**CES-33**

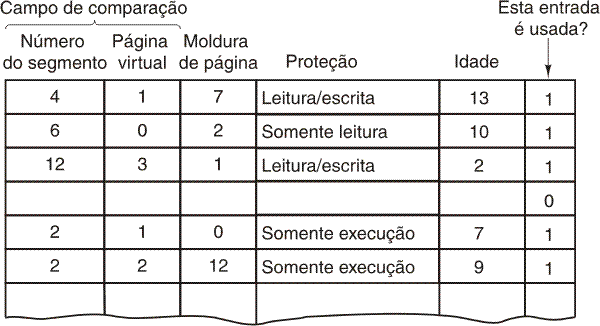
**Exemplo: Segmentação e Paginação**

Este projeto consiste em escrever um programa que traduza endereços lógicos para endereços físicos de um espaço de endereçamento virtual de 218=262144 bytes. Seu programa lerá de um arquivo contendo endereços lógicos e, usando TLB expandida para conter numero de segmento e número de páginas, traduzirá cada endereço lógico para seu correspondente endereço físico, dando saída no valor do byte armazenado no endereço físico resultante. O objetivo deste projeto é a simulação dos passos envolvidos na tradução de endereços lógicos para físicos, desta vez considerando segmentação.

**Especificidades**

Seu programa lerá um arquivo contendo vários números inteiros de 32 bits que representam endereços lógicos. No entanto, você precisa se preocupar apenas com endereços de 18 bits. Esses 18 bits são divididos em (1) Na mais alta ordem, dois bits do numero do segmento (serão gerados por vocês como explicado mais abaixo); (2) um número de página de 8 bits e (3) um deslocamento de página de 8 bits.

A novidade introduzida em relação ao projeto anterior é que os dois primeiros bits dos 18 referem-se ao endereço de segmento, que também são armazenados na TLB. Neste caso são possíveis apenas quatro segmentos diferentes. A figura extraída do SO MULTICS dá um exemplo de como seria a TLB:

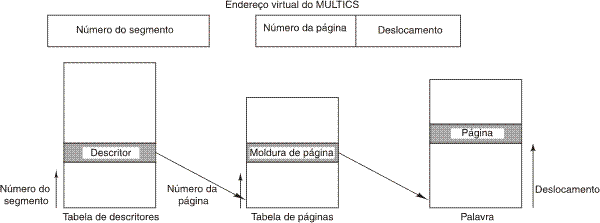


Outras especificidades incluem:

* 28 entradas na tabela de páginas;
* Tamanho da página de 28 bytes;
* 16 entradas na TLB;
* Tamanho da moldura de 28 bytes;
* 128 molduras em cada segmento.
* Tamanho da memória física de 4 x 32.768 bytes (4 x 128 molduras x molduras de 256 bytes).

**Tradução de Endereços**

Seu programa traduzirá endereços lógicos para físicos usando uma TLB, uma tabela de segmentos e uma tabela de páginas como visto na teoria. Primeiro, o número de segmento + numero da página é extraído do endereço lógico e a TLB é consultada. Em caso de sucesso da TLB, o número da moldura é obtido a partir da TLB. Em caso de omissão da TLB, a tabela de segmentos é consultada. Se o número de segmento citado não está na memória, acontece um *Segmentation Fault*. Em caso de sucesso, a tabela de páginas correspondente é consultada, nos moldes do Multics ilustrado na Figura abaixo.



Se número da página não está na tabela de páginas ocorre um *Page Fault*. (Obs: quadro=moldura).

**Manipulando erros de página**

Seu programa implementará a paginação por demanda (Só traz a página para a memória quando ela é necessária). A memória de retaguarda (swap) é representada pelo arquivo BACKING\_STORE.bin, um arquivo binário de 65.536 bytes. Quando ocorrer um erro de página, você lerá uma página de 256 bytes a partir do arquivo BACKING\_STORE.bin e a armazenará em uma moldura disponível na memória física. Por exemplo, se um endereço lógico com o número de página 15 resultar em um erro de página, seu programa lerá a página 15 em BACKING\_STORE.bin e armazenará em uma moldura na memória física. Uma vez que esse quadro seja armazenado, acessos subsequentes à página 15 serão resolvidos pela TLB ou pela tabela de páginas.

