Aula de Revisão - 16/09

quarta-feira, 16 de setembro de 2020 08:06

Explique, como a memória cache melhora o desempenho de uma seção de código que exibe o princípio da localidade espacial da referência?

- Os módulos de memória cache estão localizados entre a CPU e a memória primária (SDRAM). É uma memória rápida (*zero wait states*) e com pouca capacidade de armazenagem. Para tirar vantagem da localidade da referência, a CPU faz uma cópia de dados para o cache sempre que um endereço referenciado não estiver presente no cache (*cache miss*). Desde que há uma grande probabilidade de que o sistema acesse endereços vizinhos ao endereço referenciado, o mecanismo cache faz uma transferência de um bloco de bytes sempre que ocorrer um "*cache miss*".

As principais propriedades de um sistema de hierarquia de memória são: inclusão, coerência e localidade de referência. A propriedade de coerência determina que cópias de um mesmo item devem ser consistentes ao longo de níveis sucessivos da hierarquia de memória. Há dois métodos básicos para manutenção da consistência entre os níveis da hierarquia – Quais são? E como eles operam?

- **write-through**:
  - ✓ atualização imediata em M_{i+1} quando item é modificado em M_i ;
- **write-back**:
  - ✓ atualização só é realizada em M_{i+1} quando o item estiver sendo retirado de M_i .

Considere uma mudança no tamanho do conjunto para a organização do cache associativo por conjuntos, onde a mudança de conjuntos de 2-vias (***2-way set associative***) para conjuntos de 4 vias (***4-way set associative***) apresenta uma pequena melhoria de desempenho a um pequeno acréscimo no custo.

a) Quais são os benefícios desta mudança?

b) O que contribui para o aumento de custo?

- a)
  - ✓ Um benefício imediato é o aumento das características associativas do método, diminuind as restrições de mapeamento devido à localidade da referência.
- b)
  - ✓ Aumentando o tamanho do conjunto tem por consequência uma redução no número de conjuntos e aumento no número de bits das etiquetas (***tags***) e também, um aumento no número de comparadores para detecção da presença do bloco.

Para esta tarefa, deve-se assumir que a memória principal consiste de **32** palavras (*words*) dividida em blocos de **2** palavras cada (16 blocos). A memória cache consiste de **4** linhas cache (slots). 4 BLOCOS

a) Desenhe um diagrama de blocos que ilustre esta configuração de memória.

b) Assumindo que uma função de mapeamento direto é utilizada:

i. Para cada slot no cache, liste os blocos que seriam mapeados para o slot.

ii. Quantos bits são necessários para um endereço? Liste os bits e indique qual parte do endereço seriam utilizados para (palavra, etiqueta, etc.).

c) Assumindo que uma função de mapeamento associativo é utilizada:

i. Quantos bits são necessários para um endereço? Liste os bits e indique qual parte do endereço seriam utilizados para (palavra, etiqueta, etc.).

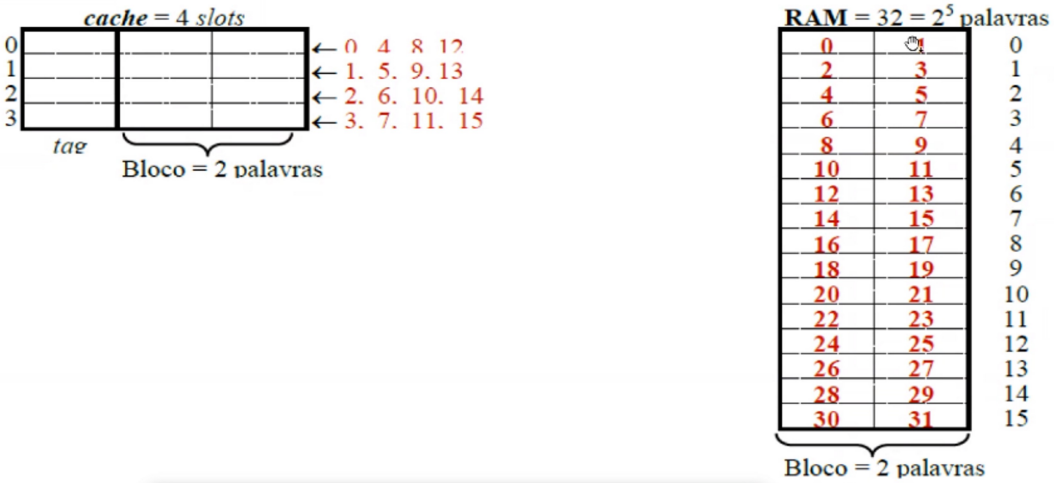
ii. Determine o tamanho deste cache considerando o mapeamento associativo.

