

First Lab Assignment: System Modeling and Profiling

STUDENTS IDENTIFICATION:

Number:	Name:
102664	Pedro Sousa
102802	Fábio Rata
103392	Nuno Gonçalves

2 Exercise

Please justify all your answers with values from the experiments.

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

Array Size	8 kB	16 kB	32 kB	64 kB	128 kB	256 kB
t2-t1	0	1.789×10^{-3} s	3.622×10^{-3} s	4.334×10^{-2} s	5.84×10^{-2} s	1.231×10^1 s
# accesses a[i]	1638400	3276800	6553600	13107200	26214400	52428800
# mean access time	1.085 ns	1.089 ns	1.097 ns	3.855 ns	4.155 ns	4.426 ns

workstation: lab6P1. Nos cálculos anteriores, utilizámos apenas os valores referentes ao stride de 4096. #accesses a[i] = 2 * 100 * array.size. Analisando a tabela, podemos observar que existe um salto temporal entre os array-sizes 32 kB e 64 kB, o que indica que a hit-rate para tamanhos inferiores a 32 kB é maior do que a de tamanhos superiores. Assim, podemos concluir que a cache capacity é de 32 kB.

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

- What is the cache capacity?

64 kB. É possível observar na figura que existe uma subida acentuada no tempo do array de 64 kB para 128 kB, o que indica que a hit-rate deste último é menor. A queda da hit-rate deve-se a misses de capacidade, isto é, todos os blocos a que o programa está a tentar aceder não cabem na cache (neste caso, o array na sua totalidade), pelo que podemos concluir que a capacidade da cache da máquina utilizada é de 64 kB.

- What is the size of each cache block?

16 B. Observando a linha referente ao array de 128 kB, conseguimos perceber que existe uma subida acentuada no tempo entre os strides de 4 a 16 B, valor após o qual estabiliza. Sabemos assim que a partir do stride de 16 B existe um nº elevado de misses, que pode ser explicado por estarmos a fazer saltos maiores que o block size (ou igual), o que gera mais acessos a blocos diferentes e consequentemente mais misses.

- What is the L1 cache miss penalty time?

Em caso de miss, o tempo de acesso é dado por: $t_{miss} = t_{hit} + t_{miss_penalty}$

Considerando o stride de 32 B por exemplo, observa-se que, quando a hit-rate é alta, o tempo é aproximadamente 370 ns (t_{hit}), enquanto que quando é baixa rondam os 1000ns (t_{miss}). Podemos aproximar o miss penalty pela diferença entre estes 2 valores, ou seja, 630 ns.

(Assumindo que o tempo do gráfico se refere a um acesso individual e não ao total de cada stride)

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.

PAPI_L1_CDM = Contabiliza o nº de misses na cache L1 durante a execução do programa.

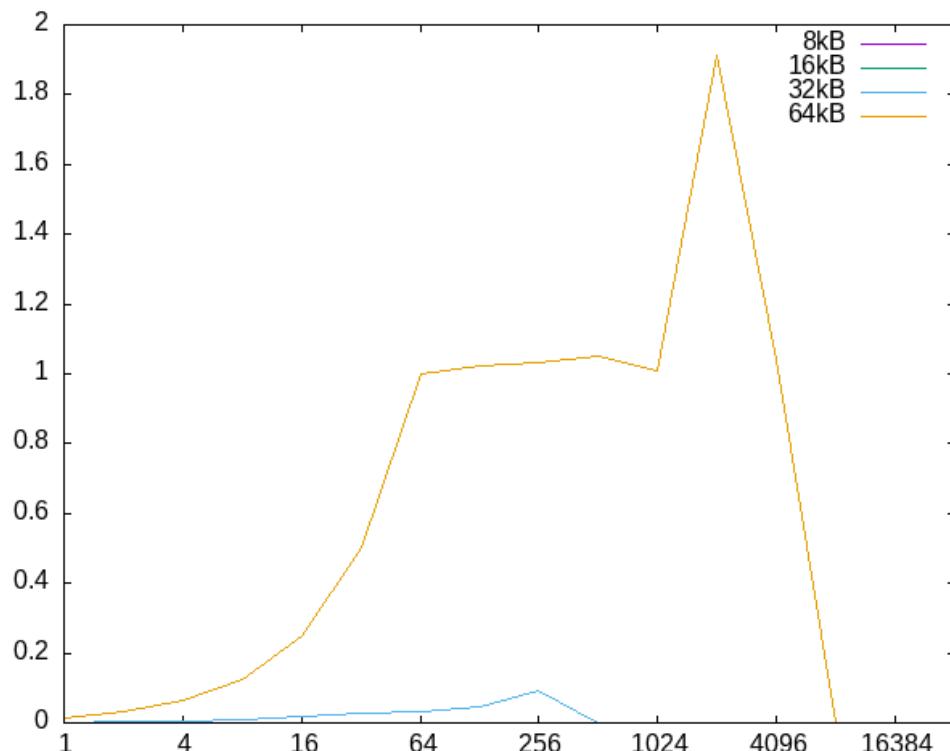
PAPI_TOT_CYC = Contabiliza o nº de ciclos que o CPU executou em user mode durante a execução do programa.

