PC Desktop Performance and Hardware Performance Counters

Andy Martin
Texas Instruments

After years of looking at software and hardware data in a mainframe environment, I had the opportunity to look at what was available on x86 based PCs running Windows 95. It took a while to find useful software data but the hardware data available through the hardware performance counters are very exciting.

The paper begins by describing the evolving architecture of desktop PCs and the performance metrics of interest in evaluating them. It then describes some of the more interesting counters, references an earlier study using the counters, talks a little bit about the software to enable use of these counters, and describes typical values of instructions per cycle and other metrics of interest running real PC applications.

1. INTRODUCTION

This paper describes some of the issues affecting the hardware performance of desktop personal computers and some of the ways in which we can evaluate that performance. The paper references a few of the available workloads that use real PC applications to exercise the PC system; such workloads provide a reasonable context for performance studies.

Since the author spent a number of years looking at the performance of mainframe hardware for a vendor and has only recently started looking at PC performance, the paper will present a mainframe context for many of the issues discussed. The primary topics are the choice of workloads and the available hardware measurement facilities.

2. DESIGN ARCHITECTURE

This paper assumes that the reader has a general familiarity with computer architecture. The book by Hennessy and Patterson is a good general discussion [Hennessy]; Flynn has recently written an alternative textbook [Flynn].

McCormack gave an overview of the basic approaches processor designers use to achieve high performance at CMG 87 [McCormack 87]. Johnson gives a more specialized treatment of some microprocessor design issues related to performance [Johnson]. Finally, Tomasulo described some of the techniques used in recent high performance microprocessors in a (much earlier) paper describing the pipeline for the 360-91 [Tomasulo].

The emphasis of this paper is on PC systems running Windows 95. [Chong] gives a good description of the structure of Windows 95 and several good references for Microsoft systems generally.

One of the difficulties in teaching computer architecture has been the absence of useful lab exercises. The measurement technologies described later in this paper may be a way to address this.

While there are lots of variations of the Pentium and P6 processors, we will ignore those for the purposes of this section. (We will use the generic term P6 to refer to the processors marketed as Pentium Pro, Pentium II, and so on.) The Intel reference manuals (often referred to as databooks) are the primary source for description of these processors [Intel 95, Intel 96a].

The important thing for this discussion is that the Pentium and P6 define somewhat different interfaces with the rest of the system.

This paper follows the recent practice of using personal computer to refer to a system based on the IBM PC definition using a microprocessor that implements the x86 or IA-32 functional architecture. (Intel tends to use the term IA-32 (Intel Architecture 32 bit); the rest of the industry tends to use the term x86, referring to the sequence of microprocessor numbers.) Some other major components in such a PC are the chipset, secondary cache, memory, graphics

accelerator, and a hard disk drive. Most of those terms are rather familiar; the term chipset refers to the system controller function on a PC. Other components, such as audio cards, are normally present as well.

This paper will have frequent reason to refer to the cycle time (or clock rate) of a PC system. In talking about that, it is important to distinguish the value for the microprocessor from that for the bus. Since the system oscillator (which determines the cycle time (or clock rate) is attached to the bus, the microprocessor clocking is derived from the bus clock.

The reader familiar with mainframe or minicomputer systems should notice that the microprocessor in a PC system contains considerably less of the processing capacity than the processor in a mainframe system. The system control functions in the chipset, along with the cache and memory, would be viewed as part of the mainframe processor.

The Pentium interface (which the vendors other than Intel continue to use and enhance) specifies the interface between the microprocessor and the rest of the system. For the results quoted later in this paper, we use a 133 MHz (millions of Hertz, or millions of cycles per second) Pentium processor attached to a 66 MHz bus. In this case, interface logic maintains the ratio of the microprocessor clock rate to the bus clock rate at 2 to 1.

The Pentium processor uses a split cache (16 kilobytes instructions, 16 kilobytes data) on the chip; systems with reasonably high performance use an external cache (256 kilobytes was common). It is important to understand that accesses to data from this external cache use bus cycles and come to the processor at bus speed.

Performance analysts tend to get nervous with such a structure, especially so when the differences between microprocessor clock rate and bus clock rate become substantial. With the early Pentium systems, that was not a major concern; in fact, the earliest Pentium systems had the same clock rate for the microprocessor and the bus.

