

QXD0013 - Sistemas Operacionais

Gerenciamento de Memória

Marcos Dantas Ortiz¹

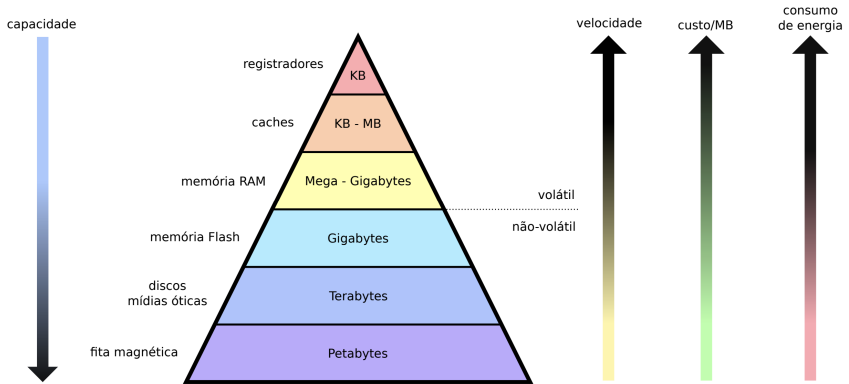
¹Universidade Federal do Ceará, Brazil

15/12/2021

Gerenciamento de Memória - Introdução

- Crescimento constante da capacidade de memória
- Crescimento ainda maior da demanda pelos programas
- Mundo ideal em termos de memória
 - Capacidade infinita
 - Rápida
 - Não-volátil
 - Baixo custo
- Impossível de se obter
- Alternativa: **hierarquias de memórias**
 - Cache: muito rápida, alto custo e volátil (MB)
 - RAM: velocidade e custo médios e volátil (GB)
 - HD: velocidade e custo baixos e não volátil (TB)
- SO → sua função é abstrair essa hierarquia em um **modelo útil** e então **gerenciar a abstração**
- A parte do SO → Gerenciador de memória → Uso/Disponível

Hierarquia de memória

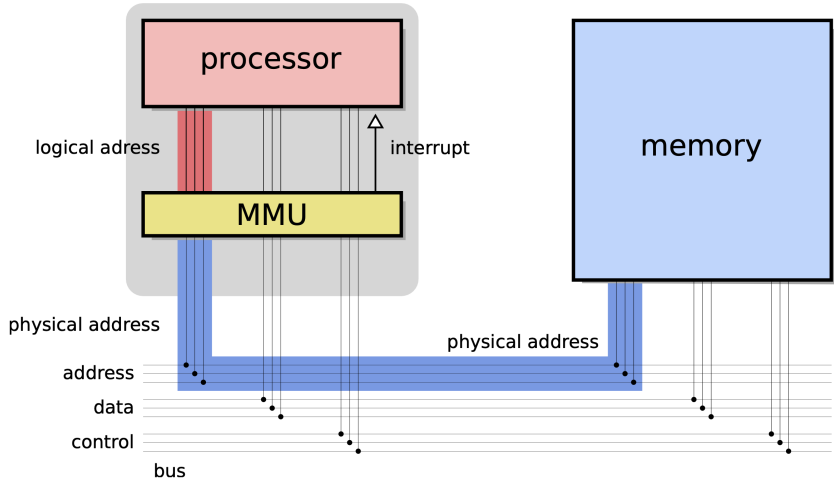


Fonte: Maziero, C. A. (2014). Sistemas operacionais: conceitos e mecanismos. Livro aberto.

