ESTGF
ESCOLA SUPERIOR DE TECNOLOGIA
E GESTÃO DE FELGUEIRAS
POLITÉCNICO DO PORTO

Gestão de Memória

Sistemas Operativos

Licenciatura em Engenharia Informática

2007/2008

António Pinto (apinto@estgf.ipp.pt)

Sumário

- Gestão de memória
  - Básica
  - Partições fixas
  - *Swapping*
  - Partições variáveis
  - Mapas de bits
  - Listas ligadas
  - *Buddy*
- Reserva de espaço em disco

Contexto

- Gestão da memória disponível compete tipicamente a um só módulo
- Denominado de *memory manager* ou gestor de memória
- Representa a parte do sistema operativo que têm responsabilidade de gerir a utilização da memória
- Actualmente existem algumas técnicas que permitem um aumento da eficácia da gestão de memória
  - *Swapping*
  - *Paging* (Memória virtual)

Contexto
- Funcionalidades do gestor de memória

- Saber que parte da memória está a ser utilizada e por quem
- Saber que parte da memória esta disponível para utilização
- Reservar memória para os novos processos consoante as suas necessidades
- "*Limpar*" zonas de memória que deixem de ser necessárias
- Trocar informação da memória principal para memórias secundárias (disco) quando a informação ocupar mais espaço do que a memória existente (*swapping*)

Contexto
- Tipos de gestão de memória

- Básica — Monoprogramados

- Partições fixas
- Partições variáveis
- Mapas de bits — *Swapping*
- Listas ligadas
- *Buddy*
- Memória virtual — *Paging*

