First Lab Assignment: System Modeling and Profiling

STUDENTS IDENTIFICATION:

Number:	Name:
103124	Gonçalo Bárias
103969	Miguel Costa
102624	Raquel Braunschweig

2 Exercise

Please justify all your answers with values from the experiments.

1. What is the cache capacity of the computer you used (please write the workstation name)?

Array Size (KB)	8	16	32	64	128	256
t2-t1 (s)	0,001937	0,003784	0,007535	0,017027	0,036395	0,089678
# accesses a[i]	819200	1638400	3276800	6553600	13107200	26214400
# mean access time (ns)	2,364502	2,309570	2,299500	2,598114	2,776718	3,420944

Usámos o computador lab-7-4. O tamanho da cache L1 é 32 KB, pois na tabela acima conseguimos observar que até ao tamanho 32 KB (inclusive), o tempo de acesso médio é, aproximadamente, o mesmo. Para 64 KB, o tempo aumenta um pouco, isto porque o tamanho da cache passou a ser inferior ao do array, o que levou a um aumento do miss rate.

Consider the data presented in Figure 1. Answer the following questions (2, 3, 4) about the machine used to generate that data.

2. What is the cache capacity?

Na figura podemos observar que até ao tamanho 64KB (inclusive), todos os arrays têm um tempo de leitura + escrita relativamente baixos. A partir do tamanho 128 KB o tempo aumenta bastante, o que nos leva a concluir que a cache L1 passou a ficar cheia para esses valores, levando a um aumento do miss rate. Assim, o tamanho da cache L1, para este computador, é 64KB.

3. What is the size of each cache block?

Dos arrays que têm um elevado tempo de leitura + escrita (≥ 128 KB), podemos observar que para um valor de 16 B para o stride, o tempo estabiliza. Assim, o tamanho de um bloco é 16B, pois quando o valor de stride para a ser igual ao tamanho de um bloco, o miss rate aproxima-se de 100 %.

4. What is the L1 cache miss penalty time?

Para um valor de stride de 16 B, o primeiro tamanho de array que não cabe totalmente na cache (128 KB) estabiliza-se com uma miss rate próxima de 100 % (alínea 2.3) e um tempo ≈ 960 ns, enquanto um de 64 KB com miss rate de quase 0 % (alínea 2.1) tem tempo ≈ 360 ns. Deste modo, 960 − 360 = 600 ns é o tempo de uma miss penalty.

3 Procedure

3.1.1 Modeling the L1 Data Cache

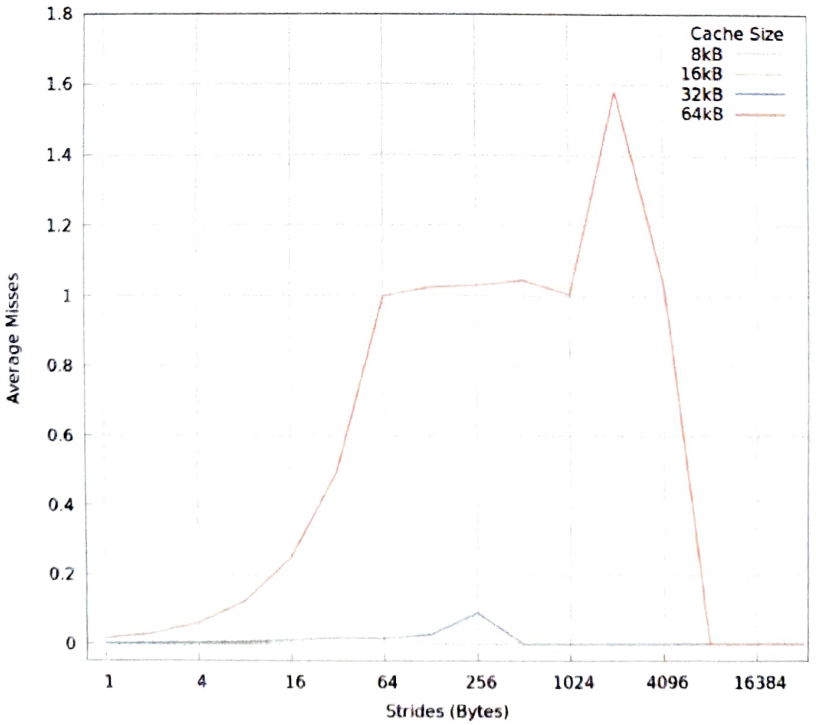
a) What are the processor events that will be analyzed during its execution? Explain their meaning.

