A Survey Paper on Processor Workloads

Christoph Jechlitschek, q.wert@gmx.de

Abstract
Processors are ubiquitous in today's world. Almost any electronic device contains a processor, and their computing power ranges from a few thousand to millions of instruction per second. Processors are built for many different purposes, and so their capabilities vary greatly. Even within a specific area, processors are different enough to make it difficult to compare them. Benchmarks have been developed to measure performance of processors and rank them among their peers. Much research has been conducted to design and develop benchmarks capable of reflecting a processor's performance.

This paper surveys benchmarks in five major processor families: general purpose processors, graphics processors, network processors, embedded processors, and processors in supercomputers. This paper also covers some basics about benchmarks.

Keywords: processor workloads, benchmarks, performance evaluation, performance analysis, performance measurements, supercomputers, graphic processors, network processors, embedded processors, general purpose processors, performance metric, Dhrystone, Whetstone, Fhourstone, SPEC CPU, SiSoft Sandra, LINPACK, LAPACK, Lawrence Livermore Loops, NAS Parallel Benchmarks, 3Dmark06, SPEC viewperf, CommBench, PacketBench, EEMBC

See also: Other Reports on Performance Evaluation

Table of Contents

---

1. Introduction

Processors are the brains of many devices. They can perform computations; interact with other devices, and many other things. They come in many sizes and shapes, vary in speed and have different instruction sets, and are designed for different uses. With this multitude of processors, it becomes important for us to determine how well a processor performs in its designed area and how it compares to other processors. In this paper, we start by explaining what benchmarks are and what they measure. It is important to understand the different metrics to make sure to select the right criteria when making comparisons. In the following sections, we present benchmarks for different classes of processors. We start by covering benchmarks for general purpose CPUs in desktop systems, followed by benchmarks for those CPUs in supercomputers. After that, we cover special purpose processors and their benchmarks. Special purpose processors covered are graphic, network, and embedded processors.

Back to Table of Contents

---

2. Benchmark Background

In this section, we cover some of the background material that is needed to understand the rest of the paper. We first explain the different kinds of benchmarks that a processor can be subjected to, and discuss what they measure and whether they reflect just the speed of the processor or also include the interactions of the processor with peripheral devices. After that, we cover some common metrics that are used to report the performance of processors. Before we start with these two sections, we cover the origin of the word benchmark as a side note.

2.1 The Origin of the word Benchmark

The term benchmark was created and used by surveyors. After surveying an area, surveyors had a great interest in indicating the exact position from where they took their measurements so that they could reposition the leveling rod at a later time. To do so, the surveyors chiseled marks in which they placed an angle-iron to bracket (bench) the rod. From there, the word benchmark found its way into computer science where it changed its meaning to mean evaluation of computer performance. Next, we describe five different groups of benchmarks.

2.2 Types of Benchmarks

Benchmarks can be roughly divided into 5 groups [Jain91]. Those groups are:

- Addition Instruction
- Instruction Mixes
- Kernels
- Synthetic Programs
- Application Benchmarks

We proceed by describing each group in more detail.

2.2.1 Addition Instruction

In the early times, the CPUs were "the most expensive and most used components of the system" [Jain91] so performance was measured by counting how many additions a computer could perform per unit time. Since the addition instruction was the most common instruction and because computers had very few other instructions, comparing the number of additions performed was a reasonable benchmark for early computers.

2.2.2 Instruction Mixes

Measurements based on a single instruction had the disadvantage that they did not take into account that other instructions had different complexities. As the number of instructions grew, it became clear that a mix of instructions was needed. The number of times each instruction appears in the mix should ideally reflect the frequency with that the instruction appears in an average program. The most popular mix is the Gibson mix which was developed by Jack Gibson in 1959. The Gibson mix is shown in table 1.

Table1: This table shows the Gibson mix ordered after the frequency of the instructions. This data was taken from [Jain91].

