First Lab Assignment: System Modeling and Profiling

STUDENTS IDENTIFICATION:

Number:	Name:
99552	Rodrigo Dias
102848	Diogo Rodrigues
103156	Francisco Nael

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	16 KiB	32 KiB	64 KiB	128 KiB	256 KiB	512 KiB
t2-t1 (ms)	3,7	7,2	14,9	36,3	87,8	192,1
# accesses a[i]	$3,3\times10^6$	$6,5\times10^6$	$1,3\times10^7$	$2,6\times10^7$	$5,2\times10^7$	$1,04\times10^8$
# mean access time (ms)	1,12	1,12	1,14	1,39	1,68	1,83

We use the LAB6PS computer
The cache capacity is 64 KiB, since we saw the first jump in mean acess time from 64 KiB to 128 KiB

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?

The cache capacity is 64 KiB, since we saw the first jump in mean acess time from 64 KiB to 128 KiB.

3. What is the size of each cache block?

consider only the arrays bigger than the cache capacity, from stride 16, there are only misses, therefore the size block is 16

4. What is the L1 cache miss penalty time?

For arrays smaller than the cache size, for the minimum stride we can infer a 100% hit rate with the time of 360 ns (hit time), and for arrays bigger than the cache size, and the stride bigger than the block size, we can aproximate a miss rate of 1000 with the time of 1000 ns. Therefore using the formula $t_{access} = t_{hit_{L1}} + P_{miss_{L1}} \times t_{penalty_{L1}} \Rightarrow t_{penalty_{L1}} = 1000 - 360 = 640$ ns

3 Procedure

3.1.1 Modeling the L1 Data Cache

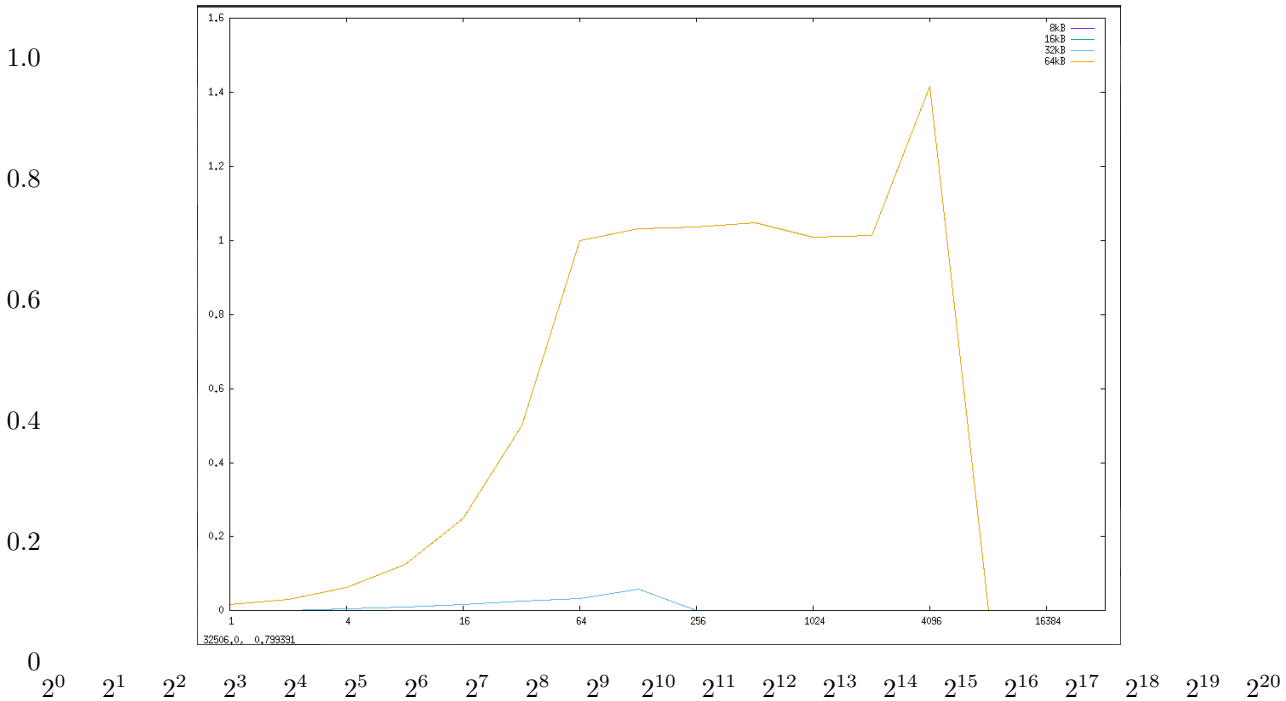
a) What are the processor events that will be analyzed during its execution? Explain their meaning.

