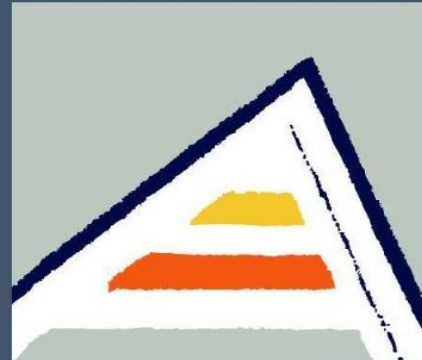


ARQUITECTURA DE LOS COMPUTADORES FASE II



05/04/2020

GRUPO 5

Elvi Miha Sabau
Francisco Javier Pérez
Joaquín Ferris
Jorge Belló Rico
Germán Berná Martínez
Sami Hadj Djilani
Pablo Ortuño Níguez

ÍNDICE

1. Descripción del objetivo de los benchmarks.
2. Especificación que cumplen nuestros benchmarks.
3. Exposición de los resultados.
4. Nuestros benchmarks en detalle.
5. Informe SPEC de los resultados de nuestros benchmarks.
6. SPEC CPU2000, evaluación de resultados y comparativas.

Descripción del objetivo de los benchmarks:

El conjunto de nuestros benchmarks se enfoca en la evaluación de la velocidad de cálculo y carga sobre la memoria y la ALU del ordenador. Realizando cálculos sobre vectores de diferentes tamaños aumentando la carga cada vez más de manera lineal, y evaluando los resultados medios obtenidos por el tiempo que tarda en realizar cada operación.

Especificaciones que cumplen nuestros benchmarks.

El algoritmo que usamos genera 2 vectores de distintas tallas y con contenido ascendente, la talla de los vectores varía de 100 a 130. Por cada tamaño se realizan 3 millones de iteraciones, en cada iteración se realiza una suma y una multiplicación entre sí, para cada una de las iteraciones del tamaño, y el resultado se guarda en una variable constante que se borrará para cuando el siguiente cálculo se realice, esto se debe a que, en versiones diferentes del compilador, cuando se realizan cálculos que no se guardan, el compilador automáticamente ignora dicha operación aritmética.

Exposición de los resultados.

Nuestros benchmarks realizan un análisis empírico / teórico del tiempo promedio que el ordenador tarda en computar varias operaciones elementales, el benchmark nos muestra el tiempo promedio que ha tardado en realizar cada una de las tallas del vector, y un tiempo total promedio de todas las tallas, con estas medidas podemos hacer un análisis temporal de cómo se comporta el ordenador al aumentar la carga de datos y el tamaño del problema del propio algoritmo, además, en la propia terminal, se nos mostrará los datos del ordenador, de esta manera podemos realizar comparativas entre diferentes ordenadores, y entre las propias versiones el benchmark. Dichos resultados después se exportan a unos archivos llamados “salida_C.txt”, “salida_MMX.txt”, “salida_SSE.txt” (dependiendo del benchmark que se ejecuta) para poder maniobrar los resultados de manera más cómoda, y si queremos un acceso más sencillo de los datos del ordenador / SO, tendremos 2 archivos, uno llamado “HWInfo.txt” que guarda la información del tipo de procesador, y otro “SWInfo.txt” que guarda la información del SO sobre el que se ejecuta. El benchmark está adaptado para funcionar de manera compatible en sistemas ejecutando tanto Windows como GNU/Linux.

Nuestros benchmarks en detalle:

Todas las versiones de nuestros benchmarks cumplen las mismas especificaciones anteriormente mencionadas, pero para ser más concretos, en este apartado explicaremos qué otras especificaciones cumplen, como funcionan internamente y de qué otras herramientas usa, y que otros aspectos posee.

1. Compatibilidad entre C/C++.

Los 3 benchmarks usan la misma base, ya que lo único que cambia entre ellos es la manera en la que realizan las operaciones. Dicho esto, las 3 versiones están hechas en código en C compilado como C++, esto se debe a que, a la hora de usar dependencias externas en un proyecto de C++ de Visual Studio, dichas dependencias están en C++, y por ende no se puede compilar en C. Para ello, para solventar dicho problema y que el código pueda ser ejecutado en cualquier otro entorno, y compilado como un programa en C, el código está totalmente en C, sobre un archivo con extensión .cpp, siendo compilado en C++.

2. Memoria dinámica con Malloc().

Nuestros benchmarks usan malloc para asignar la memoria dinámica necesaria a lo largo del proceso, esto se debe a que ya que queremos que nuestro benchmark esté en C puro, ya que la terminología de “new” fue introducida por C++, esto supuso realizar las comprobaciones, casteos y manipulaciones necesarias para que no haya fallos en tiempo de ejecución, aunque a su beneficio, nos será más fácil maniobrar más tarde cuando tengamos que realizar el benchmark en CUDA usando cudaMalloc y el resto de métodos de la librería de Nvidia.

3. Compatibilidad entre sistemas GNU/Linux y Windows.

Uno de nuestros objetivos extra fue el de hacer que la información recopilada del sistema por el benchmark fuera posible tanto en Linux como en Windows, para ello, nuestro benchmark realiza un par de comprobaciones para saber sobre qué SO se está ejecutando, y dependiendo del SO, se ejecutarán un set de comandos del sistema para extraer la información del software y del hardware del ordenador.

Informe de los resultados de nuestros benchmarks usando el standard SPEC:

El informe contendrá los siguientes apartados:

1. Descripción hardware de los ordenadores testeados.
2. Descripción software de los ordenadores testeados.
3. Manera de ejecutar nuestros benchmarks.
4. Tablas comparativas entre benchmarks y ordenadores.

Detalles de las unidades a analizar (apartado 1 y 2):

Para ello, primero antes de mostrar las gráficas y los resultados, mostraremos todos los datos, tanto hardware como software, de cada ordenador, y las etiquetamos para saber cuál es cual, debido a que son 7 unidades a testear, ya que así será más legible y fácil de entender.

	Hardware				Software	
Unidad	CPU	Núcleos	GHz	Max CLK	SO	Arch
Unid. 1	Intel(R) Core(TM) i7-6700 CPU	4	3.40GHz	3401	W10	64
Unid. 2	Intel(R) Core(TM) i5-10210U	4	1.60GHz	2112	W10	64
Unid. 3	Intel(R) Celeron(R) CPU N3060	2	1.60GHz	1601	W10	64
Unid. 4	Intel(R) Core(TM) i7-8750H	6	2.20GHz	2208	W10	64
Unid. 5	Intel(R) Core(TM) i5-7200U	2	2.50GHz	2701	W10	64
Unid. 6	Intel(R) Core(TM) i7-8550U	4	1.80GHz	1992	W10	64
Unid. 7	Intel(R) Core(TM) i7-7700HQ	4	2.80GHz	2801	W10	64

*Los datos de esta tabla son solo un resumen mínimo de las especificaciones totales de los ordenadores, si se desea consultar todas las especificaciones, por favor, los documentos aportados con este informe.

Cómo compilar (apartado 3):

-Dependiendo del SO en el que se ejecute:

- a. Windows: Los benchmarks son soluciones de visual studio, para buildear la aplicación se deberá cargar la solución (.sln), cambiar el estado de a “Debug” y apretar F6 para compilar, y en la carpeta “Debug” se debería guardar el benchmark listo para ser ejecutado. *(Compilamos con debug ya que con Release, ciertas variables acaban optimizadas, y por ello, el benchmark en C deja de ser funcional).
- b. GNU/Linux: Los benchmarks no dependen de librerías externas a C++, por ende, cambiar la sintaxis del código ensamblador a corde al compilador que uses, y compilar el archivo fuente .cpp sería suficiente para crear el archivo ejecutable. Para ello nos localizamos en la carpeta fuente, y ejecutaremos el siguiente comando:

- Sin optimizar

“g++ -o Benchmark.cpp Benchmark -O0 && ./Benchmark”

- Optimizado

“g++ -o Benchmark.cpp Benchmark -O3 && ./Benchmark”

Este comando compila la aplicación, y si la compilación fue exitosa, lo ejecutará.

Tablas comparativas y gráficas (apartado 4).

Las tablas comparativas muestran una comparativa entre ordenadores y por cada ordenador, la diferencia entre usar instrucciones en C / MMX / SSE para realizar el mismo problema.

La tabla también muestra el tiempo medio por cada tipo de set, y la ganancia comparativa entre usar cada uno de ellos entre sí.

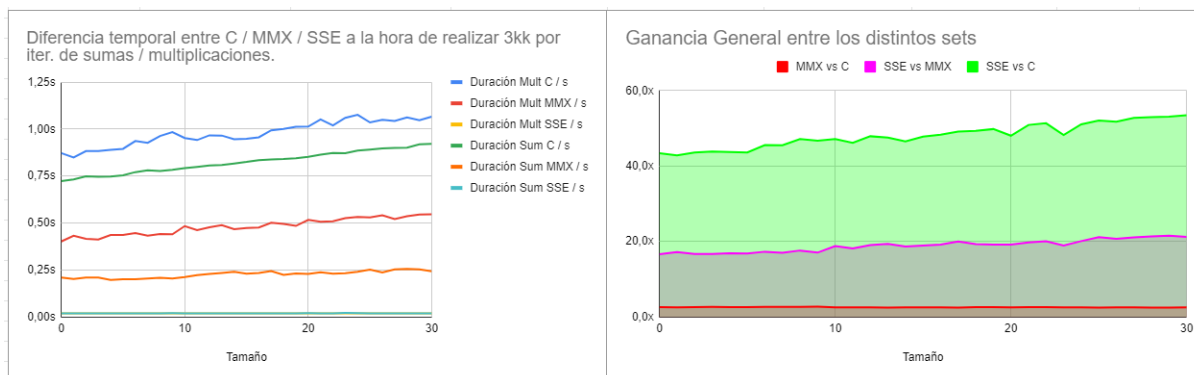
Para comparar la Ganancia / Aceleramiento rendimiento de cada set, se usará la siguiente fórmula para determinar cuántas veces es un set más rápido en comparativa con el resto.

$$\text{Aceleración Rendimiento} = \frac{\text{Rendimiento con mejora}}{\text{Rendimiento sin mejora}} = \frac{\text{Tiempo ejecución sin mejora}}{\text{Tiempo ejecución con mejora}}$$

*Se pueden acceder a todos los detalles desde la hoja Excel aportada con el informe.