Instruction	**Percentage**
Load and store	31.2
Indexing	18.0
Branches	16.6
Floating Add and Subtract	6.9
Fixed-Point Add and Subtract	6.1
Instructions not using registers	5.3
Shifting	4.4
Compares	3.8
Floating Multiply	3.8
Logical, And, Or	1.6
Floating Divide	1.5
Fixed-Point Multiply	0.6
Fixed-Point Divide	0.2

2.2.3 Kernels

Two reasons caused the focus to shift from instruction mixes to kernels. The first is that emerging processors continued to increase the number of instructions in their instruction set. Therefore, it became more and more difficult to include all instructions in the mix. The second reason was the introduction of advanced features such as pipelining and address translations. Execution time of an instruction started to depend on the instructions that were issued in its vicinity. To account for it, researchers started to use the most frequent functions in a program. Later, the focus shifted from functions to operations such as matrix inversion and sorting so that performance could be compared across different systems. Most, if not all, kernels are processing kernels. These kernels do not issue I/O instructions and therefore measure the performance of the processor without regards to the surrounding system.

2.2.4 Synthetic Programs

The disadvantage of Kernels is that they do not reflect the behavior of a typical application. Applications, unlike Kernels, issue system calls and interact with I/O devices. Synthetic programs were developed to account for this by exercising a mix of computations, system calls and I/O requests. The main advantage of synthetic programs is that they can be quickly developed and do not rely on real data. To assure portability, these programs were written in a high level language. However, these programs are far from perfect. The main drawback being that they are too small to make statistically relevant memory or disk references. Similar issues arise with page faults, disk cache hits/misses, and the CPU-I/O overlap.

2.2.5 Application Benchmarks

By far, the most accurate way to get measurements reflecting application program performance is to use applications as benchmarks. This is especially true in areas where there are only one or very few application programs such as banking (credit-debit transactions). For a more general environment, a set of 'typical' applications is used to compute an overall performance score. These benchmarks are generally identical to the original applications with the exception of small modifications made to automate the evaluation process. Application benchmarks are also not perfect since they evaluate the whole computer instead of just the processor (since we are talking about processor benchmarks).

2.3 Benchmark Metrics

Before we can start measuring and comparing the performance of different processors we need to review the metrics that are commonly used. One of the earliest metrics is MIPS - million instructions per second. This metric simply counts how many instructions a processor can execute each second. Many computer scientists regard this as meaningless [Hennessy03] and have even nicknamed it as "Meaningless Indication of Processor Speed" or "Meaningless Information on Performance for Salespeople" [wiki-MIPS]. There are two reasons for this. First, MIPS does not reflect how much work is done per instruction. For example, RISC processors have simple instructions, and even though they can execute instructions faster than CISC processors, they do not necessarily do more work. Second, different instructions can take a different amount of time to execute. For example, add instructions execute faster on most processors than branch instructions. Most MIPS ratings are based on the fastest executing instruction and should be considered as Peak-MIPS, while MIPS rating under "normal" use will be lower. Here again, it is problematic to compare different processor architectures as a processor with a single very fast instruction and other slow instructions will have a higher MIPS rating than processors with all fast instructions, although the latter will execute most programs faster. As a side note, Linux uses so called bogoMIPS. BogoMIPS are computed during the startup of a Linux system and are used to calibrate timers for waiting loops depending upon the processor speed. It is conjectured that bogo comes from the word "bogus" which means fraudulent or having a misleading appearance.

The scientific computing community was more interested in how many floating point operations per second (FLOPS) a processor can perform. Early processors did not posses a floating point unit and had to use their integer unit to perform floating point operations which was very slow. Because of that, FLOPS were much lower than MIPS. Current processors usually have multiple floating point units to efficiently pipeline floating point operations. FLOPS are a decent measure of a processor's speed, but do not reflect the computers performance as it does not account for I/O or memory performance.

