Gerência de Memória

Sistemas Operacionais

Jiyan Yari

Gerência de memória

Os computadores utilizam uma hierarquia de memória em sua organização, combinando memórias voláteis e não-voláteis, tais como:

- memória cache;

- memória principal;

- e memória secundária.

Gerência de memória

O sistema operacional tem como uma das funções gerenciar o uso dessas memórias de forma eficiente.

Este serviço é implementado pelo sistema operacional através do gerenciador de memória.

Gerência de memória

O gerenciador de memória controla as páginas de memória.

Também é responsável por alocar espaço em memória aos processos que serão executados e liberar as posições de memória ocupadas quando os processos são finalizados.

Outra função do gerenciador de memória é controlar o swap.

MMU

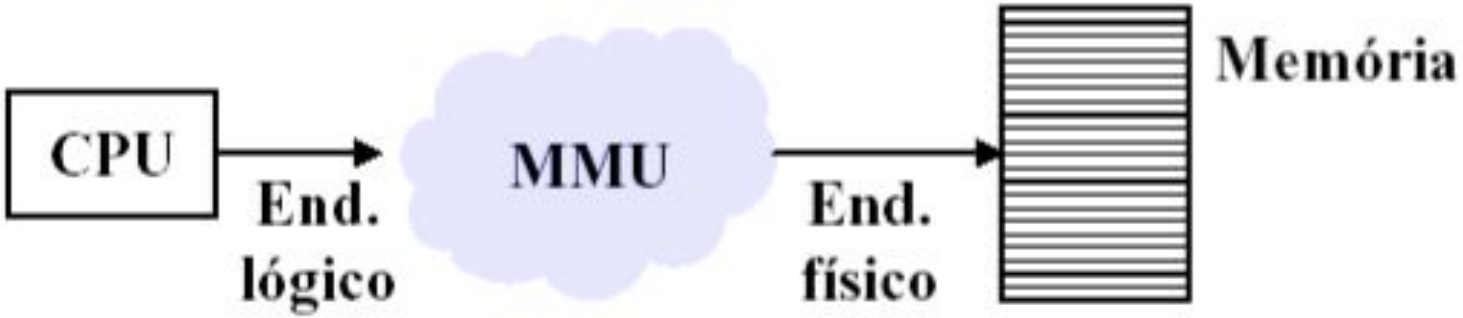
Unidade de Gerência de Memória - Memory Management Unit

A MMU é um módulo de hardware que faz o mapeamento entre os endereços lógicos (memória virtual) e os endereços físicos da memória (RAM).

A MMU é responsável por converter endereços virtuais em endereços físicos.

MMU

Neste contexto, a MMU realiza essa conversão utilizando uma cache chamada Translation Lookaside Buffer (TLB).



MMU

Um código/programa precisa ser convertido em bits para que possa ser executado pelo computador.

Essa conversão pode ser feita pelo compilador ou pelo interpretador (gerando ByteCode).

MMU

Várias processos ocorrem desde que código é compilado e a sua execução, que são:

- geração do código objeto;