Multiprogramados

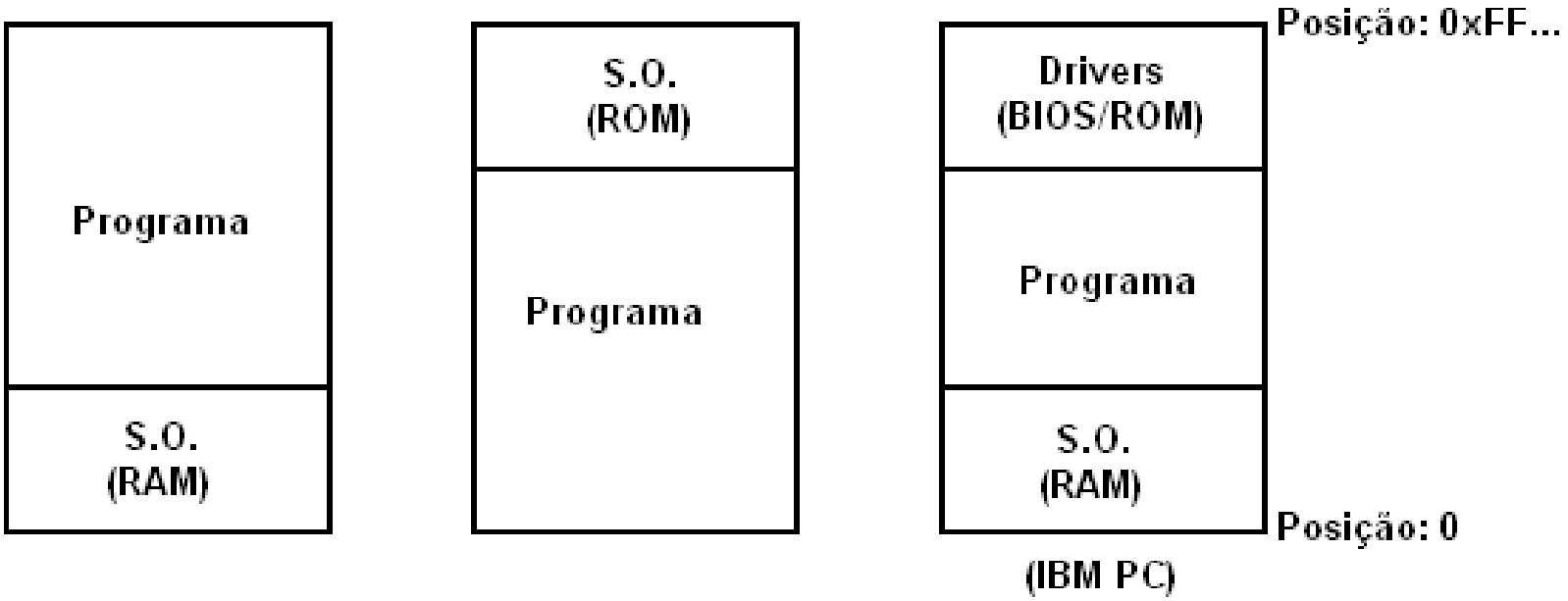
Básica

- Esquema de gestão de memória mais simples possível
- Permite apenas a execução de um processo de cada vez
- Disponibilizada toda a memória não utilizada
- Iniciar a execução do programa, este é carregado na sua totalidade para a memória principal e é executado
- Em cada instante apenas existe um processo em memória (para além do sistema operativo)

Básica

- Técnicas mais usuais consistiam na divisão da memória em duas partes
- Uma para o sistema operativo e outra para o processo do utilizador
- Sistema Operativo poder-se-ia encontrar:
  - Inicio da memória (RAM)
  - Final da memória (ROM)
- Modelo do IBM PC introduz uma terceira divisão, a BIOS
  - *Basic Input Output System*
  - Conjunto básico de *drivers* de dispositivos essenciais, usualmente armazenada em ROM

Básica


Programa
S.O.
(RAM)
S.O.
(ROM)
Programa
Drivers
(BIOS/ROM)
Programa
S.O.
(RAM)
Posição: 0xFF...
Posição: 0
(IBM PC)

Partições fixas

- Execução de mais do que um processo em simultâneo acarreta implicações
- Nomeadamente a necessidade de os manter constantemente em memória
- Como dividir a memória pelos vários processos?
- Divisão de memória por partições fixas assenta na divisão da memória disponível em várias partes (partições)
- Nem todos os programas necessitam da mesma quantidade de memória

Partições fixas

- Divisão é feita em partições de tamanhos diferentes
- Sempre que surge um novo processo, é lhe atribuída a partição disponível com o menor tamanho possível
- Considerando que as partições são de tamanho fixo, dá-se um desaproveitamento da memória
- Parte não utilizada da partição não pode ser utilizada por outros enquanto o processo existir
- Como distribuir os vários processo pelas várias partições de tamanhos variados?

Partições fixas

- Distribuição de processos pelas filas
  - Múltipla fila
    - Distribuição recorrendo a várias filas de processos
  - Fila única
    - Recorrendo a uma única fila de processos

