Sistemas de Computadores

Gestão da Memória

Luis Lino Ferreira

Maio de 2020

Gestão da Memória

- Gestão de memória? Porquê?
- Atribuição de instruções e dados à memória
- Endereços lógicos e físicos
- *Swapping*
- Alocação contígua
- Paginação
- Segmentação

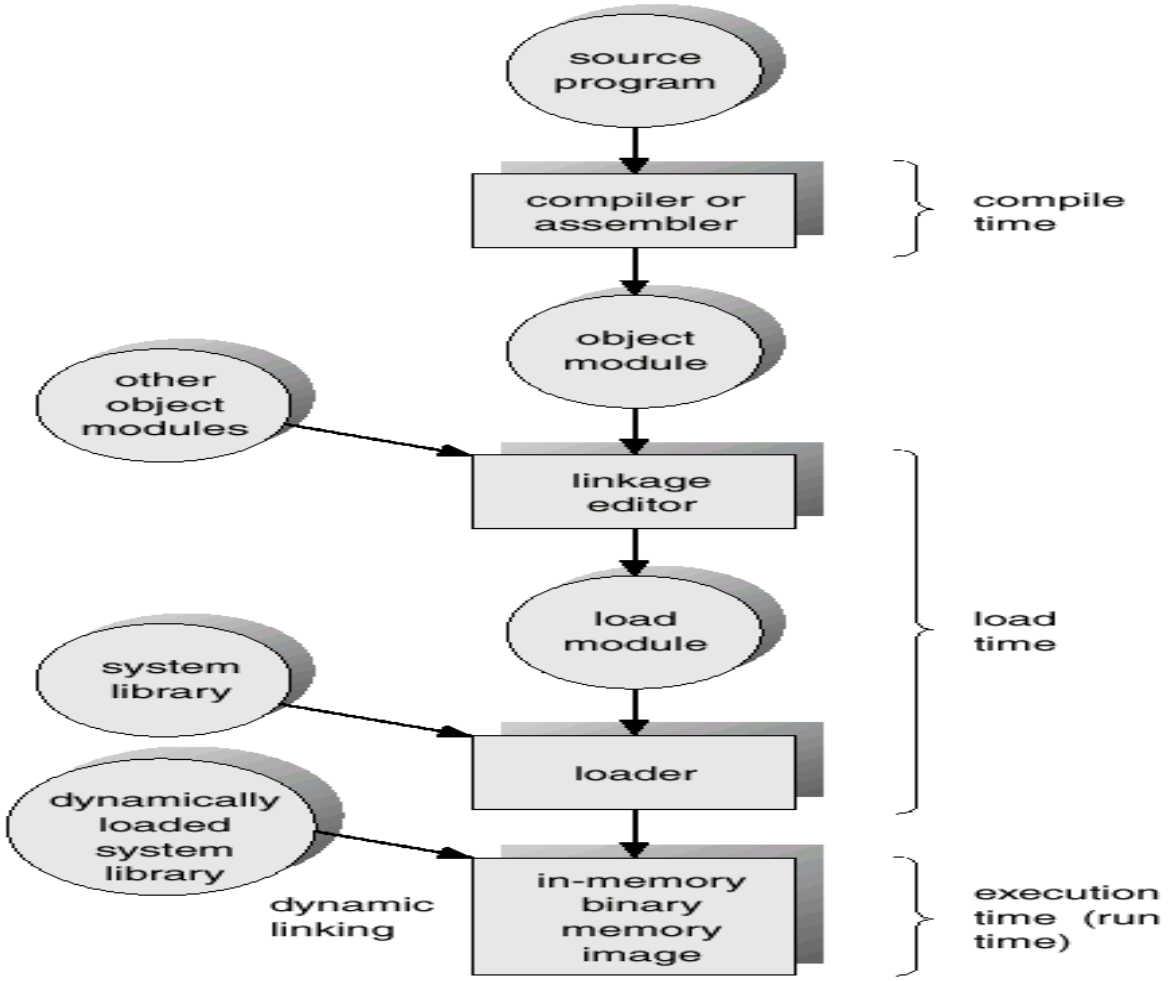
Introdução

- Um programa reside no disco sob a forma de ficheiro executável
- Para ser executado, o programa tem de ser colocado em memória e associado a um processo
- Em função da política de gestão de memória, o processo poderá transitar entre o disco e a memória durante o seu tempo de execução
- À medida que o processo é executado, este pode aceder ao subsistema de gestão de memória para:
  - aceder ao seu código
  - aceder a dados
  - requisitar espaço de memória
- Ao terminar, a memória por ele utilizada é libertada

Atribuição de instruções e dados à memória (Binding)

- A maior parte dos SO permite que um programa possa ser colocado em qualquer posição de memória. No entanto esta atribuição pode ser feita em:
  - **Tempo de compilação** – se a localização do programa poder ser conhecida à priori é gerado código com endereços absolutos. Alterações à localização do programa obrigam à sua recompilação
  - **Tempo de carga** – obriga o compilador a gerar código relocatável, a atribuição do programa a um determinado conjunto de endereços é feita na sua carga para memória
  - **Run-time** – se durante a execução do programa este poder ser recolocado noutra localização de memória. O que implica a utilização de hardware específico, mas é o mais habitual.

Atribuição de instruções e dados à memória (Binding)



Carga Dinâmica

- Uma rotina, utilizada por um programa, apenas é carregada em memória quando é necessária
- Permite melhorar a utilização da memória
  - Rotinas utilizadas com pouca frequência apenas são carregadas quando necessário
  - Exemplo: rotinas de detecção e tratamento de erros
- O processo de carga dinâmica é controlado pelo utilizador, não necessita do SO

Linkagem Dinâmica

- A linkagem dinâmica apenas é feita quando um programa é carregado em memória
- Particularmente útil para as livrarias do sistema:
  - No Windows as *Dinamic Link Libraries* (DLL)
  - No Linux as livrarias partilhadas nnn.a
- Vantagens:
  - Programas mais pequenos
    - as livrarias são adicionadas em *run-time*
    - permite partilhar as livrarias entre vários programas
  - As livrarias podem ser actualizadas sem implicações para os programas
    - Caso necessário o programa pode utilizar a versão mais adequada da livraria

Linkagem Dinâmica

- Como:
  - Uma pequena parte do código (stub) é utilizada para localizar a livraria
  - Caso a livraria ainda não se encontre em memória é carregada
  - Ao executar o código do stub, o programa está realmente a executar o código da livraria
  - O SO é responsável por permitir o acesso de múltiplos programas às livrarias

