Simulador de Memória Virtual

*Trabalho da disciplina de sistemas operacionais

1st Bruno Rodrigues Caputo
Departamento de Ciência da Computação
Universidade Federal de Roraima
Boa Vista, Roraima
brunorcx@hotmail.com

2st Bruno Cesar da Silva Claudino
Departamento de Ciência da Computação
Universidade Federal de Roraima
Boa Vista, Roraima
brunocclaudino@gmail.com

I. INTRODUÇÃO

Idealmente, podemos dizer, que a memória deveria ter as seguintes características: grande(em tamanho), rápida(em acesso), não volátil(dados persistentes) e de baixo custo. Contudo, não temos ainda uma memória como essa.

Respeitando a hierarquia de memória, podemos observar duas de suas principais características. A primeira é que quanto mais próximo do CPU a memória for, usualmente, terá um acesso mais rápido enquanto que em níveis mais distantes teremos maior tamanho da memória, na figura a seguir o CPU fica logo acima da memória de registradores [1].

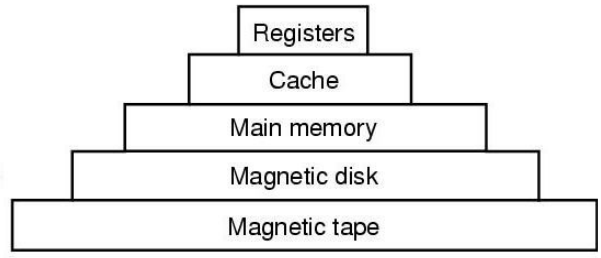

Figura 1. Imagem representativa da memória

II. GERENCIADOR DE MEMÓRIA

Para cada tipo de memória conseguimos ter um gerenciador que ficará responsável por organizar os espaços livres e ocupados, fazendo alocação e localização de processos s e/ou dados na memória em questão.Não se pode esquecer, dos problemas de *swapping*, para chaveamento entre memória principal com o disco, e memória principal e memória cache.

III. MODOS DE TRATAR E ENDEREÇAR MEMÓRIA

Existem diversos modo de se tratar a gerência de memória quando se tem múltiplos programas, uma das mais simples é fazer a divisão da memória em tamanho fixo, não necessariamente iguais, e ao encontrar um *job,* colocar na fila na sua respectiva partição [1].

IV. ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

Existem diversos algoritmo para resolução deste problema, dentre eles vale ressaltar:

LRU: Algoritmo Menos utilizado Recentemente, como o próprio nome diz ele se vale do conceito em que páginas muito usadas por último, provavelmente irão ser usadas novamente. Faz a troca da página que permaneceu sem ter muito uso pelo maior tempo.Custo elevado por precisar manter uma lista encadeada com todas as páginas mais recentes na memória.Utiliza um contador para identificar uso das páginas.

Random: Enquanto houver páginas vazias na memória elas serão preenchidas, assim que a página estiver completamente ocupada uma nova página aleatória deverá ser substituída

FIFO: utiliza a ideia geral da estrutura de dados lista, o sistema operacional mantém uma fila das

Páginas correntes na memória. A página no início da fila é a mais antiga e a página no final é a mais nova. Quando temos um *page fault*, a página do início será removida e a inserção se dará no início da fila. Pode ser ineficiente, já que uma página que está em uso constante pode ser retirada.

V. SIMULADOR

O simulador terá memórias fixas para cada programa, os quais deverão ser informadas na sua inicialização(quadros), bem como o tipo de algoritmo de substituição de páginas (chamado quando a página for completamente preenchida), tamanho da página e a memória total física.

Sua estrutura se dá através de lista encadeada, com o processo, endereço e um ponteiro para a próxima página.

Seu código apresenta, inicialmente, comparações simples para verificar o tamanho máximo e mínimo das páginas.

Em seguida fará a leitura do arquivo, verificando o “r” ou “w” para confirmar leitura e escrita respectivamente.Até finalmente chegar na função de escrita de endereço, a qual tem como verificação as páginas usadas e determina a necessidade de criar uma nova página ou substituir.

Por fim, temos a proposta principal do código, em sua utilização de substituição de páginas que irá fazer de acordo com cada tipo especificado anteriormente.

REFERÊNCIAS

[1] A. S. Tanenbaum, *Modern operating systems 3 e*. Prentice Hall, 2008.