The final Pentium processors had a clock rate of 233 MHz for the microprocessor with a bus clock rate of 66 MHz, for a ratio of 3.5; in this situation, the bus clock rate tended to dominate system performance.

The design of the P6 bus alleviated the potential bottleneck on the bus in two ways. The primary change was including the secondary cache memory on the same die as the microprocessor itself. While this increased the manufacturing cost of the product, it took this cache traffic off the bus and allowed the cache speed to increase as the speed of the microprocessor itself increased. (Some recent low cost versions of P6 have not included such a secondary cache.)

One special concern is that graphics processing can use up a great deal of bandwidth. Recent examples of P6 based systems have included a separate, special port for graphics (AGP, the Accelerated Graphics Port) which uses separate hardware and thus reduces the load on the PCI I/O bus.

Robert Collins' [Collins] web site is a good source of (rather detailed) technical information on x86 microprocessors and the PC industry in general.

3. WORKLOADS

I had occasion to look into a large number of workloads that people used in evaluating mainframe performance. Bill McCormack discussed in some detail the pitfalls of kernel benchmarking in a paper at CMG 90 [McCormack 90].

Development of realistic, representative workloads for performance testing was (and is) both challenging and expensive. Mainframe vendors do not have a realistic alternative, given the absence of available, reasonable benchmarks. For most customers, relying on published results would give reasonable results. For customers with unusual situations (such as a large IDMS system), relying on the published results increases the risk in moving to a new system.

This is a good place to note that performance groups for a processor vendor have multiple goals. The fundamental goal is an accurate assessment of overall performance in the environments of interest. Another major role involves interactions with the design group to improve product performance.

Good performance is one of the major goals of most processor design projects. While it is a longstanding dream to have an automated way to assess the impact of design decisions, achieving such an environment is still a long way off. Quantitative information on the frequency of

particular situations and the actual value of such parameters as instructions per cycle and misses per instruction is a major contribution of the performance group to the quality of the design.

In the mainframe world, the publicly available workloads are not very useful and the customers do not normally ask for results on such benchmarks. In the worlds of workstations and personal computers, there are some useful workloads that are publicly available and some workloads which vendors are expected to run.

Gibson & Miller [Gibson] reported at CMG 97 that not only are there a great many workstation benchmarks available but also that those machines classified as personal computers do quite well on workstation benchmarks when they are run under a true 32 bit operating system, such as some version of Unix.

One question of interest in considering workloads is what the workload is intended to evaluate. There is a fairly standard distinction between tests of components and tests of systems. Since we are interesting in selecting a variety of independent components separately, it is tempting to design workloads which test such components separately. Conversely, since end users use systems, it is important to understand how the components act as parts of systems.

In this section, we will primarily discuss two workload suites that illustrate some of the significant differences in approach among various practitioners. SPECint was developed by an industry consortium and is the standard workload for workstations. The Ziff-Davis benchmarks were developed by a publisher of PC magazines and are highly specific to Windows systems.

3.1. SPEC

SPEC [SPEC] stands for the Standard Performance Evaluation Corporation and has produced a series of standard benchmarks for several years. The current one is called SPEC95. SPEC has defined an integer suite of programs and a floating point suite of programs. SPEC defines CINT95 as the suite of programs and SPECint95 as the most relevant score on a series of computationally intensive C programs. Similarly, SPEC defines CFP95 as the suite of programs and SPECfp95 as the most relevant score on a set of computationally intensive FORTRAN programs. (This careful terminology is not widely used.)

The integer suite is far more widely used, especially for x86 processors. (One reason is that traditional workstations retain a significant performance advantage over personal computers in floating point computation.) It is useful to know that the floating point version of SPEC exists; we will not discuss it further in this paper.

The C in CINT95 stresses that this is a "component level" benchmark, intended to test the processor and not the system. (I am not familiar with any system level benchmarks from SPEC.) CINT95 contains eight computationally intensive applications written in C; the description that follows is adapted from [SPEC]:

The applications are:

099.go
124.m88ksim
126.gcc
129.compress
130.li
132.ijpeg
134.perl
147.vortex

"go" is an artificial intelligence program which plays the game of "Go."
"m88ksim" is a chip simulator for the Motorola 88K which runs a test program.
"gcc" is a new version of GCC (a public domain C compiler) which builds SPARC code.
"compress" compresses and decompresses a file in memory.
"li" is a LISP interpreter.
"ijpeg" does graphic compression and decompression.
'perl" manipulates strings (anagrams) and prime numbers in Perl.
"vortex" is a database program.