Unidad 1:

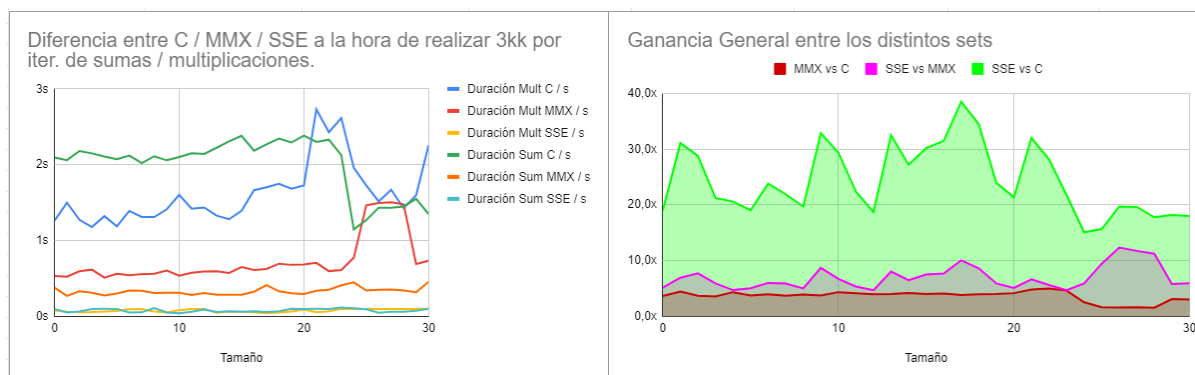
Tiempo que tarda el ordenador en hacer 3 000 000 de cálculos con arrays de tamaños variando de 100 a 130. (30 iteraciones * 3kk operaciones).									
Multiplicaciones				Sumas			Ganancia general		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	0.871932	0.401555	0.018170	0.722817	0.208960	0.018591	2.6x	16.6x	43.4x
1	0.849052	0.432004	0.018494	0.731656	0.201367	0.018386	2.5x	17.2x	42.9x
2	0.882646	0.415634	0.018906	0.748556	0.209100	0.018502	2.6x	16.7x	43.6x
3	0.883162	0.411658	0.018659	0.745755	0.208966	0.018490	2.6x	16.7x	43.8x
4	0.889796	0.435812	0.018879	0.747867	0.196900	0.018551	2.6x	16.9x	43.8x
5	0.894975	0.436644	0.018971	0.754183	0.200424	0.018864	2.6x	16.8x	43.6x
6	0.936334	0.446670	0.018889	0.770909	0.201013	0.018573	2.6x	17.3x	45.6x
7	0.926654	0.432985	0.019042	0.780512	0.204614	0.018468	2.7x	17.0x	45.5x
8	0.963230	0.440711	0.018379	0.776935	0.208775	0.018509	2.7x	17.6x	47.2x
9	0.984102	0.439865	0.018164	0.783728	0.204797	0.019651	2.7x	17.0x	46.7x
10	0.952158	0.483481	0.018409	0.792139	0.211839	0.018577	2.5x	18.8x	47.2x
11	0.941514	0.462088	0.019219	0.798964	0.222243	0.018470	2.5x	18.2x	46.2x
12	0.966859	0.477082	0.018587	0.806213	0.228187	0.018427	2.5x	19.1x	47.9x
13	0.965660	0.488126	0.018646	0.808243	0.233547	0.018672	2.5x	19.3x	47.5x
14	0.945624	0.466515	0.018691	0.816880	0.239647	0.019199	2.5x	18.6x	46.5x
15	0.948531	0.473074	0.018594	0.825037	0.229524	0.018498	2.5x	18.9x	47.8x
16	0.956140	0.475882	0.018393	0.834827	0.234294	0.018668	2.5x	19.2x	48.3x
17	0.993279	0.500866	0.018682	0.838600	0.244161	0.018586	2.5x	20.0x	49.2x
18	1.000484	0.494865	0.018844	0.840386	0.224214	0.018452	2.6x	19.3x	49.4x
19	1.012420	0.484566	0.018640	0.844682	0.231117	0.018640	2.6x	19.2x	49.8x
20	1.013143	0.516457	0.018562	0.852871	0.229046	0.020267	2.5x	19.2x	48.1x
21	1.051563	0.506119	0.018518	0.863403	0.237015	0.019099	2.6x	19.8x	50.9x
22	1.019868	0.509676	0.018471	0.872955	0.229416	0.018391	2.6x	20.1x	51.3x
23	1.060287	0.526115	0.018591	0.871520	0.232518	0.021446	2.5x	18.9x	48.3x
24	1.076128	0.531954	0.018791	0.885097	0.240145	0.019617	2.5x	20.1x	51.1x
25	1.035241	0.530093	0.018253	0.890998	0.251258	0.018747	2.5x	21.1x	52.1x
26	1.049211	0.540747	0.019244	0.897328	0.236867	0.018342	2.5x	20.7x	51.8x
27	1.042811	0.520949	0.018314	0.899140	0.253508	0.018466	2.5x	21.1x	52.8x
28	1.061754	0.535724	0.018700	0.901564	0.255119	0.018348	2.5x	21.3x	53.0x
29	1.047024	0.545151	0.018471	0.918865	0.252758	0.018562	2.5x	21.5x	53.1x
30	1.065870	0.545729	0.018622	0.921105	0.242500	0.018547	2.5x	21.2x	53.5x
Media:	1,009581733	0,4969599	0,01925983333	0,8514578333	0,2334613	0,0194202	2,6x	19,5x	49,7x



- **Resumen del análisis:** Podemos ver que los datos son estables, y se comparan de manera equivalente, por ende, podemos deducir que en test ha sido un éxito, dándonos unos resultados válidos, sin anomalías.
 - También podemos ver que los resultados del test usando instrucciones *SSE*, los resultados parecen muy estables, casi sin variación, esto significa que la talla del problema no ha variado lo suficiente para hacer fluctuar los resultados usando este set de instrucciones, y, por ende, el incremento temporal es casi irreconocible.

Unidad 2:

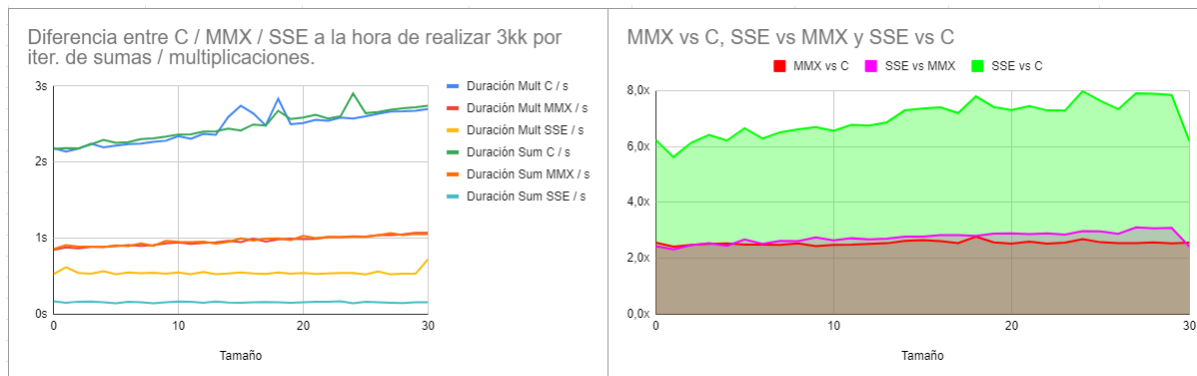
Tiempo que tarda el ordenador en hacer 3 000 000 de cálculos con arrays de tamaños variando de 100 a 130. (30 iteraciones * 3kk operaciones).									
Multiplicaciones				Sumas			Ganancia general (n veces más rápido)		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	1.261517	0.533911	0.077171	2.096770	0.380447	0.099437	3.7x	5.2x	19.0x
1	1.498716	0.523233	0.061431	2.060765	0.271970	0.053021	4.5x	6.9x	31.1x
2	1.274085	0.595658	0.054982	2.181975	0.331212	0.065134	3.7x	7.7x	28.8x
3	1.177968	0.617196	0.057272	2.150382	0.311758	0.099312	3.6x	5.9x	21.3x
4	1.322023	0.510825	0.066367	2.109292	0.277068	0.100340	4.4x	4.7x	20.6x
5	1.188478	0.563849	0.071782	2.072407	0.302417	0.099425	3.8x	5.1x	19.0x
6	1.391872	0.543903	0.095359	2.120338	0.343275	0.052119	4.0x	6.0x	23.8x
7	1.311910	0.558036	0.097797	2.021909	0.337949	0.054387	3.7x	5.9x	21.9x
8	1.310573	0.564473	0.063759	2.112381	0.308048	0.109868	3.9x	5.0x	19.7x
9	1.412661	0.605728	0.053118	2.058427	0.313878	0.052469	3.8x	8.7x	32.9x
10	1.603027	0.538484	0.083261	2.103152	0.311790	0.043003	4.4x	6.7x	29.4x
11	1.421584	0.576982	0.097708	2.152002	0.282091	0.062131	4.2x	5.4x	22.4x
12	1.436433	0.591808	0.098786	2.142945	0.307809	0.092434	4.0x	4.7x	18.7x
13	1.328844	0.596317	0.049237	2.224051	0.284855	0.060084	4.0x	8.1x	32.5x
14	1.281586	0.572693	0.063597	2.309443	0.285818	0.068403	4.2x	6.5x	27.2x
15	1.393017	0.652135	0.063220	2.382271	0.287598	0.061962	4.0x	7.5x	30.2x
16	1.663940	0.613651	0.054919	2.184910	0.326052	0.067368	4.1x	7.7x	31.5x
17	1.703063	0.625833	0.043179	2.267627	0.412510	0.059985	3.8x	10.1x	38.5x
18	1.750308	0.695871	0.051146	2.345234	0.331818	0.067761	4.0x	8.6x	34.4x
19	1.685073	0.680050	0.065758	2.293248	0.304581	0.100404	4.0x	5.9x	23.9x
20	1.726366	0.686332	0.092622	2.383466	0.296673	0.099729	4.2x	5.1x	21.4x
21	2.730218	0.706393	0.055870	2.300406	0.337723	0.101238	4.8x	6.6x	32.0x
22	2.428423	0.597972	0.069843	2.330849	0.352928	0.099130	5.0x	5.6x	28.2x
23	2.616661	0.613287	0.100185	2.127799	0.410507	0.117650	4.6x	4.7x	21.8x
24	1.960649	0.773378	0.097362	1.148168	0.450146	0.108648	2.5x	5.9x	15.1x
25	1.727826	1.466070	0.097155	1.270571	0.342500	0.094149	1.7x	9.5x	15.7x
26	1.515965	1.494910	0.099654	1.434703	0.352014	0.050027	1.6x	12.3x	19.7x
27	1.670319	1.505537	0.097281	1.434695	0.354789	0.060964	1.7x	11.8x	19.6x
28	1.427332	1.477776	0.099715	1.450168	0.342391	0.062266	1.6x	11.2x	17.8x
29	1.599564	0.690729	0.099706	1.549915	0.319568	0.073704	3.1x	5.8x	18.2x
30	2.256194	0.734134	0.099945	1.350587	0.457447	0.100286	3.0x	6.0x	18.0x
Media:	1.6692065	0.7502384667	0.07930623333	2.072361867	0.3444321	0.08122793333	3.8x	7.2x	25.1x