- 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
8 kBytes	1	0.000177	0.002311
	2	0.000101	0.002273
	4	0.000026	0.00229
	8	0.00002	0.002266
	16	0.000024	0.002262
	32	0.000023	0.002325
	64	0.000027	0.002463
	128	0.000013	0.002062
	256	0.000008	0.001998
	512	0.000002	0.001987
	1024	0.000002	0.001951
	2048	0.000002	0.00202
	4096	0.000003	0.002079
	8192	0.000015	0.002265
16 kBytes	2	0.00012	0.002267
	4	0.000103	0.002282
	8	0.000112	0.002258
	16	0.000113	0.002293
	32	0.000129	0.002349
	64	0.000114	0.002465
	128	0.000053	0.002182
	256	0.000028	0.00202
	512	0.000005	0.001955
	1024	0.000002	0.002021
	2048	0.000001	0.0021
	4096	0.000001	0.002137
	8192	0.000001	0.002027

Array Size	Stride	Avg Misses	Avg Cycl Time
32 kBytes	1	0.002188	0.002213
	2	0.003156	0.00217
	4	0.005263	0.002162
	8	0.009354	0.002117
	16	0.016862	0.002059
	32	0.025558	0.002095
	64	0.033255	0.002306
	128	0.04411	0.002158
	256	0.092897	0.002154
	512	0.000146	0.002015
	1024	0.000062	0.001948
	2048	0.000025	0.002001
	4096	0.000015	0.002135
	8192	0.000001	0.002116
	16384	0.000002	0.002033
64 kBytes	1	0.015639	0.001939
	2	0.031274	0.001852
	4	0.062612	0.002118
	8	0.125254	0.00218
	16	0.250499	0.002259
	32	0.500948	0.002059
	64	0.999961	0.001953
	128	1.023874	0.001831
	256	1.033455	0.001852
	512	1.050616	0.001878
	1024	1.008914	0.001813
	2048	1.912331	0.005308
	4096	1.040236	0.004958
	8192	0.000012	0.002137
	16384	0.000001	0.002157
	32768	0.000001	0.002043



c) By analyzing the obtained results:

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

32 kB, uma vez que para tamanhos de array entre 8 e 32 kB o número de avg misses se mantém estável em torno de 0, mas para um array de 64 kB este número escala rapidamente para valores perto de 1. Esta subida acentuada de valores deve-se a misses de capacidade, causados por o array de 64 kB não caber na sua totalidade na cache.

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

64 B, já que é possível observar que, para o array de 64 kB, que não cabe na cache, o número de avg misses sobe de 0 para 1 entre strides de 1 a 64, ficando estável a partir daí em torno de 1. Este comportamento deve-se ao número de misses aumentar com o stride até ao ponto em que o salto é maior que o block size (ou igual), gerando mais acessos a blocos diferentes e consequentemente mais misses, fazendo com que o valor de avg misses estabilize em torno de 1.

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

De forma a analisar a associatividade da cache L1, escolhemos o array de 64 kB (tamanho imediatamente acima da capacidade da cache, que deveria apresentar uma média de misses em torno de 1 utilizando mapeamento direto), pois este apresenta uma média de misses de 0 para o stride de 8192 B, o que indica que a cache tem algum tipo de associatividade.

