Práctica 5

Line.cc

En este primer apartado se pedía averiguar el tamaño de línea de los ordenadores que estamos utilizando. El profesor Gustavo proponía usar “lcpu” para averiguar el tipo de procesador.

```
ignaciove@ei142160:~/Escritorio/Practica 5$ 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:1
Núcleo(s) por «socket»:4
Socket(s):             1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                60
Model name:            Intel(R) Core(TM) i5-4460  CPU @ 3.20GHz
Revisión:              3
CPU MHz:               1423.828
CPU max MHz:           3400,0000
CPU min MHz:           800,0000
BogoMIPS:              6385.55
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              6144K
NUMA node0 CPU(s):     0-3
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx f
xsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology non
stop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_
1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb tpr_shadow vnmi flexpriorit
y ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln pts
ignaciove@ei142160:~/Escritorio/Practica 5$
```

Sabiendo que mi procesador es Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz, lo buscamos en “Cpu-World” obteniendo la siguiente tabla:

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 Direct-mapped	Inclusive Shared between all cores

En la tabla se ha remarcado el lugar donde se indica que el tamaño de línea es de 64 bytes.

La misma información se puede obtener desde la terminal de Linux con la orden “make info” del makefile del profesor Gustavo.

```
ignaciove@ei142160:~/Escritorio/Practica 5$ make info
line size = 64B
cache size = 32K/32K/256K/6144K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
ignaciove@ei142160:~/Escritorio/Practica 5$
```

De esta forma tenemos confirmación del propio Linux de la información que nos proporciona el fabricante. La misma información propone obtenerla el profesor Gustavo completando este programa que nos ha proporcionado:

```
{
        std::vector<duration<double, std::micro>> score(REP);

        for (auto &s: score)
        {
                std::vector<char> bytes(1 << 24); // 16MB

                auto start = high_resolution_clock::now();

                for (unsigned i = 0; i < bytes.size(); i += line)
                        bytes[i] ^= 1;

                auto stop = high_resolution_clock::now();

                s = stop - start;
        }
```

Se ha marcado en rojo la parte que he rellenado del código fuente.

El resultado de repetir las mediciones de tiempo con optimizaciones -Ofast, -O0, -O1, -O2, son las siguientes:

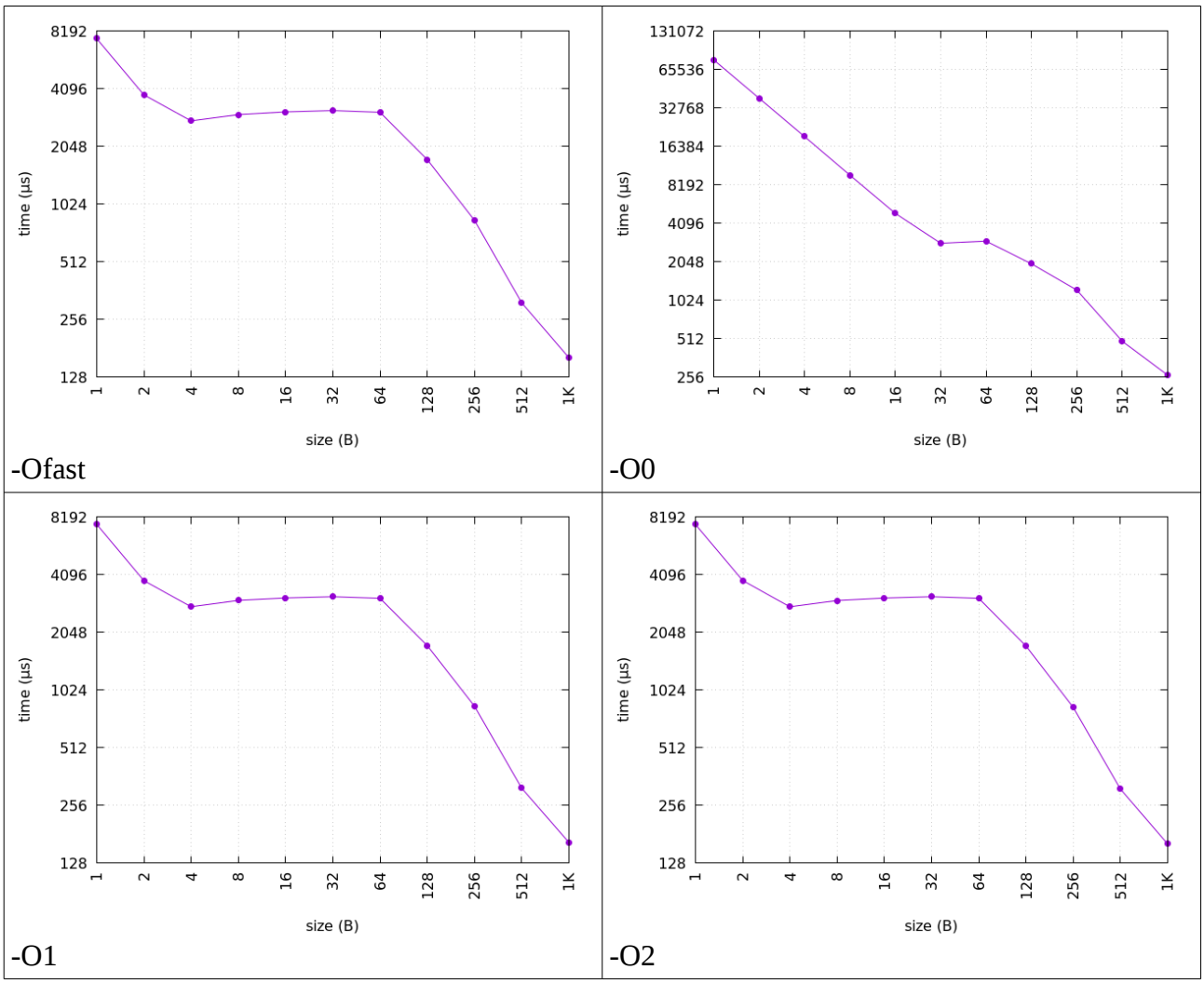
# line (B)	time (µs) -Ofast
1	7468.4
2	3800.7
4	2780.8
8	2997.7
16	3096.6
32	3146.4
64	3080.8
128	1747.1
256	843.0
512	312.8
1024	161.8

# line (B)	time (µs) -O0
1	77689.8
2	38832.7
4	19364.6
8	9741.1
16	4895.9
32	2807.7
64	2937.4
128	1939.3
256	1221.0
512	488.1
1024	265.9

# line (B)	time (µs) -O1
1	7469.0
2	3800.7
4	2777.6
8	3000.6
16	3090.5
32	3137.7
64	3082.5
128	1746.5
256	843.0
512	314.0
1024	162.9

# line (B)	time (µs) -O2
1	7471.4
2	3802.5
4	2830.2
8	3017.2
16	3098.0
32	3154.8
64	3132.5
128	1788.7
256	850.6
512	314.1
1024	163.0

Las gráficas correspondientes a estas mediciones son:



Viendo simultáneamente estas gráficas es fácil deducir que el tamaño de línea es de 64 bytes, la gráfica en la que mejor se ve es la correspondiente a -Ofast, -O1, -O2. El razonamiento es el que sigue:

Esto se deduce del cambio brusco en el tiempo a partir del tamaño de 64 bytes en la gráfica. Puesto que en la caché se almacena de línea en línea (y esta es la operación que consume el mayor número de ciclos, no la operación XOR), hasta el tamaño de 64 se van generando un gran número de aciertos de caché y un fallo por cada línea (para un tamaño de 1, serían 1 fallo y 63 aciertos; para 2, 1 fallo y 62 aciertos). Como el acierto a caché no consume muchos ciclos de CPU, la variación en la gráfica es poco apreciable.

Cuando el tamaño es mayor de 64 bytes, ya no es necesario traerse todos los bloques a la caché, por lo que el ahorro de ciclos es considerable. Esto se refleja claramente en las gráficas indicadas.

(Para un tamaño de 1, el número de iteraciones es tan grande que sí se refleja en las gráficas).