- Resumen del análisis:** Como podemos ver a primera vista, los datos parecen bastante alterados, es probable que durante el análisis otros programas hayan interferido en el uso del ordenador, creando estas anomalías que vemos. Por ende, los resultados no podrían llegar a ser válidos para realizar un estudio. aun así, decidimos dejarlos en el informe porque es un resultado curioso.

Unidad 3:

Tiempo que tarda el ordenador en hacer 3 000 000 de cálculos con arrays de tamaños variando de 100 a 130. (30 iteraciones * 3kk operaciones).									
Multiplicaciones				Sumas			Ganancia general (n veces más rápido)		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	2.183778	0.846644	0.527489	2.174837	0.855439	0.171094	2.6x	2.4x	6.2x
1	2.139889	0.878179	0.618734	2.184099	0.907637	0.150441	2.4x	2.3x	5.6x
2	2.177222	0.867263	0.544602	2.182160	0.889431	0.165753	2.5x	2.5x	6.1x
3	2.242178	0.885659	0.532057	2.235087	0.890687	0.165816	2.5x	2.5x	6.4x
4	2.196079	0.886268	0.566870	2.292016	0.883299	0.155973	2.5x	2.4x	6.2x
5	2.218166	0.894457	0.527160	2.255407	0.905161	0.145589	2.5x	2.7x	6.6x
6	2.236033	0.909704	0.551042	2.264035	0.894881	0.165218	2.5x	2.5x	6.3x
7	2.244062	0.898408	0.540557	2.302399	0.933451	0.158223	2.5x	2.6x	6.5x
8	2.266240	0.910068	0.546827	2.314816	0.899067	0.145760	2.5x	2.6x	6.6x
9	2.282018	0.933325	0.532495	2.336104	0.964982	0.157405	2.4x	2.8x	6.7x
10	2.343577	0.945905	0.550819	2.364272	0.952152	0.166559	2.5x	2.6x	6.6x
11	2.307033	0.926225	0.527096	2.367357	0.949144	0.162872	2.5x	2.7x	6.8x
12	2.371569	0.938340	0.555952	2.402711	0.956146	0.151485	2.5x	2.7x	6.7x
13	2.359793	0.946863	0.526870	2.406037	0.930224	0.167372	2.5x	2.7x	6.9x
14	2.594695	0.965370	0.536382	2.440098	0.951908	0.154230	2.6x	2.8x	7.3x
15	2.743285	0.948383	0.550916	2.415269	0.996712	0.150091	2.7x	2.8x	7.4x
16	2.642028	0.994330	0.538082	2.495500	0.969643	0.156477	2.6x	2.8x	7.4x
17	2.482982	0.954249	0.528930	2.480880	0.995572	0.161001	2.5x	2.8x	7.2x
18	2.834777	0.985774	0.550290	2.676183	0.999167	0.157007	2.8x	2.8x	7.8x
19	2.498613	0.993571	0.533921	2.567212	0.976074	0.149617	2.6x	2.9x	7.4x
20	2.513758	0.989739	0.541663	2.587906	1.031279	0.156991	2.5x	2.9x	7.3x
21	2.557383	0.990684	0.530065	2.621323	1.001367	0.165666	2.6x	2.9x	7.4x
22	2.543801	1.013652	0.536493	2.575487	1.016800	0.165784	2.5x	2.9x	7.3x
23	2.586543	1.014252	0.542033	2.603988	1.014283	0.170745	2.6x	2.8x	7.3x
24	2.575138	1.023353	0.542978	2.903970	1.018759	0.144063	2.7x	3.0x	8.0x
25	2.602740	1.021531	0.525161	2.645021	1.016355	0.162966	2.6x	3.0x	7.6x
26	2.635621	1.041186	0.563621	2.660454	1.038367	0.158743	2.5x	2.9x	7.3x
27	2.667374	1.039547	0.525339	2.688519	1.066396	0.152216	2.5x	3.1x	7.9x
28	2.668394	1.050931	0.533440	2.709981	1.042274	0.148396	2.6x	3.1x	7.9x
29	2.675332	1.070050	0.532137	2.722618	1.057380	0.156448	2.5x	3.1x	7.8x
30	2.699870	1.070598	0.721235	2.741242	1.053902	0.158293	2.6x	2.4x	6.2x
Media:	2.536359033	0.9944862667	0.5660418667	2.5538996	1.0019313	0.1632764667	2.6x	2.8x	7.2x

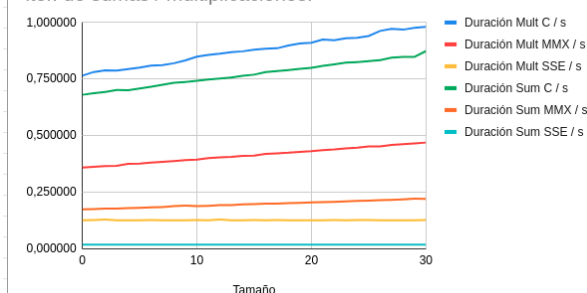


- Resumen del análisis:** Los datos a primera vista parecen concluyentes y correctos, con lo cual, serían válidos para realizar un estudio.
 - Podemos observar que la ganancia entre MMX vs C y SSE vs MMX es bastante similar (alrededor de ~2x a ~3x), esto significa que, en comparativa, SSE es ~6x más rápido que C, y MMX es ~3x más rápido que C. Un resultado un tanto peculiar, con una relación relativa entre MMX y SSE.

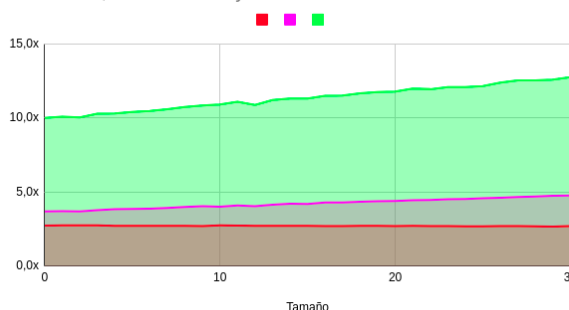
Unidad 4:

Tiempo que tarda el ordenador en hacer 3 000 000 de cálculos con arrays de tamaños variando de 100 a 130. (30 iteraciones * 3kk operaciones).									
Multiplicaciones				Sumas			Ganancia general (n veces más rápido)		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	0,765219	0,358976	0,126194	0,680457	0,174269	0,018550	2,7x	3,7x	10,0x
1	0,781493	0,362123	0,127350	0,687859	0,175949	0,018549	2,7x	3,7x	10,1x
2	0,788245	0,365130	0,129138	0,693112	0,177660	0,018600	2,7x	3,7x	10,0x
3	0,787843	0,367240	0,126311	0,701838	0,178068	0,018582	2,7x	3,8x	10,3x
4	0,794153	0,375028	0,126493	0,700530	0,179540	0,018682	2,7x	3,8x	10,3x
5	0,800634	0,376632	0,126267	0,708580	0,181594	0,018831	2,7x	3,8x	10,4x
6	0,809392	0,380549	0,127160	0,716618	0,183148	0,018642	2,7x	3,9x	10,5x
7	0,811266	0,383935	0,126539	0,724959	0,184368	0,018594	2,7x	3,9x	10,6x
8	0,819928	0,387716	0,126203	0,733250	0,188370	0,018560	2,7x	4,0x	10,7x
9	0,832843	0,392489	0,126292	0,737472	0,190973	0,018547	2,7x	4,0x	10,8x
10	0,848812	0,393870	0,127337	0,742376	0,188490	0,018536	2,7x	4,0x	10,9x
11	0,857274	0,400913	0,126255	0,748137	0,190250	0,018545	2,7x	4,1x	11,1x
12	0,862098	0,404170	0,129904	0,751891	0,192958	0,018562	2,7x	4,0x	10,9x
13	0,868948	0,406096	0,126480	0,756319	0,193666	0,018543	2,7x	4,1x	11,2x
14	0,872728	0,411167	0,126253	0,764219	0,196142	0,018546	2,7x	4,2x	11,3x
15	0,879498	0,411836	0,127137	0,769401	0,198040	0,018638	2,7x	4,2x	11,3x
16	0,883991	0,419824	0,126219	0,780577	0,199245	0,018563	2,7x	4,3x	11,5x
17	0,886001	0,421933	0,126700	0,785545	0,199652	0,018599	2,7x	4,3x	11,5x
18	0,898673	0,424570	0,126250	0,789845	0,202093	0,018537	2,7x	4,3x	11,7x
19	0,907207	0,428622	0,126283	0,795065	0,202949	0,018597	2,7x	4,4x	11,7x
20	0,911093	0,431460	0,126582	0,799715	0,205600	0,018722	2,7x	4,4x	11,8x
21	0,925086	0,436072	0,126233	0,808653	0,206018	0,018547	2,7x	4,4x	12,0x
22	0,922023	0,439074	0,126916	0,815254	0,207726	0,018637	2,7x	4,4x	11,9x
23	0,930520	0,443990	0,126644	0,822930	0,209397	0,018559	2,7x	4,5x	12,1x
24	0,932810	0,446477	0,126953	0,825150	0,211385	0,018543	2,7x	4,5x	12,1x
25	0,940624	0,452529	0,127227	0,829638	0,212821	0,018561	2,7x	4,6x	12,1x
26	0,962921	0,452221	0,126245	0,833386	0,215415	0,018778	2,7x	4,6x	12,4x
27	0,971764	0,458734	0,126299	0,844792	0,216223	0,018635	2,7x	4,7x	12,5x
28	0,968353	0,461838	0,126326	0,848141	0,218224	0,018583	2,7x	4,7x	12,5x
29	0,976924	0,465906	0,126411	0,848104	0,221451	0,018551	2,7x	4,7x	12,6x
30	0,980634	0,469482	0,126713	0,873259	0,220903	0,018635	2,7x	4,7x	12,8x
Media:	0,9059666	0,4276867333	0,1309771333	0,7972357333	0,2040862333	0,01921846667	2,8x	4,3x	11,7x

Diferencia entre C / MMX / SSE a la hora de realizar 3kk por iter. de sumas / multiplicaciones.



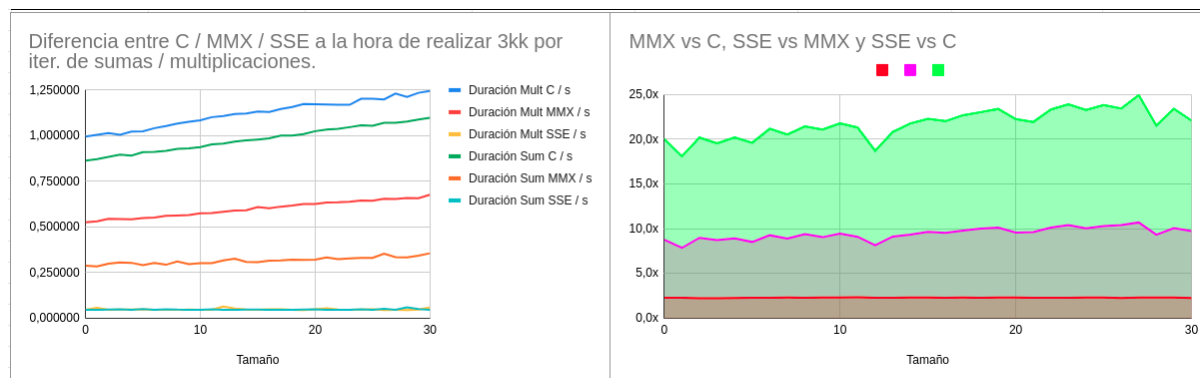
MMX vs C, SSE vs MMX y SSE vs C



- Resumen del análisis:** Los datos que obtenemos son bastante buenos, como podemos observar en comparación con el resto de resultado válidos, nos damos cuenta de que, aunque el tiempo sea diferente entre los ordenadores, el efecto sobre usar distintos sets de instrucciones es el mismo, aunque dependiendo del caso (si se usa otro tipo de algoritmo), es probable que SSE ralentice dichos cálculos.

Unidad 5:

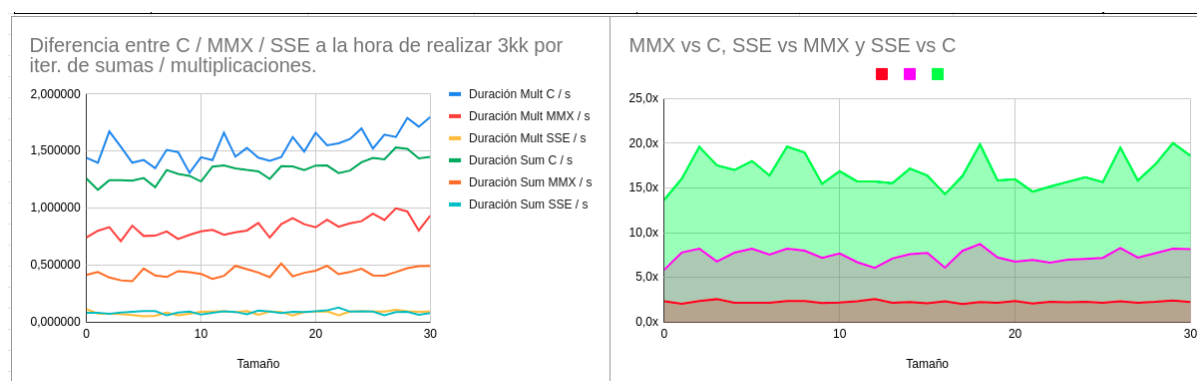
Tiempo que tarda el ordenador en hacer 3 000 000 de cálculos con arrays de tamaños variando de 100 a 130. (30 iteraciones * 3kk operaciones).									
Multiplicaciones				Sumas			Ganancia general (n veces más rápido)		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	0.995633	0.525479	0.046499	0.863202	0.287774	0.046285	2.3x	8.8x	20.0x
1	1.004850	0.531684	0.057305	0.871925	0.283701	0.046380	2.3x	7.9x	18.1x
2	1.014020	0.545145	0.046358	0.883860	0.299199	0.047623	2.2x	9.0x	20.2x
3	1.004996	0.543107	0.047779	0.896132	0.306524	0.049501	2.2x	8.7x	19.5x
4	1.022829	0.541868	0.048242	0.891501	0.303870	0.046478	2.3x	8.9x	20.2x
5	1.024674	0.549364	0.048519	0.910236	0.291546	0.050124	2.3x	8.5x	19.6x
6	1.041384	0.551929	0.045753	0.911781	0.303333	0.046447	2.3x	9.3x	21.2x
7	1.052771	0.560852	0.046086	0.917674	0.293134	0.049841	2.3x	8.9x	20.5x
8	1.066388	0.562711	0.045738	0.928368	0.311523	0.047328	2.3x	9.4x	21.4x
9	1.076235	0.565507	0.048237	0.930420	0.296347	0.046930	2.3x	9.1x	21.1x
10	1.084738	0.574963	0.046376	0.938360	0.302149	0.046490	2.3x	9.4x	21.8x
11	1.101913	0.576455	0.046578	0.952700	0.301373	0.049808	2.3x	9.1x	21.3x
12	1.107632	0.583129	0.063896	0.957411	0.316554	0.046474	2.3x	8.2x	18.7x
13	1.119417	0.590099	0.053937	0.968379	0.325961	0.046349	2.3x	9.1x	20.8x
14	1.121438	0.591219	0.048122	0.974287	0.309137	0.048131	2.3x	9.4x	21.8x
15	1.132373	0.608881	0.046292	0.979584	0.306753	0.048459	2.3x	9.7x	22.3x
16	1.129795	0.602839	0.049771	0.985699	0.315092	0.046277	2.3x	9.6x	22.0x
17	1.146035	0.611554	0.048618	1.000983	0.316365	0.046016	2.3x	9.8x	22.7x
18	1.157524	0.617531	0.047004	1.000819	0.320983	0.046719	2.3x	10.0x	23.0x
19	1.174187	0.625447	0.045320	1.008916	0.319952	0.048017	2.3x	10.1x	23.4x
20	1.172881	0.625378	0.049646	1.024989	0.321004	0.049093	2.3x	9.6x	22.3x
21	1.171336	0.633699	0.054112	1.033400	0.333879	0.046414	2.3x	9.6x	21.9x
22	1.169225	0.634969	0.047913	1.038368	0.323716	0.046707	2.3x	10.1x	23.3x
23	1.169652	0.638280	0.046365	1.046988	0.328026	0.046340	2.3x	10.4x	23.9x
24	1.202263	0.644714	0.047476	1.056997	0.331378	0.049568	2.3x	10.1x	23.3x
25	1.202385	0.643998	0.048674	1.054840	0.331209	0.046075	2.3x	10.3x	23.8x
26	1.199324	0.654606	0.045621	1.070799	0.353931	0.051229	2.3x	10.4x	23.4x
27	1.230876	0.652865	0.045879	1.071307	0.335388	0.046446	2.3x	10.7x	24.9x
28	1.212614	0.658483	0.045707	1.077835	0.333235	0.060615	2.3x	9.3x	21.5x
29	1.235170	0.658261	0.048229	1.088864	0.343046	0.051077	2.3x	10.1x	23.4x
30	1.245467	0.676863	0.059515	1.097922	0.356068	0.046566	2.3x	9.7x	22.1x
Media:	1.1596675	0.6193959667	0.0505189	1.014484867	0.3267383333	0.04966023333	2.4x	9.8x	22.5x



- *Resumen del análisis:* Este caso es similar al resto, pero podemos presenciar pequeños picos en distintos tamaños.