Velocidades distintas

Meio	Tempo de acesso	Taxa de transferência
Cache L2	1 ns	1 GB/s (1 ns/byte)
Memória RAM	60 ns	1 GB/s (1 ns/byte)
Memória <i>flash</i> (NAND)	2 ms	10 MB/s (100 ns/byte)
Disco rígido SATA	5 ms (desloc. da cabeça de leitura e rotação do disco)	100 MB/s (10 ns/byte)
DVD-ROM	de 100 ms a vários minutos (gaveta aberta ou leitor sem disco)	10 MB/s (100 ns/byte)

Memory Management Unit - MMU



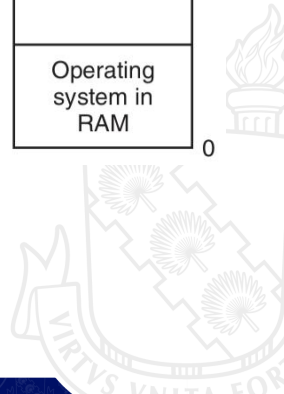
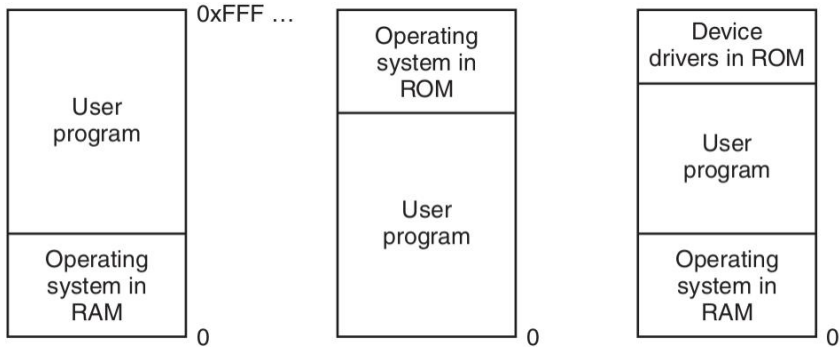
Fonte: Maziero, C. A. (2014). Sistemas operacionais: conceitos e mecanismos. Livro aberto.

Gerenciamento sem abstração

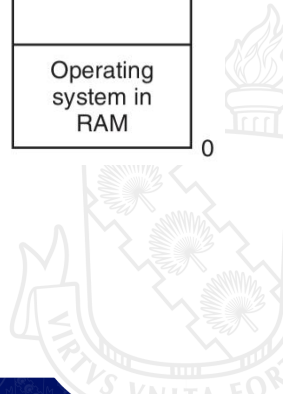
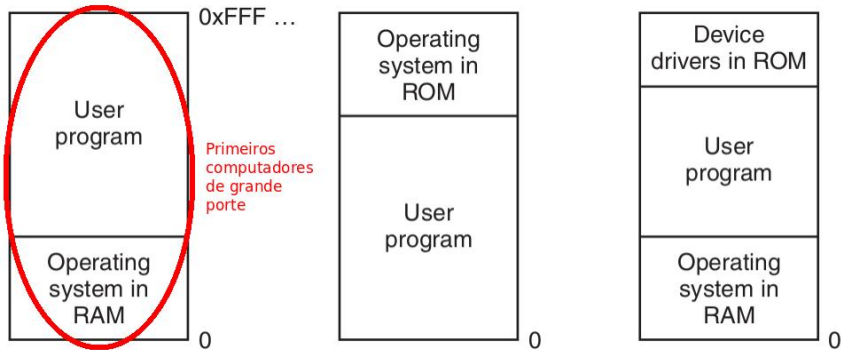
- A abstração de memória mais simples é não ter abstração alguma
- Primeiros computadores não tinham abstração de memória
 - Modelo de memória = Memória física
- Ainda utilizado em sistemas simples
- Conjunto de endereços
- Cada endereço → Célula com um número de bits



