

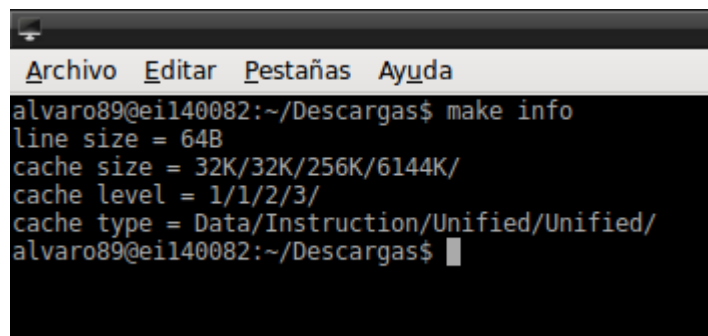
# ESTRUCTURA DE COMPUTADORES

Álvaro Fernández García. Grupo A1

## -MEMORIA CACHÉ-

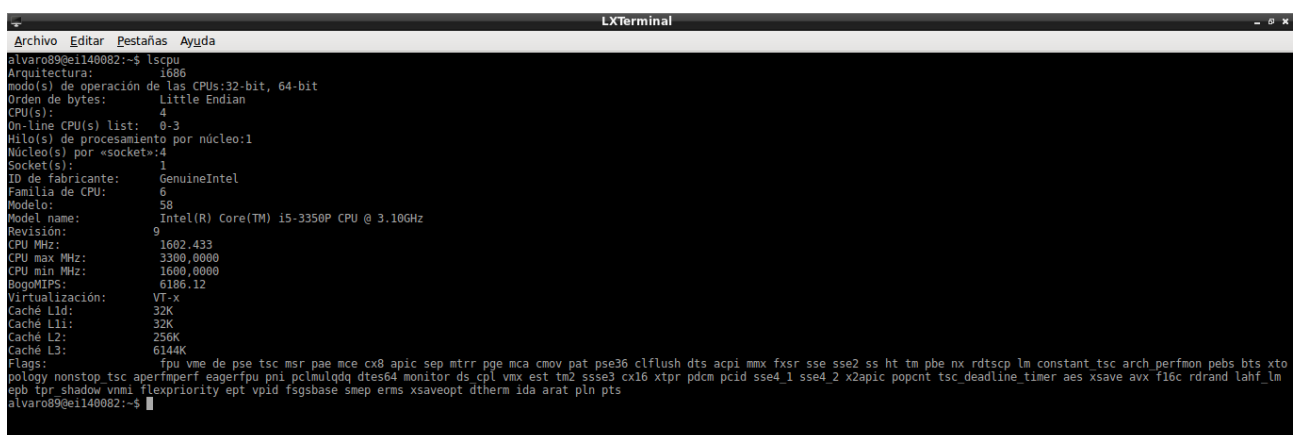
Comenzamos en primer lugar averiguando cuál es la capacidad de la memoria caché de nuestro procesador, para ello podemos utilizar varios mecanismos:

-El primero sería examinando el contenido del directorio `/sys/devices/system/cpu/cpu0/cache`. Dentro podemos encontrar una gran cantidad de archivos, todos ellos al mostrarlos con `cat` nos dan información de la caché; no obstante el `makefile` tiene ya una orden establecida para obtener de esos archivos la información que nos interesa: `make info`. Al ejecutar la orden obtenemos que el tamaño de la línea de caché es 64B (line size), que tenemos 4 memorias (cache levels: 1/1/2/3) de 32, 256 y 6144 K respectivamente (cache size):



```
alvaro89@eil40082:~/Descargas$ make info
line size = 64B
cache size = 32K/32K/256K/6144K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
alvaro89@eil40082:~/Descargas$
```

