STANDARD PERFORMANCE EVALUATION CORPORATION

spec newsletter

Volume 9, Issue 2 July 1997

On the Use of SPEC CPU Benchmarks in Computer Architecture Research

Reinhold Weicker Siemens Nixdorf Informationssysteme AG, OEC HES PM4
33094 Paderborn / Germany
weicker.pad@sni.de

Preface

The following article has been written originally for ACM Computer Architecture News, *it has appeared there in the March 1997 issue (vol. 25, no. 1, pp. 19–22). Since it deals with the SPEC CPU benchmarks, we reprint the article here, for the benefit of the SPEC Newsletter readers.*

Abstract

Benchmarks, in particular the SPEC CPU benchmarks, are frequently used in academic computer research. With ASPLOS-7 as an example, observations about such usage are reported, and suggestions are made for a meaningful use of benchmarks in computer architecture research. Forward-looking computer architecture research may need more than one benchmark collection. (Editorial Note: ASPLOS is the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) organized by the ACM.)

1. Introduction

"So do you mean that we should not use the SPEC benchmarks?"—This was a question I was asked at the October 96 ASPLOS-7 conference, after I had made a critical statement about the selection of programs that were used as benchmarks in one of the papers presented there. In no way, I meant that, but the misunderstanding showed me that it might be worthwhile to put together some general remarks regarding benchmarks and computer architecture research.

It is well known in industry, and certainly known in the SPEC Open Systems Group that the SPEC CPU benchmarks are heavily used in the development labs of all manufacturers: "Should we do the cache design this way or that way? Let´s do a simulation with the SPEC CPU benchmarks!" What is sometimes not so obvious is the impact the benchmarks have on academic research also. However, while benchmarks are ubiquitous in computer architecture research, this fact is seldom a topic of explicit deliberations in the papers themselves.

2. Observations

"A quantitative approach" is the subtitle of one of the most popular and influential textbooks in computer architecture [8]. Following this motto, many research papers at conferences like ASPLOS have the following structure:

- Some clever new idea is presented that could increase the performance of computer systems.
- A possible implementation of this idea (hardware, software, or a combination of both) is discussed.
- Since a complete implementation is often not yet possible, simulation results, often on the basis of trace-driven simulation, are presented.
- A popular collection of programs (benchmarks) serves as the basis for the quantitative computations.
- A sentence like "Our idea has resulted in a performance improvement between xx and yy %, based on the SPEC CPU benchmarks" is often the conclusion, and such a statement is considered evidence that the new idea is worth pursuing.

Since I have been active in benchmarking for more than ten years, and in the SPEC Open Systems Group for six years, I have developed a habit of scanning each research paper that I read for the benchmarks it uses. At ASPLOS-7, about one third of the papers (areas: memory hierarchy design, branch prediction, etc.) used the SPEC CPU benchmarks [2] (mostly CPU95, some still CPU92), another third (area: multiprocessor designs) used the

Continued on page 3

Use of SPEC CPU Benchmarks

continued from page 1

SPLASH benchmarks [13], the remaining third used other program collections as benchmarks. Conte and Hu [4] have compiled more detailed statistics about the use of benchmarks in computer architecture research some years ago. Since then, in particular the SPEC CPU benchmarks have become more and more popular. The indirect influence that we in the SPEC group have as "benchmark producers" is amazing, and is frightening at the same time. We know that the benchmarks we have are not God-given. We do our best to find and select good benchmarks, but the process is limited by some unavoidable circumstances: Some of the most interesting programs (e.g. a real-life operating system, or only a substantial part of it) are not freely available in source code form; some interesting progams turn out to be not portable enough, or the results could not be validated. For example, "ghostscript" and the C version of "spice" were two programs that we tried and that many of us would have wanted for the CPU95 suite, but it just wasn´t possible. Unavoidably, there are also subjective judgements about the merits of a specific benchmark candidate, and therefore differences in the final votes within SPEC.

