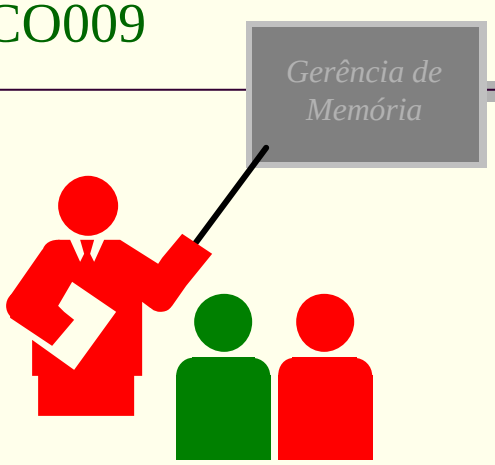
Modulo III
Gerência de Memória

Prof. Ismael H F Santos

Ementa

- Gerência de Memória
  - Introdução
  - Alocação Contígua
  - Alocação Particionada Estática
  - Swapping
  - Alocação Particionada Dinâmica
  - Políticas de Escolha de Partições
  - Fragmentação x Compactação

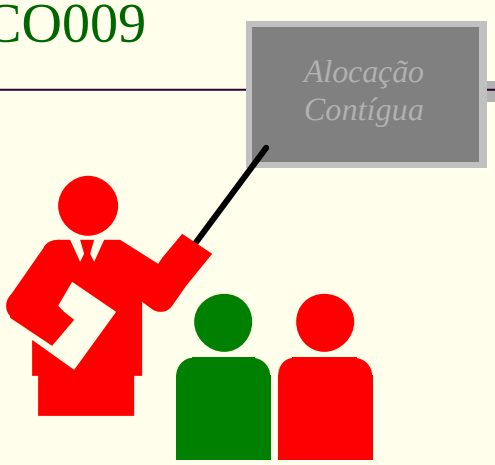
SOP – CO009



Gerente de Memória

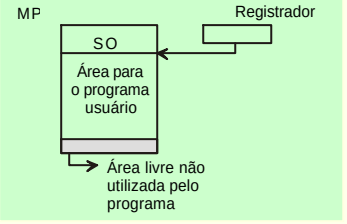
- **Objetivos da Gerência de Memória**
  - i. Tornar o mais eficiente possível o compartilhamento de memória entre os processos.
  - ii. Impedir que um processo acesse área de memória que não lhe pertence.
  - iii. Facilitar alocação de memória.
  - iv. Recuperar a memória liberada pelos processos.

SOP – CO009



Alocação Contígua

- **Alocação Contígua Simples**
  - *· A MP é dividida em 2 partes: uma para o SO e outra para o programa do usuário.*



Problemas:

· não permite a utilização eficiente da UCP e da MP, pois apenas um processo pode utilizar esses recursos.

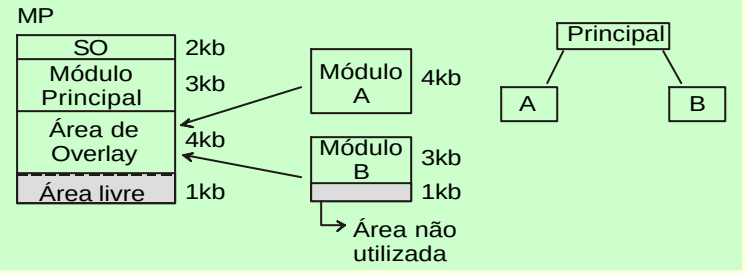
· a princípio os programas estavam limitados ao tamanho da MP disponível.

Alocação Contígua

- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory
- Single-partition allocation
  - Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data
  - Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register

Alocação Contígua

- **Alocação Contígua Overlay**



· *A **técnica de overlay** divide o programa em módulos, de forma que cada parte possa executar independentemente uma da outra, utilizando uma mesma área de memória*

SOP – CO009

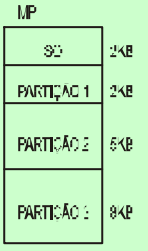
Alocação Particionada Estática

Alocação Particionada Estática

- **Alocação Particionada Estática**
  - *Partições de tamanho fixo estabelecendo na fase de inicialização do sistema (BOOT) em função dos programas que normalmente executam no ambiente.*
  - *Os programas só podiam executar em uma das partições, mesmo estando as outras livres. Essa limitação se devia aos compiladores e montadores que geravam apenas código absoluto o que gerou a* ***Alocação Particionada Estática Absoluta.***