-La segunda forma que tenemos de conocer las especificaciones es ejecutando en la terminal la orden `lscpu`, que nos lista información en general de nuestro procesador. Podemos observar una serie de campos (Caché L1d, caché L2, caché L3...) que indican los mismos resultados (pero no nos muestra el tamaño de la línea de caché).



```
alvaro89@eil40082:~$ lscpu
Arquitectura: i686
modo(s) de operación de las CPUs: 32-bit, 64-bit
Orden de bytes: Little Endian
CPU(s): 4
On-line CPU(s) list: 0-3
Hilo(s) de procesamiento por núcleo: 1
Núcleo(s) por «socket»: 4
Socket(s): 1
ID de fabricante: GenuineIntel
Familia de CPU: 6
Modelo: 58
Model name: Intel(R) Core(TM) i5-3350P CPU @ 3.10GHz
Revisión: 9
CPU MHz: 1602.433
CPU max MHz: 3300.0000
CPU min MHz: 1600.0000
BogoMIPS: 6106.12
Virtualización: VT-x
Caché L1d: 32K
Caché L1i: 32K
Caché L2: 256K
Caché L3: 6144K
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx rdtscp lm constant tsc arch perfmon pbs bts xto
pology nonstop tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm
epb tpr_shadow vmmi flexpriority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts
alvaro89@eil40082:~$
```

-Sin embargo en estos dos métodos no podemos averiguar si cada núcleo de nuestra CPU tiene su propia caché, o si las cuatro que sabemos que tiene están siendo compartidas, o solo una parte... En definitiva si queremos obtener una información detallada de la caché, lo mejor que podemos hacer es buscar en la web CPU-World el nombre de nuestro procesador (Intel Core i5 i5-3350P (Campo model name de lscpu)) y consultar el apartado de “caché details”, la columna “size”. En esta página por fin sabemos que tiene para cada núcleo (4 cores), una memoria de 32 y 256K pero que la de 6144K es compartida entre todos.

|   |  |                      |                       |
|---|--|----------------------|-----------------------|
| Type                                    | <a href="#">CPU / Microprocessor</a>   |                      |                       |
| Market segment                          | Desktop  |                      |                       |
| Family                                  | <a href="#">Intel Core i5</a>  |                      |                       |
| Model number                            | <a href="#">i5-3350P</a>   |                      |                       |
| CPU part numbers                        | CM8063701392600 is an OEM/tray microprocessor<br>CM8063701280602 is an OEM/tray microprocessor<br>CM8063701280701 is an OEM/tray microprocessor<br>BX80637153350P is a boxed processor with fan and heatsink (English version)<br>BXC80637153350P is a boxed processor with fan and heatsink (Chinese version) |                      |                       |
| Frequency                               | 3100 MHz   |                      |                       |
| Turbo frequency                         | 3300 MHz (1 or 2 cores)<br>3200 MHz (3 cores)  |                      |                       |
| Bus speed                               | 5 GT/s DMI   |                      |                       |
| Clock multiplier                        | 31   |                      |                       |
| Package                                 | 1155-land Flip-Chip Land Grid Array  |                      |                       |
| Socket                                  | <a href="#">Socket 1155 / H2 / LGA1155</a>   |                      |                       |
| Size                                    | 1.48" x 1.48" / 3.75cm x 3.75cm  |                      |                       |
| Introduction date                       | <a href="#">September 2, 2012</a>  |                      |                       |
| End-of-Life date                        | Last order date is March 28, 2014<br>Last shipment date is September 3, 2014   |                      |                       |
| Price at Introduction                   | \$177  |                      |                       |
| <b>S-spec numbers</b>                   |  |                      |                       |
| Part number                             | E5/Q5 processors   |                      | Production processors |
|   | <a href="#">QCDJ</a>   | <a href="#">QCDP</a> | <a href="#">SR0WS</a> |
| BX80637153350P                          |  |                      | +                     |
| BXC80637153350P                         |  |                      | +                     |
| CM8063701280602                         |  | +                    |                       |
| CM8063701280701                         | +  |                      |                       |
| CM8063701392600                         |  |                      | +                     |
| <b>Architecture / Microarchitecture</b> |  |                      |                       |
| Microarchitecture                       | Ivy Bridge   |                      |                       |
| Processor core                          | <a href="#">Ivy Bridge</a>   |                      |                       |
| Core steppings                          | E1 (QCDJ, SR0WS)<br>N0 (QCDP)  |                      |                       |
| CPUID                                   | 306A9 (SR0WS)  |                      |                       |
| Manufacturing process                   | 0.022 micron   |                      |                       |
| Data width                              | 64 bit   |                      |                       |
| The number of CPU cores                 | 4  |                      |                       |
| The number of threads                   | 4  |                      |                       |
| Floating Point Unit                     | Integrated   |                      |                       |
| Level 1 cache size                      | 4 x 32 KB 8-way set associative instruction caches<br>4 x 32 KB 8-way set associative data caches  |                      |                       |
| Level 2 cache size                      | 4 x 256 KB 8-way set associative caches  |                      |                       |
| Level 3 cache size                      | 6 MB 12-way set associative shared cache   |                      |                       |