These facts do not mean that benchmarks are not useful. With proper caveats, they are useful for both comparative evaluation of existing computers and for academic research: It makes sense to test the practical value of ideas using some well-known programs as benchmarks. What I´d expect more from research papers, however, is a critical reflection about the benchmarks and the influence that the benchmark selection has on a particular research topic. An example are papers on branch prediction methods: The CINT92 benchmark 023.eqntott has sometimes been used, among other programs, as a test case for branch prediction. A close look at the benchmark, however, shows that it spends about 80 % of its time in a small subroutine with an if-then-else statement in the inner loop that is written in a somewhat unusual style (This fact, and the resulting opportunities for special-case optimizations via pattern matching were the reason that SPEC dropped this benchmark for the CINT95 suite [12].) Therefore, a claim for a particular method of branch prediction that uses eqntott results as part of the argument seems questionable to me.

3. Wishlist of a Benchmarking Practitioner

Again, I repeat that computer architects should continue to use the SPEC CPU benchmarks. Usage of a well known set of programs can be very useful, and as a SPEC practicioner, of course, I appreciate the opportunity to learn more about these benchmarks. However, some points should be observed:

- The CPU95 suite should be preferred over the older and now outdated CPU92 suite [2].
- If possible, all programs in the suite should be used. There may be legitimate reasons for using only a subset, but if no reason is given for the selection, the reader may become suspicious: Why did the author not consider the other programs? Does his or her method not work so well for them? If subsetting is done for reasons of limited time or computer availability for simulation (a legitimate reason), some statement about the selection process, and why it was not arbitrary, is useful. Of course, if a certain new feature speeds up programs of a certain type only, this is legitimate also and worth reporting.
- The compilation mode used is often important, and should be reported in every case. The effect of a certain new hardware feature may be highly dependent on whether it is applied to optimized or to unoptimized programs.
- Some properties observed and used in research may be due to the fact that in the SPEC form, the programs do no longer have their original form, but have been modified for their purpose as benchmarks. For example, it is an explicit goal of the CPU benchmark suite that the influence of the I/O and operating system is limited. It is therefore no surprise that the cache behaviour is different from that of programs which have more interaction with the operating system [5].
- It is probably a good idea to use another set of (non-SPEC) programs as additional test cases. For example, I liked the approach in [1] and [10] where the authors showed that, with respect to certain criteria, the SPEC set of benchmarks behaved differently than the non-SPEC set. Such observations can be useful as hints to SPEC when it comes to the next round of benchmark selection (CPU98).
- Finally, I'd like to encourage papers that not only use the SPEC benchmarks (or other benchmarks) as test cases for some architectural idea but explicitely make them the topic of an investigation in its own right, like [3,6,7]. The value of such empirical studies should not be underestimated; they can in turn help others to use the benchmarks in a better way.

So far, there are very few papers where independent academic researchers write about benchmarking issues. This may result from a feeling that the area of benchmarking is somehow below their standard, that it is an area of dirty tricks and misleading claims. On the other hand, benchmarks continue to be used in research. So it would probably benefit everyone if they get the attention they deserve.

4. Future Benchmarks

By their very nature, benchmarks must be portable and freely distributable. These requirements, in connection with the limited resources available to benchmarking organizations, tend to favour the collection of older, sometimes even "dusty deck" programs. While these programs may be representative for the majority of current computer installations, there is an inherent danger connected with the scenario. Computers of tomorrow are optimized on the basis of the benchmarks of today, i.e. the programs of yesterday. It may be worthwhile to explicitly try to put together an alternative collection of "advanced programming style" benchmarks. Such benchmarks could be written in different languages (e.g. C++ instead of C), and could incorporate features that are not yet so frequent in "mainstream programs" (e.g. object-oriented programming style). For computer architecture research, they would certainly form a valuable "contrast set" to established benchmarks.

Recognizing the importantance of its benchmarks, and continuing in the effort to improve their quality, SPEC has initiated a search program for new CPU benchmarks [11]. Commercial and academic users can only wish that such efforts are successful, in the interest of both academic computer science and commercial computer development.