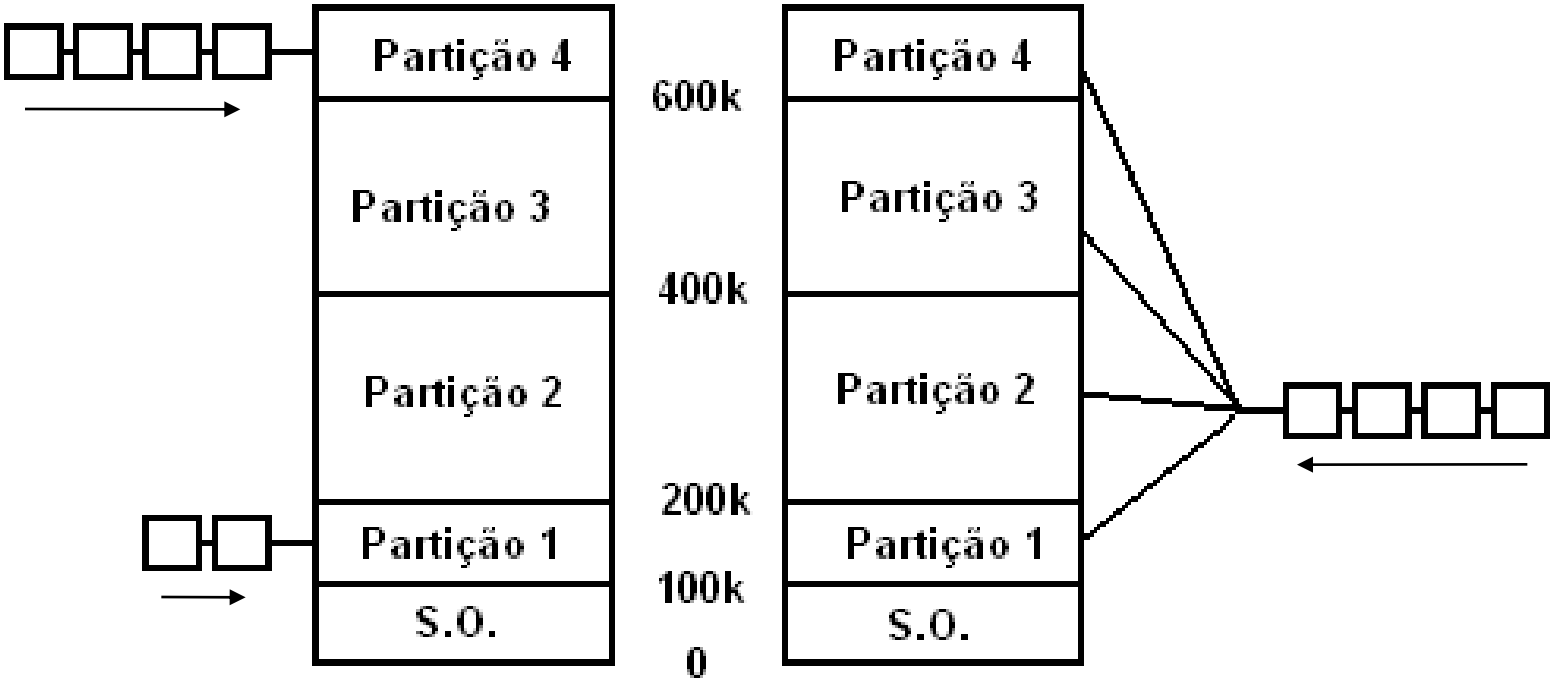
Partições fixas
- Múltipla fila

- Cada processo é atribuído á fila da partição menor onde este poderá existir
- Possibilita alguma ineficiência
- Poderá levar à possibilidade de existirem processos em espera numa fila para uma partição menor
- Estando uma fila para uma partição maior vazia

Partições fixas
- Fila única

- Cada processo é atribuída a partição livre de menor capacidade onde este poderá existir
- Poderão surgir grandes desperdícios de espaço
- Necessários esquemas de escalonamento de memória mais complexos

Partições fixas



Swapping

- Consiste na troca entre memória e disco das imagens dos processos existentes em cada instante
- Passagem de memória para disco denomina-se de *swap out*
- Passagem de disco para memória denomina-se de *swap in*
- *Swap out* dos processos em espera
- *Swap in* dos próximos processos a executar pelo processador
- Introduz dois novos problemas a "*realocação"* e a protecção

Swapping

- Realocação
  - Consiste na utilização de referências relativas de posições de memória.
  - Após compilado e carregado para memória, se um programa necessitar de efectuar uma chamada a uma função que existe na memória na endereço de memória 0x50
  - Se esta posição for estática, o processo tenta endereçar uma posição de memória (0x50) que tipicamente fica na zona de memória do sistema operativo (gera um erro de protecção)
  - Forma correcta é a de endereçar a partir de uma referência (ex.: INICIO + 0x50)
  - Codificação endereços de memória relativos a um registo dentro do CPU