| General information    |   |
|------------------------|---|
| Vendor:                | GenuineIntel  |
| Processor name (BIOS): | Intel(R) Core(TM) i5-3350P CPU @ 3.10GHz  |
| Cores:                 | 4   |
| Logical processors:    | 4   |
| Processor type:        | Original OEM Processor  |
| CPUID signature:       | 306A9   |
| Family:                | 6 (06h)   |
| Model:                 | 58 (03Ah)   |
| Stepping:              | 9 (09h)   |
| TLB/Cache details:     | 64-byte Prefetching<br>Data TLB0: 2-MB or 4-MB pages, 4-way set associative, 32 entries<br>Data TLB: 4-KB Pages, 4-way set associative, 64 entries<br>Instruction TLB: 4-KB Pages, 4-way set associative, 128 entries<br>L2 TLB: 1-MB, 4-way set associative, 64-byte line size<br>Shared 2nd-level TLB: 4 KB pages, 4-way set associative, 512 entries |

| 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 | 8-way set associative          | 12-way set associative                |
| Line size:     | 64 bytes              | 64 bytes              | 64 bytes                       | 64 bytes                              |
| Comments:      | Direct-mapped         | Direct-mapped         | Non-inclusive<br>Direct-mapped | Inclusive<br>Shared between all cores |

Una vez que conocemos teóricamente la información, pasaremos a comprobar empíricamente el valor de la línea de caché. Para ello realizaremos las mediciones (con distintos niveles de optimización) del tiempo que tarda en realizarse un bucle for (con distintos incrementos que pretenden recrear distintos tamaños de línea) que realiza la operación xor a cada uno de los elementos de un array de 16M de caracteres.

Las mediciones obtenidas son las siguientes:

## -TABLA DE TIEMPOS-

### Optimización -O0

| # line (B) | time (μs) |
|------------|-----------|
| 1          | 88125.3   |
| 2          | 44153.3   |
| 4          | 21997.9   |
| 8          | 11022.7   |
| 16         | 5541.8    |
| 32         | 2826.5    |
| 64         | 1921.8    |
| 128        | 1936.2    |
| 256        | 1389.4    |
| 512        | 600.4     |
| 1024       | 341.2     |

### Optimización -O1

| # line (B) | time (μs) |
|------------|-----------|
| 1          | 10253.8   |
| 2          | 5150.6    |
| 4          | 2666.4    |
| 8          | 1917.6    |
| 16         | 1838.2    |
| 32         | 1847.6    |
| 64         | 1880.5    |
| 128        | 1335.6    |
| 256        | 741.8     |
| 512        | 291.7     |
| 1024       | 146.2     |

