



UNIVERSIDAD DE GRANADA

GRADO INGENIERÍA INFORMÁTICA (2017 – 2018)

ESTRUCTURA DE LOS COMPUTADORES

Practica 5: Memoria Caché

Trabajo realizado por :Antonio Miguel Morillo Chica

1. Información CPU con herramientas.

El procesador de mi ordenador es un i7-4500U con 4 núcleos a 1,8 GHz, con 16GB de memoria RAM. A continuación pondré diferentes capturas de cómo ver la información más detallada de la memoria caché.

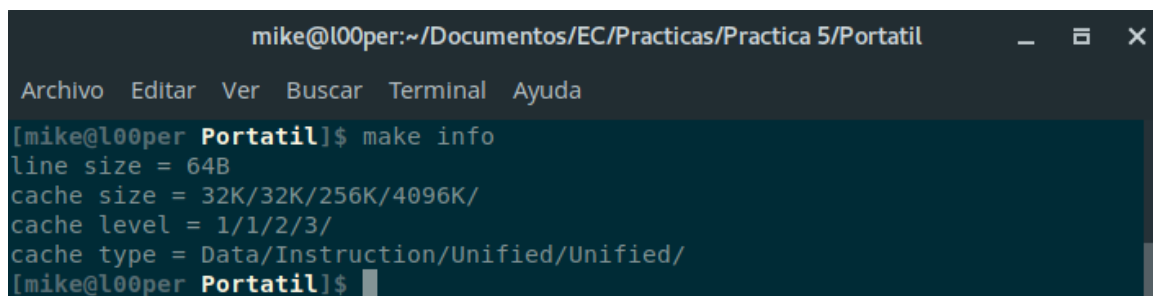
1. Usando CPUWORLD:

Cache details				
Cache:	L1 data	L1 instruction	L2	L3
Size:	2 x 32 KB	2 x 32 KB	2 x 256 KB	4 MB
Associativity:	8-way set associative	8-way set associative	8-way set associative	16-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

2. Con \$ lscpu:

```
mike@l00per:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[mike@l00per ~]$ lscpu  
Arquitectura: x86_64  
modo(s) de operación de las CPUs: 32-bit, 64-bit  
Orden de los bytes: Little Endian  
CPU(s): 4  
Lista de la(s) CPU(s) en línea: 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: 69  
Nombre del modelo: Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz  
Revisión: 1  
CPU MHz: 1139.941  
CPU MHz máx.: 3000,0000  
CPU MHz mín.: 800,0000  
BogoMIPS: 4790.37  
Virtualización: VT-x  
Caché L1d: 32K  
Caché L1i: 32K  
Caché L2: 256K  
Caché L3: 4096K  
CPU(s) del nodo NUMA 0: 0-3  
Indicadores: fpu vme de pse tsc msr pae mce cx8 apic s  
ep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe  
syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl  
xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl  
vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 movbe popcnt tsc  
_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb tpr_shadow vnmi flex  
priority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt  
t dtherm ida arat pln pts  
[mike@l00per ~]$
```

3. Con el propio makefile, usando: `$ make info`



```
mike@l00per:~/Documentos/EC/Practicas/Practica 5/Portatil
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[mike@l00per Portatil]$ make info
line size = 64B
cache size = 32K/32K/256K/4096K/
cache level = 1/1/2/3/
cache type = Data/Instruction/Unified/Unified/
[mike@l00per Portatil]$
```

4. El último método es revisando el archivo `cpuinfo` que se encuentra en `/proc/cpuinfo`. No lo aportaré porque es muy largo pero solo habría que escribir: `$ cat /proc/cpuinfo`