Tabela de Partições

PART	INÍCIO	TAM
1	2KB	2KB
2	4KB	5KB
3	9KB	8KB



Alocação Particionada Estática

- **Alocação Particionada Estática**

  Problema: *O problema da alocação particionada estática absoluta é que os programas só podem rodar na partição para qual foram compilados mesmo que existam outras partições livres.*

- **Alocação Particionada Estática - Relocação**

  · *O problema de determinados programas só rodarem em uma partição passou a ser crítico, pois trazia uma grande ineficiência para o sistema. Somente com a evolução dos compiladores, ligadores e loaders, a geração de código relocável foi possível dando origem a um novo tipo de alocação chamada* ***Alocação Particionada Estática Relocavel.***

Alocação Particionada Estática - Relocação

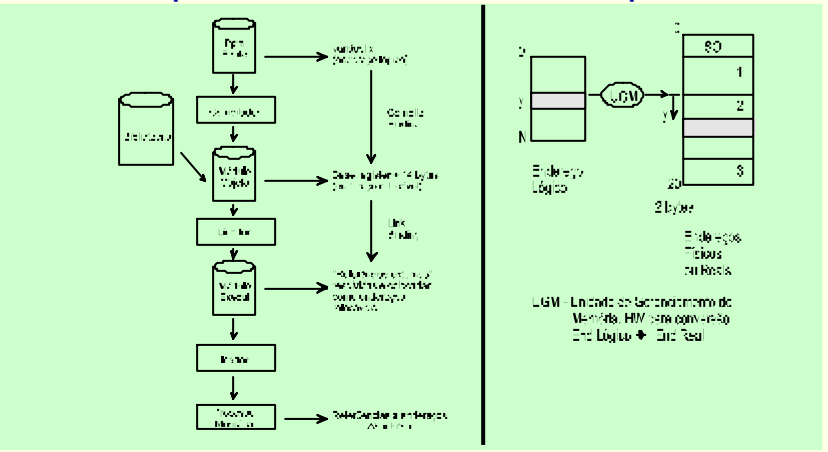
- **Alocação Particionada Estática- Relocação**

  · *A* ***técnica de relocação*** *consiste na utilização de um registrador (fence register) como endereço base a partir do qual todos os endereços são referendados. O valor desse registrador é somado a todo endereço gerado pelo processo do usuário em execução.*

  · *Até ser executado um programa do usuário passa por diversas fases. Endereços são representados de formas diferentes em cada fase conforme veremos a seguir.*

Binding of Instructions and Data to Memory

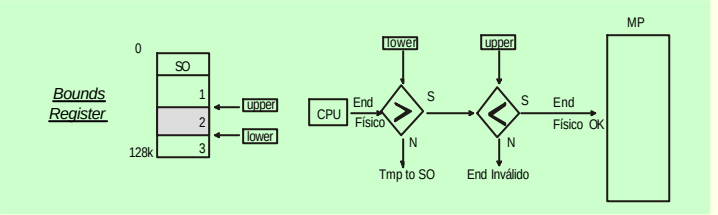
- **Alocação Particionada - Relocação**



Alocação Particionada Estática - Relocação

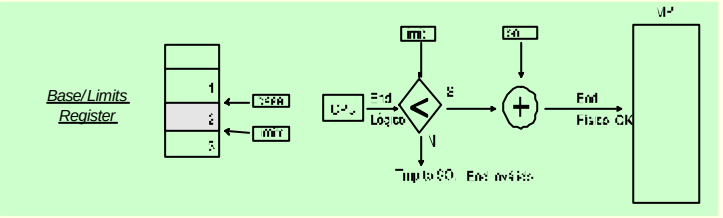
- **Alocação Particionada - Relocação**

  - *Na **Alocação Particionada**, a **tabela de partições** contém um bit de status informando se a partição está ou não livre para ser utilizada. A proteção, nesse esquema de alocação de memória, é feita com o uso de dois registradores e pode ser implementada de duas maneiras:*