Swapping

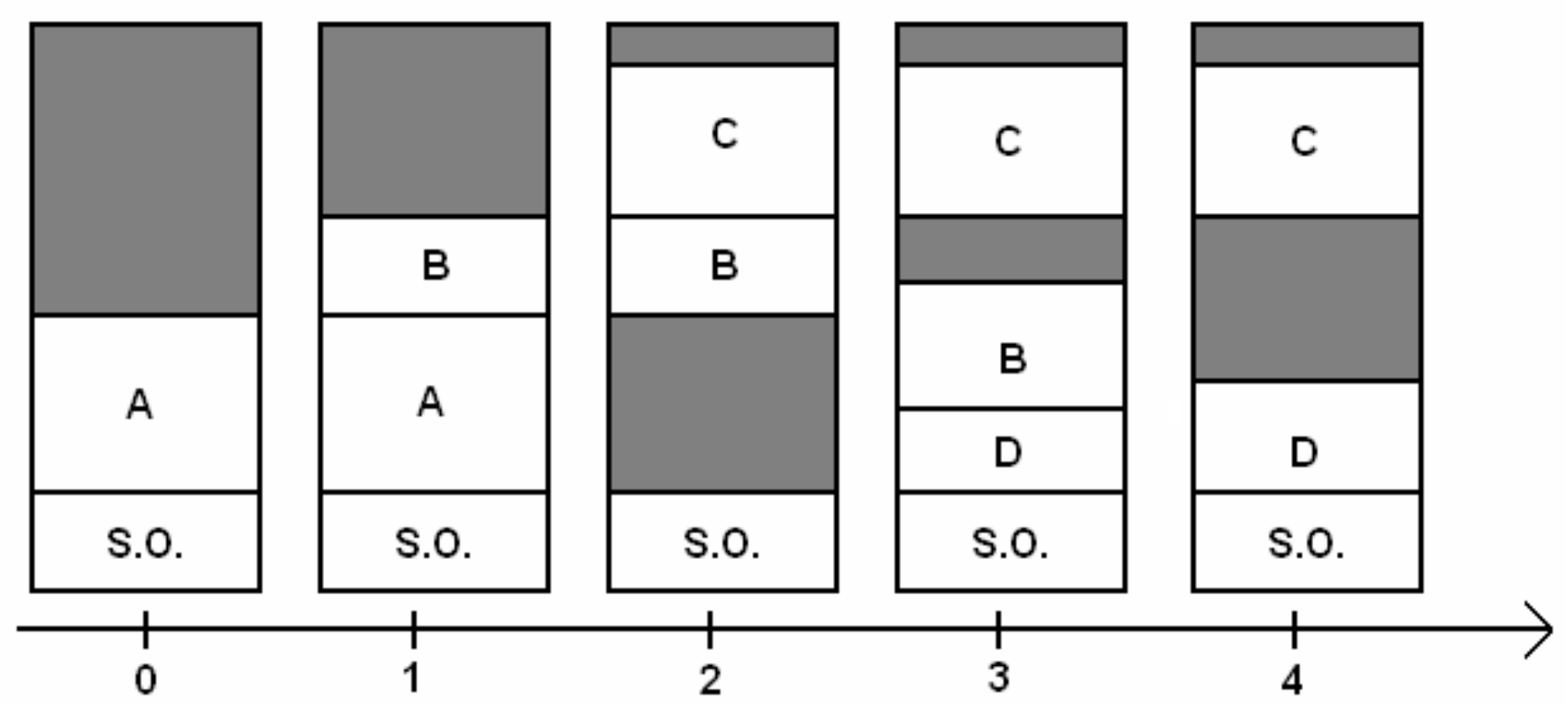
- Protecção
  - Garantir que cada processo não pode aceder a partições de memória de outros processos
  - Impedir o acesso a partições de memória de outros processos
  - Caso contrário poderão surgir incoerências de informação e erros de execução
  - Uma solução passa novamente pela utilização de registos de CPU
  - Um para indicar a posição inicial da partição
  - Outro para indicar o seu tamanho

Partições variáveis

- Utiliza partições de memória de tamanho variável para conter os processos
- Número e tamanho dos processos em memória varia constantemente
- Suporta a alteração dinâmica do tamanho das partições que contém cada processo
- Maximiza a utilização da memória disponível à custa de um aumento de complexidade nas operações de
  - Reserva e libertação de memória
  - Registo de memória utilizada

Partições variáveis

Quanta memória reservar para cada processo?

