Aula 17 – Paginação, Segmentação e Dispositivos de Entrada e Saída

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

14 de novembro de 2014

Implementação de Paginação

- Endereços de disco usados para armazenar páginas que não estão na memória não fazem parte da tabela de páginas
  - A tabela contém apenas a informação que o *hardware* precisa para traduzir endereço virtual em físico
- Onde então colocar as páginas no disco, quando retiradas da memória?
  - E como saber onde estão? Como gerenciar isso?

Implementação de Paginação

- A solução mais simples é ter uma partição especial de swap
  - Solução do Unix e Linux
  - Não possui um sistema de arquivos normal
    - São usados números de blocos relativos ao início da partição
  - Quando o SO inicia, a partição está vazia
    - Representada na memória como uma única entrada contendo sua localização e tamanho
  - À medida em que processos são iniciados, o SO reserva um pedaço da área de swap do tamanho do processo
    - Quando terminam, o espaço é liberado

Paginação – Partição de Swap

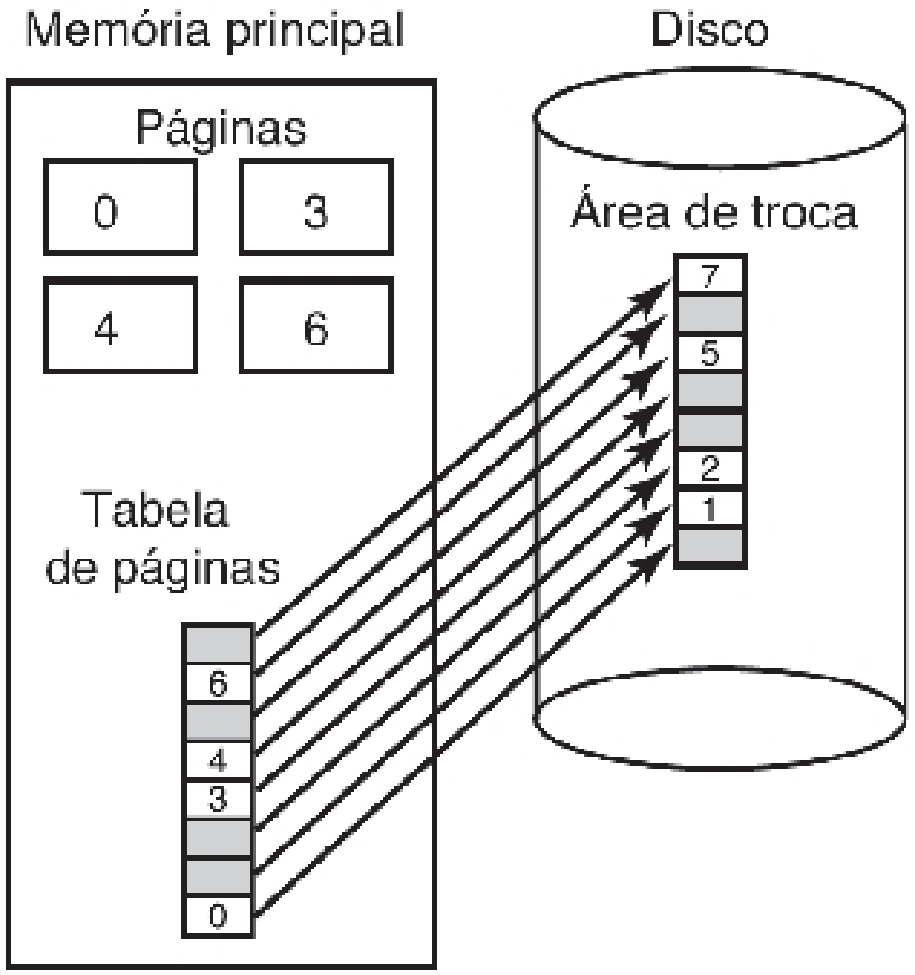
- A área de troca é gerenciada como uma lista de espaços disponíveis
  - Há algoritmos melhores, mas que não serão discutidos
- Associado a cada processo está o endereço no disco de sua área de swap
  - Mantido na tabela de processos
  - Cálculo do endereço para escrever uma página:
    - Adicione o deslocamento da página em seu espaço de endereçamento virtual ao endereço inicial da área de swap associada ao processo

Paginação – Partição de Swap

- Problema: antes de um processo iniciar, a área de swap deve ser inicializada
  - Possibilidade A – Assim que o processo é criado, ele é copiado todo para sua área de troca no disco, sendo carregado para memória quando necessário
    - Alternativamente, podemos copiá-lo todo para a memória, deixando que suas páginas sejam derrubadas se necessário
    - Problema: processos podem aumentar de tamanho enquanto executam (pilha e dados)
    - Solução: reservar áreas de troca diferentes para código, dados e pilha, permitindo que elas consistam de mais de um bloco no disco

Paginação – Partição de Swap

- Possibilidade A:
  - Basta saber o endereço do início da área de swap do processo $\rightarrow$ As páginas são espelhadas no disco
    - A área de swap é tão grande quanto o espaço de endereçamento virtual do processo
  - Área de troca (swap) estática

