ETHZ Department of Computer Science

Computer Systems Analysis and Benchmarking
37−235 WS 1999/2000

Performance Benefits of pgcc over gcc on the x86/Linux Platform

Written in March 2000

Lecturer: Prof. T. Stricker

Authors: R. Scheidegger
B. Wymann

Summary

This study has proved that the code produced by pgcc is overall faster than that of the standard gcc, even if you use the optimisation options introduced in gcc 2.95.2. The performance advantage, is, however, in most cases not that big, and in some cases, the pgcc produces even slower code than the standard gcc. The largest gains were achieved with the FPU−benchmarks on a processor with a P6 core (Celeron/Pentium II). It is not recommended to use the pgcc version available today in systems where reliability is of vital importance, because it might have more bugs than the standard gcc. But if speed is also an important issue, you should give it a try. Considering that the speed gains achieved likely match those of a one speed grade faster processor, the pgcc is really a cheap solution.

Table of Contents

1 Introduction

1.1 Goals

The goal of this study is to determine if the Pentium gcc 2.95.3 (abbreviated pgcc) produces faster code than the standard gcc (2.95.2 and 2.7.2.3), and if so, how much faster the code is. There exist Pentium optimised Linux distributions which claim to be 5% to 30% faster than normal ones (Mandrake Linux), just because they are compiled with the pgcc. The pgcc is made by the Pentium Compiler Group (http://goof.com/pcg/) and is freely available. It should run on all platforms on which the original gcc runs, however it will only improve code generated for the x86 platform. It has specifically optimising options for the Intel P5 core (Classic Pentium and Pentium MMX), Intel P6 core (Pentium Pro, Pentium II, Pentium III, Celeron) and the AMD K6 core (K6, K6/2, K6/III). An AMD K7 (Athlon) optimisation switch doesn't exist (yet), so we focused on performance on K6 and P6 cores. We didn't do any performance tests on P5 cores neither because we weren't interested that much in the performance of "old" systems (if you think your programs run too slow on a Pentium, you probably should buy a new computer rather than installing a new compiler). The pgcc does not support the additional instruction sets like AMD's 3dnow or Intel's ISSE. It does support MMX, however we didn't test it, because the Pentium Compiler Group says that this option is somewhat unlikely to produce faster code. It seems to be a problem in general to use any SIMD (Single Instruction Multiple Data) instruction set efficiently in a compiler.

1.2 System Boundaries

We decided to use Linux as the OS because of two reasons: First, the OS should have some market share, and second, the compiler which most people will use with the OS should be the gcc. Of course Windows is installed on more x86 systems than Linux, but most people will use a commercial C−compiler like Microsoft Visual C++ or the Intel Vtune C−compiler, and not gcc. Other operating systems (like OS/2, BeOS, FreeBSD, other UNIX systems) might use gcc as their default compiler, but their market share is even lower. We used a standard Linux distribution, because this is what most people will use, nowadays almost nobody builds a Linux systems from scratch. We chose SuSE Linux 6.2 because this is what we use personally, but this decision should have little effect on performance. Finally, SPEC CPU95 was used to measure the performance. We chose it because its different workloads should give a good hint of real world performance. If, for example, a benchmark waswould have been used whose workloads would only consist of compression algorithms, there would have been a big performance increase with pgcc (it is known that bzip and gzip will be about 20−30% faster if compiled with pgcc). But our goal is to find out the overall performance increase, because pgcc claims to be faster with almost any code, and for this reason a benchmark with rather distinctive workloads is better suited. We didn't care about compile time, because most people compile an application just once and run it many times afterwards. The size of the binaries generated was also ignored, because we were interested in execution time, not in code size. For the floating point part of the

measurement (SPEC CFP95) we used the fortran−to−c package (f77−f2c), not a native fortran compiler for obvious reasons.

2 Evaluation

We chose measurement as the evaluation technique because we wanted to find out if it's worth to use pgcc instead of gcc. Measurement delivers the most exact results here, because all the hardware and software exists already. Furthermore, analytical modelling would have been ways too complex, because compilers and hardware aren't that simple to model. Similar reasons apply to simulation.

2.1 Factors

Our factors are the compiler versions used and the parameters for these compilers used. Another factor is the computer used. We wanted to compare the newest pgcc compiler with the most widely used gcc versions, which were 2.7.2.3 (rather outdated, but known as very reliable, which might be the reason why many people still use it) and 2.95.2 (probably not as widely used, but the newest stable release). We decided to use only few common compiler options, mainly the −Ox, −march=y and −mz (only if the compiler didn't support the −march option), because most packages of distributions are just compiled with −O2, and nobody wants to specify many different parameters for every program to reach optimal performance.
For the pgcc, we always used the compiler switches which should give the maximum performance, that is, −O6 and −march=k6 on K6 systems and −O6 and −march=pentiumpro on P6 core systems (Pentium II, Pentium III, Pentium Pro and Celeron). We used the same switches with the gcc 2.95.2 too if available (for K6 systems, we used −march=i486, because this gcc doesn't support the −march=k6 option). The gcc 2.7.2.3 doesn't support −O6, so we used −O2 (though it would support −O3, which is the same as −O2 plus function inlining). It doesn't support any −march option, so we decided to use the −mx option instead (only −m486 was used, because −m386 is the default and others are not available). To compare the new gcc 2.95.2 to the old gcc, we ran the benchmark with −O2 on this gcc too. Just to see how fast the code generated would be with −O1 (the same as −O) we tested this with the gcc 2.7.2.3 too.

Experiment	***Version***	***Switches***
1	gcc 2.7.2.3	−O1
2	gcc 2.7.2.3	−O2
3	gcc 2.7.2.3	−O2 −m486
4	gcc 2.95.2	−O2
5	gcc 2.95.2	−O6 −march=pentiumpro or −march=i486 (for K6)
6	pgcc 2.95.3	−O6 −march=pentiumpro or −march=k6 (for K6)

The −m switch rearranges the code for a specific cpu (e.g. −m486 will arrange the

code for a 486). The resulting code still executes on any x86 processor.
The −march switch implies the −m switch and additionally the compiler will use the new CPU−specific instructions. Code generated with this option will not run on processors which do not support these instructions (e.g. code generated with −march=pentium would still run on a Pentium II, but not on a 486).

2.2 Parameters

Our parameters are mostly hardware−related, because we used the same Linux distribution on all systems, all libraries used are also identical. There are of course other parameters, but the only one which could influence the measurements is the linux kernel version used. All systems were running in multi−user mode, however there were no other users logged in. There was no network traffic, too. All normal system services like httpd, nfsd, rlogind, sshd, ftpd were running except for the cron daemon, which we stopped for reproducibility reasons (on SuSE Linux systems, from time to time severe load is generated by the cron scripts).