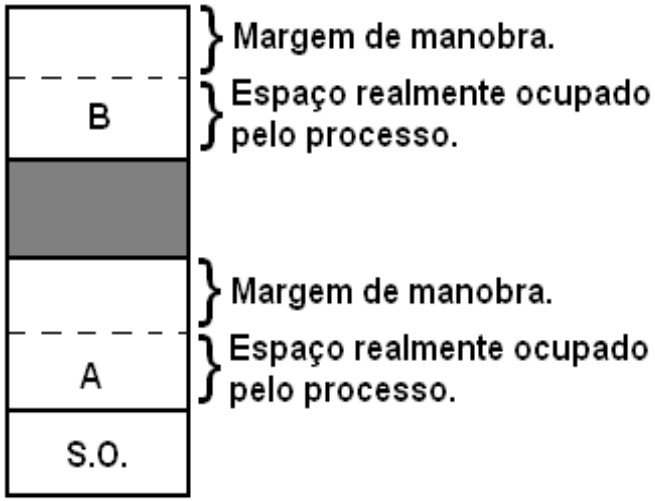


Partições variáveis

- Processo tiver uma necessidade estática de memória, o problema resolve-se facilmente, reservando exactamente a quantidade necessária
- Contudo os processos poderão variar a sua necessidade de memória ao longo da sua execução
- Tal variação só é possível se existir um espaço de memória livre e adjacente ao espaço de memória ocupado pelo processo
- Caso contrário não será possível ao processo crescer, excepto se se recorrer
  - Reposicionamento do processo num novo espaço de memória, com maior capacidade
  - *Swap out* do processo adjacente

Partições variáveis

- Boa prática é a de se reservar um pouco mais de memória, além da que o processo necessita inicialmente
- Facilita a gestão de memória
- Reduz o *overhead* necessário
- Implica uma menor ocorrência
  - Operações de *swap*
  - Operações de reposicionamento de processos na memória



Partições variáveis

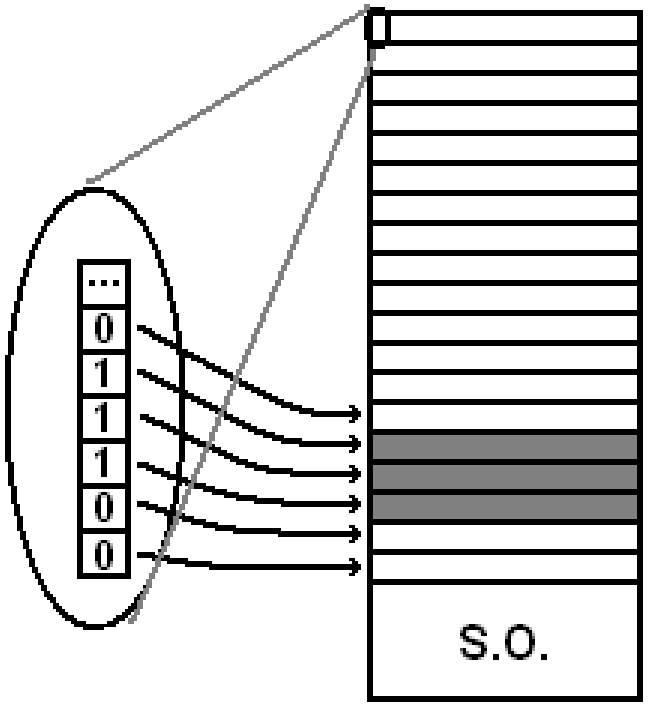
- Operações de *swap out*, implicam uma necessidade fixa de espaço
- Pode-se então reservar exactamente o espaço ocupado pelo processo
- Processos estão suspensos, logo não as suas necessidade de memória não vão crescer

Mapas de bits (*bitmaps*)

- Assenta na divisão da memória em unidades de alocação
- Mapa com tantos bits, quantas as unidades de alocação disponíveis
- Cada bit do mapa representa a ocupação de uma unidade de alocação
- Exemplo
  - Valor 1 (um) significa ocupado
  - Valor 0 (zero) significa livre

Mapas de bits (*bitmaps*)

Qual o tamanho correcto para cada unidade de alocação?