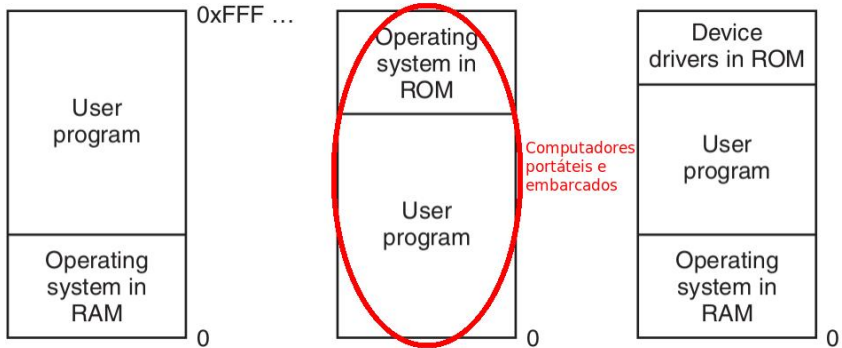
Gerenciamento sem abstração



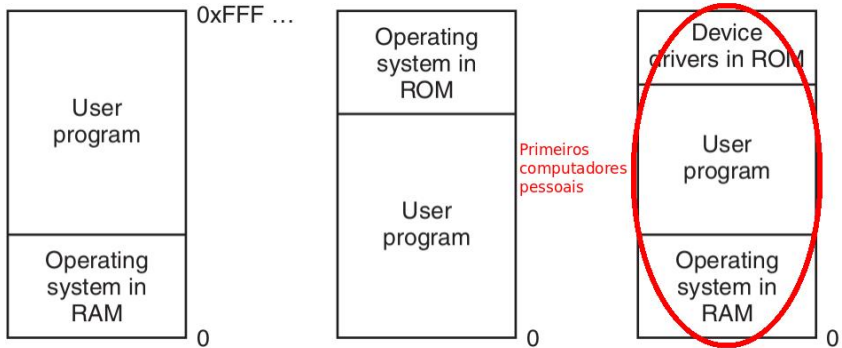
Gerenciamento sem abstração



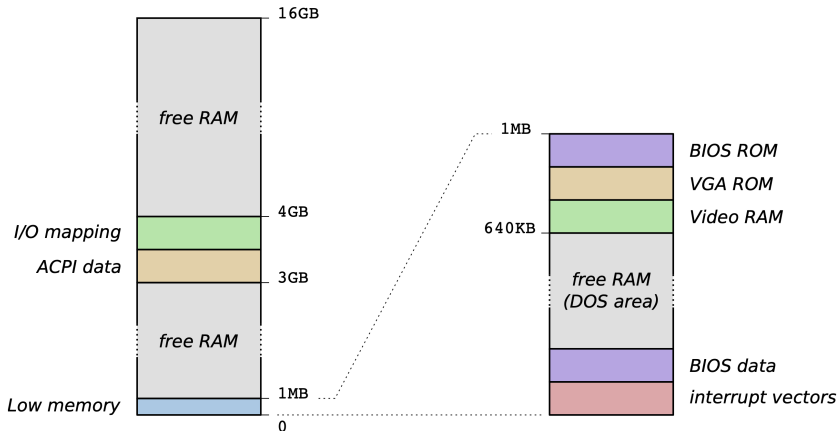
Gerenciamento sem abstração



Gerenciamento sem abstração



Organização da memória em um PC



Fonte: Maziero, C. A. (2014). Sistemas operacionais: conceitos e mecanismos. Livro aberto.

Gerenciamento sem abstração

- Apenas um processo por vez
- SO copia programa do disco para memória e executa
- Quando execução termina, SO pode executar outro processo
- Ao executar outro processo, memória é sobrescrita
- Possibilidade teórica de paralelismo: threads



Gerenciamento sem abstração

- Exemplo de paralelismo sem abstração: IBM 360
 - Memória dividida em blocos de 2KB
 - Associado a cada bloco, uma chave de 4 bits na CPU (proteção)
 - Chave também para cada processo → Program Status Word (PSW)
 - SO impede acesso a bloco sem a chave correta

