Especificação do Trabalho – Implementação de um Simulador de Caches

Objetivos:

- Implementação de um simulador de caches parametrizável em uma linguagem de programação.
- Os grupos serão compostos por 2 alunos.

Implementação do Simulador

Deverá ser implementado um simulador funcional de caches em uma linguagem de programação (livre escolha), este simulador deverá ser parametrizável (configurações da cache), quanto ao número de conjuntos, tamanho do bloco, nível de associatividade e política de substituição. Considere que a cache é endereçada a bytes e o endereço possui 32 bits.

A configuração de cache deverá ser repassada por linha de comando e formatada com os seguintes parâmetros (o arquivo de entrada poderá ter extensão):

cache_simulator <nsets> <bsize> <assoc> <substituição> <flag_saida> *arquivo_de_entrada*

Onde cada um destes campos possui o seguinte significado:

- ***cache_simulator*** - nome do arquivo de execução principal do simulador (todos devem usar este nome, independente da linguagem escolhida;
- ***nsets*** - número de conjuntos na cache;
- ***bsize*** - tamanho do bloco em bytes;
- ***assoc*** - grau de associatividade;
- ***substituição*** - política de substituição, que pode ser Random (R), FIFO (F) ou L (LRU);
- ***flag_saida*** - flag que ativa o modo padrão de saída de dados;
- ***arquivo_de_entrada*** - arquivo com os endereços para acesso à cache.

O tamanho da cache é dado pelo produto do número de conjuntos na cache (<nsets>), tamanho do bloco em bytes (<bsize>) e associatividade (<assoc>).

Para caches associativas, a política de substituição default será randômica. A implementação da opção de escolha por outros modos será considerada como bônus.

Formato de saída:

A saída do simulador será um relatório de estatísticas com o número total de acessos, número total *hits* e *misses* (os *misses* deverão ser classificados em compulsórios, capacidade e conflito), que poderá ser exibido em dois formatos, de acordo com o valor do campo *flag_saida*:

- *flag_saida* = 0
  Formato livre, pode conter textos com labels para facilitar a visualização. Ex: Taxa de hits = 90%.
- *flag_saida* = 1
  Formato padrão de saída que deverá respeitar a seguinte ordem: Total de acessos, Taxa de hit, Taxa de miss, Taxa de miss compulsório, Taxa de miss de capacidade, Taxa de miss de conflito
  Ex: 100000, 0.95, 0.05, 0.01, 0.02, 0.03

Arquivo de entrada:

O arquivo de entrada será utilizado como entrada para o simulador (armazenado em formato binário) que conterá os endereços requisitados à cache (endereços em 32 bits). Dois arquivos de teste serão fornecidos para auxiliar na verificação dos códigos, um deles com 1000 e outro com 10.000 endereços, cada um deles estará disponível no formato binário exigido, e também em .txt para facilitar o entendimento.

Exemplo 1:
Considerando a seguinte linha de comando, utilizando o arquivo de entrada "bin_100.bin":

cache_simulator 256 4 1 R 1 bin_100.bin

O resultado esperado para a saída é: 100, 0.68, 0.32, 1.00, 0.00, 0.00

Exemplo 2:
Considerando a seguinte linha de comando, utilizando o arquivo de entrada "bin_1000.bin":

cache_simulator 256 4 1 R 1 bin_1000.bin

O resultado esperado para a saída é: 1000, 0.7020, 0.2980, 0.83, 0.00, 0.17

Exemplo 3:
Considerando a seguinte linha de comando, utilizando o arquivo de entrada "bin_10000.bin":

cache_simulator 256 4 1 R 1 bin_10000.bin

O resultado esperado para a saída é: 10000, 0.9153, 0.0847, 0.30, 0.00, 0.70

Estes resultados devem ser usados como referência para os testes preliminares do código antes do envio.

Envio dos códigos:

Cada dupla deverá compactar **APENAS** o código fonte (e executáveis, caso haja) em um arquivo nomeado com o número de matrícula de um dos membros da dupla e enviar o arquivo pelo AVA.
Ex: 12345678.zip

Recomendo que um arquivo "readme.txt" seja incluído neste arquivo compactado, descrevendo as eventuais funcionalidades adicionais (por exemplo, possui mais de uma política de substituição?), bem como eventuais informações úteis para a compilação (precisa de alguma biblioteca específica?) e execução do código.

Avaliação:

A avaliação será realizada a partir dos códigos enviados em duas etapas. A primeira, será automatizada, identificando códigos "muito similares" com os demais entregues (neste semestre ou em semestres anteriores) e rodando testes preliminares com alguns benchmarks. A segunda será manual, a com foco nos casos apontados pela primeira avaliação.

Havendo necessidade, os alunos serão chamados para uma apresentação de seus trabalhos. Alunos que tiverem interesse em apresentar seus trabalhos também terão a oportunidade de fazê-los, sob demanda, entrando em contato direto com o professor para agendamento.