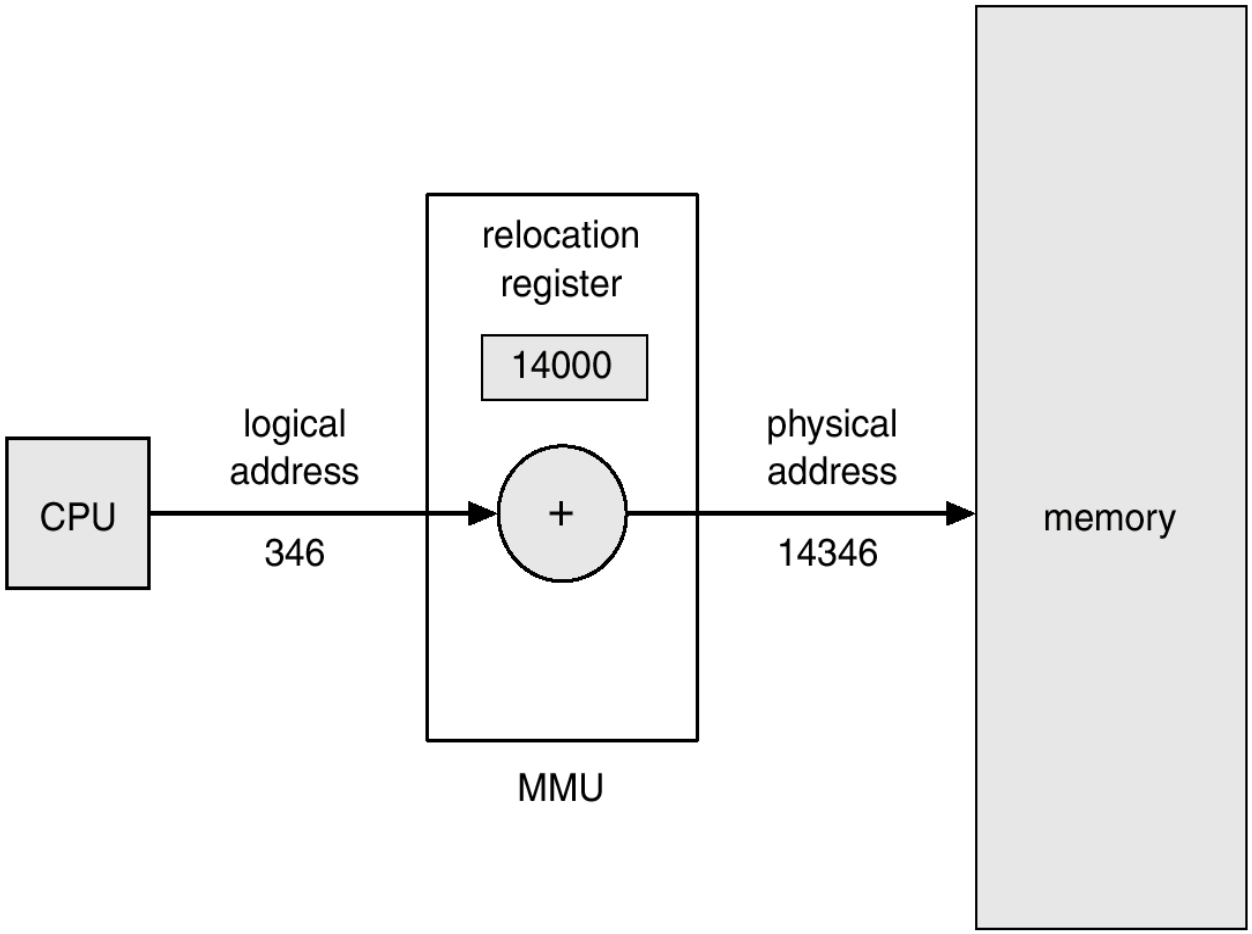
Endereços Lógicos vs Endereços Físicos

- Endereço lógico:
  - Endereço gerado pela programa em execução, também chamado de **Endereço Virtual**
- Endereço Físico:
  - Endereço real de memória

Endereços Lógicos vs Endereços Físicos

- Memory Management Unit (MMU)
  - O endereço lógico é convertido pela MMU num endereço físico
  - A MMU adiciona ao endereço lógico o valor do registo de relocação

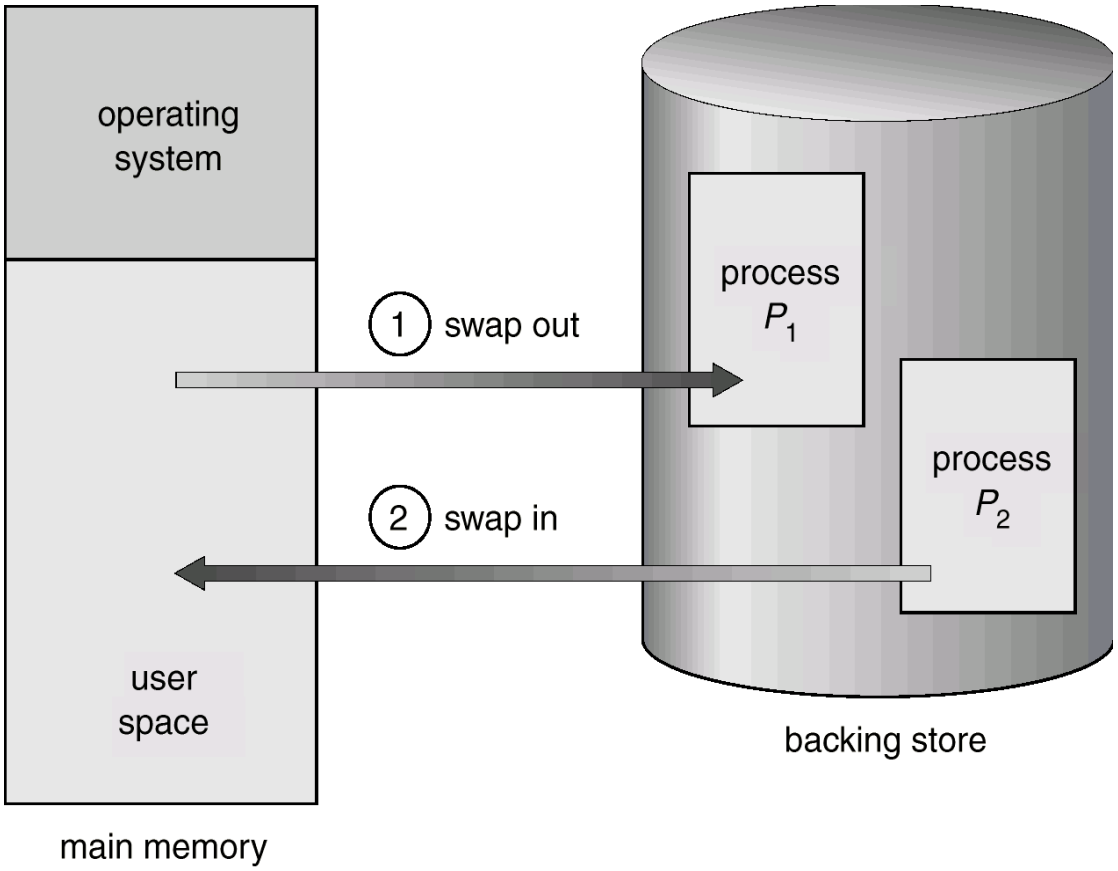
Endereços Lógicos vs Endereços Físicos



Swapping

- Mecanismo que permite retirar um processo da memória principal e para memória secundária (disco)
- Pode ser utilizado em conjunto com o critério de escalonamento
  - *Round-Robin* – o processo é retirado para disco assim que terminar o seu *time quantum*
  - Prioridade – os processo menos prioritários são retirados para disco até poderem ser executados
- Pode reduzir fortemente a performance do sistema, por ex., o tempo necessário para retirar um processo com 1MB da memória e colocá-lo em disco é aproximadamente de 208ms.

