Second Lab Assignment: System Modeling and Profiling

STUDENTS IDENTIFICATION:

Number:	Name:
106340	Francisco Uva
107482	Pedro Pais
107502	Francisco Ferro Pereira

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	8 KiB	16 KiB	32 KiB	64 KiB	128 KiB	256 KiB
t2-t1	0.001704	0.003556	0.007215	0.033128	0.112915	0.222398
# accesses a[i]	409600	1228800	2867200	6144000	12697600	25804800
# mean access time	4.16 ns	2.89 ns	2.52 ns	5.40 ns	8.89 ns	8.62 ns

WorkStation: Lab6P1. Os resultados obtidos na tabela em cima são referentes ao valor de stride de 4096B para os diferentes array sizes. Ao analisar a tabela, observamos a existência de um salto temporal significativo (cerca de 4.6 vezes mais lento) entre os arrays de 32KiB e 64KiB. Este salto temporal é também refletido no aumento da mean access time nos arrays de tamanho superior a 32KiB. Isto deve-se ao facto da hit rate ser inferior para esses tamanhos, visto que o array não cabe na sua totalidade na cache, levando a capacity misses. Assim, podemos concluir que o tamanho da cache é 32KiB, visto que para arrays de capacidade superior a performance da cache é consideravelmente pior.

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 1, podemos observar um aumento significativo no tempo de acesso ao array ao passar de 64 KiB para 128 KiB, o que sugere uma diminuição na hit-rate para o tamanho maior. Esta queda está relacionada a misses de capacidade, ou seja, o conjunto de blocos que o programa tenta aceder excede a capacidade da cache (neste caso, completamente). Portanto, podemos concluir que a capacidade da cache é de 64 KiB.

3. What is the size of each cache block?

16 B, ao observarmos a linha referente ao array de 128 KiB, podemos notar uma subida acentuada no tempo entre os strides de 4 a 16B, após o qual o tempo estabiliza. Assim, sabemos que a partir do stride de 16B ocorre um número elevado de misses, o que pode ser explicado pelo facto de estarmos a fazer saltos maiores ou iguais ao tamanho do bloco, resultando em mais acessos a blocos diferentes e, consequentemente, mais misses.

4. What is the L1 cache miss penalty time?

Em caso de penalização por cache miss, o tempo de acesso é dado por: Tmiss_penalty = Tmiss - Thit. Considerando, por exemplo, o stride de 32B, podemos observar que, quando a hit-rate é alta, o tempo de acesso é aproximadamente 375 ns (Thit), enquanto que, quando é baixa, ronda os 1000 ns (Tmiss). Podemos aproximar a cache miss penalty time pela diferença entre estes dois valores, ou seja, 625 ns.

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 número de misses na cache L1 durante a execução do programa.