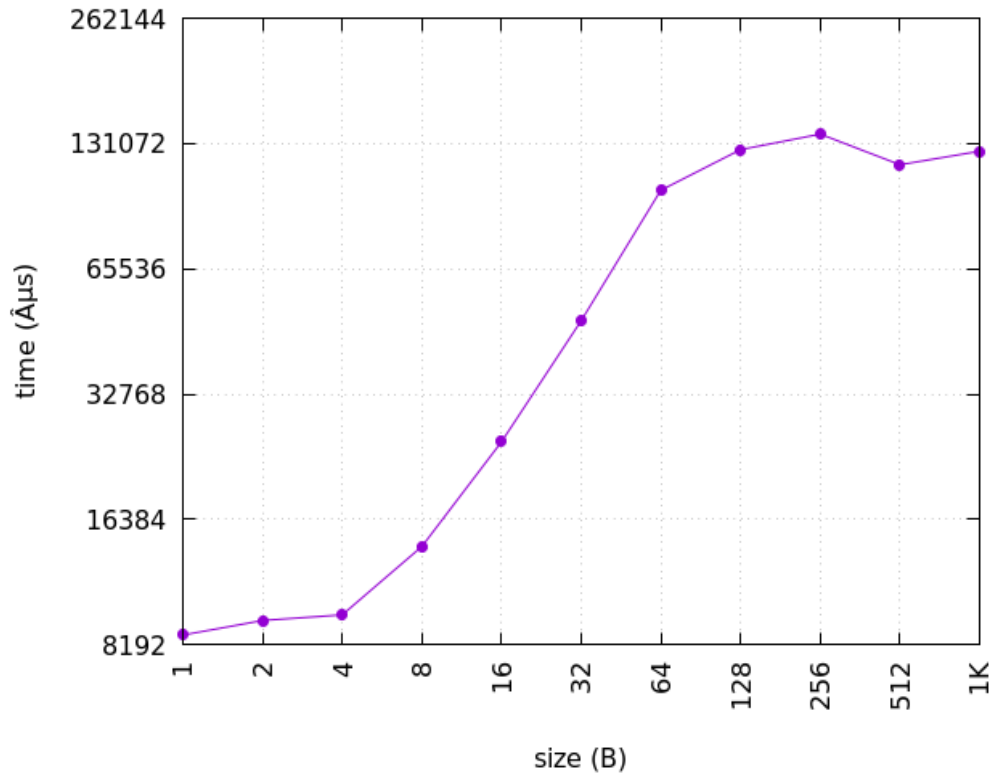
Sabemos entonces que la caché del procesador de mi portatil, i7-4500U posee una caché de 4MB que es el tamaño de la L3, la L2 y L1 son mucho más pequeñas y rápidas, de 256KB my 32KB. Hay que nombrar que la L1 se divide en L1i e L1d de instrucciones y dados.

A continuación mostraremos como darnos cuenta del tamaño de caché sin tener las herramientas online y de más arriba.

2. Información CPU sin herramientas. (Portatil)

2.1. Averiguando tamaño de linea/via.

Para averiguar el maño de la linea de caché lo que haremos es acceder a un vector de tamaño de 16MB y acceder a todo el vector realizando escrituras mediante un xor ya que esta operación es muy liana y acentuará los resultados de la gráfica.