Swapping


operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

Swapping

- Um processo não pode ser retirado quando:
  - Estiver à espera de operações de I/O
    - Por ex., as operações de DMA são configuradas para serem feitas para determinado endereço de memória física, se entretanto o processo for *swaped out* a zona de memória anteriormente configurada já não será válida

Alocação de Memória Contígua

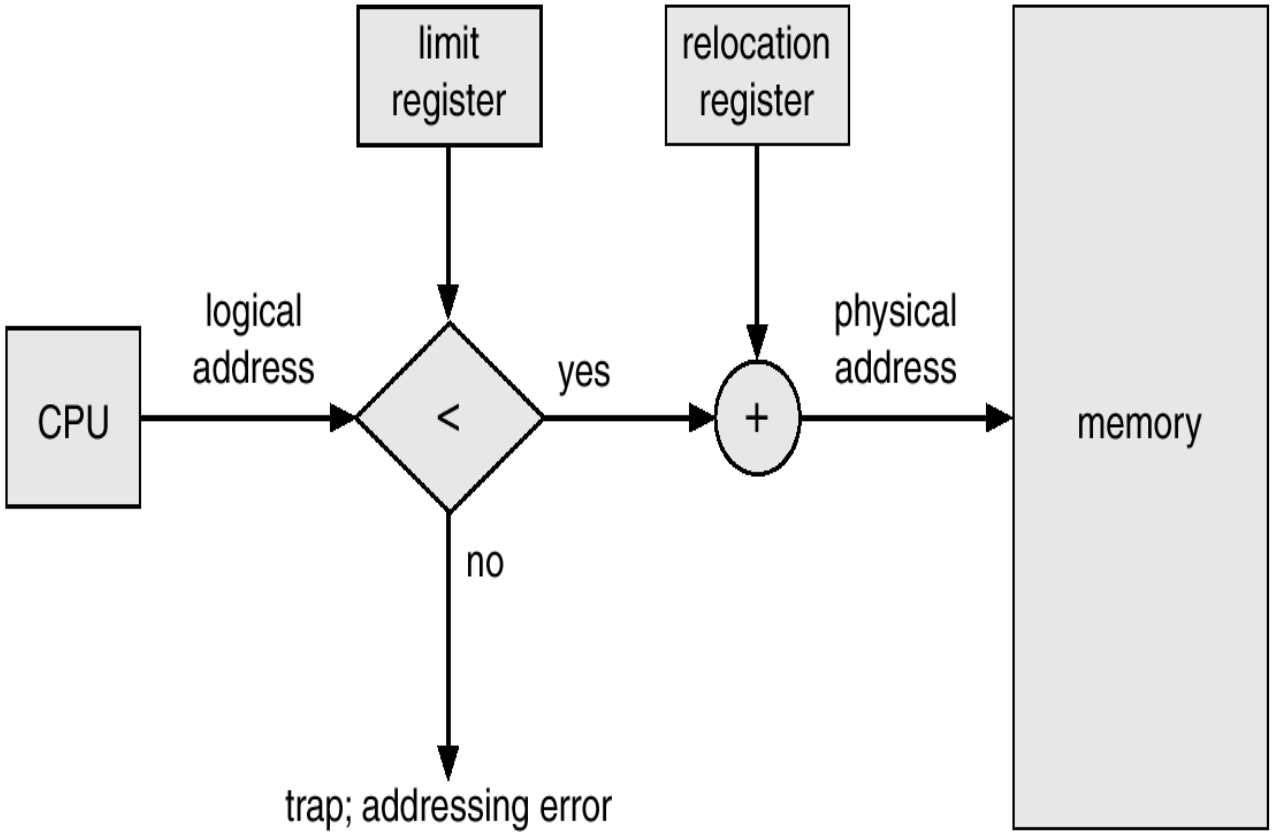
- A memória é dividida em duas partes:
  - Sistema Operativo
  - Programas do utilizador

- Como alocar a memória para um processo?

Alocação de Memória Contígua

- Protecção de memória
  - O programa é compilado como se utiliza-se a memória a partir do endereço 0
  - Quando o escalonador selecciona um processo para entrar em execução:
    - Carrega o registo *limit* com o valor máximo da memória a aceder
    - Carrega o registo de realocação com a primeira posição de memória física utilizável pelo programa

Alocação de Memória Contígua



Alocação de Memória Contígua

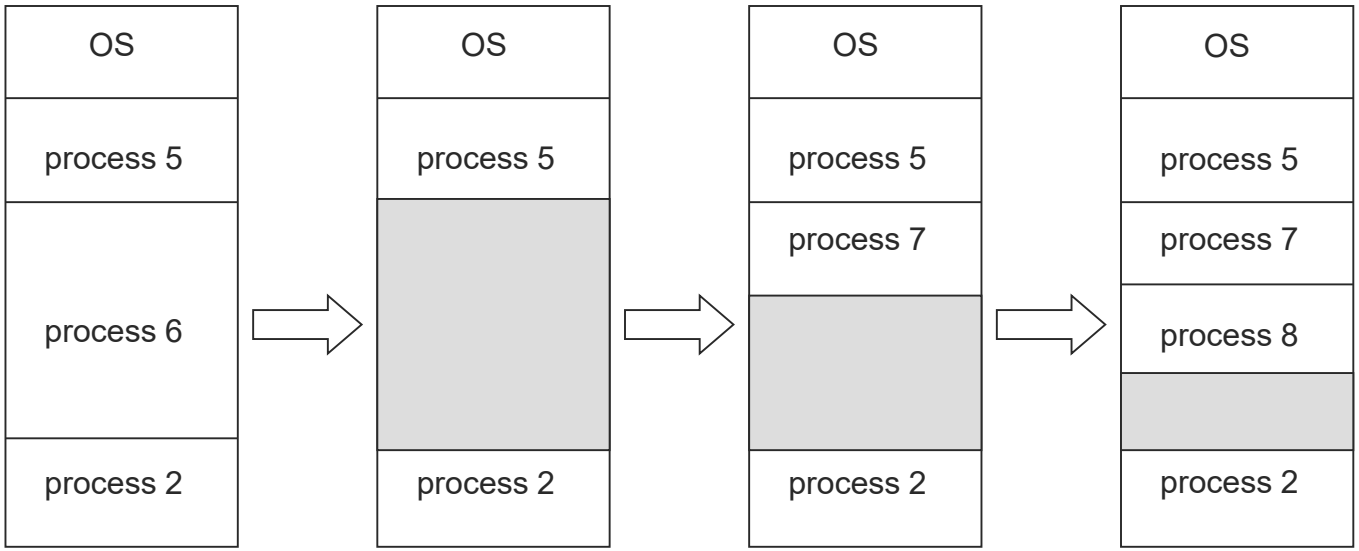
- Alocação de memória
  - *Multiple-partition allocation* (*fixed size*)
    - Partição da memória em pedaços de tamanho fixo, a cada processo é atribuída uma partição de memória
    - Sempre que um processo entra em funcionamento é-lhe atribuído um pedaço
  - Problemas:
    - Fragmentação da memória devido a um programa não ocupar todo o espaço disponível dentro de cada partição

Alocação de Memória Contígua

- Alocação de memória
  - *Multiple-partition allocation* (*variable size*)
    - O SO mantêm uma tabela com as partições livres e ocupadas
    - Quando um processo chega é-lhe atribuído um pedaço de memória livre

Alocação de Memória Contígua

Exemplo:



Alocação de Memória Contígua

- Outras soluções:
  - *First-fit*
    - É atribuído ao processo o primeiro pedaço livre com tamanho suficiente. A procura começa no princípio do conjunto de pedaços livres ou no ponto em que a busca anterior tinha terminado
  - *Best-fit*
    - É atribuído ao processo parte do pedaço livre mais pequeno e com tamanho suficiente para carregar o programa. Permite utilizar de forma eficiente a memória, mas pode ser lento
  - *Worst-fit*
    - É atribuído ao processo parte do pedaço livre maior. Permite a existência de pedaços livres de maiores dimensões, mais facilmente utilizáveis do que pedaços de pequenas dimensões