d) Mostre o estado do cache após a CPU ter requerido as seguintes palavras em sequência: 8, 10, 12, 23, 9, 19, 28, 13, 7 e 11.

e) Considerando a sequência de palavras dada no item "c", e um tempo de acesso de **10ns** quando o dado for encontrado no cache e um tempo de acesso de **100ns** se o cache não contiver o dado. Qual é o tempo de acesso médio para esta sequência? (prestar atenção no processo de transferência de dados)

a) Desenhe um diagrama de blocos que ilustre esta configuração de memória.

Resposta:



b) Assumindo que uma função de **mapeamento direto** é utilizada:

i. Para cada slot no cache, liste os blocos que seriam mapeados para o slot.

ii. Quantos bits são necessários para um endereço? Liste os bits e indique qual parte do endereço seriam utilizados para (palavra, etiqueta, etc.).

- **i) R:** Veja figura acima, lateral direita do diagrama cache.
- **ii) R:** Como a memória principal (RAM) possui **32** palavras, para endereçar estas palavras, precisamos pelo menos 5 bits, pois 2^5 é igual a 32.
  O formato do endereço para mapeamento direto é mostrado na figura abaixo:

d) Mostre o estado do cache após a CPU ter requerido as seguintes palavras em sequência: 8, 10, 12, 23, 9, 19, 28, 13, 7 e 11.

↑ AQUI UMA FUNÇÃO DE MAPEAMENTO → mais direto e simples

Resposta:

cache = 4 *slots*

	tag			
0	01			← 4. 4
1	01			← 5. 9. 5
2	01			← 6. 14. 6
3	00			← 11. 3

Bloco = 2 palavras

RAM = $32 = 2^5$ palavras

0	1	0
2	3	1
4	5	2
6	7	3
8	9	4
10	11	5
12	13	6
14	15	7
16	17	8
18	19	9
20	21	10
22	23	11
24	25	12
26	27	13
28	29	14
30	31	15

Bloco = 2 palavras

Sequência de requisições :

palavra	binário	bloco	slot
8	01000	4	0
10	01010	5	1
12	01100	6	2
23	10111	11	3
9	01001	4	0
19	10011	9	1
28	11100	14	2
13	01101	6	2
7	00111	3	3
11	01011	5	1

45:27

e) Considerando a sequência de palavras dada no item "c", e um tempo de acesso de 10ns quando o dado for encontrado no cache e um tempo de acesso de 100ns se o cache não contiver o dado. Qual é o tempo de acesso médio para esta sequência? (prestar atenção no processo de transferência de dados)

Resposta:

- $t_{am} = \sum_{i=0}^{n} t_r^i \Big/ n$, n = número de requisições
- se (*cache hit*) $t_r^i = t_c$, ou
- se (*cache miss*) $t_r^i = t_c + t_m$
- onde t_{am} é *tempo médio de acesso*, t_r^i é o tempo gasto na referência da i-ésima palavra, t_c é *tempo de acesso ao cache* (10ns) e t_m é o *tempo de acesso a memória* principal (100ns).

$$t_{am} = (10 + 9 * 110) / 10 = 100ns$$

Hierarquia de Memória: Considere uma máquina com as seguintes características:
- Endereço virtual de 32 bits
- Endereço físico de 24 bits
- Página de 2048 bytes (2 KB)
- Um cache com mapeamento direto de 64 Kbytes e tamanho de bloco de 32 bytes.

a) Quais são as larguras dos seguintes campos de endereço? Em cada caso, indique se o campo corresponde a bits do endereço virtual ou endereço físico.