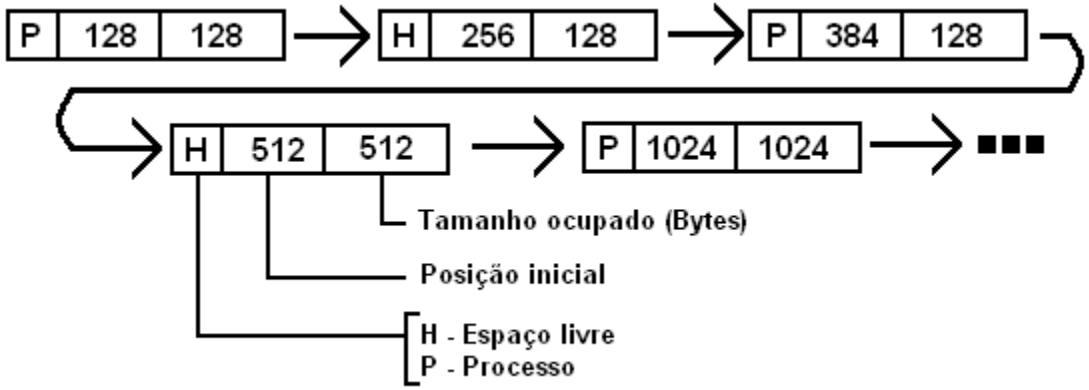
Mapas de bits (*bitmaps*)

- Se a unidade de alocação for pequena, então o mapa de bits é grande
- Se a unidade de alocação for grande, então o mapa de bits é pequeno mas existe um grande desperdício de memória
- Pode surgir a necessidade de reservar memória para um processo que ocupe **n** unidades de alocação
- Forçando o sistema a percorrer todo o mapa de bits à procura de **n** unidades de alocação contíguas e livres
- Necessidade constante de pesquisa têm um grande impacto no desempenho
- Implicou a sua pouca utilização

Listas ligadas

- Baseia-se na manutenção de uma lista ligada
- Cada elemento dessa lista representa um segmento de memória (ocupado ou livre)
- Ordenação crescente do endereçamento é recomendada
- Simplifica a actualização da lista numa situação de término de processo,ou de *swap out*

Listas ligadas



- Substituição de um P por um H quanto o processo em questão está entre dois processos (muito simples)

- Processo a retirar da lista está contíguo com uma zona de memória livre é necessário juntar as zonas de memória livre numa só (+ complexo)

Listas ligadas

- Gerir a reserva de memória para processos novos (ou processos *swapped in*) pode levantar alguns problemas
- Existem vários algoritmos para efectuar esta gestão
  - First Fit
  - Next Fit
  - Best Fit
  - Worst Fit

Listas ligadas
- *First fit*

- Algoritmo mais simples
- Percorre a lista à procura do primeiro espaço livre com capacidade para conter o processo
- Bastante rápido, já que procura o mínimo possível
- Elemento da lista encontrado será dividido em dois
  - Um para representar o espaço ocupado pelo processo
  - Outro para representar o restante espaço de memória ainda disponível

Listas ligadas
- *Next fit*

- Assenta no *First fit*
- Contém apenas uma alteração
- *First fit* procura sempre a partir do primeiro elemento da lista
- *Next fit* guarda a posição onde encontrou o último espaço e na próxima pesquisa começa nesse ponto

Listas ligadas
- *Best fit*

- Procura toda a lista até encontrar um segmento de memória com o tamanho mais aproximado possível capaz de conter o processo

- Grande consumo de tempo na pesquisa

- Tende a gerar muitos pequenos segmentos de memória que não são utilizados posteriormente

Listas ligadas
- *Worst fit*

- Neste caso procura-se pelo maior segmento de memória livre

- Assim tenta-se que o restante espaço ainda possa ser utilizado por outro processo

Listas ligadas

- Qualquer um destes algoritmos poderá ser melhorado recorrendo a listas separadas para
  - Segmentos de memória ocupados
  - Segmentos de memória livres
- Reserva de memória para um processo é rápida
- Libertação de um segmento de memória é bastante mais demorada
- Obriga a analisar os segmentos contíguos de memória livre que foram gerados
- Avaliar se é necessário (ou possível) juntar segmentos num só