Alocação de Memória Contígua

- Fragmentação
  - Externa
    - À medida que os processo são retirados e carregados em memória a memória pode ficar com espaços livres contínuos, de pequenas dimensão
    - Quando um programa requerer a sua carga em memória, embora a memória total disponível seja maior que os requisitos do programa, tal pode não ser possível dado não existir nenhum pedaço de memória com dimensão suficiente
    - Analise estatística de vários casos permitiu determinar que cerca de 1/3 da memória de um computador é desperdiçada devido à fragmentação

Alocação de Memória Contígua

- Fragmentação
  - Interna
    - Normalmente a memória é alocada utilizando pedaços de memória de tamanho fixo. Por ex., 4096 Bytes
    - Logo, se um programa não for um múltiplo da unidade fixa, vai existir sempre um pedaço de memória desperdiçado

Alocação de Memória Contígua

- Fragmentação
  - Solução
    - Compactação
      - Processo através do qual os pedaços de memória ocupados são agrupados, ficando apenas um “grande” pedaço livre
      - Apenas é possível se o SO permitir realocação dinâmica em tempo de execução
    - Permitir a um processo a alocação de espaços de memória não contíguos

Paginação de Memória

- Método de gestão de memória que permite que o espaço de armazenamento seja não contíguo

- A paginação é suportada por hardware ou por uma combinação do hardware com o software

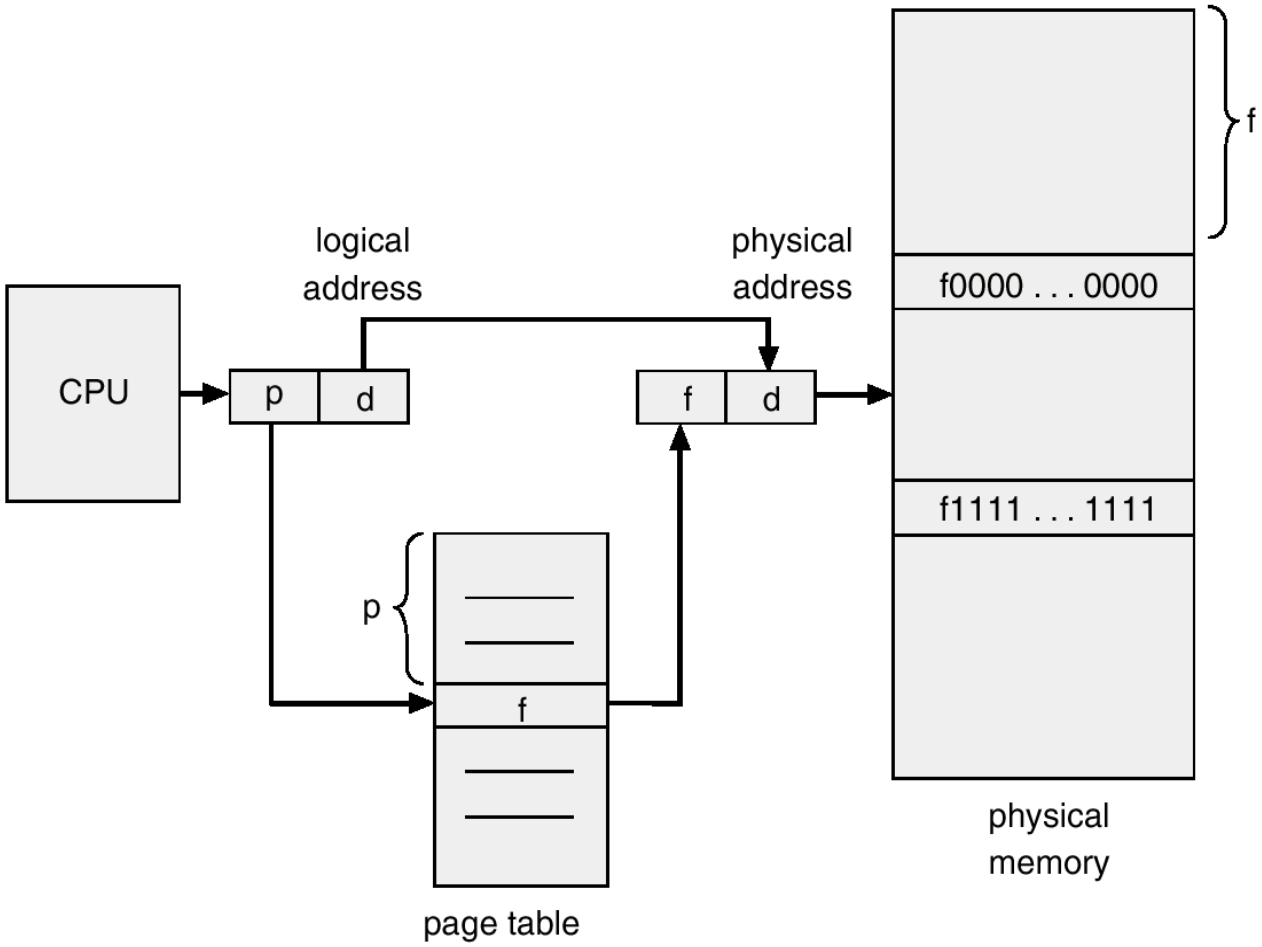
Paginação de Memória

- Divide-se a memória física em blocos de tamanho fixo, chamados *frames*, cujo tamanho é uma potência de 2, normalmente entre 512 e 16MB
- Divide-se a memória lógica em blocos do mesmo tamanho, chamados páginas (*pages*)
- Há que registar todas as *frames* livres
- Para correr um programa com um tamanho de n páginas, é necessário encontrar n *frames* livres e carregar o programa
- Activar uma tabela de páginas para converter endereços lógicos em endereços físicos
- **Nota:** A fragmentação externa é eliminada, mas não a fragmentação interna

