

1. Gerente de Memória

Nesta fase do trabalho criaremos um gerente de memória que oferece paginação.

1.1 Valores Básicos

O gerente de memória para a VM implementa paginação, onde:

A memória tem M palavras.

O tamanho de Página = P palavras (ou posições de memória).

Assim, M / P é o número de frames da memória.

O sistema deve funcionar para diferentes valores escolhidos de M e P. Por exemplo, se $M=1024$ e $P = 16$, temos 64 frames. Se $P = 8$ teremos 128 frames. Isto deve ser facilmente configurável e poderemos testar o sistema com tamanhos diferentes de memória de de página.

1.2 Funcionalidades do Gerente

Alocação: Dada uma demanda em número de palavras, o gerente deve responder se a alocação é possível e, caso seja, retornar o conjunto de frames alocados : um array de inteiros com os índices dos frames.
No nosso sistema, este tamanho deve incluir toda memória utilizada com código e dados.

Desalocação: Dado um array de inteiros com as páginas de um processo, o gerente desloca as páginas.

Interface - solicita-se a definição clara de uma interface para o gerente de memória.

Exemplo:

```
GM {  
    Boolean aloca(IN int nroPalavras, OUT tabelaPaginas []int)  
        // retorna true se consegue alocar ou falso caso negativo  
        // cada posição i do vetor de saída "tabelaPaginas" informa em que frame a página i deve ser hospedada  
    Void desaloca(IN tabelaPaginas []int)  
        // simplesmente libera os frames alocados  
}
```

Estruturas internas: controle de páginas alocadas e disponíveis.

tamMemoria = M (vide 1.1)

array mem[tamMemoria] of posicaoDeMemoria

tamPag = P (vide 1.1)

tamFrame = tamPag

nroFrames = tamMemoria / tamPag

array frameLivre[nroFrames] of boolean // inicialmente true para todos frames

Os frames terão índices.

Cada frame com índice f inicia em $(f)*tamFrame$ e termina em $(f+1)*tamFrame - 1$

Exemplo para tamFrame = 16:

frame	início	fim
0	0	15
1	16	31
2	32	47
3	48	63
4	64	...
...		
62	992	1007
63	1008	1023

1.3 Carga

Nossos programas não serão alterados. Após o GM alocar frames, devolvendo a *tabelaPaginas*, deve-se proceder a carga. Cada página i do programa deve ser copiada para o frame informado em *tabelaPaginas[i]*.

1.4 Tradução de Endereço

Durante a execução do programa, todo acesso à memória é traduzido para a posição devida, conforme o esquema de paginação. Assim, a tabelas de páginas do processo é utilizada durante a execução do processo. Lembre-se que no seu programa você utiliza endereços lógicos, considerando a abstração de que o programa está disposto contiguamente na memória, a partir da posição 0. Ao acessar a memória de fato, cada endereço lógico deve ser transladado para o físico, para que então a posição específica da memória seja acessada.

Seja *div* a operação de divisão inteira e *mod* a operação que obtém o resto da divisão inteira, cada acesso à memória passa pela translação:

seja *A* o endereço a ser acessado, temos:

$$p = A \text{ div } tamPag$$

é o índice da página que o programa acessa

$$offset = A \text{ mod } tamPag$$

é o deslocamento dentro da página

toma-se *p* e tem-se que:

$$tabPaginas[p]$$

é o frame onde a página *p* se encontra

$$tabPaginas[p] * tamFrame$$

é o início do frame a ser acessado

assim, $(tabPaginas[p] * tamFrame) + offset$ é o endereço a ser acessado na memória física

ou, da mesma forma, substituindo:

$$(tabPaginas [(A \text{ div } tamPag)] * tamFrame) + (A \text{ mod } tamPag)$$

Pode-se montar uma função de tradução de endereço *T* que, dado *A* endereço lógico do programa e a tabela de páginas do mesmo, traduz *A* para o endereço físico que deve ser acessado na memória.

$$T(A) = (tabPaginas [(A \text{ div } tamPag)] * tamFrame) + (A \text{ mod } tamPag)$$

No processo de tradução deve ser verificado se a página resultante é válida, gerando erro ou permitindo o acesso. A página é válida se o programa tem o índice *p* da página obtida em $A \text{ div } tamPag = p$.

1.5 Testes

Deve-se demonstrar que o sistema pode carregar programas diferentes em diferentes frames, não contíguos, e executar. Para tal voce pode iniciar alguns frames como alocados no gerente de memória. Desta forma estes nunca serão utilizados , gerando "buracos" na memória.

Deve-se poder executar com diferentes tamanhos de memória e de página, apenas mudando estas constantes na inicialização do sistema (no código). Todo resto deve funcionar.

Deve-se poder carregar programas e ver o conteúdo da tabela de páginas do programa e assim como da memória de forma a ver em que frames foram alocados.