

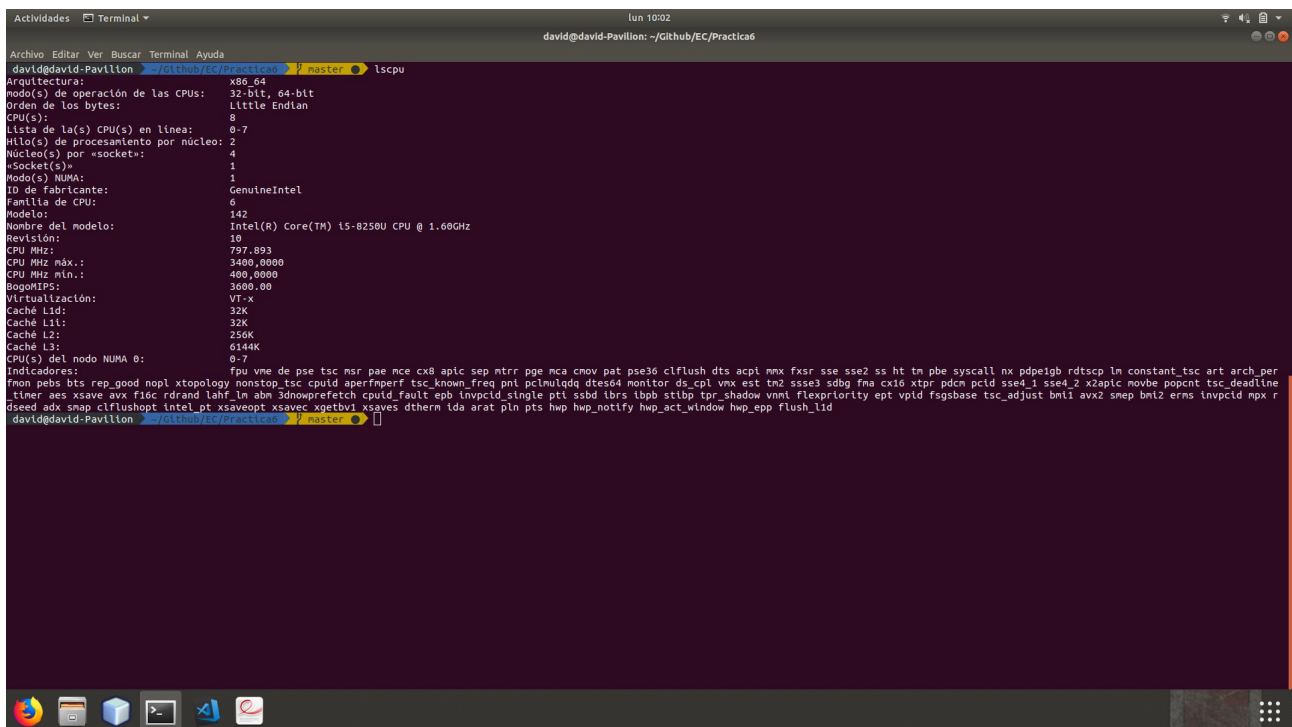
ESTRUCTURA DE LOS COMPUTADORES: Práctica 6a

David Gómez Hernández - 2ºB

En esta sesión hemos trabajado lo que sería las distintas memorias de caché y como su diferente uso en un mismo programa revela los diferentes tamaños de línea de cada uno.