Os eventos do processador que são analisados durante a execução do programa são os L1 data cache misses (PAPI_L1_DCM). Estes eventos são despoletados sempre que os dados não são encontrados na cache L1, sendo assim necessário aceder ao nível seguinte na hierarquia de memória, a cache L2, ou então a memória principal no caso da cache L2 não existir.

b) Plot the variation of the average number of misses (*Avg Misses*) with the `stride` size, for each considered dimension of the L1 data cache (8kB, 16kB, 32kB and 64kB).
Note that, you may fill these tables and graphics (as well as the following ones in this report) on your computer and submit the printed version.

Array Size	Stride	Avg Misses	Avg Cycl Time
8kBytes	1	0,000184	0,002491
	2	0,000121	0,002432
	4	0,000030	0,002405
	8	0,000023	0,002325
	16	0,000025	0,002387
	32	0,000035	0,002390
	64	0,000026	0,002310
	128	0,000011	0,002226
	256	0,000006	0,002175
	512	0,000004	0,002156
	1024	0,000003	0,002100
	2048	0,000004	0,002191
	4096	0,000003	0,002236
16kBytes	1	0,000192	0,002427
	2	0,000156	0,002425
	4	0,000155	0,002418
	8	0,000157	0,002338
	16	0,000159	0,002425
	32	0,000169	0,002424
	64	0,000172	0,002357
	128	0,000078	0,002321
	256	0,000043	0,002249
	512	0,000013	0,002162
	1024	0,000009	0,002146
	2048	0,000004	0,002251
	4096	0,000004	0,002349
	8192	0,000003	0,002246

Array Size	Stride	Avg Misses	Avg Cycl Time
32kBytes	1	0,002009	0,002397
	2	0,002760	0,002392
	4	0,004083	0,002382
	8	0,006760	0,002361
	16	0,012079	0,002331
	32	0,017888	0,002401
	64	0,016915	0,002413
	128	0,029685	0,002371
	256	0,093145	0,002357
	512	0,000166	0,002218
	1024	0,000071	0,002161
	2048	0,000030	0,002213
	4096	0,000013	0,002371
	8192	0,000002	0,002350
	16384	0,000002	0,002250
64kBytes	1	0,015655	0,002138
	2	0,031307	0,002044
	4	0,062666	0,002315
	8	0,125337	0,002422
	16	0,250581	0,002358
	32	0,501156	0,002443
	64	1,000671	0,001996
	128	1,025829	0,002027
	256	1,031328	0,002067
	512	1,046110	0,002068
	1024	1,006622	0,002006
	2048	1,579922	0,004640
	4096	1,036519	0,005486
	8192	0,000014	0,002390
	16384	0,000005	0,002376
	32768	0,000001	0,002251



c) By analyzing the obtained results:

- Determine the **size** of the L1 data cache. Justify your answer.

Dado que até 32 KB (inclusive) a average miss rate apresenta sempre valores baixos e a partir dos 64 KB, ela aumenta consideravelmente, pode-se concluir que o tamanho da cache L1 é de 32 KB. O motivo da average miss rate ter aumentado tanto, deve-se ao facto de ser impossível ter o array totalmente carregado na cache, pois este passa a exceder o tamanho da cache L1 e assim o seu limite é 32 KB.

- Determine the **block size** adopted in this cache. Justify your answer.

A miss rate vai aumentando até se atingir o valor de 64 B para o stride, a partir do qual se estagna. Assim, podemos concluir que o tamanho de um bloco da cache L1 é 64 B, pois ao se aceder ao array em múltiplos de 64, estamos a aceder a um novo bloco da cache, o que causaria a estagnação. Também podemos observar que, por exemplo, para um stride de 8 B estamos a aceder a aceder em múltiplos de 8 ao array e assim a cada 8 acessos, estaríamos a aceder a um novo bloco, algo que é corroborado pelo gráfico, já que a miss rate para um stride de 8 B é, aproximadamente, 1/8 da de um stride de 64 B.

