Prácticas Experimentales de Memorias Cache

Julio Sahuquillo, José Flich y Jorge Real
Departamento de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera, 14. 46021 Valencia
e-mail: {jsahuqui, jflich, jorge}@disca.upv.es

Resumen

El conocimiento de las memorias cache se considera básico e imprescindible en la formación de cualquier titulado universitario (ya sea técnico o superior) en Informática.

Su estudio se puede abordar desde distintos puntos de vista. Por una parte, desde un punto de vista teórico describiendo su funcionamiento básico: cómo se localiza un bloque almacenado, qué bloque debe reemplazarse, etc. Por otra parte, desde un punto de vista práctico, comprobando el funcionamiento estudiado, normalmente mediante un simulador.

Este artículo se enmarca dentro del punto de vista práctico y propone el uso de prácticas experimentales para el estudio de las memorias cache como complemento a los simuladores. Se trata de utilizar un sencillo programa de medida de tiempos de acceso al sistema de memoria, para comprobar experimentalmente el efecto del sistema de cache sobre las prestaciones de un computador personal. Mediante la realización de las prácticas el alumno deduce cuestiones como: ¿cuántos niveles de cache tiene el computador?, ¿cuántas vías tiene cada uno de ellos?, ¿cuál es su tamaño de bloque?, etc. Estas prácticas se han realizado por primera vez durante el presente curso académico en la Escuela Técnica Superior de Informática Aplicada y en la Facultad de Informática de la Universidad Politécnica de Valencia con un buen grado de aceptación tanto por parte del profesorado como del alumnado.

1. Introducción

Las memorias cache ejercen un papel de crucial importancia en las prestaciones globales del procesador. De hecho, hoy en día es insostenible el diseño de un procesador de propósito general o de altas prestaciones sin incorporar al menos dos niveles de memorias cache.

Aunque en los últimos lustros una gran cantidad de trabajos se orientaron a la propuesta de nuevas organizaciones de cache [1] con miras a aumentar su eficiencia, en esencia, su organización interna apenas ha variado desde sus orígenes.

Este motivo supone que apenas existan diferencias entre los contenidos teóricos de organización y funcionamiento que incorporan los planes de estudio de las distintas universidades españolas, europeas o de cualquier otro país. Estos contenidos en general se cubren en asignaturas de los primeros cursos de carrera. Así, en la Escuela Técnica Superior de Informática (ETSIA) de la Universidad Politécnica de Valencia (UPV) se cubren en la asignatura Estructura y Tecnología de Computadores II (ETC2), anual de segundo curso, y en la Facultad de Informática (FI) de la misma universidad se cubren en la asignatura Estructura de Computadores (EC), también anual de segundo curso. Para el estudio de estos contenidos se pueden utilizar libros de texto como [2], [3].

Aspectos más avanzados sobre las memorias cache, como pueden ser las caches libres de bloqueo, o protocolos de coherencia de cache, se tratan en asignaturas de cursos más avanzados, como es el caso de la asignatura Arquitectura de Computadores, de cuarto curso de la FI de la UPV.

El hecho de que los contenidos teóricos apenas difieran en distintos centros, no significa que los contenidos prácticos sean también similares. Hoy en día, en general, se suelen seguir dos alternativas: no realizar prácticas de memoria cache, o utilizar un simulador de su funcionamiento para realizarlas.

En líneas generales estos simuladores utilizan como parámetros de entrada las características de la cache (tamaño de la cache, tamaño de bloque, correspondencia y algoritmo de reemplazo) y una pequeña traza con instrucciones de referencia a memoria (lectura o escritura) junto con la dirección de memoria correspondiente. El objetivo del simulador es observar cómo se van reemplazando unos bloques a otros y calcular la tasa de aciertos. Estos simuladores pueden ser de gran ayuda de cara a que el alumno pueda por sí mismo realizar problemas y comparar los resultados.

Las prácticas que se proponen en este artículo se orientan a estudiar las prestaciones de las cache de un computador de manera experimental y comprobar el determinante papel que estas ejercen sobre las prestaciones globales del computador. A partir de los tiempos de acceso observados en el direccionamiento repetitivo de los elementos de un vector, se pueden deducir características como cuántos niveles de cache tiene el computador, el tamaño de las caches, el número de vías y el tamaño del bloque. Estas prácticas añaden un pequeño grado de complejidad respecto a las que se realizan con simuladores típicos, por lo que es necesario preparar una práctica muy guiada en cuanto a la secuencia de acciones a realizar y que ayude a la interpretación de unos resultados gráficos que, a primera vista, no son evidentes.

