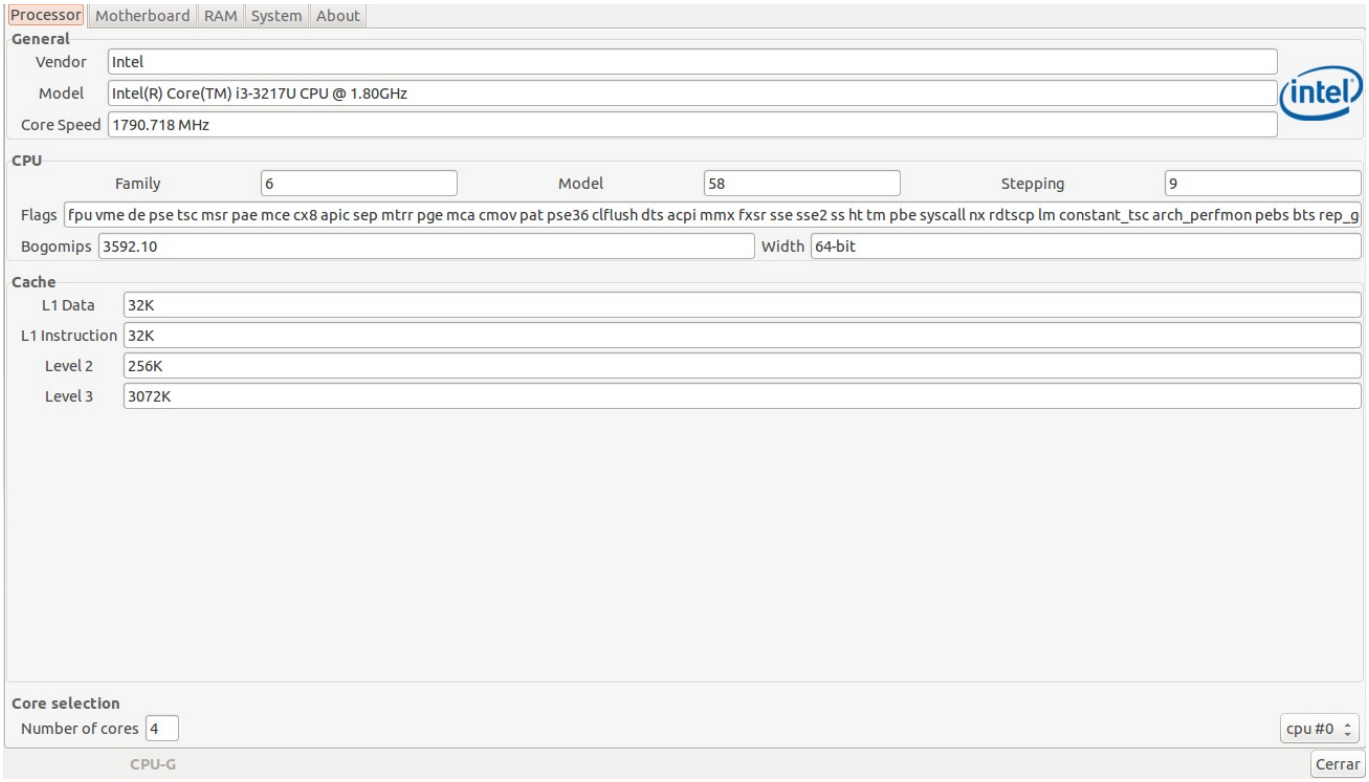
LÍNEA DE CACHÉ

Tamaño de caché y línea de caché

Para averiguar el tamaño de caché y de la línea utilizamos make info del makefile proporcionado, lscpu y cpu-g. De estas herramientas podemos obtener la siguiente información:

```
xkuzz@xKuZz:~/Escritorio$ make info
line size = 64B
cache size = 32K/32K/256K/3072K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
```

```
xkuzz@xKuZz:~/Escritorio$ lscpu
Arquitectura:            x86_64
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:2
Núcleo(s) por «socket»:2
Socket(s):               1
Modo(s) NUMA:            1
ID de fabricante:        GenuineIntel
Familia de CPU:          6
Modelo:                  58
Model name:              Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
Revisión:                9
CPU MHz:                 803.953
CPU max MHz:             1800,0000
CPU min MHz:             800,0000
BogoMIPS:                3592.10
Virtualización:          VT-x
Caché L1d:               32K
Caché L1i:               32K
Caché L2:                256K
Caché L3:                3072K
NUMA node0 CPU(s):       0-3
```