2.2.1 Machine Descriptions

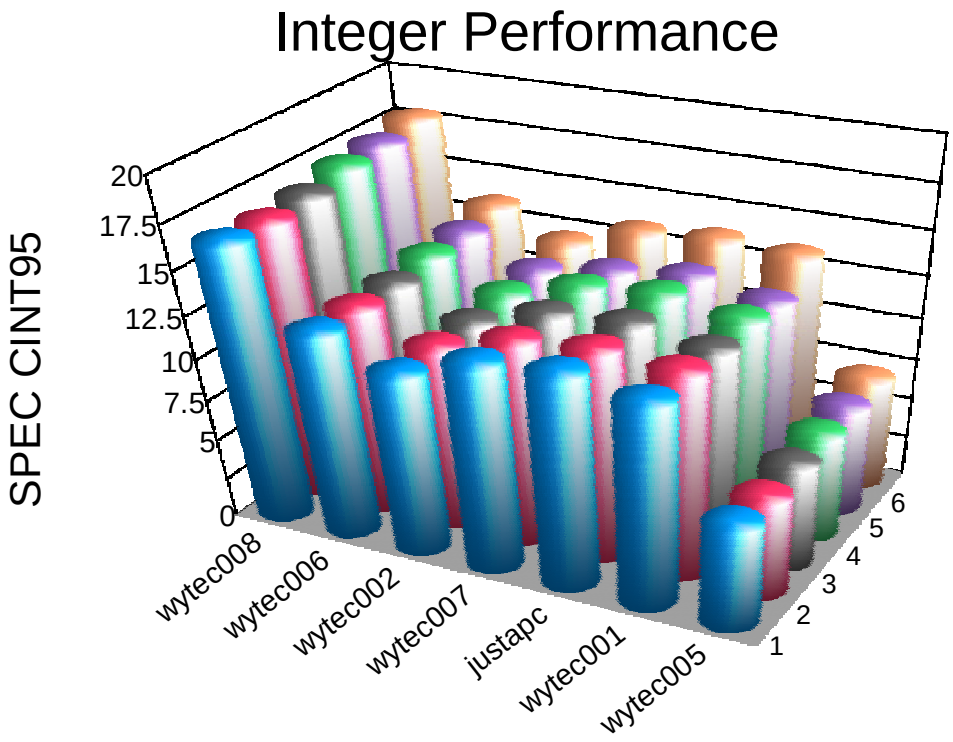
Machine	***CPU***	***L1 Cache***	***L2 Cache***	***L3 Cache***	***RAM***
wytec001	K6−III 400	32kBI+32kBD	256kB on−die	512kB on−board[1]	256MB pc−66 SDRAM
wytec007	K6−III 400	32kBI+32kBD	256kB on−die	None[1]	256MB pc−66 SDRAM
wytec005	Pentium Pro 200	16kBI+16kBD	256kB on−die	None	128MB EDO at 66 MHz
wytec006	Celeron 433	16kBI+16kBD	128kB on−die	None	64MB pc−66 SDRAM
wytec002	Pentium II 350	16kBI+16kBD	512kB off−die	None	256MB pc−100 SDRAM
wytec008	Pentium III 550	16kBI+16kBD	512kB off−die	None	256MB pc−100 SDRAM
justapc	K6−III 412	32kBI+32kBD	256kB on−die	512kB on−board	96MB EDO at 75 MHz

Machine	***Chipset***	***Board***	***Main HD***	***Disk Controller***	***Kernel***
wytec001	Intel 430TX	ASUS TX−97E	Quantum Atlas 4GB SCSI	Adaptec 2940W	2.2.14
wytec007	Intel 430TX	ASUS TX−97E	Quantum Atlas 4GB Wide SCSI	Adaptec 2940W	2.2.14
wytec005	Intel 440FX	ASUS P/I−P6NP5	IBM 4GB Wide SCSI	NCR 53C810	2.2.14
wytec006	Intel 440ZX	(Toshiba Notebook)	IBM DBCA 6GB EIDE	PIIX4 (integrated)	2.2.14
wytec002	Intel 440BX	ASUS P2B−DS[2]	IBM Ultrastar ZX 9GB Ultra SCSI	Adaptec 3940 (on board)	2.2.14
wytec008	Intel 440BX	ASUS P2B−DS[2]	IBM Ultrastar ZX 9GB Ultra SCSI	Adaptec 3940 (on board)	2.2.14
justapc	Intel 430HX	ASUS P/I−P55T2P4	Quantum FB TM 3.2GB EIDE	PIIX3 (integrated)	2.3.49

1 The TX Chipset is just able to use the 3rd level cache for the lowest 64MB area. More RAM than 64MB causes therefore always 3rd level cache−misses above 64MB. It might be faster without the 3rd level cache−controller because of the decreased latency time.
2 For measurements, only a uniprocessor kernel was used.

3 Results

Some interesting results are seen just by a quick look at the results as found here. We will just use the median of the run times, and won't consider the variation in the 3 replications of the individual runs, because if you look at the results (Appendix), you'll see that the variation is really small (with one small exception, the 126.compress benchmark does produce somewhat different results). One interesting thing is that it seems like the greatest performance gain can be achieved with a P6 core system in floating point performance, but the greatest impact here has the move from the old gcc to the new gcc, not the move from the gcc to the pgcc. Like expected, all K6 based systems are not only almost equally fast, but have the same behaviour in respect to the compiler option chosen. We will therefore skip the wytec001 and justapc to keep the tables somewhat smaller. The Celeron, Pentium II and Pentium III have the same behaviour too, which isn't a surprise either, so we will skip the wytec006 and wytec002 for the same reasons. Because it's still too much data, we will skip the measurement 3, which doesn't seem to differ that much from measurement 2.

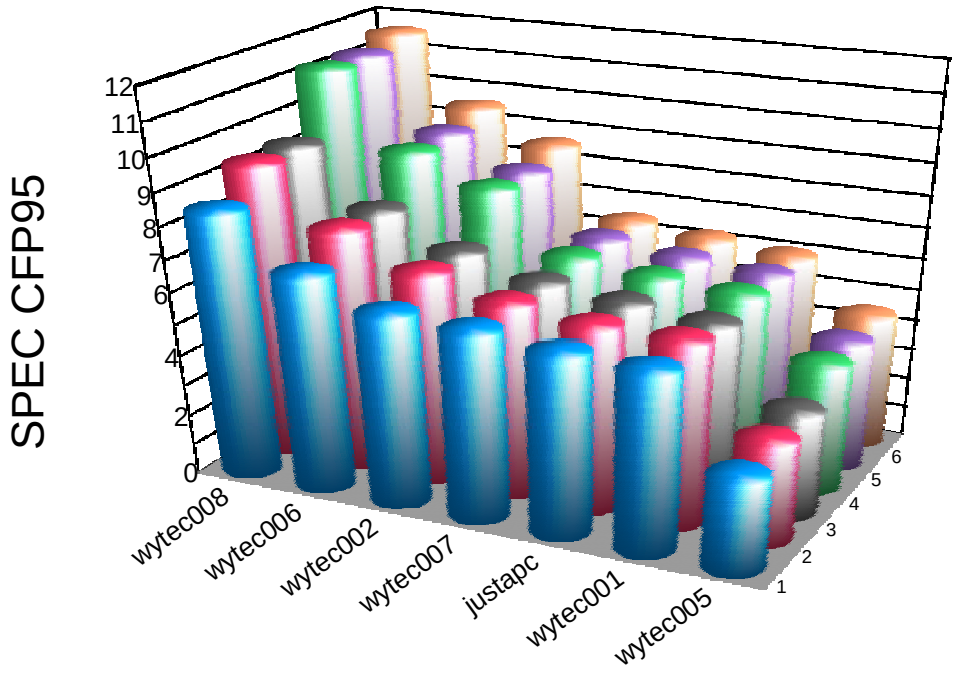


Measure	***Compiler/Options***
1	Gcc 2.7.2.3 −O1
2	Gcc 2.7.2.3 −O2
3	Gcc 2.7.2.3 −O2 −m486
4	Gcc 2.95.2 −O2
5	Gcc 2.95.2 −O6 −march=i486 or −march=pentiumpro
6	Pgcc 2.95.3 −O6 −march=k6 or −march=pentiumpro

