PUCRS - Escola Politécnica
Disciplina: Sistemas Operacionais - Trabalho Prático
Prof. Fernando Luís Dotti

Gerente de Memória para Paginação (Trabalho 1A)

1.Gerente de Memória

Nesta fase do trabalho criaremos o gerente de memória implementando **paginação**.

1.1 Valores Básicos

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

- a memória tem *tamMem* palavras.
- O tamanho de Página = *tamPg* palavras (ou posições de memória).
  Assim, *tamMem / tamPg* é o número de frames da memória.
  Lembrando que o tamanho de um frame *tamFrame = tamPg*

O sistema deve funcionar para diferentes valores escolhidos de *tamMem e tamPg.*
Se *tamMem=1024* e *tamPg=8* teremos *128* frames. Testaremos o sistema com diferentes tamanhos de memória e de página.

1.2 Funcionalidades do Gerente de Memória

Alocar: 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, para carregar um programa, deve-se alocar toda memória necessária para **código e dados.**

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

Sugestão de 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 quadros(frames) alocados e disponíveis.

Os frames terão índices.
Cada frame com índice f inicia em (f)*tamFrame e termina em (f+1)*tamFrame -1
Exemplo para tamFrame = 16 e tamMem 1024:

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

1.3 Carga

ATENÇÃO: os programas escritos para a VM não serão alterados em NADA.
Após o GM alocar frames, devolvendo a *tabelaPaginas*, deve-se proceder a carga. Cada página i do programa deve ser copiada (exatamente como tal) para o frame informado em *tabelaPaginas[i].*

1.4 Tradução de Endereço e Proteção de Memória

Durante a execução do programa, todo acesso à memória é traduzido para a posição devida, conforme o esquema de paginação. Assim, a **tabela 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. E isto não será alterado.* A memória física é um array de posições de memória. O endereço físico é um valor de 0 ao tamanho da memória. Ao acessar a memória física, cada endereço lógico deve ser transladado para o físico, para que então a posição específica da memória seja acessada. Deve-se converter o endereço lógico (linear) em página e deslocamento na página, acessar a tabela de pagina, descobrir o frame a ser acessado, calcular o endereço de início do frame e adicionar o mesmo deslocamento na página (pois página e frame tem mesmo tamanho).