i) Número de páginas virtuais -- **21 bits do end. virtual**

ii) Número de frames de páginas -- **13 bits do endereço físico**

iii) Cache tag -- **8 bits do end. físico**

iv) Cache slot -- **11 bits do end. físico**

v) Cache word offset -- **5 bits do end. físico**

Considere um cache de **128 bytes** com blocos de **32 bytes**. A organização do cache é mostrado abaixo. Considere que os seguintes programas estarão executando nesta máquina:

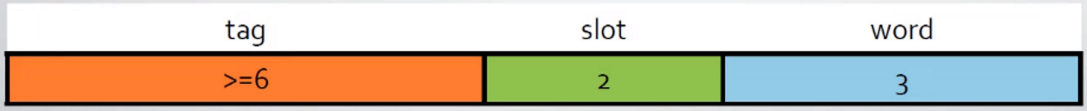
```
/* program A */
for (i=0; i<4; i++)
  for (j=0; j<2; j++)
     a[i+32*j]++;
```

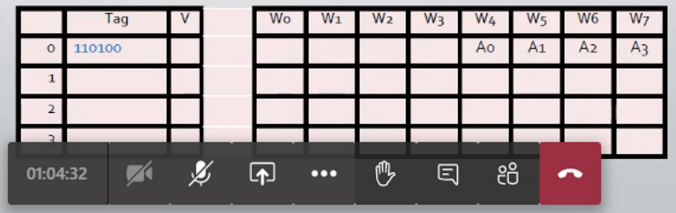
```
/* program B */
for (j=0; j<2; j++)
  for (i=0; i<4; i++)
     a[i+32*j]++;
```

O primeiro programa gera a seguinte sequência (***stream A***) de referências a memória: load a[0], store a[0], load a[32], store a[32], load a[1], store a[1], ... etc. O Segundo programa gera a seguinte sequência (***stream B***) de referências a memória: load a[0], store a[0], load a[1], store a[1], load a[2], store a[2], ... etc. (Os ***loads*** e ***stores*** são operações em palavras de 32 bits.) Assuma que os endereços de a[0] está localizado em **0x420**, e que todos os bits validos no cache são inicialmente zeros.

Considere um cache de **128 bytes** com blocos de **32 bytes**. A organização do cache é mostrado abaixo. Considere que os seguintes programas estarão executando nesta máquina:

Como o algoritmo precisa acessar os dados a partir do endereço 0x420 (**010000100000_b**). Assim, o endereço deve conter no mínimo 11 bits. Para um cache de 4 slots e blocos de 32 bytes, armazenando 8 palavras de 32 bits, temos:





Cache Metrics

Hit Ratio: $HR = \frac{hits}{hits + misses} = 1 - MR$

Miss Ratio: $MR = \frac{misses}{hits + misses} = 1 - HR$

Average Memory Access Time (AMAT):

$$AMAT = HitTime + MissRatio \times MissPenalty$$

- Goal of caching is to improve AMAT
- Formula can be applied recursively in multi-level hierarchies:

$$AMAT = HitTime_{L1} + MissRatio_{L1} \times AMAT_{L2} =$$
$$AMAT = HitTime_{L1} + MissRatio_{L1} \times (HitTime_{L2} + MissRatio_{L2} \times AMAT_{L3}) = \ldots$$

- Vamos considerar um computador com um cache **L1** e uma hierarquia de memória cache **L2**. Suponha que em **1000** referências de memória haja **80** erros em **L1** e **40** em **L2**.

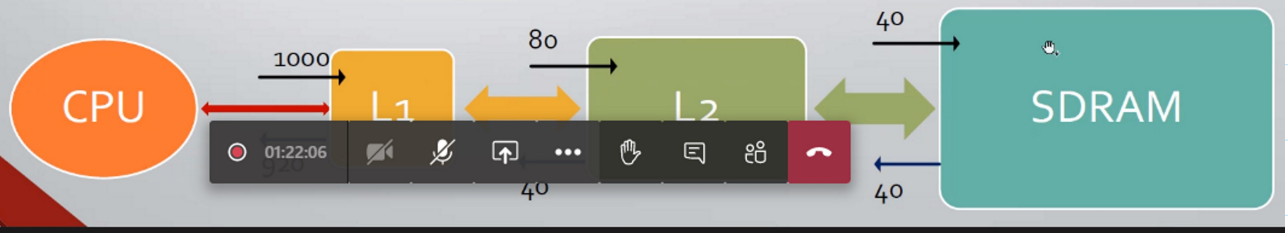
  a) Quais são as consequentes **taxas de falta de cache**?

