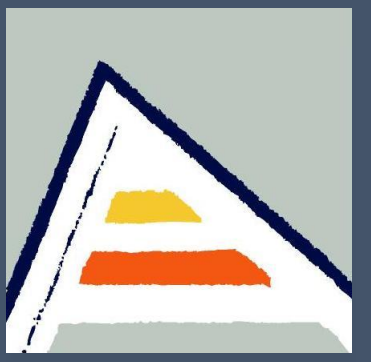
ARQUITECTURA DE LOS COMPUTADORES



07.02.2016

Grupo: 04

PRÁCTICA 2

FASE 2: Proyecto de evaluación del rendimiento

INTEGRANTES DEL GRUPO

Pavel Razgovorov | Y0888160Y (Director)
María Rico Martínez | 48775095F
Eddie Rodríguez Pastor 74391601X (Secretario)
Miguel Sánchez Moltó | 20096774H (Controlador)
César Sarrión Posas | 20456867T

ÍNDICE

- Introducción……..…..…..Página 2
- SPEC……Página 4
- Benchmark reducido……..…..Página 11
- Referencias……..…..Página 18

Introducción

Para la correcta realización del testeo de un equipo, deberemos ejecutar los que presenten una mejor compatibilidad y, además, nos generen resultados provechosos.

SPEC es un consorcio sin fines de lucro que incluye a vendedores de computadoras, integradores de sistemas, universidades, grupos de investigación, publicadores y consultores de todo el mundo. Tiene dos objetivos: crear un benchmark estándar para medir el rendimiento de computadoras y controlar y publicar los resultados de estos tests.

Entre los benchmarks creados por SPEC, cabe destacar SPEC2000, siendo el que usaremos en esta práctica con el fin de testear 6 equipos distintos. Éste benchmark, a su vez se divide en dos: CINT2000 (utilizado para medir y comparar el rendimiento de computación intensiva de enteros) y CFP2000 (utilizado para medir y compara el rendimiento de computación intensiva en flotantes).

Éstos, medirán únicamente el rendimiento del procesador, es decir, no intervendrán para nada en las funciones de entrada/salida, la funcionalidad en red o el entorno gráfico.

Así pues, la composición detallada de dichos benchmark resulta ser la siguiente:

CINT2000

El benchmark Cint2000, contiene 11 aplicaciones escritas en C y una en C++ (252.eon), las cuales serán usadas como benchmarks independientes en el testeo y, realizará pruebas en enteros. Estas aplicaciones, son las siguientes:

- 164.gzip: Utilidad de compresión de datos.
- 175.vpr: Direccionamiento y ubicación de circuitos FPGA.
- 176.gcc: Compilador C.
- 181.mcf: Costo mínimo de flujo de red.
- 186.crafty: Programa de ajedrez.
- 197.parser: Procesamiento de lenguaje natural.
- 252.eon: Efectos producidos por distintas fuentes de luz.
- 253.perlbmk: Perl.
- 254.gap: Teoría de grupo computacional.
- 255.vortex: Base de datos orientada a objetos.
- 256.bzip2: Utilidad de compresión de datos.
- 300.twolf: Simulador de ubicación y ruteo.

CFP2000

El benchmark Cfp2000, contiene 14 aplicaciones (6 en Fortran77, 4 en FORTRAN90 y 4 en C), las cuales serán usadas como benchmarks independientes en el testeo y, realizará pruebas en base coma flotante. Estas aplicaciones, son las siguientes:

- 168.wupwise: Cromodinámica de cuantos.
- 171.swim: Modelado de aguas poco profundas.
- 172.mgrid: Multi-grilla en campos potenciales 3D.
- 173.applu: Ecuaciones diferenciales parciales parabólicas/elípticas.
- 177.mesa: Biblioteca de gráficos 3D.
- 178.galgel: Dinámica de fluidos: análisis de inestabilidad oscilatoria.
- 179.art: Simulación de red neuronal: teoría de la resonancia adaptativa.
- 183.equake: Simulación de elementos finitos: modelado de terremotos.
- 187.facerec: Reconocimientos de imágenes: reconocimiento de rostros.
- 188.ammp: Química computacional.
- 189.lucas: Teoría de los números: prueba de primalidad.
- 191.fma3d: Simulación de elementos finitos en choque.
- 200.sixtrack: Modelo de acelerador de partículas.
- 301.apsi: Problemas de temperatura, viento y distribución de contaminantes.

SPEC

Caracteristicas de los equipos

Las caracteristicas de los equipos empleados son las siguientes:

PC 1

- Sistema operativo: Windows 8.1 ~ 64 bits
- Procesador: Intel® Core™ i7-4700HQ @ 2.40 GHz
- Número de núcleos: 8
- Memoria RAM: 8GB

PC 2

- Sistema operativo: Windows 7 ~ 64 bits
- Procesador: Intel® Core™ 2 Duo @ 2.20 GHz
- Número de núcleos: 2
- Memoria RAM: 4GB

PC 3

- Sistema operativo: Windows 10 ~ 64 bits
- Procesador: Intel® i5-4460 @ 3.2 GHz
- Número de núcleos: 4
- Memoria RAM: 8GB

PC 4

- Sistema operativo: Windows 10 ~ 32 bits
- Procesador: Intel® Core™ 2 Duo @ 2.16 GHz
- Número de núcleos: 2
- Memoria RAM: 4GB