Tamaño de la línea de caché: 64 B (make info)
Tamaño de caché L1 de datos: 32 K (make info, lscpu, CPU-G)
Tamaño de caché L1 de instrucciones: 32 K (make info, lscpu, CPU-G)
Tamaño de caché L2: 256 K (make info, lscpu, CPU-G)
Tamaño de caché L3: 3072 K (make info, lscpu, CPU-G)

Para fijarnos en el tamaño de la línea de caché sólo encuentro dicha información en el valor **line size** de *lscpu*.

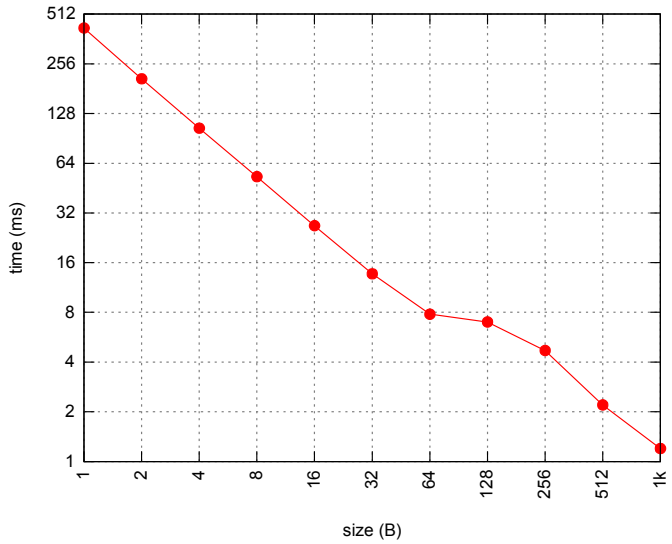
Para fijarnos en el tamaño de la caché de datos miramos make info, lscpu y CPU-G:

NIVEL DE CACHÉ	*make info*	*lscpu*	*CPU-G*
L1 datos	Miro el **primer** valor del campo *cache size*	Miro el valor **Caché L1d**	Miro el valor **L1 data**
L1 instrucciones	Miro el **segundo** valor del campo *cache size*	Miro el valor **Caché L1i**	Miro el valor **L1 instruction**
L2	Miro el **tercer** valor del campo *cache size*	Miro el valor **Caché L2**	Miro el valor **Level 2**
L3	Miro el **cuarto** valor del campo *cache size*	Miro el valor **Caché L3**	Miro el valor **Level 3**

Medición de datos y gráficas

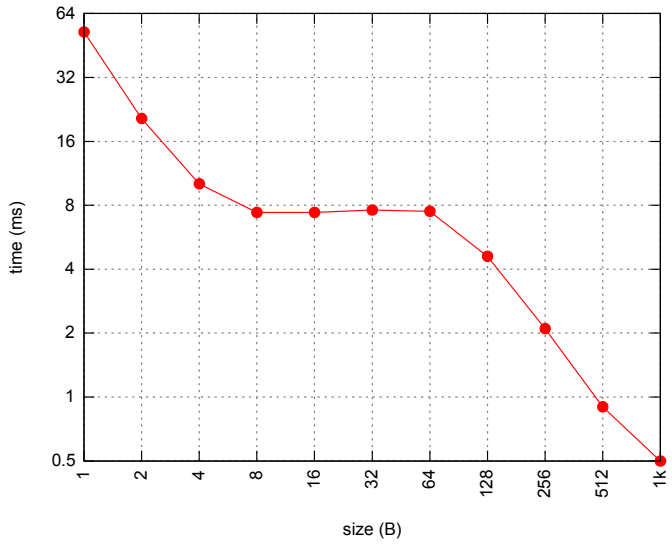
Optimización -O0

line (B)	time (ms)
1	421.1
2	207.7
4	104.3
8	53.1
16	26.8
32	13.7
64	7.8
128	7.0
256	4.7
512	2.2
1024	1.2