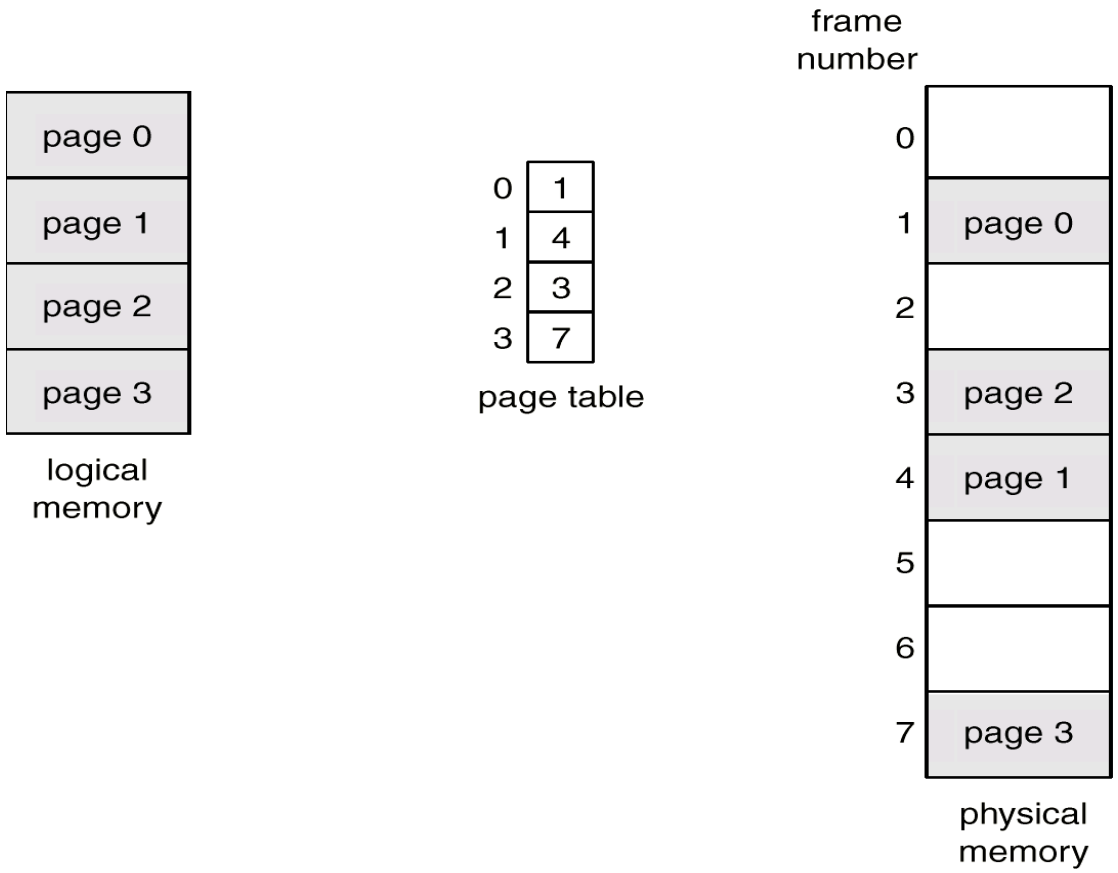
Paginação de Memória

- O endereço gerado pelo CPU é dividido em 2 partes:
  - *Page number*: usado como índice na tabela de páginas que contém o endereço base de cada página em memória física
  - *Page offset*: combinado com o endereço de base para definir o endereço físico que é enviado para a unidade de memória

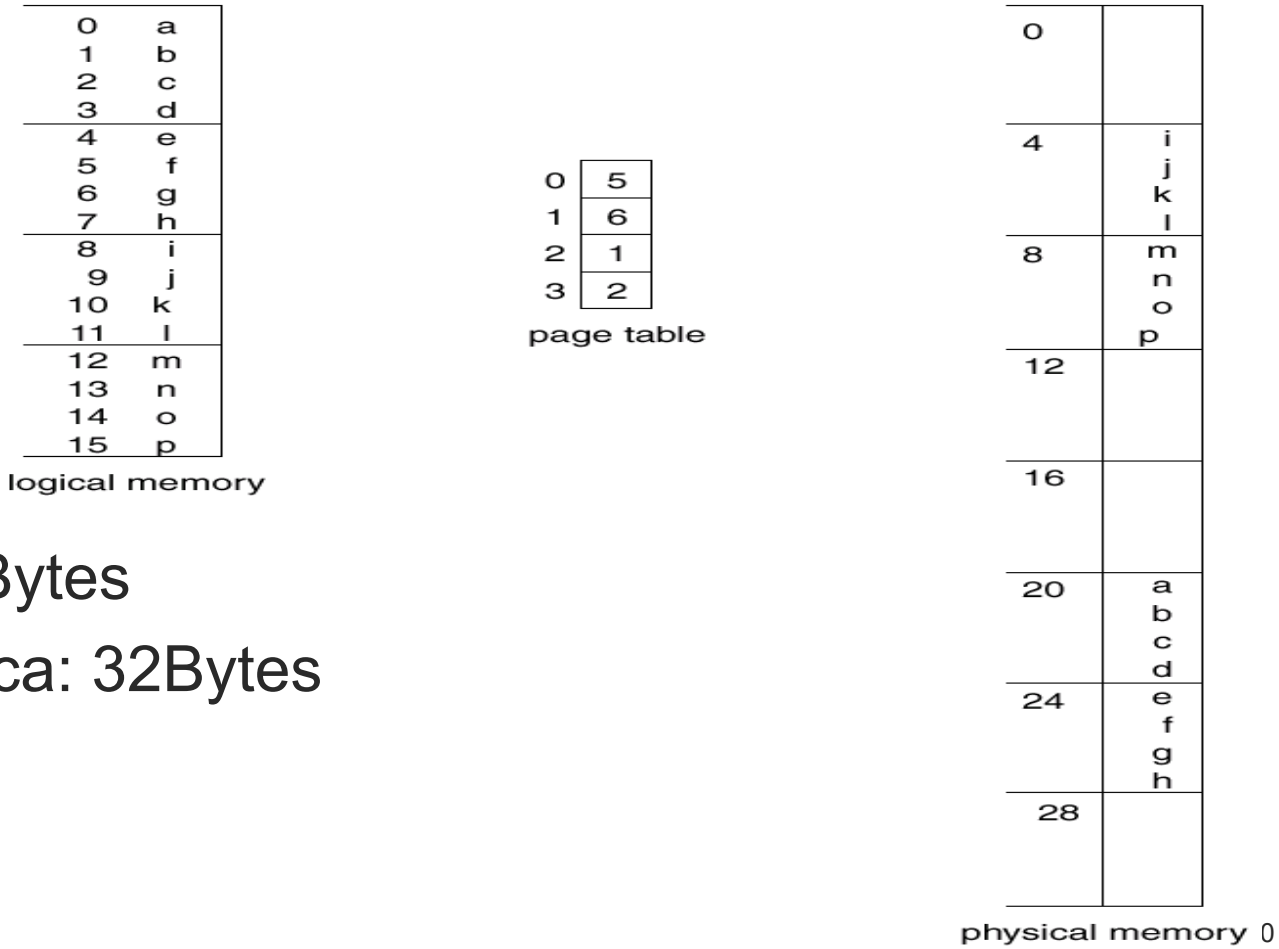
Paginação de Memória



Paginação de Memória



Paginação de Memória



Page size: 4Bytes

Memória Física: 32Bytes

Paginação de Memória

- Implementação:
  - A tabela de páginas é guardada na memória principal
  - *Page-table base register* (PTBR) aponta para a tabela de páginas
  - *Page-table length register* (PRLR) indica o tamanho da tabela de páginas
  - Qualquer acesso a dados/instruções requer 2 acessos à memória:
    - um para a tabela de páginas
    - outro para os dados/instruções
  - O problema dos dois acessos à memória pode ser resolvido através duma *cache* de pesquisa rápida, designada por memória associativa ou *Translation Look-aside Buffers* (TLBs)