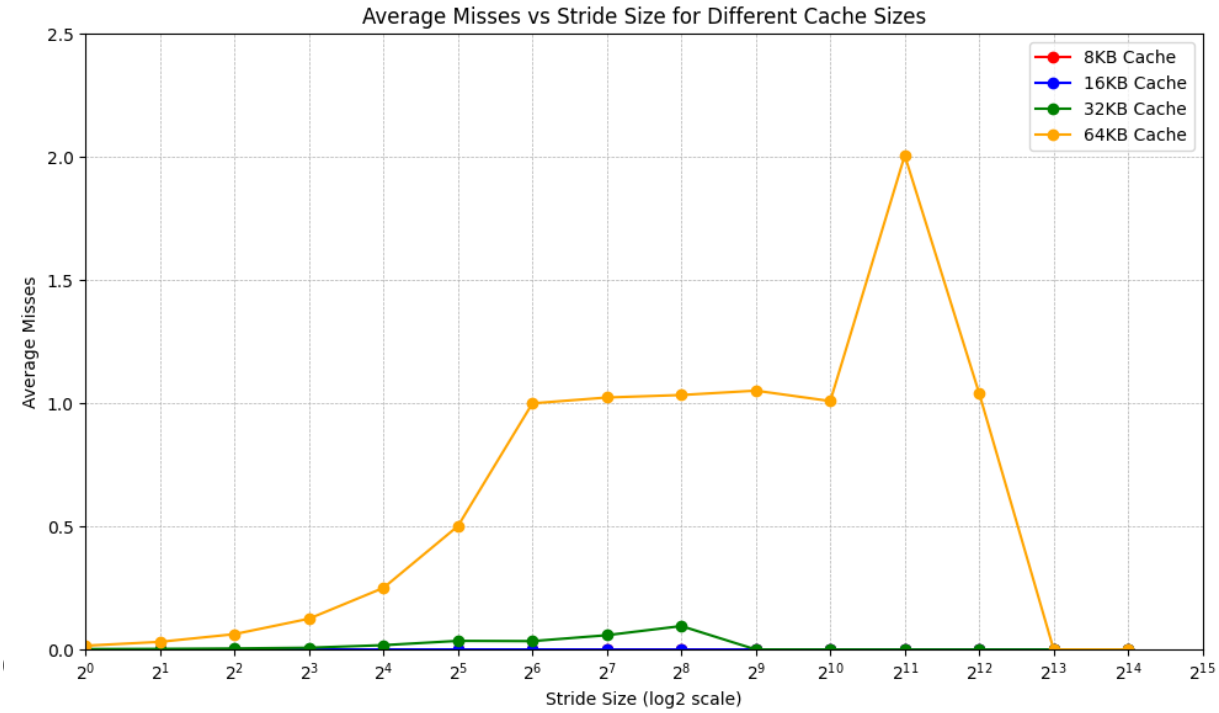
PAPI_TOT_CYC = Contabiliza o número de ciclos que o CPU executou em user mode durante a execução do programa.

PAPI_real_usec = Fornece o tempo real da execução do programa em microssegundos.

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	0.000205	0.002686
	2	0.000137	0.002894
	4	0.000071	0.002890
	8	0.000054	0.002818
	16	0.000049	0.002691
	32	0.000048	0.002202
	64	0.000045	0.002184
	128	0.000027	0.002057
	256	0.000014	0.002003
	512	0.000013	0.001986
	1024	0.000011	0.001942
	2048	0.000010	0.002018
	4096	0.000012	0.002081
16kBytes	1	0.000221	0.002243
	2	0.000234	0.002238
	4	0.000272	0.002243
	8	0.000218	0.002178
	16	0.000251	0.002293
	32	0.000292	0.002225
	64	0.000290	0.002203
	128	0.000117	0.002184
	256	0.000071	0.002060
	512	0.000038	0.001999
	1024	0.000020	0.001987
	2048	0.000009	0.002083
	4096	0.000008	0.002173
	8192	0.000013	0.002091

Array Size	Stride	Avg Misses	Avg Cycl Time
32kBytes	1	0.001990	0.002224
	2	0.002708	0.002219
	4	0.004412	0.002226
	8	0.007334	0.002201
	16	0.017803	0.002154
	32	0.035276	0.002234
	64	0.034220	0.002238
	128	0.058331	0.002208
	256	0.095471	0.002205
	512	0.000165	0.002060
	1024	0.000078	0.001997
	2048	0.000038	0.002057
	4096	0.000023	0.002194
	8192	0.000007	0.002173
	16384	0.000004	0.002078
64kBytes	1	0.015657	0.001980
	2	0.031300	0.001893
	4	0.062598	0.002168
	8	0.125309	0.002246
	16	0.250499	0.002203
	32	0.501451	0.002264
	64	0.999883	0.001848
	128	1.023650	0.001880
	256	1.033749	0.001917
	512	1.051524	0.001925
	1024	1.009163	0.001876
	2048	2.006765	0.005751
	4096	1.040095	0.005120
	8192	0.000023	0.002237
	16384	0.000009	0.002216
	32768	0.000004	0.002117



c) By analyzing the obtained results:

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

32 KB, visto que para tamanhos entre 8KB e 32KB o número de average misses mantém-se praticamente estável em torno de 0. No entanto, para um array de 64KB o número de misses dispara para valores em torno de 1, o que pode ser explicado pela ocorrência de misses de capacidade, causados pelo facto do array de 64KB não caber na sua totalidade na cache.