  b) Suponha que o tempo de acerto **L1** = 1 ciclo de clock; Tempo de acerto **L2** = 20 ciclos de clock; **Miss Penalty** L2 = 200 ciclos de clock; Acessos a memória por instrução = **150%**. Qual é o tempo médio de acesso à memória (AMAT)?

- Vamos considerar um computador com um cache **L1** e uma hierarquia de memória cache **L2**. Suponha que em **1000** referências de memória haja **80** erros em **L1** e **40** em **L2**.

  a) Quais são as consequentes **taxas de falta de cache**?

MRL1 --> Taxa de Falhas de L1$ = 80/1000 = 0.08 = 8% <-- taxa global
MRL2 --> Taxa de Falhas de L2$= 40/1000 = 0.04 = 4% <-- taxa global
Taxa local em L2 --> 40/80 = 0.50 = 50%

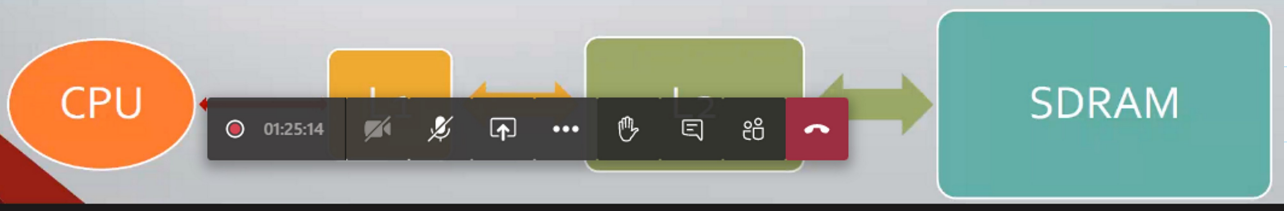


❑ Vamos considerar um computador com um cache **L1** e uma hierarquia de memória cache **L2**. Suponha que em **1000** referências de memória haja **80** erros em **L1** e **40** em **L2**.

b) Suponha que o tempo de acerto L1 = 1 ciclo de clock; Tempo de acerto L2 = 20 ciclos de clock; Miss Penalty L2 = 200 ciclos de clock; Acessos a memória por instrução = 150%. Qual é o tempo médio de acesso à memória (AMAT)?

HTL1 = 1 ciclo de clock
HTL2 = 20 ciclos de clock
MPL2 = 200 ciclos = tempo de acesso a RAM

Nota: Podemos calcular o tempo médio de acesso a memória de duas formas:
- Primeira (**modo 1**), calculamos **AMAT** = Hit TimeL1 + Miss RateL1 x Miss Penalty L1, considerando as taxas de falha locais para os caches.
- Segunda (**modo 2**), calculamos o **AMAT** considerando as taxas de acerto globais para os caches e os tempos de acesso somados.



Modo 1:

AMAT = Hit TimeL1 + Miss RateL1 x Miss Penalty L1

Miss Penalty L1 = Hit Time L2 + Miss RateL2 x Miss Penalty L2

AMAT = Hit TimeL1 + Miss RateL1 x (Hit Time L2 + Miss RateL2 x Miss Penalty L2)

AMAT = HTL1 + MRL1*(HTL2 + MRL2*(MPL2))

AMAT = 1 + 0.08*(20 + 0.50*200) = 10.6 ciclos de clock

Modo 2:

Alternativamente, pode-se calcular o AMAT empregando as taxas de acerto globais de cada cache:

HRL1 = Taxa de acerto de L1 = 0.92 = 92%

HRL2 = Taxa de acerto de L2 = 0.04 = 4% <-- 50% dos 8%

AMAT = 0.92*1 + 0.04*21 + 0.04*221 = 10.6 ciclos de clock