### Optimización -O2

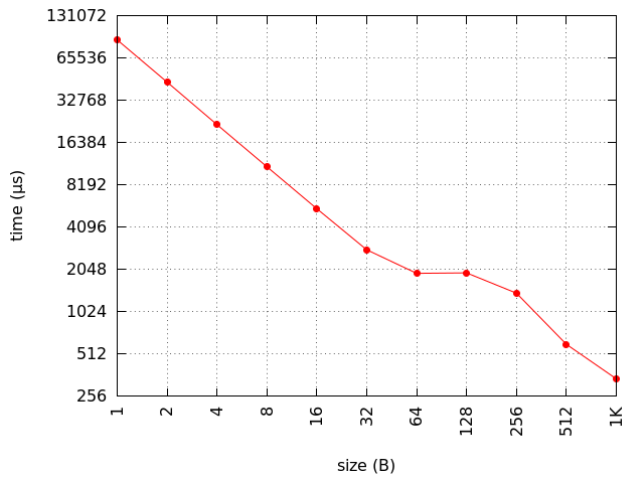
| # line (B) | time (μs) |
|------------|-----------|
| 1          | 10255.5   |
| 2          | 5154.6    |
| 4          | 2662.1    |
| 8          | 1917.2    |
| 16         | 1837.1    |
| 32         | 1845.8    |
| 64         | 1877.7    |
| 128        | 1353.1    |
| 256        | 759.6     |
| 512        | 293.0     |
| 1024       | 146.8     |

### Optimización -Ofast

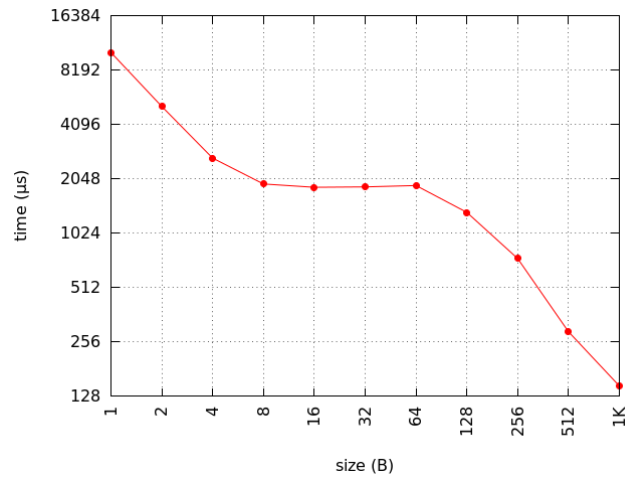
| # line (B) | time (μs) |
|------------|-----------|
| 1          | 10252.4   |
| 2          | 5150.6    |
| 4          | 2649.4    |
| 8          | 1905.4    |
| 16         | 1825.6    |
| 32         | 1838.9    |
| 64         | 1869.3    |
| 128        | 1341.6    |
| 256        | 750.8     |
| 512        | 286.1     |
| 1024       | 143.8     |

