Hierarquia de memória

O presente trabalho tem por objetivo explorar conceitos discutidos em sala de aula sobre hierarquia de memória. Deverá ser desenvolvida uma ferramenta através da qual o usuário poderá caracterizar programas a serem executados, a configuração da cache de nível 1 e as característica do restante da hierarquia de memória.

Descrição do trabalho

Conforme apresentado anteriormente, o trabalho é composto de três partes: (a) interface de caracterização de um programa, (b) a configuração da cache e (c) detalhes sobre a hierarquia de memória. Todos estes itens deverão trabalhar em conjunto para permitir que seja avaliado o impacto da hierarquia de memória.

Sobre a caracterização do programa

Deverá ser possível ler um arquivo texto que dá características de funcionamento de um programa. Um programa será caracterizado por três componentes, quais sejam: (a) o endereço final do programa, (b) onde estão localizados os saltos incondicionais e (c) onde estão localizados os saltos condicionais. Um exemplo de um arquivo de caracterização de um programa pode ser visto a seguir:

```
ep:100
ji:50:40
bi:45:51:10
```

No pseudo-código apresentado, pode-se observar os três componentes citados anteriormente. O comando ***ep*** define o final do programa. O comando ***ji*** define um saldo incondicional. O comando ***bi*** define um salto condicional.

O funcionamento básico da interpretação é tal que todo programa começa no endereço 0 e vai até o endereço definido por ***ep***. Quando do início da caracterização do programa, os endereços devem ser gerados sequencialmente.

Sobre o comando ***ep***:

- só pode existir um comando ep no arquivo de caracterização do programa
- o formado do comando é ep:<inteiro>
  - O inteiro representa o endereço final do programa
  - Sempre que o endereço final for alcançado ou superado, o programa tem fim

Sobre o comando ***ji***:

- Como dito, representa um salto incondicional

- Seu formato é ji:<adr>:<tgt>
  - adr é um inteiro e representa o endereço onde o salto incondicional existe
  - tgt é o endereço destino do salto incondicional
- Pode ser realizado um salto para um endereço posterior ou anterior a localização do comando

Sobre o comando ***bi***:

- Como dito, representa um salto condicional
- Seu formato é bi:<adr>:<tgt>:<pbd>
  - adr é um inteiro e representa o endereço onde o salto incondicional existe
  - tgt é um inteiro e representa o endereço de destino do salto condicional
  - pbd é a probabilidade de o salto ocorrer. Deve ser representado por um número inteiro entre 0 e 100, onde 0 significa que o salto nunca ocorrerá e 100 significa que o salto sempre ocorrerá.
- Assim como no salto incondicional, o ***bi*** pode caracterizar um salto para um endereço posterior ou anterior a localização do comando

O *trace* de endereços gerados pelo interpretador do programa deve ser capaz de gerar um arquivo texto com a sequência de endereços gerados para o experimento. O arquivo deve ter a lista de endereços em decimal, sendo cada endereço em uma linha. Como a caracterização do programa pode levar a um conjunto muito grande de endereços sendo gerado, deve ser possível informar um número máximo de endereço que serão gerados. Assim, ao final da interpretação do arquivo de caracterização do programa e geração da sequência de endereços, o número de endereços gerados pode ter sido definido pelo número máximo de endereços permitido ou por que o endereço final do programa foi alcançado.

Sobre a configuração da cache

Para a realização deste trabalho, será dado foco na cache de mapeamento associativo. Deve-se poder informar o tamanho em bytes da cache, o número de palavras por bloco, o tamanho da palavra e o número de vias. Considerem que o bit de validade reside na memória associativa, que não contabiliza em número de bytes da cache. Assim, uma cache associativa que armazena palavras de 4 bytes, com 2 palavras por bloco (i.e. 8 bytes de dados por linha) e que tenha 32 bytes de cache, construído a partir de 2 vias, deverá ser construído a partir de 4 linhas e 2 conjuntos associativos.

Deverão ser disponibilizados ao menos dois algoritmos de substituição: um do tipo LRU ou LFU, e outro do tipo Relógio, randômico ou sequencial. Quando da simulação, um destes algoritmos deverá ser escolhido.