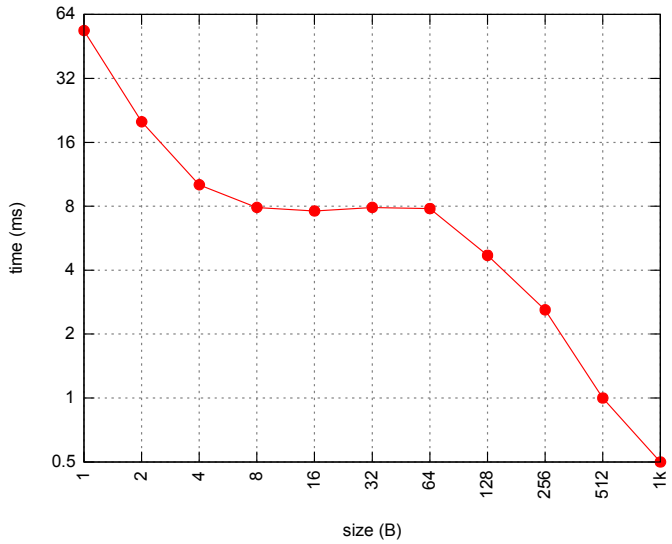
Optimización -O1

line (B)	time (ms)
1	52.4
2	20.5
4	10.1
8	7.4
16	7.4
32	7.6
64	7.5
128	4.6
256	2.1
512	0.9
1024	0.5



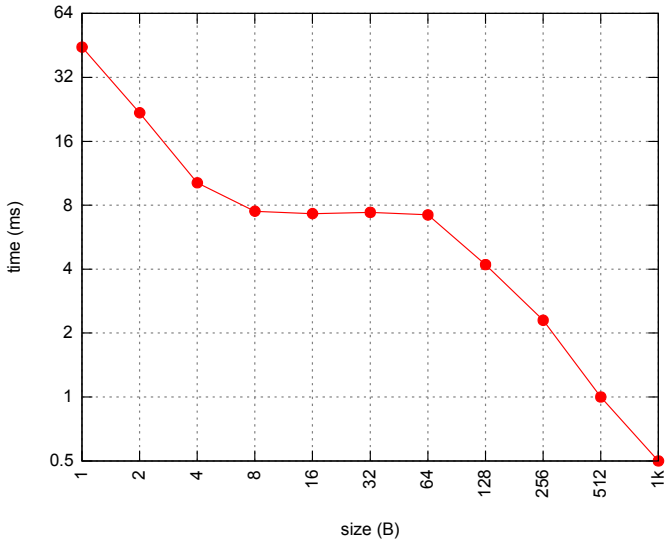
Optimización -O2

line (B)	time (ms)
1	53.7
2	20.0
4	10.1
8	7.9
16	7.6
32	7.9
64	7.8
128	4.7
256	2.6
512	1.0
1024	0.5



Optimización -Ofast

line (B)	time (ms)
1	44.4
2	21.8
4	10.2
8	7.5
16	7.3
32	7.4
64	7.2
128	4.2
256	2.3
512	1.0
1024	0.5



Conclusión

Podemos observar que con un mayor nivel de optimización obtenemos resultados más claros, resulta obvio en todas las gráficas (menos en la de O0) que el tamaño de la **línea de caché es de 64 Bytes** caracterizado por la pendiente negativa que podemos observar en la gráfica, y que podemos afirmar con seguridad debido a la información obtenida en *make info*. Los resultados por tanto son los esperados:

- Para valores menores que el tamaño de la línea de caché se tardará igual (en lo relativo al acceso a memoria) porque tendríamos que acceder a todas las líneas de caché correspondientes al array en cuestión.
- Para valores mayores que el tamaño de la línea de caché no habrá que cargar todas las líneas de caché por lo que el tiempo invertido en acceso a memoria será cada vez más rápido.
- El hecho de que los primeros valores tarden bastante más que los siguientes se debe a que tardan mucho más en llegar al último valor del array. Hay que tener en cuenta que los que utilizan tamaño de línea más grande llegaran antes al último valor y una vez haya sido accedido por primera vez desde memoria no debería ser necesario volver a cargarlo.