Cabe resaltar que estas prácticas son complementarias de las que puedan realizarse con simuladores del comportamiento de la cache.

El resto de este artículo se organiza como sigue. En la sección 2 se describe la motivación que animó a los autores a prepara un práctica de estas características. En la sección 3 se describe el concepto de *stride*, necesario para poder realizar la práctica. En la sección 4 se presenta el programa de test utilizado. La sección 5 describe la parte experimental a desarrollar por los alumnos, mientras que en la 6 se indica cómo interpretar los resultados obtenidos. Por último, en la sección 7 se presentan las conclusiones de esta nueva experiencia en el presente curso académico.

2. Motivación

Las memorias cache se incorporaron en los computadores con el objetivo de reducir la latencia de acceso a la información (datos e instrucciones). Cuando un procesador intenta acceder a un dato (o instrucción) sólo realiza el acceso físico a memoria en el caso de que el bloque que lo contiene no se encuentre almacenado en la cache. Si el procesador dispone de dos niveles de cache (que es lo usual) se busca primero en la de nivel uno, que es la más cercana al procesador y trabaja, en general, con un tiempo de acceso de uno o dos ciclos del procesador. En el caso de que la información buscada no se encuentre en la cache de nivel uno, se accede a la cache de nivel dos bastante más lenta (este acceso se realiza en paralelo con el acceso al nivel 1 en muchos procesadores). Sólo en el caso de que el acceso a ambas caches resulte en fallo, se accede a la memoria principal.

La penalización que ejercen los fallos de cache sobre el tiempo medio de acceso a los datos, y en consecuencia en las prestaciones globales del sistema, es realmente acuciante y según las predicciones tecnológicas, la gran diferencia de tiempos entre la velocidad del procesador y la velocidad de memoria continuará creciendo. Esto es debido en parte a que la tecnología empleada en la construcción de la memoria principal es de por sí mucho más lenta que la empleada en la construcción del procesador y las caches.

Los usuarios con frecuencia conocen el modelo de su procesador y la frecuencia con la que trabaja, por ejemplo, Pentium 4 a 2.4 GHz; la capacidad de memoria principal instalada y su tipo, por ejemplo 256 MB de DDR; la capacidad de su disco duro, etc. Sin embargo, la mayoría desconocen las características de la memoria cache, a pesar de su gran importancia en las prestaciones.

Los argumentos descritos, nos llevaron a diseñar una propuesta de prácticas orientadas a comprobar experimentalmente las características del sistema de cache, ya que consideramos de vital importancia en la formación de un Ingeniero en Informática ejercitar sus habilidades también sobre el *hardware* y no sólo con simuladores.

Este tipo de prácticas ya se están realizando en algunas universidades americanas de primer nivel como la de Berkeley bajo la dirección de David Patterson [5]. De hecho, el código fuente del programa que se utiliza para la realización de la práctica se encuentra disponible en el texto de Hennessy y Patterson [4]. Nuestra aportación ha sido la preparación íntegra paso a paso con el fin

de que el alumno no sólo pudiese ejecutar el programa de medida de tiempos de acceso, sino también entender sus resultados y extraer sus propias conclusiones. Con este fin, se dedica un tiempo de las clases de problemas de aula a realizar ejercicios de preparación para la práctica. Asimismo la práctica integra un conjunto de ejercicios destinados a consolidar los fundamentos teóricos y a facilitar la interpretación de los resultados experimentales.

3. Concepto de *stride*: mínimo y máximo

El programa de test que se utiliza para comprobar las características de la cache, accede de manera iterativa a los elementos de un vector de tamaño variable.