Alocação Particionada Estática - Relocação

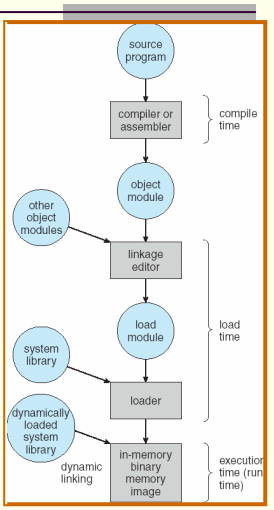
- **Alocação Particionada Estática- Relocação**



· *O HW da UGM é diferente para os dois casos. No caso i nós temos a chamada* ***Relocação Estática*** *(gerada em tempo de carga do programa para execução). No caso ii nós temos a* ***Relocação Dinâmica****, a qual exige a soma do registrador base a cada endereço referendado o que representa um custo adicional a execução que pode ser compensado com técnicas de* ***overlapping de instruções no processador.***

Binding of Instructions and Data to Memory

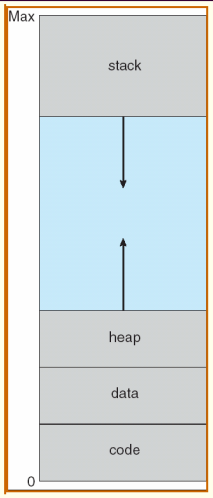
- **Address binding of instructions and data to memory addresses can happen at three different stages**
  - **Compile time**: If memory location known a priori, *absolute code* can be generated; must recompile code if starting location changes
  - **Load time**: Must generate *relocatable code* if memory location is not known at compile time
  - **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., *base* and *limit registers*).



Logical vs. Physical Address Space

- The concept of a logical *address space* that is bound to a separate *physical address space* is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as *virtual address*
  - **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

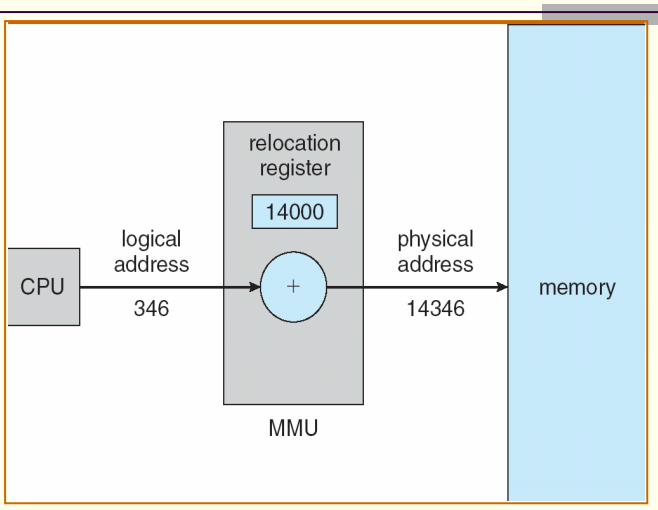
Virtual-address Space



Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses

Dynamic relocation using a relocation register



Dynamic Loading

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded
- Useful when large amounts of code are needed to handle infrequently occurring cases
- No special support from the operating system is required implemented through program design

Dynamic Linking

- Linking postponed until execution time
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine
- Stub replaces itself with the address of the routine, and executes the routine
- Operating system needed to check if routine is in processes' memory address
- Dynamic linking is particularly useful for libraries

Alocação Particionada Estática - Problemas

- **Alocação Particionada Estática Problemas**
  - *O problema básico da **Alocação Particionada Estática**, inicialmente adotada pelo OS/MFT (multiprogramming with a fixed number of tasks) da IBM é quanto à escolha do tamanho das partições para melhor atender os requerimentos de memória dos jobs atualmente processados. O **throughput** do sistema de computação é em geral proporcional ao nível de multiprogramação, que é afetado pela maneira como gerenciamos o uso da memória.*

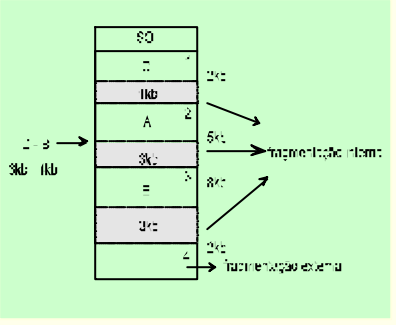
Fragmentação

- **Alocação Particionada Estática - Fragmentação**

  Fragmentação

  interna *- memória requerida pelo processo é menor que a disponível na partição.*