PC 5

- Sistema operativo: Windows 10 ~ 64 bits
- Procesador: Intel® Core™ i5-3317U @ 1.70 GHz
- Número de núcleos: 2
- Memoria RAM: 4GB

PC 6

- Sistema operativo: Windows 8-1 ~ 64 bits
- Procesador: Intel® Core™ i7-5500U @ 2.40 GHz
- Número de núcleos: 2
- Memoria RAM: 8GB

Las características para cada procesador, son las siguientes:

Procesador	Intel® Core ™ i7-4700HQ	Intel® Core™ 2 Duo	Intel® i5-4460	Intel® Core™ 2 Duo	Intel® Core™ i5-3317U	Intel® Core™ i7-5500U
Fabricación	22 nm	65 nm	22 nm	65 nm	22 nm	14 nm
Socket	1364	775	1150	479	1023	1168
Caché	6 MB	2 MB	6 MB	4 MB	3 MB	4 MB
GPU integrada	Intel HD Graphics 4600	Intel GMA 950	Intel HD Graphics 4600	Intel GMA 950	Intel HD Graphics 4000	Intel HD Graphics 5500
T. Virtualización	Si	No	Si	Si	Si	Si
T. Hyper-Threading	Si	No	No	No	Si	Si
TDP	47 w	65w	84 w	34 w	17 w	7,5 w
Frecuencia base	2.4 GHz	2.20 GHz	3.2 GHz	2.16 GHz	1.7 GHz	2.4 GHz
Frec. turbo máxima	3.4 GHz	-	3.4 GHz	-	2.6 GHz	3 GHz

Si nos disponemos a analizar las características de estos procesadores, podemos destacar algunos factores de cada uno de los mismos:

- El PC 1, presenta un procesador poseedor de un mayor número de núcleos que el resto de equipos, obteniendo de esa manera un mayor paralelismo y consigo un mayor rendimiento. Además, resulta ser de los pocos que presentan tecnología Hyper Threading.

- El procesador del equipo PC 2, sin embargo, resulta ser el que presenta mayores dimensiones de fabricación (65 nm), además de poseer una caché bastante escasa en comparación con el resto de equipos utilizados y de ser uno de los que mayor consumo presenta.

- El procesador con mayor frecuencia base resulta ser el del equipo PC 3, que con 3.2 GHz supera al resto de frecuencias. Ello repercutirá en su consumo, tal y como observamos en la tabla, siendo el equipo que más consumirá con una cifra de 84 w.

- El PC 4, presenta un procesador muy similar al del equipo PC 2, diferenciándose principalmente en los valores de caché, consumo y frecuencia base, aunque este último resulta ser muy similar entre ambos.

- El equipo PC 5, posee un procesador cuya frecuencia base y máxima, son las más escasas de entre todos los equipos. Por otra parte, resulta ser uno de los que menor consumo presentan y, de los pocos con tecnología Hyper Threading.

- Vemos que el procesador del PC 6, resulta ser claramente el que presenta menor consumo, con tan solo 7.5 w, superando con creces al resto de equipos. Además, en fabricación resulta ser el menor de toda la comparativa, con tan solo 14 nm.

Testeo

Para comenzar el testeo, primero deberemos copiar la carpeta "specpu2000" de los laboratorios, en el directorio raíz del disco duro de cada uno de los equipos implicados. Seguidamente, editaremos el archivo "*shrc.bat"* tal y como se nos especifica en el enunciado de la práctica, modificando las siguientes líneas:

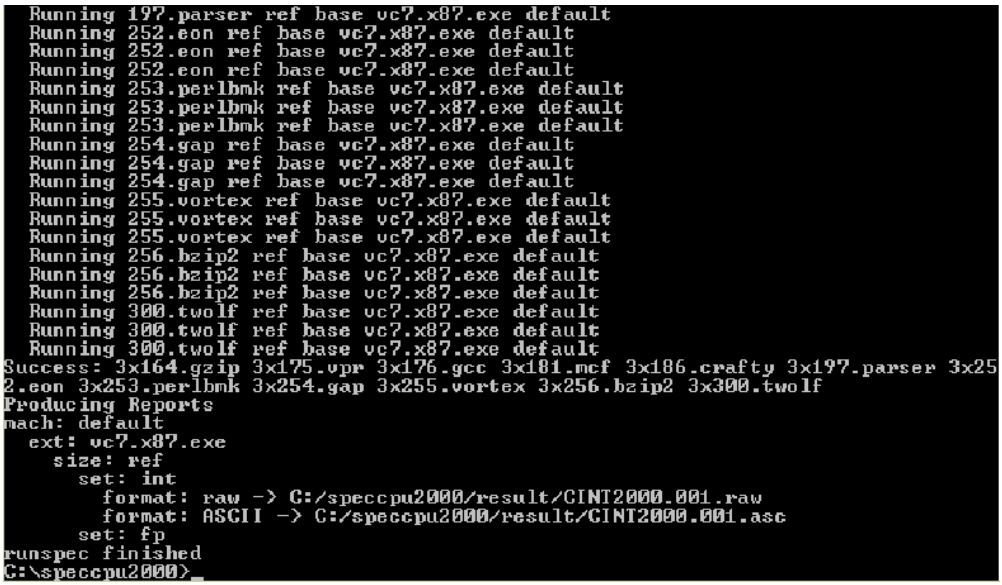
```
set SHRC_COMPILER_PATH_SET=yes
call "C:\Archivos de programa\Microsoft Visual Studio 10.0\VC\bin\vcvars32.bat"
```

Así pues, quitaremos el comentario que trae de serie el archivo en ambas líneas y, procederemos a colocar correctamente la ruta de nuestro entorno Visual Studio, variando la misma dependiendo del equipo. Finalizada la modificación, procederemos a guardar la misma y, con ello ya podremos proceder a la ejecución de los benchmark.