Si se accede a todos los elementos del vector de uno en uno (0, 1, 2, 3, 4, 5, 6, 7, etc) se dice que el vector se recorre con *stride* o distancia 1. Si se accede a los elementos del vector uno sí uno no (0, 2, 4, 6, 8, 10, etc) se dice que se recorre con *stride* 2. Si se accede uno sí tres no (0, 4, 8, etc) se recorre con *stride* 4 y así sucesivamente. En la Figura 1 se muestra un vector de 32 elementos donde aparecen sombreados los que serán accedidos para distintos valores del *stride*. El *stride* máximo es la mitad de los elementos del vector, y el *stride* mínimo es uno. En el ejemplo, el *stride* máximo es 16 (32/2) donde se accede sólo a dos elementos (0 y 16).

En la práctica se proponen una serie de ejercicios para que el alumno ejercite el concepto de *stride*, como base para ser capaces de analizar e interpretar los resultados. Por tanto, la secuencia obligatoria debe ser realizar los ejercicios antes de la práctica experimental propiamente dicha. Aunque ya se han realizado ejercicios en clases de problemas, dada la relativa complejidad de la práctica, es aconsejable realizar más ejercicios de cara a fomentar una sólida base común en los alumnos de los distintos grupos.

3.1. Ejercicios sobre *stride* de apoyo a la interpretación de resultados

En esta sección se presenta un subconjunto de los ejercicios que debe realizar el alumno antes de pasar a la parte experimental de la práctica.

Stride	Elementos del vector																														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
2	**0**	1	**2**	3	**4**	5	**6**	7	**8**	9	**10**	11	**12**	13	**14**	15	**16**	17	**18**	19	**20**	21	**22**	23	**24**	25	**26**	27	**28**	29	**30**	31
4	**0**	1	2	3	**4**	5	6	7	**8**	9	10	11	**12**	13	14	15	**16**	17	18	19	**20**	21	22	23	**24**	25	26	27	**28**	29	30	31
8	**0**	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	**16**	17	18	19	20	21	22	23	**24**	25	26	27	28	29	30	31
16	**0**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**16**	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figura 1. Ejemplo de un vector de 32 elementos accedido con distintos valores de stride.

Ejercicios básicos de *stride*:

Suponga un vector de enteros de 4096 bytes (1024 elementos de 4 bytes).

- ¿Qué elementos (palabras) se referenciarán si se accede con *stride* 256?
- ¿Cuál es el *stride* máximo con que se puede acceder al vector?
- Si se sabe que el máximo *stride* con que se puede acceder a un vector es 2048, ¿cuál es el número de elementos del vector? ¿Y el tamaño del vector en bytes?
- Si se accede a un vector de 64 KB con el máximo *stride* posible, ¿a cuántos elementos distintos (palabras) se referencia? ¿Y a cuántos bloques?
- Y si se accede a dicho vector con el *stride* anterior al máximo (notación máximo-1), ¿a cuántos elementos distintos (palabras) se referencia? ¿Y a cuántos bloques?

Ejercicios de *stride* máximo (determinación del número de vías):

Suponga una cache de 16KB, 2 vías, bloques de 16 bytes, y algoritmo de reemplazo LRU. Suponga asimismo que se accede a un vector de 32 KB (8K elementos de 4 bytes). Si se accede con el máximo *stride* (2 elementos) se accedería a los elementos 0 y 4 K (4096), pero las direcciones de palabra correspondiente (las direcciones de palabra son múltiplos de 4) serían la 0 y la $4096 \times 4 = 2^{14}$. Indique en la siguiente tabla la dirección de los mismos desglosada en etiqueta, conjunto y desplazamiento, asumiendo que el vector empieza a almacenarse a partir de la dirección cero.

Elemento referenciado (dirección de palabra)	Dirección (en bytes)		
	Etiqueta	Conjunto	despl.
0 (dir. palabra 0)			
4096 (dir. palabra 4096×4)			
	31 ... 13	12 ... 4	3 ... 0

- ¿Van al mismo conjunto?
- Si se accede en un bucle relativamente grande (5000 iteraciones) al vector con *stride máximo*, ¿cuántos fallos y cuántos aciertos se producirán en la segunda y sucesivas iteraciones?
- Indique la dirección de los elementos accedidos si se accede con el *stride* anterior al máximo (se utilizará la notación *máximo-1*).

