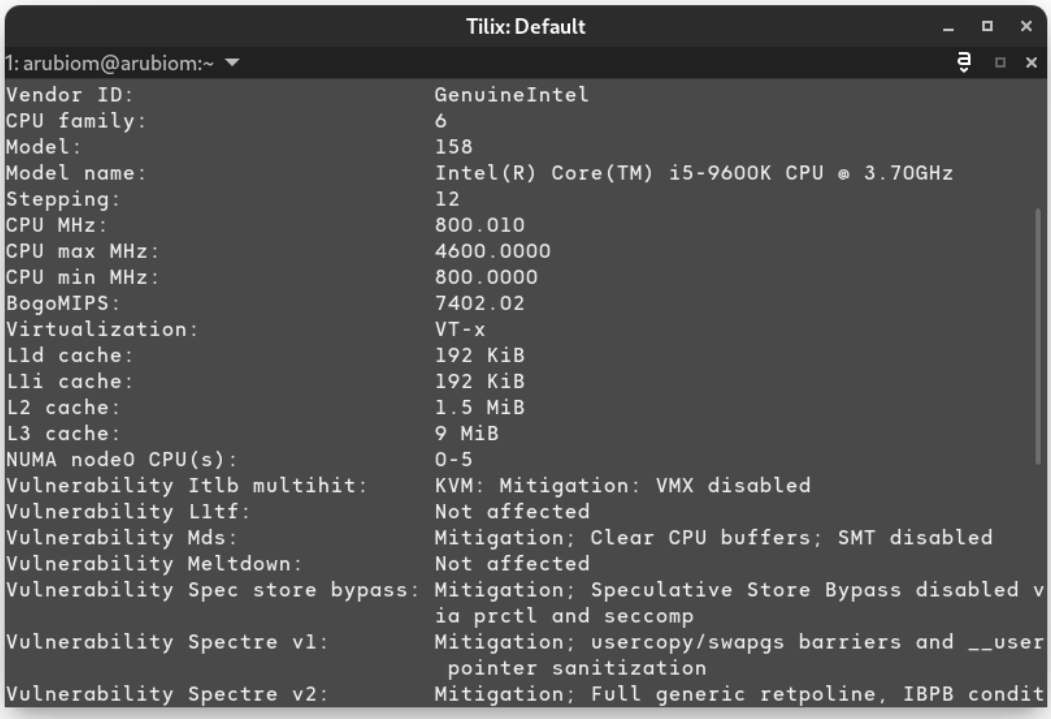
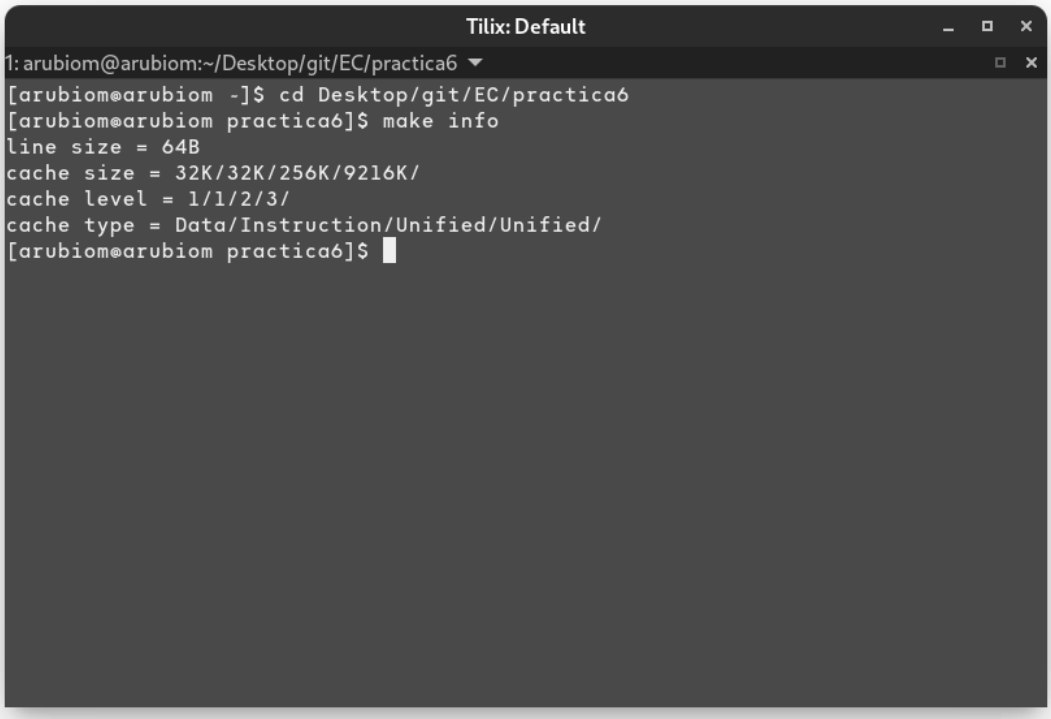
Práctica memoria caché Alejandro Rubio Martínez