Ao percorrermos o array com um stride de 8192 B, são realizados 8 acessos à memória. Simulámos estes acessos de forma a observar os índices e tags resultantes e pudemos concluir que para associatividades de tamanho 2 e 4 existiriam conflitos.

Analizando o caso de existirem 8 vias:

- cada bloco armazena 2^6 B, logo são necessários 6 bits para o offset (bits 0-5)
- existem $32\text{ kB} / (64\text{ B} \times 8) = 2^6$ índices, logo são necessários 6 bits para representar o índice (bits 6-11)
- a tag ocupa os restantes bits

Os acessos são os seguintes: (considerando que o endereçamento tem início em 0x0)

endereço ₁₀	TAG 16 15 14 13 12 11 bits	índice	
0	0 0 0 0 0 0	0	
8192	0 0 0 1 0 0	0	
16384	0 0 1 0 0 0	0	
24576	0 0 1 1 0 0	0	
32768	0 1 0 0 0 0	0	
40960	0 1 0 1 0 0	0	
49152	0 1 1 0 0 0	0	
57344	0 1 1 1 0 0	0	

Poderemos constatar que os 8 acessos foram realizados sem conflitos, visto que se accede sempre ao mesmo índice com tags diferentes. Ao repetir os acessos iremos obter 8 hits consecutivos na cache, pelo que podemos concluir que estamos perante uma associatividade de pelo menos 8 vias.

Para testar o caso de 16 vias de associatividade, precisamos de reduzir o salto feito entre acessos, de forma a serem realizados 16 acessos numa iteração pelo array. Escolhendo assim o stride de 4096 B, para o mesmo tamanho de array, estamos perante um caso semelhante ao anterior, no sentido em que ao realizar os saltos, apenas a tag é modificada (e tem sempre valores diferentes), mantendo o índice constante, logo acedendo sempre à mesma linha da cache. Assim sendo, a média de misses deveria ser próxima de 0, o que é inconsistente com o gráfico. Aplicando a mesma análise a um nº de vias superior (nas mesmas condições) iríamos chegar à mesma conclusão incorreta com o gráfico.

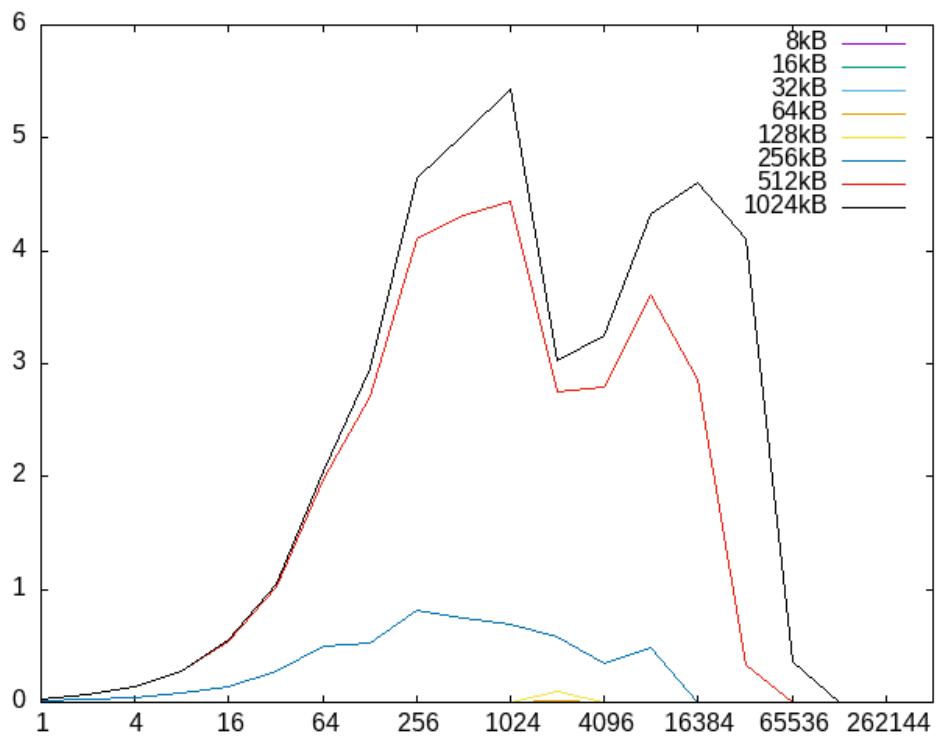
Conclui-se assim que a cache L1 tem 8 vias de associatividade.

