

PRACTICA 5

MEMORIA CACHE

Sumario

Practica 5-A.....	3
lscpu.....	3
make info.....	3
CPU-World.....	4
line.Ofast.png.....	5
line.O0.png.....	5
line.O1.png.....	6
line.O2.png.....	6
Memoria.....	7
Practica 5-B.....	7
cache.fast.png.....	7
cache.O0.png.....	8
cache.O1.png.....	8
cache.O2.png.....	9
Memoria.....	9

Practica 5-A

lscpu





```
4419 ± lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:       Little Endian
CPU(s):               8
On-line CPU(s) list:  0-7
Hilo(s) de procesamiento por núcleo:2
Núcleo(s) por «socket»:4
Socket(s):            1
Modo(s) NUMA:         1
ID de fabricante:     GenuineIntel
Familia de CPU:       6
Modelo:               158
Model name:           Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Revisión:             9
CPU MHz:              899.951
CPU max MHz:          3800,0000
CPU min MHz:          800,0000
BogoMIPS:             5616.00
Virtualización:       VT-x
Caché L1d:            32K
Caché L1i:            32K
Caché L2:             256K
Caché L3:             6144K
NUMA node0 CPU(s):   0-7
Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aper
_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdr
ep bmi2 erms invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec xge
```

make info

```
ep bmi2 erms invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec xge
antonio-Lenovo 35 ~/Documentos/git/PRACTICAS-EC/Pract
4420 ± make info
line size = 64B
cache size = 32K/32K/256K/6144K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
```