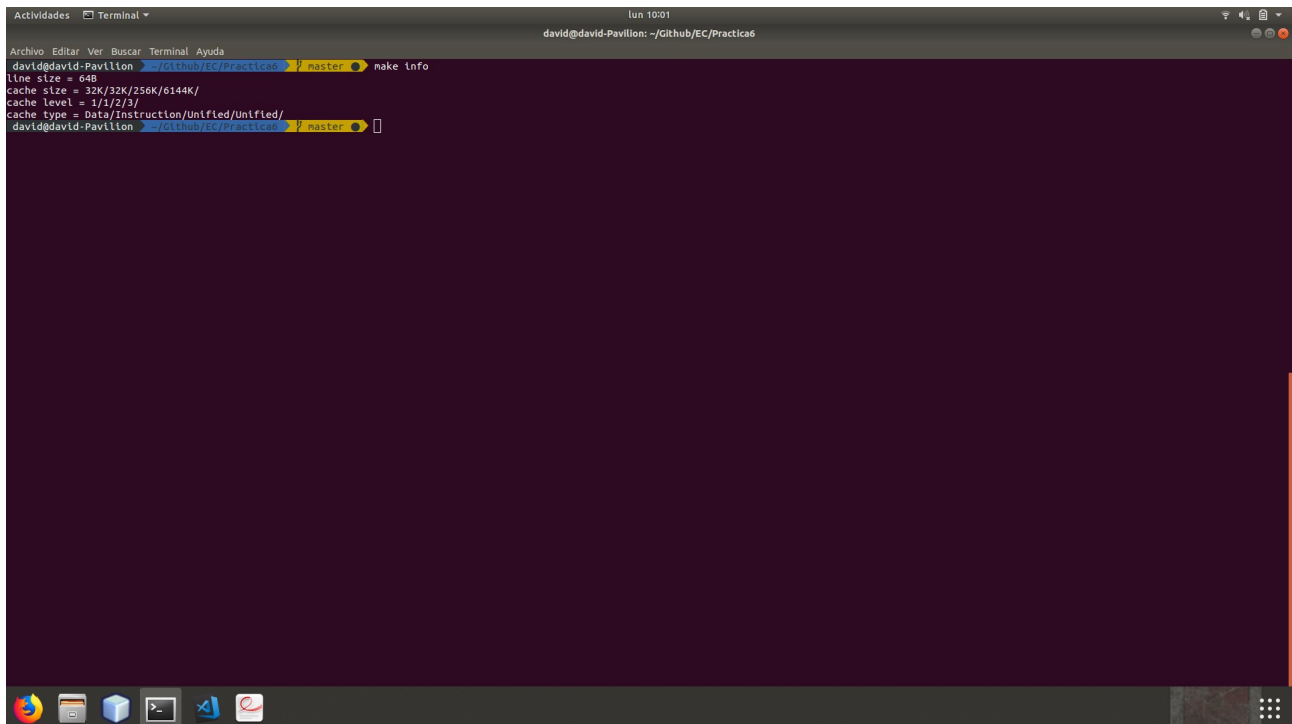
Primero hemos comprobado el tamaño de cada uno de nuestros caches usando 3 herramientas: el comando `lscpu`, haciendo uso del `makefile` ya dado y escribiendo `make info` y por último la página web CPU World dónde buscábamos nuestro procesador.

Mediante el comando `lscpu` pudimos ver el tamaño de las 3 caches del procesador, siendo la primera de 32K tanto su versión de datos como de instrucciones. La segunda tiene un tamaño de 256K y la tercera de 6144K, siendo estas dos últimas una mezcla entre datos e instrucciones.