3.1.2 Modeling the L2 Cache

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

Foi alterado o evento que queremos observar, no caso o número de misses da cache L2, pelo que mudámos na linha 37 o evento "PAPI_L1_DCM" para "PAPI_L2_DCM". Mudámos também o valor de CACHE_MAX para 1024x1024, de forma a utilizarmos arrays com tamanhos até 1024 kB, já que o tamanho original não permitia tirar conclusões relativamente à cache L2.

- 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.

256 kB, pois a partir do gráfico é possível observar que existe uma subida drástica no nº de avg misses quando passamos do array de 256kB para 512 kB, no qual toma valores acima de 1 muito rapidamente. Isto acontece devido a misses de capacidade, que surgem devido ao array de 512 kB não caber na cache, o que aumenta o número de misses.

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

256 B, já que é possível observar no gráfico que nas linhas de arrays com tamanhos superiores a 256kB, tamanhos maiores que a capacidade da cache, existe uma tendência para as avg misses estabilizarem. Esta tendência deve-se a estarem a ser feitos saltos maiores que o tamanho do bloco (ou iguais), o que gera mais acessos a blocos diferentes e consequentemente um maior nº de misses.

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

De forma a analisar a associatividade da cache L2, escolhemos o array de 512 KB (tamanho imediatamente acima da capacidade da cache, que deveria apresentar uma média de misses alta utilizando mapeamento direto), pois este apresenta uma média de misses de 0 para o stride 65536 B, o que indica que a cache tem algum tipo de associatividade.

Simulámos os acessos de forma a observar os índices e tags resultantes e pudemos concluir que para associatividades de tamanho 2 e 4 existem conflitos. Analisando o caso de existirem 8 vias (da mesma forma que na pergunta 3.1.1 C)), pudemos concluir que estamos perante uma associatividade de pelo menos 8 vias.

Aplicando a mesma análise final da pergunta 3.1.1 C), pudemos constatar que um nº de vias superiores levam-nos a conclusões incoerentes com o gráfico.

Conclui-se assim que a cache L2 tem 8 vias de associatividade.

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?

$$\begin{array}{c} 512 \times 512 \times 2 \text{ B} = 512 \text{ kB} \\ \downarrow \quad \downarrow \quad \downarrow \\ \text{dimensão} \quad \text{Short int size} \end{array}$$

- b) Fill the following table with the obtained data.

Total number of L1 data cache misses	$135.266.777 \times 10^6$
Total number of load / store instructions completed	$536.871.884 \times 10^6$
Total number of clock cycles	$577.390.246 \times 10^6$
Elapsed time	0.192464 seconds

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

$$\begin{aligned} \text{Accesses} &= 536.871.884 \times 10^6 \\ \# \text{misses} &= 135.266.777 \times 10^6 \end{aligned}$$

$$\text{Hit-rate} = 1 - \frac{\# \text{misses}}{\# \text{accesses}} \approx 74.80\%$$

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	4.215081×10^6
Total number of load / store instructions completed	536.871884×10^6
Total number of clock cycles	512.366962×10^6
Elapsed time	0.170789 seconds

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

$$\begin{aligned} \text{#accesses} &= 536.871884 \times 10^6 \\ \text{#misses} &= 4.215081 \times 10^6 \end{aligned}$$

$$\text{Hit-rate} = 1 - \frac{\text{#misses}}{\text{#accesses}} \approx 99.21\%$$

- c) Fill the following table with the obtained data.

Total number of L1 data cache misses	4.481916×10^6
Total number of load / store instructions completed	537.396174×10^6
Total number of clock cycles	519.671166×10^6
Elapsed time	0.173224 seconds

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

O tempo de execução aumentou cerca de 2.50ms, o que seria esperável, visto que foram realizadas mais instruções durante a análise. Naturalmente, ao serem feitos mais acessos à memória, o nº de misses aumenta. No entanto, este overhead causado pela transposição da matriz é menosprezável quando comparado com a performance do método straightforward.