Elemento referenciado (dirección de palabra)	Dirección (en bytes)		
	Etiqueta	Conjunto	despl.
	31 …13	12 … 4	3 …0

- ¿Van al mismo conjunto?
- Si se accede en un bucle relativamente grande (5000 iteraciones) al vector con *stride máximo-1*, ¿cuántos fallos y cuántos aciertos se producirán en la segunda y sucesivas iteraciones?

El siguiente cuadro resume al alumno las principales conclusiones de este ejercicio.

Se puede observar que, si el tamaño del vector es múltiplo en una potencia exacta de dos del tamaño de la cache (32KB, 64KB, 128KB, 256KB, etc.), sucederá lo mismo si se accede con *strides* máximo y máximo-1.

- Si la cache tuviese 4 vías manteniendo el resto de parámetros iguales que en el ejercicio anterior, ¿cuántos fallos y cuántos aciertos se habrían producido si se hubiese accedido con *stride* máximo en la segunda y resto de iteraciones del bucle?
- ¿Y con *stride máximo-1*?
- ¿Y si se hubiese accedido con *stride máximo-2*?

Accediendo al vector con valores decrecientes de *stride*, el valor de *stride* que precede al primero que alcanza una tasa de fallos de aproximada-

mente el 100%, permite determinar el número de vías.

Ejercicios de *stride* mínimo (determinación del número de palabras del bloque):

Suponga la misma cache y el mismo vector que en el ejercicio anterior, y que se accede al vector con *stride* 1. Indique la dirección de los 4 primeros elementos a los que se accedería. Para ello se le ofrece al alumno una tabla auxiliar de manera análoga a los ejercicios anteriores.

- ¿Pertenecen todos al mismo bloque?
- En el peor de los casos, es decir, suponiendo que siempre que se accede al primer elemento del bloque se produjese un fallo, ¿cuál sería la tasa de aciertos si se accediera con *stride* 1? ¿Y si se accediese con *stride* 2? ¿Y si se accediese con *stride* 4?

Luego, el primer *stride* que produce una tasa de aciertos del 0% determina el tamaño del bloque en palabras.

4. Programa de test

Para realizar la práctica se ofrecen dos ficheros: *benchmark.c* y *grafico.config*:

Benchmark.c es un pequeño programa de test cuyo bucle principal accede a los elementos de un vector. La Figura 2 muestra el código de este programa, resaltando el bucle principal. Los accesos a los elementos del vector se realizan con la instrucción: `x[index]=x[index]+1;` donde se realiza una lectura (*load*) para leer el operando `x[index]` y poder realizar la suma, y otro acceso para escribir el resultado (*store*). La suma de ambos tiempos (lectura+escritura) es lo que se mide en el programa. Para eliminar la sobrecarga impuesta por las instrucciones de control del bucle principal, se utiliza la técnica del doble bucle para restarle su efecto. El bucle que cuantifica la sobrecarga aparece a continuación del bucle principal.

Grafico.config es un pequeño *script* que se ofrece para formatear el gráfico de salida.

5. Parte experimental

Esta parte consiste en la compilación y ejecución del programa *benchmark.c* sobre un determinado ordenador y la visualización de los resultados utilizando para ello la popular herramienta *gnuplot* sobre el sistema operativo Linux. El laboratorio donde se realizan las prácticas está dotado de computadores de tres clases distintas, por lo que los resultados obtenidos dependen del puesto de trabajo donde se realice la práctica.

Tras la obtención experimental de los resultados, el alumno deberá realizar su interpretación, lo que confirmará si ha cubierto o no el objetivo de la práctica.

6. Análisis de Resultados Experimentales

El laboratorio en el que se realiza la práctica se encuentra equipado con tres tipos de ordenadores personales:

- Pentium II 350MHz
- AMD K6-2 450MHz
- AMD Athlon 700 MHz

La Figura 3 muestra como ejemplo el tiempo medio de acceso para distintos tamaños de vectores en el Pentium II. El alumno debe deducir a partir de los resultados cuestiones relativas a la cache de nivel 1, a la cache de nivel 2 y a la memoria principal.