Unidad 6:

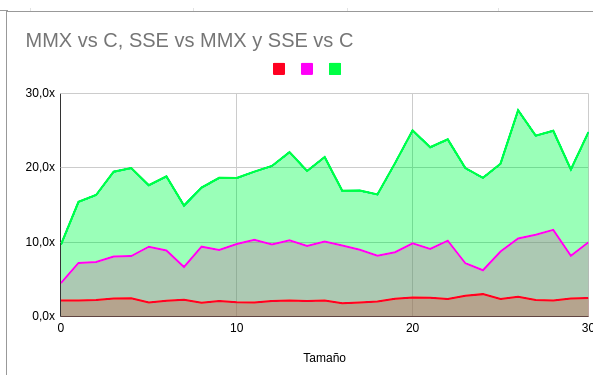
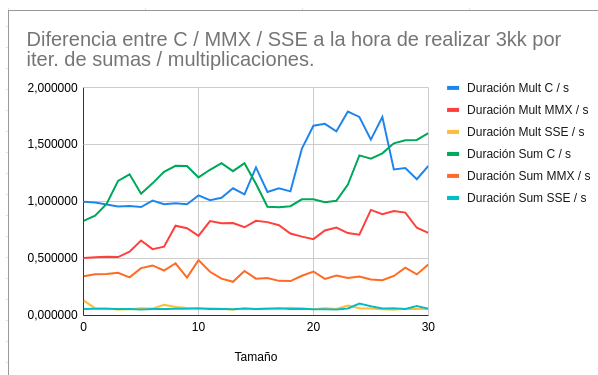
Tiempo que tarda el ordenador en hacer 3 000 000 de cálculos con arrays de tamaños variando de 100 a 130. (30 iteraciones * 3kk operaciones).									
Multiplicaciones				Sumas			Ganancia general (n veces más rápido)		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	1,442132	0,742745	0,115229	1,260258	0,415039	0,082158	2,3x	5,9x	13,7x
1	1,397752	0,802347	0,076271	1,160920	0,440340	0,083088	2,1x	7,8x	16,1x
2	1,673134	0,833255	0,073822	1,246495	0,392384	0,075039	2,4x	8,2x	19,6x
3	1,539041	0,709691	0,072463	1,244722	0,367564	0,086304	2,6x	6,8x	17,5x
4	1,399118	0,846663	0,063158	1,240848	0,359226	0,091966	2,2x	7,8x	17,0x
5	1,422920	0,756640	0,051313	1,264357	0,471244	0,097801	2,2x	8,2x	18,0x
6	1,350974	0,759153	0,056876	1,184243	0,409587	0,097615	2,2x	7,6x	16,4x
7	1,510824	0,796523	0,084443	1,334152	0,396791	0,060507	2,4x	8,2x	19,6x
8	1,491729	0,730456	0,061343	1,300911	0,448123	0,085917	2,4x	8,0x	19,0x
9	1,310118	0,765843	0,074605	1,283714	0,439863	0,093089	2,2x	7,2x	15,5x
10	1,447415	0,797489	0,090928	1,233994	0,424157	0,067928	2,2x	7,7x	16,9x
11	1,421377	0,809414	0,093210	1,364890	0,379852	0,083747	2,3x	6,7x	15,7x
12	1,661039	0,766153	0,096393	1,374477	0,407120	0,096559	2,6x	6,1x	15,7x
13	1,451492	0,788869	0,089613	1,349740	0,494587	0,090670	2,2x	7,1x	15,5x
14	1,528216	0,804448	0,097481	1,334774	0,466070	0,069268	2,3x	7,6x	17,2x
15	1,443046	0,870853	0,065896	1,322887	0,436140	0,102719	2,1x	7,8x	16,4x
16	1,414473	0,742339	0,093443	1,256247	0,395007	0,093081	2,3x	6,1x	14,3x
17	1,448384	0,859511	0,090587	1,368934	0,515879	0,081766	2,0x	8,0x	16,3x
18	1,623808	0,911984	0,059517	1,365403	0,402441	0,090828	2,3x	8,7x	19,9x
19	1,496603	0,859630	0,089118	1,333574	0,433620	0,089444	2,2x	7,2x	15,8x
20	1,661497	0,832475	0,093772	1,374005	0,452999	0,096232	2,4x	6,8x	16,0x
21	1,550971	0,898491	0,096273	1,374149	0,495110	0,104038	2,1x	7,0x	14,6x
22	1,568169	0,837973	0,061356	1,307970	0,422020	0,128083	2,3x	6,7x	15,2x
23	1,606768	0,867594	0,094033	1,329952	0,440359	0,093119	2,2x	7,0x	15,7x
24	1,698738	0,885898	0,095305	1,403041	0,468715	0,095958	2,3x	7,1x	16,2x
25	1,522949	0,951833	0,094535	1,440420	0,410913	0,094650	2,2x	7,2x	15,7x
26	1,644906	0,896766	0,094907	1,428398	0,408320	0,062452	2,4x	8,3x	19,5x
27	1,623678	0,999407	0,109926	1,532070	0,439796	0,089342	2,2x	7,2x	15,8x
28	1,790050	0,971511	0,096441	1,520156	0,474654	0,090961	2,3x	7,7x	17,7x
29	1,714371	0,804426	0,091323	1,437577	0,491053	0,065988	2,4x	8,2x	20,0x
30	1,799947	0,935115	0,093287	1,449257	0,494347	0,081428	2,3x	8,2x	18,6x
Media:	1,5685213	0,8611831667	0,0872269	1,380751167	0,4497773333	0,09072483333	2,3x	7,7x	17,4x



- *Resumen del análisis:* El siguiente caso es peculiar, ya que los picos son bastante intensos dependiendo de la talla. Es probable que el propio sistema operativo estuviese ejecutando algún servicio de fondo.

Unidad 7:

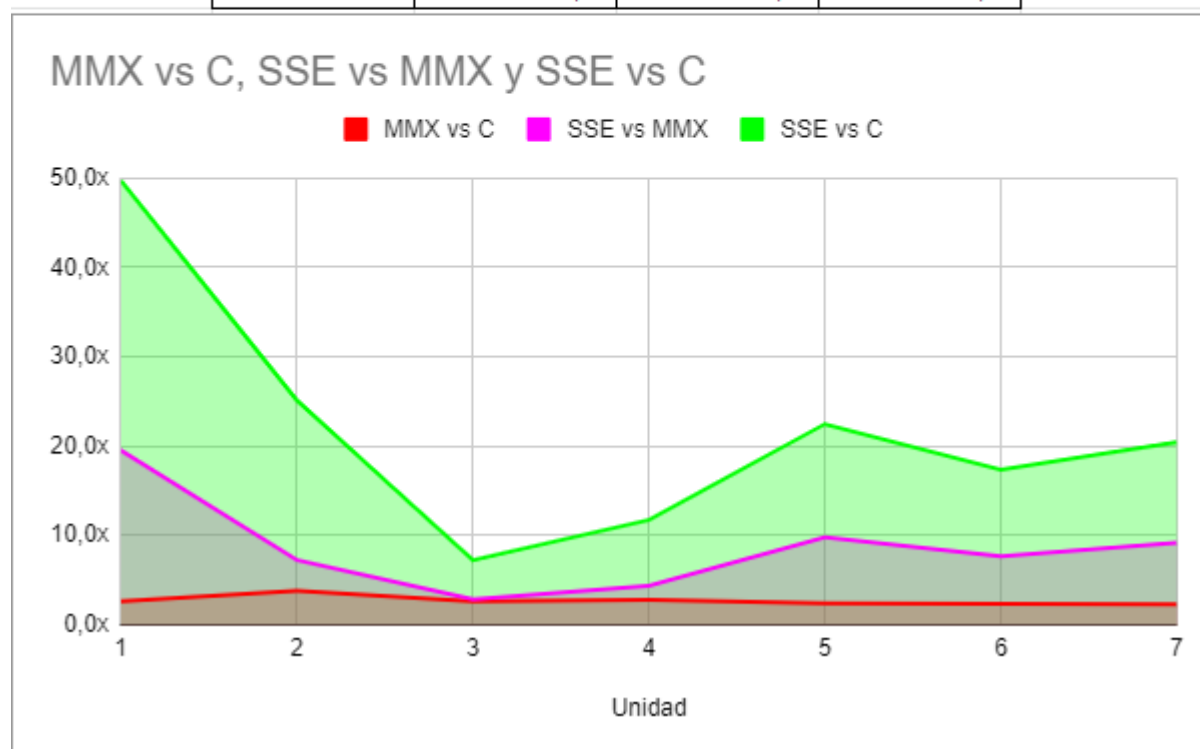
Multiplicaciones				Sumas			Ganancia general (n veces más rapido)		
Tamaño	Duración Mult C / s	Duración Mult MMX / s	Duración Mult SSE / s	Duración Sum C / s	Duración Sum MMX / s	Duración Sum SSE / s	MMX vs C	SSE vs MMX	SSE vs C
0	0.999503	0.505320	0.132436	0.831424	0.343724	0.057387	2.2x	4.5x	9.6x
1	0.992460	0.511927	0.061772	0.877767	0.361514	0.059572	2.1x	7.2x	15.4x
2	0.974428	0.514956	0.061779	0.980048	0.363338	0.057957	2.2x	7.3x	16.3x
3	0.958185	0.513478	0.052264	1.182578	0.374555	0.057793	2.4x	8.1x	19.5x
4	0.962586	0.561016	0.054295	1.240585	0.335247	0.056097	2.5x	8.1x	20.0x
5	0.952936	0.659141	0.062412	1.070534	0.416137	0.052266	1.9x	9.4x	17.8x
6	1.010254	0.582902	0.059099	1.161302	0.439494	0.056132	2.1x	8.9x	18.8x
7	0.978247	0.604185	0.093174	1.262826	0.395072	0.057173	2.2x	6.6x	14.9x
8	0.986501	0.788482	0.074100	1.315710	0.458623	0.058856	1.8x	9.4x	17.3x
9	0.977788	0.766285	0.064649	1.312785	0.333282	0.058246	2.1x	8.9x	18.6x
10	1.055222	0.699715	0.059231	1.213635	0.486316	0.062618	1.9x	9.7x	18.6x
11	1.012466	0.829029	0.060338	1.280585	0.384134	0.057481	1.9x	10.3x	19.5x
12	1.034878	0.810904	0.059759	1.338076	0.323839	0.057514	2.1x	9.7x	20.2x
13	1.117809	0.811675	0.053275	1.268164	0.295829	0.054746	2.2x	10.3x	22.1x
14	1.064427	0.775127	0.062098	1.338037	0.390544	0.060765	2.1x	9.5x	19.6x
15	1.301916	0.832310	0.057430	1.157414	0.323111	0.057318	2.1x	10.1x	21.4x
16	1.084768	0.820289	0.060778	0.954990	0.329821	0.059877	1.8x	9.5x	16.9x
17	1.118142	0.792622	0.058680	0.952096	0.304025	0.063652	1.9x	9.0x	16.9x
18	1.090764	0.719805	0.068536	0.960582	0.301773	0.056539	2.0x	8.2x	16.4x
19	1.467090	0.692976	0.061220	1.020826	0.348948	0.059491	2.4x	8.6x	20.6x
20	1.667480	0.671006	0.053589	1.020376	0.386180	0.053873	2.5x	9.8x	25.0x
21	1.684704	0.747118	0.063972	0.995500	0.321023	0.053840	2.5x	9.1x	22.7x
22	1.619353	0.773241	0.057317	1.007559	0.351179	0.053047	2.3x	10.2x	23.8x
23	1.792017	0.724378	0.086510	1.150151	0.329629	0.060925	2.8x	7.1x	20.0x
24	1.744648	0.709180	0.064079	1.406778	0.341255	0.104953	3.0x	6.2x	18.6x
25	1.545842	0.927806	0.060998	1.378315	0.315279	0.081393	2.4x	8.7x	20.5x
26	1.744668	0.890026	0.053475	1.424701	0.309309	0.060844	2.6x	10.5x	27.7x
27	1.283573	0.918013	0.052380	1.513085	0.347257	0.062691	2.2x	11.0x	24.3x
28	1.295582	0.903847	0.057378	1.540308	0.419519	0.056256	2.1x	11.6x	25.0x
29	1.197820	0.770840	0.056589	1.543038	0.360952	0.082093	2.4x	8.2x	19.8x
30	1.313895	0.726318	0.059488	1.601719	0.448411	0.058172	2.5x	10.0x	24.8x
Media:	1.267665067	0.7517972333	0.0661033333	1.243383133	0.3746439667	0.06298556667	2.3x	9.2x	20.4x



