**Lista de Exercícios 3 – 2018-1**

**Engenharia da Computação**

**1.** Descreva os conceitos e cite as vantagens e desvantagens comparado ao seu “oposto”(a em relação a b || c em relação a d):

a. Política de escrita write-back;

*Resposta: A memória principal só é atualizada quando há uma substituição de bloco na cache, sendo assim, a cache e a memória ficam momentaneamente incosistentes.*

*Vantagen(s):*

* *Reduz o acesso à memória, diferentemente da Write-Trough que precisa sempre que há uma substituição de bloco na cache, atualizar também na memória;*

*Desvantagen(s):*

* *Difícil implementação;*

b. Política de escrita write-through;

*Resposta: Sempre que há uma mudança do bloco na cache, consequentementa a memória também é atualizada.*

*Vantagen(s):*

* *Cache e memória sempre consistentes, diferente da Write-Back que por algum tempo há inconsistência entre cache e memória principal;*
* *Fácil implementação, não tem a complexidade do Write-Back;*

*Desvantagen(s):*

* *Penaliza o desempenho devido aos vários acessos a memória para manter a consistência;*

c. Princípio da localidade temporal;

*Resposta: Programa tende a referenciar instruções/dados que foram referenciados recentemente (LOOPS).*

d. Princípio da localidade espacial.

*Resposta: Programa tende a referenciar instruções/dados que estão próximos das últimas instruções/dados referenciados (ARRAYS).*

**2.** Defina os seguintes conceitos fazendo uma relação, a otimização do sistema(quando cabível):

a. Tamanho da *cache*;

*Resposta: O ideal seria que o tamanho da cache fosse grande, pois são memórias mais rápidas no sistema de hierarquia de memória tendo assim uma melhora significativa no desempenho. Mas, aliada a velocidade, essas memórias também são mais caras que outros tipos de memória(DRAM, Disco rígido, etc), tendo assim, que trabalhar com pouca memória para baratear os custos, entretanto deve-se manter o desempenho alto. Isso é possível graças a hierarquia de memória.*

b. Tamanho do bloco da *cache*;

*Resposta: O tamanho do bloco aumentam\* o desempenho pois blocos maiores favorecem o princípio da localidade espacial. Com isso, há uma diminuição do Miss Rate da cache(porcentagem vezes que uma informação não é encontrada na memória). Em contraponto\*, também há uma degradação do tempo que um dado que não está na cache seja carregado da memória principal, pois o bloco maior demora mais tempo para ser transferido.*

c. Número de conjuntos de *cache*;

*Resposta: Número de conjuntos permite saber onde determinada informação da memória principal ficará na cache. Se dividem em Mapeamento direto(o endereço da memória, através dos bits menos significativos, apontam para uma posição única na cache, onde o bloco será armazenado) e associativa, que se divide em: associativa por conjuntos(o bloco que vem da memória pode ser armazenado e alguns locais da cache, dependendo do grau de associatividade) e completamente associativa(o bloco que vem da memória pode ser armazenado em qualquer lugar da cache). Quanto maior a associatividade, menor é o miss rate, entretanto aumenta o hit time, pois há um delay nas comparações para saber qual tag é a correta).*

d. Políticas de escrita;

*Resposta: A política de escrita podem ser Write Trough ou Write Back*

*Write Trough: Quando a uma mudança de informação na cache, há simultaneamente uma atualização do valor na memória.(degradação do desempenho pois há vários acessos a memória);*

*Write Back: Quando a uma mudança de informação na cache, a memória só irá ser atualizada quando houver alguma substituição de bloco na cache( melhora o desempenho, pois acessa menos a memória).*

*Nos dois casos há uma degradação de desempenho pois no momento que o dado for atualizado na memória, a CPU fica “congelada”. Uma solução é utilizar Write Buffers.*

e. Políticas de substituição.

*Resposta: A política de escrita se faz necessária quando a cache apresenta associatividade, pois nesse caso, tem que ser decidido em que bloco ocorrerá uma sobrescrita(no caso de não haver bloco vazio). Vários algorítimos podem ser adotados, aluns deles são: LRU e randômico.*

*LRU(Least Recentily Used): O bloco a ser substituído é que foi menos usado recentemente.(Vantagem: Localidade Temporal, Desvantagem: Quanto maior o grau, mais difícil implementar).*

*Diminui o miss rate, melhorando o desempenho.*

*Randômico: O bloco a ser substituído é escolhido randomicamente.*

*(Vantagem: Fácil implementação; Quanto maior o grau de associatividade, mais se aproxima do LRU em relação a queda de rate. Desvantagem: Em geral, há um aumento de miss rate em relação a outros algorítimos, principalmente para graus de associatividades menores)*

**3.** Organização por mapeamento direto e completamente associativa são casos particulares e extremos da organização associativa por conjuntos.  
Descreva como e por que motivos hit time, miss rate e o custo de hardware mudam conforme se varia o grau de associatividade entre esses dois extremos.