References

1. I-Cheng K. Cheng, John T. Coffey, and Trevor N. Mudge: Analysis of Branch Prediction via Data Compression. 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, = *ACM SIGPLAN Notices* 31,9 (Sept. 1996), 128-137
2. Kaivalya Dixit and Jeff Reilly: SPEC95 Questions and Answers. *SPEC Newsletter* 7,3 (Sept. 1995), 7–10. Also in http://www.specbench.org/osg/cpu95/news/cpu95qa.html
3. Jeffrey D. Gee, Mark D. Hill, Dionisios N. Pnevmatikos, and Alan J. Smith: Cache Performance of the SPEC92 Benchmark Suite. *IEEE Micro* (Aug. 1993), 17–27
4. Thomas M. Conte and Wen-mei W. Hu: A Brief Survey of Benchmark Usage in the Architecture Community. *Computer Architecture News* 19,4 (June 1991), 37–44
5. John H. Fraser and David R. Kaeli: Operating System Impact on Cache Performance. Manuscript, Northeastern University, Boston, 25 pages, 1996
6. Andrew S. Huang and John Paul Shen: The Intrinsic Bandwidth Requirements of Ordinary Programs. 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, = *ACM SIGPLAN Notices* 31,9 (Sept. 1996), 105–115
7. Nikki Mirghafori, Margret Jacoby, and David Patterson: Truth in SPEC Benchmarks. *Computer Architecture News* 23,5 (Dec. 1995), 34–42
8. David A.. Patterson and John L. Hennessy: Computer Architecture. *A Quantitative Approach.* Morgan Kaufmann, San Francisco 1996 (2nd edition)
9. Jeff Reilly: SPEC Describes SPEC95 Products and Benchmarks. *SPEC Newsletter* 7,3 (Sept. 1995), 4–6. Also in http://www.specbench.org/osg/cpu95/news/cpu95descr.html
10. Stuart Sechrest, Chih-Chieh Lee, and Trevor Mudge: Correlation and Aliasing in Dynamic Branch Prediction. 23rd Ann. Symp. on Computer Architecture, = *Computer Architecture News* 24,2 (May 1996), 22–32
11. [SPEC] SPECCPU98 Development (Subpage in SPEC's World Wide Web page). http://www.specbench.org/osg/cpu98, Dec. 1996
12. Reinhold Weicker: An Example of Benchmark Obsolescence: 023.eqntott. *SPEC Newsletter* 7,4 (Dec. 1995), 5–6. Also in http://www.specbench.org/osg/cpu95/news /eqntott.html
13. Steven Cameron Woo et al.: The SPLASH-2 Programs: Characterization and Methodological Considerations. 22nd Ann. Symp. on Computer Architecture, = *Computer Architecture News* 23,2 (May 1995), 24–36

While the author draws most of his experience from his work in the SPEC group, representing Siemens Nixdorf in the SPEC Open Systems Steering Committee, the views expressed in this article are strictly personal and do not claim to represent either SPEC's or Siemens Nixdorf's opinions.