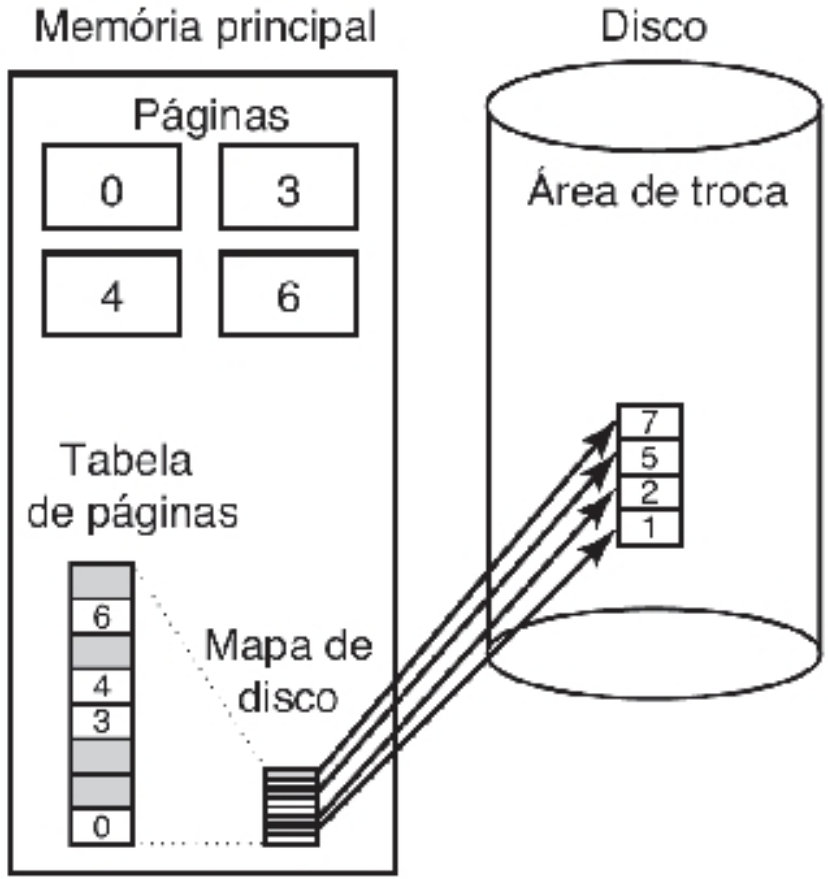


Paginação – Partição de Swap

- Possibilidade B – Nada é alocado antecipadamente.
  - Espaço é alocado em disco quando a página for enviada para lá e desalocado quando o processo termina
    - Alocação feita a cada página
    - Páginas "somente leitura" e as nunca modificadas são re-lidas dos arquivos originais, não ficando no swap
  - Assim, processo na RAM não fica "amarrado" a uma área de swap específica
  - Desvantagem: precisamos manter na memória o endereço de cada página armazenada em disco
    - Deve haver uma tabela para cada processo dizendo onde cada página está no disco (se estiver lá)
    - Antes, bastava saber onde o processo iniciava no disco

Paginação – Partição de Swap

- Possibilidade B:
  - As páginas não têm endereço fixo
    - Quando uma página é derrubada, uma vazia em disco é escolhida e o mapa atualizado
    - Páginas na memória não têm cópia em disco (a não ser que tenha sido modificada em algum momento da execução)
  - Área de troca dinâmica



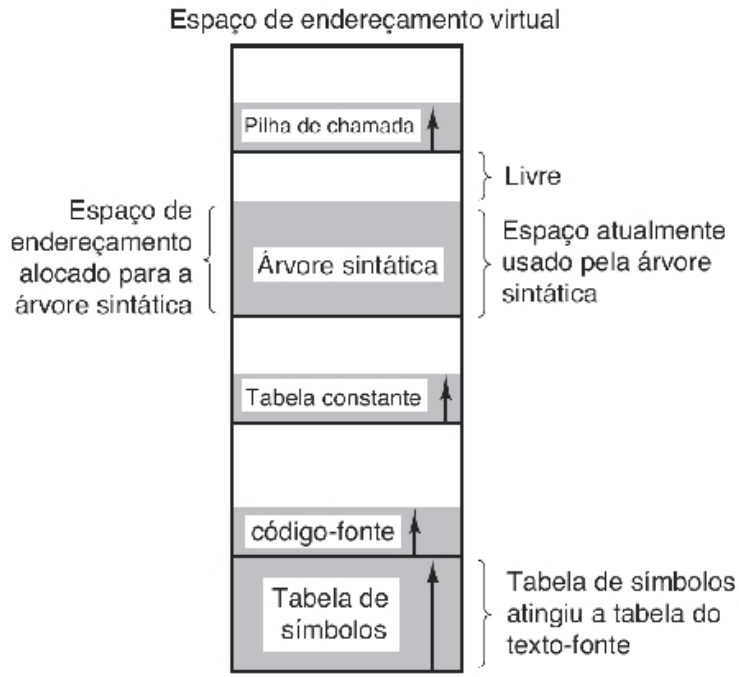
Paginação – Partição de Swap

- E quando não podemos ter partição de swap?
  - Podemos usar um ou mais arquivos grandes, pré-alocados, dentro do sistema de arquivos normal
    - Usado no Windows e em celulares (Linux também suporta)
  - Permite otimizações:
    - Uma vez que o texto do programa veio de algum executável, podemos usar o executável como área de swap para o texto
    - O mesmo esquema pode ser usado com bibliotecas compartilhadas (shared libraries)
    - Lembre que o código é em geral read-only (não precisa ser devolvido da memória para o disco)
  - Problemáticas em caso de falta de energia – o sistema de arquivos pode ficar inconsistente