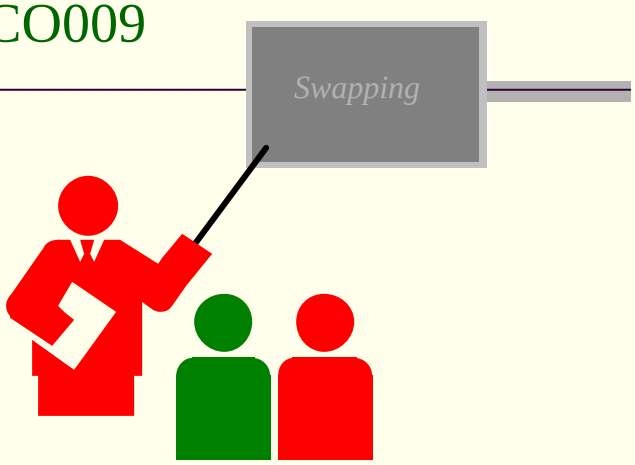
  externa *- partição livre mas não utilizada por falta de espaço.*



Fragentação Total = å Frag Int +å Frag Ext = (1+3+3) + 2 = 9Kb

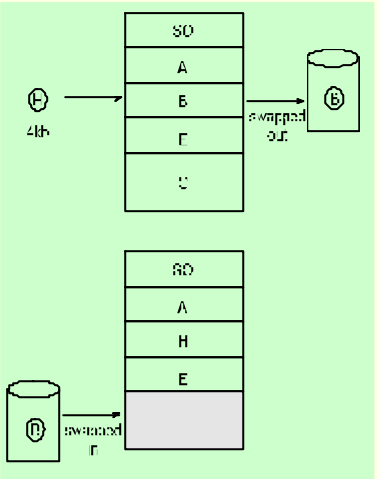
%Memória perdida = FragTot/MemTot = 9/(2+5+8+2) = 50 %

SOP – CO009



Swapping

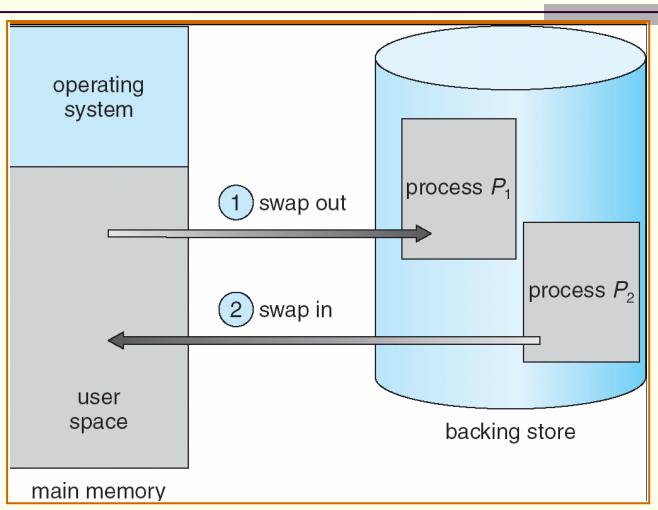
- **Swapping**
  - *Técnica para permitir que programas que esperam por memória livre possam ser processados. Nesta situação, o SO escolhe um programa residente que é levado da MP para o disco (**swapped out**) retornando posteriormente para a MP (**swapped in**) como se nada tivesse ocorrido.*



Swapping

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution
- **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
- Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows)

Swapping



Swapping

- **Swapping**
  - *Sempre que o **Escalonador** (**CPU scheduler**) decide executar um processo ele chama o **Dispatcher**. O **Dispatcher** verifica se o processo a ser executado está residente em memória, se não, ele remove (swap out) algum processo que esteja correntemente na MP e carrega (swap in) o processo escolhido para ser executado. Após isto o contexto do processo a ser executado é restaurado e o controle é passado ao processo que passa então a executar.*
  - ***Exemplo:** **Calcule o** D**Tswap = ?***

    Tamanho do processo = 1 Mb

    Taxa de transferência = 100 Mbits/s

    Tempo de seek= 10 msec, Disco = 7200 rpm

Swapping (cont.)