Paginação de Memória

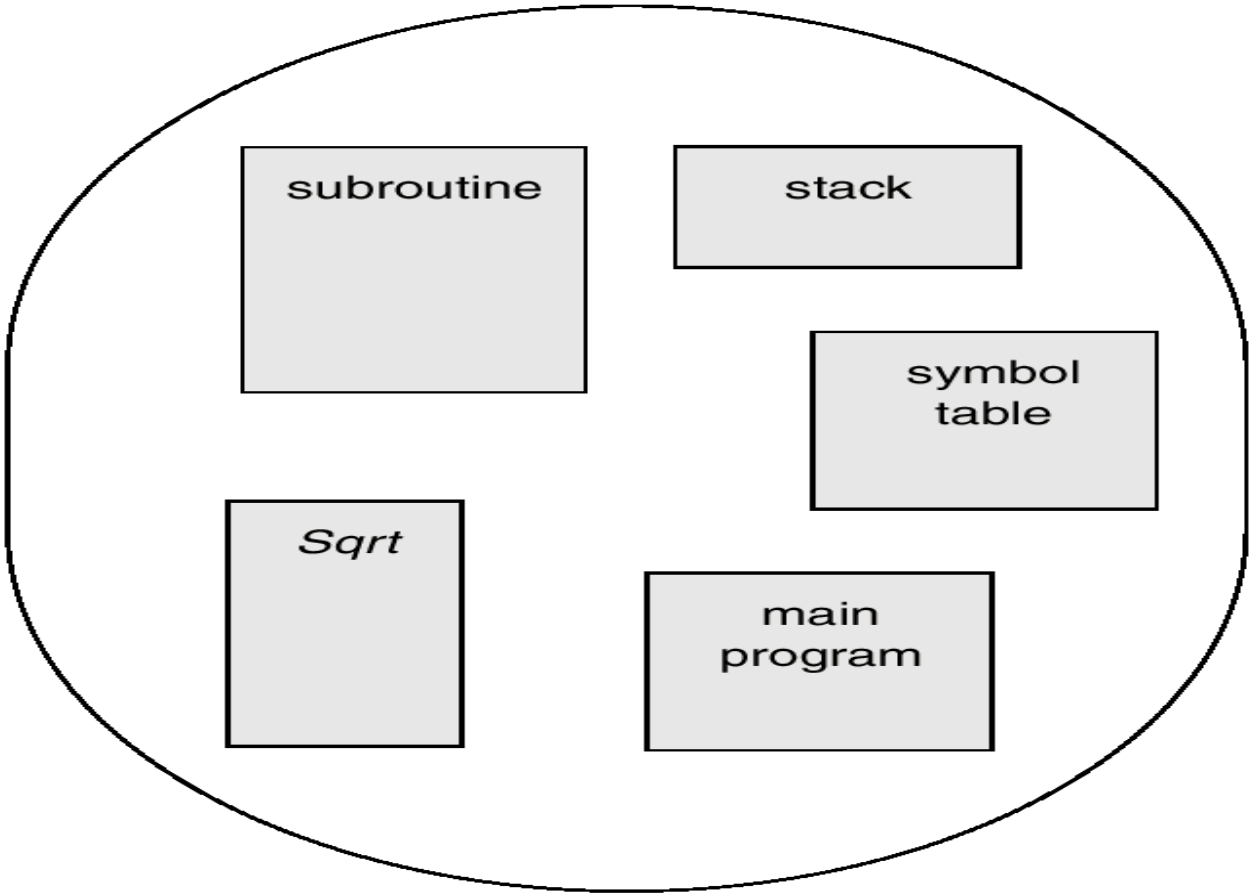
- Estrutura da tabela
  - Hierárquica
  - *Hashed*
  - Invertida

Segmentação de Memória

- A Segmentação da Memória permite que um utilizador veja o espaço de memória como se estivesse dividido em várias partes (segmentos) diferentes
  - Um endereço lógico passa a ser referenciado pelo número do segmento e pelo *offset* dentro do segmento:

    <nº do Segmento, offset>

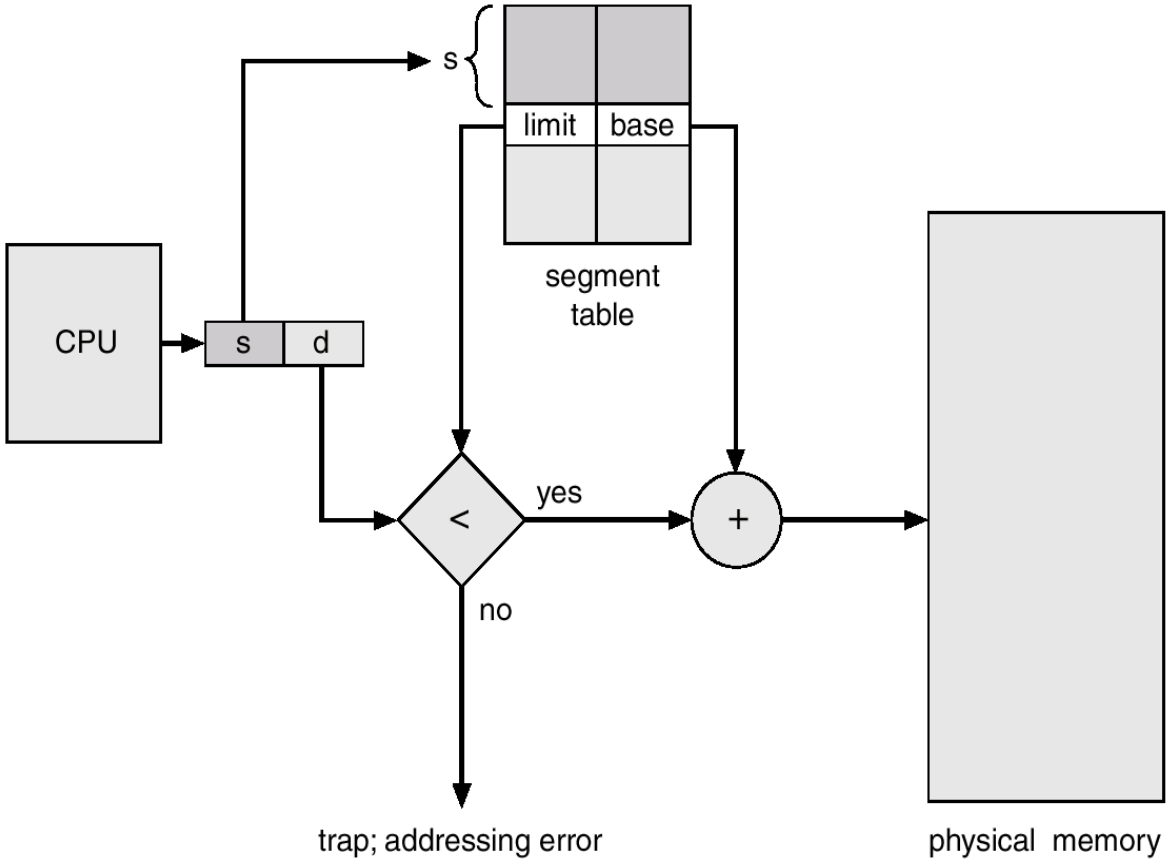
Segmentação de Memória



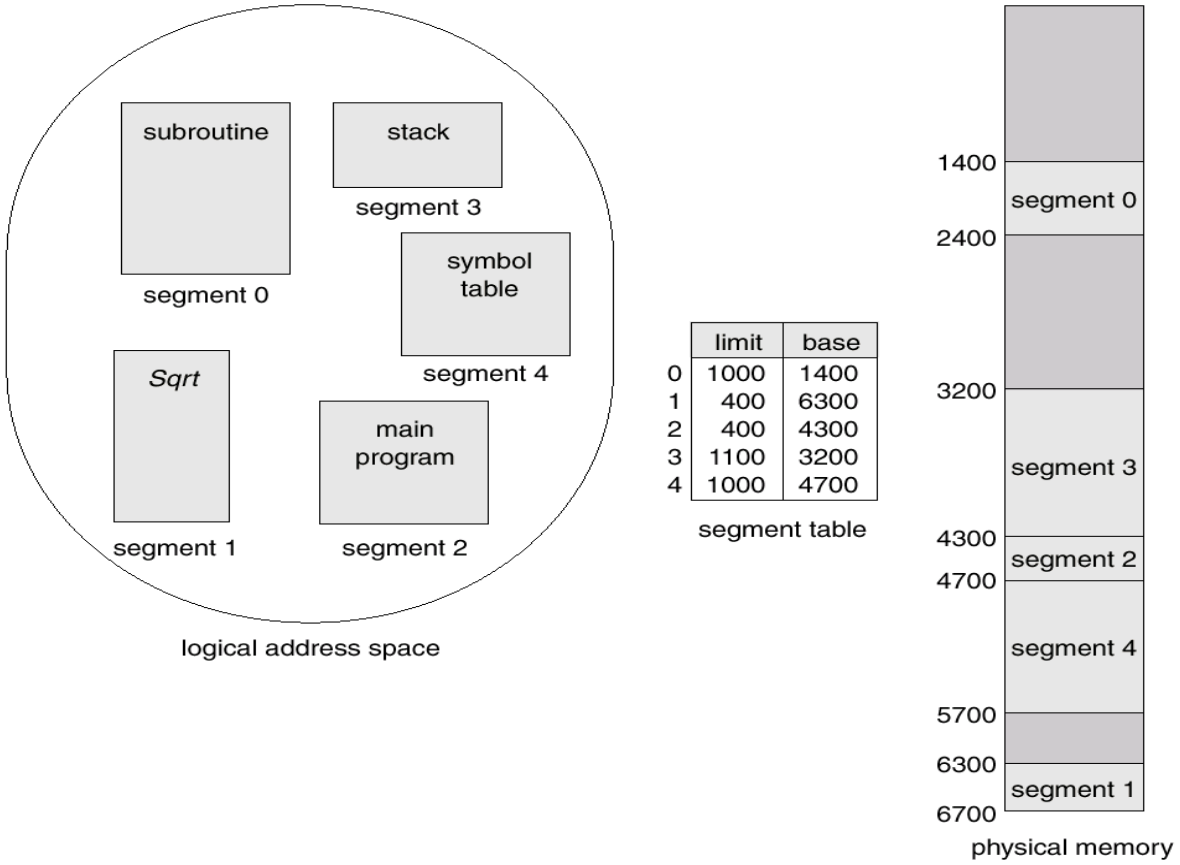
Segmentação de Memória

- Tabela de segmentos
  - traduz endereços lógicos bidimensionais em endereços físicos unidimensionais; cada entrada da tabela tem:
    - *base* – contém o endereço físico do início do segmento em memória
    - *limit* – especifica o comprimento do segmento
- *Segment-Table Base Register* (STBR)
  - aponta para a localização da tabela de segmentos em memória
  - É gravada pelo SO no PCB sempre que existe uma mudança de contexto
- *Segment-Table Length Register* (STLR)
  - indica o número de segmentos usados pelo programa; o identificador segmento *s*, é legal se *s* < STLR

Segmentação de Memória



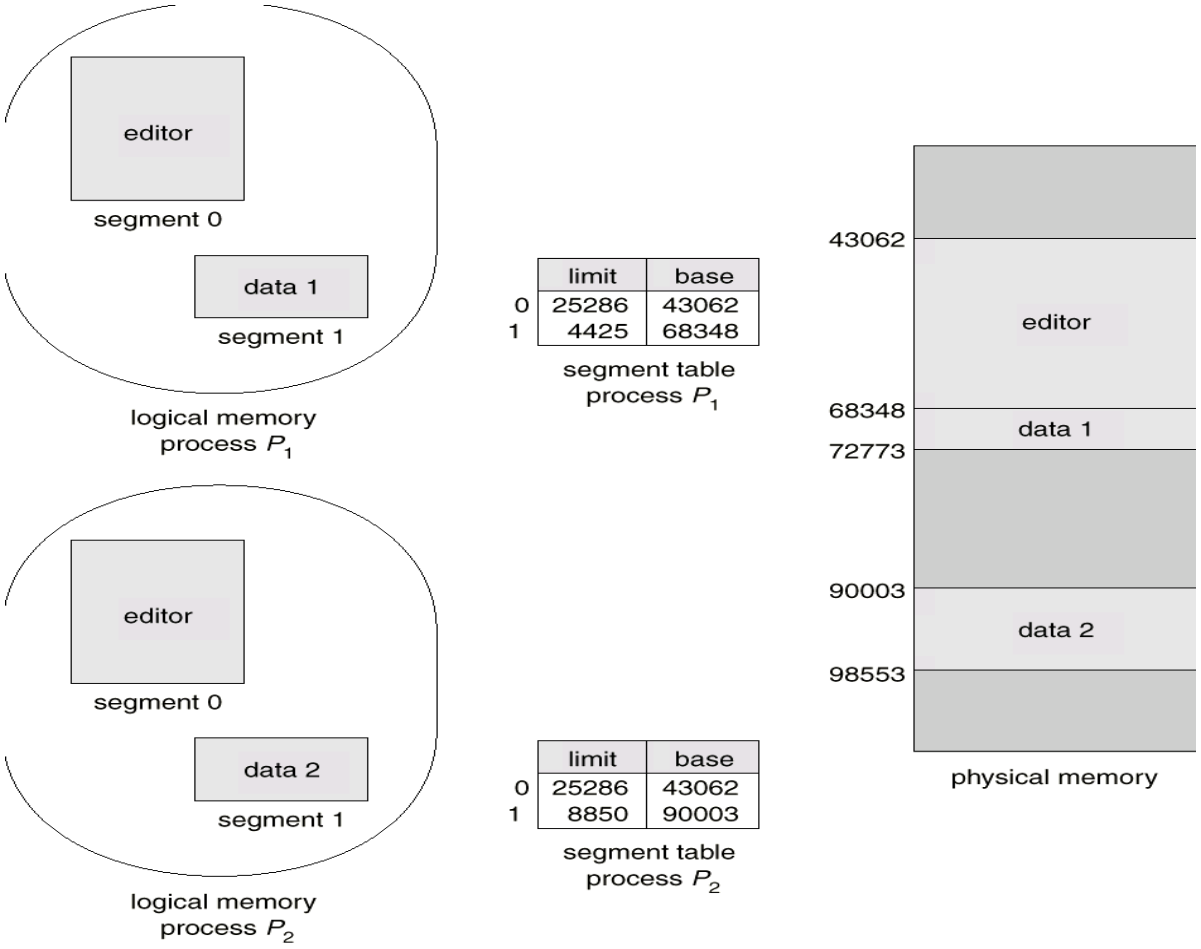
Segmentação de Memória



Segmentação de Memória

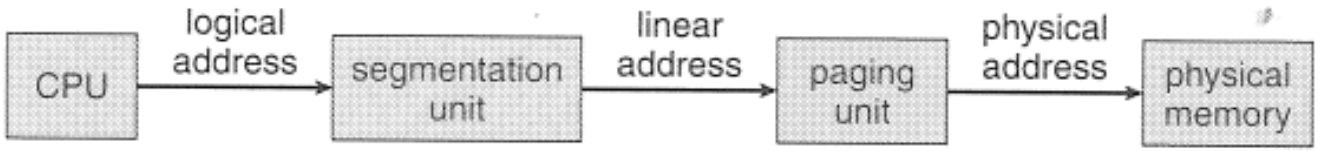
- Principal vantagem
  - Protecção no acesso aos segmentos de memória
  - Exemplo: a tentativa de acesso para além dos limites de um determinado segmento de memória causa uma interrupção de software e não permite o acesso caso o processo não tenha direito de acesso ao segmento
- A segmentação permite também a partilha de segmentos de memória entre processo diferentes de acordo com as protecções de acesso definidas