La ejecución de SPEC, la llevaremos a cabo utilizando la terminal que nos brinda Windows mediante el comando "*cmd*", siendo éste término una abreviatura de "command" (comando que se encargará de abrir la consola de MS-DOS).

CINT2000

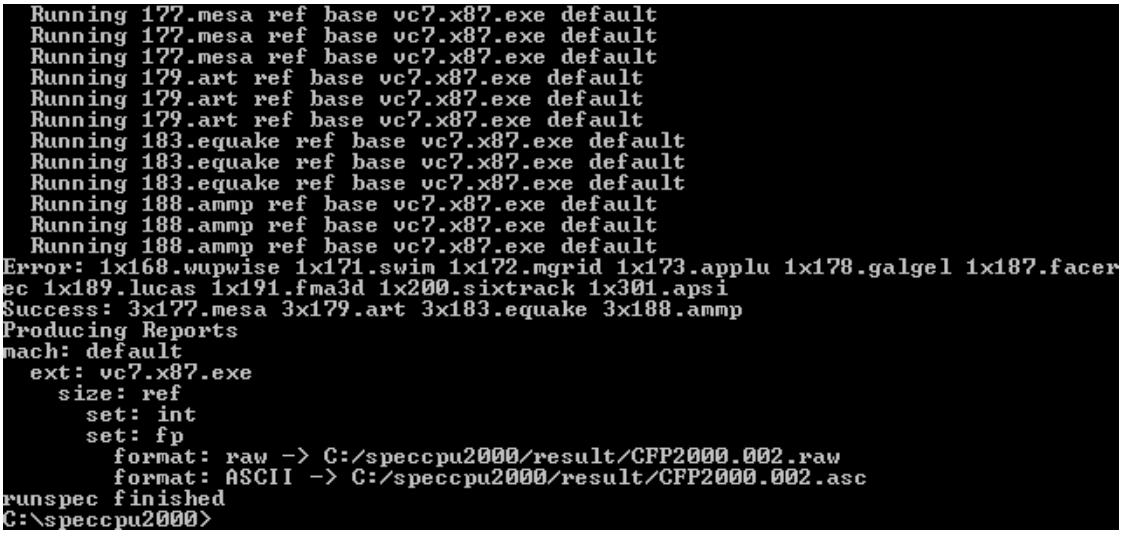
Para la ejecución de Cint2000, introduciremos la siguiente ruta de comandos en la terminal: **runspec --reportable --config=win32-x86-vc7.cfg -T base int**



En este caso, nos interesará el fichero generado con extensión ".asc" que contendrá los resultados de la prueba (CINT2000.001.asc).

CFP2000

Para la ejecución de Cfp2000, introduciremos la siguiente ruta de comandos en la terminal: **runspec --reportable --config=win32-x86-vc7.cfg -T base fp**



En este caso, nos interesará el fichero generado con extensión ".asc" que contendrá los resultados de la prueba (CFP2000.002.asc). Tras dicha ejecución, podemos observar que fallan prácticamente todos los benchmarks, a excepción de 4: 177.mesa, 179.art, 183.equake y 188.ammp.

Además, cada uno de los benchmark que utilizamos en el testeado es ejecutado 3 veces, para así, con una mayor fiabilidad, generar los resultados finales.

■ *Resultados*

Estos resultados obtenidos, aparecen organizados en una tabla mediante las siguientes columnas:

- **Base Ref Time:** son los valores de la máquina de referencia, que es una Sun Ultra 5/10 de estación de trabajo con un procesador de 300 MHz SPARC y 256 MB de memoria, que proporciona el propio SPEC.
- **Base Run Time:** será el tiempo de ejecución de la opción base.
- **Base Ratio:** esta columna indica la ganancia de rendimiento respecto del valor de referencia. Su cálculo se realiza de la siguiente manera:
  Ratio = 100 × RefTime /Run Time
- **Peak Ref Time:** será igual que Base Ref Time.
- **Peak Run Time:** será el tiempo de ejecución de la opción peak.
- **Peak Ratio:** será igual que Base Ratio pero basándonos en el resultado peak de referencia.