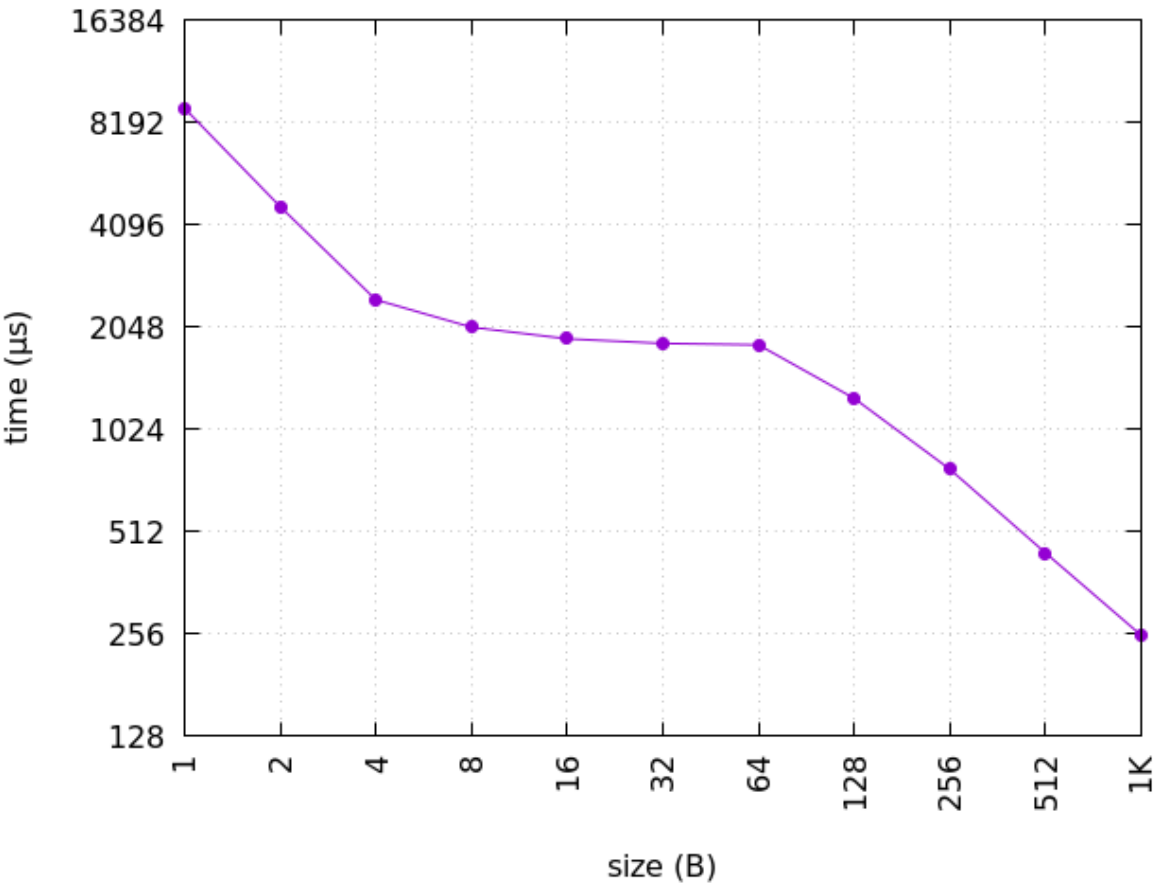
CPU-World

General information

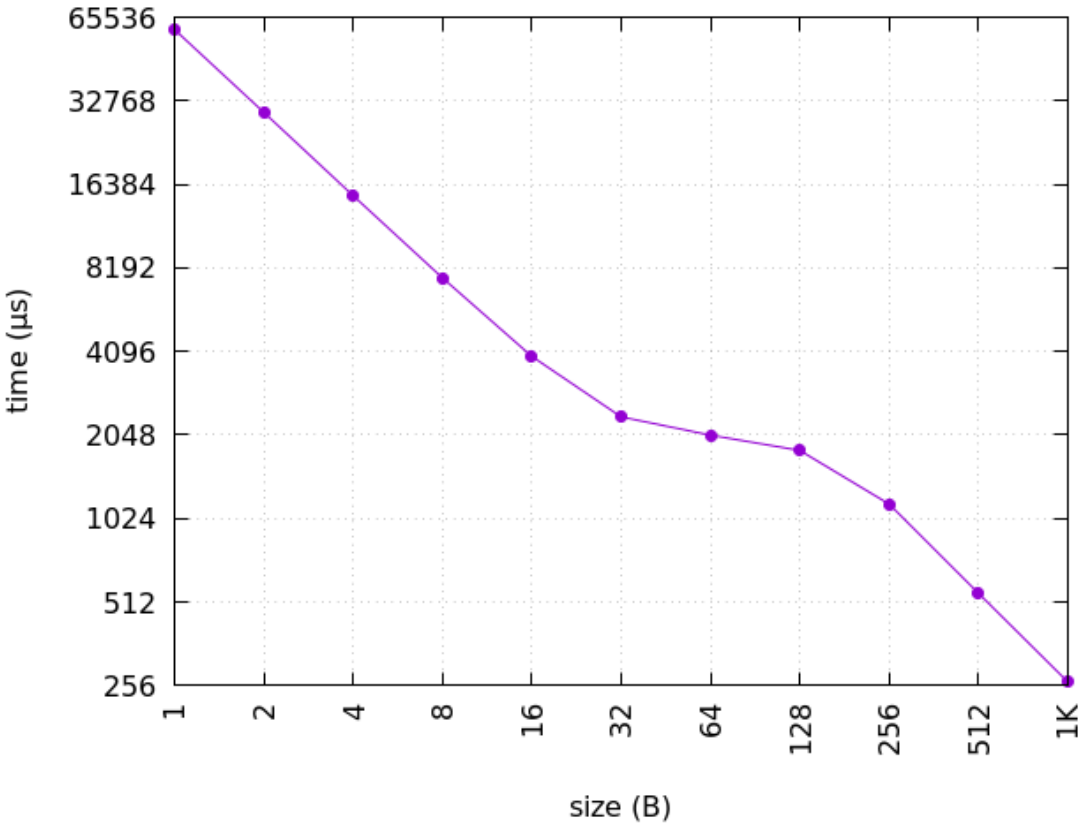
Type	CPU / Microprocessor			
Market segment	Mobile			
Family	Intel Core i7 Mobile			
Model number 	i7-7700HQ			
CPU part number	CL8067702870109 is an OEM/tray microprocessor			
Frequency 	2800 MHz			
Maximum turbo frequency	3800 MHz (1 core) 3600 MHz (2 cores) 3400 MHz (3 or 4 cores)			
Bus speed 	8 GT/s DMI			
Clock multiplier 	28			
Package	1440-ball micro-FCBGA			
Socket	BGA1440			
Size	1.65" x 1.1" / 4.2cm x 2.8cm			
Introduction date	January 3, 2017			
Price at introduction	\$378			