Segmentação de Memória



Pentium

- Supports pure segmentation and paging
  1. The CPU generates a logical address
  2. The segmentation unit generates a linear address
  3. The paging unit generates a physical address



Pentium segmentation

- Segments can be as large as 4GB
- Each process can have at most 16KB of segments
  - 8KB private to the process, stored on the **Local Allocation Table** (LDT)
  - 8KB shared with all processes, stored on the G**lobal Allocation Table** (GDT)
  - Each LDT or GDT entry is composed by:
    - Base address
    - Limit
    - Options: type of segment, privilege levels, etc

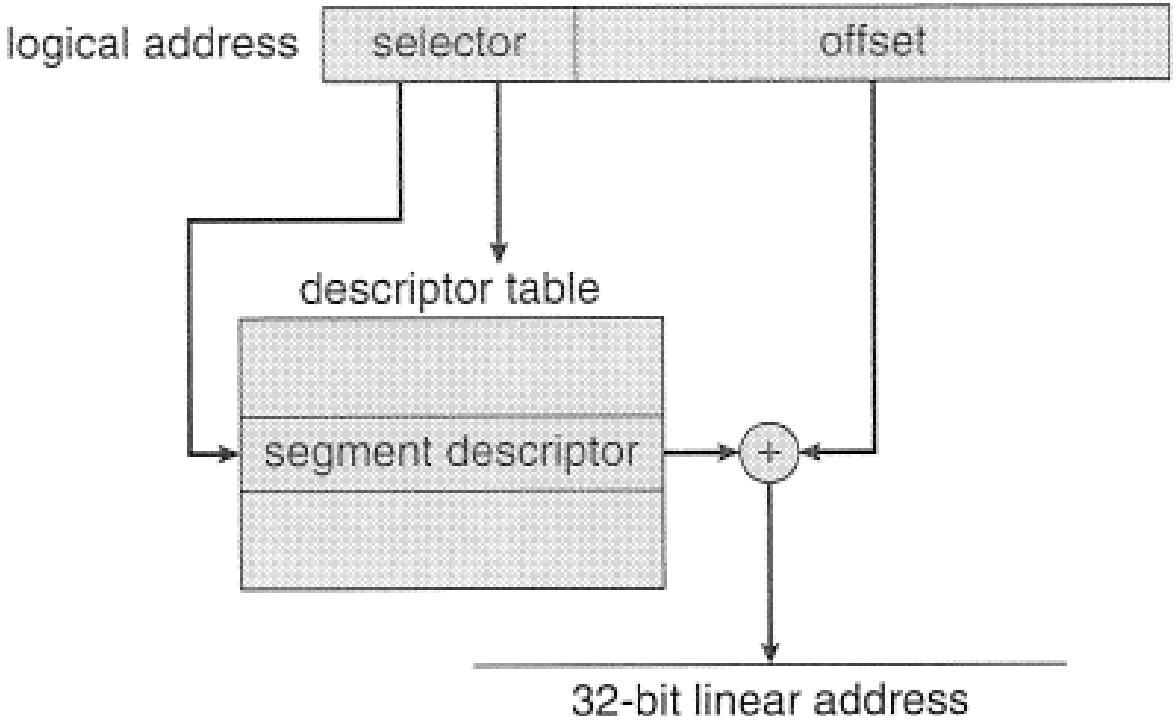
Pentium segmentation

- A logical address is a pair
  - (selector, offset)
- The CPU has 6 segment registers, thus enabling the addressing of 6 segments, at a time, by each process
  - Additionally, it also has 6 8-Byte microprogram registers to hold the corresponding descriptors from the LDT or GDT table (**why?**)

Pentium segmentation

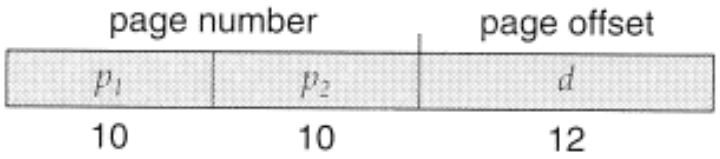
- The base and limit information about each segment is used to generate a linear address:
  1. The CPU checks if the segment limit is respected
  2. If invalid a memory fault is generated
  3. Otherwise, the base is added to the address generating a 32 bits address

Pentium segmentation



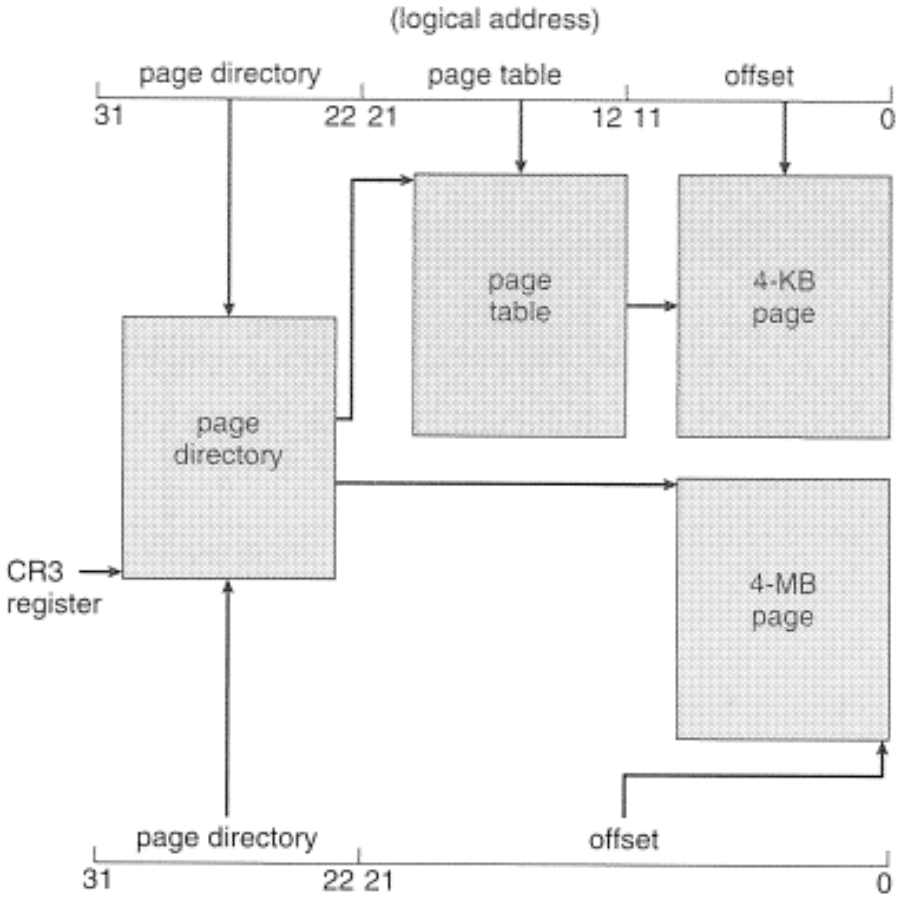
Pentium paging

- Page size between 4kB until 4MB
- Two level paging



  - The first 10 bits refers to the outermost page table directory
  - The second 10 bits refer to the page entry

Pentium paging



Linux on Pentium Systems

- Linux has been designed to operate in several kinds of CPUs, many without support for segmentation or paging
- Linux uses 6 segments:
  - Kernel Code
  - Kernel Data
  - User code
  - User data
  - Task-state segment (TSS)
  - LDT segment