In summary, SPEC is a benchmark that primarily evaluates the processor, memory, and the compiler. (Vendors can run SPEC on a great variety of processors, since the benchmarks are in C source.)

3.2. Ziff-Davis benchmarks

Ziff-Davis has produced a series of benchmarks, primarily so they can publish results for their publications. For the Windows environment, the primary benchmarks are WinBench for components and Winstone for the system.

Ziff-Davis developed Winstone 96 to run under

Windows 95; the application vendors (including Microsoft) had not released versions of their software that took advantage of Windows 95 features [Ziff-Davis 95a]. One implication of this is that the applications are using the 16 bit system interfaces in this workload.

Winstone 96 is a PC system benchmark that uses real applications and appears to use a version of Microsoft Test to drive them. (Microsoft Test is a program that allows you to emulate arbitrary user requests; this program is useful for regression testing of PC applications as well as performance testing.)

WinBench is another suite from Ziff-Davis that tests components, notably graphics and disk drives [Ziff-Davis 95b].

The categories and applications included in Winstone 96 are:

Business Graphics/DTP:

Adobe PageMaker 5.0a for Windows
Corel Corporation CorelDRAW! 5.0E2
Microsoft PowerPoint 4.0c for Windows

Database:

Borland dBASE 5.0 for Windows
Borland Paradox 5.0 for Windows
Microsoft Access 2.0c for Windows
Microsoft Works 3.0b for Windows

Spreadsheet:

Lotus 1-2-3 Release 5 for Windows
Microsoft Excel 5.0c for Windows
Microsoft Works 3.0b for Windows
Novell Quattro Pro 6.01 for Windows

Word Processing:

Lotus Ami Pro Release 3.1 for Windows
Microsoft Word 6.0c for Windows
Microsoft Works 3.0b for Windows
Novell WordPerfect 6.1 for Windows

There are several other popular benchmarks (notably BAPCo) but I did not look at those in detail. Intel uses the iCOMP metric (which is a composite of several benchmarks; the benchmarks included change over time) to rate their processors.

My initial reaction to these two suites was that Winstone was much closer to what typical users would run than SPECint was. The command mix used by Winstone seemed reasonable; I was interested in hearing about their methodology for selecting the command mix but did not succeed in getting any information on this. Given the market segmentation discussed in the next section, Winstone seemed even more clearly the right choice.

4. MARKET SEGMENTATION

While working for a plug-compatible mainframe (PCM) vendor, the question of the potential PCM market and its characteristics had considerable interest. In particular, if the PCM market was different than the overall mainframe market, what should one use as a design target. By the late 80s, the two markets looked quite similar; at one time customers were extremely reluctant to use a processor vendor other than IBM on systems that were critical to their business.

Given that reluctance, the potential PCM market had, for example, a smaller proportion of production batch than the overall mainframe market and it was arguably appropriate to keep that in mind when evaluating design decisions.

When looking at the x86 microprocessor market, I wondered if similar considerations were relevant; in fact, they are probably more relevant. The primary opportunity (for x86 vendors other than Intel) was in the home market rather than the business market.

Michael Slater [Slater] is arguably the most influential analyst of the PC industry; he made the following comment in discussing the position of Intel in the industry: "When it comes to processors for high-end PCs, or for x86-based workstations, or for x86-based servers, Intel's monopoly is absolute."

A variety of characteristics segment the PC market but one of the most important is the Microsoft operating system used. PCs viewed as workstations or servers will normally use Windows NT as the operating system; other business systems use a mix of Windows NT and Windows 95/98 and home systems normally use Windows 95/98.

Given this segmentation of the market, it quickly became clear that, for example, the characteristics of Windows 95 were far more relevant (to x86 vendors other than Intel) than Windows NT characteristics. One reason we did look at NT a little was the expectation that NT would be the basis for future Microsoft systems.