- *Resumen del análisis:* Como en caso de la unidad 2, los resultados parecen bastante inconsistentes, muy probable que el ordenador estuviese ejecutando alguna aplicación que cargase el ordenador, por ende, estos no serían válidos para un estudio.

Comparativa entre ganancias:

Ganancia general (n veces más rapido)			
Unidad	MMX vs C	SSE vs MMX	SSE vs C
1	2,6x	19,5x	49,7x
2	3,8x	7,2x	25,1x
3	2,6x	2,8x	7,2x
4	2,8x	4,3x	11,7x
5	2,4x	9,8x	22,5x
6	2,3x	7,7x	17,4x
7	2,3x	9,2x	20,4x



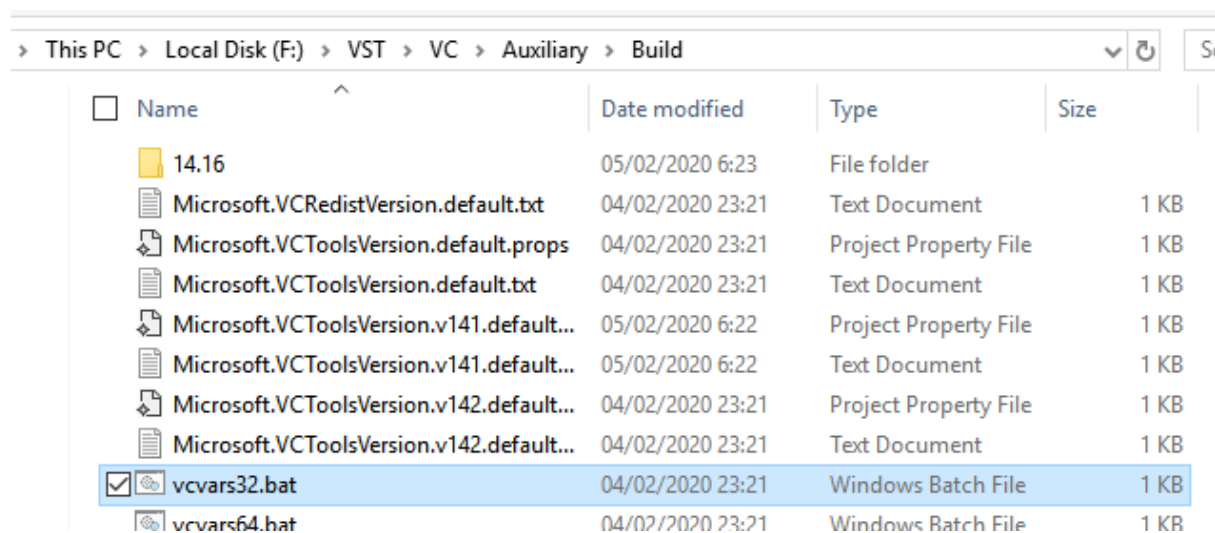
Conclusión General:

Como podemos comprobar en las diferentes ejecuciones del benchmark, SSE aumenta considerablemente la velocidad de cálculo de datos paralelizables, MMX mejora el rendimiento general de la cualquier operación aritmética debido a que trabajamos directamente en ensamblador.

SPEC2000, como ejecutar, evaluación de resultados, y comparativas.

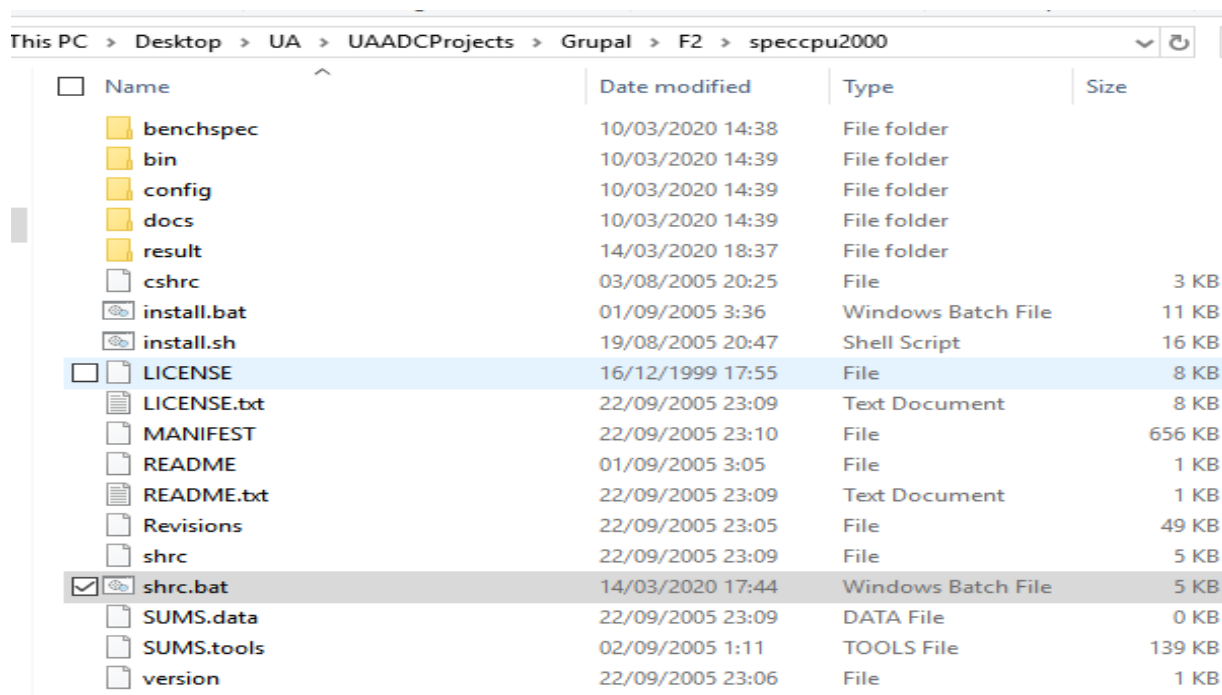
Para ejecutar el análisis con los benchmarks de speccpu2000, tenemos que realizar los siguientes pasos:

1. Copiaremos la carpeta speccpu2000 desde la raíz de los ordenadores del laboratorio a los ordenadores que queremos analizar.
2. Buscaremos el archivo vcvars32.bat en la carpeta de instalación de nuestro visual studio, esta suele estar ubicada en la subcarpeta vc/build y copiaremos la ruta del archivo.