7200 voltas ® 1 minuto = 60 s

t 1volta *® 60/7200 s*

*Assumimos **Tlatência = ½ t 1volta** = ½ (60/7200) = 1/240*

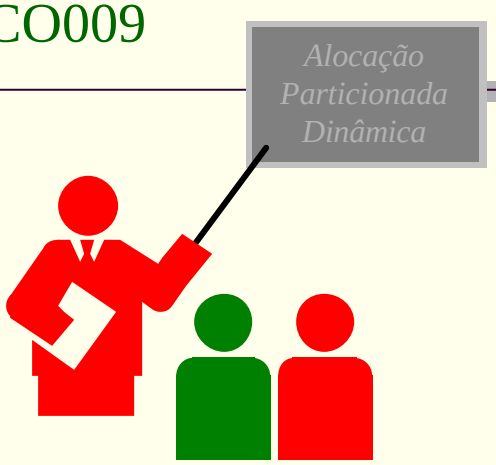
Ttransf disco-memória = Tseek + Tlatência + Ttransf =

= 10 x 10 $^{-3}$ + 1/240 + (1Mb x 8)/100 Mbits /s =

= 0,01 + 0,042 + 8/100 = 0,01 + 0,042 + 0,08 = 0,132 s

D ***Tswap = 2 * Transf disco-memória*** *= 0,26 seg*

SOP – CO009

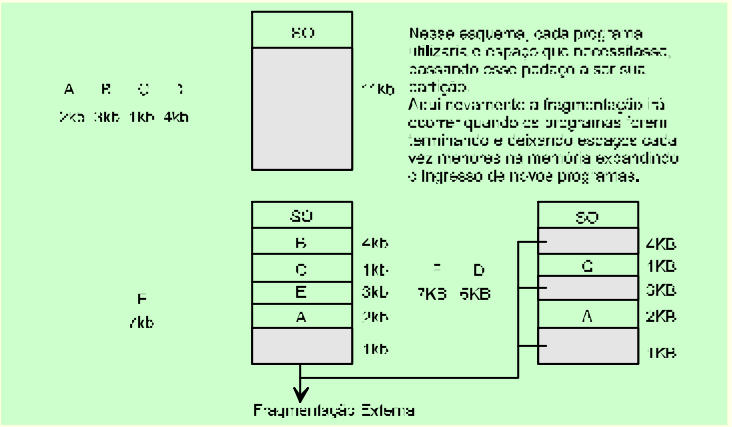


Alocação Particionada Dinâmica

- **Alocação Particionada Dinâmica**
  - *O problema principal da **Alocação Particionada Estática** era a escolha do número e tamanho das regiões de forma a diminuir a fragmentação (interna e externa). É praticamente impossível definir estes valores devido a diversidade dos jobs presentes em um Sistema de Computação. A solução é permitir que as regiões (partições) variem seu tamanho de forma dinâmica que é a chamada **Alocação Particionada Dinâmica** (ou variável) adotada pelo OS/MVT (multiprograming with a variable number of tasks) da IBM.*

Alocação Particionada Dinâmica

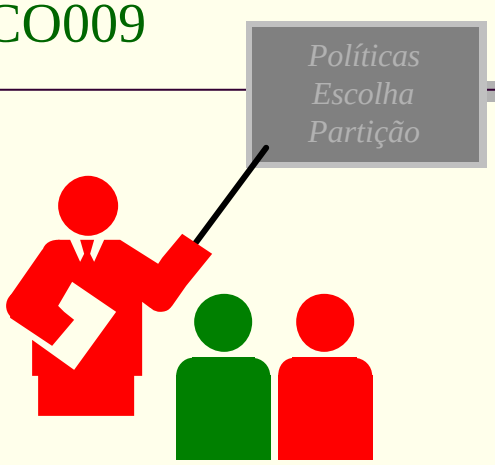
- **Alocação Particionada Dinâmica**



Alocação Particionada Dinâmica

- **Alocação Particionada Dinâmica**
  - *Apesar do esquema MVT ser melhor que o MFT, apresentando em geral uma menor fragmentação, o problema de* ***Fragmentação Externa*** *no MVT pode se tornar crítico. No pior caso podemos ter um bloco livre (não utilizado) entre cada dois processos. Se toda essa memória fosse reunida em um único bloco livre mais processos poderiam rodar nesta nova partição. A* ***política de escolha da partição*** *também afeta a* ***Fragmentação.***
  - *Políticas de escolha da Partição se dividem em:* ***FIRS-FIT, BEST-FIT*** *e* ***WORST-FIT****.*

