UNIVERSIDADE FEDERAL DE LAVRAS

Departamento de Ciência da Computação

Disciplina: **Arquitetura de Computadores II - GCC123**

5ª Lista de Exercícios

Professor: **Luiz Henrique A. Correia**

Data de entrega: 29/04/22

Observação: a lista deve ser MANUSCRITA, digitalizada e enviada para o Campus Virtual.

1. As máquinas virtuais podem perder desempenho devido a uma série de eventos, como a execução de instruções privilegiadas, faltas de TLB, traps e E/S. Esses eventos normalmente são tratados no código do sistema. Assim, um modo de estimar a lentidão na execução sob uma VM é a porcentagem de tempo de execução da aplicação no sistema contra o modo usuário. Por exemplo, uma aplicação gastando 10% de sua execução no modo do sistema poderia retardar em 60% quando fosse executada em uma VM. A Figura 1 lista o desempenho inicial de diversas chamadas de sistema sob execução nativa, virtualização pura e paravirtualização para LMbench usando Xen em um sistema Itanium com tempos medidos em microssegundos.

   a) Que tipos de programa poderiam ter maior lentidão quando executados sob VMs?

   b) Se a lentidão fosse linear, como função do tempo do sistema, dada a lentidão anterior, quão mais lentamente um programa será executado se estiver gastando 20% de sua execução no tempo do sistema?

   c) Qual é a lentidão média das funções na tabela sob a virtualização pura e paravirtualização?

   d) Quais funções da tabela possuem os menores atrasos? Qual você acha que poderia ser a causa disso?

Benchmark	Nativa	Pura	Para
Null call	0,04	0,96	0,5
Null I/O	0,27	6,32	2,91
Stat	1,1	10,69	4,14
Open/close	1,99	20,43	7,71
Install sighandler	0,33	7,34	2,89
Handle signal	1,69	19,26	2,36
Fork	56,00	513,00	164,00
Exec	316,00	2.084,00	578,00
Fork + exec sh	1.451,00	7.790,00	2.360,00

Figure 1: Desempenho inicial de diversas chamadas do sistema sob execução nativa, virtualização pura e paravirtualização.

Respostas:
Exercício 2.21 do livro.
a) Programas que possuem frequentes chamadas de sistema (system calls) ou constantes acessos a dispositivos externos (I/O).
Programas que fazem cálculos exaustivos mas que fazem muito acesso a memória principal ou que as VM possuam pequenos conjuntos de memórias.
b) 10% de execução –> 60% de lentidão
=> 120% 20% de execução – > X
c) A média de slowdown usando virtualização pura é 10,3 e para virtualização é 3,76.

d) As chamadas Null call e Null I/O possuem os menores atrasos. Eles não têm trabalho real para compensar a sobrecarga de virtualização de alterar os níveis de proteção, portanto, eles têm as maiores lentidões.

2. A definição de uma máquina virtual de Popek e Goldberg estabelecia que ela seria indistinguível de uma máquina real, exceto por seu desempenho. Neste exercício, usaremos essa definição para descobrir se temos acesso à execução nativa em um processador ou se estamos executando em uma máquina virtual. A tecnologia VT-x da Intel, efetivamente, oferece um segundo conjunto de níveis de privilégio para o uso da máquina virtual. O que uma máquina virtual sendo executada sobre outra máquina virtual precisaria fazer, considerando a tecnologia VT-x? *Respostas:*
Exercício 2.22 do livro.
A máquina virtual rodando em cima de outra máquina virtual teria que emular níveis de privilégio como se estivesse rodando em um host sem tecnologia VT-x.

3. O processador Core I7 da Intel possui uma hirarquia de memória com três níveis de cache. Todas as memórias cache são Write-Back e com blocos de 64 bytes. As características dessa hierarquia de memória são apresentadas na Tabela 1. Observando os dados da tabela responda:

   a) Quantos bits de índice são usados para o endereçamento de cada cache (L1-I/D, L2 e L3)?

   b) Porque a associatividade é dobrada para cada nível da cache?

   c) Porque a latência aumenta a medida que descemos na hierarquia de memória?

   d) Explique o sistema de substituição (*Replacement scheme*) empregado para cada nível da cache.