Size.cc

El trabajo de esta semana consisten en adivinar el tamaño de las cachés del procesador que estamos utilizando. Con el comando lscpu cúal es nuestro modelo:

```
ignaciove@ei142160:~/Escritorio/Practica 5$ 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:1
Núcleo(s) por «socket»:4
Socket(s):             1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                60
Model name:            Intel(R) Core(TM) i5-4460  CPU @ 3.20GHz
Revisión:              3
CPU MHz:               1423.828
CPU max MHz:           3400,0000
CPU min MHz:           800,0000
BogoMIPS:              6385.55
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              6144K
NUMA node0 CPU(s):     0-3
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx f
xsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology non
stop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_
1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb tpr_shadow vnmi flexpriorit
y ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dtherm ida arat pln pts
ignaciove@ei142160:~/Escritorio/Practica 5$
```

Sabiendo el modelo de la CPU podemos buscar en CpuWorld sus tamaños de caché:

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 Direct-mapped	Inclusive Shared between all cores

Como vemos, CpuWorld informa que el procesador Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz tiene cachés L1 de 32KB, L2 de 256KB y L3 de 6MB, que coincide con los datos del profesor Gustavo.

```
ignaciove@ei142160:~/Escritorio/Practica 5$ make info
line size = 64B
cache size = 32K/32K/256K/6144K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
ignaciove@ei142160:~/Escritorio/Practica 5$
```

Como podemos comprobar, coinciden los resultados de “make info” con los proporcionados por CpuWorld, pero el profesor Gustavo nos sugiere adivinarlos ejecutando el siguiente código:

```
for (auto &s: score)
{
        std::vector<char> bytes(size);

        auto start = high_resolution_clock::now();

        for (unsigned i = 0; i < STEPS; ++i)
                bytes[(i*64) & (size-1)] ^= 1;

        auto stop = high_resolution_clock::now();

        s = stop - start;
}
```

La línea que he tenido que añadir al código es la que ha sido resaltada.

El resultado de repetir las mediciones de tiempo con optimizaciones -Ofast, -O0, -O1, -O2, son las siguientes:

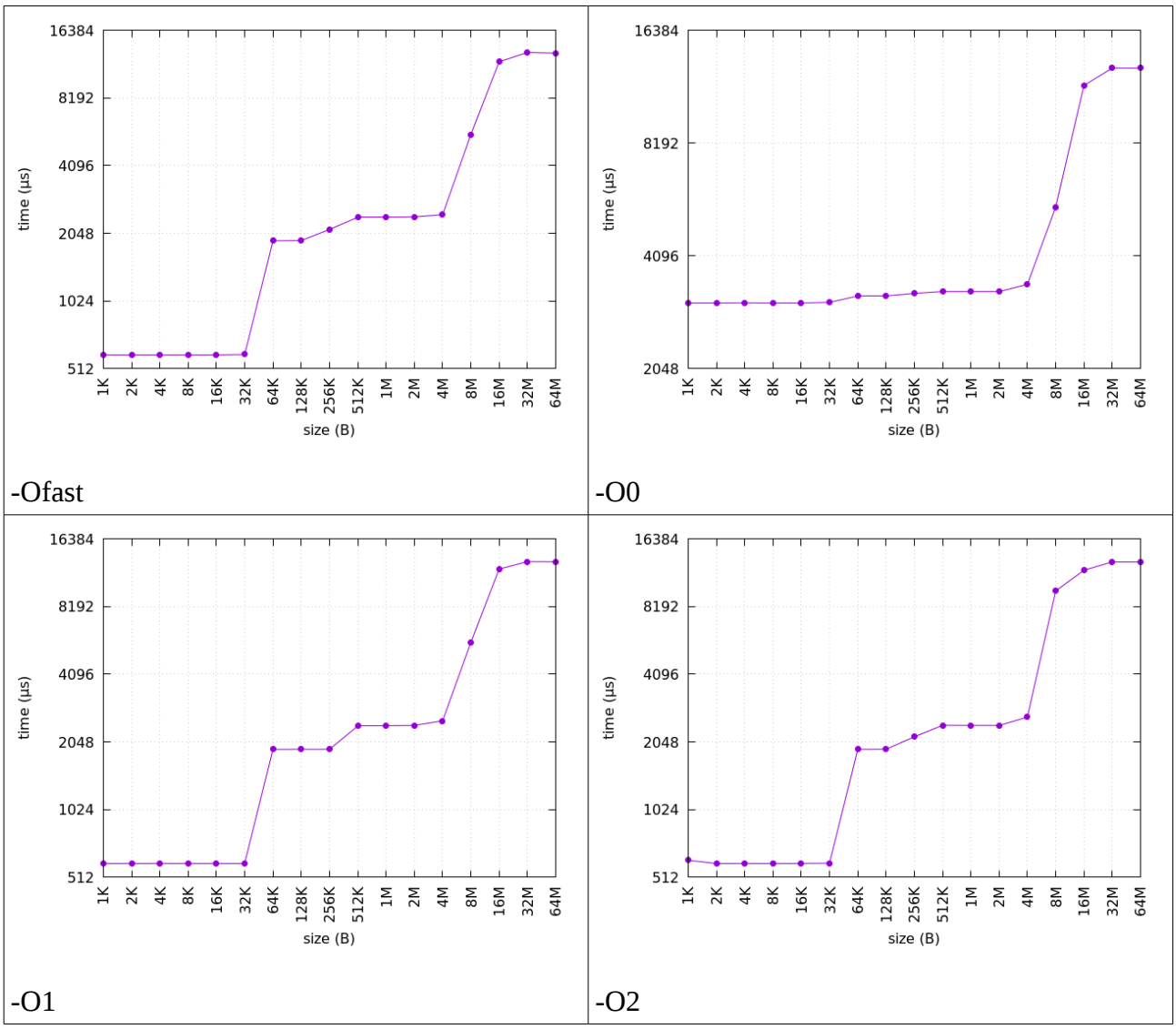
#	line (B)	time (µs) -Ofast
	1024	589.6
	2048	589.6
	4096	589.6
	8192	589.6
	16384	589.6
	32768	592.0
	65536	1896.4
	131072	1901.8
	262144	2130.5
	524288	2418.7
	1048576	2419.2
	2097152	2421.8
	4194304	2476.4
	8388608	5631.8
	16777216	11864.9
	33554432	13044.0
	67108864	12946.6

