Universidade Federal do Rio Grande do Norte
DIM0121 – Arquitetura de Computadores
Descrição dos Trabalhos

1. Grupos de NO MÁXIMO 3 alunos.
2. Cada trabalho poderá ser selecionado por, no máximo, 3 grupos.
3. A formação do grupo (mesmo que individual) e o trabalho por ele a ser executado deverá ser informado ao professor até no mínimo 2 semanas antes do primeiro dia de apresentação dos trabalhos. Em caso de descumprimento ou omissão, a nota 0.0 será atribuída a todos os membros do grupo.
4. Todos os integrantes do grupo serão avaliados individualmente, devem saber responder as perguntas relativas a todo o trabalho e devem estar presentes no sorteio para a apresentação.
5. O sorteio do grupo ocorrerá na sala de aula, nos dias da apresentação. Após o sorteio, um grupo deverá apresentar o trabalho e responder os questionamentos que possam surgir. Terminado, o próximo grupo será sorteado e o clico de apresentação retomado.
6. O tempo disponível de cada grupo será definido 2 semanas antes do primeiro dia da apresentação, a depender da quantidade de grupos, sendo a primeira metade para a apresentação, contado a partir do sorteio do grupo, e a segunda metade para questionamentos.
7. A entrega do trabalho final é considerada completa com a entrega de três itens:
   a) Implementação
   b) Apresentação
   c) Relatório final
8. A NÃO entrega de um dos itens listados acima implica em desclassificação do grupo e nota **0,0** para todos os integrantes.
   a. O trabalho deverá ser enviado pelo SIGAA;
   b. Seu prazo de entrega estará no SIGAA;
   c. Nenhum trabalho será recebido fora do período ou por outro meio de comunicação que não o SIGAA;
   d. Qualquer alteração da data de submissão do trabalho será analisada sob rígidos critérios.
9. O relatório final deve conter:
   a. Introdução: descrição do trabalho e sua contextualização com o estudo de arquiteturas de computadores.
   b. Solução Implementada: detalhes sobre a implementação.
   c. Análise de resultados: resultados de simulação, tempo, área e demais aspectos a serem considerados devem ser reportados e analisados.

Gráficos de análise, como barras, pizza, dentre outros podem (e devem) ser utilizados para auxiliar na análise dos resultados.

d. Conclusão: breve descrição do que foi implementado com análise, também breve, dos resultados.
e. Todo material utilizado: livros, links, códigos, e qualquer outro material deve ser descrito da seção de Referências.
f. Descrição de organização do código: pastas, subpastas, etc.
g. Descrição de como fazer para compilar e executar o projeto

10. Casos de plágio e cópias não referenciadas devidamente serão tratados com reprovação dos integrantes do grupo e denúncia aos órgãos competentes.
11. Exceções e casos omissos devem ser tratados diretamente com o professor da disciplina.

Simulador de Memória Virtual

Descrição: implementar um simulador de memória virtual que realize acessos à memória, tanto leitura quanto escrita, com base no endereço virtual, tabela de páginas e acesso à endereço físico.

O simulador receberá como entrada um arquivo que conterá a sequência de endereços de memória acessados. Esses endereços estarão escritos como números hexadecimais, seguidos por uma letra R ou W, para indicar se o acesso foi de leitura ou escrita.

Ao iniciar o programa, será definido o tamanho da memória (em quadros - *frames*) para aquele programa e qual o algoritmo de substituição de páginas a ser utilizado. O programa deve, então, processar cada acesso à memória para atualizar os bits de controle de cada *frame*, detectar faltas de páginas (*page faults*) e simular o processo de carga e substituição de páginas. Durante todo esse processo, estatísticas devem ser coletadas, para gerar um relatório curto ao final da execução.

Forma de operação

O programa deverá ser iniciado com quatro argumentos:

virtual lru arquivo.log 4 128

Esse argumentos representam, pela ordem:

1. o algoritmo de substituição a ser usado (lru, fifo, nru, lfu e random);
2. o arquivo contendo a sequência de endereços de memória acessados (arquivo.log, nesse exemplo);
3. o tamanho de cada página/frame de memória, em kilobytes -- faixa de valores razoáveis: de 2 a 64;
4. o tamanho total da memória física disponível para o processo, também em kilobytes -- faixa de valores razoáveis: de 128 a 16384 (16 MB).

Formato da saída

Ao final da simulação, quando a sequência de acessos à memória terminar, o programa deve gerar um pequeno relatório, contendo:

- a configuração utilizada (definida pelos quatro parâmetros);
- o número total de acessos à memória contidos no arquivo;
- o número de *page faults*;
- o número de páginas ”sujas” que tiveram que ser escritas de volta no disco (lembrando-se que páginas sujas que existam no final da execução não precisam ser escritas).