Segmentação

Segmentação

- A memória virtual é unidimensional
  - Endereços virtuais vão de 0 a um máximo
  - Considere um compilador:
    - Código-fonte
    - Tabela de símbolos
    - Constantes
    - Pilha de execução
    - Árvore Sintática
    - Todas podem crescer... e podemos ficar sem memória para alguma, dada a alocação contígua dessas estruturas

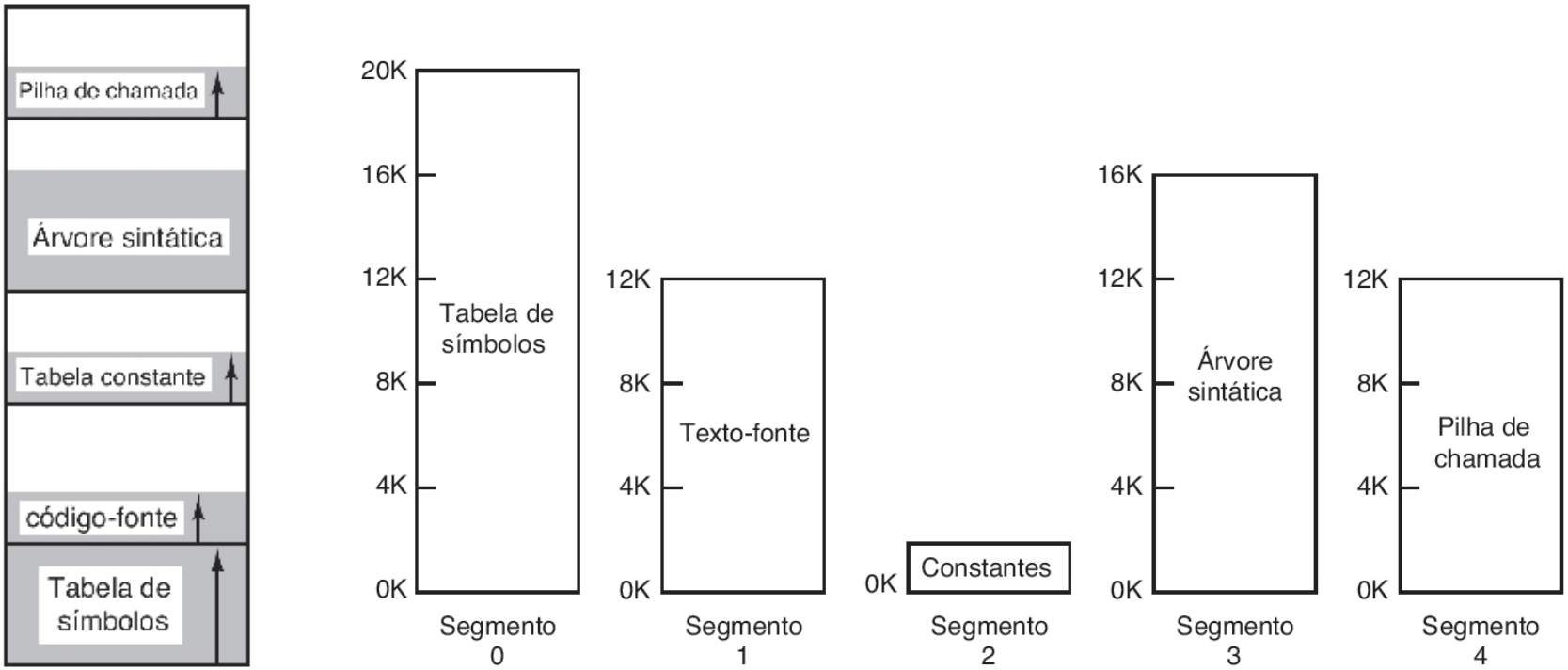


Segmentação

- Solução: Segmentos
  - Espaços de endereçamento independentes e separados
  - Sequência linear de endereços, de 0 a um máximo
    - Seu tamanho está entre 0 e esse máximo
  - Podem ter tamanhos diferentes, e o tamanho pode variar durante a execução
    - Ainda assim, até um máximo $\rightarrow$ pode "encher"
  - Especificação de endereço feita em 2 partes:
    - Número do segmento e o endereço dentro do segmento