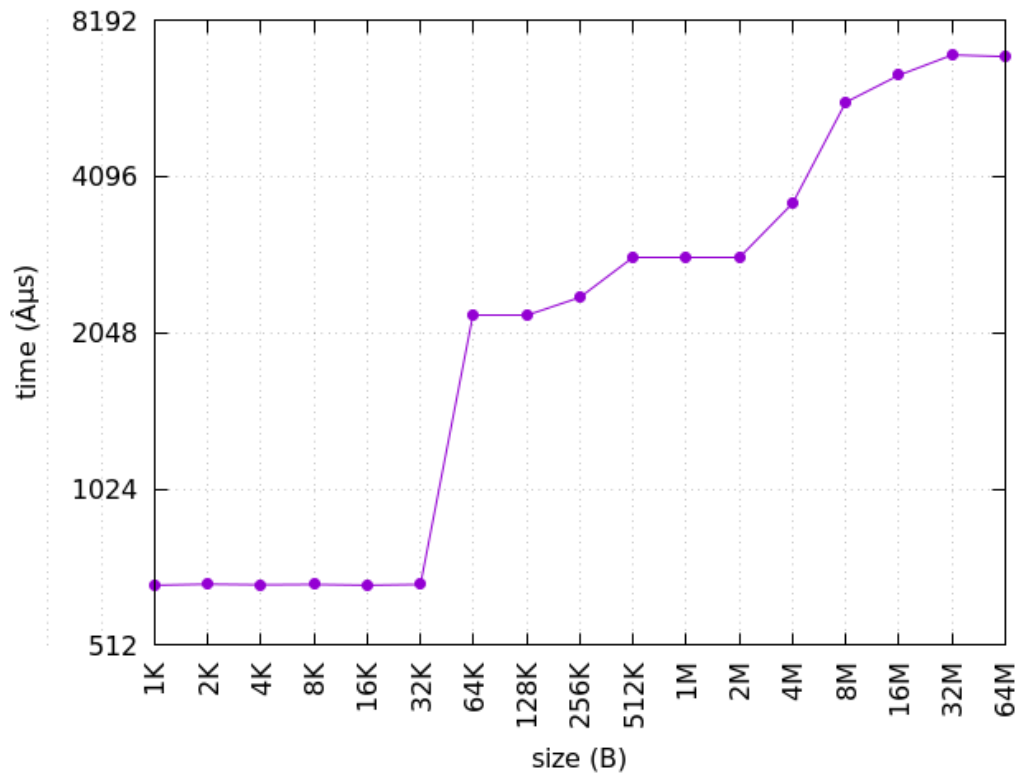


Como podemos ver una vez superados la cantidad de 64B la grafica se estabiliza en un tiempo constante esto es porque a partir de ah  tendr  siempre la misma cantidad de fallos por lo que el tiempo para todos ser  parecido.

2.2. Averiguando tama o de las cach s.

Para averiguar el tama o de las cach s lo que hacemos es trabajar con un vector de longitud variable, es decir, en cada iteracci n vamos a cambiar el tama o del vector del doble de tama o, una vez superado los ma os de los diferentes niveles lo que ocurrir  es que, por ejemplo, si supera los 32K de la L1 por ejemplo este vector no cabr  en cach  y el de una tacada se traer  a memoria 32K en un caso perfecto y posteriormente lo que quede.

Si el tama o del vector es tam, y est  est  comprendido entre: $L1 < tam < L2$ se produzcan dos accesos. Como podemos ver en la imagen siguiente esto se puede comprobar:



Con respecto a la segunda imagen podemos observar tres crecimientos distintos por lo que aparentemente tiene 3 niveles de caché. Desde los tamaños 32K, 256K y 4096K se ve que el tiempo de acceso es cada vez mayor debido a la velocidad de acceso de las diferentes cachés, L1, L2 y L3 respectivamente, L1 es la más rápida y L3 la más lenta.

3. Información CPU sin herramientas (Raspberry Pi 3B)

Como trabajo extra he realizado una prueba como curiosidad de que tiene una Raspberry Pi 3B. La Raspberry Pi es un computador de placa reducida, computador de placa única o computador de placa simple (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

La información del hardware específica que aporta Wikipedia es la siguiente:

	Raspberry Pi 3B
SoC	Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + Puerto USB)
CPU	1.2GHz 64-bit quad-core ARMv8/7
Juego de Instrucciones	RISC de 32 bits
GPU	Broadcom VideoCore IV,,61 OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia),59 1080p30 H.264/MPEG-4 AVC3
Memoria (SDRAM)	1GB (compartidos con la GPU)
Sistemas soportados	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, SUSE 68 Linux Enterprise Server for ARM. RISC OS2