# -REPRESENTACIÓN GRÁFICA-

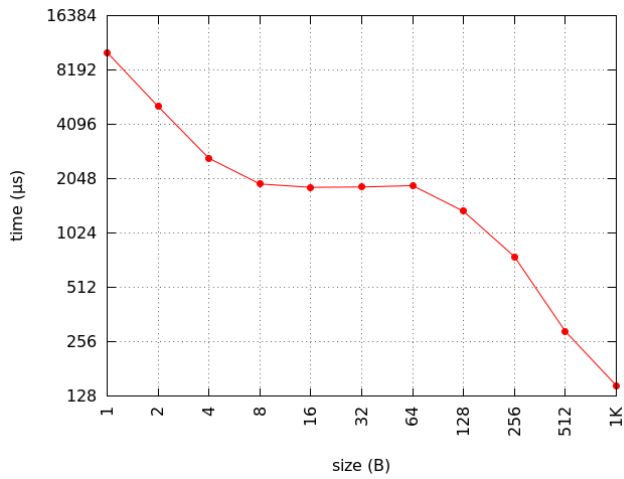
## Optimización -O0



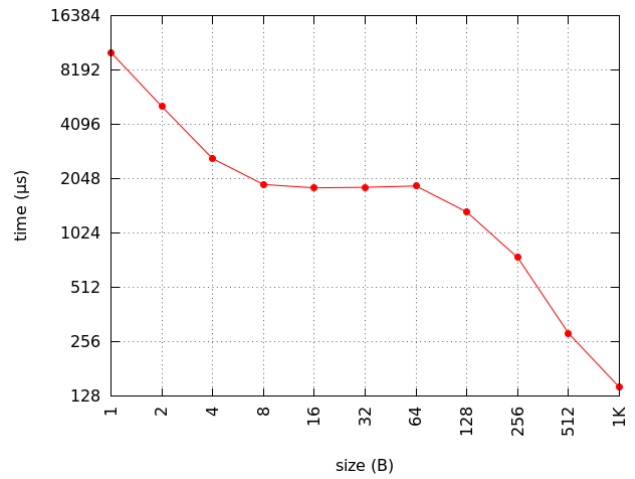
## Optimización -O1



## Optimización -O2



## Optimización -Ofast



Procederemos a continuación a interpretar los resultados, sirviéndonos de las gráficas, que muestran con mayor claridad la relación existente.

Obviando el comportamiento del programa de prueba con optimización 0, nos centraremos en las restantes, ya que las tres tienen un comportamiento similar y cualquiera de ellas nos servirá para explicar el comportamiento de la caché.

Podemos distinguir en ellas tres partes claramente diferenciadas:

- Cuando el incremento de las iteraciones se encuentra en el rango de 8 a 64, podemos observar que el tiempo que tarda el bucle for apenas varía, esto se debe fundamentalmente a que acceder a otros valores de la misma línea de caché no supone prácticamente ningún coste.

- Sin embargo a partir de 64, podemos observar que el tiempo de ejecución se reduce a la mitad cada vez que se duplica el paso. Esto se debe a que ya no vamos tener que acceder a todas las líneas de cache en el array, si no que por ejemplo cuando el paso vale 128 tocaremos más o menos la mitad de las líneas de caché, cuando vale 256, tocaremos un cuarto de las líneas de caché, y así sucesivamente. Esto explica en consecuencia que los tiempos se reduzcan en dicha proporción y que por tanto llegamos al punto en que en cada iteración se necesita una línea de caché nueva. (Por tanto el tamaño de línea es 64K lo cual concuerda con la información de make info).

- Por último queda un intervalo (de 1 a 8), en el que los tiempos aumentan a medida que el incremento por iteración es menor, esto se debe a que acceder en múltiples ocasiones a elementos de la misma línea de caché, no resulta totalmente libre de coste, si no que por el contrario, los efectos de las operaciones empiezan a hacerse visibles.

## **-Segunda parte: Tamaño de los distintos niveles de caché-**

Una vez que ya hemos averiguado el tamaño de la línea de caché (64B) pasaremos en este apartado a intentar averiguar el tamaño de las distintas cachés que tenemos en nuestro computador. En el apartado donde vimos la información de lscpu, make info o cpu-world, pudimos comprobar que nuestro procesador tenía 4 memorias (Levels: 1/1/2/3) de 32, 256 y 6144 K respectivamente. Pero, ¿hay alguna forma de obtener experimentalmente esos valores?. Para ello realizaremos un programa sencillo denominado size.cc el cual, para vectores de caracteres, (los cuales tendrán en cada iteración un tamaño distinto que va desde los 1KB hasta 64MB), se realizará una operación xor con 1 para cada uno de sus elementos, además cuando se llega al último valor, el ciclo vuelve al principio para modificar todos los elementos de la línea. (La operación empleada tiene el menor coste posible).

Las mediciones obtenidas para distintas optimizaciones son las siguientes:

## -TABLA DE TIEMPOS-

### Optimización -O0

| # | line (B) | time (µs) |
|---|----------|-----------|
|   | 1024     | 5848.8    |
|   | 2048     | 5832.1    |
|   | 4096     | 5831.0    |
|   | 8192     | 5830.6    |
|   | 16384    | 5829.6    |
|   | 32768    | 5838.2    |
|   | 65536    | 5877.1    |
|   | 131072   | 5877.3    |
|   | 262144   | 5906.9    |
|   | 524288   | 5938.9    |
|   | 1048576  | 5943.2    |
|   | 2097152  | 5943.2    |
|   | 4194304  | 5960.9    |
|   | 8388608  | 6373.9    |
|   | 16777216 | 6983.6    |
|   | 33554432 | 7482.7    |
|   | 67108864 | 7512.8    |

