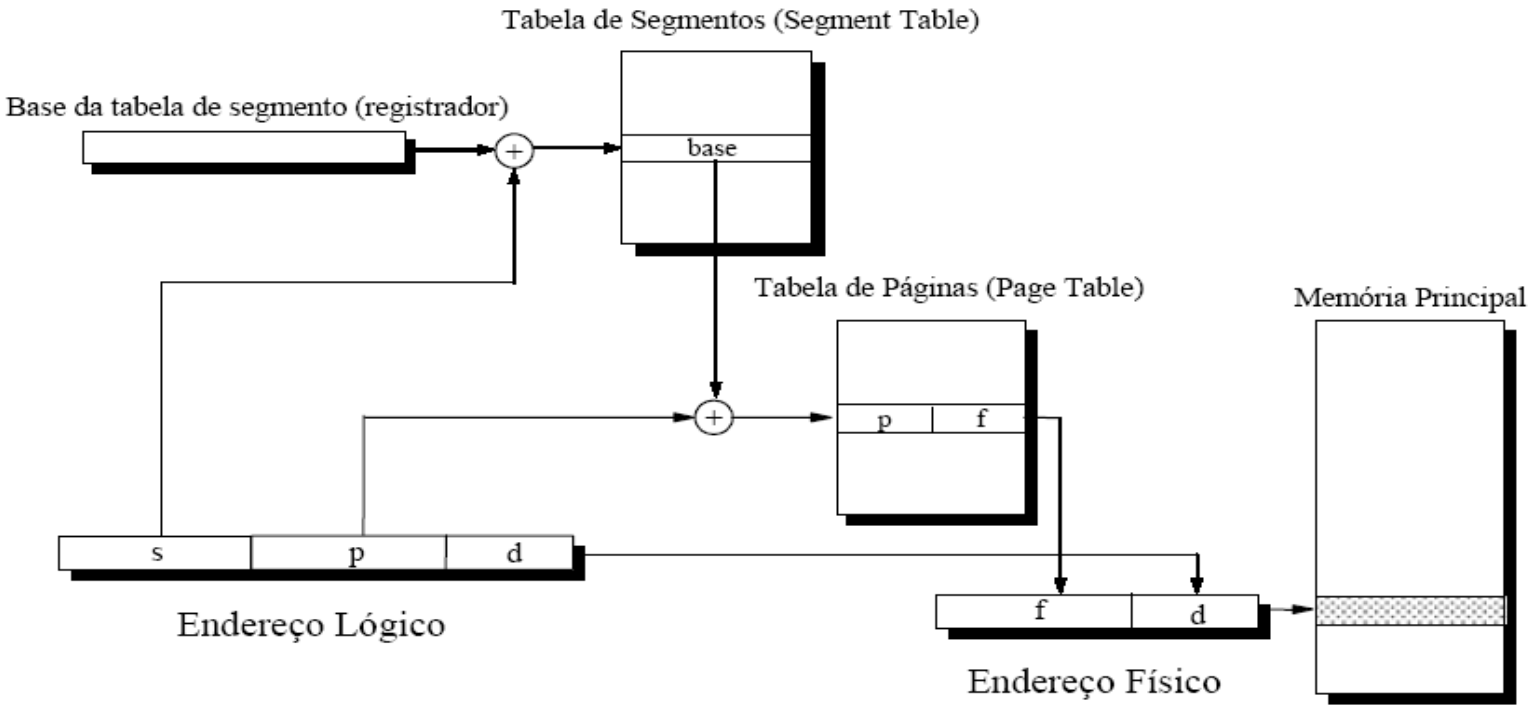
- Combina segmentação e paginação
  - Mapeamento segmentado: aproveita visão lógica de memória próxima do usuário
  - Mapeamento paginado: aproveita a gerência de memória mais simples
- Usuário aloca segmentos e estes são mapeados em grupo de páginas
- Organização
  - Memória física quebrada em frames de tamanho fixo, como na paginação
  - Informação das páginas que compõem cada segmento
- Unidade de gerência de tamanho fixo gera fragmentação interna

Endereçamento Não-Contíguo Segmento-Paginado

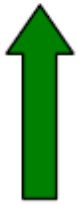
- Tabela de Segmentos usada para obter base da tabela de páginas do segmento desejado (uma por processo)
- Tabela de páginas usada para converter páginas de cada segmento em frames (uma tabela por segmento de cada processo)
- Tabela de Frames usada para controle dos frames livres
  - Uma tabela de frames para todo sistema
  - Tamanho fixo → Número de frames da MP é conhecido (tamanho da memória / tamanho da página)
  - Procura por first-"found" (qualquer lacuna serve para qualquer página)
- Quando processos morrem, suas páginas na tabela de frames são marcadas como livres

Endereçamento Não-Contíguo Segmento-Paginado (conversão de endereços)

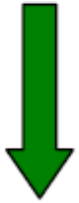
- Seqüência de acesso
  1. Número do segmento fornece na tabela de segmentos a base da tabela de páginas deste segmento
  2. Número da página e base da tabela de páginas fornece frame correspondente
  3. Número do frame e deslocamento dentro dele fornece acesso à memória



Endereçamento Não-Contíguo Segmento-Paginado (Vantagens e Desvantagens)



Gerência da memória principal é simples: - Substituição 1:1 (página e frame tem mesmo tamanho) - Alocação com primeiro frame livre
Segmento dá noção de localidade à um grupo de páginas
Sem fragmentação externa (área livre em frames, sempre cabem páginas)



Três acessos são necessários para cada operação com a memória
Preciso mais memória para tabelas
Fragmentação Interna