- Characterize the **associativity set size** adopted in this cache. Justify your answer.

Para o primeiro array que não cabe totalmente na cache L1 (64KB) e um stride de 32KB, a miss rate é praticamente 0%. Uma vez que para este stride se acedem a dois blocos distintos da cache e tendo uma miss rate de quase 0%, então sabemos que a cache L1 é pelo menos 2-way associativa. No entanto, podemos observar o mesmo a acontecer para strides de 16KB e 8KB, logo garantimos cache 4-way e 8-way associativos, respetivamente. Dado que o mesmo já não acontece para 4KB, então temos que a associatividade da cache L1 é 8.

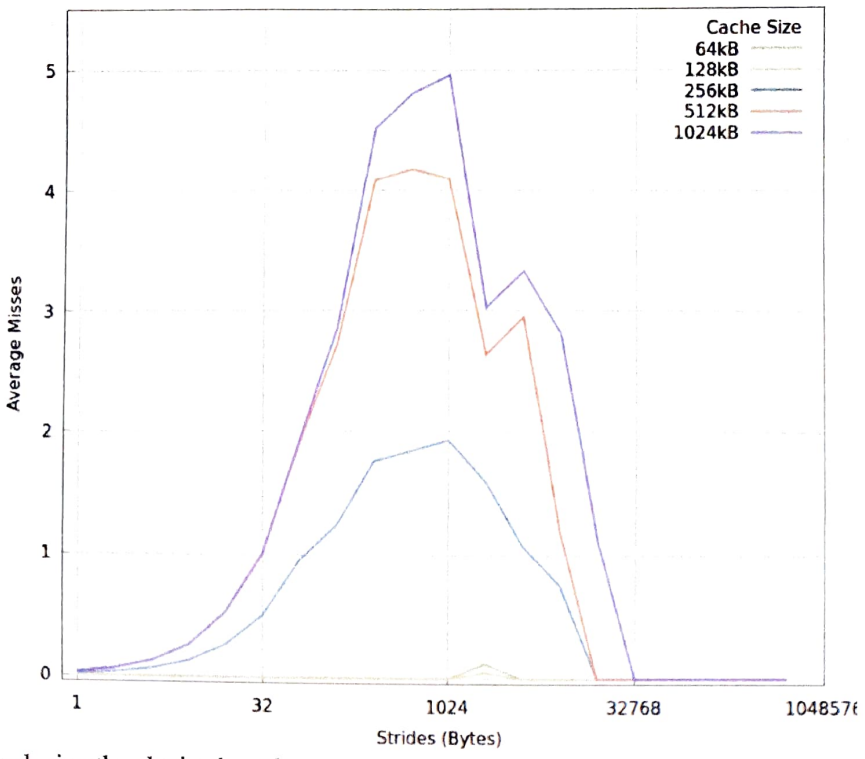
3.1.2 Modeling the L2 Cache

a) Describe and justify the changes introduced in this program.

As duas alterações efetuados foram:

1. O PAPI event que agora monitoriza a miss rate da cache L2, passando de PAPI_L1_DCM para PAPI_L2_DCM.
2. O valor de CACHE_MIN e CACHE_MAX passaram a ser 64KB e 1024KB, respetivamente. Uma vez que a cache L2 é normalmente maior que a L1, faz sentido realizar arrays de tamanho superior à da cache L1.

b) Plot the variation of the average number of misses (*Avg Misses*) with the `stride` size, for each considered dimension of the L2 cache.



c) By analyzing the obtained results:

- Determine the **size** of the L2 cache. Justify your answer.

No gráfico podemos observar um salto entre 128 KB e 256 KB, e depois um outro entre 256 KB e 512 KB. Dado que o segundo salto é consideravelmente maior, podemos concluir que até 256 KB (inclusive) o array ainda cabia na totalidade na cache L2, porém acima desse tamanho já excede a cache L2. Deste modo, concluímos que o tamanho da cache L2 é 256 KB.

- Determine the **block size** adopted in this cache. Justify your answer.

