ARQUITETURA DE COMPUTADORES

2020-2021

Laboratório 5 – Memória Cache

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

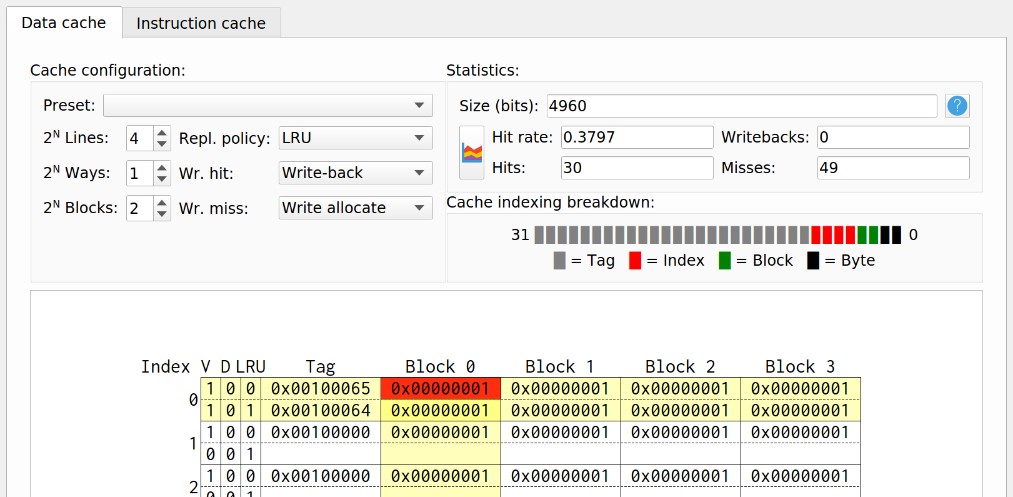
Para garantir uma perfeita compreensão dos recursos e do modo de funcionamento deste simulador de cache, recomenda-se a leitura da respetiva documentação, disponível na seguinte página web:

<https://github.com/mortbopet/Ripes/wiki/Cache-Simulation>

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: 𝐃=𝛼𝐀𝐁+𝛽𝐂

Nesta equação, considera-se que **A**, **B**, **C** e **D** são matrizes quadradas com NxN elementos, enquanto  e  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:

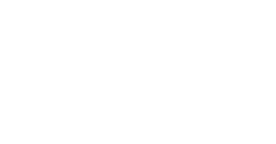
|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* This function implements the General Matrix Multiply (GMM) operation \*/  /\* for 32-bits integer numbers (IGEMM): D = alpha\*A\*B + beta\*C \*/  /\* \*/  /\* The global variables A, B, C and D are already defined and initialized \*/  /\* (NxN) square matrices. For the students' convenience, two extra (NxN) \*/  /\* auxiliary matrices were also defined (T1 and T2), to be used in the \*/  /\* exercise (if necessary). \*/  /\* \*/  /\* All these matrices are assumed to be adjacent and allocated in a \*/  /\* continuous memory map. \*/ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void igemm() { register int i,j,k; register int aux1;    /\* Preliminary part: ... \*/    /\* First part: T1 = A\*B \*/ for (i=0;i<N;i++) for (j=0;j<N;j++){ aux1=0;  for (k=0;k<N;k++)  T1[i][j] += A[i][k] \* B[k][j];  }    /\* Second part: D = alpha\*T1 + beta\*C \*/ for (j=0;j<N;j++) for (i=0;i<N;i++)  D[i][j] = alpha\*T1[i][j] + beta\*C[i][j]; } |

# 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 da função *igemm()* é realizada sobre um conjunto de matrizes quadradas (NxN) de dimensão fixa, com N=64. 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[32,24] | 10002060h |
| B[32,24] | 10006060h |
| C[32,24] | 1000A260h |
| D[32,24] | 1000E360h |
| T1[32,24] | 10012460h |
| T2[32,24] | 10016560h |

1. 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:

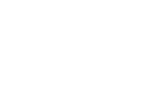


**RISC**

**-**

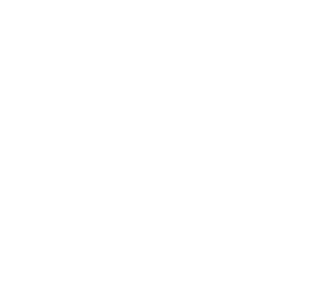
**V**

**Processor**



**L1 Data**

**Cache**



**DRAM**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **L1 DATA CACHE** | | | **Nº de vias de associatividade:** | 4 | | **Número de linhas:** | 64 | | **Dimensão do bloco (em palavras):** | 4 | | **Política de substituição:** | *Least Recently Used* | | **Política de escrita:** | *Write-Back* | | **Política de alocação:** | *Write-Allocate* | | **Tempo de acesso (tHit):** | 1 ns | |  |

|  |  |
| --- | --- |
| **DRAM** |  |
| **Dimensão total:** | 1 GByte |
| **Tempo de acesso:** | 100 ns |