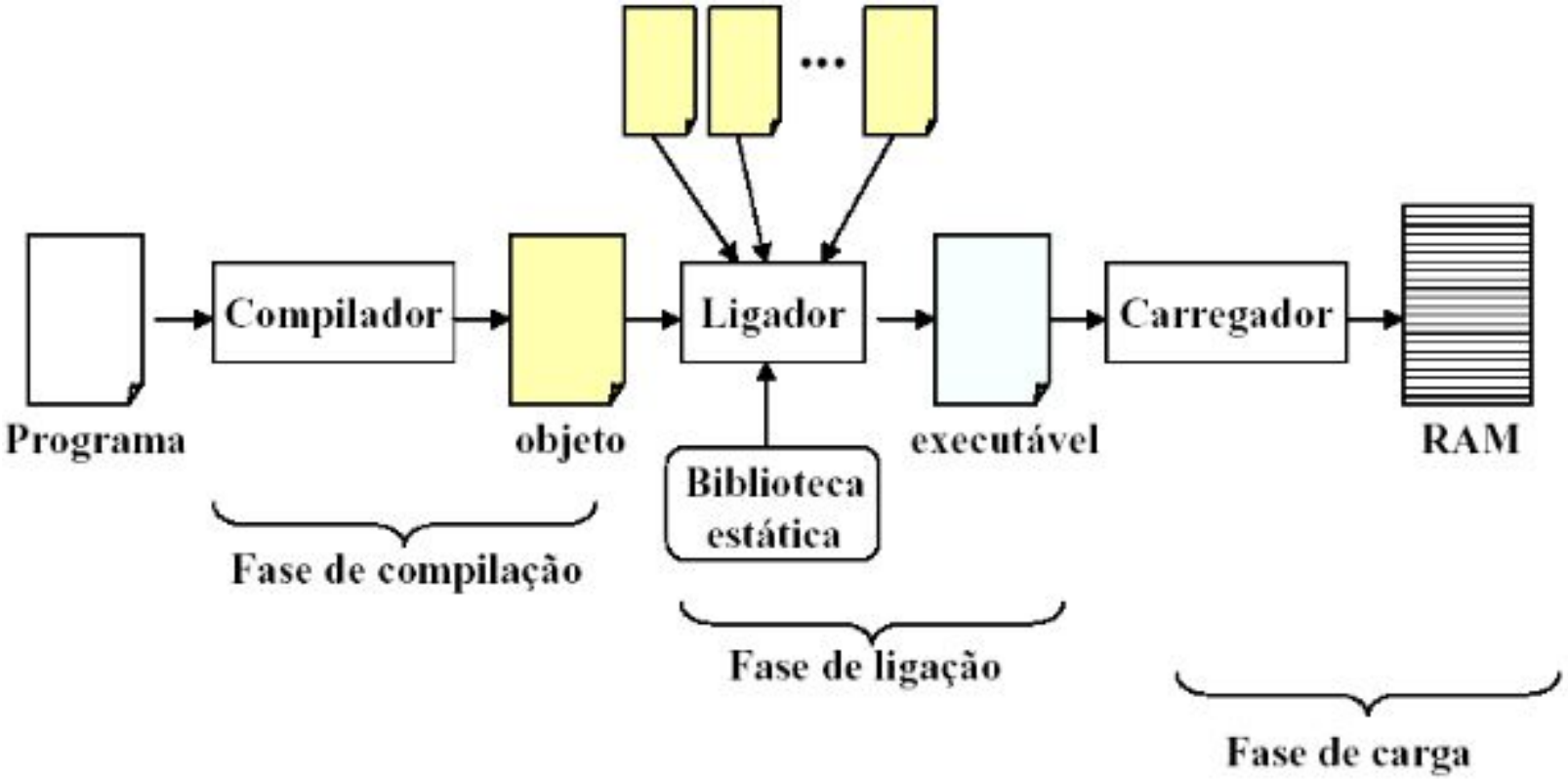
- código executável;

- alocação em memória;

- nova entrada no PCB;

- inserção da referência do processo na fila de apto e etc.

MMU


Programa
Compilador
objeto
Ligador
executável
Carregador
RAM
Biblioteca estática
Fase de compilação
Fase de ligação
Fase de carga

MMU

Várias processos ocorrem desde que código é compilado e a sua execução, que são:

- geração do código objeto;

- código executável;

- alocação em memória;

- nova entrada no PCB;

- inserção da referência do processo na fila de apto e etc.

Gerência de memória

Tipos de gerenciadores de memória:

- que permitem as trocas de processos entre a memória principal e o disco, como troca de processos e paginação; são mais complexos;

- os que não permitem trocas de processos entre a memória principal e o disco, são sistemas operacionais mais simplificados e limitados, para uso especializado.

Gerência de memória

Swap

A necessidade da troca de processos e paginação acontece devido a quantidade insuficiente de memória principal para armazenar vários programas ao mesmo tempo.

Gerência de memória

Sistemas atuais utilizam o modelo chamado de multiprogramação.

Desta forma, os algoritmos precisam gerenciar várias aplicações que concorrem ao uso das unidades de processamento e armazenamento de dados.