Segmentação

Ex: Um segmento para cada estrutura do compilador



Segmentação

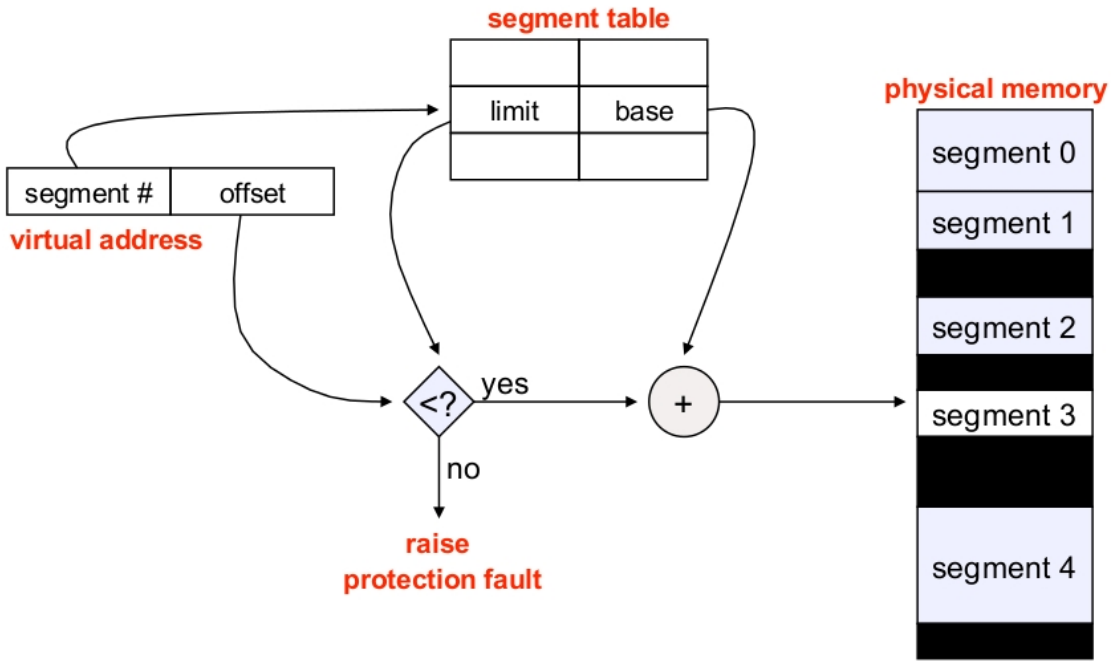
- Segmentos são entidades lógicas, e assim são vistos pelo programador
  - Podem conter uma rotina, um arranjo, uma coleção de variáveis, pilha etc
  - Normalmente não contêm misturas de tipos
  - O usuário tem a ilusão de que todos os segmentos estão na memória principal, o tempo todo
- Implementação
  - Difere-se da paginação em que os segmentos não possuem tamanho fixo, enquanto que páginas possuem

Segmentação

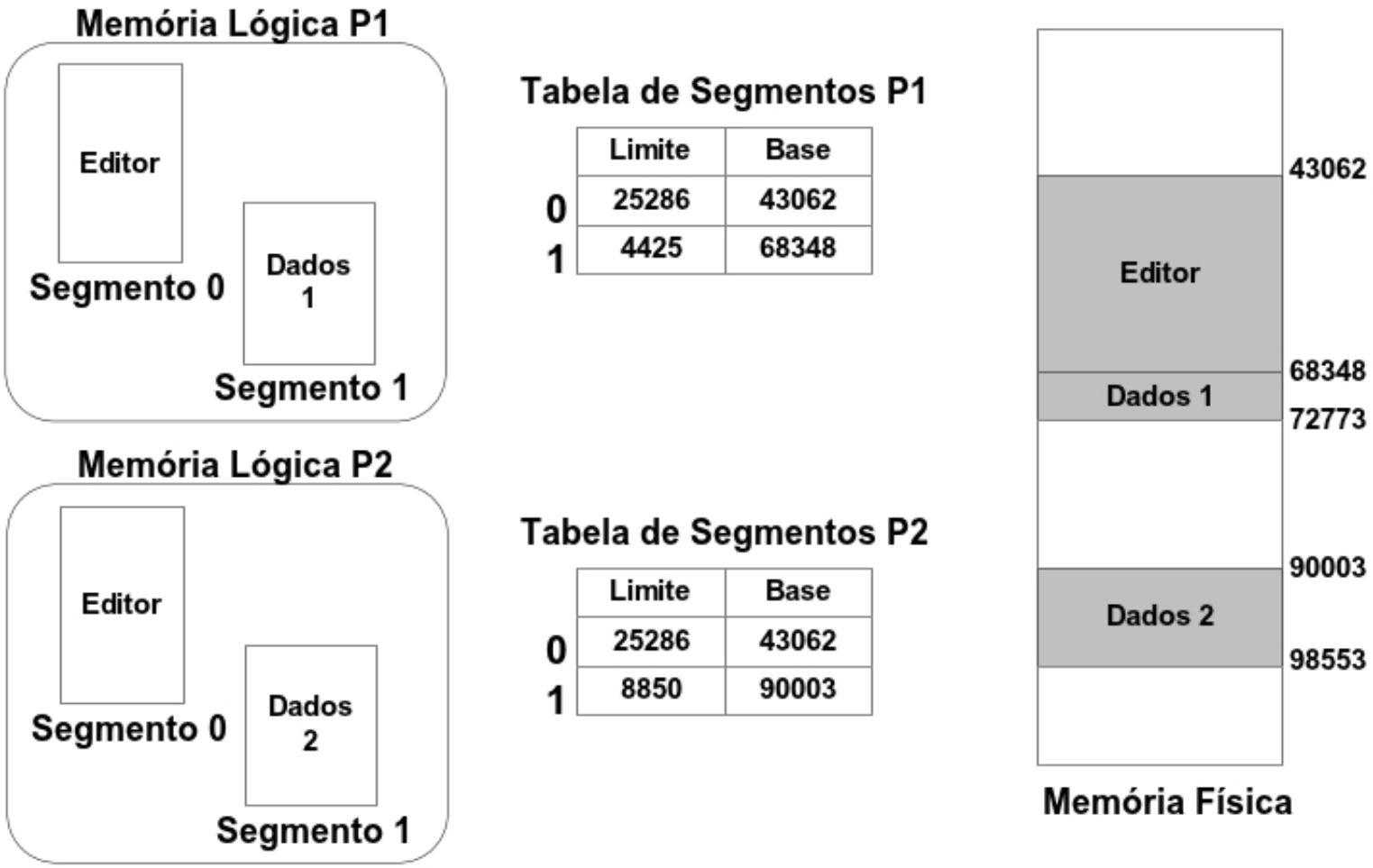
- Implementação
  - Tabelas de segmentos com n linhas, cada qual apontando para um segmento de memória
    - Vários espaços de endereçamento
  - Alocação de segmentos segue os algoritmos já vistos
    - FIRST-FIT, BEST-FIT, NEXT-FIT, WORST-FIT, QUICK-FIT
  - MMU também é utilizada para mapeamento entre os endereços lógicos e físicos
    - Tabela de segmentos informa qual o endereço da memória física do segmento e seu tamanho