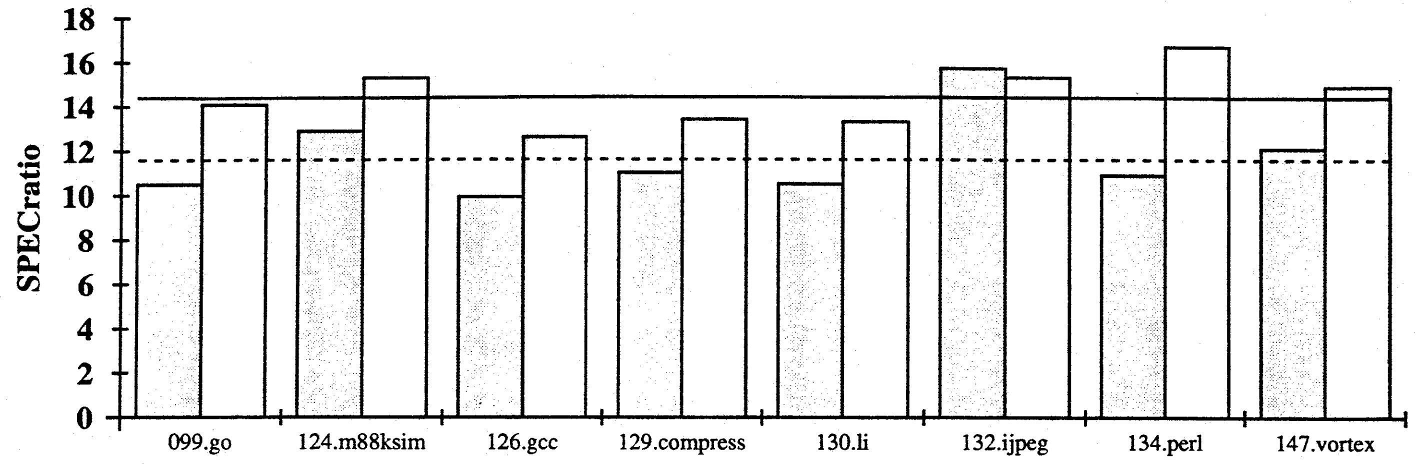
spec

SPEC CINT95 Results

Digital Equipment Corp.
AlphaServer 1000A 5/500

SPECint95 =	14.4
SPECint_base95 =	11.6

SPEC license # 2	Tested By: Digital NH	Test Date: Mar-97	Hardware Avail: Apr-97	Software Avail: Dec-96



Hardware/Software Configuration for:
AlphaServer 1000A 5/500

Hardware

Model Name:	AlphaServer 1000A 5/500
CPU:	500 MHz 21164
FPU:	Integrated
Number of CPU(s):	1
Primary Cache:	8KB(I)+8KB(D) on chip
Secondary Cache:	96KB
Other Cache:	8MB
Memory:	512MB
Disk Subsystem:	2GB
Other Hardware:	Ethernet

Software

Operating System:	Digital UNIX V4.0B
Compiler:	DEC C V5.4-045 cc.alt/protect_headers.sh
File System:	UFS
System State:	Multi User

Benchmark # and Name	Reference Time	Base Run Time	Base SPEC Ratio	Run Time	SPEC Ratio
099.go	4600	437	10.5	327	14.1
124.m88ksim	1900	148	12.9	124	15.3
126.gcc	1700	172	9.91	135	12.6
129.compress	1800	163	11.0	134	13.4
130.li	1900	181	10.5	142	13.3
132.ijpeg	2400	153	15.7	156	15.3
134.perl	1900	174	10.9	114	16.7
147.vortex	2700	223	12.1	181	14.9
SPECint_base95 (G. Mean)			11.6		
		SPECint95 (G. Mean)			14.4

Notes/Tuning Information

```
Compiler: cc.alt -std1    Base optimizations: -O4 -arch ev56 -non_shared -om
Portability flags: m88ksim: -DLEHOST    perl: -DI_TIME    vortex: -D__RISC_64__

Peak flags: all use -ifo -non_shared.  Other flags are:
go: -g3 -O3 -inline speed EXTRA_LDFLAGS -om     m88ksim: -g3 -O4 -speculate all -inline speed with feedback
gcc: -g1 -O4 -inline speed -arch ev56 -xtaso_short -taso with feedback    compress: -g3 -O4 -tune ev5 -32data
-inline speed -arch ev56 -assume whole_program -om -lsys5    li: -g1 -O4 -inline speed -xtaso_short -speculate
all -lsys5 -taso with feedback    ijpeg: -g3 -O4 -fast -unsigned -inline speed -arch ev56 -speculate all -lsys5
with feedback    perl: -g3 -O4 -inline speed -arch ev56 -lsys5 with feedback pass 2 only    vortex: -g3 -O4
-fast -inline speed -speculate all with feedback

feedback: PASS1=-gen_feedback
fdo_pre1=mv %binary% %exename%.orig; pixie -pids %exename%.orig -o %exename%
fdo_post1=prof -pixie -feedback ../../src/%exename%.fb %exename%.orig %exename%.orig.Addrs %exename%.orig.Counts.*
PASS2=-feedback $(EXENAME).fb -r
fdo_pre2=mv %exename% %exename%.rr; ld -o %exename%fb %exename%.rr -lexc; pixie -pids %exename%fb -o %exename%
fdo_run2=%command%
fdo_post2=ld -om %exename%.rr -o %exename%.rrom -lexc; prof -pixie -merge %exename%fb.Counts %exename%fb %exename%fb.Addrs
%exename%fb.Counts.*; /usr/lib/cmplrs/cc.alt/om -om_ireorg_feedback %exename%fb -o %binary% %exename%.rrom
gcc and li add -taso to feedback commands om and ld
```