One aspect of PC performance which none of the standard tools capture well is multimedia and games. It is tempting to dismiss this as frivolous and irrelevant; for good or bad, it is pretty clear that this is the performance driver for PCs, especially for PCs for home use. One of many complications for assessing multimedia performance is that both feature sets and performance are evolving rapidly; this makes it impossible to make 'apples to apples' comparisons between two generations of products or for the most part two products in the same generation.

There are some other forms of segmentation of the market that we did not pursue. For example, we did not pursue geographic differences in PC workloads even though we knew that Intel's market share varies significantly by region.

5. USE OF WORKLOADS

In a design support role, you want to do a variety of things with a workload other than run it for overall results. Of course, one of the things of interest is using the hardware measurement capabilities to measure design parameters of interest; we touch on that later in this paper. One other major thing is collecting instruction traces of the workload.

A workload based on source code is ideal for this sort of thing. You have full control of the execution environment; you can run the workload in a different environment; you can run pieces of it. So SPEC is an attractive workload for these reasons.

Winstone is rather the other extreme. Winstone gives a system a score, apparently based on elapsed time. This scoring method keeps track of the time that the various programs run and excludes the time for setting up each of the measurement environments.

Our interim approach was to trace and measure the full workload, including both the setup code and the actual runs. I believe that this gave reasonably accurate results; since we did not have a way to trace or measure only the part of the workload that counted toward the score, we could not prove anything about the accuracy of our approach.

We looked at several possibilities in some detail but did not achieve a solution before the group was redirected to a different project. One approach would use the architectural timing facilities to identify the source of the measurement data. A second would develop internal workloads similar to those in Winstone. A third would involve an agreement with Ziff-Davis on a 'special' license that would give us more access to the internals of the benchmark.

6. HARDWARE MEASUREMENT

I will recall some historical background and then discuss current microprocessor measurement capabilities.

6.1. Background

Well into the 1980s, all IBM processors (and those PCM processors I am familiar with), had a way to connect an external hardware monitor to acquire hardware measurement information. The IBM facility was informally referred to as the tailgate.

In the 1970s, several vendors were supplying hardware monitors to attach to such tailgates and a number of customers were using them routinely. Typically, customers used the monitors to track CPU busy (sometimes CPU busy by key) and channel busy rather than the more detailed measurement data that was available [Raymond].

Since the tailgate provided electrical signals which the customers would probe, use of this facility required some skills (hardware design skills) that were less and less common at customer sites. As RMF (and the other software monitors) matured, sites could get the data with acceptable accuracy in a less painful way. (Lindsay [Lindsay] did point out a systematic bias in the RMF channel busy numbers, but the accuracy provided was apparently sufficient for most requirements.)

During the 1980s, first the monitors became hard to find and then the tailgate interface was not present on new models.

Some of the signals that were available on the IBM 308x series and 3090 series (in addition to CPU busy, key, and channel busy) were:

End of a unit of operation
Data access
Cache movein
Castout

Cross-interrogate hit
Castout due to cross interrogate hit
End of unit of operation was used as an approximation of instruction complete to compute MIPS.

Instructions, accesses, and moveins allowed the computation of either miss ratio or miss rate. (As with some other measures, this is a compromise; the more interesting measure is demand miss, where the CPU is idle for lack of data; moveins are close enough for most purposes.)

Castout describes a write operation from the cache to memory.

Cross-interrogate hit relates to cache coherency in the computer architecture literature. It indicates that the address of a memory reference matched the address of a line in the cache.

Castout due to cross-interrogate hits indicates that the address match resulted in a write to memory. This typically means that the data was modified in the cache.

When the external interfaces went away, the vendors went to internal counters to satisfy their performance measurement requirements. What is new is that some of those internal capabilities are now available to users.

I believe that the capabilities I describe have become fairly standard across current microprocessor families. I will specifically mention references to the measurement counters for Alpha and for PowerPC.

Dick Sites has written about a continuous profiling project using the DEC Alpha measurement counters [Sites]; Black and Shen have mentioned the PowerPC counters in their paper on microprocessor performance models [Black]. As noted, the primary focus of this paper is the performance counters on Intel processors, especially Pentium.

6.2. Pentium counters

Brad Chen and others have described a fascinating series of experiments they have performed using the Pentium counters, primarily looking at personal computer operating system performance [Chen]. They also provide a useful bibliography on PC system topics.

They also describe some of the history of the Pentium counters; early Intel documentation did not reference them but a large number were documented after being described in the trade press.