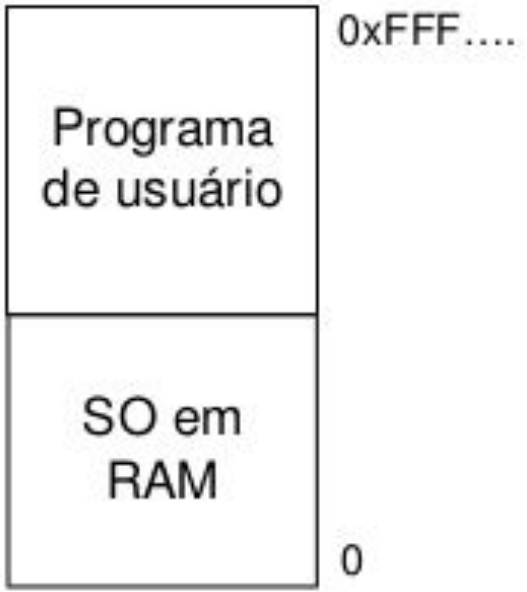
Monoprogramação sem Troca de Processos ou Paginação

A memória é compartilhada entre o sistema operacional e o programa usuário.

Na monoprogramação somente um código/programa do usuário é carregado na memória e executado por vez.

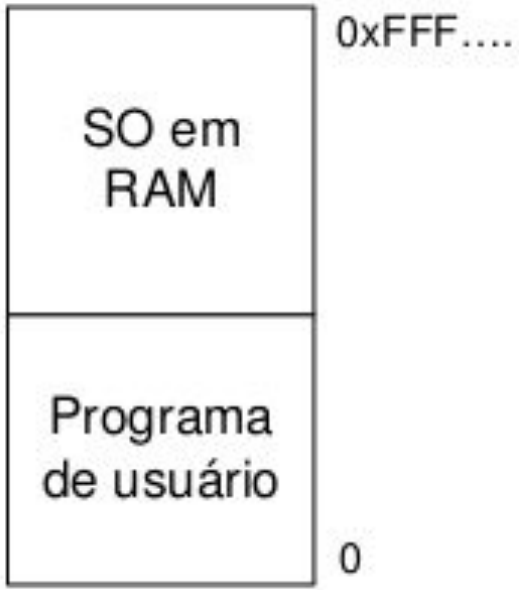
Monoprogramação sem Troca de Processos ou Paginação

O modelo apresentado em figura foi utilizado em computadores de grande porte, mas não é mais utilizado atualmente.



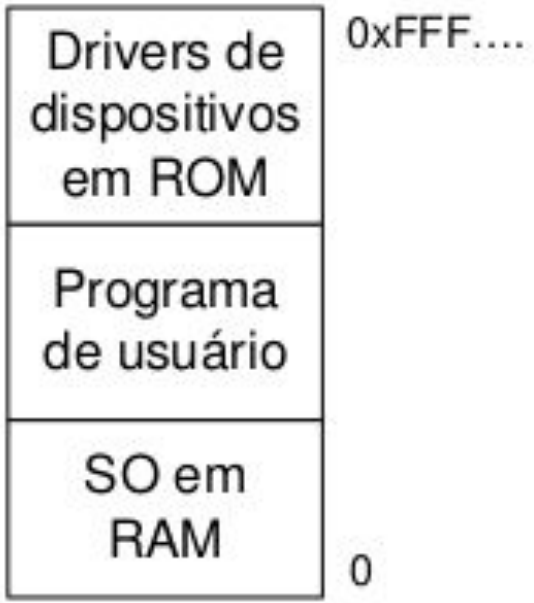
Monoprogramação sem Troca de Processos ou Paginação

Este outro modelo de organização ainda é utilizada em alguns palmtops e em sistemas embarcados.