Table 1: Core I7: hierarquia de memória

Características	**L1**	**L2**	**L3**
Tamanho	32KB - I 32KB - D	256KB	2 MB por Core
Associatividade	4-way - I 8-way - D	8-way	16-way
Latência de acesso	4 ciclos, pipelined	10 ciclos	35 ciclos
Técnica de substituição	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU, mas com um algoritmo de reordenação

a) Para L1-I: $2^i = \frac{32KB}{64B \times 4} = \frac{32768B}{256B} = 128 = log_2\ 128 = 7\ bits$
Para L1-D: $2^i = \frac{32KB}{64B \times 8} = \frac{32768B}{512B} = 64 = log_2\ 64 = 6\ bits$
Para L2: $2^i = \frac{256KB}{64B \times 8} = \frac{262144B}{512B} = 512 = log_2\ 512 = 8\ bits$
Para L3: $2^i = \frac{2MB}{64B \times 16} = \frac{2097152B}{1024B} = 2048 = log_2\ 2048 = 11\ bits$

b) A associatividade é aumentada a cada nível para evitar misses de capacidade e garantir que a quantidade de misses seja menor, já que existem mais posições disponíveis por bloco.

c) A latência aumenta por dois motivos:
- maior capacidade de memória maior a latência de leitura.
- por causa do comprimento do barramento.

d) O esquema empregado para substituição dos blocos é baseado no LRU (*Least Recently Used*), o bloco a ser substituído é aquele que não é usado a mais tempo.

4. Uma hierarquia de memória é organizada como mostrado na Figura 5b, incluindo uma TLB e uma cache. Uma referência à memória pode encontrar três tipos de faltas: uma falta no acesso à cache, uma falta no acesso à TLB e/ou uma falta de página. Considere todas as combinações desses três eventos com uma ou mais ocorrências, como mostrado na Tabela 2. Para cada uma das possibilidades, informe se tal evento pode ou não ocorrer na prática, e em quais circunstâncias.

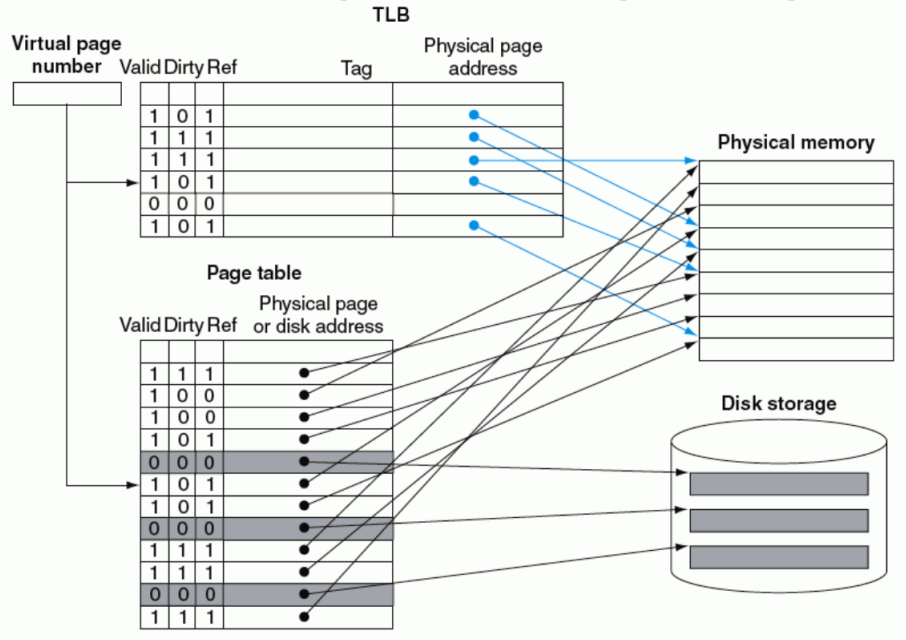


Figure 2: Hierarquia de Memória.

TLB	Page Table	Cache	Possível? Se for, em que circunstâncias?
Hit	Hit	Miss	Yes, although the page table is not checked if the TLB
Miss	Hit	Hit	Yes, TLB miss, PA in page table
Miss	Hit	Miss	Yes, TLB miss, PA in page table, but data not in cache
Hit	Miss	Miss/Hit	Impossible, TLB translation not possible if page is not present in memory

Table 2: Combinação de eventos na TLB.

5. Uma série de referências e endereços de palavras é dada por:
$$2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, 11$$
Considerando uma cache mapeada diretamente com 16 blocos de uma palavra, que estava inicialmente vazia, identifique cada referência listada como *miss* ou *hit* no acesso à cache, preenchendo a tabela abaixo.

   (a) Mostre o processamento de todas essas referências (riscando cada uma que foi sobrescrita) e o conteúdo final da cache.