Para empezar la práctica primero empezamos visualizando la información que nos da la orden `lscpu`:



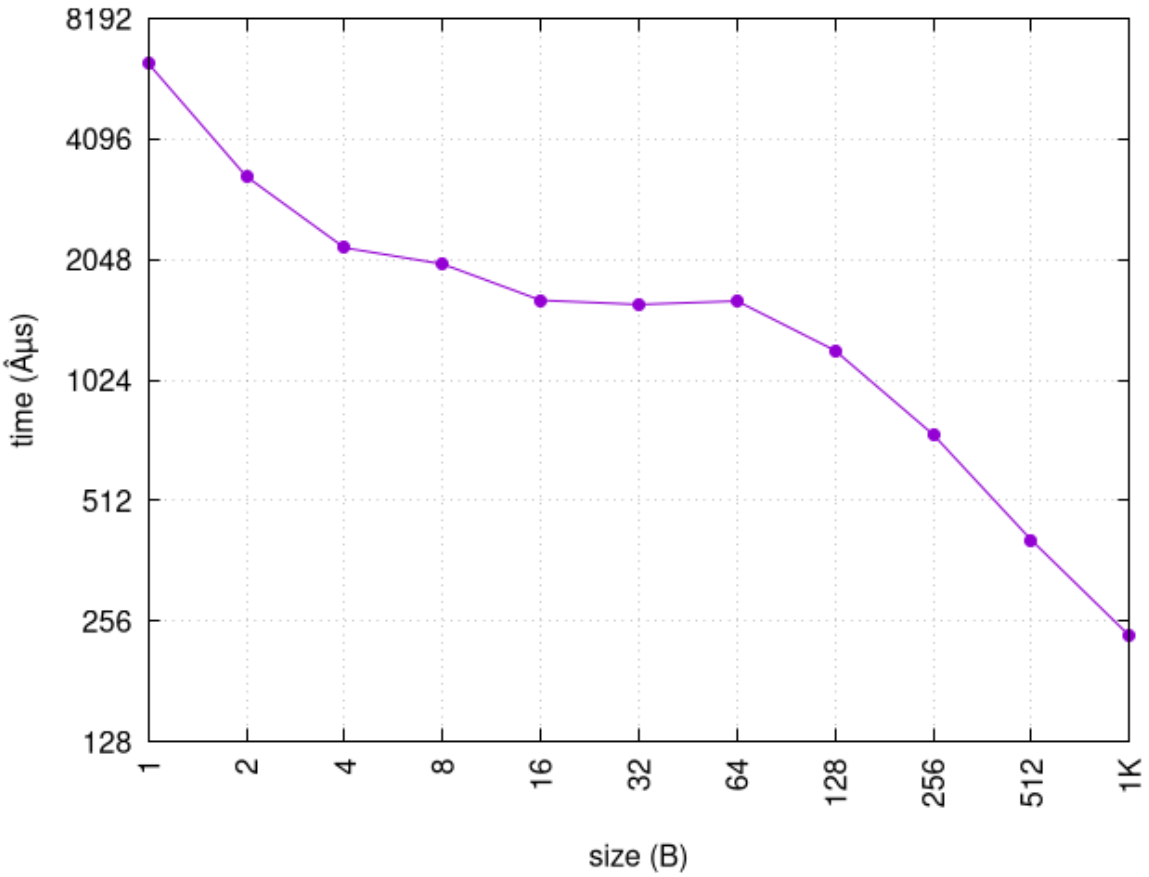


Podemos ver las caches de tipo Lld, LLi, L2 y L3 de 192 KiB,192 Kib, 1.5 MiB y 9 MiB respectivamente.

Primero vamos a comprobar el tamaño de líneas. Para ello utilizando el esqueleto de line.cc en SWAD tan solo buscamos realizar una pequeña modificación al vector, por ejemplo, un XOR con 1.

```
bytes[i] ^= 1;
```

Como tenemos un bucle que hace que cada vez vayamos recorriendo el vector `bytes` en saltos más grandes, ahora lo que hacemos es representar el tiempo que se tarda dependiendo de la longitud de esos saltos, y utilizando el Makefile me queda un gráfica:

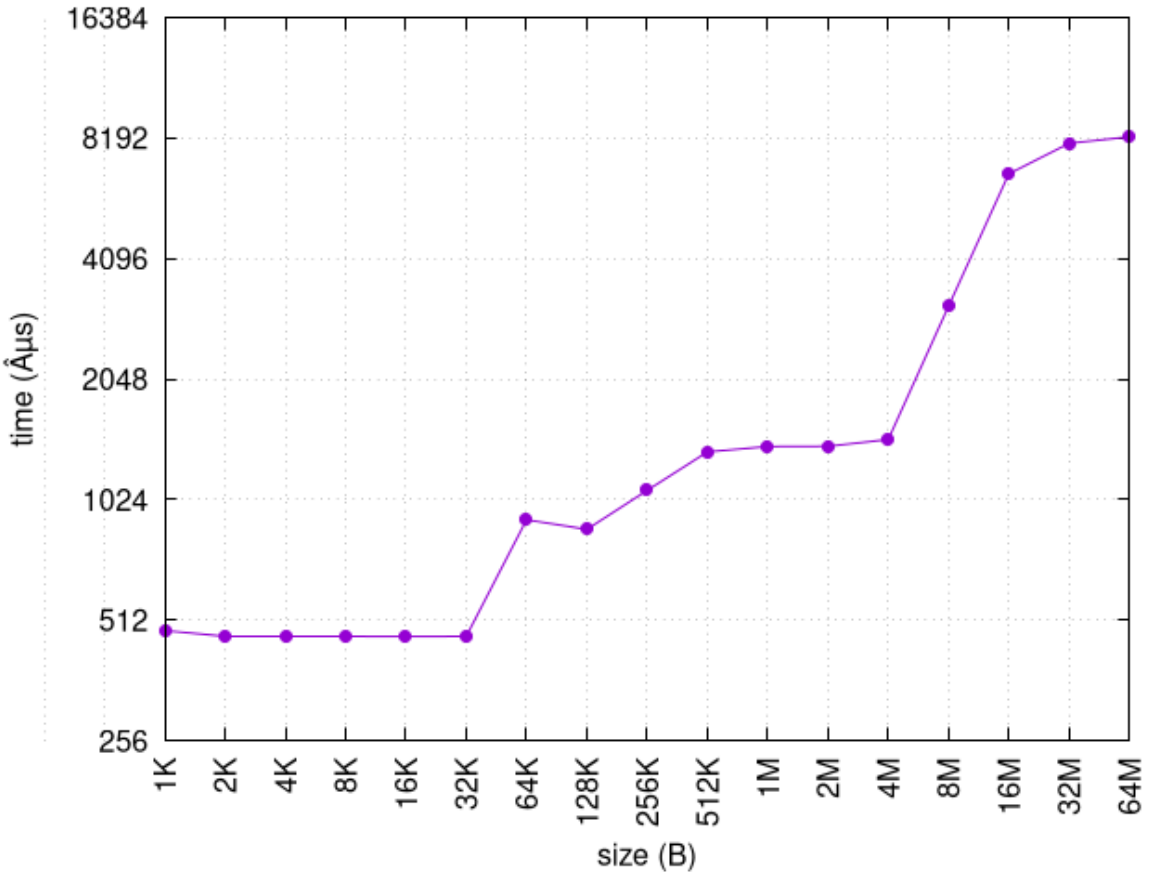


Al principio esperaba que la gráfica fuera proporcional a los saltos, es decir, que como cada anchura de los saltos era el doble que la anterior, el tiempo fuera disminuyendo a la mitad, pero en la gráfica es fácil comprobar que esto no es así. Mis sospechas apuntan a que esto es debido a la velocidad no lineal de la caché debido al uso de localidad espacial ("si un elemento es referenciado los elementos cercanos a este se van a referenciar pronto") y a la eficiencia tampoco lineal del `operator[]` a vectores.

Ahora vamos con el tamaño para ello reutilizando el esqueleto de size.cc que hay en SWAD. Ahora queremos recorrer la caché, y para ello el primer impulso es hacerlo de la manera que se viene usando siempre utilizando el `operator[]` con un iterador i. Pero tras su compilación y ejecución obtengo un `segmentation fault`. Entonces investigo que está pasando y veo que el recorrido una caché puede llegar a tardar más de 20 minutos, entonces busco una nueva forma de iterar el vector. Para ello lo que hago es :

```
bytes[(i*64)&(size-1)] ^= 1
```

donde i es el iterador. El 64 es debido a la longitud de palabra que se puede ver utilizando `make info` y luego utilizando el `&(size-1)` para asegurarnos que nunca nos salgamos de la longitud de palabra y no volver a obetener el `segmentation fault`. Veamos la gráfica que obtengo:



Aquí podemos ver como aproximadamente se corresponden los saltos abruptos en la gráfica con los tamaños de la caché obtenidos en `make info`.

Bibliografía

https://swad.ugr.es/swad/tmp/8A/lVAlB-bMZ9FRkKeD7QvDkCszujsxkHtakX7-VA9BQ/Practica%206%20Guion.pdf

https://youtu.be/UfeTFsYmB-s

https://stackoverflow.com/