ARQUITETURA DE COMPUTADORES

2021-2022

Laboratório 5 – Memória Cache

Este laboratório destina-se a consolidar conhecimentos lecionados sobre memórias cache, utilizando o simulador Ripes (https://github.com/mortbopet/Ripes).

O trabalho deve ser realizado fora do horário de laboratório, destinando-se este à demonstração e avaliação do trabalho realizado. Ao longo do enunciado serão apresentadas várias tabelas e questões às quais os alunos deverão responder (de forma sucinta) num documento Word, que irão submeter através do Fénix, identificando de forma clara a questão à qual estão a dar a resposta. Exemplo:

Identifique a sua resposta no documento Word que vai entregar com o número **Q3.5**

No horário de laboratório serão fornecidos exercícios adicionais que devem ser realizados durante a aula. No final da aula de laboratório deverá submeter no Fénix um ficheiro zip com um documento Word com cópia das tabelas preenchidas e repostas às perguntas do guia, e todos os ficheiros Assembly correspondentes às diferentes versões dos programas concebidos.

Simulador de Cache

Para além de simular diferentes modelos de arquiteturas de processadores, o Ripes integra também um simulador de memórias cache, permitindo estudar o comportamento deste elemento de memória perante o padrão de endereços (de instruções ou de dados) requeridos pela aplicação.

Para garantir uma perfeita compreensão dos recursos e do modo de funcionamento deste simulador de cache, recomenda-se a leitura da respetiva documentação, disponível na seguinte página web: https://github.com/mortbopet/Ripes/blob/master/docs/cache_sim.md

Em particular, o simulador de cache permite a definição de um conjunto alargado de parâmetros de configuração da mesma, nomeadamente:

- O número de vias de associatividade
- O número de linhas (de cada via)
- O número de blocos (de 32-bits) presente em cada linha
- A política de substituição (em caches associativas)
- A política de escrita
- A política de alocação

Permite também a contagem do número de Hits e de Misses, bem como o cálculo do Hit-Rate resultante, à medida que o programa vai sendo executado pelo processador (ver Figura 1).

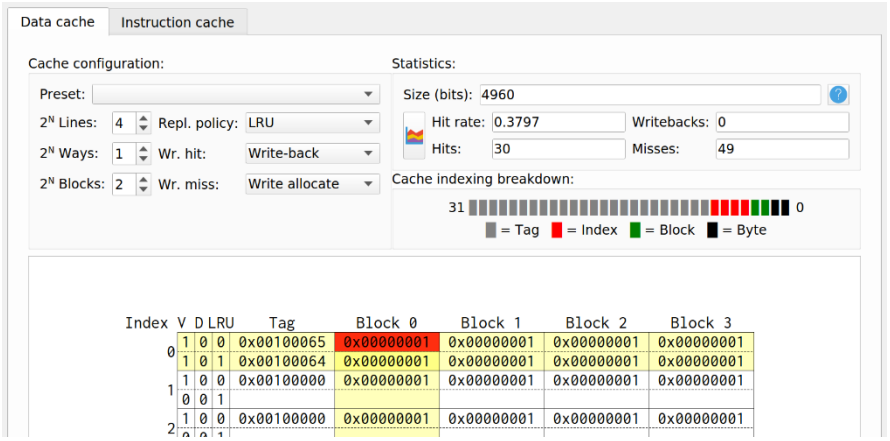


Figure 1 - Configuração e estatísticas de utilização da cache de dados.

Programa a Executar

No presente trabalho será estudada a influência da cache no tempo médio de acesso aos dados presentes na hierarquia de memória ligada a um determinado processador. Em particular, irá ser considerada a implementação de uma operação genérica de multiplicação de matrizes:

$$\mathbf{C} = \alpha\mathbf{AB} + \beta\mathbf{C}$$

Nesta equação, considera-se que **A**, **B** e **C** são matrizes quadradas com NxN elementos, enquanto α e β são coeficientes escalares. Todos os operandos são números inteiros de 32-bits. Para o efeito, o seguinte programa (em C) foi traduzido para linguagem Assembly RISC-V, facultado através do ficheiro L5.s (de notar que a matriz B está já transposta, simplificando a execução do código):

```
for (i=0, checksum=0; i<N; i++){
    for (j=0; j<N; j++){
        for (k=0, sum=0; k<N; k++)
            sum+= A[i][k]*B[j][k];
        aux = alpha*sum + beta*C[i][j];
        checksum += aux;
        C[i][j] = aux;
    }
}
```

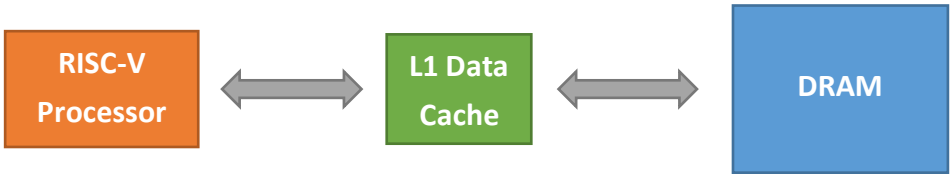
Exercício 1

1. Interprete o código fornecido no ficheiro L5.s e identifique-o com o programa em C e com a operação algébrica que se pretende realizar.
2. Considere que a execução do programa é realizada sobre um conjunto de matrizes quadradas (NxN) de dimensão fixa, com N=16. Cada elemento da matriz armazena um número inteiro de 32-bits.
3. Assuma que cada matriz é alocada em memória utilizando o padrão *row-major-order*, sendo o elemento A[0,0] armazenado no endereço `10000000h` e o elemento A[0,1] armazenado no endereço `10000004h`. Todas as matrizes são alocadas em espaços de memória adjacentes (i.e.: primeiro A, depois B, etc…).