Monoprogramação sem Troca de Processos ou Paginação

Finalmente, o modelo a seguir esteve presente nos primeiros computadores pessoais, em que parte do sistema operacional contida em ROM é denominada BIOS (Basic Input Output System). Ex: TK-80

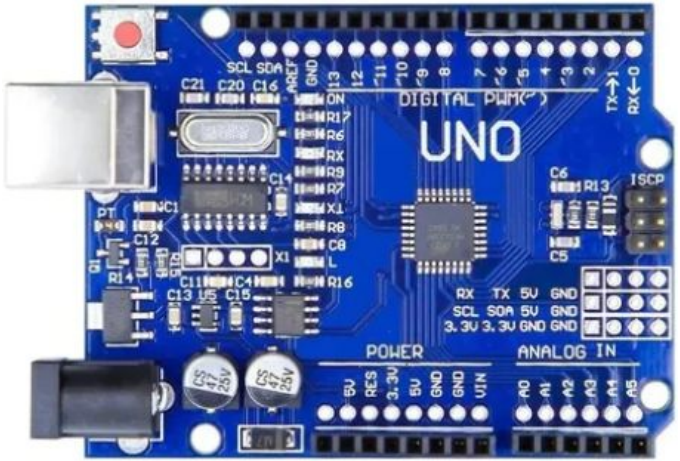


Monoprogramação sem Troca de Processos ou Paginação

Por permitir que apenas um único código/programa do usuário seja carregado na memória a cada instante, a monoprogramação raramente é usada hoje em dia, a não ser em sistemas embarcados simples.

Exemplo:

Arduino Uno R3



Multiprogramação com Partições Fixas

Os sistemas operacionais modernos permitem que mais de um processo seja carregado em memória, de modo que quando um fica bloqueado esperando por uma operação de E/S outro, que esteja carregado em memória, poderá usar a CPU.

Neste contexto, a multiprogramação ajuda a melhorar a utilização da CPU evitando desperdícios de ciclo de processamento.

Multiprogramação com Partições Fixas

Em sistemas com multiprogramação, a memória é dividida em inúmeras partições, ou páginas, em sua maioria de tamanhos diferentes.

As tarefas/jobs serão colocados em filas de entrada associadas à menor partição capaz de armazenar um determinado job.

Multiprogramação com Partições Fixas

Por serem usadas partições de tamanho fixo, todo o espaço de memória não utilizado pelo job será perdido.

Este desperdício de memória é chamado de fragmentação interna, que se trata do espaço de memória perdido dentro da área alocada ao processo.

Multiprogramação com Partições Fixas

página de memória	tamanho
80 K (20 K)	100 K
15 K (5 K)	20 K
50 K (10 K)	60 K
65 K (15 K)	80 K

Multiprogramação com Partições Fixas

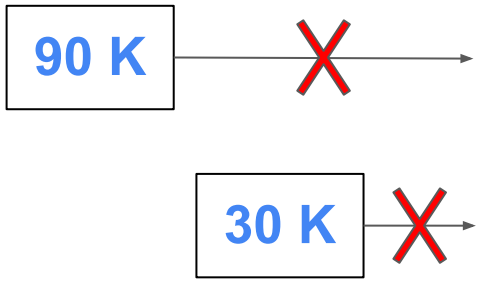
Exemplo:

Tem-se 2 partições livres, uma de 25 e outra de 100 Kbytes e não contíguas.

É criado um processo de 110 Kbytes que não poderá ser carregado em memória pela forma como ela é gerenciada.

Este problema ocasiona o que chamamos de fragmentação externa, ou seja, memória perdida fora da área ocupada por um processo.

Multiprogramação com Partições Fixas



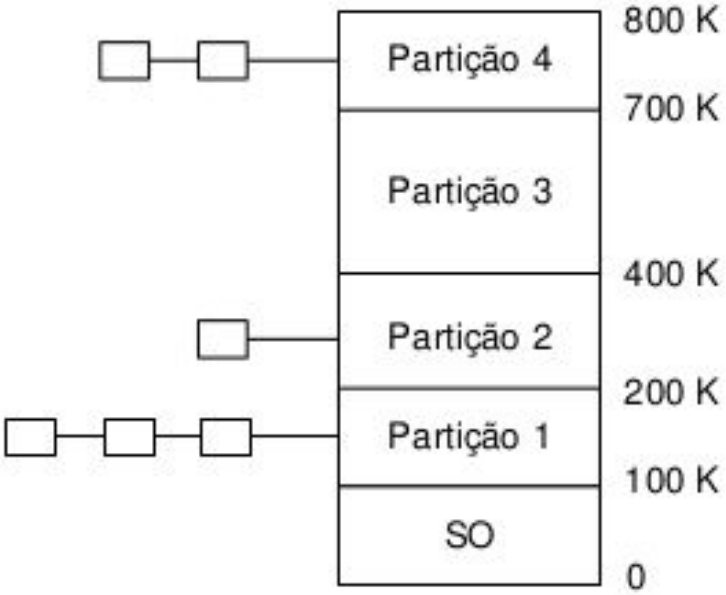
página de memória	tamanho
80 K	100 K
	20 K
50 K	60 K
	80 K