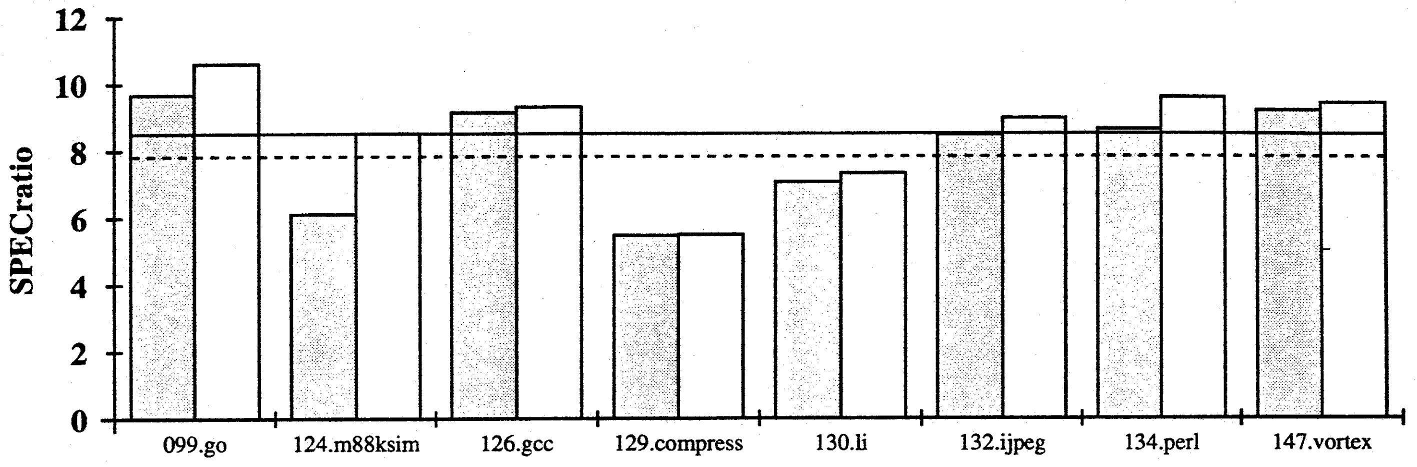
SPEC CINT95 Results

HAL Computer Systems
Fujitsu HALstation 300

SPECint95 = 8.51

SPECint_base95 = 7.83

SPEC license #	27	Tested By:	HAL Computer Systems	Test Date:	Apr-97	Hardware Avail:	Nov-96	Software Avail:	Oct-97



Hardware/Software Configuration for:
Fujitsu HALstation 300

Hardware

Model Name:	385
CPU:	161 MHz SPARC64
FPU:	Integrated
Number of CPU(s):	1
Primary Cache:	128KBI,128KBD on chip
Secondary Cache:	None
Other Cache:	None
Memory:	128MB
Disk Subsystem:	1 x 2GB 1 x 2GB
Other Hardware:	Ethernet

Software

Operating System:	SPARC64/OS 2.4
Compiler:	Fujitsu C V3.01
File System:	UFS
System State:	Multiple User
Kernel Extensions:	none