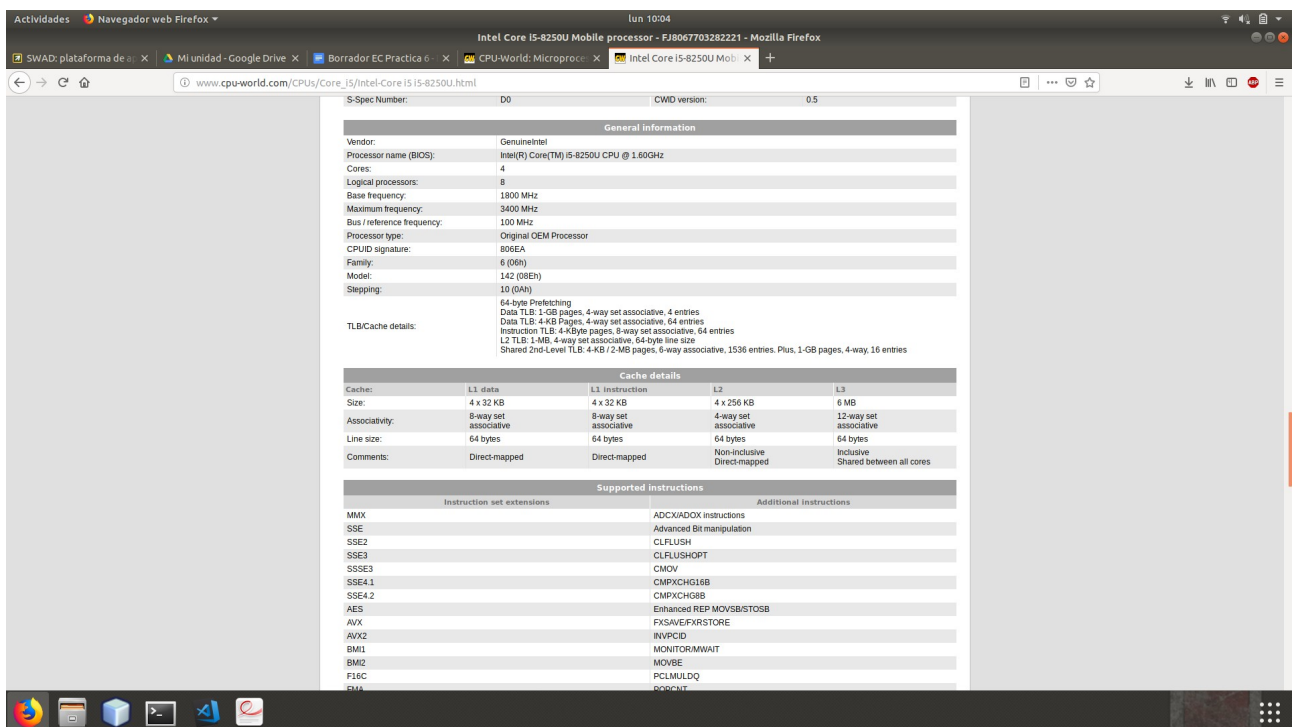
```
Archivos Editar Ver Buscar Terminal Ayuda
david@david-Pavillon: ~/Github/EC/Practicas
$ lscpu
Architectura:             x86_64
Modo(s) de operación de las CPUs: 32-bit, 64-bit
Orden de los bytes:       Little Endian
CPU(s):                   8
Lista de la(s) CPU(s) en línea: 0-7
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»: 4
«Socket(s)»               1
Modo(s) NUMA:             1
ID de fabricante:         GenuineIntel
Familia de CPU:           6
Modelo:                   142
Nombre del modelo:        Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Revisión:                 10
CPU MHz:                  797.893
CPU MHz máx.:             3400.0000
CPU MHz mín.:             400.0000
BogoMIPS:                 3600.00
Virtualización:           VT-x
Cache L1d:                32K
Cache L1i:                32K
Cache L2:                 256K
Cache L3:                 6144K
CPU(s) del nodo NUMA 0:   0-7
Indicadores:              fpu vme de pse tsc nr pae mce cx8 apic sep ntrr pge nca cnov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_per
mon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcd sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline
_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs lbrs stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bti1 avx2 smep bti2 erns invpcid npx r
dseed adx snap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp flush_lid
david@david-Pavillon: ~/Github/EC/Practicas
```

Usando el make info obtenemos la misma información y nos confirma el tipo de caché y el tamaño de linea de toda la memoria que es de 64B.



```
Archivo Editar Ver Buscar Terminal Ayuda
david@david-Pavillon ~/Github/EC/Practicas
line size = 64B
cache size = 32K/32K/256K/6144K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
david@david-Pavillon ~/Github/EC/Practicas
```

Finalmente usando la página web CPU World obtenemos la información al detalle de cada memoria cache.