Graphic cards and game consoles often report a FLOPS rating that is much higher than that of a general purpose processor [wiki-FLOPS]. Such ratings have to be treated with caution. Sometimes, these ratings are based on the performance of a single-purpose texture filtering unit that performs a simple computation very fast. Even if this is not the case, the graphic processors (GPU) are advantageous as most 3D graphic operations are highly parallelizable and therefore can be executed on multiple units to increase the throughput. Moreover, GPUs do not implement slow branch instructions which frees a lot of transistors (especially from the branch predictor) that can be used for additional processing units.

Application benchmarks usually report either the runtime of the test or how many iterations of the test completed in a specific amount of time. While such results are better suited to compare processors with different architectures, they usually contain more than just the processor speed. Most benchmarks are rather complex and rely on caches and fast I/O to keep the processor busy. Benchmarks that make many system calls also reflect the performance of the underlying operating system.

Last but not least, the compiler used to compile the benchmarks has a big impact. Optimizing compilers, for example, were reported to discard 25 percent of the Dhrystone code [Hennessy03]. Not only discarding code, but also rearranging instructions to fill in branch delay slots, will increase performance. Another possibility is to rewrite parts of the source code to take advantage of a processor's special instructions. To address such compiler issues, different benchmarks employ different methods: some require that the score for the optimized as well as the unoptimized benchmark be reported, while some do not allow changes to the source code, while still others allow the source code to be rewritten (even with different algorithms) as long as the final result is the same. Some benchmarks restrict the processor manufacturers to a single compiler or to a certain set of compiler flags, while others, especially for embedded processors, allow assembly language coding [Hennessy03].

A popular benchmark metric for computer game players is the frame rate that can be achieved in 3D games. 3D games put a high load on the CPU and GPU, as well as on the main memory and graphic card memory [tomshardware]. So the result is a combination of the performance of at least those four entities. Mostly, the games Quake and Unreal tournament are used to report the frame rate. Both games are regarded to be the de facto standard benchmarks in this field. Graphics card manufacturers often report the frame rate achievable with their graphic card to appeal to their targeted audience.

In this section, we discussed some of the most widely used metrics to report the outcome of benchmarks. The ones that were mentioned are MIPS, FLOPS, time taken or number of iterations, and frame rate. We saw that there are several difficulties in comparing processors of different architectures for some metrics while others report the performance of more then just the processor. In conclusion, we can say that there is no single benchmark metric that is best for every purpose.

Back to Table of Contents

---

3. Benchmarking for Desktop Computers

In this section we present some popular benchmarks that test the processor in desktop computers. We cover

the following five benchmarks:

- Whetstone
- Dhrystone
- Fhourstone
- SPEC CPU 2006
- Sisoft Sandra

3.1 Whetstone

Whetstone was created by Brian Wichmann in 1972 in the United Kingdom (British Central Computer Agency [Jain91], National Physic Laboratory [wiki-Whetstone]). It was first written in ALGOL60 and contains 11 modules, which where designed to resemble the instruction mix used in 949 ALGOL programs. Since then, it has been extended several times by Roy Longbottom and others [longbottom]. Because there are different versions available, it is important that the same version of Whetstone is used when comparing two systems. Although it measures integer and floating point (FP) operations, it is widely considered to be a FP benchmark. The results of the benchmark are reported as KWIPS (Kilo Whetstone Instructions Per Second). Whetstone is a synthetic benchmark because it contains only an instruction mix and no "real" code. Since its initial appearance, Whetstone has been translated to FORTRAN, PL/I, C, and other languages. The benchmark was described in the paper "A Synthetic Benchmark" written by Curnow and Wichmann in 1976. A copy of the paper can be found at [Curnow76].