Multiprogramação com Partições Fixas e múltiplas filas

O problema da organização em múltiplas filas é que os jobs pequenos podem ter de esperar pela liberação de memória, ou partição mais adequada ao seu tamanho, embora exista memória disponível, que seria uma partição grande.

Multiprogramação com Partições Fixas e múltiplas filas

Partições fixas com filas de entrada separadas.



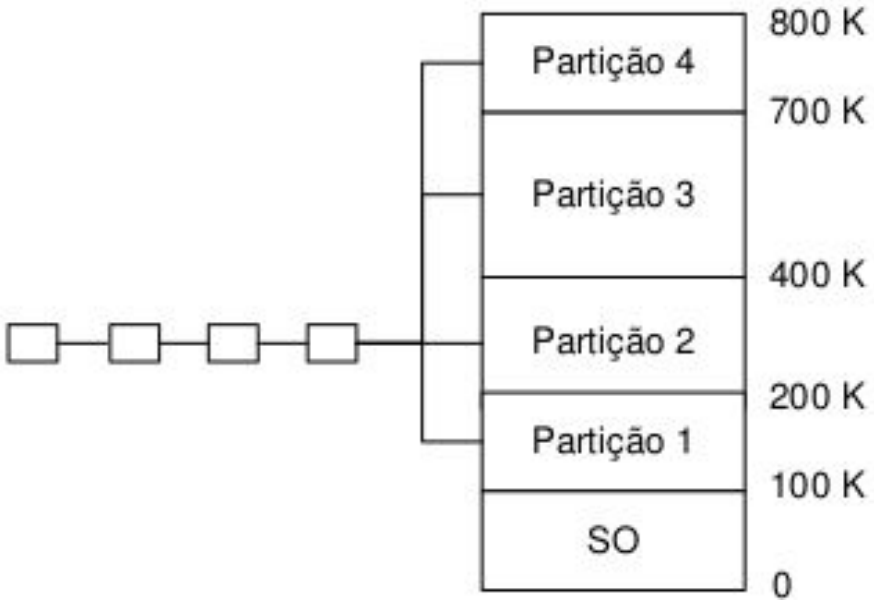
Multiprogramação com Partições Fixas com fila única

No entanto, esse problema não ocorre no esquema de uma única fila.

Nesta configuração sempre que uma nova partição é liberada o job mais próximo do início da fila e que caiba na partição é carregado para ser executado pela CPU.

Multiprogramação com Partições Fixas com fila única

Partições fixas com fila única.



Multiprogramação com Partições Fixas com fila única

No entanto, esta estratégia pode desperdiçar muito espaço ao armazenar um job pequeno em uma partição grande.

Dessa forma, uma opção mais interessante seria pesquisar em toda a fila de entrada e alocar a partição disponível ao maior job que pudesse ser carregado.

Multiprogramação com Partições Fixas com fila única

Problema:

Jobs pequenos ficam em starvation (inanição).

Solução:

Ter pelo menos uma partição pequena.

Multiprogramação com Partições Fixas com fila única

Uma possibilidade consiste em estabelecer uma quantidade máxima “n” vezes que um job pode ser rejeitado a receber uma partição.