Gerenciamento sem abstração

0	16380
⋮	
ADD	28
MOV	24
	20
	16
	12
	8
	4
JMP 24	0




Gerenciamento sem abstração

0	16380
⋮	
ADD	28
MOV	24
	20
	16
	12
	8
	4
JMP 24	0 ←



Gerenciamento sem abstração

0	16380
⋮	
ADD	28
MOV	24 
	20
	16
	12
	8
	4
JMP 24	0



Gerenciamento sem abstração

0	16380	0	16380
⋮		⋮	
ADD	28	CMP	28
MOV	24		24
	20		20
	16		16
	12		12
	8		8
	4		4
JMP 24	0	JMP 28	0

Gerenciamento sem abstração

0	16380	0	16380
⋮		⋮	
ADD	28	CMP	28
MOV	24		24
	20		20
	16		16
	12		12
	8		8
	4		4
JMP 24	0	JMP 28	0 ←

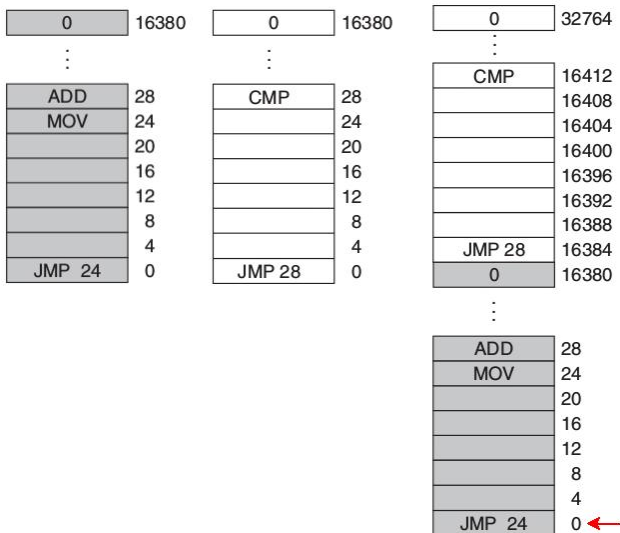
Gerenciamento sem abstração

0	16380	0	16380
⋮		⋮	
ADD	28	CMP	28 ←
MOV	24		24
	20		20
	16		16
	12		12
	8		8
	4		4
JMP 24	0	JMP 28	0

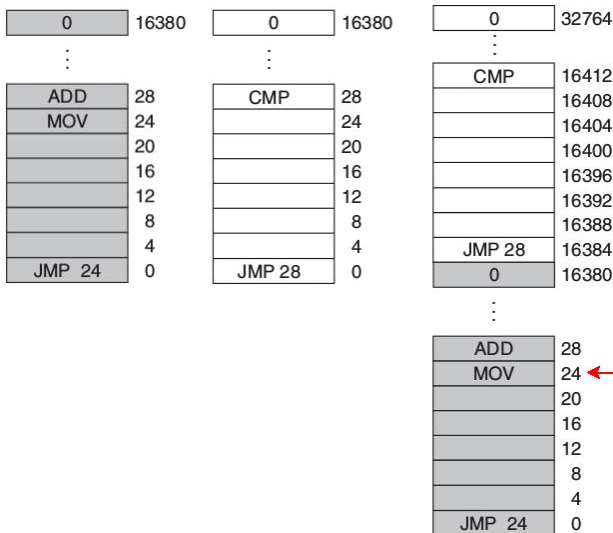
Gerenciamento sem abstração

0	16380	0	16380	0	32764
⋮		⋮		⋮	
ADD	28	CMP	28	CMP	16412
MOV	24		24		16408
	20		20		16404
	16		16		16400
	12		12		16396
	8		8		16392
	4		4		16388
JMP 24	0	JMP 28	0	JMP 28	16384
				0	16380
				⋮	
				ADD	28
				MOV	24
					20
					16
					12
					8
					4
				JMP 24	0