Você tratará BACKING\_STORE.bin como um arquivo de acesso aleatório para que possa pesquisar aleatoriamente certas posições do arquivo para leitura. (Use as funções de C fopen(), fread(), fseek() e fclose().

**Arquivo de Texto**

O arquivo addresses.txt contém valores inteiros representando endereços lógicos que variam de 0 a 65.535 ( o tamanho do espaço de endereçamento virtual de um segmento). Seu programa abrirá esse arquivo, lerá cada endereço lógico, o traduzirá para o físico e dará saída no valor do byte sinalizado contido no endereço físico. Na i-esima leitura, assuma que o número de segmento é seg = (i mod 4). Assuma que o conteúdo dos segmentos no BACKING\_STORE.bin é o mesmo para os quatro segmentos.

**Como Executar Seu Programa**

Seu programa deve ser executado como descrito a seguir:

./a.out addresses.txt

Seu programa lerá o arquivo addresses.txt que contém 1000 endereços lógicos variando de 0 a 65.535. Seu programa deve traduzir cada endereço lógico para um endereço físico e determinar o conteúdo do byte sinalizado armazenado no endereço físico correto. (sugere-se que use o tipo char do C que ocupa um byte de memória).

O programa deve dar saída nos seguintes valores:

1. O endereço lógico que está sendo traduzido ( escreva número de segmento gerado antes do endereço lógico );
2. O endereço físico correspondente (aquele para o qual seu programa traduziu o endereço lógico, preceda também o número do segmento);
3. O valor do byte sinalizado armazenado no endereço físico resultante, extraído do BACKING\_STORE.bin.

Exemplo do arquivo que será gerado:

**Virtual address: 0-16916 Physical address: 0-20 Value: 0**

**Virtual address: 1-62493 Physical address: 1-29 Value: 0**

**Virtual address: 2-30198 Physical address: 2-246 Value: 29**

**Virtual address: 3-53683 Physical address: 3-179 Value: 108**

As quatro primeiras referências serão mapeadas na moldura 0 dos respectivos segmentos 0,1,2,3.

Todas vão pegar o valor das páginas correspondentes do BackingStore.bin.

As próximas quatro primeiras referências serão mapeadas na moldura 1 dos respectivos segmentos 0,1,2,3.

**Virtual address: 0-40185 Physical address: 0-505 Value: 0**

**Virtual address: 1-28781 Physical address: 1-365 Value: 0**

**Virtual address: 2-24462 Physical address: 2-398 Value: 23**

**Virtual address: 3-48399 Physical address: 3-271 Value: 67**

A partir da memória cheia, usar o FIFO para substituição de página.

**Estatísticas**

Após concluído, seu programa deve relatar as seguintes estatísticas:

1. Taxa de erro de segmento; o percentual de referências de segmentos que resultaram em Segmentation Fault;
2. Taxa de erros de página - o percentual de referências de endereços que resultaram em Page Fault;
3. Taxa de sucesso do TLB - o percentual de referências de endereços que foram resolvidas na TLB.

Já que os endereços lógicos em addresses.txt foram gerados aleatoriamente e não refletem quaisquer localidades de acesso à memória, não espere obter uma alta taxa de sucesso de TLB.

No dia 28/Junho aconteceria a apresentação final com entrega de relatório.

Como todos os alunos vão usar FIFO e a mesma entrada (addresses.txt), o arquivo de saída deve ser o mesmo. Portanto, antes de vir para a apresentação, rodem o fc do Windows ou o diff do Linux entre os arquivos de saídas dos grupos de vocês para comparar.

O relatório simples deve conter:

1) Introdução com alguma explicação que julgar necessária para a compreensão de sua implementação;

2) Explicação do código fonte anexado;

3) As estatísticas comentadas; Comentário da comparação da saída com outros grupos.

4) Conclusão.