4. Determine o valor dos endereços (em hexadecimal) dos seguintes elementos das matrizes utilizadas ao longo da execução do programa (copie esta tabela para o ficheiro de resposta):

Elemento	Endereço (Hexadecimal)
A[0,15]	
A[1,0]	
A[7,11]	
B[7,11]	
C[7,11]	

5. Considere agora a utilização de um sistema de memória constituído por uma cache de dados (L1) e uma memória primária (DRAM), com as seguintes características:



L1 DATA CACHE	
Nº de vias:	1
Número de linhas:	256
Dimensão da linha(ver nota):	16B (4 palavras)
Política de substituição:	*Least Recently Used*
Política de escrita:	*Write-Back*
Política de alocação:	*Write-Allocate*
Tempo de acesso (t_{Hit}):	1 ns

DRAM	
Dimensão total:	1 GByte
Tempo de acesso:	100 ns

Nota: o simulador define a dimensão da linha em função de palavras de 4B, a que denomina de bloco (note-se que a utilização do termo bloco é contrário à gíria mais usual, que utiliza bloco como sinónimo dos dados de uma linha).

6. Com base na especificação descrita na alínea anterior, e assumindo um barramento de endereços de 32-bits, determine o número de bits associados à etiqueta (*tag*), ao índice (*index*) e ao deslocamento (*offset*) de cada endereço invocado pelo processador. Justifique sucintamente.

Tag	Index	Offset

7. Determine o <u>valor</u> da etiqueta (*tag*), índice (*index*) e deslocamento (*offset*) dos endereços dos elementos das matrizes referidas na alínea 4:

Elemento	Endereço (Hexadecimal)	Tag	Index	Offset
A[0,15]				
A[1,0]				
A[7,11]				
B[7,11]				
C[7,11]				

Exercício 2

1. Configure o simulador Ripes de modo a que seja utilizado o modelo de processador de ciclo único. Para tal, deverá clicar sobre o símbolo do processador do canto superior esquerdo da janela e selecionar a opção *"Single Cycle Processor"*.
2. Abre a vista do simulador que lhe permita observar o funcionamento e a ocupação da memória cache. Execute alguns ciclos de relógio de modo a observar os critérios de preenchimento e de acesso referentes aos primeiros acessos à memória.
3. Execute o programa até ao fim, carregando na tecla >> do simulador
4. Observe e interprete o resultado, consultando as várias regiões da memória de dados (secção .data) de modo a identificar os endereços que guardam a matriz com o resultado da função.
5. Preencha a tabela seguinte:

Nº de Hits:		Nº total de acessos à memória:	
Nº de Misses:		Miss-Rate (total) [%]:	

6. Considerando o sistema de memória definido, determine:

Tempo médio de cada acesso aos dados [ns]:	
Tempo total de acesso aos dados [ms]:	

7. Compare os valores obtidos com aqueles que iria obter caso o sistema de memória não incorporasse a cache (i.e., todos os acessos seriam feitos diretamente à memória):

Tempo médio de cada acesso aos dados [ns]:	
Tempo total de acesso aos dados [ms]:	

Exercício 3

1. Varie agora os parâmetros da cache, aumentando o número de vias para 2, 4 e 8, mantendo a capacidade total da cache e a dimensão da linha. Justifique os resultados no miss-rate total em função do número de vias.

Nº de vias:	1	2	4	8
Número de linhas:	256			
Dimensão do bloco:	16B			
Miss-Rate (total) [%]:				

2. Considerando que o número de vias aumenta o tempo de acesso à cache, qual a configuração que escolheria?
3. Varie agora a dimensão do bloco para 8B, 32B e 64B, mantendo a capacidade total da cache e o número de vias escolhido no ponto anterior. Justifique os resultados no miss-rate total em função da dimensão do bloco.

Nº de vias escolhidas:				
Número de linhas:				
Dimensão do bloco:	8B	16B	32B	64B
Miss-Rate (total) [%]:				

4. Considere a melhor configuração de cache obtida em 3.3. Altere agora a dimensão das matrizes de 16x16 para 64x64 alterando a linha 12 do ficheiro de código (colocando N=64). Repita a alínea 1 e comente os resultados obtidos.

Nº de vias:	1	2	4	8
Número de linhas:				
Dimensão do bloco:				
Miss-Rate (total) [%]:				

Exercício 4

Este é um exercício surpresa que será divulgado pelo docente durante a aula.

Método de avaliação

O laboratório será avaliado numa escala de 0-8, com a seguinte ponderação:

- Exercício 1 – 0.5 pontos
- Exercício 2 – 0.5 pontos
- Exercício 3 – 1 ponto
- Exercício 4 – 3 pontos
- Mini-Teste (Moodle) – 3 pontos