Benchmark # and Name	Reference Time	Base Run Time	Base SPEC Ratio	Run Time	SPEC Ratio
099.go	4600	476	9.67	435	10.6
124.m88ksim	1900	311	6.11	223	8.52
126.gcc	1700	186	9.14	183	9.31
129.compress	1800	328	5.48	327	5.50
130.li	1900	269	7.07	259	7.33
132.ijpeg	2400	284	8.47	268	8.97
134.perl	1900	220	8.64	198	9.60
147.vortex	2700	293	9.21	286	9.44
SPECint_base95 (G. Mean)			7.83		
		SPECint95 (G. Mean)			8.51

Notes/Tuning Information

```
Baseline flags were: -Kfast,GREG,V8PLUS -x-
Nonbaseline flags were: ALL -dn -Kfast,GREG,V8PLUS
099: -O4 -Kpopt -Knounroll
124: -O4 -Kpopt -x15 -Kprefetch
126: -O4 -Kpopt -x10 -Knounroll
129: -O4 -Kpopt -x100
130: -O4 -Kpopt -x100 -Knounroll
132: -O4 -x- -Kprefetch
134: -Kpopt -x- -Kprefetch
147: -O4 -Kpopt -x- -Kprefetch
Portability: 124, 132: -DSYSV
Portability: 126: -Dalloca=__builtin_alloca
```

For More Information Contact:
SPEC
10754 Ambassador Drive, Suite 201
Manassas, VA 20109
(703) 331-0180
info@specbench.org
http://www.specbench.org/

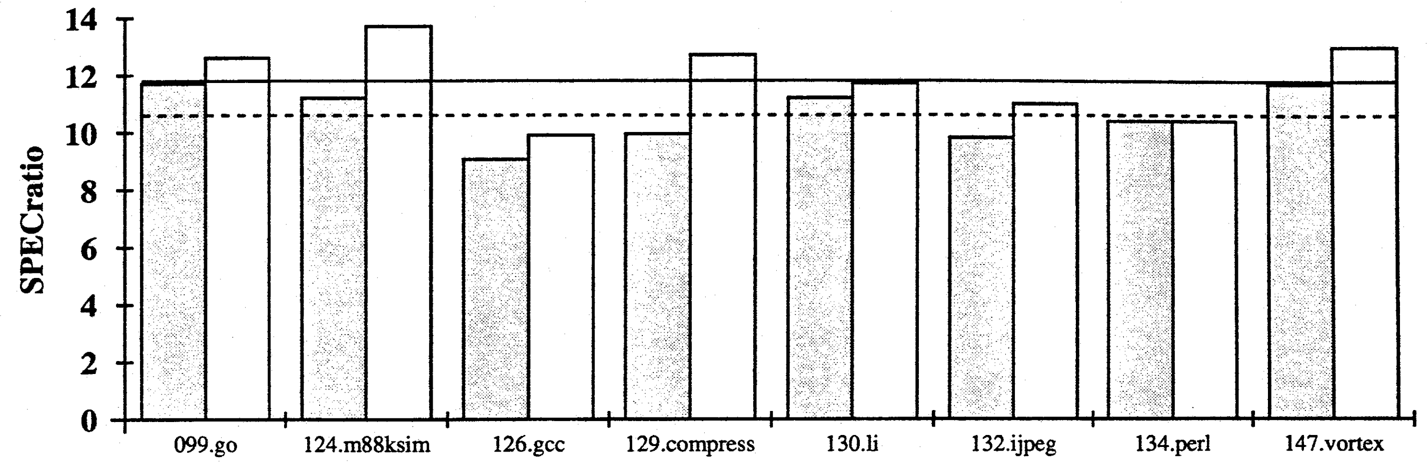
SPEC CINT95 Results

Hewlett-Packard
HP 9000 Model T600 1-CPU

SPECint95 = 11.8

SPECint_base95 = 10.6

SPEC license #	3	Tested By:	HP Cupertino, CA	Test Date:	May-97	Hardware Avail:	Jun-97	Software Avail:	Aug-97