The events analysed are data cache L1 misses. This event counts how many L1 data cache misses occur during the program execution. A data cache L1 miss occurs whenever you try to access an adress in the cache, and it's not stored in the cache. After that the computer loads an entire block from the memory to the cache.

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 (10^{-4})	Avg Cycl Time (10^{-3})
8kBytes	1	1,97	2,13
	2	1,39	2,24
	4	0,94	2,30
	8	1,34	2,21
	16	0,71	2,19
	32	0,70	2,19
	64	1,07	2,24
	128	0,38	2,10
	256	0,29	2,09
	512	0,19	2,06
	1024	0,12	2,01
	2048	0,09	2,07
	4096	0,25	2,17
16kBytes	1	2,40	2,30
	2	1,67	2,24
	4	1,95	2,22
	8	2,38	2,17
	16	2,10	2,24
	32	2,51	2,23
	64	2,24	2,22
	128	0,97	2,20
	256	0,85	2,16
	512	0,25	2,09
	1024	0,10	1,99
	2048	0,05	2,08
	4096	0,09	2,18
	8192	0,07	2,12

Array Size	Stride	Avg Misses (10^{-4})	Avg Cycl Time (10^{-3})
32kBytes	1	17,8	2,23
	2	22,6	2,25
	4	31,9	2,26
	8	49,0	2,15
	16	85,7	2,12
	32	137,1	2,11
	64	159,9	2,24
	128	125,6	2,21
	256	249,3	2,24
	512	794,6	2,17
	1024	0,78	2,00
	2048	0,24	2,05
	4096	0,16	2,20
	8192	0,17	2,19
	16384	0,11	2,11
64kBytes	1	157	2,00
	2	313	1,91
	4	627	2,14
	8	1253	2,22
	16	2506	2,28
	32	5008	2,05
	64	10 000	1,90
	128	10 299	1,97
	256	10 457	2,03
	512	10 568	2,14
	1024	10 710	1,90
	2048	10 016	2,21
	4096	10 154	5,05
	8192	3,27	2,20
	16384	0,03	2,17
	32768	0,04	2,11



c) By analyzing the obtained results:

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

The L1 data cache has a size of 32 KiB because from 64 KiB there is a non zero amount of misses which cannot be justified by the initial loading of the cache by due to the use of consective repetitions

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

The block size is 64, because for vectors bigger than the cache size and strides bigger or equals than 64 (block size) we get a 100 % miss rate.

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

num reads = $\frac{\text{vector size}}{\text{stride}}$

For an array size of 64KB, and a stride of 8, there are 8 elements read in the vector, and there are no misses.

Due to the fact that the array size is bigger than the cache size, all array elements do not fit together in the cache. For a large stride all address point to the same index.

Since we can observe that for less or equal than 8 elements the miss count goes to zero, all elements fit in the same set.

