Informe Práctica
LAB2
Arquitectura de Computadores

Ramon Guimerà Ortuño 1400214
Ibai Zak Allan Aldanondo 1397750
Grupo: 419
Fecha: 15/04/2016
Departamento: CAOS

Indice

Indice de tablas y códigos

Objetivo de la práctica:

La práctica 2 de laboratorio tiene como objetivo medir el rendimiento de diversos programas en lo que se refiere a tiempos de ejecución y, sobretodo, cantidad de accesos a memoria cache y que efectos producen estos accesos sobre la ejecución del programa.

Ejercicios:

Notas de la práctica:

G = 10^9

Frecuencia del procesador: 3.33GHz

Ejercicio 1A:

a) Fusión de bucles → Reduce todas las instrucciones relacionadas con memoria. De 4 floats + 5 loads + 2 store pasa a 3 float + 1 load + 0 store.

b) Instrucciones de bucles → Reducción de misses en memoria cache.

c) Transformación de operaciones → Reducción de lecturas en memoria.

Ejercicio 1B:

Se ejecutan 21 instrucciones en el bucle interno.

Problem Size		Inst. Count (G)	Iteraciones Bucle Interno (G)	Tiempo Bucle interno
N	M			
1000	100000000	139,2	6,63	15,086
10000	10000000	132,1	6,29	15,897
100000	1000000	131,4	6,26	15,982
1000000	100000	131,3	6,25	15,994
10000000	10000	131,5	6,26	15,970
100000000	1000	133,5	6,36	15,730
1000000000	100	152,9	7,28	13,734

Tabla ejercicio 1B

Ejercicio 1C:

La optimización que realiza el compilador es una vectorización, coge las operaciones vectoriales y las hace de 4 en 4. Además, hace un desenroll del bucle externo 2 veces.

Ejercicio 1D:

Resultados obtenidos y calculados en el ejercicio 1D:

N	M	Cycles (G)	Instructions (G)	IPC	Time (s)
1000	100000000	54,8	139,2	2,540145985	16,44
10000	10000000	50,8	132,1	2,600393701	15,24
100000	1000000	51,9	131,4	2,531791908	15,57
1000000	100000	52	131,3	2,525	15,6
10000000	10000	72,1	131,5	1,823855756	21,63
100000000	1000	85,9	133,5	1,554132712	25,77
1000000000	100	97,2	152,9	1,573045267	29,16

Tabla ejercicio 1D

A mayor número de M, menor el IPC y menor el número de ciclos y por el contrario, a mayor número de N, menor IPC y mayor el número de ciclos.
Hay más instrucciones en el bucle M con accesos a memoria que en el bucle N.
El hecho de que el IPC sea menor, provoca un mayor tiempo de ejecución.

Ejercicio 1E:

Se adjunta la tabla de resultados obtenida para el ejercicio 1E:

N	M	V1 (bytes)	V2 (bytes)	R (bytes)	X (bytes)	Cache misses
1000	100000000	400000000	400000000	400000000	4000	19383010
10000	10000000	40000000	40000000	40000000	40000	1975050
100000	1000000	4000000	4000000	4000000	400000	243401
1000000	100000	400000	400000	400000	4000000	101434
10000000	10000	40000	40000	40000	40000000	165327378
100000000	1000	4000	4000	4000	400000000	167090497
1000000000	100	400	400	400	4000000000	242484343

Tabla ejercicio 1E

Comparando los resultados obtenidos con las tablas 1 y 2 podemos deducir que, a un número parecido de N y M, es decir, las mismas iteraciones de bucle interno que externo es cuando obtenemos menos fallos en la cache y por lo tanto, menos tiempo de ejecución y menos iteraciones del bucle interno.

Ejercicio 1F:

Resultados obtenidos del ejercicio 1F:

BLK	N	M	Time (s)	Inst.	Cycles	IPC	Cache misses
1000000	10000000	10000	21,738	162793444175	73275428226	2,21	722036
1000000	100000000	1000	22,06	164733651059	74369174535	2,22	6734100
1000000	1000000000	100	25,51	184171221808	86021789449	2,14	67508218
10000	10000000	10000	20,1	16323935102	6826815804	2,39	696372
10000	100000000	1000	20,59	16518610138	69418323118	2,38	6857949
10000	1000000000	100	24,1	18460098280	8111834985	2,28	73330280

Tabla ejercicio 1F

Aprovechando la comparación realizada en el ejercicio anterior, podemos observar que estamos en lo correcto, a un número parecido entre N y M, menor son los fallos en cache y menor el tiempo de ejecución.

Ejercicio 1G:

Resultados obtenidos del ejercicio 1G:

BLK	N	M	Time (s)	Inst.	Cycles	IPC	Cache misses
1000000	10000000	10000	15,81	144019080054	53274006276	2,7	665987
1000000	100000000	1000	16,17	145961671634	54503347649	2,68	6460143
1000000	1000000000	100	19,46	165397106594	65606972496	2,52	64467602
10000	10000000	10000	15,6	144077555705	52605871529	2,74	662483
10000	100000000	1000	16,73	146020890021	54374133548	2,69	6697862
10000	1000000000	100	19,57	165454233065	65941184173	2,51	68859958

Tabla ejercicio 1G

Los resultados obtenidos mejoran entre 5 y 6 segundos de ejecución, debido entre otras cosas, a la reducción de fallos de cache. Se adjunta el codigo siguiente, el cuál es una optimización sobre el código del ejercicio 1F.