Así pues, los resultados obtenidos para SPECCint2000, podemos resumirlos en las siguientes tablas:

SPECCINT2000 -RUN TIME						
Benchmarks	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6
164.gzip	71	117,0	67,7	142,0	96,7	306
175.vpr	55,3	99,6	49,2	142,0	75,3	220
176.gcc	26,4	52,9	24,5	69,1	37,4	113
181.mcf	25,9	99,3	25,5	159,0	40,6	119
186.crafty	29,8	53,8	29,5	69,3	44,4	134
197.parser	67,7	127,0	66	178,0	93,7	296
252.eon	33,4	70,3	31,4	82,2	47,2	145
253.perlbmk	46	90,0	40,6	132,0	78,1	193
254.gap	30,6	57,0	26,6	75,9	39,4	122
255.vortex	40,7	84,4	34,5	124,0	58,9	149
256.bzip2	62,2	106,0	55,3	145,0	83,6	243
300.twolf	86,8	131,0	80,1	222,0	109	343

SPECCINT2000 -RATIO						
Benchmarks	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6
164.gzip	1973	1196	2069	988	1448	457
175.vpr	2533	1406	2843	987	1859	637
176.gcc	4165	2081	4493	1592	2942	977
181.mcf	6939	1813	7057	1131	4429	1509
186.crafty	3354	1860	3390	1443	2250	744
197.parser	2659	1419	2726	1013	1921	608
252.eon	3888	1850	4146	1581	2757	898
253.perlbmk	3915	2000	4430	1361	2305	935
254.gap	3599	1929	4135	1450	2789	900
255.vortex	4670	2251	5502	1536	3223	1272
256.bzip2	2411	1414	2712	1038	1795	618
300.twolf	3456	2288	3747	1354	2763	876

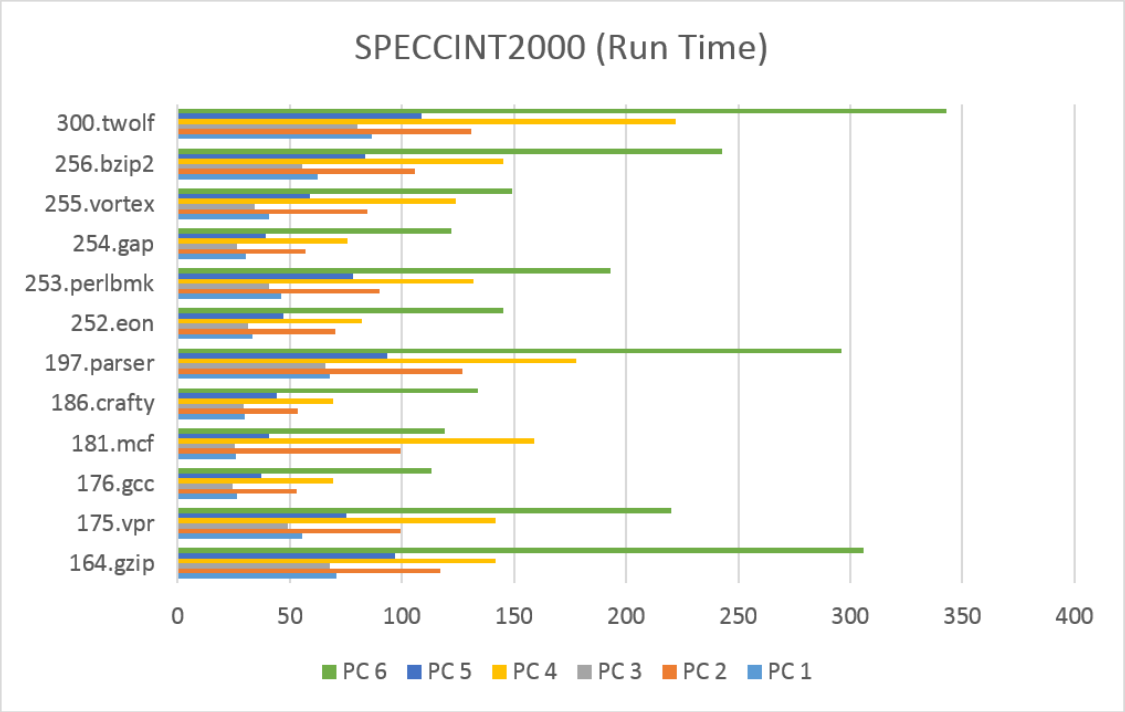
Y, de igual forma, los resultados obtenidos para SPECfp2000, podemos resumirlos en las siguientes tablas:

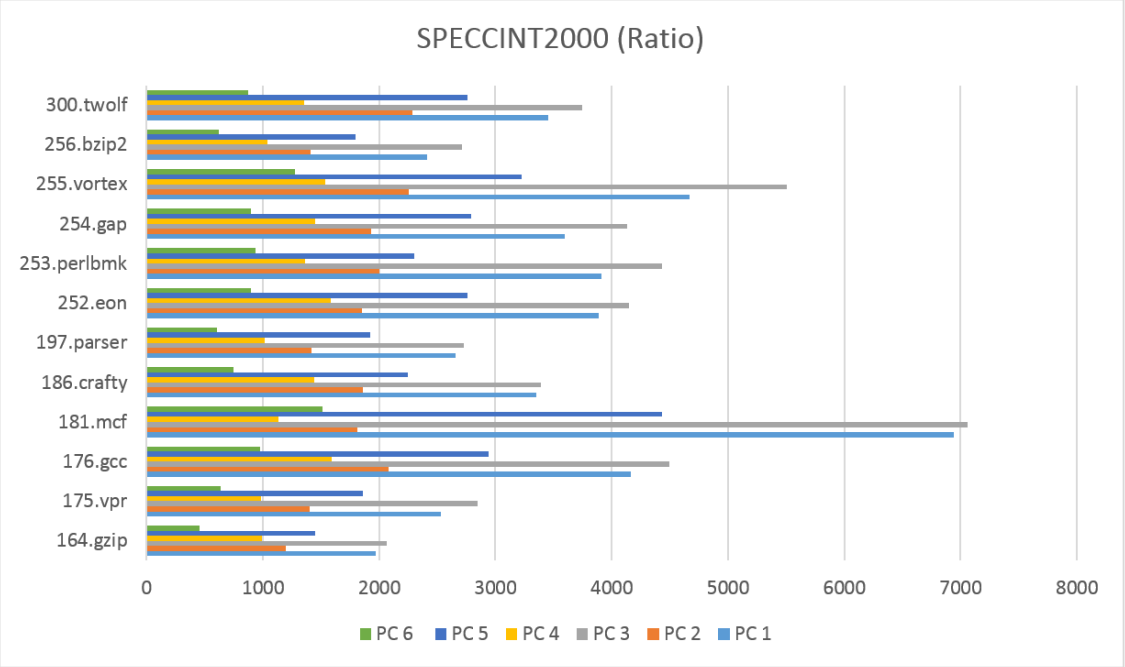
SPECFP2000 -RUN TIME						
Benchmarks	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6
177.mesa	49,1	103	45,2	134	68,8	233
179.art	23,5	56,2	21,7	111	33,2	101
183.equake	22,9	54,7	21,5	106	30,8	87,3
188.ammp	71,5	153	65,1	209	97,0	329