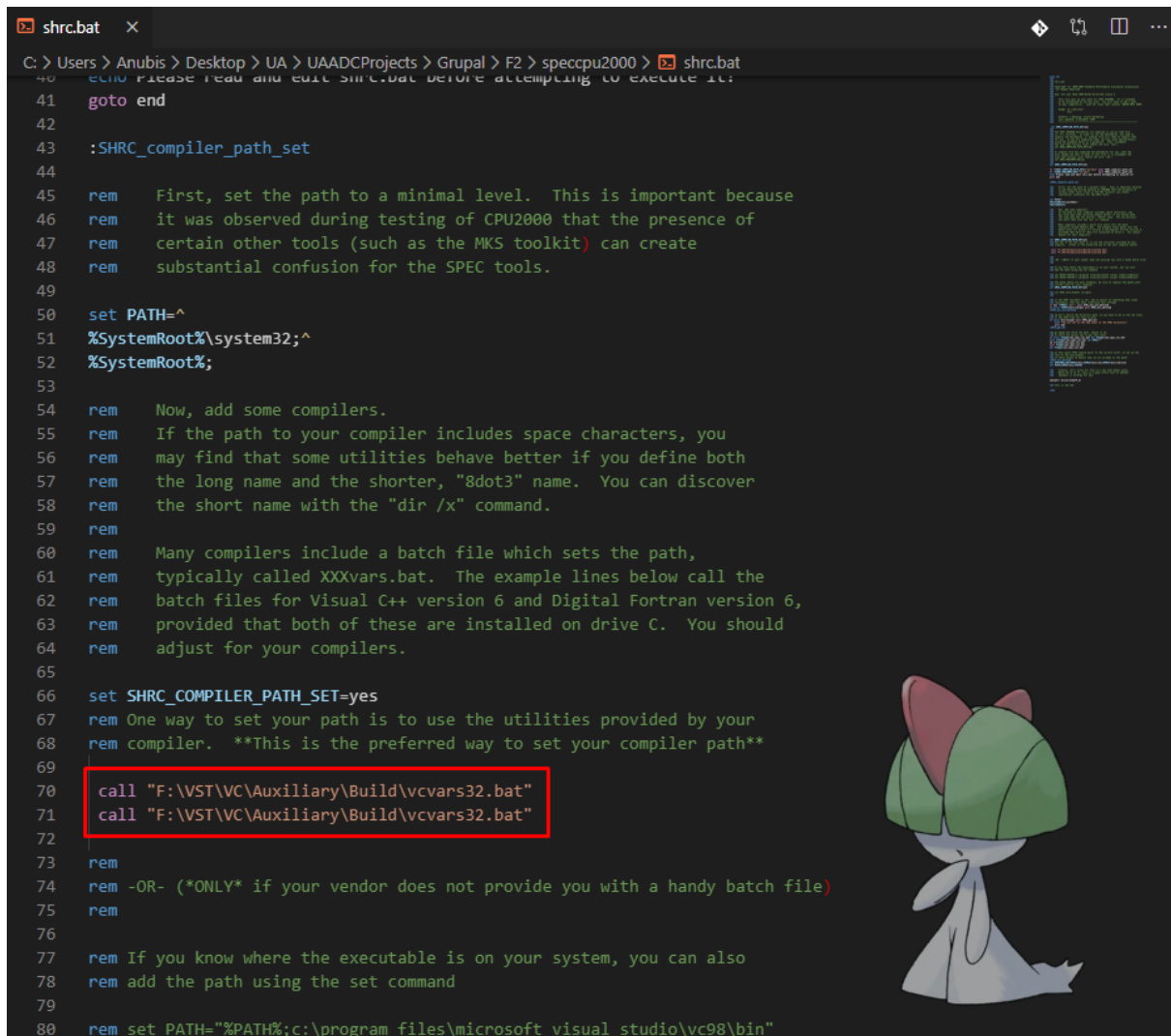
<input type="checkbox"/> Name	Date modified	Type	Size
14.16	05/02/2020 6:23	File folder	
Microsoft.VCRedistVersion.default.txt	04/02/2020 23:21	Text Document	1 KB
Microsoft.VCToolsVersion.default.props	04/02/2020 23:21	Project Property File	1 KB
Microsoft.VCToolsVersion.default.txt	04/02/2020 23:21	Text Document	1 KB
Microsoft.VCToolsVersion.v141.default...	05/02/2020 6:22	Project Property File	1 KB
Microsoft.VCToolsVersion.v141.default...	05/02/2020 6:22	Text Document	1 KB
Microsoft.VCToolsVersion.v142.default...	04/02/2020 23:21	Project Property File	1 KB
Microsoft.VCToolsVersion.v142.default...	04/02/2020 23:21	Text Document	1 KB
<input checked="" type="checkbox"/> vcvars32.bat	04/02/2020 23:21	Windows Batch File	1 KB
vcvars64.bat	04/02/2020 23:21	Windows Batch File	1 KB

3. Volveremos a la carpeta speccpu2000 que hemos copiado previamente y editaremos el archivo shrc.bat



<input type="checkbox"/> Name	Date modified	Type	Size
benchspec	10/03/2020 14:38	File folder	
bin	10/03/2020 14:39	File folder	
config	10/03/2020 14:39	File folder	
docs	10/03/2020 14:39	File folder	
result	14/03/2020 18:37	File folder	
cshrc	03/08/2005 20:25	File	3 KB
install.bat	01/09/2005 3:36	Windows Batch File	11 KB
install.sh	19/08/2005 20:47	Shell Script	16 KB
<input type="checkbox"/> LICENSE	16/12/1999 17:55	File	8 KB
LICENSE.txt	22/09/2005 23:09	Text Document	8 KB
MANIFEST	22/09/2005 23:10	File	656 KB
README	01/09/2005 3:05	File	1 KB
README.txt	22/09/2005 23:09	Text Document	1 KB
Revisions	22/09/2005 23:05	File	49 KB
shrc	22/09/2005 23:09	File	5 KB
<input checked="" type="checkbox"/> shrc.bat	14/03/2020 17:44	Windows Batch File	5 KB
SUMS.data	22/09/2005 23:09	DATA File	0 KB
SUMS.tools	02/09/2005 1:11	TOOLS File	139 KB
version	22/09/2005 23:06	File	1 KB

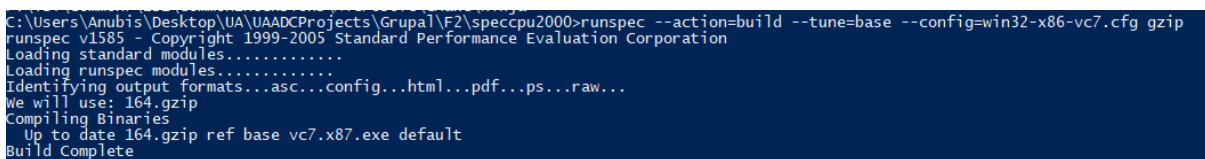
- En el archivo que vamos a editar, habrá unas rutas que se estarán ejecutando con el comando call. Esas rutas las cambiaremos por la ruta copiada en el paso anterior.



```
shrc.bat
C:\Users\Anubis\Desktop\UA\UAADCPProjects\Grupal\F2\speccpu2000> shrc.bat
40 echo Please read and edit shrc.bat before attempting to execute it:
41 goto end
42
43 :SHRC_compiler_path_set
44
45 rem First, set the path to a minimal level. This is important because
46 rem it was observed during testing of CPU2000 that the presence of
47 rem certain other tools (such as the MKS toolkit) can create
48 rem substantial confusion for the SPEC tools.
49
50 set PATH=^
51 %SystemRoot%\system32;^
52 %SystemRoot%;
53
54 rem Now, add some compilers.
55 rem If the path to your compiler includes space characters, you
56 rem may find that some utilities behave better if you define both
57 rem the long name and the shorter, "8dot3" name. You can discover
58 rem the short name with the "dir /x" command.
59 rem
60 rem Many compilers include a batch file which sets the path,
61 rem typically called XXXvars.bat. The example lines below call the
62 rem batch files for Visual C++ version 6 and Digital Fortran version 6,
63 rem provided that both of these are installed on drive C. You should
64 rem adjust for your compilers.
65
66 set SHRC_COMPILER_PATH_SET=yes
67 rem One way to set your path is to use the utilities provided by your
68 rem compiler. **This is the preferred way to set your compiler path**
69
70 call "F:\VST\VC\Auxiliary\Build\vcvars32.bat"
71 call "F:\VST\VC\Auxiliary\Build\vcvars32.bat"
72
73 rem
74 rem -OR- (*ONLY* if your vendor does not provide you with a handy batch file)
75 rem
76
77 rem If you know where the executable is on your system, you can also
78 rem add the path using the set command
79
80 rem set PATH="%PATH%;c:\program files\microsoft visual studio\vc98\bin"
```