	Measurements CINT95 Ratios					
	1	2	3	4	5	6
wytec008	16.5	16.4	16.6	17.1	17.3	17.7
wytec006	12.4	12.4	12.4	12.8	12.9	13.3
wytec002	10.9	11	11	11.3	11.4	11.8
wytec007	12.4	12.2	12.3	12.5	12.1	13.1
justapc	12.8	12.6	12.6	12.9	12.6	13.4
wytec001	12.2	12.3	12	12.3	11.9	13.2
wytec005	6.25	6.03	6.29	6.45	6.5	6.71

	Measurements CFP95 Ratios					
	1	2	3	4	5	6
wytec008	8.44	9.3	9.34	11.2	11.2	11.5
wytec006	6.91	7.66	7.68	8.96	9.07	9.42
wytec002	6.1	6.72	6.73	8.15	8.14	8.46
wytec007	6	6.15	6.14	6.33	6.32	6.31
justapc	5.77	5.88	5.89	6.08	6.07	6.07
wytec001	5.7	5.86	5.66	5.92	5.92	5.86
wytec005	2.96	3.26	3.36	4.14	4.15	4.3

Floating Point Performance



Measurement	***Compiler/Options***
1	Gcc 2.7.2.3 −O1
2	Gcc 2.7.2.3 −O2
3	Gcc 2.7.2.3 −O2 −m486
4	Gcc 2.95.2 −O2
5	Gcc 2.95.2 −O6 −march=i486 or −march=pentiumpro
6	Pgcc 2.95.3 −O6 −march=k6 or −march=pentiumpro

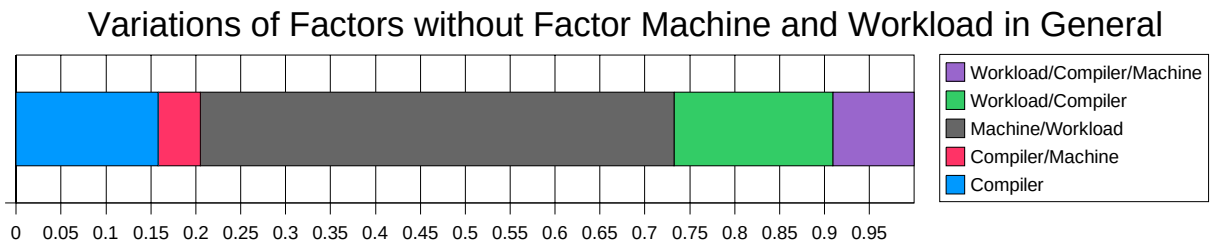
Complete Results can be found in the appendix, page 12 for all CINT run times, page 13 for all CFP run times, and pages 14/15 for the medians of these values.

4 Analysis

We chose a full factorial design because this made it possible to extract the effects of all the different combinations of the factors (e.g. this would allow us to determine which compiler generated the fastest code on a given workload and a given machine). However, we quickly found out that it wouldn't make much sense to answer the main question of our study (how much faster the pgcc in general is) in a simple manner. The performance raises (or drops) are just too much dependent on the cpu type and the workload, but nevertheless we made some general analysis.
We will present here some results from the analysis of variance. For this analysis we took the spec ratios of the individual workloads and used a natural log to transform these values (page 16). We then calculated the main effects and all interactions as described in [3], chapter 23.

Main Effect – Compiler (in all Analyses)						
Analysis	***1***	***2***	***4***	***5***	***6***	***Table***
General	−0.09	−0.06	0.04	0.04	0.07	Page 17
K6−III	−0.02	−0.02	0.01	0	0.03	Page 18
Pentium III	−0.12	−0.07	0.05	0.06	0.08	Page 18
Pentium Pro	−0.14	−0.1	0.06	0.07	0.1	Page 18

Although the absolute numbers are rather small if compared to the other main factors in the analyses (which is logical, because it was never expected that the effect of the compiler was greater than the effect of the machine or the workload), this data proves that the pgcc in general indeed generates faster code. Even if we don't take the factors machine and workload into account, the variation caused by the compiler alone (compared to the variance caused by the interactions) is small. The interaction between machine/workload is much more significant. This is most likely due to the fact that the K6 machine has rather small execution times on the integer workloads and large execution times in most of the floating point workloads if compared to the Pentium III or the Pentium Pro.

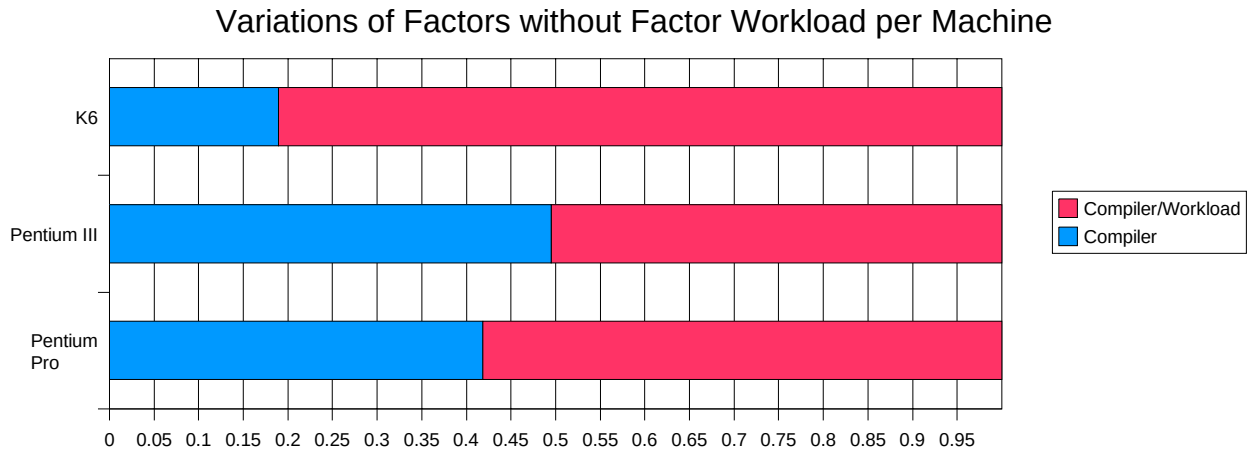


If we look at the variation of the factors if we consider only one machine, some interesting things are noted. On the K6 the compiler alone caused a very low variation, it very much more depends on the compiler/workload combination than on the compiler alone. This indicates that the compiler doesn't optimise all workloads equally well. On the Pentium III and the Pentium Pro, the

3 Jain Raj, John Wiley & Sons, Inc., 1991, ISBN 0−471−50336−3

compiler/workload combination was significant, too, but the compiler alone caused a comparable variance. The absolute value of the variance caused by the compiler and compiler/workload factors is ways higher on these machines too. This was expected too, because a look at the CINT and CFP summaries shows that the new gcc and the pgcc produce significantly faster code in the floating point area on P6 core based systems, whereas all values on K6 systems were relatively constant. Of course, much more analysis could be performed with the data measured, for instance which compiler with which optimisation options should be used for the individual workloads to achieve the highest SPEC peak ratios.