SPECFP2000 -RATIO						
Benchmarks	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6
177.mesa	2853	1356	3097	1046	2034	601
179.art	11085	4627	11962	2336	7829	2567
183.equake	5667	2376	6033	1232	4224	1489
188.ammp	3076	1442	3382	1051	2269	670

Una vez obtenidos estos datos, procederemos a representar las gráficas y a comentar los resultados correspondientes a los datos de "Run time" y de "Ratio", tanto para CINT2000 como para CFP2000.

CINT2000





Tal y como podemos observar en ambas gráficas, el PC 3 es el que mejores resultados ha obtenido en cuanto a tiempo de ejecución y rendimiento, siendo de esta manera el equipo claramente vencedor de esta comparativa. Ello es debido a que, aunque exista otro equipo con un mayor número de núcleos, su velocidad de procesamiento de la CPU es la mayor de todos.

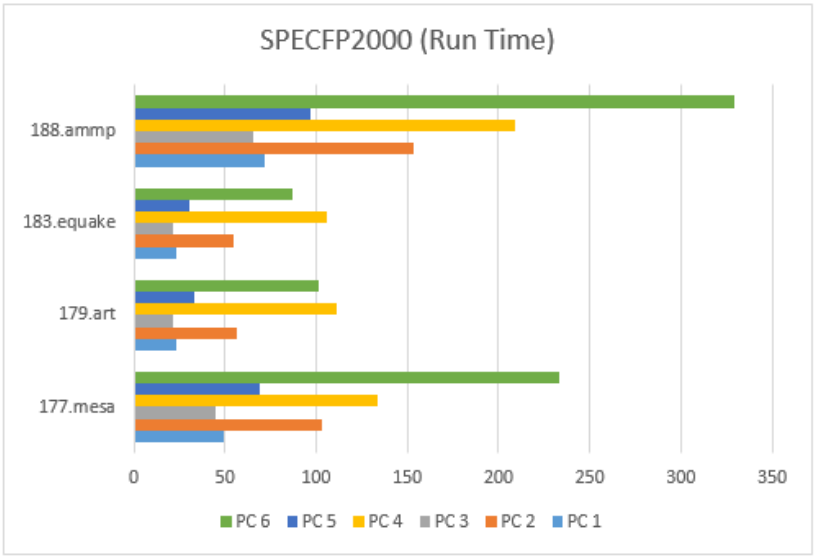
Los siguientes equipos con mejores resultados, son el PC 1 y PC 5, aunque si comparamos sus características, poseen una velocidad de procesamiento, memoria RAM y número de núcleos muy dispares entre sí.

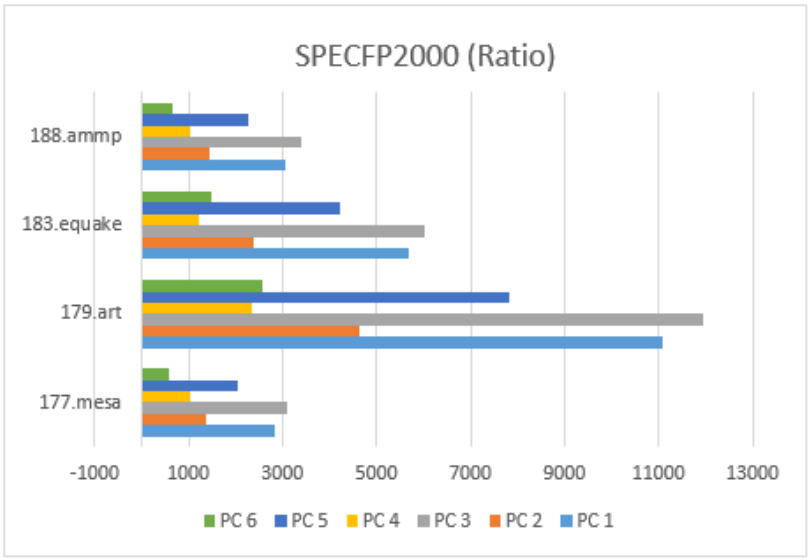
Por tanto, los resultados tan cercanos entre ambos equipos en la gráfica, deben de verse influenciados por otros factores como pueden ser el sistema operativo, la cantidad de procesos que estaban ejecutándose en cada equipo cuando se realizó la prueba, temperatura, etc.