O tamanho de um bloco da cache L2 aparenta ser 64B. Dado que para valores de stride menores que 64 B, a miss rate vai crescendo lentamente e ao chegar aos 64 B atinge um pico, então concluímos que a average miss rate estagna devido a cada acesso conduzir a um novo bloco da cache.

- Characterize the **associativity set size** adopted in this cache. Justify your answer.

Para o array de 1MB e um stride de 512 KB, a miss rate é praticamente 0%. Uma vez que para este stride se acedem a dois blocos distintos da cache e tendo uma miss rate praticamente de 0, estão sabemos que a cache L2 é pelo menos 2-way associativa. O mesmo acontece para os valores de stride entre 32 KB e 256 KB, logo concluímos que a associatividade da cache L2 é 32.

$$\frac{2^{20}}{2^{15}} = 2^5 = 32$$

2^{15} strides menor que este fazem a miss rate aumentar para 100% (por exemplo, 2^{14} bytes)

3.2 Profiling and Optimizing Data Cache Accesses

3.2.1 Straightforward implementation

a) What is the total amount of memory that is required to accommodate each of these matrices?

N = 512 Número de elementos = $N^2 = 512^2 = 262144$

mul1 [N][N]
mul2 [N][N]
res [N][N]
} int16_t têm sizeof = 2 bytes

Tamanho da matriz = Número de elementos × × sizeof(int16_t) = = 262144 × 2 = 524288 B = 512 KB

R: Cada uma destas matrizes ocupa 512 KB.

b) Fill the following table with the obtained data.

Total number of L1 data cache misses	135,293818	$\times 10^6$
Total number of load / store instructions completed	536,871884	$\times 10^6$
Total number of clock cycles	725,153346	$\times 10^6$
Elapsed time	0,212776	seconds

c) Evaluate the resulting L1 data cache *Hit-Rate*:

$$\text{Hit-rate}_{mm1} = 1 - \frac{\text{L1 data cache misses}}{\text{L1 data cache inst.}} = 1 - \frac{135,293818}{536,871884} \approx 0,7480 = 74,80\%$$

3.2.2 First Optimization: Matrix transpose before multiplication [2]

a) Fill the following table with the obtained data.

Total number of L1 data cache misses	4,216280	$\times 10^6$
Total number of load / store instructions completed	536,871884	$\times 10^6$
Total number of clock cycles	625,869758	$\times 10^6$
Elapsed time	0,183644	seconds

b) Evaluate the resulting L1 data cache *Hit-Rate*:

$$\text{Hit-rate}_{mm2} = 1 - \frac{\text{L1 data cache misses}}{\text{L1 data cache inst.}} = 1 - \frac{4,216280}{536,871884} \approx 0,9921 = 99,21\%$$

c) Fill the following table with the obtained data.

Total number of L1 data cache misses	4,481656	$\times 10^6$
Total number of load / store instructions completed	537,396174	$\times 10^6$
Total number of clock cycles	626,015786	$\times 10^6$
Elapsed time	0,183687	seconds

Comment on the obtained results when including the matrix transposition in the execution time:

Hit-rate$_{mm2_transp.} \approx 99,17\%$
Após incluir a transposição da matriz, observamos que os valores aumentaram, mas não de forma significativa. Isto ocorre, pois a operação de transposição é $O(N^2)$, que é menos computacionalmente intensivo que multiplicar matrizes ($O(N^3)$).

d) Compare the obtained results with those that were obtained for the straightforward implementation, by calculating the difference of the resulting hit-rates (ΔHitRate) and the obtained speedups.

$\Delta\text{HitRate} = \text{HitRate}_{mm2} - \text{HitRate}_{mm1}$: 0,9917 - 0,7480 = 0,2437 = 24,37%
$\text{Speedup}(\#\text{Clocks}) = \#\text{Clocks}_{mm1}/\#\text{Clocks}_{mm2}$: 725 153346 / 626 015786 $\approx$ 1,16
$\text{Speedup}(\text{Time}) = \text{Time}_{mm1}/\text{Time}_{mm2}$: 0,212776 / 0,183687 $\approx$ 1,16
Comment: Uma vez que o Speedup é de 1,16, isto significa que a implementação mm2 acaba por ter um melhoramento pequeno em termos de ciclo de relógio, e o hit rate apresenta um aumento considerável de 24,37%. Uma vez que o Speedup não é muito considerável, talvez para sistemas com menos memória, não compensaria, pois a implementação necessita de memória adicional para uma matriz.