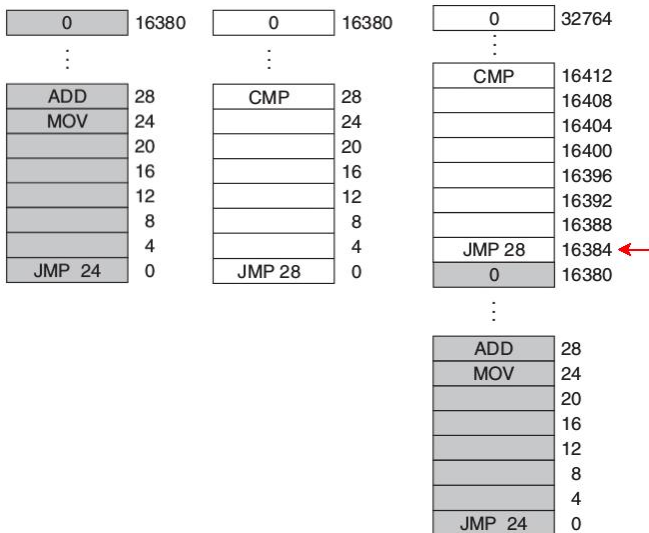
Gerenciamento sem abstração



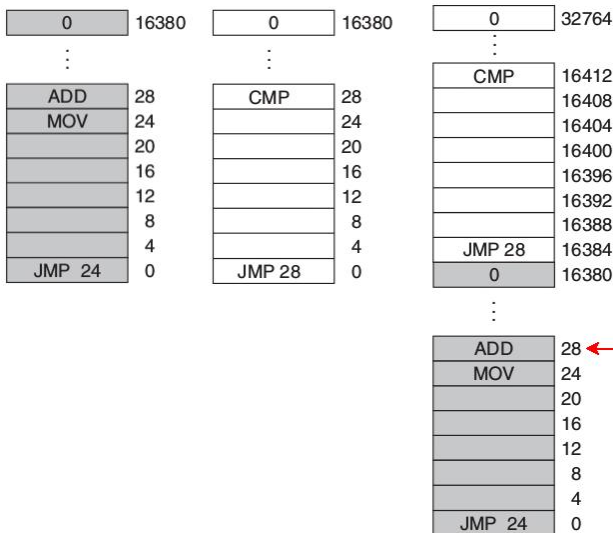
Gerenciamento sem abstração



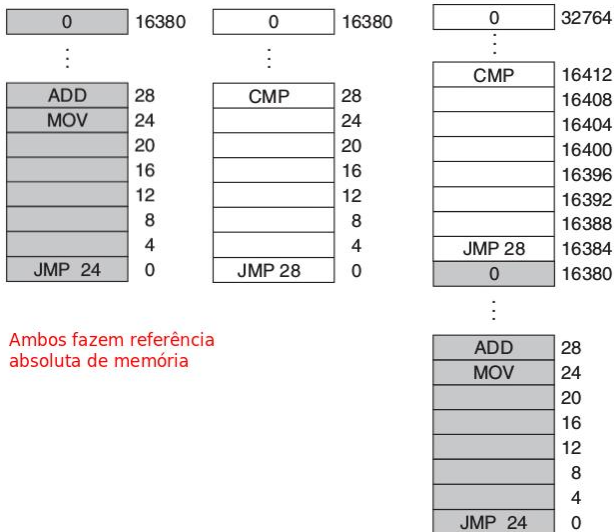
Gerenciamento sem abstração



Gerenciamento sem abstração



Gerenciamento sem abstração



Ambos fazem referência absoluta de memória

Gerenciamento sem abstração

0	16380	0	16380	0	32764
⋮		⋮		⋮	
ADD	28	CMP	28	CMP	16412
MOV	24		24		16408
	20		20		16404
	16		16		16400
	12		12		16396
	8		8		16392
	4		4		16388
JMP 24	0	JMP 28	0	JMP 28	16384
				0	16380
				⋮	
				ADD	28
				MOV	24
					20
					16
					12
					8
					4
				JMP 24	0

Ambos fazem referência absoluta de memória

Solução: adicionar 16384 a todos os endereçamentos do segundo programa

Gerenciamento sem abstração

0	16380	0	16380	0	32764
⋮		⋮		⋮	
ADD	28	CMP	28	CMP	16412
MOV	24		24		16408
	20		20		16404
	16		16		16400
	12		12		16396
	8		8		16392
	4		4		16388
JMP 24	0	JMP 28	0	JMP 28	16384
				0	16380