Código del ejercicio 1G:

```
for (j=0; j<M; j += 4)
{
	R[j] = V1[j]*V2[j]*N;
	R[j+1] = V1[j+1]*V2[j+1]*N;
	R[j+2] = V1[j+2]*V2[j+2]*N;
	R[j+3] = V1[j+3]*V2[j+3]*N;
}

for (k=0; k<N; k+=BLK)
	for (j=0; j<M; j+=4)
	{
		float Rj0 = R[j];
		float Rj1 = R[j+1];
		float Rj2 = R[j+2];
		float Rj3 = R[j+3];
		float V1plusV2 = V1[j] + V2[j];
		float V1plusV2uno = V1[j+1] + V2[j+1];
		float V1plusV2dos = V1[j+2] + V2[j+2];
		float V1plusV2tres = V1[j+3] + V2[j+3];

		for (i=k; i<k+BLK; i += 4)
		{
			Rj0 += X[i]*(X[i] - V1plusV2);
			Rj1 += X[i+1]*(X[i+1] - V1plusV2uno);
			Rj2 += X[i+2]*(X[i+2] - V1plusV2dos);
			Rj3 += X[i+3]*(X[i+3] - V1plusV2tres);
		}
		R[j] = Rj0;
		R[j+1] = Rj1;
		R[j+2] = Rj2;
		R[j+3] = Rj3;
	}
```

Ejercicio 2A:

Tabla con los resultados obtenidos en el ejercicio 2A:

N	Param	Time(s)	Inst.	Cycles	L3 Cache Misses	IPC	Eficiency
1000	n	1.65	5.6G	5.6G	0.039G	1	$1*10^{-9}$
1000	s	1.9	4.2G	6.4G	0.039G	0.656	$6.56*10^{-10}$
2000	n	17.6	44.2G	54.5G	1G	0.742	$7.42*10^{-10}$
2000	s	20.9	33.4G	70.5G	1G	0.473	$4.73*10^{-10}$

Tabla ejercicio 2A

Cuando el parámetro es n, el programa tarda menos en ejecutarse, hace más instrucciones pero menos ciclos dándole un IPC mayor.

Ejercicio 2B:

Cuando se incrementa el parametro N a 2000, el doble del original, las instrucciones, ciclos, nº de cache misses en L3 y tiempo aumentan más del doble.

Ejercicio 2C:

LOAD: $3*N^3$
STORE: 0
FAD: $2*N^3$
FMUL: N^3

Ejercicio 2D:

Tamaño L1: 32kB
Tamaño L2: 256kB
Tamaño L3: 8MB

Tamaño a[]: 8 000 bytes
Tamaño b[]: 8 000 000 bytes
Total a+b: 8 008 000 bytes

Si solo esta a[] en L1 si que se puede reutilizar los datos ya que cabe entero dentro de la cache, pero si estan a[] y b[] juntos, no caben ni en L1 ni en L2 ni en L3.

Ejercicio 2E:

Resultados obtenidos en el ejercicio 2E:

N	Param	Time	Inst.	Cycles	L3 misses	IPC
1000	t	0.58s	1.95G	1.96G	0.005G	0.994
2000	t	5.75s	15.3G	19.4G	0.122G	0.788

Cuando N vale 2000, vuelve a aumentar los fallos en L3 debido a que los vectores no caben en la cache de ningún nivel.

A mayor N, más lento es el programa debido al exceso de fallos en L3 con una disminución en IPC.

4 LOAD + 6 FAD + 4 FMUL, todo N^3 veces.

Para este ejercicio, el vector “a” da el mismo resultado que en el ejercicio anterior (2D), pero el vector “b” tiene menos fallos debido a que accedemos más veces al mismo lugar.

Ejercicio 2F:

Resultados obtenidos en el ejercicio 2F, la tabla ha sido dividida en 2 para que quepa en la misma página.

N	Param	Time (seg)	Inst.	Cycles	L3 Cache Misses	IPC	Eficiency
1000	r	0.35	2.32G	1.19G	1.57M	1.94	$1.94*10^{-9}$
2000	r	3.68	18.25G	12.42G	77.95M	1.47	$1.84*10^{-10}$

Tabla ejercicio 2F

N	Param	Reutilización de datos	Nº LOAD	Nº STORE
1000	r	58.21%	4.64M	1.69M
2000	r	23.38%	84.75	5.29M

Tamanyo a[]: 8*N*N+N = 8 001 000 bytes si N=1000
Tamanyo a[]: 8*N*N+N = 32 002 000 bytes si N=2000

Cuando N vale 1000 cabe justo dentro de la cahe de nivel 3 pero cuando vale 2000 sobrepassa su capacidad, por eso los misses en dicha cache incrementa de tal manera.

Ejercicio 2G:

Resultados obtenidos en el ejercicio 2G:

Step	Time(s)	IPC
200	2.69	2.13
250	2.56	2.16
400	2.695	2.12
500	2.98	1.93
1000	3.86	1.98

Tabla ejercicio 2G

Obtenemos como mejor step un valor de 250.
Entregar un step mayor no proporciona ninguna ventaja pero si desventajas, tales como menor IPC y mayor tiempo de ejecución.

Ejercicio 2H:

Para el mejor step seleccionado, obtenemos los datos siguientes:
IPC: 2.16
Nº LOAD: 41609533
Nº STORE: 5452487
Instrucciones: 19245450130
Ciclos: 8918397529
Reutilización de datos: 94,54%

Conclusiones:

Teniendo entonces finalizada la práctica podemos concluir que se han cumplido con los objetivos propuestos. Para aclarar, eran ver el desarrollo de un programa y sus efectos de él al acceder repetidamente a memoria cache.

Teniendo en cuenta los resultados, podemos concluir que el rendimiento de un programa disminuye cuanto mayor es su cantidad de accesos a L3 y a diversos niveles de cache. Además, que un vector no quepa en memoria o se acceda a una matriz por columnas y no por filas produce un aumento de fallos en cache, lo que conlleva a una ralentización del programa.