Buddy

- Computadores utilizam números binários para endereçar a memória
- Visando facilitar a junção de espaços de memória livre gerados pela remoção de um processo
- Poder-se-á utilizar esse facto como vantagem
- *Buddy* utiliza um conjunto de listas de segmentos de memória livre
  - Uma lista mantém apenas os segmentos de 1 byte;
  - Outra de 2 bytes;
  - Outra de 4 bytes;
  - etc.;
  - Até obter uma lista de segmentos de tamanho igual ao da memória total

Buddy
- Exemplo

- Considerando um sistema com 1 Mb de memória, no qual inicialmente toda a memória está disponível

- Serão necessárias 21 listas de segmentos de memória disponível (1, 2, 4, ..., 1024)

- Apenas a lista de 1 Mb contém um elemento

Buddy
- Exemplo

- Considerando um sistema com 1 Mb de memória, no qual inicialmente toda a memória está disponível
- Serão necessárias 21 listas de segmentos de memória disponível (1, 2, 4, ..., 1024)
- Apenas a lista de 1 Mb contém um elemento
- Assumindo agora a necessidade de reservar memória para um processo com 70 KB

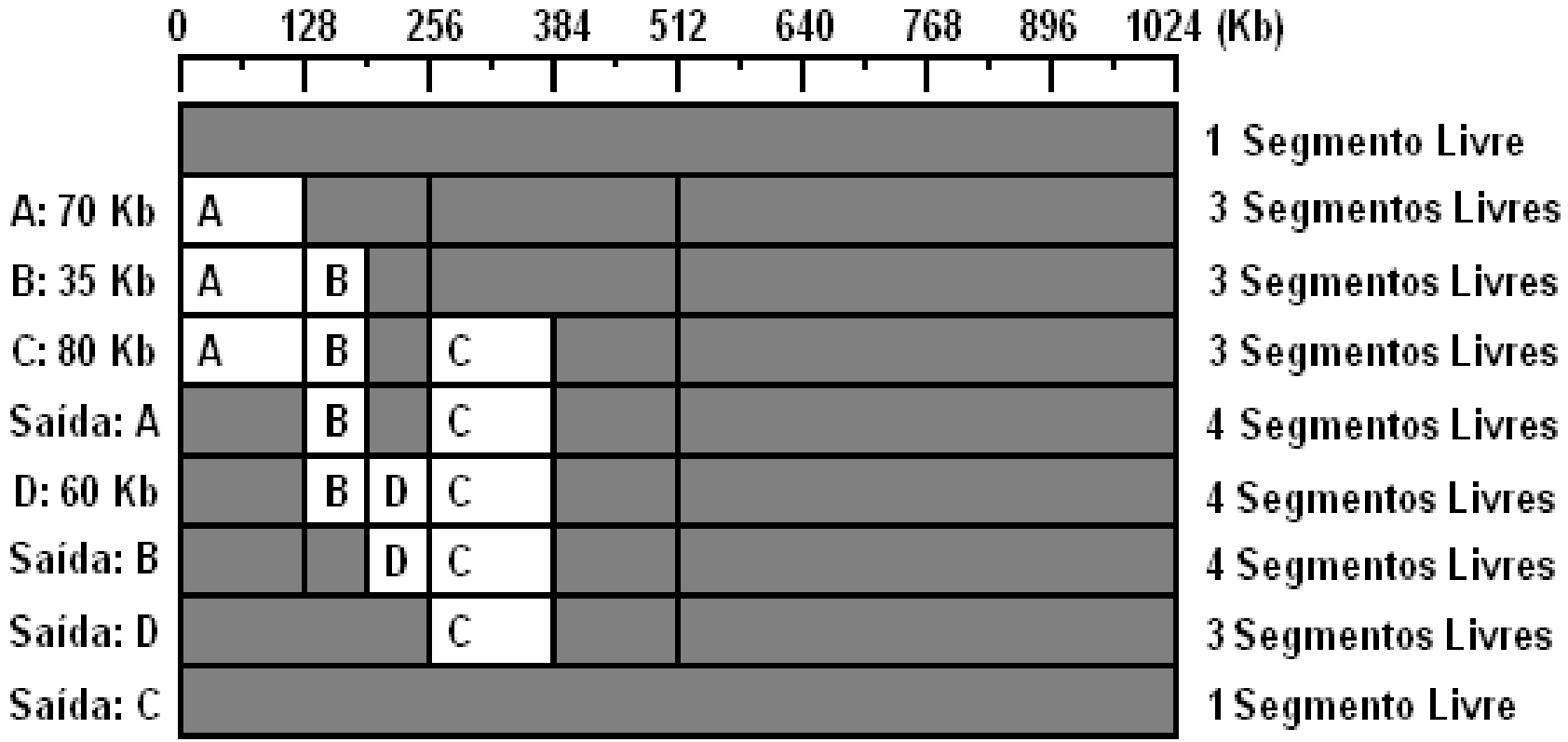
Buddy
- Exemplo

- Próximo passo é o de calcular o tamanho de memória necessário
- São necessários neste caso 128 KB de espaço de memória
- 128, porque é a potência de 2 mais baixa com capacidade para conter 70 KB necessários
- Não há blocos de 128 Kbytes, nem de 256 Kbytes, nem de 512 Kbytes

Buddy
- Exemplo

- Necessário dividir o segmento inicial de 1 Mb em dois de 512 KB
- Um destes novos segmentos será então dividido em dois novos segmentos de 256 KB
- Um destes será também dividido em 2 de 128 KB

Buddy
- Exemplo



Buddy

- Vantagem deste sistema advém da facilidade de libertação de memória
- Supondo que se pretende libertar um bloco com 128 KB
- Basta analisar a fila de segmentos livres de 128 KB para saber se é possível efectuar a junção de segmentos de contíguos
- Desvantagem reside no fraco aproveitamento de memória
- Todos os segmentos são arredondados por excesso até à próxima potência de dois