- Abriremos una terminal en la carpeta raíz de speccpu2000 y ejecutaremos el archivo shrc.bat, esperaremos a que acabe de procesarse y ejecutaremos los siguientes dos comandos:

runspec --action=build --tune=base --config=win32-x86-vc7.cfg gzip



```
C:\Users\Anubis\Desktop\UA\UAADCPProjects\Grupal\F2\speccpu2000>runspec --action=build --tune=base --config=win32-x86-vc7.cfg gzip
runspec v1585 - Copyright 1999-2005 Standard Performance Evaluation Corporation
Loading standard modules.....
Loading runspec modules.....
Identifying output formats...asc...config...html...pdf...ps...raw...
We will use: 164.gzip
Compiling Binaries
Up to date 164.gzip ref base vc7.x87.exe default
Build Complete
```

runspec --reportable --config=win32-x86-vc7.cfg -T base int

```
C:\Users\Anubis\Desktop\UA\UAADCPProjects\Grupal\F2\speccpu2000>runspec --reportable --config=win32-x86-vc7.cfg -T base int
runspec v1585 - Copyright 1999-2005 Standard Performance Evaluation Corporation
Loading standard modules.....
Loading runspec modules.....
Identifying output formats...asc...config...html...pdf...ps...raw...
We will use: 164.gzip, 175.vpr, 176.gcc, 181.mcf, 186.crafty, 197.parser, 252.eon, 253.perlbmk, 254.gap, 255.vortex, 256.bzip2, 300.twolf
Compiling Binaries
Up to date 164.gzip ref base vc7.x87.exe default
Up to date 175.vpr ref base vc7.x87.exe default
Up to date 176.gcc ref base vc7.x87.exe default
Up to date 181.mcf ref base vc7.x87.exe default
Up to date 186.crafty ref base vc7.x87.exe default
Up to date 197.parser ref base vc7.x87.exe default
Up to date 252.eon ref base vc7.x87.exe default
Building 253.perlbmk ref base vc7.x87.exe default
Error with make 'specmake build > make.out > make.err': check file 'C:\Users\Anubis\Desktop\UA\UAADCPProjects\Grupal\F2\speccpu2000\benchspec\CINT2000\253.perlbmk\run\00000002\make.err'
Error with make!
Error building 253.perlbmk
Up to date 254.gap ref base vc7.x87.exe default
Up to date 255.vortex ref base vc7.x87.exe default
Up to date 256.bzip2 ref base vc7.x87.exe default
Up to date 300.twolf ref base vc7.x87.exe default
Setting Up Run Directories
Setting up 164.gzip ref base vc7.x87.exe default: created
Setting up 175.vpr ref base vc7.x87.exe default: created
Setting up 176.gcc ref base vc7.x87.exe default: created
Setting up 181.mcf ref base vc7.x87.exe default: created
Setting up 186.crafty ref base vc7.x87.exe default: created
Setting up 197.parser ref base vc7.x87.exe default: created
Setting up 252.eon ref base vc7.x87.exe default: created
Setting up 254.gap ref base vc7.x87.exe default: created
Setting up 255.vortex ref base vc7.x87.exe default: created
Setting up 256.bzip2 ref base vc7.x87.exe default: created
Setting up 300.twolf ref base vc7.x87.exe default: created
Running Benchmarks
Running 164.gzip ref base vc7.x87.exe default
Running 164.gzip ref base vc7.x87.exe default
Running 175.vpr ref base vc7.x87.exe default
Running 175.vpr ref base vc7.x87.exe default
Running 176.gcc ref base vc7.x87.exe default
Running 176.gcc ref base vc7.x87.exe default
Running 181.mcf ref base vc7.x87.exe default
Running 181.mcf ref base vc7.x87.exe default
Running 186.crafty ref base vc7.x87.exe default
Running 186.crafty ref base vc7.x87.exe default
Running 197.parser ref base vc7.x87.exe default
Running 197.parser ref base vc7.x87.exe default
Running 252.eon ref base vc7.x87.exe default
Running 252.eon ref base vc7.x87.exe default
Running 254.gap ref base vc7.x87.exe default
Running 254.gap ref base vc7.x87.exe default
Running 254.gap ref base vc7.x87.exe default
Running 255.vortex ref base vc7.x87.exe default
Running 255.vortex ref base vc7.x87.exe default
Running 256.bzip2 ref base vc7.x87.exe default
Running 256.bzip2 ref base vc7.x87.exe default
Running 300.twolf ref base vc7.x87.exe default
Running 300.twolf ref base vc7.x87.exe default
Running 300.twolf ref base vc7.x87.exe default
Error: 1x253.perlbmk
Success: 3x164.gzip 3x175.vpr 3x176.gcc 3x181.mcf 3x186.crafty 3x197.parser 3x252.eon 3x254.gap 3x255.vortex 3x256.bzip2 3x300.twolf
Producing Reports
mach: default
ext: vc7.x87.exe
set: int
size: ref
Format: raw -> C:\Users\Anubis\Desktop\UA\UAADCPProjects\Grupal\F2\speccpu2000\result\CINT2000_004.raw
Format: ASCII -> C:\Users\Anubis\Desktop\UA\UAADCPProjects\Grupal\F2\speccpu2000\result\CINT2000_004.asc
set: fp
runspec finished
```

Una vez que el último comando se haya acabado de ejecutar, en la raíz de la carpeta de spec cpu 2000 habrá una carpeta nuevamente generada llamada result en la cual, habrá dos archivos con extensión. raw y. asc, estos archivos son los resultados del análisis realizado anteriormente con speccpu2000 Modalidad CINT (Rendimiento de enteros intensivos en cómputo).

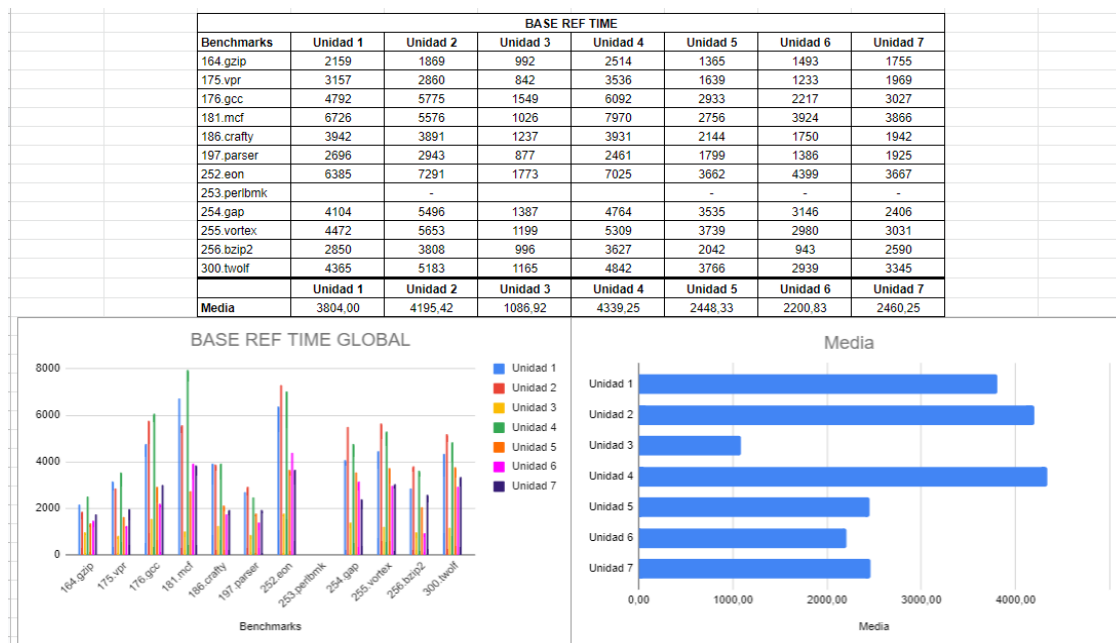
This PC > Desktop > UA > UAADCPProjects > Grupal > F2 > speccpu2000 > result					Search
<input type="checkbox"/>	Name	Date modified	Type	Size	
	images	10/03/2020 14:39	File folder		
<input checked="" type="checkbox"/>	CINT2000.004.asc	14/03/2020 18:12	ASC File	7 KB	
<input checked="" type="checkbox"/>	CINT2000.004.raw	14/03/2020 18:12	RAW File	25 KB	
	log.001	10/03/2020 14:25	Archivo WinRAR	11 KB	
	log.002	10/03/2020 14:39	002 File	357 KB	
	log.003	14/03/2020 17:48	003 File	1 KB	
	log.004	14/03/2020 18:12	004 File	78 KB	
	log.005	14/03/2020 18:37	005 File	78 KB	
	log.lock	14/03/2020 18:14	LOCK File	0 KB	

El archivo “. asc” nos proporcionará toda la información que necesitaremos, los resultados de la compilación y ejecución de los benchmarks en el ordenador.

Desarrollo de los resultados:

Se debe mencionar que debido a que en todas las unidades que hemos ejecutado SPEC200, el benchmark “perlbnk” ha fallado, lo omitiremos de nuestro análisis.

A continuación, mostraremos el comparativo general entre los diferentes ordenadores ejecutando el mismo benchmark.



Una de las conclusiones que podemos tomar de estos resultados es el que la unidad 4 parece ser la más potente mientras que la unidad 3 es la más lenta.

Si comparamos estos resultados con nuestros benchmarks podremos ver que:



Los resultados son conclusivos, siendo la unidad 4 la más rápida, y la unidad 3 la más lenta.

Recursos y fuentes usadas para la fase:

- Libros y listas de comandos:

<https://books.google.es/books?id=bt3ZjeB4v9gC&printsec=frontcover&hl=es#v=onepage&q&f=false>

<https://books.google.es/books?id=c3SlgrqMid4C&printsec=frontcover&hl=es#v=onepage&q&f=false>

https://books.google.es/books?id=C3_WIQOYE2EC&printsec=frontcover&hl=es#v=onepage&q&f=false

<https://books.google.es/books?id=FGxOp0lAljUC&printsec=frontcover&hl=es#v=onepage&q&f=false>

<http://softpixel.com/~cwright/programming/simd/>

https://docs.oracle.com/cd/E18752_01/html/817-5477/eojdc.html

https://docs.oracle.com/cd/E26502_01/html/E28388/eojde.html

<http://www.jegerlehner.ch/intel/IntelCodeTable.pdf>

https://cs.brown.edu/courses/cs033/docs/guides/x64_cheatsheet.pdf

https://en.wikibooks.org/wiki/X86_Assembly/SSE

- Plataformas de coordinación.

➔ Google drive.

➔ [Trello](#).

➔ Discord.