Detalhamento da hierarquia

Conforme visto em sala de aula, uma hierarquia de memória é composta por um conjunto de memórias com capacidade de armazenamento crescente, mas com desempenho decrescente. No desenvolvimento deste trabalho, isto deve ser levado em consideração. A cache configurada anteriormente é a memória mais próxima do processador e seu atraso é nulo em relação ao processador. Isto significa que cada *hit* ocorrido naquela cache ocorre em uma unidade de tempo (1ut), que é o mesmo do processador. Todavia, quando um *miss* ocorre, há uma penalidade para cada nível da hierarquia da memória que é acessada. Para considerar este atraso, deverá ser incluído um arquivo que dá detalhes sobre a penalidade de acesso a cada nível inferior de memória e a probabilidade do endereço que se busca estar disponível naquele nível. O conteúdo do arquivo deve ser similar a figura a seguir.

```
CL2:10:20
CL3:30:20
MR:100:40
HD:1000:100
```

Ali é descrita uma hierarquia de memória que (além da cache configurada por padrão) lista outros quatro níveis de memória. O formato de descrição de cada nível da hierarquia de memória é <nome>:<penalidade>:<probabilidade>, onde

- Nome: define uma nomenclatura a ser usada
- Penalidade: define a quantidade de unidades de tempo decorrente de ter acessado este nível da hierarquia.
- Probabilidade: Define a probabilidade de o endereço ser encontrado naquele nível. O valor é entre 0 e 100, onde 0 significa que não há a menor chance de o endereço ser encontrado naquele nível e 100 significa que o endereço certamente será encontrado naquele nível

É importante lembrar que sempre deverá ser incluído um nível onde a probabilidade de encontrar um determinando endereço seja 100. Outro ponto importante a ser considerado é a ordem com que as memórias são listadas no arquivo. As do topo são as mais próximas ao processador, enquanto as mais abaixo são as mais afastadas do processador.

Detalhamento sobre a execução do sistema

Para a validação do sistema, deverá ser possível ler tanto um arquivo texto que valores que conterá uma sequência de endereços representando as requisições feitas por um processador para acesso à memória. Será assumido que um endereço conterá 32 bits de espaço de endereçamento. Cada endereço requisitado pelo processador será disponibilizado linha a linha, sendo tal endereço apresentado em decimal. Sendo assim, apesar de ser possível caracterizar um programa, é mandatória a geração de um arquivo de trace que deverá ser consumido pelo sistema desenvolvido.

A sequência de requisições do processador deverá ser executada sobre a arquitetura de cache configurada. A cada endereço requisitado deverá ser anotada se ela gerou um acerto ou uma falha na cache, conforme os critérios estabelecidos em sala de aula.

Resultado da execução do sistema

Ao final, deverá ser entregue o resultado contendo:

- Informar o número de conjuntos associativos
- O formato de interpretação do endereço (tag, conjunto, bloco)
- O número/taxa de acerto por nível hierárquico
- O número/taxa de falha por nível hierárquico
- O tempo médio de acesso
- O tempo total de execução

Entregáveis

O trabalho será julgado a partir dos códigos fonte e de um documento no qual deverá ser relatado detalhes da implementação. As seguintes informações são requeridas no documento:

1) Introdução
   - Resumo da ferramenta
   - Organização do documento
2) Compilação
   - Ambiente de desenvolvimento
   - Método de compilação
3) Interface
   - Descrição de como manipular a ferramenta
   - Detalhamento sobre os campos a serem preenchidos
   - Detalhamento sobre como carregar arquivos de teste
   - Detalhamento sobre os resultados obtidos da operação
4) Exemplo de uso
5) Avaliação de resultados
   - Deve-se assumir um ou mais programas a serem avaliados
   - Fazer uma comparação com execução em diferentes configurações
6) Conclusão

A escolha da linguagem e o do ambiente de desenvolvimento fica a critério do aluno. O trabalho pode ser desenvolvido em duplas.

Resumindo, deve-se entregar ao professor relatório com:

1 – Código Fonte
2 – Documentação (PDF)

Este deve ser entregue até o dia definido na agenda da disciplina em um arquivo compactado contendo todos os itens acima. Para trabalhos realizados em grupo, a entrega por apenas um dos elementos do grupo é suficiente

A entrega deverá ser feita via moodle, até o dia previsto na agenda da disciplina com o fechamento da sala ocorrendo no horário de início da aula.

Trabalhos que não compilam, não executam ou que forem considerados plágio serão automaticamente anulados.