Variations of Factors without Factor Workload per Machine				
	K6	***Pentium III***	***Pentium Pro***	***Table***
Compiler	18.97%	49.49%	41.86%	Page 18
Compiler/Workload	81.03%	50.51%	58.14%	Page 18



5 Conclusions

First of all, we'll try to answer our main question: Is the pgcc overall faster than the standard gcc (either new or old ones), and, if so, how much faster? The answer to the first part of the question is yes. But the second part is difficult to answer. Of course you could use the SPEC ratio to say it is xxx per cent faster, but this would not really answer the question appropriately. It just depends too much on the workload and the CPU type. If you have a FPU intensive workload and are using a P6 core based machine, you are very likely to experience noticeably performance improvements if you're using the pgcc instead of the old gcc. But if you compare the pgcc to the new gcc, the improvements will be much smaller, or not measurable at all. This probably can be explained by the fact that some of pgcc's optimisations techniques have been integrated into the new gcc. What we should mention here is that all SPEC CFP95 measurements we made used the fortran−to−c package (instead of a native fortran compiler), so this code might not represent "normal" native floating−point intensive C−code. The results obtained therefore might not be applicable to native well−written C−code. For instance, multidimensional fortran arrays are organised the other way round than C−arrays, which typically causes a lot of cache misses. So if you wrote your program native in C, you might have experienced less cache misses, and because we don't really know what the new gcc does differently than the old, we cannot know if the same optimisations are also applicable to native C−code. An interesting aspect was the stability that the pgcc has achieved. While earlier versions of the pgcc didn't turn out to be reliable (for instance, linux kernels compiled with them didn't run too stable), the reliability it has achieved is quite good. There were no problems at all running the SPEC benchmarks (of course, there were problems to make them work with any compiler, but there were no problems related to pgcc). One of the test machines (the overclocked justapc) even ran all benchmarks with the developer linux kernel 2.3.49 compiled with pgcc and optimised with −O6 and −march=k6 (we didn't see a difference in kernel speed, however...). Probably there would have been even some more performance advantages if the whole distribution (at least the libraries) would have been compiled with the pgcc, however it is doubtful that this would have made a huge difference.

6 Appendix

SPEC CPU95 INT all results

	099.go			124.m88ksim			126.gcc			129.compress			130.li			132.ijpeg			134.perl			147.vortex		
1just	242	242	241	150	150	150	116	114	114	204	215	223	158	151	154	238	237	237	123	123	123	220	217	217
2just	238	237	238	151	151	151	118	116	116	214	233	220	156	161	159	246	246	246	128	127	127	221	220	220
3just	238	235	236	151	151	151	118	116	116	218	215	204	161	157	169	246	246	246	128	127	127	222	221	220
4just	261	259	258	153	153	153	113	111	111	188	207	211	155	153	150	212	211	211	125	125	125	218	217	216
5just	259	259	260	157	157	157	116	114	114	213	218	206	148	151	148	251	251	251	124	124	124	217	215	214
6just	253	254	253	148	148	148	112	110	110	213	214	201	137	147	140	200	200	200	120	119	119	210	206	207
1wy1	248	248	255	154	154	154	117	117	117	198	243	239	167	169	160	242	241	241	126	126	126	229	232	230
2wy1	245	245	245	155	155	155	119	118	118	196	215	216	175	164	159	252	252	252	131	131	131	225	224	224
3wy1	247	247	247	155	155	155	119	119	119	246	277	249	167	166	170	252	252	252	131	131	131	237	237	236
4wy1	271	277	272	157	157	157	115	115	114	252	255	244	162	157	157	217	217	217	127	127	127	227	230	228
5wy1	268	275	266	161	161	161	119	118	118	262	253	256	160	157	155	257	257	257	128	128	128	231	231	228
6wy1	261	259	260	152	152	152	113	113	113	194	180	212	146	148	144	206	206	206	123	123	123	216	216	217
1wy7	250	249	251	154	154	154	117	116	116	231	231	229	156	160	159	240	240	240	126	126	126	226	225	228
2wy7	251	250	245	155	155	155	118	118	118	226	226	224	168	168	163	249	249	249	131	131	131	226	226	225
3wy7	248	247	248	155	155	155	118	118	118	226	226	226	164	161	164	249	249	249	131	131	131	226	224	225
4wy7	271	271	270	157	157	157	114	113	113	230	230	228	157	159	156	214	214	214	127	127	127	222	225	225
5wy7	269	268	270	161	161	161	116	116	116	245	246	234	156	156	157	255	254	254	128	128	128	224	222	224
6wy7	259	257	257	152	152	152	112	112	112	219	216	216	148	148	147	202	201	201	123	122	122	214	213	214
1wy2	340	339	339	202	202	202	165	165	165	193	188	187	176	173	173	229	229	229	142	141	142	257	258	256
2wy2	339	339	339	195	195	195	164	164	164	184	184	185	172	172	172	238	238	238	138	138	138	255	254	255
3wy2	339	340	339	195	195	195	164	164	164	183	185	185	177	177	172	238	238	238	138	138	138	254	255	254
4wy2	372	372	372	184	184	184	156	156	156	184	184	183	167	166	167	207	207	207	136	136	136	254	253	253
5wy2	373	372	372	181	181	181	158	158	158	173	179	178	166	164	166	213	212	212	136	136	136	247	246	247
6wy2	367	367	368	180	180	180	164	164	164	160	158	158	156	156	156	193	193	193	133	133	133	241	241	241
1wy8	224	224	224	129	129	129	111	111	111	128	131	136	113	113	114	150	150	150	91.7	91.6	91.6	174	174	174
2wy8	225	224	224	125	125	125	110	110	110	137	145	141	115	115	115	155	155	155	89	89	89	172	174	173
3wy8	224	225	224	125	125	125	110	110	110	127	134	130	112	113	113	155	155	155	89	89	89	172	172	172
4wy8	248	247	247	118	118	118	104	104	104	125	125	137	109	109	109	136	135	136	88	88	88	172	172	172
5wy8	246	246	246	115	116	116	106	106	106	118	128	117	110	107	107	139	139	139	87.3	87.3	87.3	168	168	168
6wy8	242	243	242	115	115	115	111	111	111	111	123	116	103	104	102	127	127	127	86.1	86	86	164	165	164
1wy5	597	597	596	368	368	368	280	279	279	338	327	328	307	308	306	395	393	394	256	255	255	438	433	433
2wy5	593	597	595	355	355	355	279	277	278	327	327	327	301	322	332	448	447	447	273	273	273	476	478	473
3wy5	600	599	599	355	355	355	279	278	278	327	330	326	305	302	305	409	408	408	248	248	248	433	429	429
4wy5	650	651	650	327	327	327	269	269	269	326	325	323	295	294	298	356	355	355	250	249	249	430	429	430
5wy5	651	653	648	339	339	339	269	269	268	299	298	299	291	295	298	365	364	364	249	249	249	416	416	415
6wy5	659	649	651	329	328	328	279	278	278	280	292	289	266	266	277	331	330	330	245	244	244	418	415	415
1wy6	283	281	280	164	163	163	137	133	133	222	222	222	162	163	163	188	188	188	111	111	111	247	247	245
2wy6	283	281	279	159	158	158	135	132	132	220	222	222	162	162	166	196	196	195	109	108	108	242	244	244
3wy6	280	279	281	159	158	158	135	134	134	223	221	221	163	163	165	195	195	195	108	108	108	242	245	242
4wy6	315	309	306	149	149	149	130	127	127	224	221	221	160	157	156	170	170	170	108	108	108	241	241	241
5wy6	309	316	307	147	146	146	131	127	127	204	203	203	155	155	153	175	175	175	106	106	106	236	239	239
6wy6	306	304	305	147	146	146	136	133	133	196	192	192	149	147	147	159	158	158	104	104	104	232	235	235