SOP – CO009



Políticas de escolha da Partição

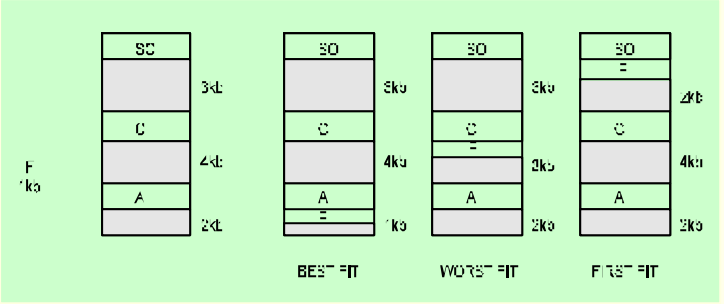
- **How to satisfy a request of size n from a list of free holes**
  - **First-fit**: Allocate the *first* hole that is big enough
  - **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
  - **Worst-fit**: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.
- **First-fit and best-fit better than worst-fit in terms of speed and storage utilization**

Políticas de escolha da Partição

- ***First-Fit*** *- escolha a primeira partição livre, de tamanho suficiente para carregar o processo. O Buraco é dividido em duas partes, um para o processo e outro para o novo buraco. É um algoritmo rápido pois ele não faz muitas pesquisas na lista. Existe uma variação do FIRST FIT chamada a* ***NEXT FIT*** *que começa a pesquisa na lista a partir da última posição encontrada ao invés de começar desde o princípio*
- ***Best-Fit*** *- escolhe a melhor partição, i.e., aquela em que o programa deixa o menor espaço sem utilização. Nesse algoritmo, a lista está ordenada por tamanho, diminuindo o tempo de busca por uma área desocupada. A desvantagem é que cada vez mais a memória fica com pequenas áreas não contíguas, aumentando o problema de fragmentação externa.*

Políticas de escolha da Partição

- ***Worst-Fit*** *- a idéia deste algoritmo é tentar diminuir o problema da fragmentação externa gerada pelo* ***BEST FIT*** *escolhendo a partição que deixa o maior espaço sem utilização.*

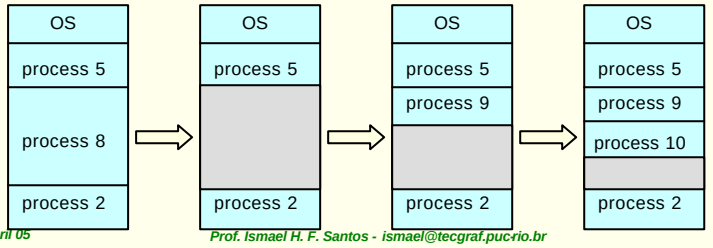


SOP – CO009

Fragmentação x Compactação

Fragmentação

- **Alocação Particionada Dinâmica**
  - *Hole* – block of available memory; holes of various size are scattered throughout memory
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it
  - Operating system maintains information about:
    a) allocated partitions b) free partitions (hole)



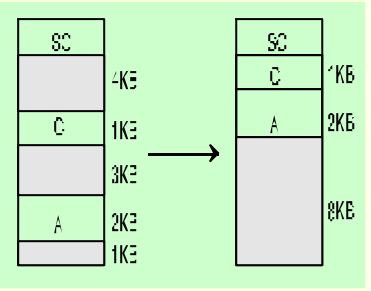
Fragmentação

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
  - I/O problem
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers

Compactação

- **Compactação**

  · *Uma solução para o problema da **Fragmentação** é o uso da técnica de **Compactação**. O objetivo desta técnica é reunir toda memória livre num único bloco de memória, movendo de lugar as partições ocupadas.*



  · *Existem vários algoritmos de compactação, o mais simples seria aquele que move todos processos para o topo (ou base) da memória enquanto os blocos livres são movidos em direção contrária. Este algoritmo normalmente tem um custo proibitivo devido a grande movimentação de memória requerida*

Compactação

Compactação

- ***Swapping*** *pode ser combinado com MVT. Se existe* ***Relocação Estática*** *todo job que é swapped in precisa ser executado na mesma região de memória que ocupava anteriormente, o que força a necessidade de se remover da memória (swap out) os processos que por ventura estejam utilizando a sua região. Se existe* ***Relocação Dinâmica*** *então o job pode ser swapped in em uma posição diferente.*