Segmentação – Implementação

- Suporte do hardware:
  - Múltiplos pares base/limite, um por segmento
    - Armazenados na tabela de segmentos
    - O número do segmento é usado como índice na tabela
    - Há um número fixo desses pares



Segmentação – Implementação



Segmentação: Problemas Encontrados

- Embora haja espaço na memória, não há espaço contínuo:
  - Política de realocação: um ou mais segmentos são realocados para abrir espaço contínuo
  - Política de compactação: todos os espaços são compactados
  - Política de bloqueio: fila de espera
  - Política de troca: substituição de segmentos
- Não possui fragmentação interna, embora sofra de fragmentação externa

Segmentação: Vantagens

- Facilita proteção dos dados
  - Diferentes segmentos podem possuir diferentes tipos de proteção
- Facilita compartilhamento de procedimentos e dados entre processos (Ex: bibliotecas compartilhadas)
  - Ex: Suponha que cada rotina ocupe um segmento
    - A rotina inicia no endereço (n,0) $\rightarrow$ (segmento, deslocamento)
  - Se mudamos a rotina em (5,0) e recompilamos, nada mais tem que ser mudado (não mudamos nenhum outro segmento)
    - Em um esquema unidimensional, ao aumentarmos uma rotina, podemos mudar o endereço inicial da outra (podemos ter que rever todo o código)

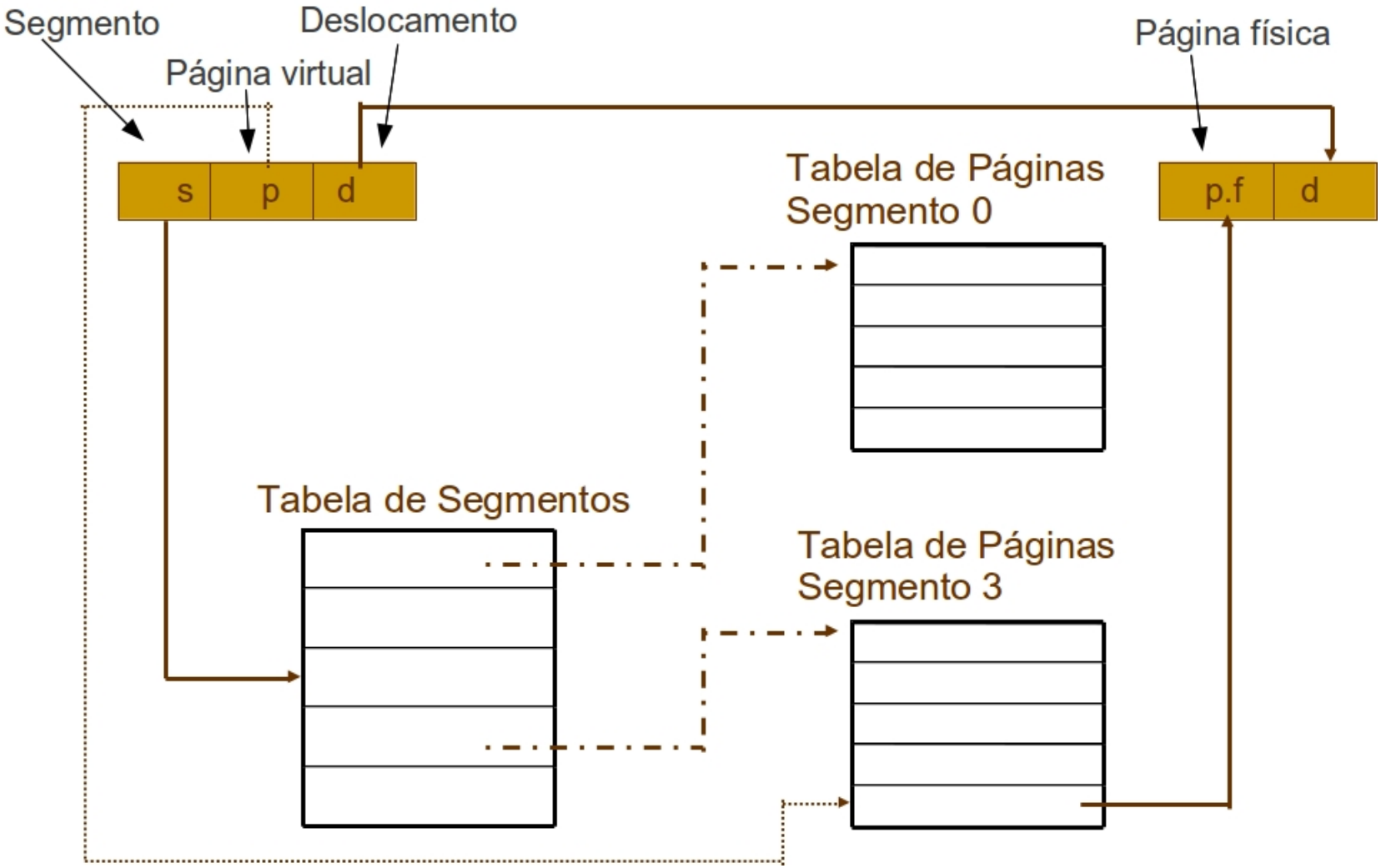
Segmentação Paginada

- Segmentos grandes podem não caber na memória
  - Devemos paginá-los (Ex: pentium)
- Espaço lógico é formado por segmentos
  - Cada segmento é dividido em páginas lógicas
  - Cada segmento possui uma tabela de páginas
    - Usada para mapear o endereço de página lógica do segmento em endereço de página física
  - No endereçamento, a tabela de segmentos indica, para cada segmento, onde sua respectiva tabela de páginas está
    - Semelhante à paginação multinível