   (b) Para uma cache com associatividade 4-way e um total de 16 palavras. Desenhe uma tabela e mostre o processamento de todas essas referências (riscando cada uma que foi sobrescrita) e o conteúdo final da cache. Lembre-se que a localização do endereço será dada por (Endereço *mod* Número de conjuntos na cache).

Cache Set	Address
0	~~16~~, ~~64~~, 48,
1	
2	2
3	~~3~~,~~19~~,3
4	4
5	21
6	~~22~~,6
7	
8	
9	
10	
11	~~11~~,27
12	
13	13
14	
15	
16	

2	miss
3	miss
11	miss
16	miss
21	miss
13	miss
64	miss
48	miss
19	miss
11	hit
3	miss
22	miss
4	miss
27	miss
6	miss
11	miss

Cache Set	Tag	Address
00	[0,1,2,3]	[16,64,48,4]
01	[0,1,2,3]	[21,13,
10	[0,1,2,3]	[2,22,6,
11	[0,1,2,3]	[3,11,19,27]

2	miss
3	miss
11	miss
16	miss
21	miss
13	miss
64	miss
48	miss
19	miss
11	hit
3	hit
22	miss
4	miss
27	miss
6	miss
11	hit

6. Uma memória cache está organizada em 512 linhas de 64 bytes cada. Sejam duas matrizes A e B ($m = 8 \times n = 8$), cujos elementos armazenados estão representados em **precisão dupla**. Responda:

   a) Qual a capacidade da memória cache?

   b) Se a memória cache possui blocos de 16 palavras de 32 bits, quantos *misses* e *hits* de leitura irão ocorrer para o produto $A \times B$? Indique os valores para cada matriz A e B.

   c) Que tipo de localidade cada matriz aproveita?

   d) Explique como a otimização do produto linha x linha para $A \times B^T$ pode tornar a computação do produto mais rápida?

   Respostas:

   a) Capacidade da memória: linhas x palavras por linha = 512 x 64 bytes = 32768 bytes = 32 KB.

   b) Cada linha da matriz tem 64 bytes, ou 16 palavras de 32 bits. Logo, cada linha da memória tem capacidade de armazenar 16/2 = 8 palavras de precisão dupla - DW.
   Para a matriz A cada linha da memória irá armazenar 8 palavras de precisão dupla. Logo, para cada linha buscada na memória haverá 1 Miss para a primeira palavra DW, seguidos de 7 hits para os elementos de $a_{1,1}$ até $a_{1,8}$. Dessa forma para a matriz A, com 8 linhas haverá: 1 Miss por linha = 8 misses e 7 hits por linha = 7 x 8 = 56 Hits.

Para a matriz B, os elementos da primeira coluna e os subsequentes estão armazenados por coluna, por exemplo $b_{1,8}$ até $b_{8,1}$. Dessa forma, para a matriz B, com 8 colunas haverá: 1 Miss por elemento x 8 linhas x 8 colunas = 64 misses.

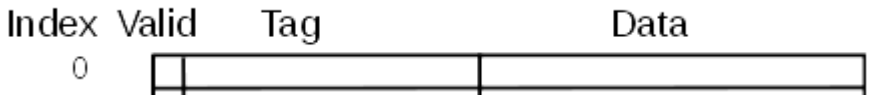
c) *A matriz A aproveita a localidade espacial, já que ao buscar a primeira palavra, traz todo o bloco de 16 bytes, ou 8 palavras DW. Já a matriz B não aproveita nenhum tipo de localidade.*

d) *A otimização de transpor a matriz B, rearranja os elementos de B por linha, e a multiplicação do produto por linha melhora a localidade espacial, já que todos os elementos da matriz são rearrajandos nos blocos da memória. Haverá, somente 1 Miss e 7 hits por bloco para cada matriz, logo: 8 misses de A + 8 misses de B = 16 misses e 7 hits x 8 blocos x 2 matrizes = 112 hits.*

7. Considere uma cache de 16K blocos e endereços de 32 bits da memória principal, responda:

   (a) Qual o número de bits de índice e o número de bits de rótulo para os seguintes mapeamentos: Mapeamento Direto, Associatividade 2, Associatividade 4 e Totalmente Associativo.

   (b) Represente por meio de uma figura os difentes tipos de mapeamento, considerando os campos: Index, Valid bit, Tag e Data.