1st digit is the experiment number as in the other tables, the other letters identify the machine

All values in the table are the execution run times in seconds as reported by SPEC CPU95

SPEC CPU95 CFP all results

	101.tomcatv			102.swim			103.su2cor			104.hydro2d			107.mgrid			110.applu			125.turb3d			141.apsi			145.fpppp			146.wave5		
1just	464	463	463	646	646	647	431	426	429	853	853	853	572	573	573	624	624	624	920	920	918	411	403	407	667	666	666	379	366	367
2just	456	455	455	624	625	624	430	424	426	814	814	815	497	498	498	607	606	606	920	921	921	405	404	406	742	741	741	358	358	355
3just	455	459	456	628	625	625	412	423	419	815	816	816	489	488	489	610	610	610	922	922	923	417	411	406	741	741	741	355	355	355
4just	435	436	434	645	646	645	422	421	422	784	785	785	457	456	457	558	558	559	879	878	879	395	397	400	753	753	754	344	338	336
5just	445	443	442	654	653	653	430	434	424	786	782	786	461	456	456	560	559	560	884	885	884	396	394	401	737	737	738	337	336	335
6just	454	452	452	658	657	658	420	415	419	790	792	791	454	450	452	555	556	556	846	846	846	400	404	398	743	744	744	358	344	342
1wy1	463	463	463	643	643	644	422	434	424	858	858	862	583	582	581	623	622	623	932	932	932	441	436	439	684	685	684	369	369	371
2wy1	446	451	445	614	614	614	407	406	410	811	816	819	502	503	501	616	615	616	938	937	939	419	423	419	764	763	762	360	360	362
3wy1	478	479	479	657	657	658	437	420	419	862	861	863	522	524	524	628	626	626	944	944	944	450	446	448	763	761	762	359	364	362
4wy1	442	442	442	658	658	659	431	438	426	799	795	794	473	473	474	567	567	567	899	899	899	431	426	430	773	774	774	348	347	347
5wy1	441	439	439	646	647	648	459	462	454	785	785	785	466	470	467	559	560	560	888	890	888	448	447	447	757	756	758	345	345	342
6wy1	474	473	473	686	686	686	418	417	417	825	829	830	474	475	474	584	584	584	868	868	868	417	414	416	765	764	763	360	355	356
1wy7	411	412	412	572	572	572	410	410	413	794	795	795	569	570	570	586	586	586	912	910	912	422	419	422	685	684	684	356	360	360
2wy7	399	398	398	548	549	549	406	405	406	753	753	753	487	487	488	569	569	569	914	913	914	420	419	420	761	761	762	349	354	346
3wy7	402	401	401	549	548	548	407	398	407	753	753	754	488	488	487	569	568	569	914	913	912	421	420	421	761	761	761	347	351	350
4wy7	398	399	398	578	578	577	404	405	400	716	716	716	442	443	444	523	523	523	868	869	868	413	415	411	773	773	773	330	339	329
5wy7	394	393	393	582	581	581	410	409	410	721	718	720	446	447	446	525	525	525	872	873	873	413	413	411	757	756	757	330	339	332
6wy7	412	413	413	592	592	591	403	402	403	728	728	729	442	443	443	521	521	521	832	834	832	413	415	414	767	764	763	337	335	335
1wy2	367	367	367	464	464	464	371	370	370	751	751	751	491	491	490	604	604	604	1015	1015	1015	288	288	289	1203	1203	1203	415	414	414
2wy2	347	346	347	435	435	435	354	354	354	753	754	754	414	415	415	526	526	526	859	859	859	257	257	256	1007	1007	1007	395	396	396
3wy2	346	346	346	435	435	435	354	355	356	752	752	752	414	414	414	526	526	526	851	851	852	257	256	257	1007	1007	1007	395	395	395
4wy2	332	332	332	480	481	480	252	250	250	489	491	492	383	383	383	437	437	437	602	602	602	240	242	244	793	793	793	298	299	299
5wy2	346	346	346	480	480	480	248	248	248	489	489	489	382	382	382	437	437	437	598	599	598	240	240	239	793	793	793	298	299	299
6wy2	330	331	331	484	484	485	241	241	242	426	428	427	378	378	379	399	399	399	591	591	591	235	234	232	765	765	765	295	295	295
1wy8	278	278	278	363	362	363	290	285	291	592	592	591	353	353	353	427	427	427	673	673	673	199	200	200	767	766	767	297	297	298
2wy8	266	266	266	333	333	333	279	280	278	588	588	588	300	300	300	375	375	375	564	565	565	175	177	175	666	666	666	281	281	281
3wy8	266	265	265	333	333	333	275	276	276	588	588	588	300	300	299	376	376	376	565	564	565	175	175	176	643	643	643	281	282	281
4wy8	246	246	246	374	374	374	194	199	192	392	393	393	282	281	282	313	313	313	410	410	409	165	166	167	527	527	527	208	208	208
5wy8	256	256	255	375	375	375	197	194	196	393	393	394	282	281	282	313	312	313	401	401	401	163	164	162	527	526	527	208	208	208
6wy8	246	244	244	380	380	380	195	192	194	363	363	364	277	277	277	289	289	289	394	394	394	160	160	163	519	519	519	207	206	207
1wy5	631	632	631	767	767	766	831	822	824	1524	1524	1524	838	840	838	1438	1438	1438	2254	2256	2257	593	594	591	3916	3883	3883	739	736	736
2wy5	537	539	538	715	714	714	790	785	784	1438	1438	1438	688	691	691	1327	1326	1326	1683	1684	1685	582	579	578	3631	3633	3633	771	767	772
3wy5	538	536	535	713	714	714	785	784	786	1437	1439	1436	689	688	690	1326	1326	1325	1678	1678	1679	529	525	525	3316	3316	3316	699	701	698
4wy5	721	718	719	788	788	788	550	553	551	876	878	875	624	620	629	903	901	901	1251	1250	1248	486	492	493	2136	2136	2136	488	492	486
5wy5	726	725	725	790	791	790	552	560	543	876	876	876	625	622	627	892	890	890	1220	1220	1220	487	489	491	2152	2151	2151	484	482	481
6wy5	664	663	662	784	784	782	531	529	538	788	786	784	620	618	619	846	846	846	1177	1177	1177	481	480	483	2099	2099	2098	500	502	502
1wy6	305	307	305	389	388	389	402	401	400	687	686	687	470	470	472	525	523	523	822	826	826	281	280	280	993	993	992	319	321	319
2wy6	275	276	277	366	365	366	389	386	386	689	690	688	404	402	404	462	461	461	687	687	687	246	247	246	827	827	828	304	299	300
3wy6	276	277	275	365	366	367	389	386	385	680	681	680	400	400	400	460	460	459	686	687	689	244	243	243	828	829	831	300	300	300
4wy6	312	311	310	410	408	408	299	297	295	473	474	471	375	374	374	377	377	377	497	497	497	234	232	232	666	640	639	235	240	229
5wy6	297	294	295	408	408	408	299	297	297	478	477	478	378	378	378	368	367	366	499	498	498	233	233	233	613	617	612	239	229	228
6wy6	281	277	277	410	411	410	291	286	285	419	418	418	377	376	377	347	347	347	484	484	484	226	224	225	609	609	610	225	225	223

1st digit is the experiment number as in the other tables, the other letters identify the machine

All values in the table are the execution run times in seconds as reported by SPEC CPU95

SPEC CPU95 INT median results

	099.go	124.m88ksim	126.gcc	129.compress	130.li	132.ijpeg	134.perl	147.vortex
1just	242	150	114	215	154	237	123	217
2just	238	151	116	220	159	246	127	220
3just	236	151	116	215	161	246	127	221
4just	259	153	111	207	153	211	125	217
5just	259	157	114	213	148	251	124	215
6just	253	148	110	213	140	200	119	207
1wy1	248	154	117	239	167	241	126	230
2wy1	245	155	118	215	164	252	131	224
3wy1	247	155	119	249	167	252	131	237
4wy1	272	157	115	252	157	217	127	228
5wy1	268	161	118	256	157	257	128	231
6wy1	260	152	113	194	146	206	123	216
1wy7	250	154	116	231	159	240	126	226
2wy7	250	155	118	226	168	249	131	226
3wy7	248	155	118	226	164	249	131	225
4wy7	271	157	113	230	157	214	127	225
5wy7	269	161	116	245	156	254	128	224
6wy7	257	152	112	216	148	201	122	214
1wy2	339	202	165	188	173	229	142	257
2wy2	339	195	164	184	172	238	138	255
3wy2	339	195	164	185	177	238	138	254
4wy2	372	184	156	184	167	207	136	253
5wy2	372	181	158	178	166	212	136	247
6wy2	367	180	164	158	156	193	133	241
1wy8	224	129	111	131	113	150	91.6	174
2wy8	224	125	110	141	115	155	89	173
3wy8	224	125	110	130	113	155	89	172
4wy8	247	118	104	125	109	136	88	172
5wy8	246	116	106	118	107	139	87.3	168
6wy8	242	115	111	116	103	127	86	164
1wy5	597	368	279	328	307	394	255	433
2wy5	595	355	278	327	322	447	273	476
3wy5	599	355	278	327	305	408	248	429
4wy5	650	327	269	325	295	355	249	430
5wy5	651	339	269	299	295	364	249	416
6wy5	651	328	278	289	266	330	244	415
1wy6	281	163	133	222	163	188	111	247
2wy6	281	158	132	222	162	196	108	244
3wy6	280	158	134	221	163	195	108	242
4wy6	309	149	127	221	157	170	108	241
5wy6	309	146	127	203	155	175	106	239
6wy6	305	146	133	192	147	158	104	235

1st digit is the experiment number as in the other tables, the other letters identify the machin

All values in the table are the execution run times in seconds as reported by SPEC CPU95

SPEC CPU95 CFP median results

	101.tomcatv	102.swim	103.su2cor	104.hydro2d	107.mgrid	110.applu	125.turb3d	141.apsi	145.fpppp	146.wave5
1just	463	646	429	853	573	624	920	407	666	367
2just	455	624	426	814	498	606	921	405	741	358
3just	456	625	419	816	489	610	922	411	741	355
4just	435	645	422	785	457	558	879	397	753	338
5just	443	653	430	786	456	560	884	396	737	336
6just	452	658	419	791	452	556	846	400	744	344
1wy1	463	643	424	858	582	623	932	439	684	369
2wy1	446	614	407	816	502	616	938	419	763	360
3wy1	479	657	420	862	524	626	944	448	762	362
4wy1	442	658	431	795	473	567	899	430	774	347
5wy1	439	647	459	785	467	560	888	447	757	345
6wy1	473	686	417	829	474	584	868	416	764	356
1wy7	412	572	410	795	570	586	912	422	684	360
2wy7	398	549	406	753	487	569	914	420	761	349
3wy7	401	548	407	753	488	569	913	421	761	350
4wy7	398	578	404	716	443	523	868	413	773	330
5wy7	393	581	410	720	446	525	873	413	757	332
6wy7	413	592	403	728	443	521	832	414	764	335
1wy2	367	464	370	751	491	604	1015	288	1203	414
2wy2	347	435	354	754	415	526	859	257	1007	396
3wy2	346	435	355	752	414	526	851	257	1007	395
4wy2	332	480	250	491	383	437	602	242	793	299
5wy2	346	480	248	489	382	437	598	240	793	299
6wy2	331	484	241	427	378	399	591	234	765	295
1wy8	278	363	290	592	353	427	673	200	767	297
2wy8	266	333	279	588	300	375	565	175	666	281
3wy8	265	333	276	588	300	376	565	175	643	281
4wy8	246	374	194	393	282	313	410	166	527	208
5wy8	256	375	196	393	282	313	401	163	527	208
6wy8	244	380	194	363	277	289	394	160	519	207
1wy5	631	767	824	1524	838	1438	2256	593	3883	736
2wy5	538	714	785	1438	691	1326	1684	579	3633	771
3wy5	536	714	785	1437	689	1326	1678	525	3316	699
4wy5	719	788	551	876	624	901	1250	492	2136	488
5wy5	725	790	552	876	625	890	1220	489	2151	482
6wy5	663	784	531	786	619	846	1177	481	2099	502
1wy6	305	389	401	687	470	523	826	280	993	319
2wy6	276	366	386	689	404	461	687	246	827	300
3wy6	276	366	386	680	400	460	687	243	829	300
4wy6	311	408	297	473	374	377	497	232	640	235
5wy6	295	408	297	478	378	367	498	233	613	229
6wy6	277	410	286	418	377	347	484	225	609	225

1st digit is the experiment number as in the other tables, the other letters identify the machine

All values in the table are the execution run times in seconds as reported by SPEC CPU95

Data for Analyses

	Integer Base										Floating Point Base							
	4600	1900	1700	1800	1900	2400	1700	2700	3700	8600	1400	2400	2500	2200	4100	2100	9600	3000
1wy7	250	154	116	231	159	240	126	226	412	572	410	795	570	586	912	422	684	360
2wy7	250	155	118	226	168	249	131	226	398	549	406	753	487	569	914	420	761	349
4wy7	271	157	113	230	157	214	127	225	398	578	404	716	443	523	868	413	773	330
5wy7	269	161	116	245	156	254	128	224	393	581	410	720	446	525	873	413	757	332
6wy7	257	152	112	216	148	201	122	214	413	592	403	728	443	521	832	414	764	335
1wy8	224	129	111	131	113	150	91.6	174	278	363	290	592	353	427	673	200	767	297
2wy8	224	125	110	141	115	155	89	173	266	333	279	588	300	375	565	175	666	281
4wy8	247	118	104	125	109	136	88	172	246	374	194	393	282	313	410	166	527	208
5wy8	246	116	106	118	107	139	87.3	168	256	375	196	393	282	313	401	163	527	208
6wy8	242	115	111	116	103	127	86	164	244	380	194	363	277	289	394	160	519	207
1wy5	597	368	279	328	307	394	255	433	631	767	824	1524	838	1438	2256	593	3883	736
2wy5	595	355	278	327	322	447	273	476	538	714	785	1438	691	1326	1684	579	3633	771
4wy5	650	327	269	325	295	355	249	430	719	788	551	876	624	901	1250	492	2136	488
5wy5	651	339	269	299	295	364	249	416	725	790	552	876	625	890	1220	489	2151	482
6wy5	651	328	278	289	266	330	244	415	663	784	531	786	619	846	1177	481	2099	502