Ambos fazem referência absoluta de memória

Solução: adicionar 16384 a todos os endereçamentos do segundo programa

REALOCAÇÃO ESTÁTICA

⋮	
ADD	28
MOV	24
	20
	16
	12
	8
	4
JMP 24	0

Abstração de memória: Espaço de endereçamento

- Exposição da memória física aos processos (endereçamento direto)
 - Danos intencionais ou acidentais ao sistema
 - Difícil implementação de paralelismo
 - Problemático mesmo sem paralelismo
- Espaço de endereçamento
 - Solução de dois problemas: proteção e realocação
 - Conjunto de endereços que o processo pode usar
 - Cada processo tem o seu
- Exemplos:
 - Portas de E/S do Pentium: 0 a 16383
 - Endereços IPv4: 0 a $2^{32} - 1$



Memória virtual por partições

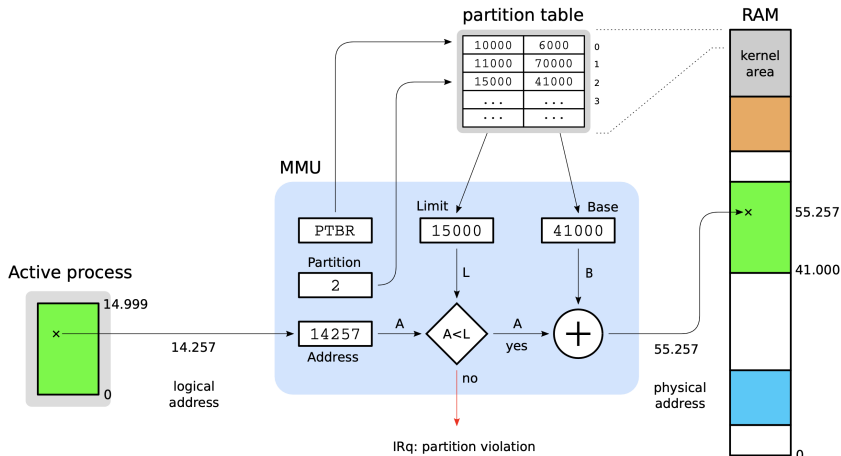


Registradores-base e registradores-limite

- Solução simples da **realocação dinâmica**
- O que ela faz é mapear o **espaço de endereçamento** de cada processo em uma parte diferente da **memória física**
- Dois registradores de hardware especiais
 - Registradores-base
 - Registradores-limite
- Cada referência à memória tem acrescentado o valor base
 - Executado pelo hardware da CPU
 - Também verifica se resultado é menor que limite

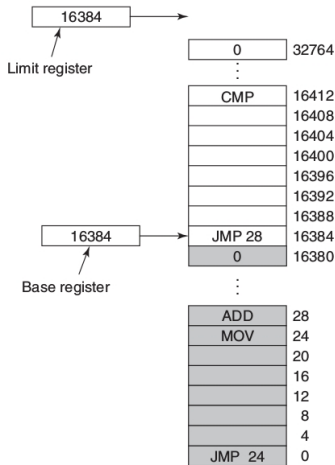


Registradores-base e registradores-limite



Fonte: Maziero, C. A. (2014). Sistemas operacionais: conceitos e mecanismos. Livro aberto.