Cache details				
Caches:	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	Inclusive

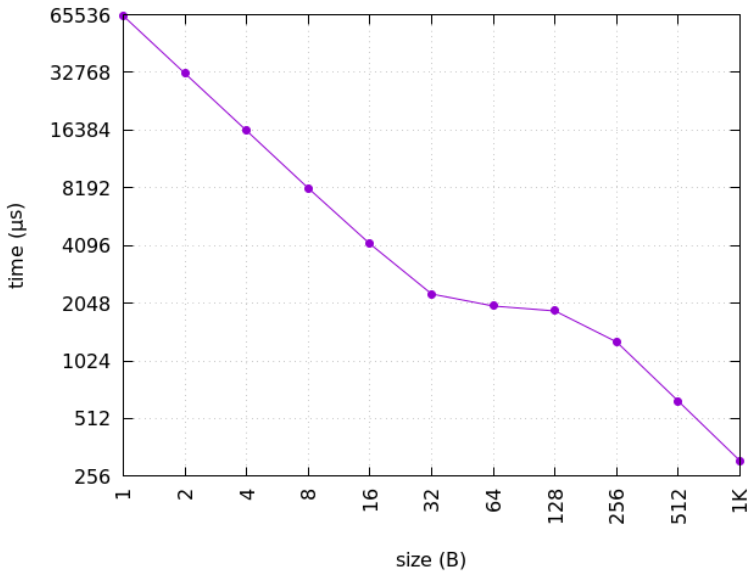
Supported instructions	
Instruction set extensions	Additional instructions
MMX	ADCX/ADOX instructions
SSE	Advanced Bit manipulation
SSE2	CLFLUSH
SSE3	CLFLUSHOPT
SSE4.1	CMOV
SSE4.2	CMPPXCHG16B
AES	CMPPXCHG8B
AVX	Enhanced REP MOVSB/STOSB
AVX2	FXSAVE/FXRSTOR
BM1	INVPID
BM2	MONITOR/MWAIT
F16C	MOVBE
EMM	PCLMULQ
	POPCNT

Una vez que sabemos la información de nuestra caché procedemos a ver qué optimización es la que mejor revela el tamaño de linea.

Antes de ejecutar el programa tenemos que completar la linea que falta, en la que aplicamos la operación lógica EXOR a cada una de las posiciones del vector bytes.

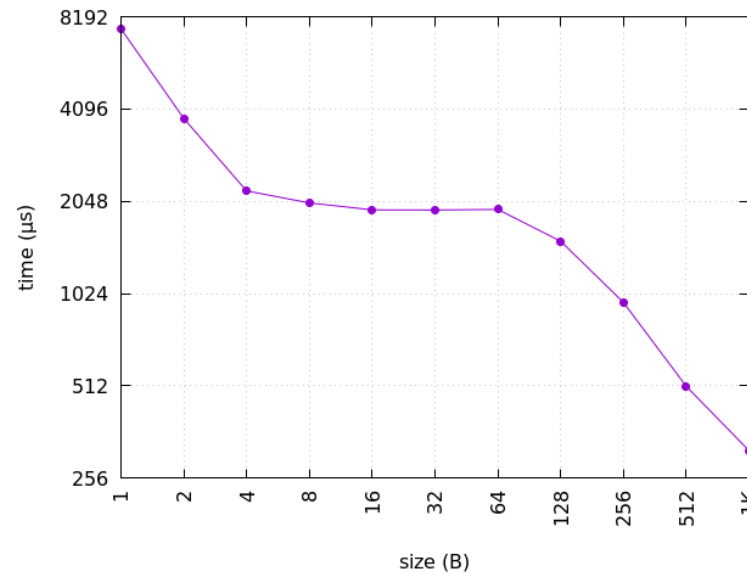
Una vez que ejecutamos el programa con cada una de las optimizaciones obtenemos las siguientes gráficas y tablas.