Segmentação Paginada

- Ao referenciar um endereço de memória:
  - Endereço virtual: Número do segmento + Número da página + Deslocamento
  - Verifica se o segmento buscado está na memória
    - Há bits específicos para isso na tabela
    - Senão estiver $\rightarrow$ page fault
  - Obtém, na tabela de segmentos, o endereço da tabela de páginas para aquele segmento
    - Usa o endereço da página virtual como offset nessa tabela
  - Da tabela de páginas, obtém o endereço físico da moldura
    - Age como em paginação

Segmentação Paginada



Segmentação Paginada

Consideração	Paginação	Segmentação
O programador precisa saber que essa técnica está sendo usada?	Não	Sim
Há quantos espaços de endereçamento linear?	1	Muitos
O espaço de endereçamento total pode superar o tamanho da memória física?	Sim	Sim
Rotinas e dados podem ser distinguidos e protegidos separadamente?	Não	Sim
As tabelas cujo tamanho flutua podem ser facilmente acomodadas?	Não	Sim
O compartilhamento de rotinas entre os usuários é facilitado?	Não	Sim
Por que essa técnica foi inventada?	Para obter um grande espaço de endereçamento linear sem a necessidade de comprar mais memória física	Para permitir que programas e dados sejam divididos em espaços de endereçamento logicamente independentes e para auxiliar o compartilhamento e a proteção

Entrada e Saída

Entrada e Saída

- Funções específicas do S.O.:
  - Enviar sinais para os dispositivos
  - Atender interrupções
  - Gerenciar comandos aceitos e funcionalidades (serviços prestados)
  - Tratar possíveis erros
  - Prover interface entre os dispositivos e o sistema
- Geralmente, o código para tratamento da entrada e saída representa uma fração significativa do sistema operacional total

Entrada e Saída

Classificação quanto ao Tipo de Conexão

- Conexão serial:
  - Uma única linha de sinal é utilizada para o estabelecimento de toda a conexão, protocolo e transferência de dados, entre o módulo de E/S e o periférico
  - Características principais:
    - Mais barata e mais lenta
    - Relativamente confiáveis
    - Usada em dispositivos mais baratos e lentos, como impressoras e terminais
    - Embora Firewire, USB e Ethernet sejam seriais, se refere geralmente ao padrão RS-232

Entrada e Saída

Classificação quanto ao Tipo de Conexão

- Conexão paralela:
  - Várias linhas de sinais são usadas, de forma que vários bits de dados possam ser transferidos em paralelo
  - É comum que existam linhas independentes para tráfego de sinais de controle
  - Características principais:
    - Mais complexa, mais cara, e mais rápida que a serial
    - Altamente confiável
    - Usada em dispositivos mais velozes, como unidades de disco, fita ou impressoras rápidas

Entrada e Saída

Classificação quanto ao Tipo de Transferência

- Dispositivos de bloco (Block devices):
  - Armazenam informação em blocos de tamanho fixo, cada um com seu próprio endereço
    - Os tamanhos dos blocos variam de 512 B a 32 KB
  - Transferências são feitas com um ou mais blocos consecutivos
  - Cada bloco é lido ou escrito independentemente dos demais
  - Ex: HD, CD-ROM, Bastão USB
    - O sistema de arquivos, por exemplo, trata com dispositivos de bloco abstratos

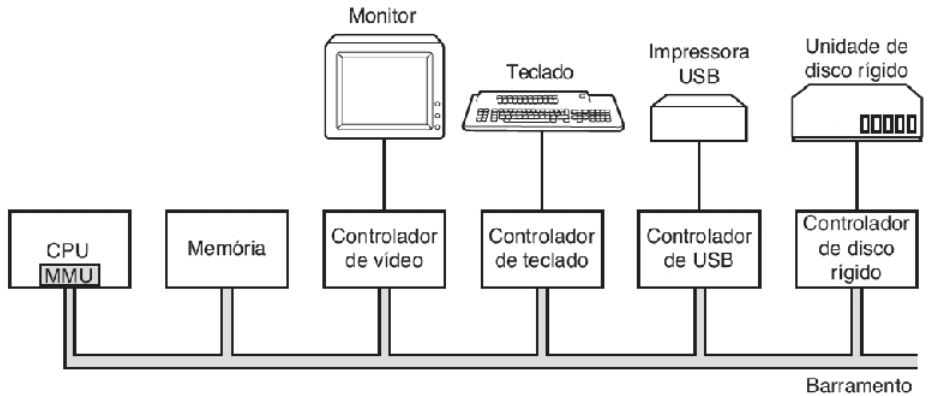
Entrada e Saída

Classificação quanto ao Tipo de Transferência

- Dispositivos de caractere (Character devices)
  - Enviam ou recebem um fluxo de caracteres, sem considerar qualquer estrutura de bloco
  - Não são endereçáveis e não possuem qualquer operação de posicionamento ("seek operation")
  - Ex: impressoras, interfaces de rede, mouses
- Este esquema de classificação não é perfeito
  - O timer (clock) não se encaixa em nenhuma categoria dessas

E/S: Princípios de Hardware

- O S.O. sempre trata com a controladora, não com os dispositivos
  - A Comunicação entre CPU e controladoras é feita através de barramentos comuns (interface de alto nível)
  - Interface entre controladora e dispositivo: baixo nível
  - Mainframes: múltiplos barramentos e processadores especializados em E/S (canais de E/S).