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

64B, visto que no array de 64KB (que não cabe totalmente na cache), o número de average misses sobe de forma acentuada de 0 até 1 entre os strides 1B e 64B, respetivamente, ficando estável em torno de 1. Isto deve-se ao facto do número de misses aumentar com o stride até ao ponto em que o stride é maior que o tamanho do bloco (ou igual), gerando mais acessos a diferentes blocos e consequentemente mais misses, o que resulta no valor de average misses estabilizar em volta de 1.

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

Visto que a cache tem 32KB, os arrays de tamanho superior ao mesmo, (neste caso 64KB) deveriam apresentar uma average miss de aproximadamente 1 (assumindo que a cache é diretamente mapeada).
No entanto, para stride de 8192B verificamos que o average miss é novamente 0. Isto indica que existe uma certa associatividade nesta cache, visto que à medida que aumentamos a stride, os acessos começam a se dispersar, sendo mapeados para sets diferentes (e ocorrendo menos conflict misses).

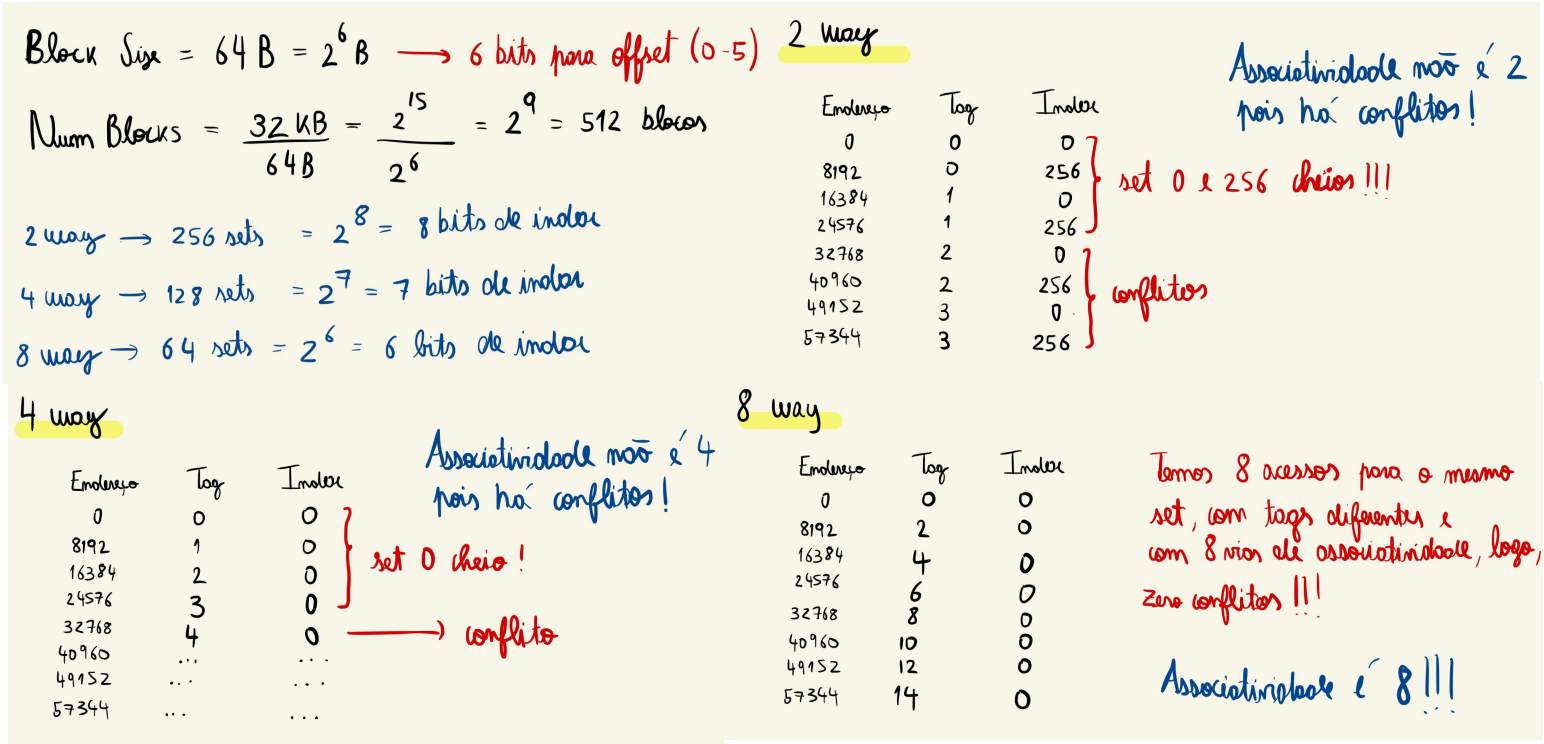
Percorrendo o array de 64KB com stride 8192B temos:

nºAcessos = (Cache Size) / (Stride) = 2^16 / 2^13 = 2^3 = 8 acessos

Esses acessos são feitos nos seguintes endereços:

0, 8192, 16384, 24576, 32768, 40960, 49152, 57344

Para determinar qual é a correta associatividade desta cache, temos de simular os 8 acessos de forma a que não haja conflitos.

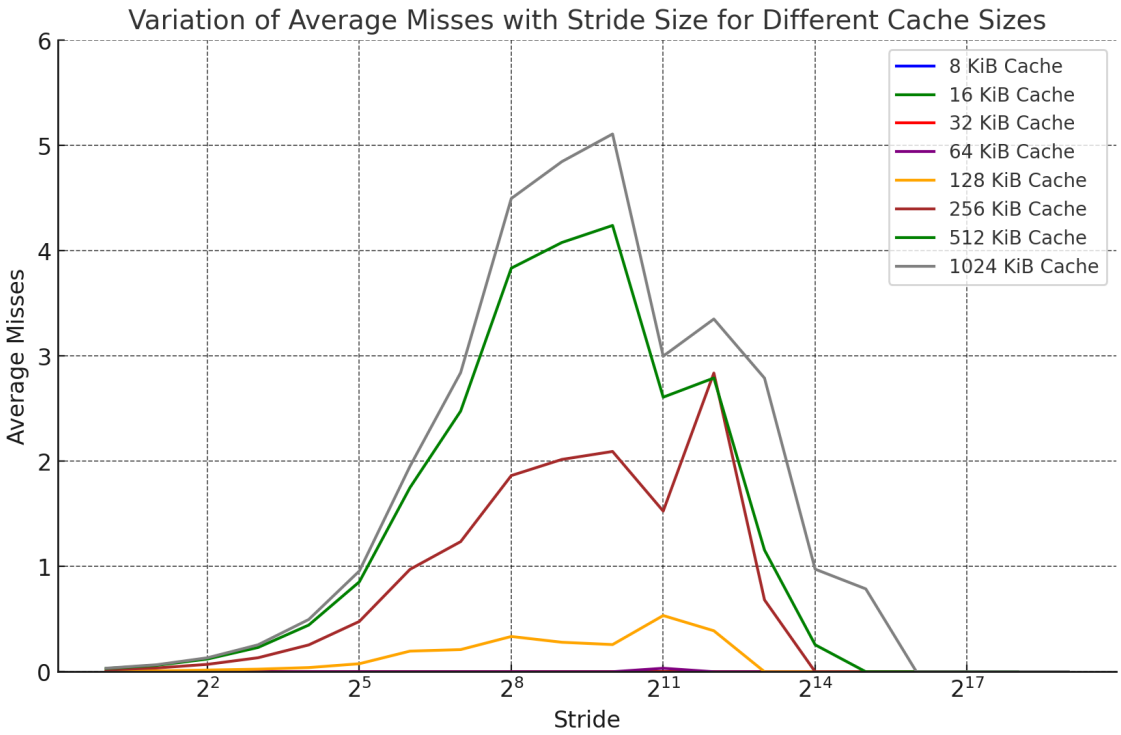


3.1.2 Modeling the L2 Cache

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

Alterámos o evento "PAPI_L1_DCM" para o evento que queremos observar "PAPI_L2_DCM", assim como o tamanho de MAX_CACHE para 1024KB, visto que o tamanho original (64KB) não permitia monitorizar todos os eventos sem perdas de informação, uma vez que o tamanho real da cache L2 é muito superior ao limite originalmente imposto. Isto impedia-nos de tirar conclusões precisas 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.

