Exercício Programa 2 - Simulador de Caches

Arquitetura de Computadores - 2020.Q1

Emilio Francesquini - e.francesquini@ufabc.edu.br

7 de maio de 2020

1 O Projeto

Neste projeto colocaremos em prática todo o conhecimento acumulado sobre o funcionamento de memórias cache para construirmos o nosso próprio simulador!

Para isto utilizaremos alguns *traces* de memória para simular o comportamento que diferentes configurações de memória cache possuem. Alguns traces de exemplo são fornecidos juntamente com este enunciado mas você pode gerar traces adicionais utilizando a ferramenta Valgrind e Lackey. Para isto basta instalar o Valgrind na sua máquina (em máquinas Linux basta um `apt-get` ou similar e executar `valgrind --tool=lackey --trace-mem=yes <PROGRAMA>`).

O EP2 vale de 0 a 13. Descrevo abaixo as diferentes *milestones* que marcam as funcionalidades mínimas exigidas para alcançar cada uma das notas.

1.1 Nota 5

O seu simulador deve ser capaz de simular uma cache unificada de nível único (CPU -> Cache -> RAM) que dado um dos arquivos de entrada (listados abaixo) seja capaz de:

- Ser configurada quanto ao seu tamanho total em bytes (apenas conta os dados e não os outros bits de controle) e quanto ao seu número de linhas;
  - Note que o par tamanho em bytes e número de linhas determina automaticamente o tamanho de cada linha. Qualquer combinação de dois elementos do trio (número de linhas; tamanho em bytes;

tamanho de cada linha em bytes) é aceitável como configuração do seu simulador já que determinam automaticamente a terceira;
    - Fica a seu critério se a configuração será passada como parâmetro à execução do simulador ou em um arquivo de configuração;
- Usar mapeamento direto;
- Usar política de *write-through*;
    - No caso de escritas em posições que não estão na cache, deve-se gerar um miss, trazer o valor para a cache e reiniciar o processo de escrita;
- Imprimir ao final da execução:
    - Total de cache hits (escritas, leituras e total);
    - Total de cache misses (escritas, leituras e total);
    - Total de acessos à memória (hits + misses, quebrado por escritas, leituras e total);

1.2 Nota 7

Além dos itens do milestone anterior será exigido:

- Configurações de cache com mapeamento direto, totalmente associativa e associativa com n-vias;
    - Note que basta variar o n da versão com n vias e configurá-lo apropriadamente que o comportamento será exatamente o das caches de mapeamento direto ou de caches totalmente associativas;

1.3 Nota 9

Além dos itens dos milestones anteriores será exigido:

- Configuração de políticas de write-back além da write-through do milestone anterior;
- Algoritmos de substituição de linhas LRU, MRU, FIFO além do algoritmo aleatório;

1.4 Nota 11

Além dos itens dos milestones anteriores será exigido:

- Split cache - Uma cache de instruções e uma cache de dados de tamanhos e configurações independentes. Para saber se um acesso à memória é dados ou instrução veja mais abaixo o formato do arquivo de entrada.

1.5 Nota 13

Além dos itens dos milestones anteriores será exigido:

- Incluir a possibilidade de simulação de caches com múltiplos níveis. Neste caso apenas o primeiro nível precisa ser split, os demais podem ser unificados e devem poder ser configurados individualmente.
  - Dica: Se você fez um código limpo e bem organizado nos passos anteriores, este item deve sair quase que de graça!

1.6 Relatório

Independentemente do milestone alcançado, você deve entregar junto com o seu código um relatório de no máximo 5 páginas. O relatório deverá descrever sua implementação e trazer uma avaliação do comportamento de diferentes configurações da cache para as entradas de teste fornecidas. Converse com o professor da disciplina em caso de dúvidas.

2 Entrada

A entrada será dada por um arquivo texto que contém um acesso feito à memória por linha. O arquivo tem um formato CSV, o que deve evitar quaisquer problemas para a sua leitura. Ele tem o formato a seguir:

```
I,04002e1e,4
S,0402cb10,8
I,04002e22,4
L,0402bf48,8
I,04002e26,4
I,04002e2a,3
M,0402be20,8
```

```
I,04002e5f,7
L,0402c9e0,8
```

O formato é: `operação, endereço em hexa, tamanho em bytes`.
As operações são as seguintes:

- `I` - *instruction* - é uma leitura (instrução)
- `S` - *store* - é uma escrita (dados)
- `L` - *load* - é uma leitura (dados)
- `M` - *memory* - ao contrário do MIPS, na arquitetura x86 onde este trace foi gerado, é permitido um modo de endereçamento no qual instruções fazem escrita à memória diretamente. Assim, `M,0402be20,8` indica que foi feita uma escrita na posição `0402be20` de `8` bytes. A instrução que causou a escrita é a instrução que foi carregada no passo imediatamente anterior.

Alguns cuidados:

- A arquitetura x86 onde o trace foi gerado permite escritas e leituras não alinhadas. Ou seja, dependendo do tamanho da leitura/escrita e do alinhamento inicial é possível que efetivamente seja necessário acesso a mais de uma linha de cache.
- Os endereços são de 64 bits, apesar da maior parte dos endereços dos traces caberem em menos bits.

2.1 Entradas de teste

Essas são entradas de teste já formatadas e obtidas a partir do método usando o Valgrind descrito acima. **Atenção**, você é responsável por gerar casos de teste, manualmente se necessário, para testar *corner cases*. **Dica:** Lembre-se, se sua cache for pequena não é preciso que os testes sejam grandes. Testes automatizados te ajudarão muito no desenvolvimento dessa tarefa.

Tamanho	Programa	Acessos	Link (Bytes Compactado/Descompactado)
Minúsculo	Hello World	260K	hello.tar.xz (85,3 KiB/3,4 MiB)
Médio	`ls`	9,8M	ls.tar.xz (1,4 MiB/124 MiB)
Médio	`cal`	11M	cal.tar.xz (1,5 MiB/142 MiB)
Grande	`nano`	15,7M	nano.tar.xz (2,3 MiB/198 MiB)

3 A entrega

A entrega do código e do relatório deve ser feita pelo GitHub Classroom através do link https://classroom.github.com/a/-5PQS7rF. Será considerado como entrega o último *commit* (não esqueça de dar *push*) no repositório até a **data limite de 07/06/2020 às 23h00**.

Para discussões, dúvidas e comentários utilize o Discord em https://discord.gg/9RtRcx3.

3.1 Política de atrasos

Exercícios entregues com atraso sofrerão desconto nas notas conforme a tabela abaixo:

Dias em atraso	Fator aplicado à nota
1	0,7
2	0,6
3	0,5
>=4	0,0