departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Arquitetura de Computadores 2017/18

TPC 5

Prazo de entrega: 23:59 de 11 de junho de 2018

Tópicos: *Endereços virtuais e páginas.*

Este trabalho de casa consiste em dois exercícios de programação **individual**. Pode esclarecer dúvidas gerais com colegas, mas a solução e a escrita do código deve ser estritamente individual. Todas as resoluções serão comparadas de forma automática e os casos de plágio serão punidos de acordo com os regulamentos em vigor.

A entrega será via Mooshak do DI (`http://mooshak.di.fct.unl.pt/~mooshak/`). Todo o código necessário para este trabalho é fornecido via CLIP. Para compilar, basta dar o comando `make` no terminal.

Introdução

Obtenha a partir do CLIP o ficheiro *pagsim.zip* que contém um simulador de memória virtual paginada a pedido com *Translation Lookaside Buffer* (TLB). O simulador deve ser executado com pelo menos dois argumentos:

1. A dimensão da TLB (número de entradas nesta cache da tabela de páginas) a usar na simulação; e
2. Um ficheiro com um traço (registo) dos endereços de memória gerados na execução real de um programa.

O ficheiro com um traço dos endereços de memória gerados na execução real de um programa, tal como para a ficha 11, é um ficheiro de texto com dois dados por linha, com o seguinte formato (exemplo):

```
0041f7a0 R
31348900 W
```

O primeiro dado é um inteiro (hexadecimal) com o endereço de memória virtual que o CPU gerou. O segundo dado é o tipo de operação realizada naquele endereço de memória: R (leitura) ou W (escrita).

Este programa simula as ações do *hardware* de paginação, com TLB, assim como as ações da rotina de tratamento da interrupção por falta de página. A simulação é feita nas seguintes condições:

- Só há um processo, cujos acessos a memória são os descritos no ficheiro indicado.
- A arquitetura baseia-se em endereços virtuais e físicos de 32bits, com páginas de 4Kbytes. Assim 12 bits são para o deslocamento e 20 bits para o número da página virtual e o processo terá no máximo 2^{20} páginas (tamanho da tabela de páginas). A tabela de páginas é simulada por um vetor de estruturas que, para cada página, contém os seguintes atributos:
  - o bit de validade; e
  - o número da *frame* em que a página reside em memória física.
- A MMU tem uma TLB do tamanho indicado como argumento da linha de comando, sendo as páginas procuradas primeiro na TLB antes de se ir à tabela de páginas. Quando se consulta a tabela de páginas, acrescenta-se essa entrada na TLB, substituindo outra mais antiga se necessário.
- A memória física simulada tem 8Mbytes, sendo a ocupação das *frames* (páginas físicas) mantida numa tabela de *frames* do SO, no ficheiro "`so.c`". Esta tabela contém, para cada *frame*, a indicação se esta é válida ou não e, se válida, o número da página guardada nessa *frame*,

A gestão das páginas é feita em paginação a pedido controlada pelo SO e implementado em "`so.c`". Isto significa que inicialmente o processo não ocupa nenhuma *frame* na memória física e toda a memória física está livre. As páginas vão sendo carregadas pelo SO à medida que vão sendo referenciadas. Quando não existe nenhuma *frame* livre é preciso libertar uma, sendo a escolha da *frame* vítima feita por um algoritmo de substituição de páginas em "`so.c`".

Procure perceber a organização geral do programa. Para sua referência, o código é o seguinte:

`Makefile`	Ficheiro de configuração para o comando "`make`".
`simulador.c`	Contém o programa principal do simulador que lê as referências no ficheiro e pede a conversão de cada endereço virtual em físico.
`mmu.h mmu.c`	Contêm o código que simula funcionalidades associadas à *Memory Management Unit:* converter o endereço virtual procurando a respetiva *frame* física, procurando primeiro na TLB e, se não encontra, usando a tabela de páginas.
`tlb.h tlb.c`	Contêm o código que simula funcionalidades normalmente associadas com a unidade *Translation Lookaside Buffer:* procura a página devolvendo a respetiva *frame* se encontrar.

`so.h so.c`	Contêm o código que simula funcionalidades normalmente associadas ao Sistema de Operação: atribuir *frames* às páginas, atualizando a respetiva tabela de páginas.
`stats.h stats.c`	Contêm o código que implementa a funcionalidade de coligir e imprimir as estatísticas do processo de simulação.

Para compilar o programa execute o comando `make` no terminal. Deverá ser produzido um executável com o nome `simulador` que, para funcionar corretamente, necessita que termine a implementação de algumas das funções.

Exercício 1

Neste exercício deve completar a implementação da função `translateOneAddr` em `mmu.c`. Esta converte um endereço virtual no real (físico). Para tal tem a função `page2frame` que consulta a TLB e, se necessário, a tabela de páginas para obter a respetiva página física (*frame*) correspondente à página virtual que contém o endereço referenciado.

Para testar a sua solução, execute a simulação com o seguinte comando:

```
simulador -v 32 bzip.trace
```

A opção "-v" indica ao simulador que deve mostrar os endereços obtidos pela conversão pretendida. Por agora, **sem implementar** o exercício 2, deve obter para os primeiros endereços os seguintes valores:

```
13a0 R
2d40 R
3d60 R
43c0 R
5360 R
6308 R
7380 R
82f0 R
93a0 R
```

Neste exercício submeta ao mooshak apenas o seu ficheiro **`mmu.c`**

Exercício 2

Neste exercício deve completar a implementação das funções em `tlb.c`. Estas permitem procurar páginas e acrescentar novas páginas à TLB. Quando a TLB está cheia, para acrescentar novas páginas, é necessário substituir uma entrada da TLB, usando um algoritmo do tipo LRU.

Algoritmo a implementar: para contabilizar a passagem do tempo vamos usar um relógio virtual (`clock` em `tlb.c`) que é incrementado a cada acesso a memória, ou melhor, a cada procura de uma página por `findTLB`. Cada entrada na TLB tem o seu atributo `lastref` atualizado com o valor em `clock`. Quando uma página não está na TLB será acrescentada por `addTLB`. Nesta função, tal será conseguido colocando a nova página numa entrada livre se existir (`findFree`), se não existir, usa `findVictim` para escolher a que há mais tempo não é acedida, ou seja, a que tem menor valor em `lastref` (tal significa que é a que foi acedida há mais tempo).

Uma vez completada a implementação dos elementos referidos acima, teste o programa com os ficheiros de endereços e veja as taxas de faltas de páginas obtidas, com diferentes dimensões de TLB. Por exemplo, para executar o simulador com uma TLB de 32 *páginas* para o ficheiro `bzip.trace`, deve escrever o seguinte comando (repare que agora não se utiliza a opção "-v"):

```
simulador 32 bzip.trace
```

Tendo implementado corretamente ambos os exercícios, deve obter o seguinte resultado:

```
INPUT FILE = bzip.trace
number of memory references = 1000000
number TLB misses = 2133
number of TLB victims = 2101
TLB hitratio = 99.8%
```

Neste exercício submeta ao mooshak apenas o seu ficheiro **`tlb.c`**

Nota extra: Para além de *passar nos testes do* Mooshak, sugerimos que analise e tente explicar as diferenças de resultados obtidos para diferentes dimensões de TLB (p.e., 2, 8, 16, 32, 64, 128).