128KiB, pois segundo o gráfico podemos observar uma subida significativa no número de average misses do array de 128 KB para 256 KB, o qual toma valores muito acima de 1 muito rapidamente, isto acontece devido a misses de capacidade dados ao facto do array de 256 KB não caber na sua totalidade na cache o que aumenta significativamente o número de misses.

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

Com base no gráfico, podemos observar que entre as strides de 32B e 64B há um aumento acentuado no número de misses. Normalmente, quando o tamanho do stride começa a exceder o tamanho do bloco da cache, o número de misses começa a aumentar significativamente.

No gráfico, se observarmos o comportamento da linha de 256 KiB (vermelho), vemos que a curva começa a aumentar significativamente entre 2^5 e 2^6 (32 e 64), atingindo o valor 1 por volta da stride de 64B. Isto sugere que o tamanho do bloco da cache seja esse valor.

Além disso, o block size é geralmente consistente nos diferentes níveis de cache.
Visto que o tamanho do bloco da cache L1 é 64B, é bastante provável que o tamanho do bloco na cache L2 seja igual. Este valor é também coerente com tamanhos típicos de bloco em arquiteturas de cache modernas.

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

Para calcular a associatividade da cache L2, utilizámos o array de 256KB (tamanho logo acima ao da capacidade da cache que deveria apresentar uma média de misses alta, assumindo que a cache é diretamente mapeada). No entanto, olhando para o gráfico, podemos observar que para uma stride de 16384B (16KB) a média de misses é 0. Isto sugere a existência de algum tipo de associatividade nesta cache.

Para descobrir a associatividade correta desta cache simulámos os acessos com diferentes associatividades e de forma a que não ocorressem conflitos, da mesma forma como em 3.1.1 c).

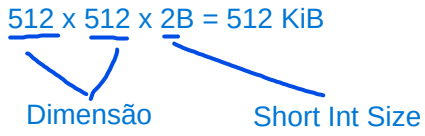
Testámos a associatividade 2 e 4 e registámos conflitos. No entanto, para associatividade 8 não obtivemos nenhum conflito pelo que podemos concluir que a associatividade é 8.

No entanto, segundo as informações disponibilizadas pelo fabricante a associatividade da cache L2 na arquitetura do processador (Coffee Lake) em questão é 4. Esta discrepância entre a associatividade inferida e a real pode ser relativa a inúmeros fatores tais como: Dados registados pelos eventos são imprecisos, contenção de recursos devido a múltiplas conexões via ssh à mesma máquina, system noise, entre outros.

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?



b) Fill the following table with the obtained data.

Total number of L1 data cache misses	134.992496 $\times 10^6$
Total number of load / store instructions completed	536.871899 $\times 10^6$
Total number of clock cycles	592.759071 $\times 10^6$
Elapsed time	0.197587 seconds

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

#accesses = 536.871899 x10^6
#misses = 134.992496 x10^6

Hit-rate = 1-(#misses/#accesses) = 1 - (134.992496/536.871899) = 1 - 0.251144 = 74.8556%

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	4.218805 $\times 10^6$
Total number of load / store instructions completed	536.871896 $\times 10^6$
Total number of clock cycles	535.897638 $\times 10^6$
Elapsed time	0.178632 seconds

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

#accesses = 536.871896 x10^6
#misses = 4.218805 x10^6

Hit-rate = 1-(#misses/#accesses) = 1 - (4.218805/536.871896) = 1 - 0.007858122266 = 99.21418777%

c) Fill the following table with the obtained data.

Total number of L1 data cache misses	4.486649 $\times 10^6$
Total number of load / store instructions completed	537.65833 $\times 10^6$
Total number of clock cycles	522.788699 $\times 10^6$
Elapsed time	0.174263 seconds

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

O tempo de execução aumentou em cerca de 1.106 ms, o que era esperado, uma vez que foram realizadas mais instruções durante a análise. Naturalmente, com o aumento do número de acessos à memória, o número de misses também aumentou. No entanto, este overhead causado pela transposição da matriz é insignificante quando comparado com o desempenho 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 ($\Delta\mathrm{HitRate}$) and the obtained speedups.