Reserva de espaço em disco

- Necessário reservar espaço em disco para o armazenamento temporário dos processos
- Alguns sistemas reservam espaço em disco para um processo quando este é criado
- Mantendo o espaço reservado enquanto o processo existir
- Quando o processo é *swapped out*, é armazenado sempre na mesma posição do disco

Reserva de espaço em disco

- Sistemas fazem uma gestão em que um processo quando está em memória não ocupa espaço em disco
- Leva a uma menor necessidade de espaço em disco, mas implica a gestão dos processos em disco
- Algoritmos para a gestão dos processos em disco nestas situações, são os mesmos que os utilizados para a gestão dos processos em memória
- Única diferença consiste na não variação do espaço a reservar enquanto o processo está em disco
- No entanto, o tamanho a reservar tem de ser múltiplo do tamanho de bloco do disco

Exercícios de Exame
Exercício 1

- Considere um computador com 2MB de memória que utiliza um sistema operativo que faz a gestão de memória pelo algoritmo *Buddy*. Apresente uma representação de como a memória ficaria dividida após a seguinte lista de acontecimentos:
  - Chegada de um novo processo (P1) com 77K tamanho;
  - Chegada de um novo processo (P2) com 33K tamanho;
  - Chegada de um novo processo (P3) com 255K tamanho;
  - Chegada de um novo processo (P4) com 27K tamanho;
  - Chegada de um novo processo (P5) com 62K tamanho;
  - Saída do processo P1;
  - Chegada de um novo processo (P6) com 27K tamanho;
  - Saída do processo P3;
  - Saída do processo P2;
  - Chegada de um novo processo (P7) com 17K tamanho;

Exercícios de Exame
Exercício 2

- Considerando que num dado instante de tempo um sistema operativo dispõem da seguinte lista de partições de memória:

  [h|4K]→ [h|7K]→ [h|3K]→ [h|4K] → [h|15K]→ [h|18K]→ [h|10K]→ [h|8K]

- Apresente a lista resultante da aplicação dos algoritmos estudados *worst-fit* e *next-fit* para a lista de requisições de memória: 7K, 4K, 12k, 12k, 3K, 7K, 8K.