Registradores-base e registradores-limite



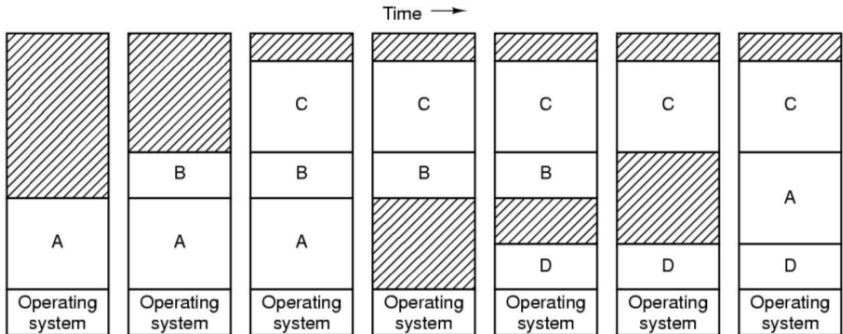
- Solução simples que dá a cada processo um espaço exclusivo
- Em alguns casos apenas o SO pode acessar os registradores
- Desvantagem: a cada acesso deve ser executada
 - Operação de adição
 - Operação de comparação
- Exemplos:
 - CDC6600
 - Intel 8088

Troca de Memória (Swapping)

- RAM necessária para todos os processos maior que a disponível
- Estratégia mais simples: Swapping
 - Carregar processo para execução
 - Devolvê-lo ao disco quando inativo
- Muitas trocas podem deixar espaços vazios: fragmentação
 - Compactação de memória
 - Aumenta espaços contínuos
 - Pouco utilizada por questões de desempenho

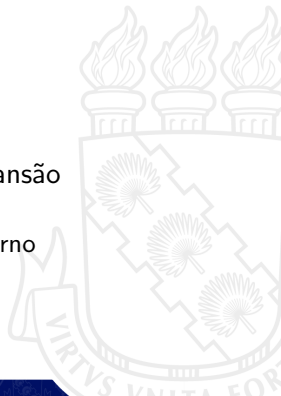


Troca de Memória (Swapping)

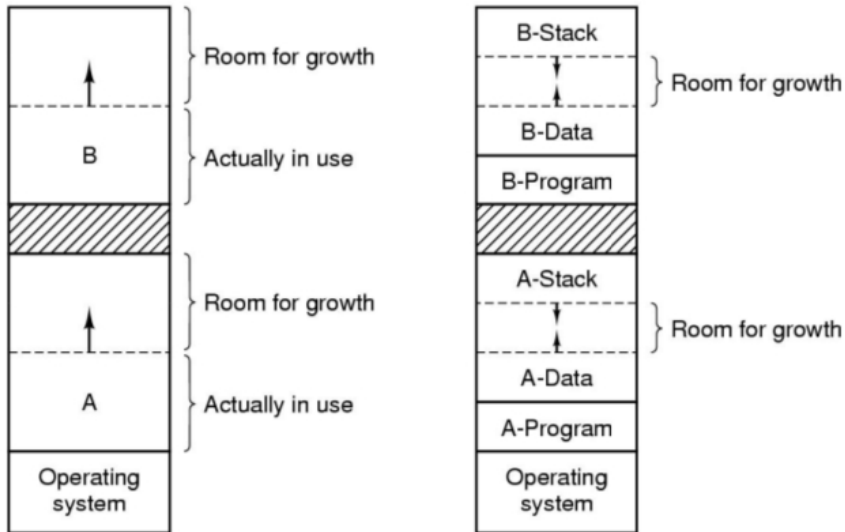


Troca de Memória (Swapping)

- Qual deve ser a quantidade de memória alocada a um processo?
- Processos com tamanho fixo → Solução simples
- Processos com alocação dinâmica
 - Simples havendo espaço livre adjacente
 - Caso contrário, deverá ocorrer realocação
 - Pode não ser possível → Suspensão ou término
- Pode-se alocar memória extra a priori
- Processos também podem ter duas áreas de expansão
 - Dados (heap): variáveis dinâmicas
 - Pilha: Variáveis locais comuns e endereços de retorno

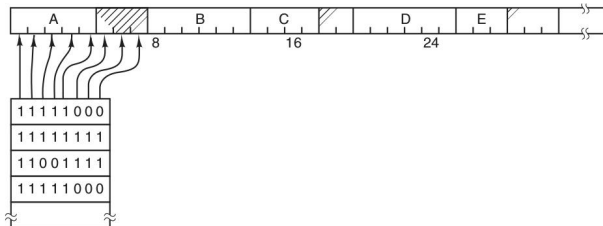


Troca de Memória (Swapping)