Hardware/Software Configuration for:
HP 9000 Model T600 1-CPU

Hardware

Model Name:	HP 9000 Model T600 1-CPU
CPU:	180MHz PA-RISC 8000
FPU:	Integrated
Number of CPU(s):	1
Primary Cache:	1MBI+1MBD off-chip
Secondary Cache:	8MB(I+D) off-chip
Other Cache:	None
Memory:	2GB
Disk Subsystem:	1 FWD SCSI-2 1.0 GB
Other Hardware:	None

Software

Operating System:	HP-UX B.10.30
Compiler:	A.10.33.03 HP C Compiler
File System:	HP-UX HFS
System State:	Multi-user

Benchmark # and Name	Reference Time	Base Run Time	Base SPEC Ratio	Run Time	SPEC Ratio
099.go	4600	392	11.7	365	12.6
124.m88ksim	1900	170	11.2	138	13.7
126.gcc	1700	187	9.07	172	9.91
129.compress	1800	181	9.95	142	12.7
130.li	1900	170	11.2	162	11.7
132.ijpeg	2400	245	9.81	219	11.0
134.perl	1900	183	10.4	183	10.4
147.vortex	2700	231	11.7	207	13.0
SPECint_base95 (G. Mean)			10.6		
		SPECint95 (G. Mean)			11.8

Notes/Tuning Information

```
Portability Flags (base & peak): All: -Ae

Base Flags:
All: fastmem.o +Oall +I/+P
Linker Flag:  All: -Wl,-aarchive

Peak Flags:
All (except 134): +ESfic +ESlit +O4 +I/+P
099: +Oentrysched +Olibcalls +nofastaccess +Onolimit +Onoloop_unroll +Optrs_strongly_typed +Ostaticprediction
124: +Oaggressive +Onoparmsoverlap +Optrs_strongly_typed +Ostaticprediction
126: -DSPEC +Olibcalls +Onolimit +Ostaticprediction
129: +Odataprefetch +Olibcalls +Onofltacc +Onolimit +Optrs_strongly_typed +Owhole_program_mode
130: -lm +ESsfc +Oentrysched +Olibcalls +Onolimit +Onoloop_unroll +Onoptrs_to_globals +Owhole_program_mode
132: +ESsfc +Odataprefetch +Olibcalls +Onolimit +Optrs_strongly_typed +Owhole_program_mode
134: fastmem.o +Oall +I/+P
147: fastmem.o -lm +Oentrysched +Olibcalls +Onolimit +Onoloop_unroll +Optrs_strongly_typed +Ostaticprediction
Linker Flag:  All: -Wl,-aarchive

Note: The +I/+P indicates the use of profile based optimization.
```

For More Information Contact:	SPEC 10754 Ambassador Drive, Suite 201 Manassas, VA 20109	(703) 331-0180 info@specbench.org http://www.specbench.org/