#	line (B)	time (µs) -O0
	1024	3069.2
	2048	3069.1
	4096	3068.1
	8192	3068.9
	16384	3069.3
	32768	3077.4
	65536	3201.8
	131072	3201.9
	262144	3251.6
	524288	3289.0
	1048576	3290.4
	2097152	3289.4
	4194304	3438.5
	8388608	5528.0
	16777216	11672.3
	33554432	12981.8
	67108864	12976.0

#	line (B)	time (µs) -O1
	1024	589.6
	2048	589.6
	4096	589.6
	8192	589.6
	16384	589.6
	32768	589.6
	65536	1896.9
	131072	1897.6
	262144	1898.6
	524288	2422.9
	1048576	2422.3
	2097152	2423.1
	4194304	2540.5
	8388608	5672.0
	16777216	12016.0
	33554432	12964.1
	67108864	12946.7

#	line (B)	time (µs) -O2
	1024	609.1
	2048	589.6
	4096	589.6
	8192	589.6
	16384	589.6
	32768	591.8
	65536	1897.6
	131072	1899.6
	262144	2157.1
	524288	2425.6
	1048576	2423.2
	2097152	2422.8
	4194304	2637.5
	8388608	9610.6
	16777216	11873.4
	33554432	12953.5
	67108864	12950.2

Las gráficas correspondientes a estas mediciones son:



-Ofast -O0

-O1 -O2

Observando las gráficas se concluyen que el mejor resultado se obtiene con optimización -O1. Fijándose en este resultados, comprobamos que coincide con los tamaños que tenemos de “make info” y de CpuWorld:

Esto se deduce de los 3 cambios que hay en las gráfica: la caché L1 con un tamaño de 32KB; la L2 con un tamaño de 256KB; y la L3 con un tamaño de 6MB (puesto que el cambio no es tan acrecentado).

Cuando el array es de un tamaño de 32KB o menor, este cabe totalmente en la caché L1 (la más rápida), por tanto todos los accesos son aciertos y la duración es muy baja. Al aumentar el tamaño, el array no cabe al completo por lo que se dan accesos a L2 (y respectivamente a L3 al seguir aumentando el tamaño). En el momento en el que el array sobrepasa los 6MB, el procesador debe acceder directamente a memoria principal, que queda reflejado claramente en la gráfica.

Resumidamente, los tamaños y tiempos de accesos de las cachés son los siguientes:

	L1	*L2*	*L3*
Tamaño	32KB	256KB	6MB
Tiempo	589.6µs	2422.3µs	12946.7µs (para 4MB)