El PC 5 resulta ser superado en características por el PC 1, puesto que éste último posee un número mayor de núcleos (8 cores) y una mayor memoria RAM (8 MB), lo cual permitirá un mayor paralelismo y ello, fomentará una mayor productividad.

Además, podemos añadir, que se observa con notable claridad que el equipo que presenta una mayor lentitud y una productividad menor resulta ser el equipo PC 6.

CFP2000





En el benchmark CFP2000, los resultados en cuanto a la distribución de rendimientos son muy similares a la prueba CINT2000, por tanto podemos decir que el Hardware más influyente sigue siendo la velocidad de procesamiento de la CPU y no el número de núcleos que tiene.

Benchmark reducido

En este apartado, procederemos a desarrollar un benchmark reducido para la evaluación del rendimiento que, posteriormente, utilizaremos en cada uno de los equipos para, de esta manera testearlos.

Por consiguiente, el benchmark desarrollado es ***GuessFibonacci***, cuya funcionalidad será la siguiente: dado un array de números enteros generados aleatoriamente (C), el programa adivina si el número pertenece a la serie de Fibonacci y, además, su paridad (ASM).

Posteriormente, con estos resultados, generará un fichero de texto con cada una de comprobaciones realizadas. Además, cada una de las pruebas ejecutadas en el testeo, será medida en milisegundos, mostrándose estos tiempos de ejecución por pantalla y, de igual forma, exportando estos resultados a un nuevo fichero, el cual se creará tantas veces como ejecutemos el benchmark.

Así pues el algoritmo creado, quedaría de la siguiente manera:

- ***FunctionsASM.cpp***

```
// GuessFibonacci

void guessIfFibonacciASM(unsigned long int* n, bool* g, int tam) {
	unsigned long int i, j, a, b;
	__asm {
		/**
		* VARIABLES:
		* eax -> vector de enteros. Intocable
		* ebx -> vector de booleanos. Intable
		* ecx -> auxiliar para hacer a = b; (imposible mover de memoria a
			memoria)
		* edx -> num. fibonacci calculado. Cambia segun la iteracion
		* edi -> auxiliar para hacer comparaciones entre memoria y memoria
		* esi -> numero de la posicion i del vector
		*/
		mov eax, n;
		mov ebx, g;
		mov i, 0;

	bucle_guessIfFibonacci:
		mov edi, tam;
		cmp i, edi;
		je fin_bucle_guessIfFibonacci;

		mov a, 0; //a corresponde con el primer numero de fibonnacci
		mov b, 1; //b con el segundo elemento de Fibonacci
			  //la suma de ambos sera j nuestro numero actual

		mov edi, 1; //Indice de los k-ésimos números fibonacci que iremos
			    //calculando

		mov j, 0; //numero j-ésimo de la serie de Fibonacci

	bucle_isFibonacci:
		//getFibonacci
```

```
            cmp j, 0; //Si es el primer número, por norma es el 0. Si no, lo
                      //calculamos
            je primerFibonacci;

            mov edx, 1; //Si no es el primer número, como mínimo será 0

      bucle_getFibonacci:
            cmp j, edi; // j < k
            je fin_bucle_getFibonacci;

            xor edx, edx;
            add edx, a;
            add edx, b; //t = a+b
            mov ecx, b;
            mov a, ecx; // a = b
            mov b, edx; //b = t

            inc edi; //k++
            jmp bucle_getFibonacci;

      primerFibonacci:
            mov edx, 0;
            jmp fin_bucle_getFibonacci;

      fin_bucle_getFibonacci:
            //isFibonacci despues de la llamada de getFibonacci
            mov esi, i;
            mov esi, [eax + esi * 4]; //cada n que recibe isFibonacci
            cmp esi, edx; // n > number -> nextFibonacci
            jg nextFibonacci;
            cmp esi, edx; // n == number -> fibonacciTrue
            je fibonacciTrue;
            jmp fibonacciFalse; // en cualquier otro caso (n < number, nos
                                // hemos pasado) --> fibonacciFalse

      nextFibonacci:
            inc j; //Probamos a encontrar el siguiente numero de Fibonacci
            jmp bucle_isFibonacci;

      fibonacciTrue:
            mov esi, i;
            mov[ebx + esi], 1; //Metemos en la posicion i el valor true
            inc i; //i++ para avanzar en el proximo numero que adivinemos
            jmp bucle_guessIfFibonacci;

      fibonacciFalse:
            mov esi, i;
            mov[ebx + esi], 0; //Metemos en la posicion i el valor false
            inc i; //i++ para avanzar en el proximo numero que adivinemos
            jmp bucle_guessIfFibonacci;

      fin_bucle_guessIfFibonacci:
      }
}

void guessIfEvenASM(unsigned long int* n, bool* e, int tam) {
      //even true, odd false
      __asm {
            mov eax, 0; //i=0
            mov edi, n; //puntero al vector
```

```
	buclei:
		mov ecx, [edi + eax * 4]; //copiamos el numero a tratar
		and ecx, 00000001; //hacemos una and para actualizar el flag PF
					//(parity flag)

		jp isEven; //si hay paridad par, saltamos a EVEN
		jnp isOdd; //en caso contrario a ODD
	continuarBucle:
		inc eax; //i++
		cmp eax, tam; //mientras i<VSIZE
		jl buclei; //continuamos
		jmp fin; //finalizamos

	isEven:
		mov edx, e; //vector de booleanos
				//mov ebx, i; //posicion del vector
		mov[edx + eax], 1; //Metemos en la posicion i el valor true
		jmp continuarBucle;

	isOdd:
		mov edx, e; //vector de booleanos
				//mov ebx, i; //posicion del vector
		mov[edx + eax], 0; //Metemos en la posicion i el valor false
		jmp continuarBucle;
	fin:
	}
}
```