This is the point where the reader should be wondering what fascinating counters are available; I will describe the Pentium counters in some detail and will give a more general description of the Pentium Pro counters.

The description of the Pentium counters is in the Pentium documentation [Intel 95] section 33.4.

The first item mentioned is becoming fairly common on microprocessors and is clearly regarded as a standard feature for subsequent Intel processors. The Time Stamp Counter is a cycle counter (with 64 bits) for Pentium; Intel specifies (apparently for future processors using this architecture) that the values returned are guaranteed to be unique and monotonically increasing but may not be a cycle count. There is a user level Read Time Stamp Counter instruction (RDTSC) to sample the value. (I doubt the similarities to TOD clock handling on mainframes are a coincidence; the Real Time Clock on PCs is not part of the microprocessor.)

Even if the time stamp counter remains a cycle counter, wrapping during the life of a product is not an immediate concern. Even with a product running at 1 GigaHertz, it would take about 600 years to wrap.

The actual recording of performance metrics is in 2 counters, which are each 40 bits long. The size of these counters does provide some limitations; even at 133 MHz, a counter that could increment by 2 each cycle could wrap in slightly more than an hour.

Now I will begin to describe the available metrics that you can select for the counters. I am not trying to provide enough detail to use these facilities; I have referenced the official documentation and will discuss software tools shortly.

Many of the metrics are self explanatory; when an explanation seems appropriate, I will interrupt the list to include it. Most of these signals are of the occurrence type which count occurrences of a particular event; I will indicate the duration type metrics, which count the number of cycles a situation exists.

Intel has documented a total of 38 Pentium performance monitoring events. The field has six bits and, of course, people have

experimented with all the possible values; there is some speculation on the web on what these values may be measuring.

I have used the order in the Intel documentation and will number them in sequence (which is not done in the documentation).

1. Data Read
2. Data Write
3. Data Read or Data Write
4. Data TLB Miss

As with other processors, the Translation Lookaside Buffer (TLB) stores page translations. A TLB miss typically indicates that a page translation was required.

5. Data Read Miss
6. Data Write Miss
7. Data Read Miss or Data Write Miss

These metrics indicate that the data accessed was not in the on chip cache.

8. Write (hit) to M or E state lines

Pentium follows the MESI (Modified, Exclusive, Shared, Invalid) protocol for cache coherency; the on chip cache is writeback (allows updates to cache lines without a memory access). This counts the write hits that this feature allows.

9. Data Cache Lines Written Back

This counts writes from the on chip cache.

10. External Snoops

A snoop or inquire cycle asks whether a given address is in the cache; external snoops are initiated by some system component other than the microprocessor.

11. Data Cache Snoop Hits

A snoop hit indicates that the specified address was in the cache.

12. Memory accesses in both pipes

The Pentium processor could process two instructions per cycle (with some rather arcane restrictions); each instruction was in a separate pipe. This indicates that both pipes generated a data memory access.

13. Bank conflicts

This is the number of actual conflicts for the same bank of memory.

14. Misaligned data memory or I/O references

Memory in x86 is addressable at the byte level; two or four byte accesses that cross a four byte boundary are misaligned as are eight byte accesses that cross an eight byte boundary. Misaligned accesses see a performance penalty.

15. Code read
16. Code TLB miss
17. Code cache miss

Pentium has a separate cache for code (or instructions). They indicate that code lines are either shared or invalid. The Pentium literature does not mention any special handling for the case where code and data are in the same cache line.

18. Any segment register loaded

The x86 architecture (especially with code that uses 16 bit addressing) makes extensive use of segment registers to maintain the ability to access code and data. This is the number of times one of these registers is loaded.

19. Branches
20. BTB hits

The Pentium uses a Branch Target Buffer (BTB) to minimize the disruptive effect of taken branches. This is the number of times that the branch is in the BTB.

21. Taken branch or BTB hit
22. Pipeline flushes

When the prefetched instructions may be invalid, the pipeline is flushed. Some of the things which cause this are BTB misses on taken branches, mispredictions, exceptions, interrupts, and some segment descriptor loads.

23. Instructions executed
24. Instructions executed in v-pipe

Pentium describes the primary pipe as the u-pipe and the parallel pipe as the v-pipe. The proportion of instructions in the v-pipe is a good indication of the value of the second pipe.