-O0



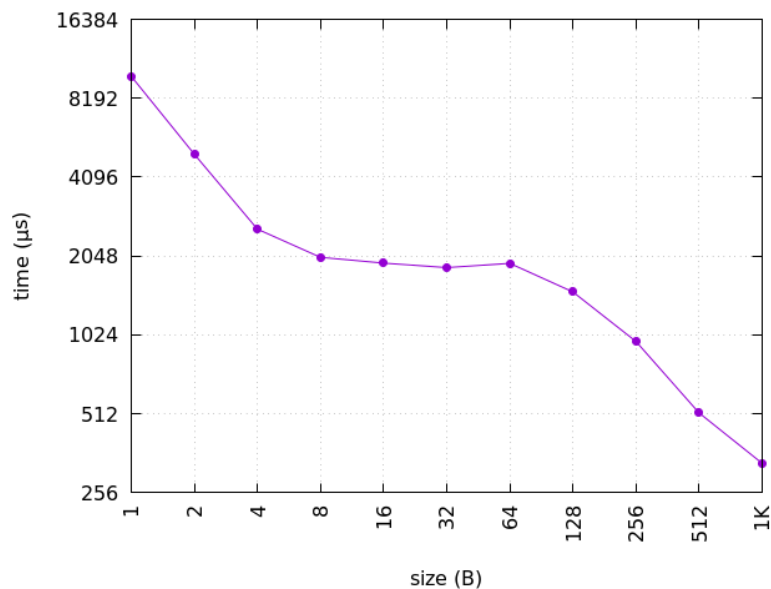
line (B)	time (µs)
1	64630.8
2	32379.8
4	16265.0
8	8151.5
16	4175.2
z32	2287.0
64	1978.6
128	1864.3
256	1284.7
512	630.8
1024	308.2

-O1



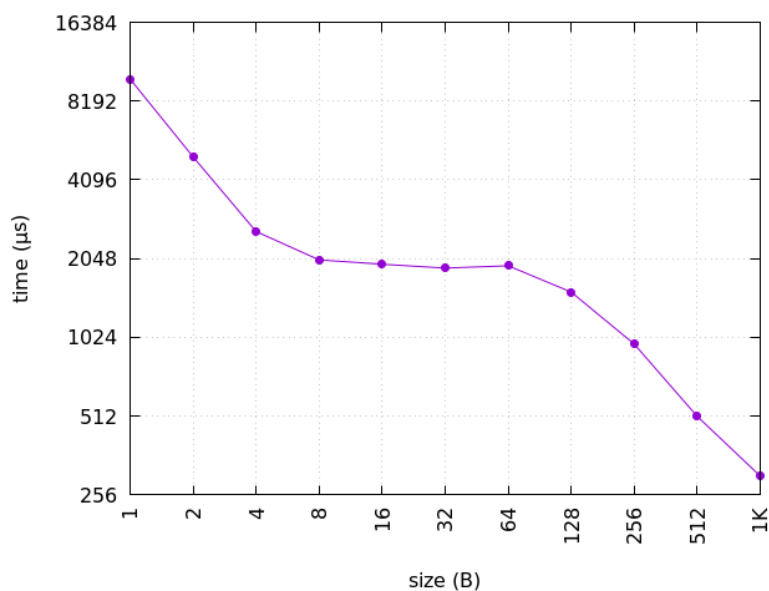
line (B)	time (µs)
1	7478.6
2	3817.5
4	2213.7
8	2024.0
16	1920.9
32	1919.4
64	1926.8
128	1518.6
256	959.0
512	510.7
1024	315.0

-O2



line (B)	time (μs)
1	9934.6
2	5009.0
4	2592.4
8	2024.0
16	1924.8
32	1849.5
64	1920.6
128	1494.3
256	966.8
512	516.0
1024	331.5

-Ofast



line (B)	time (μs)
1	9941.7
2	5014.2
4	2598.6
8	2024.4
16	1949.1
32	1884.2
64	1922.6
128	1525.1
256	965.6
512	510.9
1024	302.3

Tras ver todos estos datos podemos apreciar que el que mejor revela el tamaño de línea es la optimización o1 ya que en menor espacio de tiempo el tamaño se vuelve mayor y además este se vuelve mayor mucho antes que las otras.