Concurrency.cpp

```
#include "Concurrency.h"

void generateRandomNumbers(unsigned long int* v, int tam) {
	srand(0);
	//for (int i = 0; i < VSIZE; i++) v[i] = rand() % MAXRAND;
	for (int i = 0; i < tam; i++) v[i] = rand() % MAXRAND;
}

int numberCores() {
	SYSTEM_INFO sysinfo;
	GetSystemInfo(&sysinfo);
	return sysinfo.dwNumberOfProcessors;
}

void concurrentBenchmark() {
	vector<thread> threads;
	int cores = numberCores();

	printf("(numero de cores: %i)\n\n", cores);
	for (int i = 0; i < cores; i++) {
		//Por cada core que tengamos, creamos un nuevo thread al que le
		//pasamos la funcion lambda que sigue
		threads.push_back(thread([](int id) -> void {
			printf("START Thread %i\n", id);
			int tam = VSIZE / numberCores();

			unsigned long int* numbersConcurrent = new unsigned long
int[tam];
			generateRandomNumbers(numbersConcurrent, tam);
```

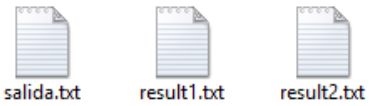
```
                    bool* fibonacciConcurrent = new bool[tam];
                    guessIfFibonacciASM(numbersConcurrent, fibonacciConcurrent,
tam);

                    bool* parityConcurrent = new bool[tam];
                    guessIfEvenASM(numbersConcurrent, parityConcurrent, tam);

                    delete[] numbersConcurrent, fibonacciConcurrent,
parityConcurrent;
                    printf("END Thread %i\n", id);
             }, i + 1));
      }
      //Liberamos todos los cores antes de terminar
      auto firstThread = threads.begin();
      while (firstThread != threads.end()) {
             firstThread->join();
             firstThread++;
      }
}
```

Testeo

Una vez implementado el algoritmo principal, procederemos a probar la ejecución del programa:

En primer lugar, ejecutaremos el archivo con extensión .exe que lanzará el testeo sobre el equipo, tal y como se observa en la siguiente imagen.

En segundo lugar, una vez ejecutadas dichas pruebas, procederemos a visualizar la correcta creación y contenido de los archivos que contendrán los resultados obtenidos de este testeo.



Tras haber ejecutado el benchmark 2 veces, observamos la obtención de 2 archivos distintos con resultados en milisegundos sobre la ejecución del mismo y, el fichero salida.

■ *Resultados*

Tras ello, comprobamos el contenido de los mismos, apareciendo los resultados en el siguiente formato:

salida.txt

NUM	FIB	EVEN
38	F	T
7719	F	F
21238	F	T
2437	F	F
8855	F	F
11797	F	F
8365	F	F
32285	F	F
10450	F	T
30612	F	T
5853	F	F
28100	F	T
1142	F	T
281	F	F
20537	F	F
……..	..	..

result.txt

```
----------------------------------------
Resultados:
----------------------------------------
Test generacion:                2716 ms
Test fibonacci:                 6790 ms
Test paridad:                   611 ms
Test total single thread:       10117 ms
Test multithreading:            2642 ms
Test escritura en memoria:      66227 ms
----------------------------------------
```

Estos resultados obtenidos, aparecen organizados en una tabla mediante las siguientes columnas:

- **Test generación:** temporizará lo tardado en generar el archivo con los resultados.
- **Test Fibonacci: se encargará de** adivinar si el número pertenece a la serie de Fibonacci. La sucesión de Fibonacci, consta de una serie de números naturales que se suman de a 2, a partir de 0 y 1. Básicamente, la sucesión de Fibonacci se realiza sumando siempre los últimos 2 números.
  Serie → 0,1,1,2,3,5,8,13,21,34 …
  Sucesión → 0+1=1 / 1+1=2 / 1+2=3 / 2+3=5 / 3+5=8 / 5+8=13 / 8+13=21 …
- **Test paridad:** consiste en la adición de un bit más en cada Byte de memoria, que pasa a tener 9 bits, teniendo este último la función de diagnosticar alteraciones en los datos.

- **Test total single thread:** resulta ser el resultado de la suma de los 3 anteriores, tarea la cual realizará el propio procesador.