3.2.3 Second Optimization: Blocked (tiled) matrix multiply [2]

a) How many matrix elements can be accommodated in each cache line?

Dado que o tamanho de um bloco da cache L1 é 64B, esta é 8-way associativa e temos que sizeof(int16_t) = 2, então:

Conseguimos acomodar $\frac{64 \times 8}{2} = 256$ elementos num bloco da cache L1.

b) Fill the following table with the obtained data.

Total number of L1 data cache misses	3,858573	$\times 10^6$
Total number of load / store instructions completed	537,80223	$\times 10^6$
Total number of clock cycles	260,213142	$\times 10^6$
Elapsed time	0,076352	seconds

c) Evaluate the resulting L1 data cache *Hit-Rate*:

$$\text{Hit-rate}_{mm3} = 1 - \frac{\text{L1 data cache misses}}{\text{L1 data cache inst.}} = 1 - \frac{3,858573}{537,80223} \approx 0,992825 = 99,2825\%$$

d) Compare the obtained results with those that were obtained for the straightforward implementation, by calculating the difference of the resulting hit-rates (ΔHitRate) and the obtained speedup.

$\Delta\text{HitRate} = \text{HitRate}_{mm3} - \text{HitRate}_{mm1}$: 0,992825 - 0,7480 = 0,2448 = 24,48%
$\text{Speedup}(\#\text{Clocks}) = \#\text{Clocks}_{mm1}/\#\text{Clocks}_{mm3}$: 725 153346 / 260 213142 ≈ 2,77
Comment: Conseguimos um Speedup de, aproximadamente, 2,77, o que significa que dividir a matriz em vários sub-matrizes, de modo a que cada caiba num bloco da cache, ajuda a tirar proveito da localidade espacial. O aumento de quase 24.48% na hit rate, deve-se ao facto da possibilidade dos sub-matrizes caberem num bloco, o que ajuda a aceder à informação durante a computação.

e) Compare the obtained results with those that were obtained for the matrix transpose implementation by calculating the difference of the resulting hit-rates (ΔHitRate) and the obtained speedup. If the obtained speedup is positive, but the difference of the resulting hit-rates is negative, how do you explain the performance improvement? (Hint: study the hit-rates of the L2 cache for both implementations;)

$\Delta\text{HitRate} = \text{HitRate}_{\text{mm3}} - \text{HitRate}_{\text{mm2}}$: 0,992825 - 0,9917 = 0,001125
$\text{Speedup}(\#\text{Clocks}) = \#\text{Clocks}_{\text{mm2}}/\#\text{Clocks}_{\text{mm3}}$: 626015786 / 260213142 ≈ 2,41
Comment: O valor da L1 Δ Hit Rate era suposto ser negativo de acordo com o enunciado, porém após diversas tentativas, nunca foi possível obter esse valor, logo respondemos a esta pergunta assumindo-o como negativo. É possível observar um Speedup de 2,41 ao se comparem as implementações. Ao se analisarem os eventos de misses para a cache L2, tal como sugerido, podemos observar, aproximadamente, 8,95 misses para a transposição e 2,60 misses para a matriz por blocos. Isto faz compensar o valor mau da L1 Δ Hit Rate, pois a miss penalty para L1 misses na implementação da matriz por blocos acaba por ser menor e assim o Speedup vale a pena.

3.2.3 Comparing results against the CPU specifications

Now that you have characterized the cache on your lab computer, you are going to compare it against the manufacturer's specification. For this you can check the device's datasheet, or make use of the command `lscpu`. Comment the results.

Ao executarmos o comando "lscpu -C", conseguimos concluir que o tamanho da cache L1, o tamanho de um bloco L1 e a sua associatividade estão corretos. O tamanho da cache L2 e o tamanho de um bloco L2, também estão corretos, contudo obtemos uma associatividade de 32, quando o valor real é 4. Isto pode ter ocorrido devido a otimizações ao CPU de que desconhecemos.