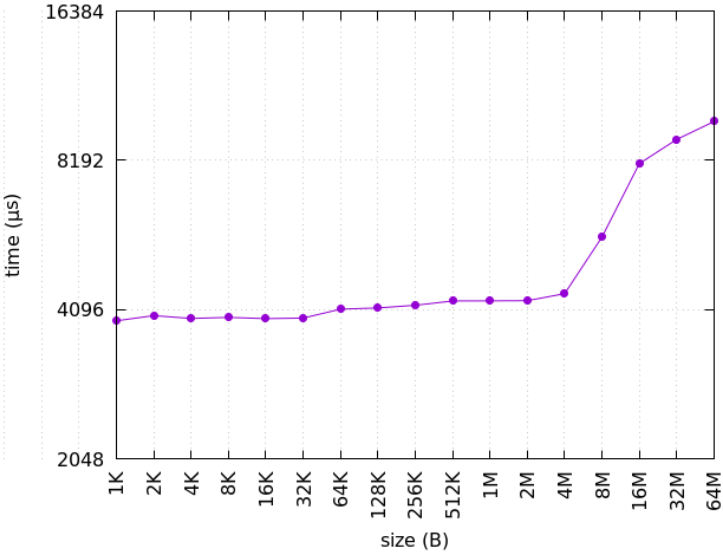
ESTRUCTURA DE LOS COMPUTADORES: Práctica 6b

En esta práctica hemos hecho un proceso parecido al de la anterior práctica solo que ahora en vez de ver el tamaño de líneas vemos el tamaño de la memoria caché qué hay en nuestro ordenador.

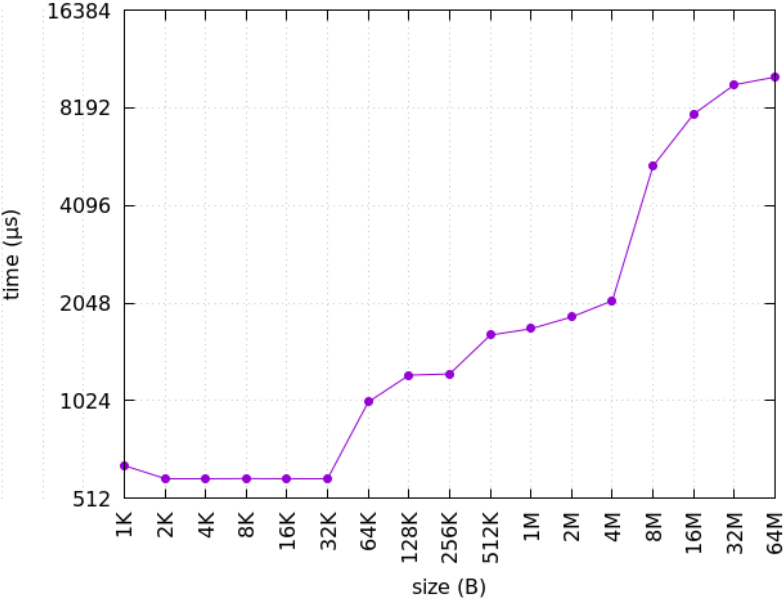
Disponemos de las mismas herramientas usadas en la práctica anterior y explicadas al principio de la memoria. Una vez que sabemos los datos de nuestra caché debemos completar la línea que falta en el archivo size.cc.

En esta ocasión en la línea a completar debíamos hacer una especie de módulo 64 para las posiciones del vector. Al no poder hacerlo hemos hecho que compare con un AND la posición del vector con el tamaño del vector, de tal forma que nos movemos entre módulos de 64 B y comparando con el anterior.