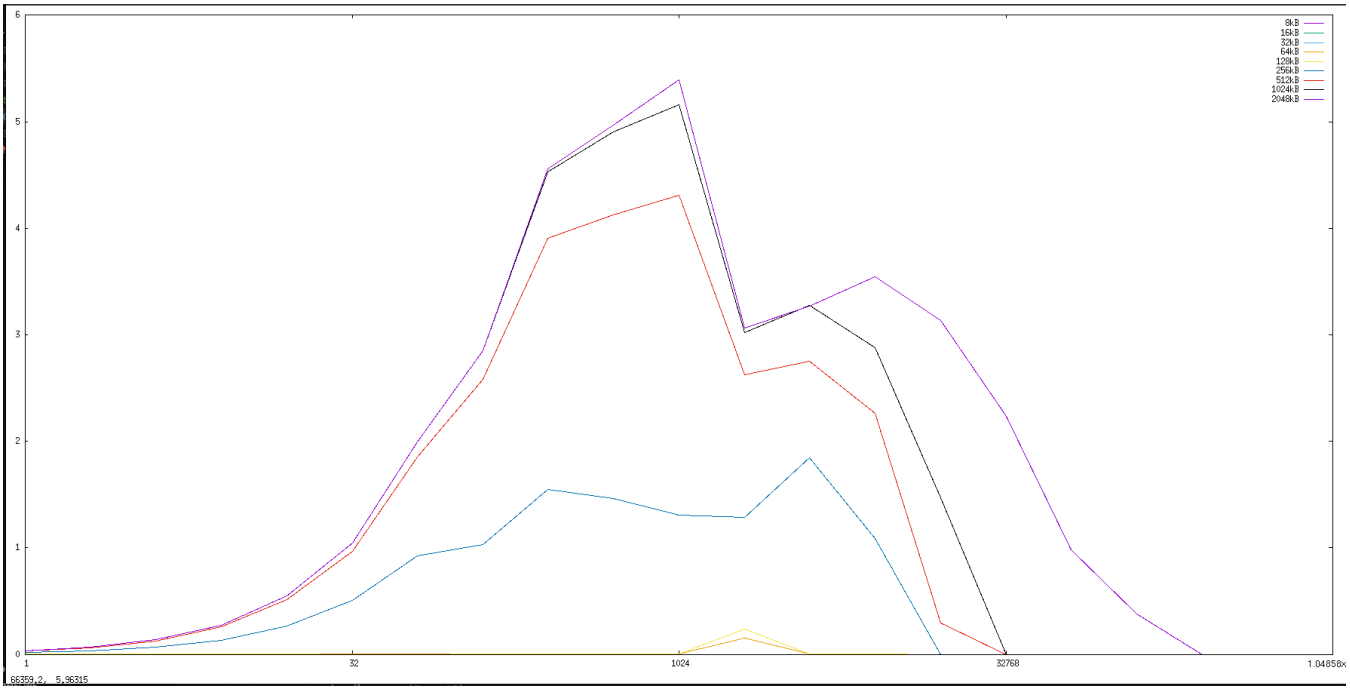
Hence, the L1 cache is 8-way set associative

3.1.2 Modeling the L2 Cache (We starting using the LAB7P6)

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

The PAPI event was changed from PAPI_L1_DCM to PAPI_L2_DCM. To analyze the L2 data cache misses

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.

The cache capacity is 256 KB, since we saw the biggest jump in misses count, with arrays bigger than the cache size, due to the appearance of cache capacity misses.

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

The block size is 256, because for vectors bigger than the cache size and strides bigger than 256 (block size) we get the maximum L2 misses count

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

num reads = $\frac{\text{vector size}}{\text{stride}}$

For an array size of 1024KB, and a stride of 32768, there are 32 elements read in the vector, and there are no misses.

Due to the fact that the array size is bigger than the cache size, all array elements do not fit together in the cache. For a large stride all adress point to the same index.

Since we can observe that for less or equal than 32 elements the miss count goes to zero, all elements fit in the same set.

Hence, the L2-cache is 32-way set associative

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?

$512 \times 512 \times 16 = 2^{22}$ bits $= 2^{19}$ bytes

512×512: number of elements; 16: number of bits per element in the matrix

b) Fill the following table with the obtained data.