### Optimización -O1

| # | line (B) | time (µs) |
|---|----------|-----------|
|   | 1024     | 607.5     |
|   | 2048     | 607.5     |
|   | 4096     | 607.5     |
|   | 8192     | 607.5     |
|   | 16384    | 607.5     |
|   | 32768    | 607.5     |
|   | 65536    | 1933.2    |
|   | 131072   | 1940.0    |
|   | 262144   | 2059.8    |
|   | 524288   | 2611.9    |
|   | 1048576  | 2690.9    |
|   | 2097152  | 2691.2    |
|   | 4194304  | 2815.2    |
|   | 8388608  | 4085.1    |
|   | 16777216 | 6453.7    |
|   | 33554432 | 7085.8    |
|   | 67108864 | 7101.3    |

### Optimización -O2

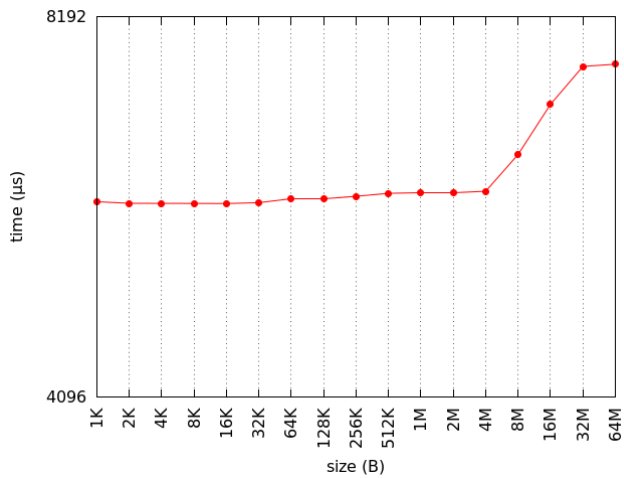
| # | line (B) | time (µs) |
|---|----------|-----------|
|   | 1024     | 911.2     |
|   | 2048     | 911.2     |
|   | 4096     | 911.2     |
|   | 8192     | 911.2     |
|   | 16384    | 911.2     |
|   | 32768    | 911.2     |
|   | 65536    | 1953.3    |
|   | 131072   | 1960.4    |
|   | 262144   | 2128.1    |
|   | 524288   | 2632.7    |
|   | 1048576  | 2689.2    |
|   | 2097152  | 2697.9    |
|   | 4194304  | 2697.9    |
|   | 8388608  | 3940.3    |
|   | 16777216 | 7038.0    |
|   | 33554432 | 7045.3    |
|   | 67108864 | 7124.7    |

### Optimización -Ofast

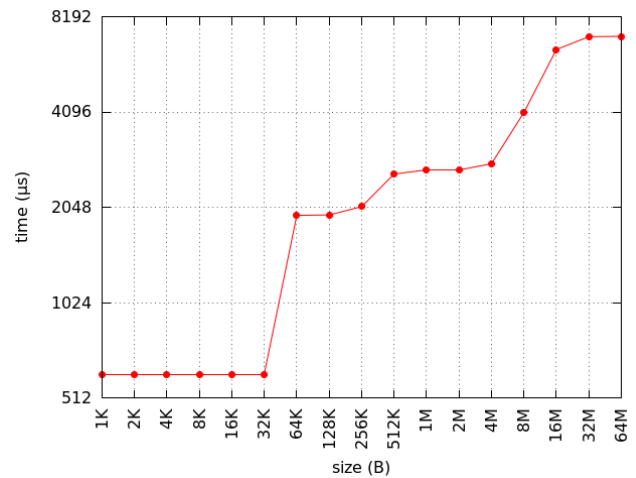
| # | line (B) | time (µs) |
|---|----------|-----------|
|   | 1024     | 911.2     |
|   | 2048     | 911.2     |
|   | 4096     | 911.2     |
|   | 8192     | 911.2     |
|   | 16384    | 911.2     |
|   | 32768    | 911.2     |
|   | 65536    | 1933.2    |
|   | 131072   | 1952.7    |
|   | 262144   | 2116.6    |
|   | 524288   | 2533.1    |
|   | 1048576  | 2690.8    |
|   | 2097152  | 2691.2    |
|   | 4194304  | 2805.0    |
|   | 8388608  | 3992.9    |
|   | 16777216 | 7016.8    |
|   | 33554432 | 7049.5    |
|   | 67108864 | 7074.7    |