$\Delta\mathrm{HitRate} = \mathrm{HitRate}_{\mathrm{mm2}} - \mathrm{HitRate}_{\mathrm{mm1}}$: 99.21418777 - 74.8556 = 24.35858777%
$\mathrm{Speedup}(\#\mathrm{Clocks}) = \#\mathrm{Clocks}_{\mathrm{mm1}}/\#\mathrm{Clocks}_{\mathrm{mm2}}$: 592.759071 / 535.897638 = 1.106
$\mathrm{Speedup}(\mathrm{Time}) = \mathrm{Time}_{\mathrm{mm1}}/\mathrm{Time}_{\mathrm{mm2}}$: 0.197587 / 0.178632 = 1.106
Comment: Ao utilizar a versão otimizada, observa-se um aumento de 24,3585% na Hit-rate, 11,06% em termos de speedup em ciclos e tempo de execução. Este ganho de desempenho ocorre porque, embora sejam executadas mais instruções, os acessos à memória são mais localizados, resultando numa redução significativa do número 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?

Cache Line: 64B
Matrix Element Size: 2B

R: 64/2 = 32 elementos

b) Fill the following table with the obtained data.

Total number of L1 data cache misses	4.212627 $\times 10^6$
Total number of load / store instructions completed	536.871896 $\times 10^6$
Total number of clock cycles	209.364786 $\times 10^6$
Elapsed time	0.069788 seconds

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

#accesses = 536.871896 x10^6
#misses = 4.212627 x10^6
Hit-rate = 1-(#misses/#accesses) = 1 - (4.212627 /536.871896) = 1 - 0.007846614865 = 99.21533851%

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

$\Delta\mathrm{HitRate} = \mathrm{HitRate}_{\mathrm{mm3}} - \mathrm{HitRate}_{\mathrm{mm1}}$:	99.21533851 - 74.8556 = 24.35973851
$\mathrm{Speedup}(\#\mathrm{Clocks}) = \#\mathrm{Clocks}_{\mathrm{mm1}}/\#\mathrm{Clocks}_{\mathrm{mm3}}$:	592.759071 / 209.364786 = 2.831226217

Comment:

Conseguimos um aumento de 24,36% na taxa de acertos e uma aceleração de 283,12% em termos de ciclos de clock. Ao multiplicar a matriz por grupos, as entradas a serem acedidas têm uma maior probabilidade de estar no mesmo bloco de cache, aumentando assim a taxa de acertos.

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

$\Delta\text{HitRate} = \text{HitRate}_{\text{mm3}} - \text{HitRate}_{\text{mm2}}$:	99.21533851 - 99.21418777 = 0.00115074%
$\text{Speedup}(\#\text{Clocks}) = \#\text{Clocks}_{\text{mm2}}/\#\text{Clocks}_{\text{mm3}}$:	535.897638 / 209.364786 = 2.559635974

Comment:

Obtivemos uma aceleração de 255,96%, apesar de não haver uma diferença significativa nas taxas de acerto da cache L1, o que provavelmente indica uma diferença nas penalidades de falha de cache. Se a diferença nas taxas de acerto fosse negativa, mas o ganho de desempenho fosse positivo, poderíamos explicar o aumento de desempenho através da análise da cache L2. Analisando as falhas de cache de dados nos dois programas, constatamos que essas falhas são aproximadamente 3,5 vezes maiores no MM2 (o que está relacionado com o padrão de acesso à memória em cada algoritmo). Assim, podemos concluir que o aumento de desempenho é justificado pela diferença nas taxas de acerto da cache L2, apesar das taxas da 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	32 KiB	32 KiB	Podemos concluir que a nossa previsão da cache L1 está de acordo com as especificações listadas pelo fabricante.
L2	256 KiB	128 KiB	Podemos concluir que a nossa previsão da cache L2 não está de acordo com as especificações listadas pelo fabricante.

Esta imprecisão pode resultar de inúmeros fatores tais como:

- Dados imprecisos devido a overhead ao monotorizar eventos no PAPI
- System noise
- Cache pollution
- Contenção de recursos devido a múltiplas conexões ssh à mesma máquina