E/S: Princípios de Hardware

- Ex: Controladora de disco
  - Recebe, da unidade de disco, um fluxo de bits, com:
    - Preâmbulo (veremos mais adiante)
    - Os bits do setor (veremos mais adiante)
    - Checksum (Error-Correcting Code – ECC)
  - Converte o fluxo serial de bits (vindo do dispositivo) em um bloco de bytes, executando qualquer correção necessária
    - Monta o bloco de bytes em um buffer interno à controladora
    - Após verificar a checksum, copia o bloco na memória

Princípios de Hardware: Controladores

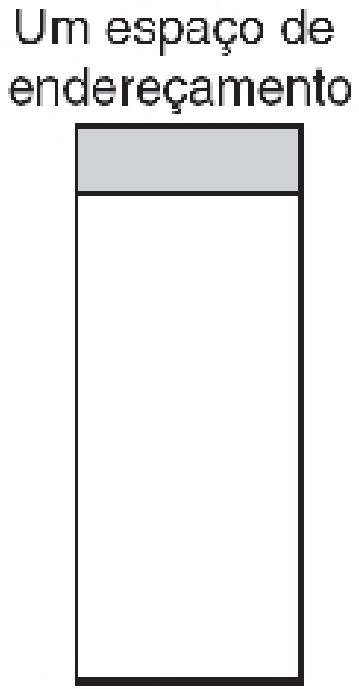
- Cada controlador possui registradores, usados para comunicação com a CPU
  - Ao escrever nesses registradores, o SO pode comandar o dispositivo
    - Executa E/S escrevendo comandos (e seus parâmetros, se existirem) nesses registradores
    - Quando um comando é aceito, a CPU pode deixar que a controladora trabalhe sozinho, indo executar outra tarefa.
    - Quando o dispositivo termina, avisa a CPU através de uma interrupção.
  - Ao ler desses registradores, o SO pode saber o estado do dispositivo etc

Princípios de Hardware: Controladores

- Muitos dispositivos possuem também um buffer, que o SO pode ler (ou escrever). Ex: memória de vídeo
- Como a CPU se comunica com esses registradores e buffers de dados?
  - Necessidade de associação de endereços a estes dispositivos

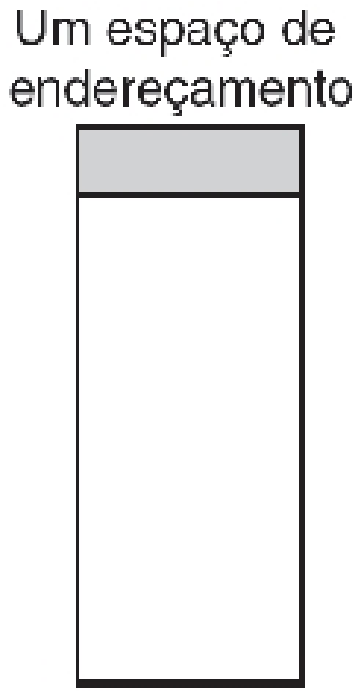
Comunicação CPU – Controladores

- Existem 2 formas de interpretação desses endereços, pelo módulo de E/S, quando o barramento é compartilhado:
  - Mapeado em memória ou em E/S
  - Implementados pelo hardware
- Mapeado em Memória:
  - O módulo de E/S opera dentro do espaço de endereçamento de memória, usando um conjunto de endereços reservados (registradores são tratados como posições de memórias)



Comunicação CPU – Controladores

- Mapeado em Memória:
  - Mapeia todos os registradores de controle no espaço de memória
    - Usa o mesmo barramento para endereçar memória e dispositivos
  - Cada registrador de controle recebe um endereço de memória único
    - Geralmente o topo do espaço de endereços
    - Nenhuma memória é associada a esses endereços
    - Quando um endereço de memória correspondente a um registrador é usado, a CPU faz o desvio