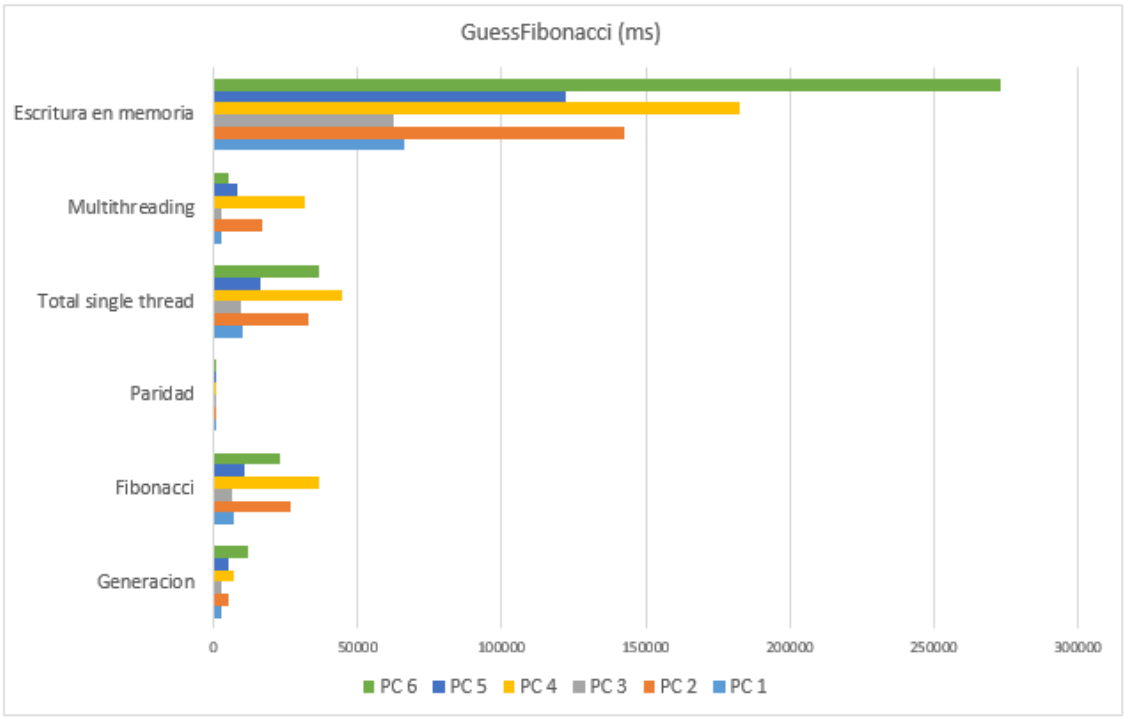
- **Test multithreading:** técnica en la cual determinaremos que, un único conjunto de código puede ser utilizado por varios procesos en diferentes etapas de ejecución.

- **Test escritura en memoria:** tal y como indica su término asociado, se encargará de temporizar la escritura en memoria sobre cada uno de los equipos.

De esta manera, los resultados obtenidos para ***GuessFibonacci***, podemos resumirlos en la siguiente tabla:

GuessFibonacci (ms)						
Benchmarks	**PC 1**	**PC 2**	**PC 3**	**PC 4**	**PC 5**	**PC 6**
Generacion	2716	5321	2915	7228	5058	12054
Fibonacci	6790	26995	6287	36523	10452	23267
Paridad	611	770	553	950	892	1000
Total single thread	10117	33086	9755	44701	16402	36321
Multithreading	2642	16756	2718	31612	8294	5243
Escritura en memoria	66227	142296	62628	182449	122144	273079

Así pues, una vez testeados los 6 equipos con el algoritmo propuesto, procederemos a realizar una comparativa de los resultados obtenidos para los mismos.



Tras realizar la gráfica comparativa de los resultados obtenidos para cada uno de los 6 equipos, podemos observar nuevamente que el PC 3 es el que mejores resultados ha obtenido en cuanto a tiempo de ejecución, siendo de esta manera el equipo claramente vencedor de esta comparativa y de la anterior.

Los siguientes equipos con mejores resultados, al igual que anteriormente, son el PC 1 y PC 5 que, poseyendo una velocidad de procesamiento, memoria RAM y número de núcleos muy dispares entre sí, ocupan la posición 2 (PC 1) y la posición 3 (PC 5) respectivamente, de nuestro ranking.

Además, podemos añadir, que se observa con notable claridad que el equipo que presenta una mayor lentitud en la ejecución de los test, resulta ser el equipo PC 6 tal y como concluimos también en la anterior comparativa con los resultados proporcionados por SPEC.

Referencias

- **SPEC**

http://es.wikipedia.org/wiki/Standard_Performance_Evaluation_Corporation
http://www.spec.org/

- **Especificaciones de los procesadores**

http://ark.intel.com/es-es/products/

- **Paridad**

https://books.google.es/books?id=Kq29CC8RfGEC&pg=PA159&lpg=PA159&dq=que+es+la+paridad+de+un+pc&source=bl&ots=CNSH7SOKl_&sig=QVcv4j8IQgTFTJDNLHo9JPSor7s&hl=es&sa=X&ved=0ahUKEwjar-rNyavLAhVBBBoKHeEuCnkQ6AEIOjAE#v=onepage&q=que%20es%20la%20paridad%20de%20un%20pc&f=false

- **La sucesión de Fibonnaci**

http://www.batanga.com/curiosidades/4461/que-es-la-sucesion-de-fibonacci

- **Repositorio**

github.com/paveltrufi/fase2-ac