Resposta:
a) Mapeamento Direto: $16Kblocos = 16384 = log_2\ 16384$
Índice = 14 bits
Rótulo = $32 - 14 - 2 = 16\ bits$
2 Way: $16Kblocos\ /\ 2 = 8192 = log_2\ 8192$
$ndice = 13bits$
$Rtulo$ =32 - 13 - 2 = 17 bits

Totalmente Associativay: Índice = 0 bits
Rótulo = 32 -0 - 2 = 30 bits

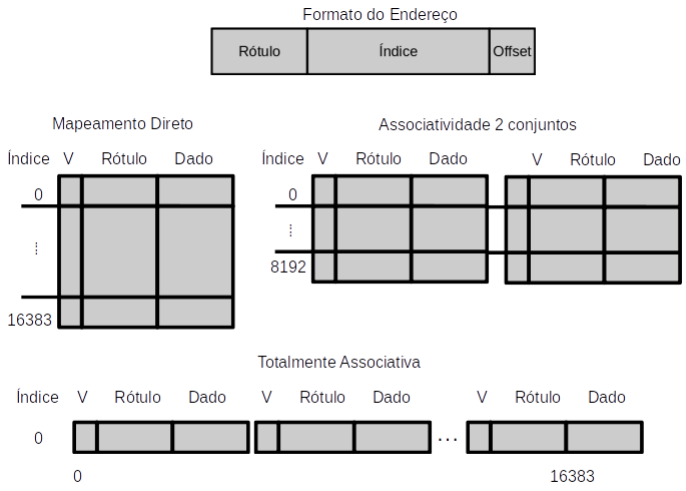


Figure 3: Associatividade de Memórias.

8. Dada a Figura 4 descreva detalhadamente os passos para tradução de endereços de acordo com os passos:
   (a) 1.a *O endereço foi encontrado na TLB e foi traduzido de 123456 para 987654*
   (b) 2.a *Houve um hit na memória para o endereço 987654.*
   (c) 2.b *O endereço não está na TLB e foi traduzido na Page Table.*
   (d) 3.a *O endereço foi traduzido com sucesso e atualizou a TLB, indo para os passos de 1 e 2.a.*
   (e) 3.b *O endereço não é válido e o SO sinalizou Segmentation Fault.*
   (f) 3.c *O endereço não foi encontrado na Page Table e houve falta de página, sendo que a página deve ser buscada na memória de armazenamento.*
   (g) 4. *A página foi trocada, sendo copiado da memória de armazenamento para a memória principal.*
   (h) 5. *O endereço da nova página é atualizado na Page Table.*

9. Colete os dados da hierarquia de memória do seu computador e descreva:
   a) Níveis de cache (L1, L2...)
   b) Tipo de associatividade por nível.
   c) Tamanho das caches (KB).
   d) Tamanho da linha da cache.
   e) Quantidade de bits para os endereços virtual e físico da memória.
   f) Descreva o comando ou a aplicação usado.

References

[1] Hennessy, John L. and Patterson, David A.. Arquitetura de computadores: uma abordagem quantitativa. Vol. 5. Elsevier Brasil, 2014.

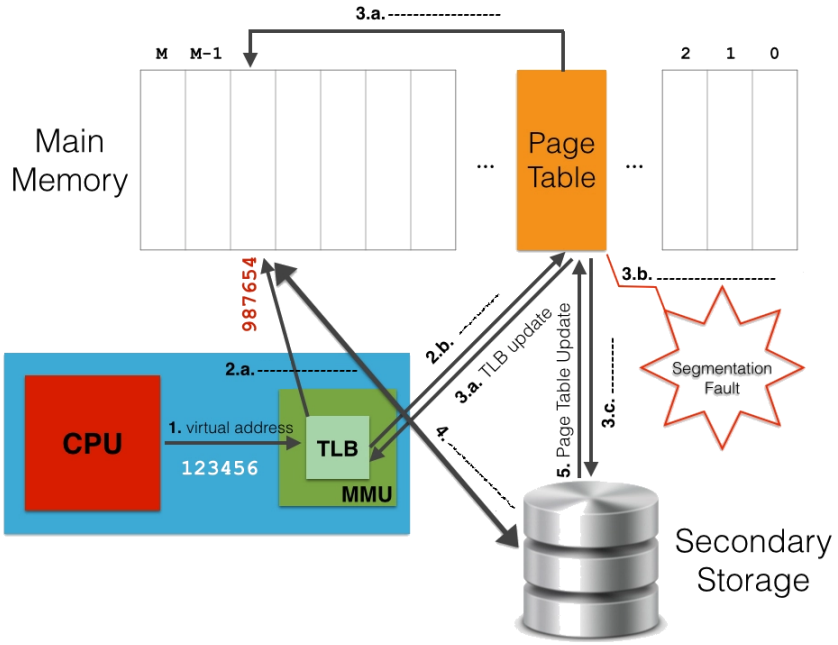


Figure 4: Tradução de endereço.