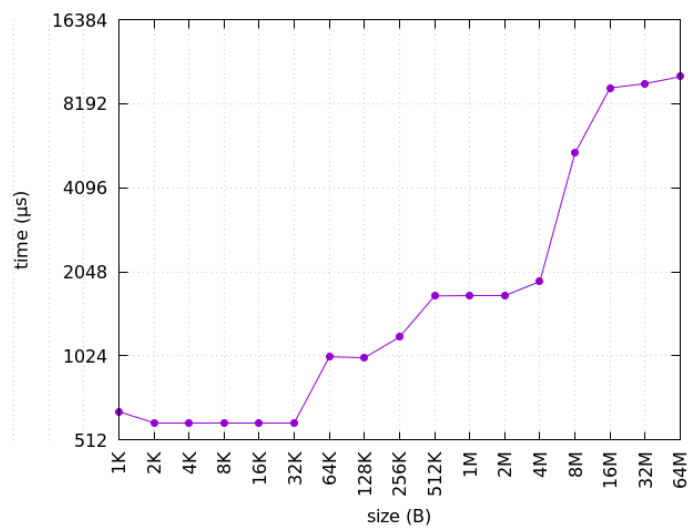
Si esto lo hace simplemente como operación vuelve a hacer un EXOR cómo ya pasó en la anterior práctica. Una vez tenemos todo listo ejecutamos el archivo con las diferentes opciones de optimización y nos da las siguientes tablas y gráficas:



line (B)	time (μs)
1024	3891.0
2048	3986.5
4096	3933.9
8192	3953.6
16384	3930.6
32768	3940.4
65536	4108.9
131072	4131.4
262144	4182.1
524288	4266.6
1048576	4269.2
2097152	4272.6
4194304	4416.6
8388608	5743.5
16777216	8073.0
33554432	9032.4
67108864	9813.1

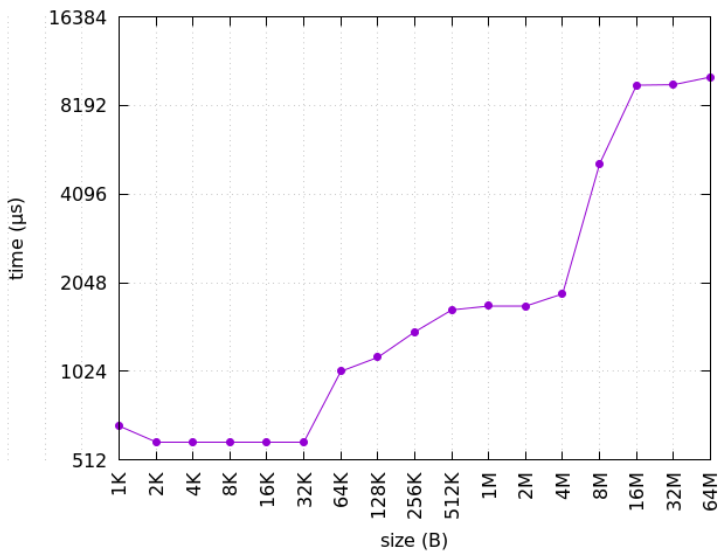


line (B)	time (μs)
1024	646.7
2048	589.7
4096	589.7
8192	589.8
16384	589.7
32768	589.7
65536	1016.1
131072	1228.4
262144	1239.4
524288	1633.0
1048576	1707.4
2097152	1857.7
4194304	2082.8
8388608	5413.6
16777216	7834.9
33554432	9643.2
67108864	10186.7



line (B)	time (μs)
1024	646.9
2048	589.7
4096	589.7
8192	589.7
16384	589.7
32768	589.7
65536	1017.3
131072	1009.3
262144	1197.6
524288	1679.2
1048576	1688.7
2097152	1688.7
4194304	1892.5
8388608	5454.1
16777216	9327.6
33554432	9667.9
67108864	10233.7

-Ofast



line (B)	time (μs)
1024	668.9
2048	589.7
4096	589.7
8192	589.7
16384	589.7
32768	589.7
65536	1025.2
131072	1143.6
262144	1395.4
524288	1656.7
1048576	1707.9
2097152	1707.1
4194304	1879.0
8388608	5187.3
16777216	9594.8
33554432	9634.3
67108864	10229.9

Una vez que hemos sacado las gráficas y vemos los datos observamos que la mejor optimización para ver los tamaños de caché es la -o1 ya que se observan mejor los saltos de memoria que realiza el procesador (32K en la primera memoria, 256K en la segunda y 4M en la tercera).