1. 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:

Linha: 4 palavras \* 4 B = 16 B

Cache: 64 linhas \* 16B = 1kB

Offset = log216 = 4

Index = log216 = 4

Tag = 32 – 4 – 4 = 24

|  |  |  |
| --- | --- | --- |
| **Tag** | **Index** | **Offset** |
| 24 | 4 | 4 |

1. Determine o valor 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[32,24] | 10002060h | 100020h | 6h | 0h |
| B[32,24] | 10006060h | 100060h | 6h | 0h |
| C[32,24] | 1000A260h | 1000A2h | 6h | 0h |
| D[32,24] | 1000E360h | 1000E3h | 6h | 0h |
| T1[32,24] | 10012460h | 100124h | 6h | 0h |
| T2[32,24] | 10016560h | 100165h | 6h | 0h |

1. Em que medida o número de vias de associatividade da cache pode interferir com o desempenho desta função? Justifique. Pode utilizar, como exemplo, um troço de código do programa.

Identifique a sua resposta no documento Word que vai entregar com o número **Q1.8**

A picture containing text

Description automatically generatedNeste troco de código são necessárias 3 vias no mínimo para que seja possível guardar a informação das 3 matrizes, caso contrário, se por exemplo o numero de vias fosse apenas 2, o número de MISSs seria muito maior (passaria de 3 MISSs por cada 4 iterações // MISS RATE = 25% // do código para 3 MISSs na primeira iteração, e 2 nas seguintes, o que daria um total de 9 MISSs por cada 4 iterações //MISS RATE = 75% //), ou ainda pior, se fosse apenas 1 via, a taxa de MISS RATE seria de 100%. Como não é possível ter apenas 3 vias porque temos 64 linhas, são necessárias 4 vias de cache.

# Exercício 2

1. Configure o simulador [Ripes](https://github.com/mortbopet/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: | ﻿256130 | Nº total de acessos à memória: | 540 675 |
| Nº de Misses: | ﻿284545 | Miss-Rate (total) [%]: | 52,63 |

1. Considerando o sistema de memória definido, determine:

|  |  |
| --- | --- |
| Tempo médio de cada acesso [ns]: | 53,63 |
| Tempo total de acesso à memória [ms]: | 28, 996 400 25 |

1. 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 [ns]: | 100 |
| Tempo total de acesso à memória [ms]: | 54, 067 500 |

1. Comente os resultados obtidos:

Identifique a sua resposta no documento Word que vai entregar com o número **Q2.8**

A cache permite uma redução do tempo de acesso a memoria para quase metade do tempo que se perderia sem cache.

# Exercício 3

1. Interprete o código do ficheiro L5.s fornecido e identifique-o com a operação algébrica que se pretende realizar, através das duas partes que constituem a função *igemm()*.
2. Observe os dois ciclos implementados na segunda parte do programa. Até que ponto a eventual troca da ordem dos dois ciclos "for" influencia o resultado da operação?

Justifique.

Identifique a sua resposta no documento Word que vai entregar com o número **Q3.2**

Alterar os ciclos for ia reduzir os MISSs porque a matriz esta disposta na memoria linha a linha, então seria mais inteligente percorrer linha a linha em vez de coluna a coluna.

1. Em que medida a ordem destes dois ciclos influencia a taxa de sucesso nos acessos à cache? Justifique.

Identifique a sua resposta no documento Word que vai entregar com o número **Q3.3**

A taxa de sucesso ia aumentar de 0% para 75% porque a cache tem capacidade para 4 words, logo só de 4 em 4 acessos e que iria dar miss (as palavras na memoria estão seguidas linha a linha) (em vez de dar sempre miss pois as palavras que eram acedidas nunca estavam seguidas na memoria).

1. Qual a ordem mais favorável? Justifique.

Identifique a sua resposta no documento Word que vai entregar com o número **Q3.4**

A ordem contraria a que vem por base no ficheiro ‘L5.s’ porque a ordem que esta no ficheiro está a ler coluna a coluna e não linha a linha.