Cache details				
Cache:	L1 data	L1 instruction	L2	L3
Size:	4 x 32 KB	4 x 32 KB	4 x 256 KB	6 MB
Associativity:	8-way set associative	8-way set associative	4-way set associative	12-way set associative
Line size:	64 bytes	64 bytes	64 bytes	64 bytes
Comments:	Direct-mapped	Direct-mapped	Non-inclusive Direct-mapped	Inclusive Shared between all cores

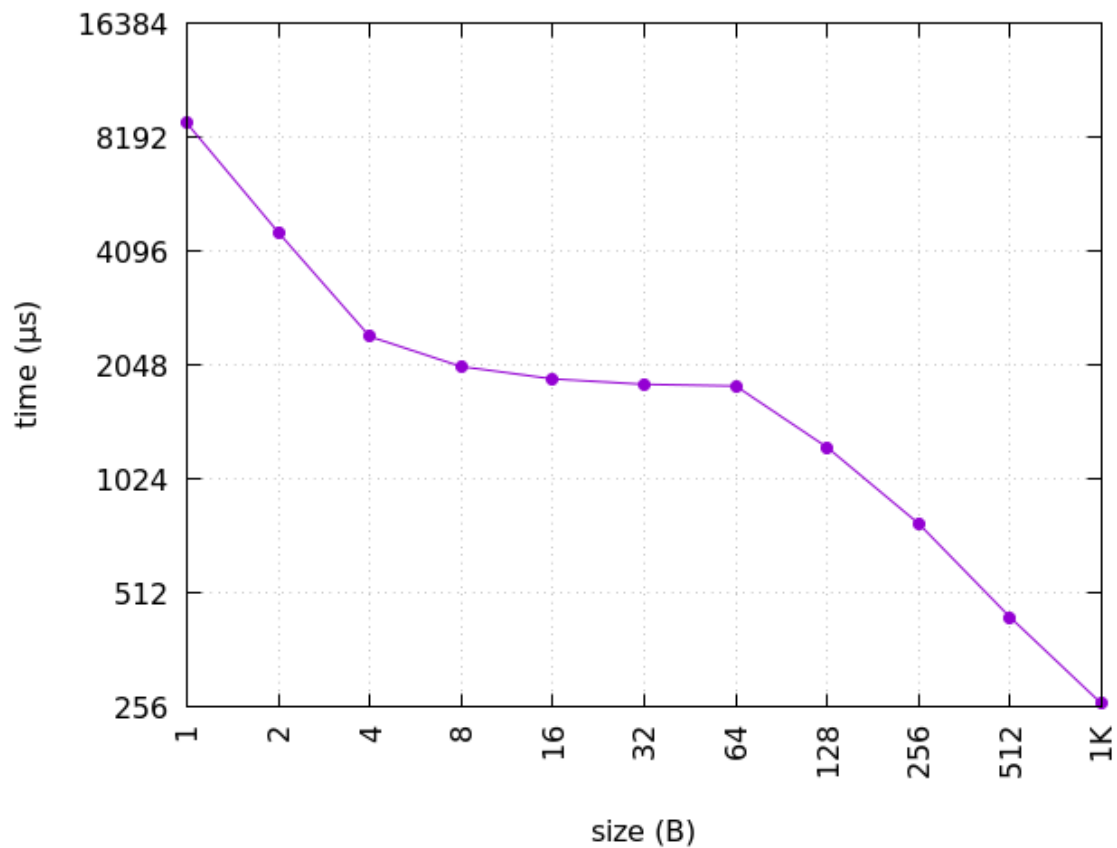
line.Ofast.png



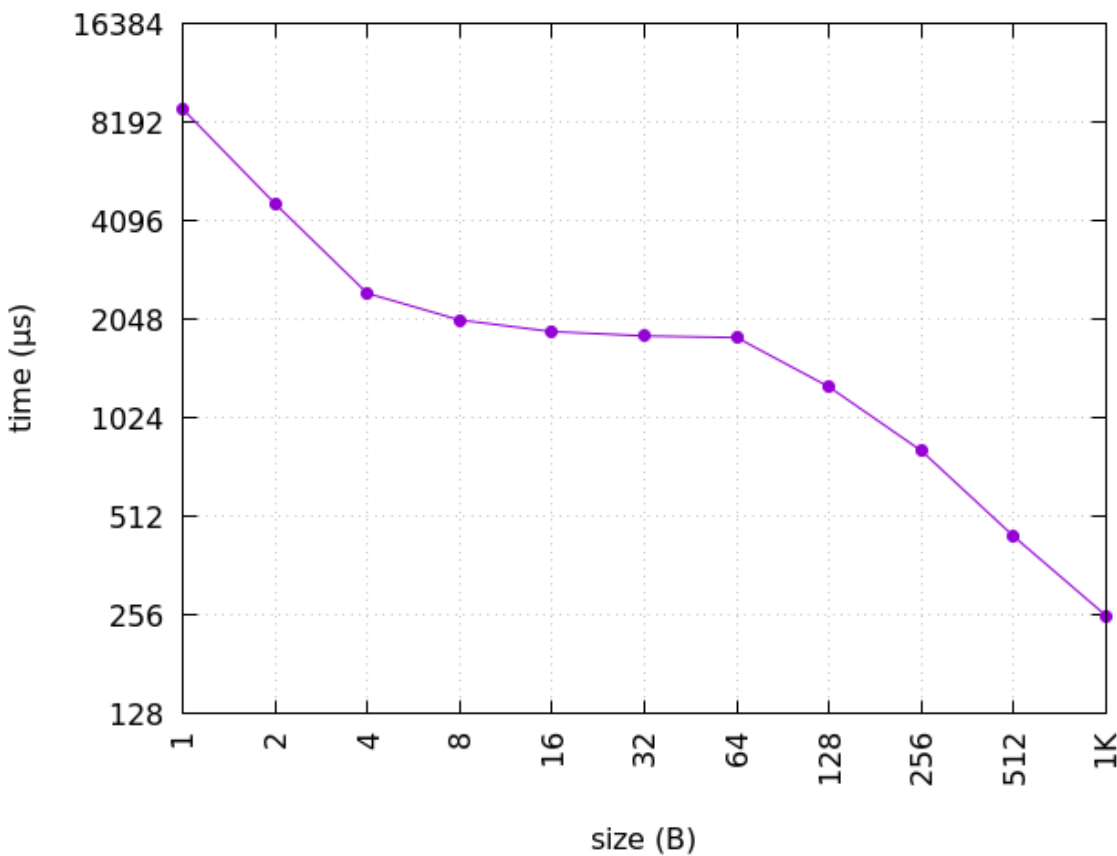
line.O0.png



line.O1.png



line.O2.png



Memoria

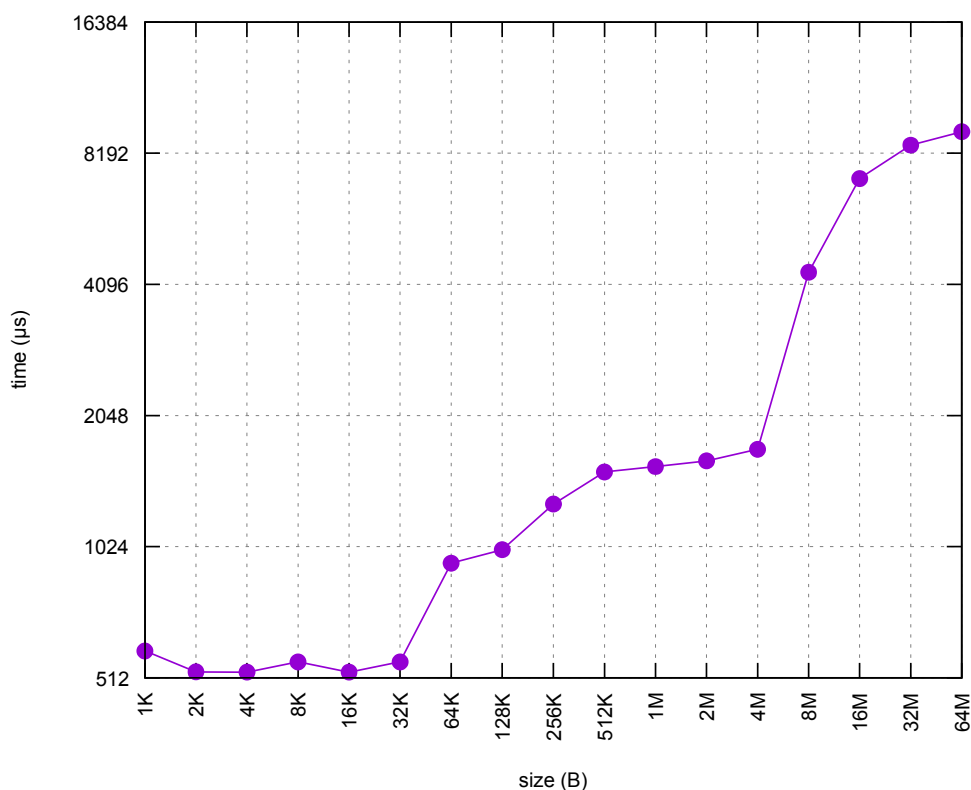
El objetivo de la practica ha sido ver la importancia que tiene la el tamaño de linea de cache en los procesadores y ademas poder calcular la capacidad que tiene la linea de cache de nuestro procesador.

Para ello hemos cogido diferentes tamaños de bloques a leer que tienen de tamaño $2^n B$. y luego con cada tamaño de bloque realizamos una cantidad elevada de operaciones, pero esas operaciones van a tener muy poco peso. Es decir son rápidas de realizar. Con ello podremos ver en como influye el tamaño del bloque. Para cada tamaño de bloque tomamos el tiempo antes y después de realizar las operaciones y asi podemos luego realizar la gráfica.