6.1. Cuestiones relativas a la cache de nivel 1

Si se sabe que la cache de datos del Pentium II de nivel 1 es de 16KB, la del K6 de 32KB y la del Athlon de 64KB, responda a las siguientes preguntas:

- ¿Qué tipo de procesador se encuentra instalado en su computador?
- ¿Cuántas vías tiene la cache?
- ¿Cuál es el tamaño del bloque en palabras?
- ¿Cuál es el tiempo aproximado de cada acceso (lectura+escritura) a la cache?

```
#include <stdio.h>
#include <sys/times.h>
#include <sys/types.h>
#include <time.h>
#define CACHE_MIN (1024)        /* cache más pequeña */
#define CACHE_MAX (1024*1024)   /* cache más grande  */
#define SAMPLE 10
#define CLOCK_TCK 60.0     /* número de ticks de reloj por ciclo */
int x[CACHE_MAX];

double get_seconds() {    /* rutina para leer el tiempo */
        struct tms rusage;
        times(&rusage);    /* utilidad UNIX para medir el tiempo */
        return (double) (rusage.tms_utime)/CLOCK_TCK;
}

int main() {
int register i, index, stride, limit, temp;
int steps, tsteps, vsize;
float sec0, sec, den;

for (vsize=CACHE_MIN; vsize <= CACHE_MAX; vsize=vsize*2)
  {
    for (stride=1; stride <= vsize/2; stride=stride*2)
      {
        sec = 0;
        limit = vsize - stride + 1;  /* tamaño de vector para este bucle */
        steps = 0;

        do {                    /* repetir durante un segundo */
          sec0 = get_seconds(); /* instante de tiempo antes del bucle */
          for (i=SAMPLE*stride; i !=0; i=i-1)
            for (index=0; index<limit; index=index+stride)
              x[index] = x[index]+1;   /* bucle de accesos a cache = lectura + escritura */

          steps = steps+1;                    /* contador de iteraciones del bucle while */
          sec = sec+ (get_seconds() - sec0);   /* acumula tiempo transcurrido en el bucle */
        } while (sec < 1.0);

        /* repetir un bucle vacío para restar la sobrecarga */
        tsteps = 0;   /* variable auxiliar para realizar el mismo numero de iteraciones */
        do {
          sec0 = get_seconds();
          for (i=SAMPLE*stride; i !=0; i=i-1)
            for (index=0; index<limit; index=index+stride)
              { temp = temp+index;  }  /* dummy code */

          tsteps = tsteps+1;
          sec = sec - (get_seconds() - sec0);
        } while (tsteps < steps);

        den = steps*SAMPLE*stride;
        den = den*((limit-1)/stride+1);
        printf("Tamaño: %4d KB Stride: %7d T(lec+esc): %5.0f ns\n",
                        vsize*sizeof(int)/1024, stride, sec*1e9/den);

        /* printf(" %4d %7d  %5.0f \n", vsize*sizeof(int)/1024 , stride, sec*1e9/den); */
      }
    printf("    0     0      0 \n");  /*linea auxiliar para la presentación de resultados */
  }
}
```

Figura 2. Programa benchmark.c.

6.2. Cuestiones relativas a la cache de nivel 2

Los procesadores en los que se encuentra trabajando incorporan una cache de nivel 2 unificada para instrucciones y datos, por lo que ambos compiten por un espacio en dicha cache. Esto significa, que no se producirá un cambio brusco en el tiempo de acceso sino que habrá un cambio paulatino. El tamaño de la cache de nivel 2, lo define el tamaño del vector previo a la estabilización del tiempo de acceso para la memoria principal.

- ¿Cuál es el tamaño de la cache en KB?
- ¿Cuál es el tiempo aproximado de cada acceso (lectura+escritura)?
- ¿Cuántas veces más grande es el retardo de la cache de nivel 2 que el de la de nivel 1?

6.3. Cuestiones sobre la memoria principal

- ¿Cuál es el tiempo aproximado de cada acceso (lectura+escritura) en nanosegundos?
- ¿Cuántas veces más grande es el retardo de memoria respecto a la de nivel 1?
- ¿Qué indican los picos en los tiempos de acceso?

7. Conclusiones

Es aconsejable realizar prácticas sobre memorias cache para afianzar los conocimientos teóricos. En la mayoría de las universidades estas prácticas se realizan mediante un simulador, de gran utilidad para afianzar los conceptos básicos como determinar el conjunto donde se almacenará el bloque, aplicación de los algoritmos de reemplazo, etc.