Total number of L1 data cache misses	$135{,}0 \times 10^6$
Total number of load / store instructions completed	$536{,}9 \times 10^6$
Total number of clock cycles	$741{,}2 \times 10^6$
Elapsed time	0,217 seconds

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

Hit Rate = 1 − Miss Rate = $1 - \frac{135 \times 10^6}{536{,}9 \times 10^6} = 74{,}86\%$.

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	4,279 $\times 10^6$
Total number of load / store instructions completed	536,9 $\times 10^6$
Total number of clock cycles	660,7 $\times 10^6$
Elapsed time	0,194 seconds

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

$$\text{Hit rate} = 1 - \text{Miss Rate} = 1 - \frac{\text{Misses}}{\text{Accesses}} = 1 - \frac{4,279 \times 10^6}{536,9 \times 10^6} = 99,21\%$$

c) Fill the following table with the obtained data.

Total number of L1 data cache misses	4,486 $\times 10^6$
Total number of load / store instructions completed	537,9 $\times 10^6$
Total number of clock cycles	673,6 $\times 10^6$
Elapsed time	0,198 seconds

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

The transposition resulted in mere 0.004 seconds of added time, which, compared to the marginal improvements of this method (better hit rate, less time), is of very small importance.

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 speedups.

$\Delta\text{HitRate} = \text{HitRate}_{\text{mm2}} - \text{HitRate}_{\text{mm1}}$: 99,21 − 74,86 = 24,35%.
$\text{Speedup}(\#\text{Clocks}) = \#\text{Clocks}_{\text{mm1}}/\#\text{Clocks}_{\text{mm2}}$: 741,2 / 660,7 = 1,122
$\text{Speedup}(\text{Time}) = \text{Time}_{\text{mm1}}/\text{Time}_{\text{mm2}}$: 0,217 / 0,194 = 1,119
Comment: The implementation of transposition resulted in a major improvement to the hit-rate (24,36%), and a speedup of , which demonstrates the theoretical principles of locality, in this case, the benefits of sequential memory accesses.

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

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

Since the block size is 64 B, and each short integers is two bytes, we can fit 32 elements in a single cache line.

b) Fill the following table with the obtained data.

Total number of L1 data cache misses	2,524	$\times 10^6$
Total number of load / store instructions completed	537,8	$\times 10^6$
Total number of clock cycles	287,9	$\times 10^6$
Elapsed time	0,0827	seconds

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

Hit Rate = 1 - Miss Rate = $1 - \frac{2,524 \times 10^6}{537,8 \times 10^6} = 99,53\%$.

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.

$\Delta\mathrm{HitRate} = \mathrm{HitRate}_{\mathrm{mm3}} - \mathrm{HitRate}_{\mathrm{mm1}}$: 99,53 - 74,86 = 24,67%.
$\mathrm{Speedup}(\#\mathrm{Clocks}) = \#\mathrm{Clocks}_{\mathrm{mm1}}/\#\mathrm{Clocks}_{\mathrm{mm3}}$: 741,2 / 281,9 = 2,629
Comment: The submatrix approach led to a major improvement in hit-rate (24,67%), and a big speedup (2,629). By accessing memory by blocks whose size equal the cache block size, the sequentiality is maximized, since every submatrix entry is guaranteed to be in cache from the moment the first submatrix entry is accessed (and loaded to cache).

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;)

$\Delta\text{HitRate} = \text{HitRate}_{\text{mm3}} - \text{HitRate}_{\text{mm2}}$: 99,53 − 99,21 = 0,32
$\text{Speedup}(\#\text{Clocks}) = \#\text{Clocks}_{\text{mm2}}/\#\text{Clocks}_{\text{mm3}}$: 660,7 / 287,9 = 2,344
Comment: Althought the difference between mm2 and mm3 in L1 DCM is almost zero, the difference in L2 data cache misses (DCM) is significatively lower in mm3. which results in a lower number of access in L3 cache and memory (slower access than L2). And therefore results in an increase of performance.

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.

According to our results, we got the L1 and L2 capacities, L1 block sizes and L1 associative set sizes right. Unfortunately we didn't got the L2 associative set size right neither the L2 block size.