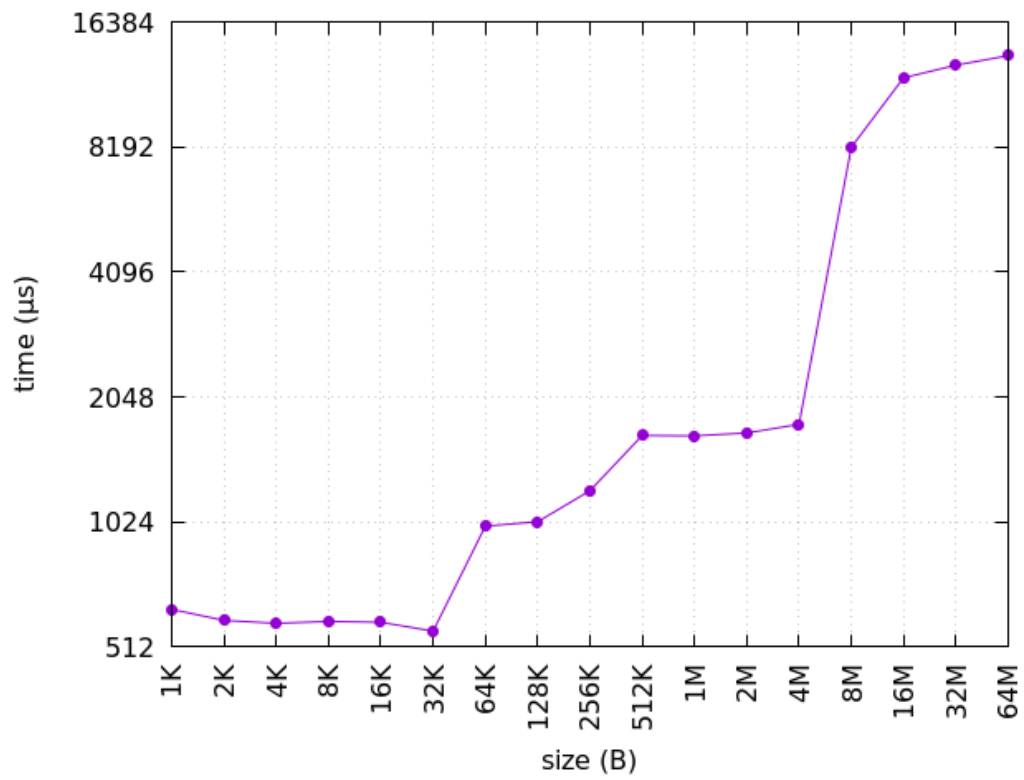
Yo diría que la optimizan de nivel 0 es la que mejor representa la linea, ya que muestra un dato inequívocamente de cual es el tamaño de la cache, ya que las otras pueden dar lugar a confusiones.

Practica 5-B

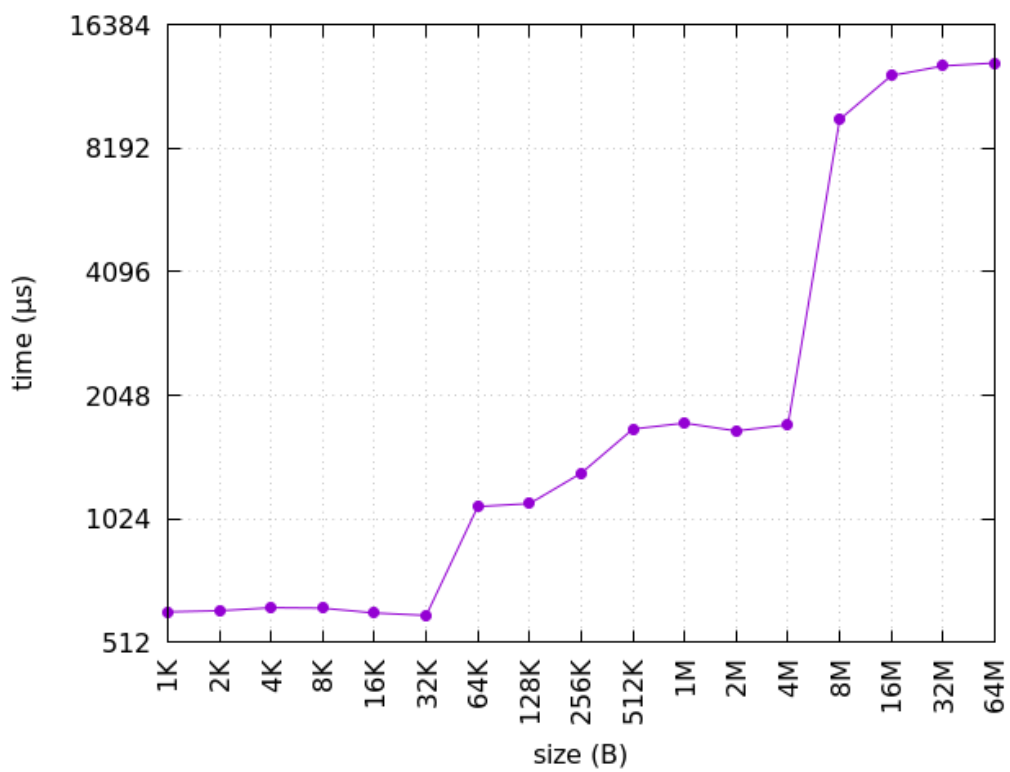
cache.fast.png



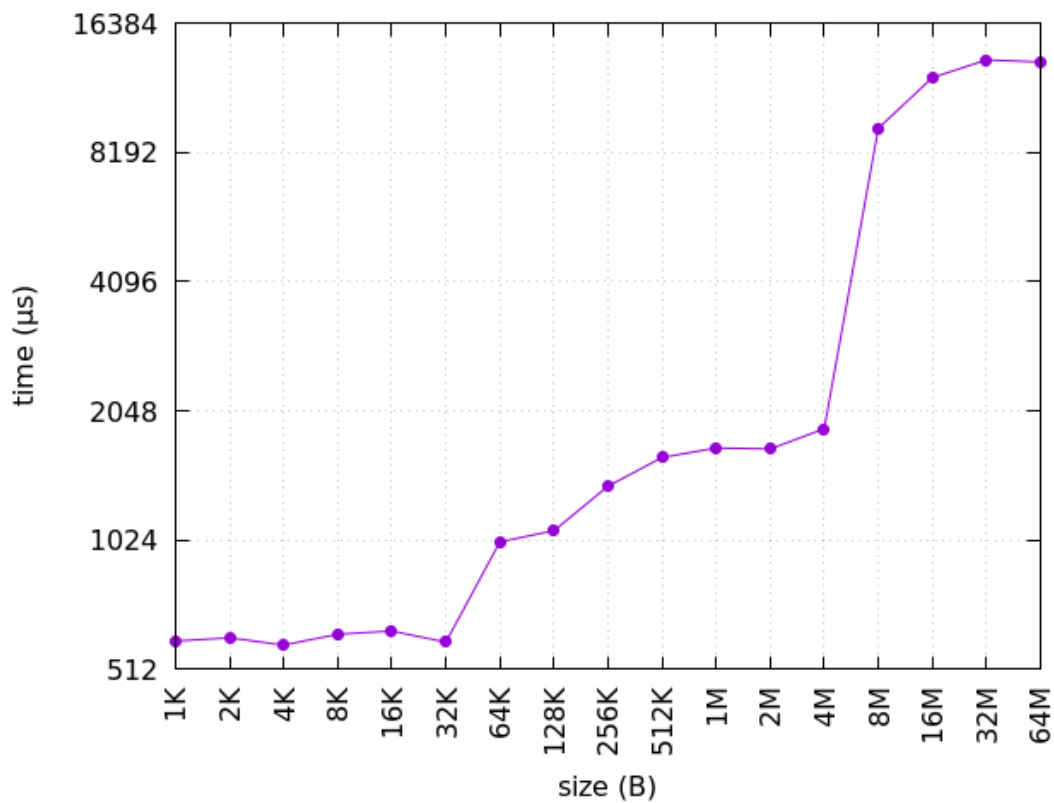
cache.O0.png



cache.O1.png



cache.O2.png



Memoria

El objetivo de la practica ha sido ver el tamaño de las distintas caches que tiene nuestra CPU.

Para ver el tamaño de cada cache del procesador tenemos que mirar en los saltos que se ven en las graficas. Ya que hay indica donde se produce la falta de cache.

Para cada tamaño de caché tenemos que crear un vector de dicho tamaño y realizar una pequeña operación un numero muy elevado de veces y que cuánto más ligero sea el bucle mejor se evidenciará la diferencia de tiempo entre cálculo y acceso a memoria.

Yo diría que la optimizan de nivel 0 es la que mejor se puede ver el tamaño de las distintas caches, ya que no hace trampas el compilador y ademas es donde mas marcadas se ven los saltos de los distintos tamaños.