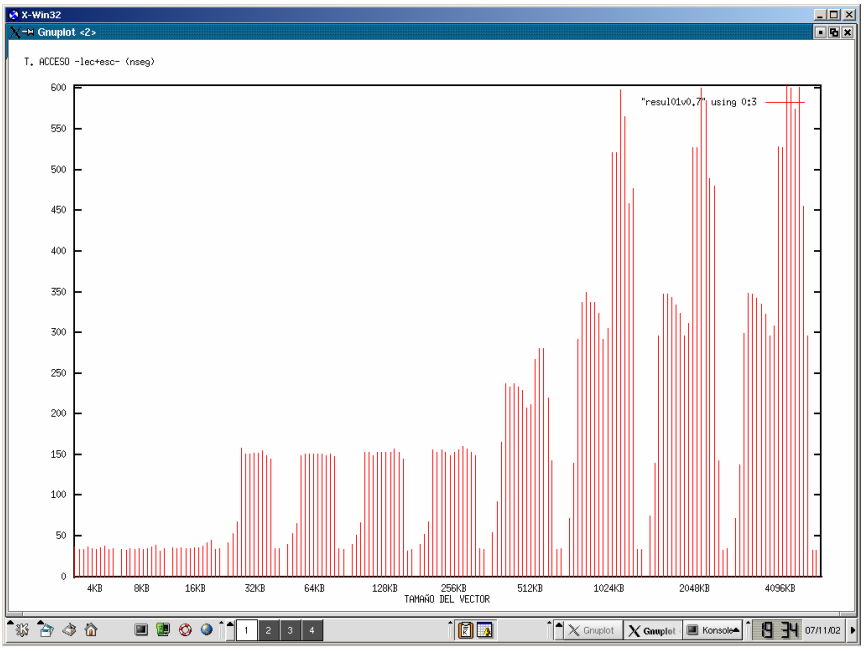

Figura 3. Resultados obtenidos sobre el Pentium II.

En este trabajo se ha presentado una práctica complementaria a las anteriores, con la peculiaridad de ser totalmente experimental ya que se realiza sobre la propia memoria cache que integra el procesador.

Como principales ventajas ofrecidas por la práctica, cabría resaltar:

- Se realiza directamente sobre el *hardware* del PC sin la utilización de simuladores.
- Permite comprobar la necesidad de la memoria cache y su importante papel en las prestaciones, "escondiendo" la latencia a memoria.
- Cuantifica (aproximadamente) los retardos de la cache, tanto de nivel 1 como de nivel 2.
- Permite comprender por qué con los retardos actuales (procesadores del laboratorio de prácticas) no es necesaria una cache de nivel 3.
- Se aprende a deducir experimentalmente las características (tamaño, nº de vías, y tamaño del bloque) de la cache de nivel 1.

Como principal inconveniente habría que mencionar que la práctica incorpora cierto grado de dificultad por lo que es necesario, dedicar una clase de teoría a explicar conceptos que ayuden al alumno a la interpretación de los resultados de la práctica.

Por último, cabe mencionar que la práctica se ha realizado por primera vez durante el presente curso, siguiendo las directrices descritas en este trabajo, y se ha podido comprobar que el alumno muestra cierto grado de entusiasmo cuando logra resolver y comprender los resultados experimentales, lo cual lo han podido conseguir un porcentaje relativamente elevado de ellos. Además, muchos han podido realizar el mismo experimento sobre sus propios ordenadores personales, lo cual es un factor adicional de motivación.

Referencias

[1] J. Sahuquillo and A. Pont, "Splitting the Data Cache: A Survey," July-September 2000 special issue of the IEEE Concurrency. ISSN1092-3063/00.

[2] D. A. Patterson, J. L. Hennessy, Estructura y Diseño de Computadores: interficie circuitería programación, Reverté, S.A. 2000 (2ª edición).

[3] V. C. Hamacher, Z. G. Vranesic, S. G. Zaky, Computer Organization McGraw-Hill, 2002 (5ª edición)

[4] J. L. Hennessy, D. A. Patterson, Computer Architecture. A quantitative approach, Morgan Kauffmann, 2002 (3ª edición.

[5] http://inst.eecs.berkeley.edu/~cs61c/fa01/calendar/week13/lab10.