25. Clocks while a bus cycle is in progress (bus busy) (duration)

This is the first duration signal included; note that it counts bus cycles, not processor cycles.

26. Clocks stalled due to full write buffers (duration).

Pentium defers writes until idle cache cycles except when the write buffers are full.

27. Pipeline stalled waiting for data memory read (duration)

This includes delays due to data TLB miss.

28. Stall on write to an E or M state line

29. Locked bus cycle

The Pentium bus architecture implements locked memory accesses using a bus lock and this indicates the number of times that happened.

30. I/O read or write cycle

Bus cycles to I/O space.

31. Non-cacheable memory reads

The x86 architecture allows considerable control over whether data is allowed in the cache. Graphics buffers are the best example of why you might not want data in the cache; this category also includes read cycles due to TLB misses.

32. Stall on address generation interlock (duration)

An address generation interlock occurs when an instruction in the execute stage writes to either the base or index register of an instruction in the address generate stage. (The Pentium pipeline is described in chapter 3 of [Intel 95].)

33. FLOPS

Count of floating points operations executed, which allows computation of floating point operations per second (or FLOPS).

34-37. Match on debug register
38. Interrupts

The debug register counts deserve additional comment. Debug registers are usually thought of as being used for functional checkout while performance counters are used for performance studies. Pentium gives a mechanism to allow one to feed the other, so you can count matches to conditions in debug registers and set an external pin based on the performance count value.

The ability to count debug register matches allows you to count some rather exotic values that you might never have thought you would want to count, such as data accesses to a particular part of memory. It was not hard to convince someone who was responsible for checking out hardware that having the ability to trigger a logic analyzer after, say, 100, 000 data cache misses might be useful in tracking down a relatively obscure problem.

The additional comment is organizational. When I am looking for additional measurement capabilities, it often seems like a hard sell. By contrast, it is a rather easy sell to suggest increased debugging capabilities to anyone who has gone through a full cycle. The incremental effort to tie the two together is relatively slight.

6.3. Pentium Pro counters

An overview of the P6 internal design architecture can be found in [Intel 96a]. Very briefly, in contrast to the relatively conventional pipeline organization of Pentium, Intel describes the P6 as having three separate engines communicating using an instruction pool. (One of the architects has said that they typically have 60-80 instructions in flight at any one time.)

My description of more current counters will be specific to Pentium Pro. I suspect the other P6 based products are similar. Again, very briefly, the Pentium Pro counters are far more specific to the design architecture than the Pentium ones.

In the Pentium Pro documentation the time stamp counter is described as a separate facility from the performance monitoring counters and Intel gives a very specific statement on its characteristics in section 10.5 of [Intel 96b]. “Intel guarantees, architecturally, that the time-stamp counter frequency and configuration will be such that it will not wraparound within 10 years after being reset to 0.”

The performance monitoring counters are listed in Appendix B of [Intel 96b]. I will not list all of these counters but will note some interesting additions.

There is a count called data cache unit miss outstanding, which is described as imprecise but a useful approximation. This should allow an approximate computation of what is sometimes called the miss penalty, the average number of

cycles to resolve a cache miss.

There is a count of the number of cycles for which interrupts are disabled and one for the number of cycles interrupts are disabled and interrupts are pending.

There is a collection of BTB counts including most of the cases of interest. There is an intriguing count called bogus branches (not further defined).

The Pentium Pro designers appear to have taken a systematic approach to identifying the counters of interest.

7. SOFTWARE SUPPORT

Accesses to the performance counters use a programming interface similar to device drivers. We started our work on the Pentium counters with a freeware utility P5Mon [O'Carroll] that reads the counters at regular intervals and puts a graph on the screen. At the time, there were versions of P5Mon for DOS and Windows 3.1/95; a more recent version runs on Windows NT. (Pentium is sometimes called P5.)

Akinlar has a Linux device driver for the Pentium performance counters [Akinlar].

We made a couple changes to P5Mon for our major runs with Winstone; Winstone wants to have full control of the screen and we wanted to collect the measurement data in a file.

Intel offers a number of software performance tools; for instance, they have produced a rather nice profiling tool called VTune [Intel 98a]. Intel also offers a tool for using the P6 counters, as an addition to the Windows NT performance monitor [Intel 98b].