*Resposta*: *Mapeamento Direto: A posição do bloco na cache é mapeado a partir do endereço de memória. Nesse caso só existe uma única posição na cache em que o bloco da memória possa ser armazenado, determinado pelo index(bits menos significativos do endereço). Com isso, há um aumento do miss rate, pois como a memória principal é maior(hierarquia de memória) há mais de um endereço de memória que corresponde ao mesmo bloco na cache, ocasionando muitas faltas e consequentemente sobrescritas. Para saber qual endereço está na cache, deve-se comparar a tag desse endereço(bits mais significativos) com o que está na cache. No caso do mapeamento direto, só basta ter um comparador, pois cada endereço está em uma posição única na cache. Com isso, exige pouco hardware e o hit time é pequeno pois o delay da comparação é baixo, devido à existência de um único comparador.*

*Ao aumentar a associatividade, aumenta-se também a quantidade de posições em que um bloco pode ser armazenado na cache, sendo esta, dividida em conjuntos. Exempo.: tow-way set associative = dois conjuntos / four-way set associative = quatro conjuntos / Completamente Associativa = qualquer posição. Nesse caso, em comparação com mapeamento direto, há redução de faltas(miss rate) pois blocos com o mesmo index nem sempre são sobrescritos(pois há mais de uma posição onde os blocos possam ser armazenados) permanecendo assim, mais tempo na cache. O problema é que quanto maior a associatividade, mais comparadores são necessário para comparar as tags dos diferentes blocos, aumentando assim o hardware utilizado em comparação ao mapeamento direto. Outra consequência do aumento da associatividade, é o aumento do hit time, pois como há mais comparadores, aumenta o tempo que são realizadas as comparações.*

**4.** Uma cache está sendo projetada para um computador com 2³² bytes de memória, a qual terá 2K slots e usará blocos de 32 bytes. Calcule o número de bytes que a cache terá considerando as duas organizações: associativa de grau 4 e mapeamento direto.

*O endereçamento é de 32 bits(2³² bytes memória). Como existe 32 bytes em cada bloco, um endereço de 32 bits produz 32 - 5 = 27 bits para endereçar índice e Tag. Como no mapeamento direto não há divisão de slots por conjuntos, então os bits para identificar o índice é 2¹¹ (2K Slots), sobrando para a tag 16bits. Logo a quantidade em bits da cache com mapeamento direto é: 16(tag) \* 2K = 32Kbits. Convertendo para Bytes = 4Kbytes.*

*Ao aumentar o grau de associatividade em 2, dividisse o indicie por 2(pois não irá ter apenas um conjunto de slots) e aumente em 1 o número de bits que fica na tag(para melhorar a comparação). Logo numa memória Associativa de grau 4, ficaria:*

*18(tag)\* 4(número de conjuntos) \* 0,5K = 36Kbits. Convertendo para Bytes = 4.5KBytes.*

**5.** Considere a referências aos seguintes endereços de blocos de memória: 3, 9, 4, 20, 4, 2, 15, 5, 5, 20, 6, 1, 3 numa cache de 32 palavras e blocos de 4 palavras. Calcule o número de faltas, o tamanho da cache em bytes e o estado final da cache para cada uma das configurações (considere que a cache está inicialmente vazia e quando necessário uso como política de substituição o algoritmo LRU):

a. Mapeamento direto;

*Número de faltas = 9;*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** |  |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |
| **8** |  |  |  |  |
| **9** |  |  |  |  |
| **10** |  |  |  |  |
| **11** |  |  |  |  |
| **12** |  |  |  |  |
| **13** |  |  |  |  |
| **14** |  |  |  |  |
| **15** |  |  |  |  |
| **16** |  |  |  |  |
| **17** |  |  |  |  |
| **18** |  |  |  |  |
| **19** |  |  |  |  |
| **20** |  |  |  |  |
| **21** |  |  |  |  |
| **22** |  |  |  |  |
| **23** |  |  |  |  |
| **24** |  |  |  |  |
| **26** |  |  |  |  |
| **26** |  |  |  |  |
| **27** |  |  |  |  |
| **28** |  |  |  |  |
| **29** |  |  |  |  |
| **30** |  |  |  |  |
| **31** |  |  |  |  |

b. Associativa por conjunto de grau 4; 3, 9, 4, 20, 4, 2, 15, 5, 5, 20, 6, 1, 3

*Número de faltas = 9;*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** |  |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |
|  |  |  |  |  |
| **0** |  |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |
|  |  |  |  |  |
| **0** |  |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |
|  |  |  |  |  |
| **0** |  |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |

c. Completamente associativa. 3, 9, 4, 20, 4, 2, 15, 5, 5, 20, 6, 1, 3

*Número de faltas = 9;*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**6.** Considere a seguinte configuração de CPU com dois níveis de cache:

· CPIbase = 1

· Freqüência do clock = 2 GHz

· Hit timeL1 = 1 ns

· Hit timeL2 = 25 ns

· Miss rateL1 = 5%

· Miss rateL2 = 0,25%

· Tempo de acesso à memória principal = 150ns

a. Qual o CPI considerando um programa com 20% das instruções sendo de

load/store?

*CPI\_Total = CPIbase + Stalls Primário + Stalls Secundário*

*Stalls Primário = Miss rateL1 \* (Hit timeL2 / Periodo)*

*Stalls Primário = 5% \* (25ns / 0.08ns/ciclos)*

*Stalls Primário = 5% \* (25ns / 0.08ns/ciclos)*

*Stalls Primário = 15,625*

*Stalls Secundário = Miss rateL2 \* (Tempo de acesso à memória / Periodo)*

*Stalls Secundário = 0,25% \* (150ns / 0.08ns/ciclos)*

*Stalls Secundário = 4,6875*

*CPI\_Total = 1 + 15,625 + 4,6875 = 21,3125*

b. Qual seria o CPI para esse mesmo programa considerando uma CPU como a da questão, mas com apenas o primeiro nível de cache?

*CPI\_Total = CPIbase + Stalls Memória*

*Stalls Memória = Miss rateL1 \* (Tempo de acesso à memória / Periodo)*

*Stalls Memória = 5% \* (150ns / 0.08ns/ciclos)*

*Stalls Memória =* *93,75*

*CPI\_Total = 1 + 93,75 = 94,75*

c. Quais são os objetivos de se acrescentar níveis em uma cache?

Explique usando como exemplo os resultados nesta questão.