Reference time divided by base time, logarithmized

1wy7	2.91	2.51	2.68	2.05	2.48	2.3	2.6	2.48	2.2	2.71	1.23	1.1	1.48	1.32	1.5	1.6	2.64	2.12
2wy7	2.91	2.51	2.67	2.08	2.43	2.27	2.56	2.48	2.23	2.75	1.24	1.16	1.64	1.35	1.5	1.61	2.53	2.15
4wy7	2.83	2.49	2.71	2.06	2.49	2.42	2.59	2.48	2.23	2.7	1.24	1.21	1.73	1.44	1.55	1.63	2.52	2.21
5wy7	2.84	2.47	2.68	1.99	2.5	2.25	2.59	2.49	2.24	2.69	1.23	1.2	1.72	1.43	1.55	1.63	2.54	2.2
6wy7	2.88	2.53	2.72	2.12	2.55	2.48	2.63	2.54	2.19	2.68	1.25	1.19	1.73	1.44	1.59	1.62	2.53	2.19
1wy8	3.02	2.69	2.73	2.62	2.82	2.77	2.92	2.74	2.59	3.17	1.57	1.4	1.96	1.64	1.81	2.35	2.53	2.31
2wy8	3.02	2.72	2.74	2.55	2.8	2.74	2.95	2.75	2.63	3.25	1.61	1.41	2.12	1.77	1.98	2.48	2.67	2.37
4wy8	2.92	2.78	2.79	2.67	2.86	2.87	2.96	2.75	2.71	3.14	1.98	1.81	2.18	1.95	2.3	2.54	2.9	2.67
5wy8	2.93	2.8	2.77	2.72	2.88	2.85	2.97	2.78	2.67	3.13	1.97	1.81	2.18	1.95	2.32	2.56	2.9	2.67
6wy8	2.94	2.8	2.73	2.74	2.91	2.94	2.98	2.8	2.72	3.12	1.98	1.89	2.2	2.03	2.34	2.57	2.92	2.67
1wy5	2.04	1.64	1.81	1.7	1.82	1.81	1.9	1.83	1.77	2.42	0.53	0.45	1.09	0.43	0.6	1.26	0.91	1.41
2wy5	2.05	1.68	1.81	1.71	1.78	1.68	1.83	1.74	1.93	2.49	0.58	0.51	1.29	0.51	0.89	1.29	0.97	1.36
4wy5	1.96	1.76	1.84	1.71	1.86	1.91	1.92	1.84	1.64	2.39	0.93	1.01	1.39	0.89	1.19	1.45	1.5	1.82
5wy5	1.96	1.72	1.84	1.8	1.86	1.89	1.92	1.87	1.63	2.39	0.93	1.01	1.39	0.9	1.21	1.46	1.5	1.83
6wy5	1.96	1.76	1.81	1.83	1.97	1.98	1.94	1.87	1.72	2.4	0.97	1.12	1.4	0.96	1.25	1.47	1.52	1.79

General Analysis, Full Factorial 3*5*18, Factors Machine, Compiler/Options, Workload

Sums of Squares (SSY)	1270.96		
Arithmetic Mean	2.07		
SS0	1161.86		
SSY−SS0	109.1	100.00%	

Main Effects

Machine	0.06	0.47	−0.53																0
Compiler/Options	−0.09	−0.06	0.04	0.04	0.07														0
Workloads	0.54	0.25	0.35	0.08	0.33	0.27	0.41	0.29	0.13	0.69	−0.79	−0.86	−0.38	−0.74	−0.5	−0.24	0.13	0.04	0

45.28	41.50%	Variation of Machine	
1.14	1.04%	Variation of Compiler	
56.63	51.91%	Variation of Workload	
103.04	94.45%		

First Order Interaction, Machine/Compiler

	Wy7	Wy8	Wy5	
1	0.07	−0.03	−0.04	0
2	0.05	−0.01	−0.04	0
4	−0.03	0.01	0.02	0
5	−0.05	0.02	0.03	0
6	−0.04	0.01	0.03	0
	0	0	0	

7.19

Variations of Factors without Factor Machine and Workload

Compiler	15.78%
Compiler/Machine	4.71%
Machine/Workload	52.77%
Workload/Compiler	17.65%
Workload/Compiler/Machine	9.08%

0.34	0.31%	Variation of Machine/Compiler

First Order Interaction, Machine/Workload

	99	124	126	129	130	132	134	147	101	102	103	104	107	110	125	141	145	146	
Wy7	0.21	0.12	0.22	−0.15	0.03	−0.06	0.06	0.08	−0.04	−0.11	−0.1	−0.1	−0.09	0.01	−0.09	−0.27	0.29	0	0
Wy8	−0.12	−0.04	−0.14	0.03	−0.02	0.02	0	−0.07	−0.01	−0.07	0.07	−0.03	−0.04	0.06	0.11	0.19	0.11	−0.05	0
Wy5	−0.09	−0.09	−0.07	0.12	−0.02	0.04	−0.06	−0.01	0.06	0.18	0.03	0.13	0.14	−0.07	−0.02	0.08	−0.4	0.05	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

3.8	3.48%	Variation of Machine/Workload

First Order Interaction, Compiler/Workload

	99	124	126	129	130	132	134	147	101	102	103	104	107	110	125	141	145	146	
1	0.14	0.05	0.08	0.06	0.07	0.04	0.08	0.08	0.07	0.1	−0.08	−0.14	−0.1	−0.11	−0.18	0	−0.09	−0.08	0
2	0.11	0.04	0.04	0.01	0	−0.05	0.02	0.02	0.12	0.13	−0.08	−0.13	0.04	−0.06	−0.05	0.02	−0.09	−0.1	0
4	−0.08	−0.02	−0.02	−0.05	−0.04	0.01	−0.04	−0.05	−0.06	−0.06	0.06	0.08	0.02	0.05	0.07	−0.01	0.06	0.07	0
5	−0.08	−0.03	−0.03	−0.03	−0.03	−0.06	−0.03	−0.02	−0.07	−0.06	0.05	0.08	0.02	0.05	0.08	0	0.07	0.07	0
6	−0.09	−0.03	−0.07	0	0.01	0.05	−0.04	−0.03	−0.07	−0.1	0.04	0.11	0	0.07	0.08	−0.02	0.05	0.03	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

1.27	1.16%	Variation of Compiler/Workload
5.4	4.95%	Variation of First Order Interactions