1. Para as duas alternativas possíveis, apresente o valor de:

|  |  |  |
| --- | --- | --- |
|  | Ordem i,j | Ordem j,i |
| Nº de acessos à memória (total): | 540 675 | 540 675 |
| Miss-Rate (total) [%]: | ﻿50,92 | 52,63 |
| Tempo médio de cada acesso [ns]: | 51,92 | 53,63 |
| Tempo total de acesso à memória [ms]: | 28, 071 846 | 28, 996 400 25 |

1. Esperava obter mais ganhos, face à versão original? Qual das duas partes do programa (primeira/segunda) representa a maioria dos misses observados? Justifique.

Identifique a sua resposta no documento Word que vai entregar com o número **Q3.6**

Não esperava obter mais ganhos pois a esmagadora maioria dos misses acontecem na primeira parte do programa.

# Exercício 4

1. Uma forma habitual de reduzir o número de *misses* na primeira parte do programa consiste em aplicar a técnica da transposição da matriz **B**. Adapte o código, preenchendo as linhas necessárias na zona identificada por "*Preliminary part*" e adaptando a invocação da leitura da segunda matriz na primeira parte do código. Pode utilizar a matriz auxiliar T2, para cálculos intermédios. Apresente o excerto do código alterado.

Identifique a sua resposta no documento Word que vai entregar com o número **Q4.1**

Text

Description automatically generated with medium confidenceA picture containing graphical user interface

Description automatically generated

1. Repita o procedimento de execução adotado nos exercícios anteriores, certificando-se que o resultado da operação está correto.
2. Apresente os novos valores de:

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de Hits: | ﻿473090 | Nº total de acessos à memória: | 548 867 |
| Nº de Misses: | ﻿﻿﻿75777 | Miss-Rate (total) [%]: | ﻿13, 81 |

1. Como justifica as diferenças observadas? Justifique.

Identifique a sua resposta no documento Word que vai entregar com o número **Q4.4**

Fazendo a transposta deixa de ser necessário andar a saltar de linha em linha na matriz B, e passa-se a andar de coluna em coluna, o que permite muito menos misses.

1. Calcule o ganho, em termos do tempo total de acesso à memória, face à melhor configuração do exercício 3 (alínea 3.5).

|  |  |
| --- | --- |
| Tempo médio de cada acesso [ns]: | 14, 81 |
| Tempo total de acesso à memória [ms]: | 8, 128 720 27 |
| Speedup: | 3, 453 415 183 2 |

# Exercício 5

1. Repita a execução do código Assembly a que chegou no exercício anterior, mas utilizando agora uma cache de mapeamento direto. Assuma que esta cache contempla o mesmo espaço de armazenamento de dados utilizado pela cache de 4 vias utilizada nos exercícios anteriores, mantendo o mesmo número de blocos em cada linha da cache.
2. Apresente os novos valores de:

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de Hits: | ﻿﻿﻿343 298 | Nº total de acessos à memória: | 548 867 |
| Nº de Misses: | ﻿﻿﻿205 569 | Miss-Rate (total) [%]: | 37, 45 |

|  |  |
| --- | --- |
| Tempo médio de cada acesso [ns]: | 38,45 |
| Tempo total de acesso à memória [ms]: | 21, 103 936 15 |
| Speedup: | 0,385 175 552 7 |

1. Explique (exemplo: através de excertos do código) porque razão se verificaram as alterações observadas. Justifique.

Text

Description automatically generated

Como da para ver no troço de código temos vários acessos a memoria seguidos, e estando as matrizes e como o índex e sempre igual, mas as TAGs são diferentes, sempre que tenta aceder a um novo dado há um miss. Isto leva a um significativo aumento de misses pois esta e a parte do código que se repete mais vezes, o que leva a um aumento do tempo de execução.

(Parte alterada esta com tab)

Graphical user interface

Description automatically generated with low confidence

6.3

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de Hits: | ﻿466946 | Nº total de acessos à memória: | 540 675 |
| Nº de Misses: | ﻿﻿﻿﻿73729 | Miss-Rate (total) [%]: | 13, 64 |

6.4 Deixou de ser necessário guardar os valores numa matriz auxiliar, então reduziu se o número de acessos a memoria.

6.5

|  |  |
| --- | --- |
| Tempo médio de cada acesso [ns]: | 14, 64 |
| Tempo total de acesso à memória [ms]: | 7, 915 482 |
| Speedup: | 1, 026 939 391 7 |