# -REPRESENTACIÓN GRÁFICA-

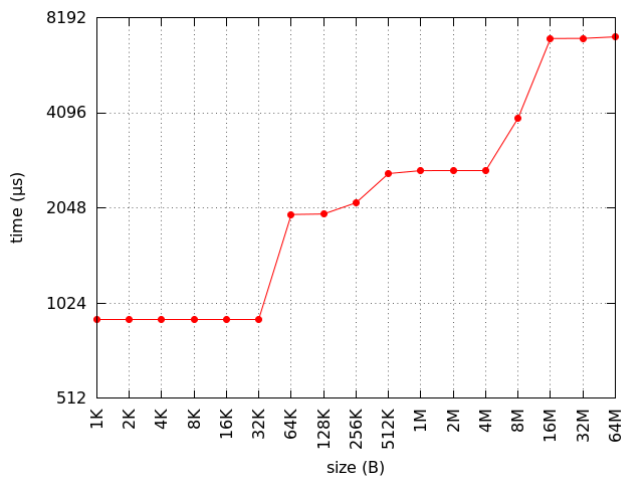
## Optimización -O0



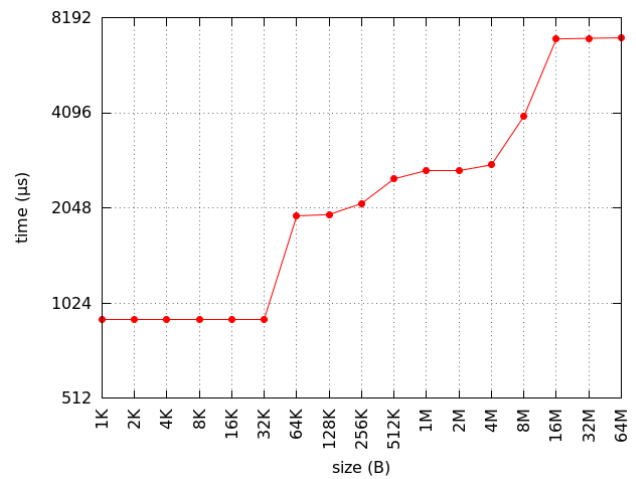
## Optimización -O1



## Optimización -O2



## Optimización -Ofast





Procederemos a continuación a interpretar los resultados, sirviéndonos de las gráficas, que muestran con mayor claridad la relación existente. Obviando el comportamiento del programa de prueba con optimización 0, nos centraremos en las restantes, ya que las tres tienen un comportamiento similar (aunque la optimización 1 no está completamente correcta debido a que tarda menos que las demás optimizaciones, debido posiblemente a la carga del sistema al realizar las mediciones) Dentro de ellas creo que la que mejor evidencia el comportamiento es la optimización -O2, en la que se aprecian con mayor claridad los saltos.

Podemos ver que en la gráfica se producen dos grandes saltos en el rendimiento del programa (hacia tiempos mayores, es decir peor funcionamiento), uno de ellos se produce cuando el tamaño del vector llega justo a los 32K, el segundo se produce justo cuando el tamaño llega a los 256K, y el último en los 4M. Esto se debe a que el vector ya no entra completamente en el mismo nivel de caché y es necesario utilizar el siguiente.

En definitiva podemos deducir que en los puntos donde se producen los saltos en la gráfica coincide con el tamaño de las distintas líneas de caché, de hecho se corresponde, 32K, la de nivel 1 y 256K la de nivel 2, y a partir de 4M, que aunque no se corresponde del todo con los 6M que tiene la caché de nivel 3, si que se empieza a observar que la el tiempo de ejecución del programa es cada vez mayor.