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

$\Delta\text{HitRate} = \text{HitRate}_{\text{mm2}} - \text{HitRate}_{\text{mm1}}: 99.21 - 74.80 = 24.41\%$
$\text{Speedup}(\#Clocks) = \#\text{Clocks}_{\text{mm1}} / \#\text{Clocks}_{\text{mm2}}: 577.396174 / 512.366962 \approx 1.1269$
$\text{Speedup}(\text{Time}) = \text{Time}_{\text{mm1}} / \text{Time}_{\text{mm2}}: 0.192464 / 0.170789 \approx 1.1269$
Comment: (Para os cálculos e comentários foi utilizada a versão original do mm2.c) Ao utilizar a versão otimizada, consegue-se um aumento de 24.41% na Hit-rate, 12.69% em termos de speedup por clocks e tempo de execução. Este aumento de performance deve-se a, apesar de serem executadas mais instruções, os acessos à memória serem feitos de forma mais localizada, o que reduz significativamente o nº de cache misses.

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

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

$$\begin{aligned} \text{Cache line size} &= 64 \text{ B} \\ \text{matrix element size} &= 2 \text{ B} \quad R: \frac{64}{2} = 32 \text{ elem} \end{aligned}$$

- b) Fill the following table with the obtained data.

Total number of L1 data cache misses	3.584017×10^6
Total number of load / store instructions completed	537.80223×10^6
Total number of clock cycles	213.107224×10^6
Elapsed time	0.071036 seconds

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

$$\begin{aligned} \# \text{accesses} &= 537.80223 \times 10^6 \\ \# \text{misses} &= 3.584017 \times 10^6 \quad \text{Hit-rate} = 1 - \frac{\# \text{misses}}{\# \text{accesses}} \approx 99.33\% \end{aligned}$$

- d) Compare the obtained results with those that were obtained for the straightforward implementation, by calculating the difference of the resulting hit-rates ($\Delta \text{HitRate}$) and the obtained speedup.

$\Delta \text{HitRate} = \text{HitRate}_{\text{mm3}} - \text{HitRate}_{\text{mm1}}: 99.33 - 74.80 = 24.53\%$
Speedup(#Clocks) = #Clocks _{mm1} / #Clocks _{mm3} : $537.390246 / 213.107224 \approx 2.5094$
Comment: <i>Obtemos um aumento de 24.53% na hit-rate e um speedup de 170.94% em termos de clock cycles. Ao multiplicar a matriz por grupos, as entradas a serem acessadas têm uma chance maior de estar no mesmo bloco de cache, aumentando assim a hit-rate.</i>

- e) Compare the obtained results with those that were obtained for the matrix transpose implementation by calculating the difference of the resulting hit-rates ($\Delta \text{HitRate}$) 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;)

$$\Delta \text{HitRate} = \text{HitRate}_{\text{mm3}} - \text{HitRate}_{\text{mm2}}: 99.33 - 99.21 = 0.12\%.$$

$$\text{Speedup}(\# \text{Clocks}) = \# \text{Clocks}_{\text{mm2}} / \# \text{Clocks}_{\text{mm3}}: 512.366962 / 213.107224 = 2.4043$$

Comment: Obtivemos um Speedup de 140.42%, apesar de não haver uma diferença significativa nas hit-rates da L1, o que provavelmente indica que existe uma diferença nas miss-penalties. Se a diferença de hit-rates for negativa mas o speedup for positivo, podemos explicar o aumento de performance através da análise da cache L2. Analisando os data cache misses desta cache em ambos os programas, podemos constatar que estes são aproximadamente 9.5x maiores no mm2 (o que está relacionado com a ordem de acessos à memória em cada algoritmo). Assim, podemos concluir que o speedup cache L1 é justificado pela diferença nas hit-rates da cache L2, apesar de na cache L1 serem semelhantes.

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.

	real	previsto
L1 d	32 kB	32 kB
L2	256 kB	256 kB

Podemos concluir que a nossa análise está de acordo com os dados reais.