8. WHAT WE FOUND OUT

"Instructions per cycle" (or IPC) is the most quoted metric for a microprocessor. In discussions of mainframes, "cycles per instruction" (or CPI) was more commonly used.

IPC and CPI are inverses, that is, IPC * CPI = 1. They each have a simple relationship to MIPS,

$$\text{MIPS} = \text{IPC} * \text{MHz} = \text{MHz} / \text{CPI}.$$

The original Pentium products had bus clocks equal to processor clocks, with clock rates of 60 MHz and 66 MHz. Based on SPECint results, Intel indicated a value of IPC of about 1.

SPEC has some favorable characteristics for IPC, including low I/O rates, a low level of operating system activity, and relatively low cache miss rates. In addition, the 1:1 ratio of bus clock to processor clock had the effect of reducing the effective cache miss penalty.

Recall that we are reporting on measurements of Winstone running on a 133 MHz Pentium with a 66 MHz system bus. The most interesting values were ratios and we followed the rule of only comparing the two values we obtained in a given run.

While measuring Winstone, we consistently saw values of instructions per cycle of 0.3 to 0.4, giving a value for MIPS in the approximate range 40-50. My belief is that the primary reason for the lower IPC is the difference in workload between SPEC and Winstone.

High level overviews of why the IPC for Pentium should be twice that of the 486 models tended to say that the Pentium had two copies of the x86 pipeline and so would process about twice as many instructions. The proportion of the instructions executed in the second pipe seems relevant in evaluating the effectiveness of the second pipe. This proportion was about 20%.

Finally, the proportion of segment register loads is a good indication of the amount of 16 bit code in the system. (Since 16 bit code requires small segments, crossing segment boundaries happens often.) Recall that this version of Winstone runs under Windows 95 but uses applications written for Windows 3.1. In any case, the number of segment registers loads per instruction was about .07.

This is actually pretty phenomenal, indicating that 1 in every 14 instructions involved a segment register load. This is probably not surprising to anyone who observed the glacial pace of converting mainframe software to 31 bit addressing when it became available; I would be interested in repeating this with current hardware and software.

We did measure bus busy, though not for the configuration described. The values we measured were in the 20% range, though two major factors have tended to increase that. One is the increase in the ratio of the processor clock rate to the bus clock rate; the other is the increase in memory intensive graphics context. The level 2 cache on chip and AGP are intended to compensate for these factors on P6.

I believe that the primary reason for the lower IPC we measured on Winstone is the difference in workload characteristics. The high rate of segment register loads and the moderately high bus busies reflect the differences in workload.

9. SIGNIFICANCE

George Dodson pointed out that, while the performance counters answer some interesting questions, they do not answer some of the most pressing questions an installation has [Dodson]. As I understood it, he described a scenario where end user response time increased dramatically as an example.

Since the design of Windows 95 tends to keep the processor busy with wait loops, there is no simple measurement that will indicate the amount of reserve capacity in a PC system. In addition, this type of data is not very useful in directing an analyst in the direction of a bottleneck, particularly a software bottleneck.

Having said that, this type of data is valuable in gaining insight in ways that have been nearly impossible to get in other ways. I will illustrate that with a couple examples.

I think the greatest value, even with a relatively mature organization, is pedagogical. A reasonable understanding of hardware organization is valuable for either an analyst or a software developer who needs to be concerned about performance. Particularly for those who are habitually hands on, it is difficult to acquire such an understanding without the ability to try things and see the impact. This technology allows such an approach.

This can also be used on a finer grained approach, looking at a particular application. This type of measurement capability can facilitate an enhanced approach to application tuning that would assess the impact of changes on various aspects of the hardware as well as the characteristics visible from the software.

More generally, I have approached a few performance problems over the years for which I wanted any source of data I could find. (Especially the problems that cannot possibly be happening.) Having additional tools is always valuable in that context; this tool does not seem to be widely known in the performance evaluation community and I wanted to help remedy that.

10. ACKNOWLEDGEMENTS

As is usually the case, a great many people were involved in the work I am reporting on.

In addition to the many people who I have worked with over the years on performance and mainframe hardware, I wanted to specially acknowledge a few people who were heavily involved in either the work reported or the preparation of this paper.