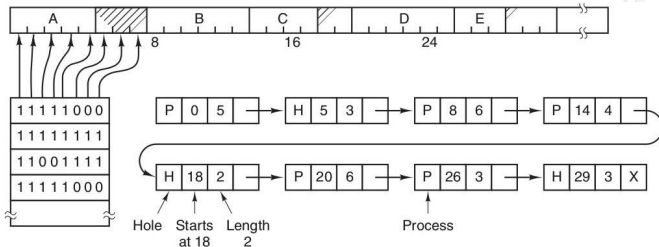
Gerenciamento de Memória Livre

- Duas maneiras de verificar utilização de memória
 - Mapa de bits
 - Listas encadeadas
- Mapa de bits
 - Memória dividida em unidades de alocação
 - A cada unidade é atribuído um bit (0=livre/1=ocupado)
 - Tamanho da unidade: parâmetro de projeto
 - Alocação lenta: procura de sequência de unidades livres



Gerenciamento de Memória Livre

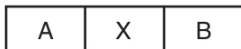
- Listas encadeadas
 - Lista de segmentos alocados e disponíveis
 - Cada elemento da lista:
 - Especifica status (L=livre / P=ocupado com um processo)
 - Endereço de início
 - Comprimento
 - Ponteiro para próximo elemento



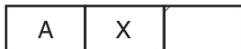
Gerenciamento de Memória Livre

- Listas encadeadas
 - Lista ordenada permite atualização rápida
 - Processo que termina execução tem dois vizinhos na lista
 - Segmentos alocados a um processo
 - Memória livre
 - Existem 4 combinações de vizinhos

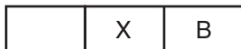
Before X terminates



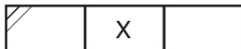
becomes



becomes



becomes

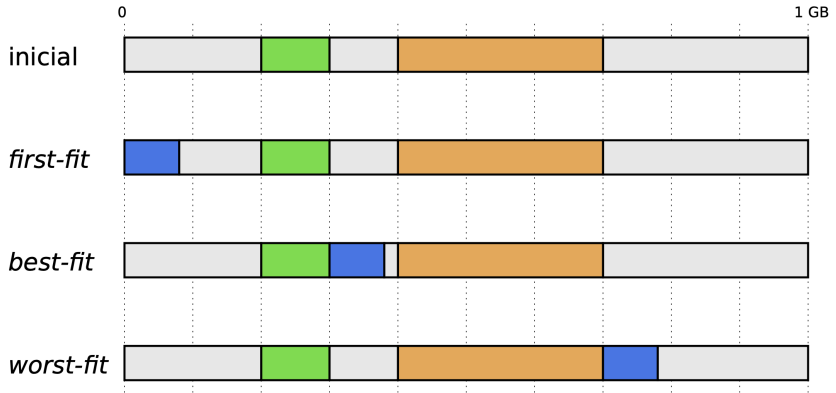


becomes

After X terminates



Gerenciamento de Memória Livre



Fonte: Maziero, C. A. (2014). Sistemas operacionais: conceitos e mecanismos. Livro aberto.

Gerenciamento de Memória Livre

- Diversos algoritmos de alocação
 - First fit: rápido
 - Next fit: memoriza posição encontrada
 - Best fit: escolhe menor segmento (mais lento)
 - Worst fit: escolhe pior segmento (escassez de espaços grandes)
- Listas separadas
 - Alocação mais rápida
 - Maior complexidade
 - Liberação mais lenta: troca de listas
 - First fit e best fit igualmente rápidos



Exercício

- Considere um sistema de troca no qual a memória consiste nos seguintes tamanhos de lacunas na ordem da memória: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB e 15 MB. Qual lacuna é pega para sucessivas solicitações de segmentos abaixo para o algoritmo primeiro encaixe? Agora repita a questão para melhor encaixe, pior encaixe e próximo encaixe.
 - 12MB
 - 10MB
 - 9MB