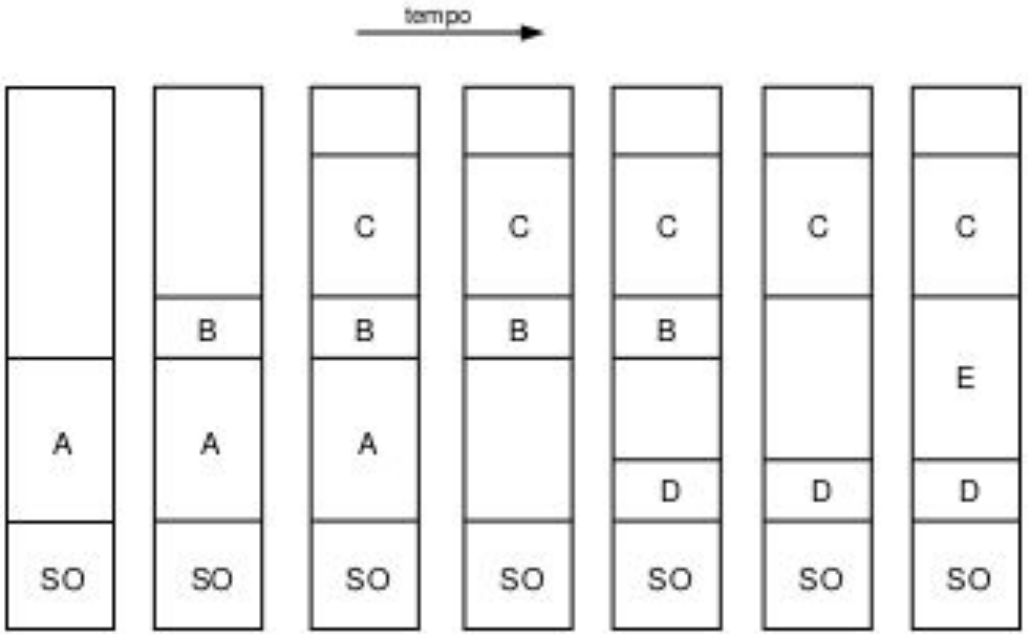
Dessa forma, sempre que ele for rejeitado terá seu contador incrementado e, ao chegar em “n” vezes, deverá receber obrigatoriamente uma partição.

Multiprogramação com Partições Variáveis

Neste modelo a quantidade e o tamanho dos processos na memória podem variar dinamicamente.

Desta forma, o tamanho das partições é ajustado dinamicamente às necessidades exatas dos processos.

Multiprogramação com Partições Variáveis



Multiprogramação com Partições Variáveis

Diferentemente do esquema de partição fixa, na multiprogramação com partições variáveis o tamanho e a localização dos processos variam à medida que o processo deixa e retorna à memória.

Uma das grandes vantagens desta estratégia é que a flexibilidade obtida melhora bastante a utilização da memória, evitando desperdícios de espaço.

Multiprogramação com Partições Variáveis

No entanto, a gerência dos espaços vazios é mais complicada, bem como a alocação e liberação das partições.

O sistema operacional mantém uma lista de espaços livres na memória física.

Multiprogramação com Partições Variáveis

Sempre que um novo processo é criado esta lista é percorrida e será usada uma lacuna maior ou igual ao tamanho do processo em questão.

O espaço que ultrapassar o tamanho do processo pode dar origem a uma nova partição.

Multiprogramação com Partições Variáveis

Alguns dos principais algoritmos para percorrer esta lista são:

- first-fit;
- best-fit;
- worts-fit;
- nest-fit.

Multiprogramação com Partições Variáveis

Algoritmo first-fit:

Inicia a procura a partir da primeira página de memória, chamada de parte baixa, e vai varrendo a memória até encontrar a primeira lacuna suficientemente grande para armazenar o processo.

Multiprogramação com Partições Variáveis

Algoritmo best-fit:

varre toda a memória e escolhe a página mais ajustada ao tamanho do processo.

Multiprogramação com Partições Variáveis

Algoritmo worst-fit:

varre toda a memória e escolhe a página menos ajustada ao tamanho do processo.

Multiprogramação com Partições Variáveis

Algoritmo next-fit:

Segue o mesmo padrão do first-fit, no entanto, a mudança é que somente a primeira busca é iniciada na parte baixa da memória, ou primeira página, as outras iniciam onde terminou a última.

Usa uma lista circular para permitir que toda a memória seja percorrida.