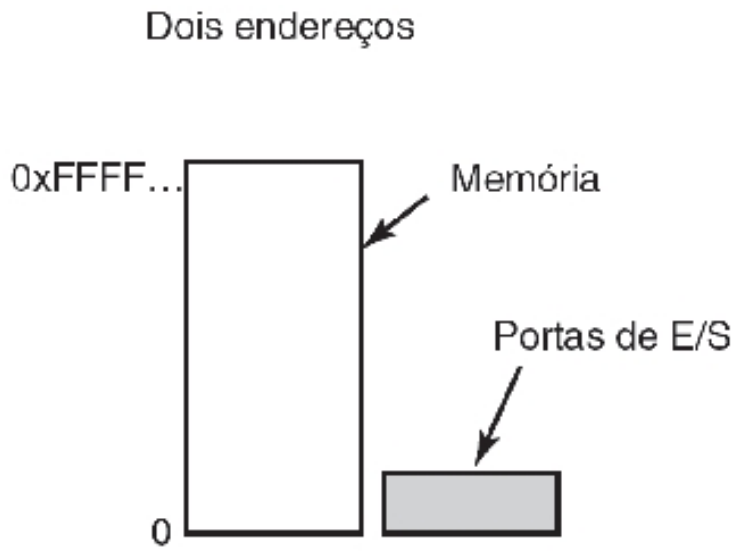


Comunicação CPU – Controladores

- Mapeado em Memória – Vantagens:
  - Não há a necessidade de instruções especiais de E/S
    - A interação se dá como se fosse uma posição de memória comum
  - Para impedir que determinados usuários acessem o dispositivo, basta não mapear as páginas em seu espaço de endereçamento virtual
- Desvantagens:
  - Consome parte do espaço de endereçamento
  - Permitir o uso da cache para os registradores de controle pode ser desastroso
    - O hardware deve poder desabilitar a cache seletivamente (em cada página, por exemplo)

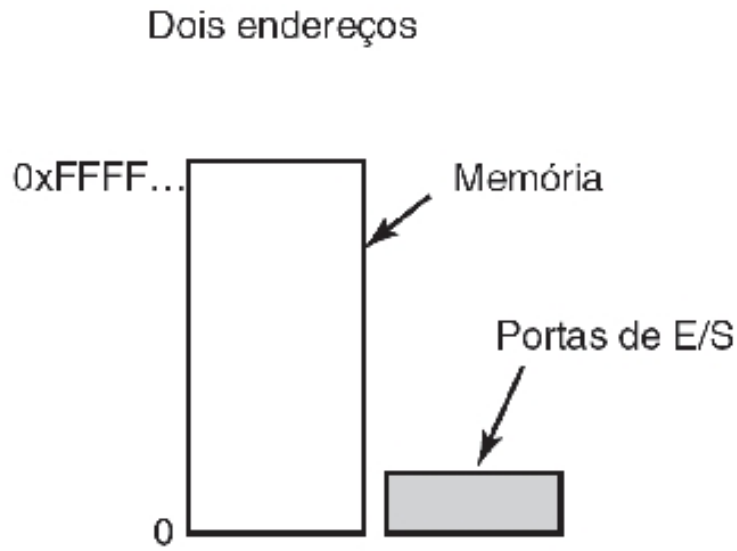
Comunicação CPU – Controladores

- Mapeado em E/S ou E/S isolada (I/O Ports):
  - Cada registrador de controle recebe um número de porta de E/S (inteiro de 8 ou 16 bits)
  - Acessada via instruções especiais (Intel), usadas apenas pelo SO
    - IN REG, PORT → leia do registrador de controle PORT e armazene no registrador REG da CPU
    - OUT PORT,REG → caminho inverso de IN



Comunicação CPU – Controladores

- Mapeado em E/S:
  - Espaços de endereços diferentes para memória e E/S
    - Existe um espaço de endereçamento independente para os dispositivos de E/S
    - IN RO,4 $\rightarrow$ lê da porta de E/S 4 para RO
    - MOV RO,4 $\rightarrow$ lê do endereço 4 da RAM para RO

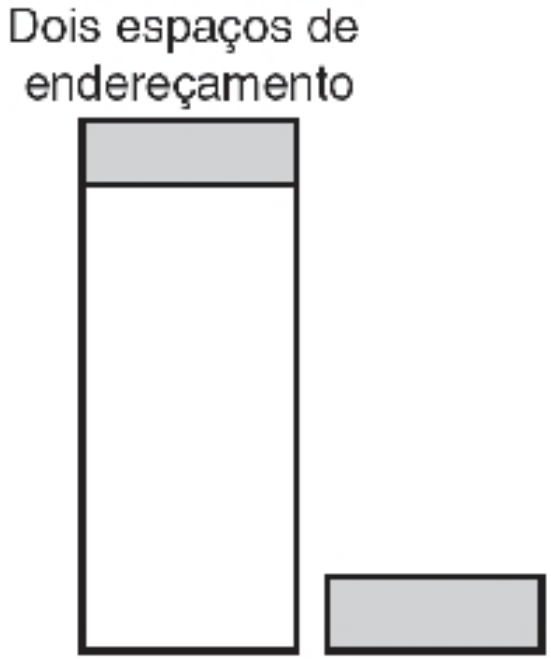


Comunicação CPU – Controladores

- Mapeado em E/S – Vantagens:
  - Não gasta espaço de endereçamento
  - Não sofre com o uso da Cache
- Desvantagens:
  - Requer instruções especiais para acesso
    - Drivers necessariamente tem que usar algum assembly para acessar as instruções

Comunicação CPU – Controladores

- Sistemas híbridos
  - Buffers de dados de E/S mapeados na memória
  - Portas de E/S para os registradores de controle
  - Ex: Pentium
    - Portas de E/S de 0 a 64K-1
    - Endereços de 640K a 1M-1 reservados aos buffers



Comunicação CPU – Funcionamento

- Quando a CPU quer ler uma palavra (E/S ou RAM):
  - Coloca o endereço nas linhas de endereço do barramento
  - Emite um sinal READ sobre uma linha de controle do barramento
  - Em uma segunda linha de sinal informa se o espaço é de E/S ou de memória

Referências Adicionais

- https://www.kernel.org/doc/gorman/html/understand/understand014.html
- http://en.wikipedia.org/wiki/Memory-mapped_I/O
- http://docstore.mik.ua/orelly/linux/run/ch06_02.htm