0.65	0.60%	Estimated Variation of Second Order Interactions

Second Order Interactions, Compiler/Workload/Machine

1wy7	−0.08	−0.02	−0.06	−0.05	−0.05	−0.06	−0.05	−0.07	−0.07	−0.07	0.09	0.09	−0.06	0.06	0.16	0.01	0.2	0.05	0
2wy7	−0.06	−0.02	−0.05	0.02	−0.05	−0.01	−0.04	−0.02	−0.09	−0.07	0.09	0.13	−0.05	0.03	0.03	−0.01	0.08	0.09	0
4wy7	0.03	0	0.02	0.04	0.03	0.05	0.02	0.03	0.06	0.04	−0.06	−0.06	0.03	−0.02	−0.06	0	−0.11	−0.05	0
5wy7	0.05	0.01	0.03	−0.04	0.04	−0.03	0.03	0.02	0.09	0.06	−0.06	−0.05	0.04	−0.01	−0.07	0.01	−0.08	−0.04	0
6wy7	0.07	0.03	0.07	0.03	0.03	0.05	0.04	0.04	0.01	0.04	−0.06	−0.12	0.04	−0.06	−0.06	−0.01	−0.1	−0.04	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1wy8	0.04	0	0.02	0.02	0.02	0.02	0	0.02	−0.03	0.03	−0.05	0	0.05	0	−0.05	−0.03	−0.05	−0.03	0
2wy8	0.01	−0.01	0.01	−0.06	0.02	0.03	0.04	0.03	−0.08	0.03	−0.06	−0.05	0.02	0.03	−0.05	0.03	0.04	0	0
4wy8	−0.01	−0.01	0	0.01	−0.01	−0.03	−0.01	−0.02	0.05	−0.02	0.04	0.01	−0.02	−0.02	0.03	−0.01	0.01	0.01	0
5wy8	−0.02	0.02	−0.01	0.03	−0.01	0.01	−0.01	−0.02	0.02	−0.02	0.04	0.01	−0.03	−0.03	0.03	−0.01	−0.01	0	0
6wy8	−0.02	0	−0.03	0	−0.03	−0.03	−0.02	−0.01	0.04	−0.02	0.03	0.04	−0.01	0.01	0.02	0.01	0.01	0.02	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1wy5	0.05	0.01	0.04	0.03	0.03	0.05	0.05	0.06	0.1	0.04	−0.04	−0.09	0.02	−0.06	−0.12	0.02	−0.15	−0.02	0
2wy5	0.04	0.03	0.04	0.04	0.02	−0.02	0	−0.01	0.17	0.04	−0.03	−0.08	0.03	−0.07	0.02	−0.02	−0.12	−0.08	0
4wy5	−0.01	0.01	−0.03	−0.05	−0.02	−0.02	−0.01	−0.01	−0.11	−0.03	0.02	0.04	−0.01	0.04	0.03	0.01	0.1	0.04	0
5wy5	−0.03	−0.02	−0.02	0	−0.04	0.02	−0.02	0	−0.11	−0.03	0.02	0.04	−0.02	0.04	0.03	0	0.08	0.05	0
6wy5	−0.05	−0.02	−0.04	−0.02	0	−0.02	−0.03	−0.03	−0.05	−0.02	0.04	0.09	−0.02	0.05	0.03	0	0.09	0.02	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

0.65	0.60%	Variation of Second Order Interactions

K6 Analysis, Full Factorial 5*18, Factors Compiler/Options, Workload

Sums of Squares (SSY)			434.02																
Arithmetic Mean			2.13													0.17			
SS0			408.13											Variations of Factors without Factor Workload					
														Compiler		18.97%			
SSY−SS0			25.89	100.00%										Compiler/Workload		81.03%			
	Main Effects																		
Compiler	−0.02	−0.02	0.01	0	0.03														0
Workload	0.75	0.37	0.56	−0.07	0.36	0.21	0.47	0.36	0.09	0.58	−0.89	−0.96	−0.47	−0.73	−0.59	−0.51	0.42	0.04	0
	0.03		0.12%	Variation of Compiler															
	25.72		99.35%	Variation of Workload															

1wy7	0.06	0.03	0.01	0.01	0.01	−0.02	0.03	0.01	0	0.03	0.01	−0.05	−0.16	−0.05	−0.01	0.01	0.11	−0.03	0
2wy7	0.05	0.02	−0.01	0.03	−0.05	−0.06	−0.02	0	0.03	0.06	0.02	0	−0.01	−0.03	−0.02	0.01	0	−0.01	0
4wy7	−0.06	−0.02	0.01	−0.01	−0.01	0.06	−0.01	−0.02	0	−0.02	−0.01	0.02	0.06	0.03	0	0	−0.05	0.02	0
5wy7	−0.03	−0.03	0	−0.06	0.01	−0.09	−0.01	0	0.03	−0.01	0	0.03	0.07	0.04	0.01	0.01	−0.01	0.03	0
6wy7	−0.02	−0.01	0	0.03	0.03	0.11	0.01	0.01	−0.06	−0.06	−0.02	−0.01	0.04	0.01	0.03	−0.02	−0.05	−0.01	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0.14		0.52%	Variation of Compiler/Workload															

Pentium Pro Analysis, Full Factorial 5*18, Factors Compiler/Options, Workload

Sums of Squares (SSY)			236.22																
Arithmetic Mean			1.55													2.05			
SS0			215.56											Variations of Factors without Factor Workload					
														Compiler		41.86%			
SSY−SS0			20.66	100.00%										Compiler/Workload		58.14%			
	Main Effects																		
Compiler	−0.14	−0.1	0.06	0.07	0.1														0
Workload	0.44	0.16	0.28	0.2	0.31	0.31	0.35	0.28	0.19	0.87	−0.76	−0.73	−0.24	−0.81	−0.52	−0.16	−0.27	0.09	0
	0.86		4.15%	Variation of Compiler															
	18.61		90.08%	Variation of Workload															

1wy7	0.19	0.07	0.12	0.09	0.1	0.09	0.13	0.14	0.17	0.14	−0.12	−0.23	−0.08	−0.18	−0.29	0.01	−0.24	−0.1	0
2wy7	0.15	0.07	0.09	0.06	0.02	−0.07	0.03	0.01	0.29	0.17	−0.11	−0.21	0.08	−0.13	−0.04	0	−0.21	−0.18	0
4wy7	−0.1	−0.02	−0.04	−0.1	−0.06	−0.01	−0.04	−0.06	−0.16	−0.09	0.08	0.12	0.01	0.09	0.1	0	0.16	0.11	0
5wy7	−0.1	−0.06	−0.05	−0.02	−0.06	−0.04	−0.05	−0.03	−0.18	−0.1	0.07	0.12	0.01	0.1	0.12	0	0.15	0.12	0
6wy7	−0.14	−0.06	−0.11	−0.02	0.01	0.03	−0.06	−0.06	−0.12	−0.12	0.08	0.19	−0.02	0.12	0.12	−0.02	0.14	0.05	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1.19		5.77%	Variation of Compiler/Workload															

Pentium III Analysis, Full Factorial 5*18, Factors Compiler/Options, Workload

Sums of Squares (SSY)			600.73																
Arithmetic Mean			2.55													1.18			
SS0			583.45											Variations of Factors without Factor Workload					
														Compiler		49.49%			
SSY−SS0			17.28	100.00%										Compiler/Workload		50.51%			
	Main Effects																		
Compiler	−0.12	−0.07	0.05	0.06	0.08														0
Workload	0.42	0.21	0.21	0.11	0.31	0.29	0.41	0.22	0.12	0.61	−0.72	−0.88	−0.42	−0.68	−0.39	−0.05	0.24	−0.01	0
	0.58		3.38%	Variation of Compiler															
	16.1		93.17%	Variation of Workload															

1wy7	0.18	0.05	0.1	0.08	0.09	0.06	0.09	0.1	0.05	0.13	−0.13	−0.14	−0.05	−0.11	−0.22	−0.03	−0.13	−0.1	0
2wy7	0.12	0.03	0.06	−0.04	0.02	−0.02	0.06	0.05	0.04	0.16	−0.14	−0.19	0.06	−0.03	−0.1	0.05	−0.05	−0.1	0
4wy7	−0.1	−0.03	−0.01	−0.05	−0.05	−0.02	−0.05	−0.06	−0.01	−0.08	0.1	0.09	0	0.03	0.1	−0.02	0.07	0.08	0
5wy7	−0.1	−0.02	−0.04	0.01	−0.04	−0.04	−0.05	−0.04	−0.05	−0.09	0.09	0.09	0	0.03	0.12	0	0.06	0.07	0
6wy7	−0.11	−0.04	−0.11	0	−0.02	0.02	−0.05	−0.04	−0.03	−0.12	0.07	0.14	−0.01	0.08	0.11	−0.01	0.05	0.05	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0.6		3.45%	Variation of Compiler/Workload															