I worked with Fred Lewis and Gene Meyer in the Pentium studies reported. Fred taught me about Windows software and adapted the counter software; Gene did the PC system experiments and collected most of the measurement data I have quoted.

Bob Milhaupt and Jim Walsh taught me a great deal about PC system architecture.

Finally, Jim Lawrence and Joan Arnold-Roksandich persuaded me to prepare this material for CMG.

11. TRADEMARKS

marketed and that the trademarks went to the purchasing company when any of these companies were acquired.

12. REFERENCES

[Akinlar] Akinlar, Cuneyt, "A Device Driver for Pentium Performance Counters," http://www.cs.umd.edu/users/akinlar/driver.html

[Black] Black, Bryan and John Paul Shen, "Calibration of Microprocessor Performance Models" IEEE Computer May 1998, p. 59

[Chen] Chen, J. Bradley, Yasuhiro Endo, Kee Chan, David Mazieres, Antonio Dias, Margo Seltzer, and Michael D. Smith "The Measured Performance of Personal Computer Operating Systems, *ACM Transactions on Computer Systems*, February, 1996 p.3

[Chong] Chong, Herbert A. "The Design and Tuning of Microsoft Windows 95," CMG 95

[Collins] Collins, Robert, web site http://www.x86.org/

[Dodson] Dodson, George, comments at a local CMG meeting, 6/97

[Flynn] Flynn, Michael J. *Computer Architecture: Pipelined and Parallel Processor Design,* Jones and Bartlett, 1995, ISBN 0867202041

[Gibson] Gibson, Timothy J. and Ethan L. Miller, "The Case for Personal Computers as Workstations," CMG 97

[Hennessy] Hennessy, John L. and David A. Patterson, *Computer Architecture: A Quantitative Approach* Second Edition, Morgan Kaufman, 1996, ISBN 1558603298

[Intel 95] Pentium Processor Family Developer's Manual, Volume 1, Pentium Processors: Order Number 241428-004, ISBN 1555122450, 1995

[Intel 96a] Pentium Pro Family Developer's Manual Volume 1: Specifications: Order Number 242690-001 ISBN 1555122590, 1996

[Intel 96b] Pentium Pro Family Developer's Manual Volume 3: Operating System Writer's Manual: Order Number 242692-002 ISBN 1555122612, 1996

[Intel 98a] The web site for Intel's software development tools (such as VTune) is http://developer.intel.com/design/perftool/index.htm

[Intel 98b] P6 Family Processor Performance Measurement Utility for Windows NT, a tool called p6perfnt, described at http://developer.intel.com/drg/pentiumII/appnotes/p6perfnt.htm

[Johnson] Johnson, Mike *Superscalar Microprocessor Design*, Prentice Hall, 1991, ISBN 0138756341

[Lindsay] Lindsay, David S. "RMF I/O Time Validation" CMG 80, page 112

[McCormack 87] McCormack, Bill, "High Performance Processor Design" CMG 87, page 736

[McCormack 90] McCormack, Bill, "Kernel Benchmarking: A Technique Past Its Prime," CMG 90, page 1340

[O'Carroll] O'Carroll, Philip, author of P5Mon, up to date copies on ZDnet

[Raymond] Raymond, Bill, personal communication

[Sites] Sites, Dick, Jennifer Anderson, Lance Berc, Jeff Dean, Sanjay Ghemawat, Monika Henzinger, Shun-Tak Leung, Mitch Lichtenberg, Mark Vandevoorde, Carl Waldspurger, Bill Weihl, "Continuous Profiling (It 10:43; Do You Know Where Your Cycles Are?), Hot Chips IX, page 149

[Slater] Slater, Michael, "How Much Power Is Too Much?" MicroDesign Resources, May 12, 1998

[SPEC] Answers to Common Questions About SPEC95 Benchmark Suites, the National Computer Graphics Association is the administrator for SPEC

[Tomasulo] Tomasulo, R. M. "An Efficient Algorithm for Exploiting Multiple Arithmetic Units." *IBM Journal*, volume 11, (January, 1967) pp. 25-33

[Ziff-Davis 95a] *Understanding and Using Winstone 96* Ziff-Davis Publishing Company, 1995

[Ziff-Davis 95b] *Understanding and Using WinBench 96* Ziff-Davis Publishing Company, 1995