Um exemplo de saída poderia ser da forma (valores completamente fictícios):

```
prompt> virtual lru arquivo.log 4 128
```

```
Executando o simulador...
Arquivo de entrada: arquivo.log
Tamanho da memoria: 128 KB
Tamanho das páginas: 4 KB
Técnica de reposição: lru
Paginas lidas: 520
Paginas escritas: 352
```

Formato do arquivo de entrada

Como mencionado anteriormente, cada linha contém um endereço de memória acessado, seguido das letras R ou W, indicando um acesso de leitura ou escrita, respectivamente. Por exemplo, as linhas a seguir foram retiradas de um dos arquivos utilizados:

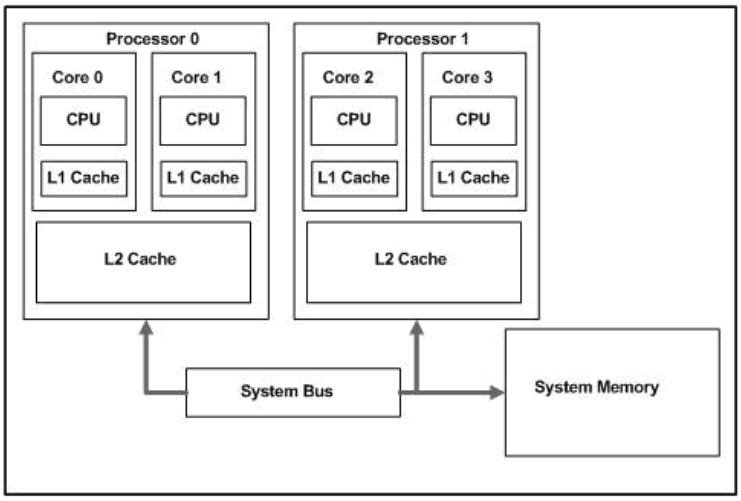
```
0785db58 W
000652d8 R
0005df58 W
000652e0 R
0785db50 W
000652e0 R
31308800 R
00062368 R
```

O trabalho deverá conter uma análise estatística das técnicas que foram implementadas. Para tal, os arquivos de testes (pelo menos 3 arquivos diferentes) devem conter muitas instruções R e W (pelo menos 1000 em cada arquivo, no total) e devem ser testadas sobre **diversas configurações diferentes** do simulador sobre os itens *tamanho da memória*, *tamanho das páginas*, *técnicas de reposição*.

O documento deve explorar gráficos para resumir informações complexas e simplificar o entendimento do texto. Cada gráfico e análise deve ser explicada e, se aplicável, possuir conclusões a respeito de suas informações.

A
Simulador de hierarquia de memória em Multicore

Descrição: o sistema de memória terá dois níveis de cache (L1 e L2) e a memória principal. Cada core terá uma cache L1. Uma cache L2 será compartilhada entre dois cores. A memória principal será compartilhada por todos os cores. Observar figura abaixo.



No simulador, o usuário informará quantos cores o sistema terá (somente quantidades múltiplas de 2 são permitidas). O sistema automaticamente cria as caches e a memória principal. Um arquivo, lido pelo simulador, carrega os dados na memória principal. ***O usuário*** *informa o endereço da memória principal para leitura e o core que irá utilizar aquele dado*. O sistema automaticamente carrega o dado (o bloco que contém o dado requerido) na cache L2 e na cache L1 do respectivo core. O sistema também deve ser capaz de atualizar o dado, caso o core modifique esse valor, e atualizar todos os níveis da hierarquia imediatamente (*write-through*).

Atenção:

- Assumir que os dados são números inteiros e explore o princípio da localidade;
- Assuma que o tamanho da linha de cache L2 é um múltiplo do tamanho da linha de cache L1. Assuma também que o tamanho do bloco da memória é um múltiplo do tamanho da linha de cache L2;
- O sistema deverá dar opção de **ler** ou **escrever** o dado, indicando qual core fará a operação
    - Em caso de **leitura**: o dado (através de seu endereço de memória) será carregado nas caches (se ainda não estiver)
    - Em caso de **escrita**: o dado (através de seu endereço de memória) será alterado e atualizado em todos os níveis da hierarquia.
    - Coerência de cache deve ser implementada: Qualquer dado sujo terá toda a sua linha de cache “apagada”.
- A hierarquia de memória deve ser mostrada a todo momento de execução.

Implementar um jogo em Assembly-MIPS usando o MARS

Jogos possíveis: É obrigado usar o simulador de interface gráfica do MARS, exceto se dito contrário no jogo. Jogos extras poderão ser conversados com o professor.

a) Gênius: versão para 2 jogadores.

b) Resta 1.

c) Pong: versão para 2 jogadores.

d) Batalha naval 10x10. Versão para 1 jogador. Inicialização do campo aleatória

e) Jogo da velha 6x6: Todos os estados das células do jogo devem ser gravados na memória. Os registradores são utilizados para realizar as ações.

f) Campo minado: sem necessidade de modo gráfico. Inicialização aleatória.

g) BlackJack: versão para 2 jogadores sem necessidade de modo gráfico.