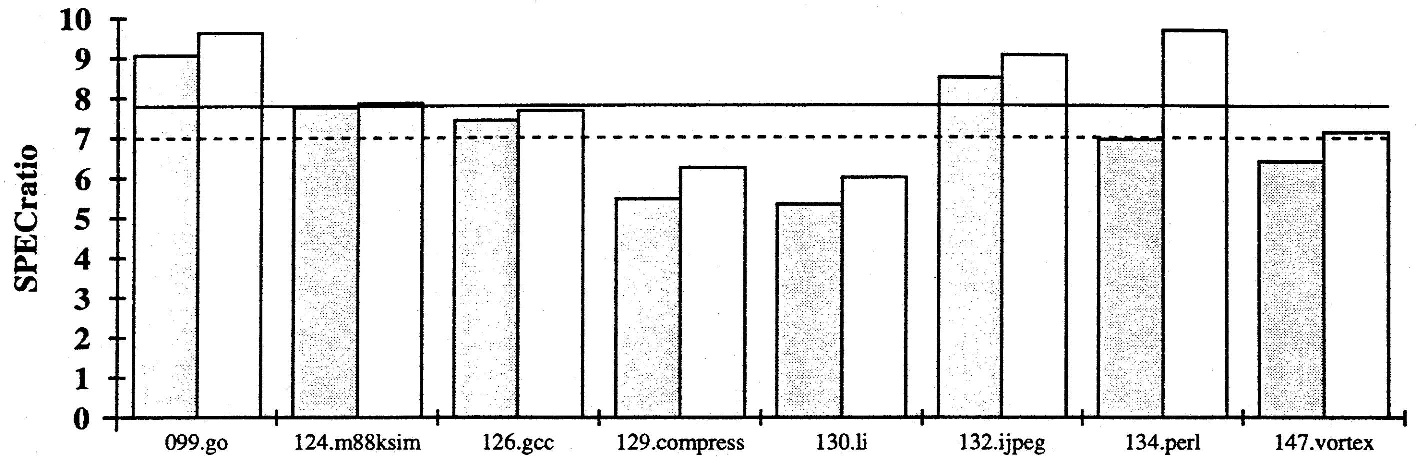
SPEC CINT95 Results

IBM Corporation
RS/6000 43P-140 (200MHz)

SPECint95 = 7.79

SPECint_base95 = 6.99

SPEC license #	Tested By:	Test Date:	Hardware Avail:	Software Avail:
11	IBM, Austin TX	Mar-97	Nov-96	Apr-97



Hardware/Software Configuration for:
RS/6000 43P-140 (200MHz)

Hardware

Model Name:	RS/6000 43P-140
CPU:	200 MHz PowerPC 604e
FPU:	Integrated
Number of CPU(s):	1
Primary Cache:	32KBI+32KBD on chip
Secondary Cache:	1MB(I+D) off chip
Other Cache:	None
Memory:	64MB
Disk Subsystem:	1x2.2GB SCSI
Other Hardware:	None

Software

Operating System:	AIX 4.2.1
Compiler:	IBM CSet++ 3.1.4.5
File System:	AIX/JFS
System State:	Multi-user

Benchmark # and Name	Reference Time	Base Run Time	Base SPEC Ratio	Run Time	SPEC Ratio
099.go	4600	507	9.07	478	9.63
124.m88ksim	1900	245	7.75	242	7.86
126.gcc	1700	229	7.42	222	7.66
129.compress	1800	330	5.45	289	6.23
130.li	1900	357	5.32	317	5.99
132.ijpeg	2400	283	8.48	265	9.05
134.perl	1900	274	6.94	196	9.68
147.vortex	2700	422	6.40	379	7.13
SPECint_base95 (G. Mean)			6.99		
		SPECint95 (G. Mean)			7.79

Notes/Tuning Information

```
Compatibility Flags: gcc -ma; perl -DI_TIME -DI_SYS_TIME
Base: -O3 -qarch=ppc -Q=500 -qpdf1/pdf2
Peak: all used -qpdf1/pdf2
099: -O -qarch=ppc -Q=500 -qansialias -qdatalocal -qproclocal -qunroll=0
-bnso -bI:/lib/syscalls.exp
124: -O3 -qansialias -qarch=ppc -Q=200 -bnso -bI:/lib/syscalls.exp; fdpr -R2
126: -O -qarch=ppc -qdatalocal -bnso -bI:/lib/syscalls.exp; fdpr -R2
129: -O -qarch=ppc -Q=200 -qdatalocal -qassert=allp
130: -O -Q=1000 -qdatalocal -qunroll=2 -Dsetjmp=_setjmp -Dlongjmp=_longjmp
-bnso -bI:/lib/syscalls.exp
132: -O3 -Q=100 -bnso -bI:/lib/syscalls.exp
134: -O3 -qarch=ppc -qdatalocal -qansialias -Dsetjmp=_setjmp -Dlongjmp=_longjmp
-bnso -bI:/lib/syscalls.exp /usr/ccs/lib/bmalloc.o
147: -O3 -qarch=ppc -Q=200 -qdatalocal -bnso -bI:/lib/syscalls.exp; fdpr -R2
```