Debido a que no tengo ninguna pantalla asociada lo que he hecho ha sido redireccionar la salida a un archivo de la orden: *\$ cat /etc/cpuinfo*

```
processor      : 0
model name    : ARMv7 Processor rev 4 (v7l)
BogoMIPS     : 38.40
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant   : 0x0
CPU part      : 0xd03
CPU revision: 4

processor      : 1
model name    : ARMv7 Processor rev 4 (v7l)
BogoMIPS     : 38.40
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
```

```
CPU architecture: 7
CPU variant : 0x0
CPU part    : 0xd03
CPU revision: 4

processor    : 2
model name  : ARMv7 Processor rev 4 (v7l)
BogoMIPS    : 38.40
Features     : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part    : 0xd03
CPU revision: 4

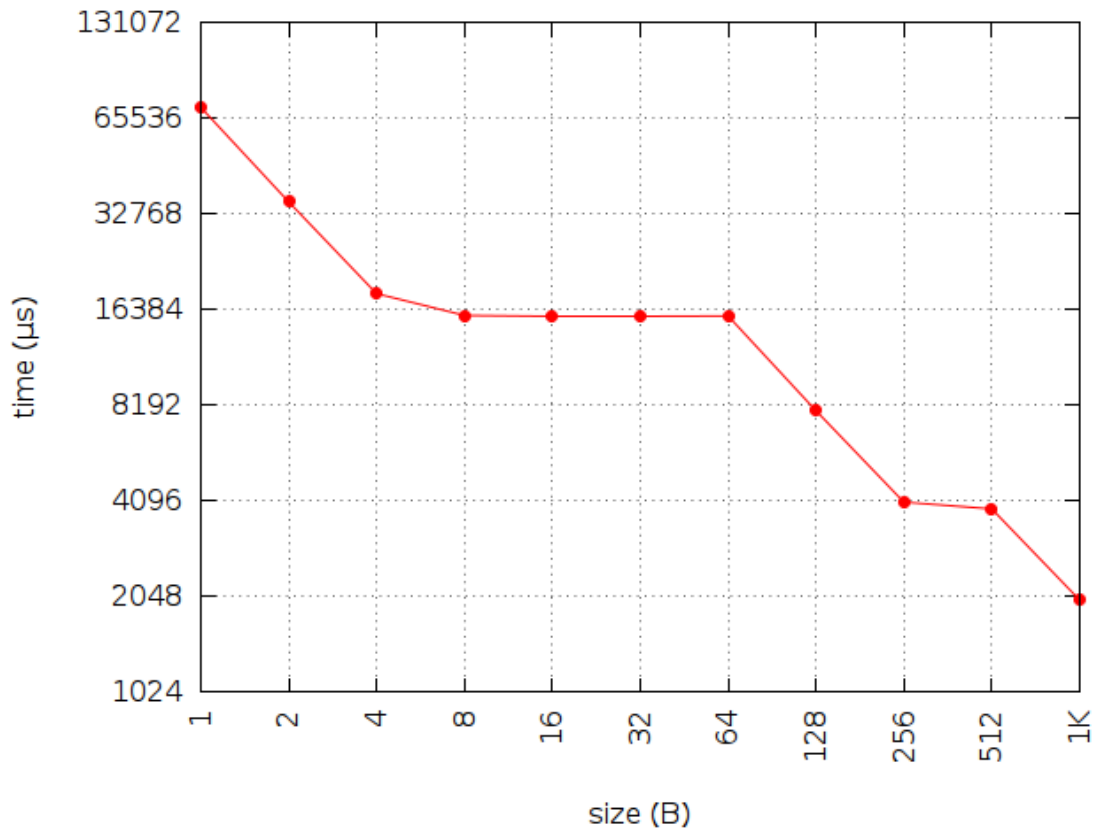
processor    : 3
model name  : ARMv7 Processor rev 4 (v7l)
BogoMIPS    : 38.40
Features     : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part    : 0xd03
CPU revision: 4

Hardware     : BCM2709
Revision     : a02082
Serial       : 00000000551be89e
```

3.1. Averiguando tamaño de línea/vía.

Tuve que compilar los archivos en la propia máquina ya que tienen una arquitectura distinta y el ejecutable compilado en nuestro ordenador no funciona en la raspberry. De igual forma hay que cambiar el makefile. La orden para compilar ha de ser sin el flag mach, o en caso contrario usar -mcpu=cortex-a53.

La gráfica obtenida es la siguiente:



Como podemos ver a partir de 64 bits el tiempo baja, esto es debido a que se producirán muchos menos errores en el acceso a los datos por lo que podemos afirmar que el tamaño máximo de línea es de 64B.

3.2. Averiguando tamaño de las cachés.

El procesador Cortex A53 tiene dos niveles de caché, L1 y L2, donde la L1 es de 32K y la L2 de 512K. Como podemos ver en la grafica, el tiempo de acceso para la L1 es insignificante, en cambio para la L2 crece donde su punto más alto se ajusta el propio tamaño de la caché 512K. A partir de ahí lo que se hace es acceder a la memoria principal que es de 1GB, si el vector hubiese superado este tamaño veriamos que se accede a disco que en este caso sería una tarjeta micro SD donde el tiempo de acceso se dispararía ya que no llega a tener integrado un disco sólido sino que el disco es otro periférico.

La imagen de los resultados la podemos ver a continuación:

