The Case for Personal Computers as Workstations

Timothy J. Gibson and Ethan L. Miller
Department of Computer Science and Electrical Engineering
University of Maryland - Baltimore County
5401 Wilkens Avenue
Baltimore, MD 21228
E-mail: {tgibso2, elm}@cs.umbc.edu
Telephone: (410)455-3972 FAX (410)455-3969

Abstract

This paper presents the case for using personal computers as workstations. The argument is based on the performance of CISC processors using a 32 bit operating system. The authors have developed a set of simple benchmarks that measure integer, floating point, and memory manipulation. The benchmarks were used to measure the performance of a variety of RISC based workstations and Intel CISC based personal computers under several operating systems. On the personal computers based on Intel x86 and Pentium processors, the benchmarks are run under both the Microsoft Disk Operating System (MS-DOS) with Windows and the Linux Operating System. While the benchmarks reveal nothing surprising about the relative speed within the different processor families (for example, a MIPS 4000 versus a MIPS 4400), they did reveal that the Intel CISC processors perform the benchmarks 1.25 to 7.25 times faster using Linux than with the MS-DOS and Windows combination. This is because Linux is a 32 bit operating system and also accesses memory faster than the 16 bit MS-DOS and Windows combination [Chen]. As a result, the CISC based machines with Linux run the benchmarks as well or better than many of the more expensive RISC machines. In addition to these simple benchmarks, the authors also benchmarked one of the Intel CISC computers under Xwindows, the most common workstation environment. Again, the personal computer performed adequately. Given the price difference between personal computers and workstations, the case for using personal computers as general purpose workstations is very strong.

I - Introduction.

The Intel chip based personal computer world is awash with benchmarks. Open any personal computer magazine and you will find personal computers rated with WinMarks, iComps, Bytemarks or some other benchmark. On the other hand, the RISC (Reduced Instruction Set Computer) and Unix operating system workstation are usually benchmarked with the SPECMarks from the Standard Performance Evaluation Corporation (SPEC). While there are other benchmarks for RISC machines, most people agree the SPECint and SPECfp provide a standard for comparing the integer and floating point capabilities of the different types of RISC computers. Two of the most important characteristics of a networked RISC workstation are the CPU's integer and floating point performance; the SPECMarks measure these two features very well.

The SPECMarks also run on Intel based personal computers and the statistics are available [SPEC-b], but they are not normally used to evaluate personal computers. There are several reasons SPECMarks are not frequently used to judge personal computers. To begin with, the SPECMarks are expensive, take the better part of a day to run on any one machine, and require a minimum of 64 Mbytes of main memory [Yager]. In addition to not being practical for the average person to run, the SPECMarks only test CPU and memory performance. They do not test many of the components personal computer users want measured, such as disk I/O and video speed.

The SPECMarks have other problems beyond being impractical for many machines. The SPECMarks commonly available from the manufacturers and other sources usually do not list important factors that are needed to make valid comparisons. When a benchmark is run there are many characteristics that should be reported, such as the operating system type and version, the size of main memory and cache, and compiler and compiler settings used to create the executable code [Ahmad, Fleming, Muchsel, and Sill]. These and many other factors are usually missing from the manufacturer's technical reports - often only the SPECMarks themselves are available.

Because of these problems with the SPECMarks, we developed a small set of benchmarks to compare different CPU types, memory systems, and operating systems. We wanted the benchmarks to be able to scale well across different CPU types and operating systems. For example, if a benchmark test runs at a certain speed on one computer, a second computer that is identical to the first except for the CPU should run the test faster or slower by a factor depending only on the difference in the CPU. Similarly, if the benchmark is testing memory access or capacity, the size and speed of the memory can change the results. On the other hand, the tests must provide a fair assessment of a computer's overall capabilities and allow comparisons of different computers and operating systems. If there are anomalies in the data the benchmarks produce, these anomalies should be explainable by the different CPU, memory architectures, or operating systems. Once the tests are run, a fair comparison can be made between the different CPU types and different operating systems.

II - Background.

Nearly all personal computers use MS-DOS with Microsoft's Windows as a user interface. MS-DOS has been around in one form or another since the early 1980s, and was written for IBM's original IBM PC. Many of the hardware limitations of the original IBM PC continue to plague current versions of MS-DOS (and the personal computers using it) because consumers demand that new computers still run programs purchased several years ago. The requirement for backward compatibility is a heavy burden to carry. For example, the original IBM PC only had 256 KB of memory (upgradeable to 640 KB). IBM and Microsoft addressed this memory in 64 KB segments. It is now common for personal computers to have 8 - 16 MB of memory. While a personal computer's memory has increased by an order of magnitude, MS-DOS still addresses the memory in 64 KB segments. This means that while newer personal computers have faster CPUs and more memory, they cannot use these faster and larger resources as well as possible because of the MS-DOS operating system. Many people have recognized

This paper appeared in the 22nd Annual International Conference on Computer Measurement and Performance (CMG '96), San Diego, CA, December 1996, pages 644–652.

these flaws and a variety of products that solve (or at least alleviate) these problems are becoming available. However, for the foreseeable future personal computers will have either 16 bit or hybrid 16 and 32 bit operating systems.

Personal computers use either Intel CPU chips or a derivative of the Intel chips. The Intel CPU chips are Complex Instruction Set Computer (CISC) chips. The machine-level instructions can be fairly complex as the name implies. When Intel was designing their CPU chips in the early 1980s, one of the ways Intel held down costs was by limiting the number of registers on the chip itself. While limiting the registers saves money, it also means more loads and stores from memory, and hence, slower program execution. Because Intel must maintain backward compatibility with older software and CPU chips, new CPU chips must be organized similar to the earlier CPU chips. This similar organization can slow down the newer CPU chips. While Intel has made some substantial speed increases with their newer chips, the problem of backward compatibility continues to nag the designers and hamper performance.

Reduced Instruction Set Computers (RISC) differ from CISC primarily in how the CPU works. RISC only take short, simple instructions and do each very quickly. While a similar operation may take many more instructions on a RISC CPU than a CISC CPU, the general idea is that the RISC CPU will do the same operation faster than the CISC CPU - despite the larger number of instructions [Patterson80]. One of the reasons the RISC CPUs can execute many instructions quickly is their compilers are well engineered and generate very fast, optimized source code. Because the RISC CPUs execute comparatively simple instructions, the hardware on the chip is simpler and faster. As a result, RISC CPUs usually have features normally not found on CISC CPUs. For example, RISC CPUs have many more registers to store data and reduce the amount of memory operations. Additionally, new CPU features like out-of-order instruction execution or parallel execution are more difficult to implement on CISC than RISC machines. All of these factors, compilers, quick and simple hardware, and advanced features combine to make RISC CPUs fast.

RISC based workstations usually run some type of Unix operating system. In its own way, the Unix world is as confused as the MS-DOS and Intel one. For years, Unix has been touted as "the standard" by the academic and scientific communities. It is interesting to observe that there are at least twenty different Unix vendors. The operating system level commands between these different brands may or may not be the same and binary files are usually not compatible between operating system vendors (even on the same processor). In fact, one major company has two different versions of Unix that will not allow binary (machine executable) files from one version to work with the other version, even on the same CPU. This means that different machines must have different applications or the source code for the applications must be recompiled for all the different machine types. One reason Unix based machines are not often found as personal computers in homes is because few people have the knowledge necessary to maintain the operating system.

Recompiling applications to run on different CPU chips or operating systems is less of a problem than it sounds because Unix computers are rarely found alone. Rather, they come in groups, inter-connected by a network with a system administrator whose duty is to care for the machines. The system administrator recompiles and fine-tunes the different programs so they run on all the different machines. If there are problems, the administrator changes the source code. This is completely different from the world of personal computers. Major corporations that write programs for personal computers like Microsoft or Novell rarely provide source code. This is not a problem for the Unix system administrator because Microsoft and Novell rarely write programs for the RISC based Unix workstations; most Unix programs are either in the public domain or are homegrown. Nor is this fact likely to change in the near future. The diversity of RISC CPU chips and their binary incompatibilities make it difficult to profitably market a product across all the different combinations of Unix systems.

One final difference between RISC / CISC processors chips and Unix / MS-DOS is how they represent different data types. Table 1 shows the size in bytes for the different CPU chips and operating systems we tested.

	short int	integer	long int	float	double float	long double	pointer
Intel/MS-DOS	16	16	32	32	64	80	16 or 32
Intel/Linux	16	32	32	32	64	96	32
MIPS/Irix	16	32	32	32	64	64	32
SPARC/SunOS	16	32	32	32	64	64	32

Table 1 - Data Representations for Different CPU Chips and Operating Systems

With this information as background we chose to test CPU speed with emphasis on memory performance. Table 2 shows the type/brand of CPU, the CPU speed, the size of the data cache and the instruction cache, the size of the unified level 2 (L2) cache, and the main memory of the machines we tested.

CPU	Speed (MHz)	D-Cache	I-Cache	L2 Cache	Main Memory
Intel 486SX	33	8K unified	--	128K	4 MB
Intel 486DX	33	8K unified	--	256K	16 MB

Intel 486DX4	75	8K unified	--	256K	16 MB
Intel 486DX4	100	8K unified	--	256K	16 MB
Intel Pentium	75	8K	8K	512K	16 MB
SunSPARC 10	33	20K	16K	1 MB	32 MB
SGI Indigo MIPS R2000/3000	33	32K	32K	0	32 MB
SGI Indy MIPS R4000	100	16K	16K	1 MB	64 MB
SGI Crimson MIPS R4400	150	16K	16K	1 Mb	208 MB

Table 2 - Types and Characteristics of Tested Machines.

The Intel based machines in this test all ran MS-DOS 6.22 as an operating system. We also loaded Linux (version 2.2.0) onto the 486DX/33, the 486DX4/100, and the Pentium 75 and ran the benchmarks on these machines again with the new operating system. The SunSPARC 10 used SunOS version 4.1.3, and the three Silicon Graphics (SGI) computers use Irix version 5.3.

The Intel 486DX4/75 and the 486DX4/100 deserve special mention. The first is really a 486DX/25 chip and the second is a 486DX/33, both have a clock tripler for the on-chip operations. These two chips still communicate with the memory, disks, and video at 25 MHz. The "4" after the DX should really be a "3" - the 4 is an Intel marketing ploy. Based just on the CPU's clock speed, the 486DX4/75 should always be 2.25 times faster than a 486DX/33, and the 486DX4/100 should be 3 times faster. However, the 486DX4/75 is usually only 1.5 times faster than the 486DX/33 on any benchmark. The 486DX4/100 is normally 2.6 times faster than the 486DX/33. The reason these chips' performance does not increase commensurate with the CPU speed is that the off-chip communications to memory and disk are still running at 25 MHz. On a final note, the 486DX/33 and the 486DX4/100 are actually the same machine chassis with different CPU chips mounted in the main system board.

III - The Benchmarks.

We used the SPECMark results as a measure to test our benchmarks against because the SPECMarks test a wide range of CPU and memory capabilities, and because the SPECMarks are available for nearly all CPUs. However, while we wanted our benchmarks to provide results similar to the SPECMarks, we wanted our benchmarks to be able to run within a reasonable period of time. Because all the manufacturers do not provide complete data on their SPECMark tests, we made the following simplifying assumptions.

1) All the Intel provided SPECMarks were run under MS-DOS. MS-DOS is the most common Intel based operating system so this is a fairly safe assumption.

2) The different versions of SunOS, Sun Solaris, or Irix do not have a significant impact on the speed of the benchmarks. For example, SunOS version 4.1.0 versus version 4.1.3 will not have an effect on the run time of the programs.

As a partial remedy to the problem with the SPECMarks we obtained a copy of the Dhrystone program [Weicker-a]. Dhrystone is one of the original integer benchmarks; although Dhrystone is not the best benchmark available, we could compile and run Dhrystone on all the different CPUs and have another means to compare the new benchmarks.

There are three basic benchmarks in our suite: an integer, a floating point, and a pointer chasing benchmark. The integer and floating point programs both work in the same basic manner, although the specific operations they perform are different. Both of these programs work on the principal of "The Sieve of Erasthosenes" but they are not searching for prime numbers. Instead, these programs iteratively process an array. Every pass through the array does one array cell fewer than the last pass. For example, if the first pass through the array does an operation on cells 1..2000, the second pass only does the operation on cells 2..2000. Because the size of the loop changes with every pass, the possibility of the compiler doing loop unrolling is lower. The user can choose the number of times the benchmark runs so accurate readings can be made on different speed machines. The programs do a significant number of operations within a fairly small code and a variable amount of memory space. Memory access is also stressed because the array is too large to fit into the on-chip cache.

The integer array program initializes the array with the same number as the array index. For example, the array cell number [641] will initially have the value 641 placed in it. The program executes an additive pseudo-random number generator on every cell. The formula for this additive number generator is $X[n] = (X[n-1] + X[n-j]) \bmod M$.

The floating point program has two arrays, X and Y. The program initializes and computes the first array with the same pseudo-random number generator the integer program uses. However, every cell in the second array (Y) is set to $Y[n] = \sqrt{X[n] * X[n-1]}/n$. This formula performs one floating point multiplication, one floating point division, and one square root operation, in addition to the floating point addition from array X.

Both the integer and floating point benchmarks use a fair amount of memory space. The integer array has 4,000 cells and the floating point array has 3,000. The integer array requires 16,000 (4,000 cells * 4 bytes/cell) bytes of memory for storage while the floating point array

needs 24,000 bytes (3,000 cells * 8 bytes/cell). Because of the differences in how MS-DOS and the various Unix systems allocate memory space for the different data types, we did not always use the same data type for every benchmark. However, we did always use the same size data type for the benchmarks. As shown in Table 1, MS-DOS uses 16 bits to represent integers while all the Unix systems use 32 bits. The same size must to be used on the different platforms to have a consistent benchmark. To correct the problem we used the data type "long integer" on the MS-DOS integer benchmark and the data type "integer" on the Unix systems to make all systems to use 32 bit integer representations. Similarly, we forced 64 bit representations for all floating point operations. Both of these changes caused significant performance drops in the benchmarks on the Intel machines when running MS-DOS.

The final benchmark tests pointer chasing, and hence memory manipulation. The program initializes the memory space into a series of pointer blocks arranged in a circular list. Both the number of blocks and the number of pointers per block can vary, as can the percentage of the blocks that are filled. In addition to controlling the number and size of the blocks and the percentage of blocks filled, the number of chases and the number of pointers to chase per iteration can also vary.

Every pointer block is a sequential group of pointers with the last pointer directed towards the next block. The location of the next block is randomly selected when the program initializes, so the distance between any two blocks can differ. When a chase begins, the starting block is chosen randomly; the chase proceeds through the sequential pointers in that block, and then goes to the next block of pointers. This continues until the number of pointers to chase is satisfied. If another chase needs to be executed because the total number of chases has not been reached, another start point is chosen randomly and the process begins again. The three settings we used on all machines are shown in Table 3. The number and size of the memory blocks used on this test are rather small. However, we found that if we increased the number of blocks and pointers per block past the point where their product exceeds 16,384, MS-DOS would not support the benchmark because of the 64 KB addressing scheme. This forced us to keep block size small because we wanted identical benchmarks for all the computers. The pointer chasing benchmark allows some interesting comparisons of memory bandwidth if the number of blocks being tested will not fit into the computer's level 2 cache, but those results are not yet complete and are not included in this paper.

Test	# of blocks	pointers/block	blocks filled (%)	chase length	# of chases
Balanced	128	128	50%	10,000	10,000
Small Blocks	16	1024	50%	10,000	10,000
Large Blocks	1024	16	50%	10,000	10,000

Table 3 - Pointer Chasing Tests

On the Unix and Linux systems, the benchmarks were compiled using the system-provided GNU C compiler from the command line. For the MS-DOS and Windows tests we used the Borland command line compiler (BCC) provided with the Borland C++ compiler, version 4.5. The code was optimized on the GNU C compiler with the "-O" flag and with the "-Oxt" flag on the Borland compiler. We optimized the code in order to "level the playing field." The benchmark tests compare two sets of machines, Intel based machines running either MS-DOS or Linux, and machines running Linux or Unix. Because the Unix machines with RISC processors rely heavily on optimizing compilers for their speed, not optimizing the code for the RISC machines might provide the Intel CISC CPUs an unfair advantage. Once we optimized the code for the RISC CPUs, we had to provide the Intel machines the same privilege. In fairness, this paper's most important result comes when comparing CISC CPUs under Linux against RISC CPUs under Unix. In this case, all the systems used the same basic GNU compiler with the same optimization settings.

IV - Results.

Because of the varieties in times and statistics reported on the different benchmarks, it is not possible to graph all the tests on the same chart unless they are normalized on one machine. The 486DX/33 machine was used on every test and under both operating systems. We used this machine as the base machine to normalize all the other results.

The first group of machines is shown in Figure 1. The only benchmark that data is not available for is the SPECfp for the 486SX/33. Intel did not publish SPECfp marks for the 486SX chips. We assume these figures were not posted because of these CPU chip's poor performance on the floating point benchmarks. In all fairness, these type chips do not have a floating point unit and must do all floating point operations with software emulation. The 486SX/33 machine's performance on our floating point benchmark was sixteen times slower than the 486DX/33 machine that has a floating point unit.

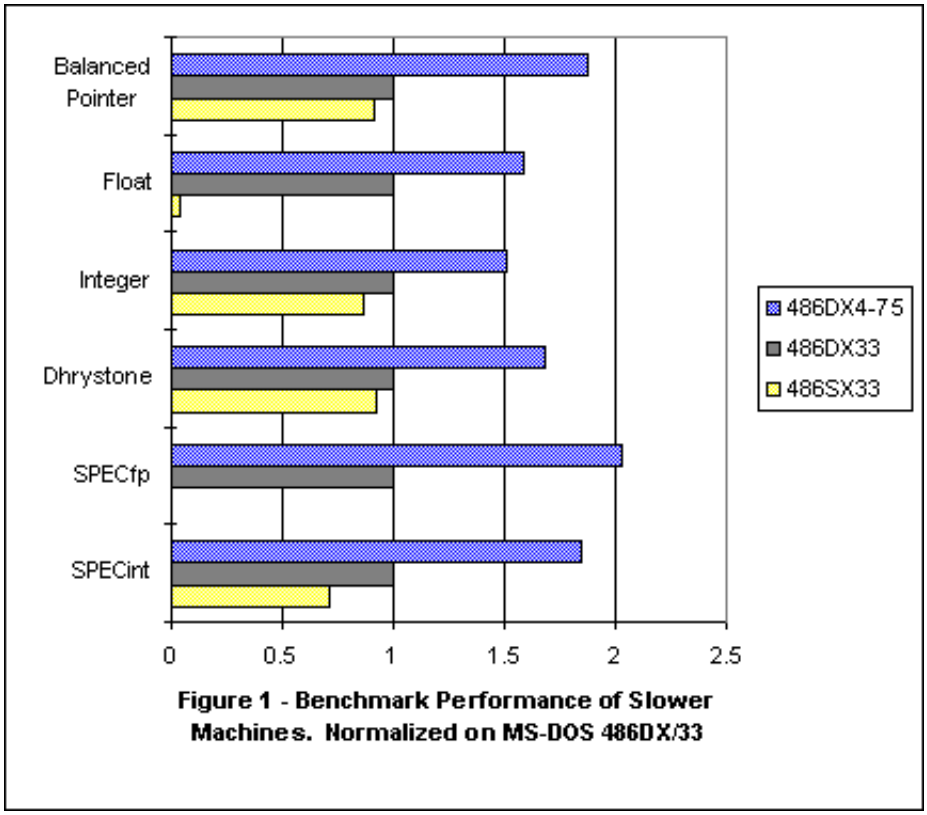

Figure 1 - Benchmark Performance of Slower Machines. Normalized on MS-DOS 486DX/33

Figure 2 shows the performance of all the other machines on the benchmarks. Please note that the SPECMarks shown for the Intel 486DX/33, 486DX4/100, and the Pentium 75 are the same under both MS-DOS and Linux. Again, this is because there is only one set of numbers from Intel for each processor.

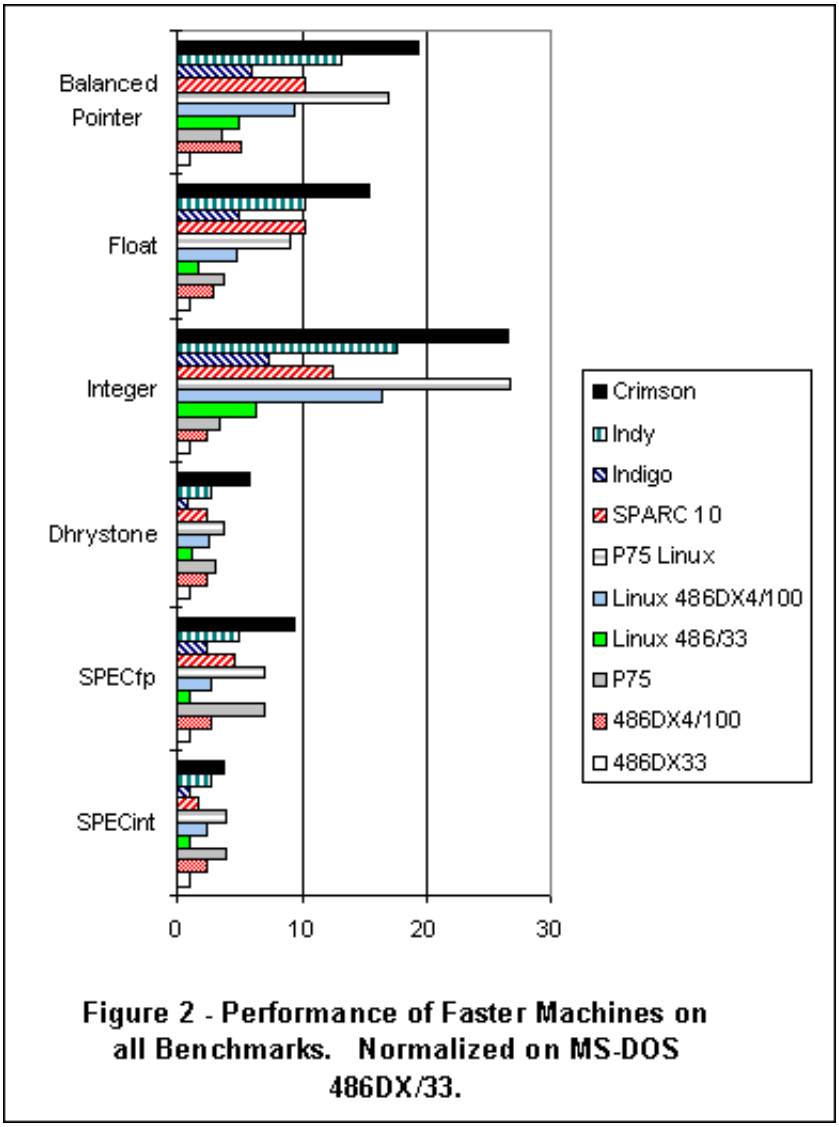

Figure 2 - Performance of Faster Machines on all Benchmarks. Normalized on MS-DOS 486DX/33.

Again, the benchmarks scale well, although not perfectly, across the different processors. For example, the SPARC 10 is faster than the SGI Indigo in all benchmarks. Not only is the SPARC faster than the Indigo, the relative difference between the two stays the same. The most interesting thing to note on the chart is the increased performance of the 486DX/33, the 486DX4/100, and the Pentium 75 when they run the same benchmarks under Linux instead of MS-DOS and Windows. Figure 3 shows just these three processors under both MS-DOS and Linux for all the benchmarks.

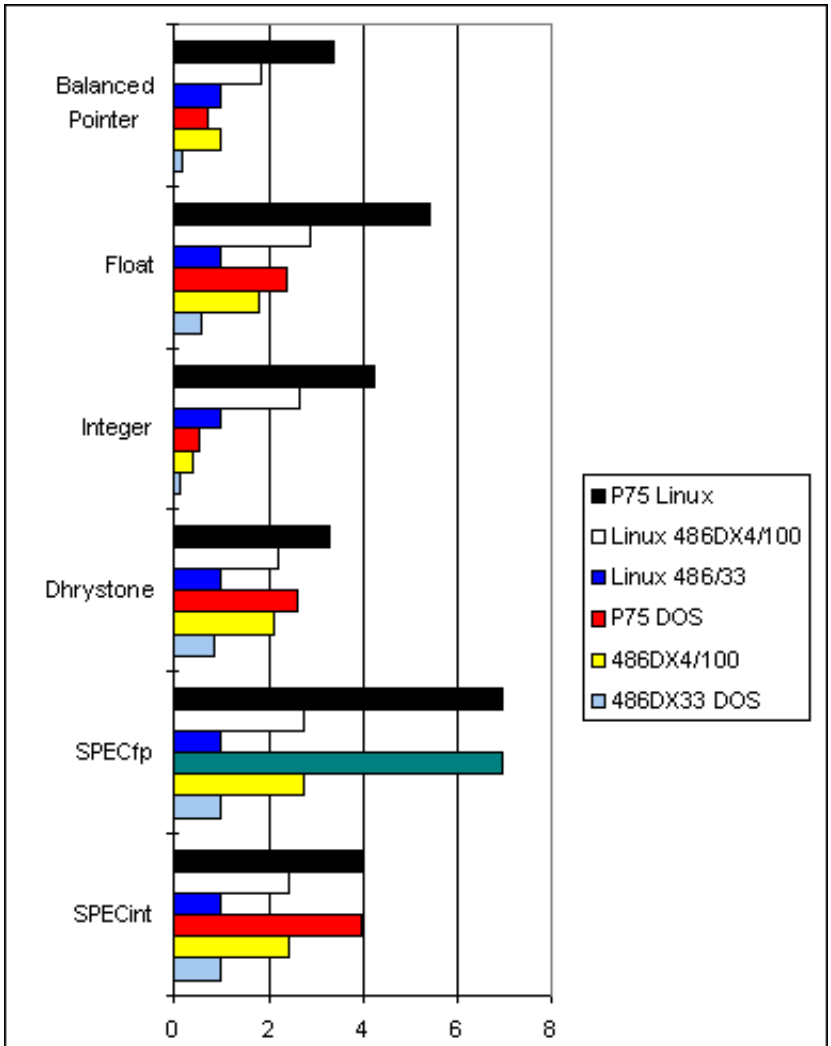

Figure 3 - Performance of 486DX/33, 486DX4/100, and Pentium 75 on all Benchmarks with both MS-DOS and Linux. Normalized on Linux 486DX/33.

With Linux, the two 486 chip's performance increased by a factor of approximately 1.6 on the floating point benchmark and by 6.5 on the integer benchmark. The Pentium's performance improved by 2.29 and 7.67 for the floating point and integer benchmarks, respectively. Some of the increased speed can be attributed to the fact that Linux is a 32 bit operating system while MS-DOS and Windows are 16 bit operating systems. Part of the improvements in the Dhrystone, integer, and floating point benchmarks can be credited to the 32 bit operating system. However, much of the increased speed for these benchmarks, and all of the increased speed for the pointer benchmark is for another reason. When MS-DOS and Windows do pointer operations on large memory blocks, the CPU must execute four instructions to load a memory location into the CPU. This is from the 64 KB segmented addressing scheme MS-DOS uses [Chen]. Linux does not have this problem and only uses one instruction instead of four to load or store values from memory. Hence, the four-fold performance increase in the pointer chasing benchmark that primarily tests only this memory loading function.

The Intel CISC based machines clearly perform better with Linux than with MS-DOS. Figure 4 shows the different systems and how they performed on the benchmarks when all the systems used only Linux or Unix. An abbreviated version of Table 1 follows the chart.

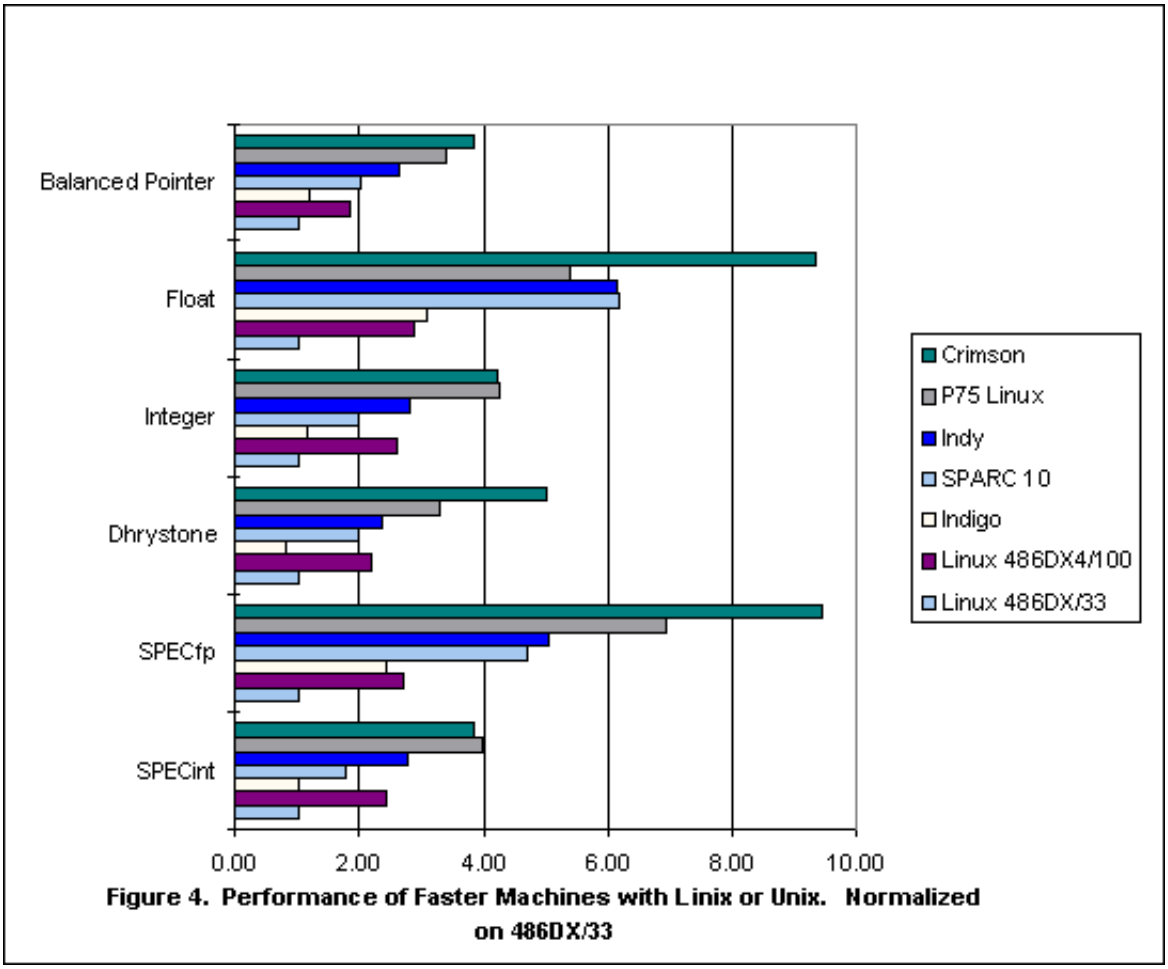

Figure 4. Performance of Faster Machines with Linix or Unix. Normalized on 486DX/33

CPU	Speed (MHz)	Data Cache	Instruction Cache	L2 Cache	RAM
Intel 486DX	33	8K unified	--	256K	16 MB
Intel 486DX	100	8K unified	--	256K	16 MB
Intel Pentium	75	8K	8K	512K	16 MB
SGI Indigo R2000/3000	33	32K	32K	0	32 MB
SunSPARC 10	33	20K	16K	1 MB	32 MB
SGI Indy R4000	100	16K	16K	1 MB	64 MB
SGI Crimson R4400	150	16K	16K	1 Mb	208 MB

Table 4 - Processor Statistics and Associated Memory

The surprising fact that Figure 4 and Table 4 establish is that the Intel CISC CPUs do well against the RISC CPUs when they run a similar 32 bit operating system. This is in spite of the fact that the Intel machines are much less capable in terms of caches and memory than their RISC counterparts. The only area the Intel CPUs are slightly off the pace is the floating point benchmark. For example, Figure 4 clearly shows the Pentium 75's performance drops off compared with the SGI Indy on the floating point benchmark. The benchmarks also have a close correlation with the SPECMarks; this is discussed more in Section V.

Figure 5 shows how many pointer operations per second the different Unix/Linux machines did on the three different pointer chasing benchmarks (these are **not** normalized). The three benchmarks all chased 100,000,000 pointers. The balanced test had an equal number of blocks and pointers per block. The small block test had a 1 to 64 ratio between blocks and pointers while the large block test used a 64 to 1

ratio (see Table 3). On the CISC machines the performance stayed constant on the different pointer tests, regardless of the block size. The RISC machines' performance was inconsistent; their worst showing was with many small memory blocks.

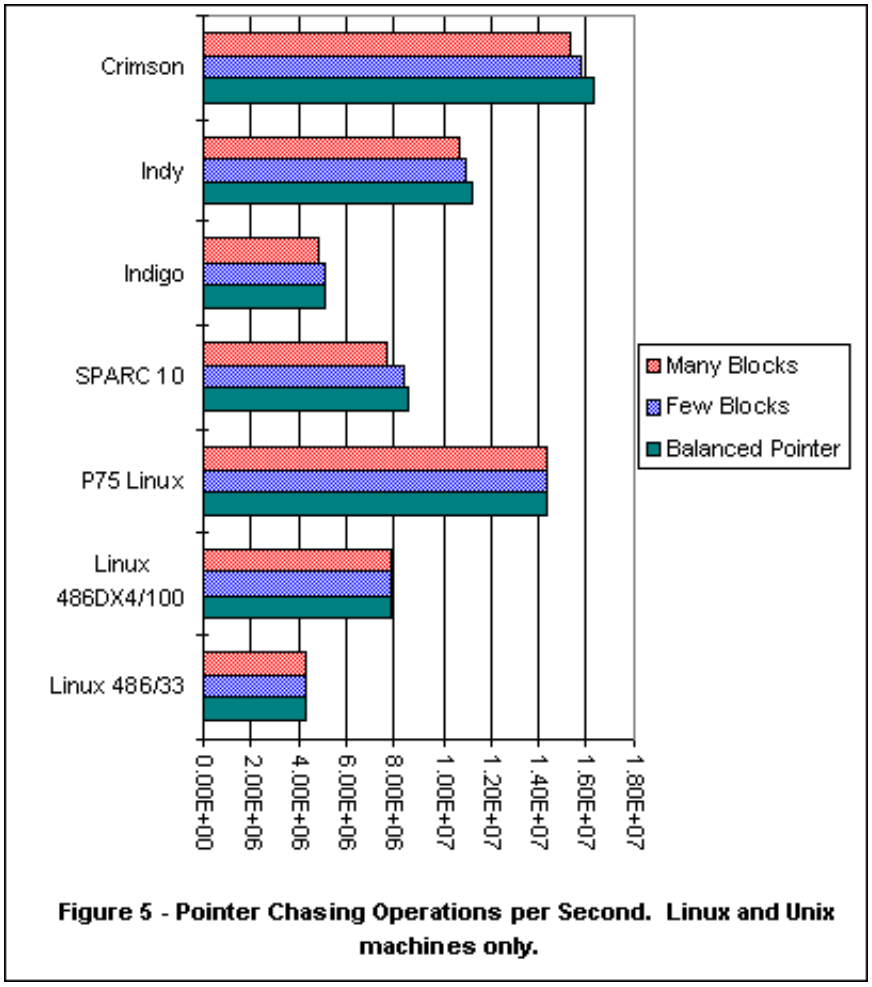

Figure 5 - Pointer Chasing Operations per Second. Linux and Unix machines only.

V - Assessing the Benchmarks.

In general, the benchmark suite we wrote scaled fairly well across the different processors and provided the same general information the SPECMarks gave. Figure 6 shows the correlation between the SPECMarks and our integer and floating point benchmarks. The integer benchmark has a very good correlation with the SPECint. While the floating point benchmark has less correlation with SPECfp, it is fairly close. The benchmarks are not a perfect representation of the SPECMarks; but they do provide a good estimate of SPEC performance, particularly if two machines are being compared against one another. An advantage our benchmarks provide that SPECMarks do not is that they can be run quickly on fairly ordinary workstations.

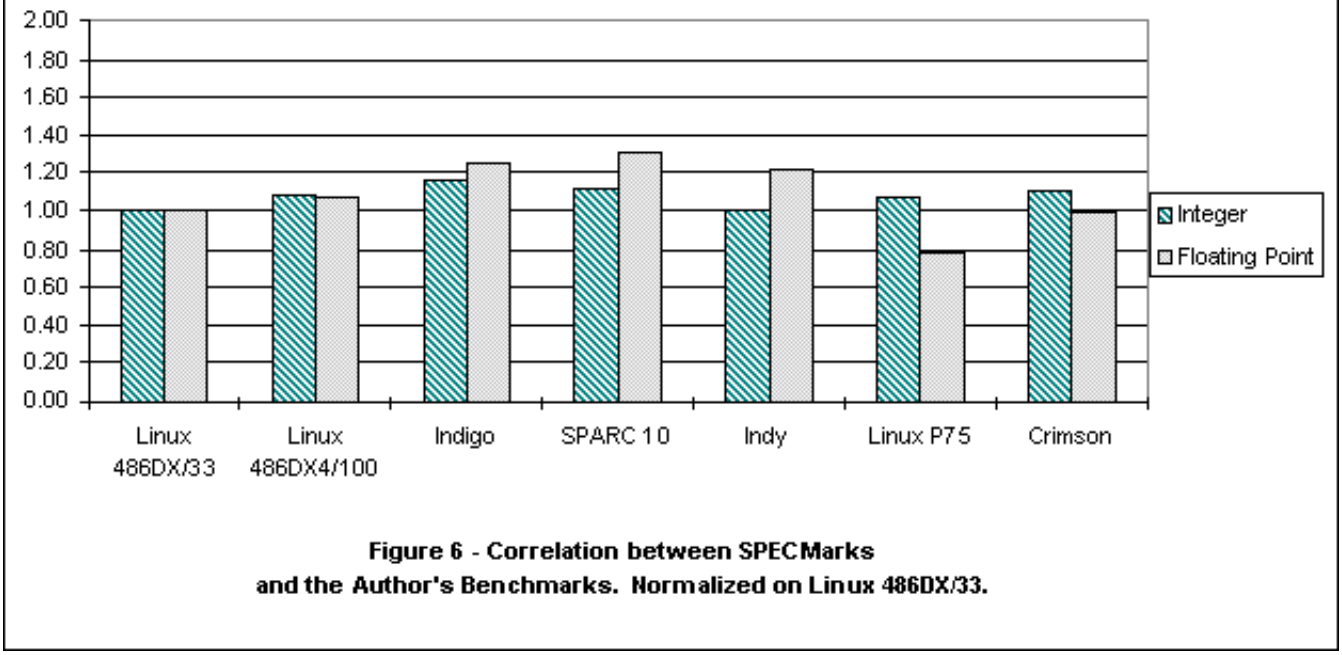

Figure 6 - Correlation between SPECMarks and the Author's Benchmarks. Normalized on Linux 486DX/33.

VI - Conclusion.

When we looked at our results, we were initially surprised. The Intel CISC chips receive a good deal of criticism in the academic and scientific computing communities over alleged performance shortcomings. In contrast, our benchmarks showed that the CISC based personal computers performed well when provided with a level playing field. We decided to investigate further - tests with the venerable ISA bus 486DX/33 and the Xstone benchmark under Linux and X windows clocked in at just over 50,000 Xstones. While this is not close to the over 200,000 Xstones fast X terminals produce [NCD], a 486DX/33 costs significantly less than a X terminal. EISA personal computers can be counted on to provide up to 125,000 Xstones. Pentium based machines with PCI and VL busses are even faster and can provide up to 150,000 Xstones. [Phillips]

When both the performance and the price of the Intel CISC based machines are compared with those of the RISC workstations you find that the CISC machines do as well or better than many of their RISC counterparts and cost a fraction as much as the RISC machines. The major architectural difference between a workstation and a fast Pentium PC is the Pentium's hard disk, multimedia add-ons, and slower bus speed. If need be, the hard disk and add-ons can be removed and the machine can boot off the network with either MS-DOS or Linux. While the PC's slower bus speed is a definite drawback for some scientific and graphics work, most of the tasks an X-terminal or workstation are used for will not even strain an ancient 8 MHz ISA bus from the early 1980's. The faster PCI and VL bus architectures should be adequate for almost any job. The argument for the CISC machines increases even more when cost is considered. A Pentium 75 MHz machine without any multimedia add-ons costs around $1,200, or approximately one-eighth the cost of an SGI Indy. A Pentium 133 MHz machine costs less than $2,500 and should outperform the more expensive SGI Crimson on integer and memory operations.

We believe that many organizations planning to buy workstations are well advised to use CISC based personal computers running Linux or another free version of Unix. Adding another machine architecture and another "flavor" of Unix is not a significant problem for the Unix system administrator. The organization probably has some personal computers already and will gain economy of scale when purchasing more personal computers. Because the same personal computer can be set up to use both Unix and MS-DOS, the new computer can be used for more than one purpose. University's labs can be set up so that one type of machine could serve as both a Linux workstation for the engineering or computer students and as a MS-DOS / Windows personal computer for the other students. Business or administrative staff can have a similar setup. Programmers without high speed graphics requirements may use a Linux/Unix configured personal computer. This is not to say CISC based personal computers can replace all RISC workstations. They cannot, nor do we suggest this. But, there is definitely a place for personal computers in the low and mid-level workstation market. When new networks and upgrades are being planned and purchased, the Intel CISC architecture machines should not be ignored because of the old MS-DOS shackles. They are perfectly adequate for most jobs when provided with a good operating system.

Sources

[Aburto] A. Aburto, "Benchmark Problems and Pitfalls," *Byte*, June 1988.

[Ahmad] M. Ahmad, The Computing Speed of a New Machine," *Computer Journal*, May 1973.

[Byte-a] "The Byte Benchmarks" (with explanations), *Byte*, from the Internet, 1995.

[Byte-b] "A World of Benchmarks," *Byte*, October 1994, from the Byte Internet archive.

[Chen] J. Bradley Chen, et al., "The Measured Performance of Personal Computer Operating Systems," Conference Proceedings, 15th ACM Symposium on Operating System Principles, Association of Computing Machinery, December 1995.

[Fox] R. Fox, "Why MIPS are Meaningless," *Byte*, June 1988.

[Dowd] K. Dowd, *High Performance Computing: RISC Architectures, Optimizations, and Benchmarks*, O'Reilly and Associates, 1993.

[Fleming] P. Fleming and J. Wallace, "How not to lie with statistics: The correct way to summarize benchmark results." *Communications of the ACM*, March 1986.

[Intel-a] "*Performance Data for Mature Intel Processors*," Intel Corporation,

1995, from the Internet.

[Intel-b] *Pentium Processor Performance Report*, Intel Corporation,

1995, from the Internet.

[Jain] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, New York, 1991.

[Kendal] B. Kendal, "How Fast is Fast?" *Byte*, February 1989.

[Muchsel] R. Muchsel, "Performance Measures," *Computer Journal*, May 1982.

[NCD] "NCD Unveils Industry's First 64 bit X Terminals," Network Computing Devices. Downloaded from the Internet.

[Nichols] B. Nichols, "That B Word," *Byte*, June 1988.

[Patterson80] D.A. Patterson, D.R. Ditzel, "The case for the reduced instruction set computer." *Computer Architecture News*, October 1980.

[Phillips] K. Phillips, "Feature Toting PC X-servers rival X terminals," Sun World Online. November 1, 1995. Downloaded from www.sun.com.

[Price] W. Price, "A Benchmark Tutorial," *IEEE Micro*, October 1989

[Sill] David Sill, *Benchmark FAQ*, from the Internet, 1995.

[SPEC-a] *Frequently Asked Questions and Answers about SPEC Benchmarks*, System Performance and Evaluation Cooperative (SPEC), SPEC Group, 1995.

[SPEC-b] *SPEC Benchmarks*, SPEC Group, 1995.

[Weicker-a] R. Weicker, "Dhrystone: A synthetic systems programming benchmark," *Communications of the ACM*, October, 1984.

[Weicker-b] R. Weicker, "An Overview of Common Benchmarks," *IEEE Computer*, December 1990.

[Yager] T. Yager, "Bringing Benchmarks up to SPEC," *Byte*, March 1996