3.2 Dhrystone

Like Whetstone, Dhrystone is a synthetic benchmark. It was developed in 1984 by Reinhold Weiker. Dhrystone consists of an instruction mix that was collected from a wide range of programs written in FORTRAN, PL/1, SAL, ALGOL 68, and Pascal [wiki-Dhrystone]. Unlike Whetstone, Dhrystone does not contain any FP operations and is therefore considered to be integer benchmark. The reason why there are no FP operations is because the instruction mix was taken from common but non-scientific programs that did not need such operations. It reflects the change in direction, because Whetstone was created when the computers were mostly used for scientific computing, and 10 years later, the typical use for computers shifted to end user applications. The Dhystone benchmark is available in at least three languages: C, Pascal, and Ada. Dhrystone was named as a pun of Whetstone. The results of the benchmark are reported as DIPS (Dhrystone Instructions Per Second). Dhrystone is credited to be more meaningful than MIPS since it allows comparing processors with different architectures. Another common representation of the results is DMIPS or Dhrystone MIPS. DMIPS can be obtained by dividing the reported DIPS by 1757. 1757 was the DIPS score of the VAX 11/780 which was rated as a 1 MIPS machine.

Dhrystone was considered to be representative of general purpose processors until it was superseded by SPECint benchmark. Although it is obsolete for general purpose processors, it is still a popular benchmark for embedded processors. The benchmark was described in the paper "Dhrystone: A Synthetic Systems Programming Benchmark" in 1984 by Reinhold Weiker and can be found at [Weiker84].

3.3 Fhourstone

Fhourstone [fhourstone] is an integer benchmark that solves the game Connect-4. On a contemporary computer the benchmark runs in about 10 minutes. It is available in Java and ANSI-C, and it is portable and very compact. Fhourstone is an integer benchmark and it is not synthetic. This benchmark should be only considered with caution when comparing processors, as it has not been scientifically evaluated. Fhourstone also follows the "naming scheme" of Whetstone and Dhrystone.

3.4 SPEC CPU 2006

This benchmark is published by the Standard Performance Evaluation Corporation (SPEC) [spec]. SPEC is a non-profit organization aiming at providing fair and meaningful benchmarks. It was founded in 1998 by a small number of workstation vendors. Today, it counts more than 60 member companies (most of which are leading computer and software manufactures). The SPEC CPU benchmark is updated every few years, with the current version (at the time of writing) being SPEC CPU 2006. The SPEC CPU benchmarks consist of the integer benchmark CINT and the floating point benchmark CFP. The current CINT consists of 12 benchmarks and the current CFP has 17 benchmarks. These benchmarks are selected to stress the CPU (integer and FP) and the memory subsystem. All benchmarks are written in the platform independent languages C, C++, and FORTRAN. The benchmarks are delivered as source code, and the manufacturer is allowed to use a compiler of choice but is not allowed to change the source code. Although SPEC is a non-profit organization and funded by its members, the benchmarks are not for free. A license has to be acquired for several hundred dollars.

3.5 SiSoft Sandra

Sandra [SiSoft] is a benchmarking and diagnostic software utility and its name is short for System ANalyser, Diagnostic and Reporting Assistant. The utility is only available for MS Windows systems. Sandra contains an abundance of reporting utilities and many benchmarks. The two benchmarks that are of interest to us are the CPU Arithmetic Benchmark and the CPU Multi-Media Benchmark. The latter benchmark also takes advantage of possible processor extensions such as MMX, 3DNow!, and SSE. Since Sandra is a commercial product, no further details about the benchmarks are posted.

In this section, we covered five benchmarks for general purpose computers. We started by covering two of the oldest benchmarks, Whetstone and Dhrystone. We explained the reasoning behind their design, and why it is no longer applicable. Then, we covered some newer benchmarks and pointed out the differences in comparison to the older ones.

Back to Table of Contents

---

4. Benchmarks for Supercomputers

Supercomputers differ from standard desktop computers in many ways. They usually have tens or hundreds of processors and different memory architectures. They also rely on the assumption that the workload is parallelizable so that multiple processors can work on it. A typical application for supercomputers involves the multiplication of long vectors or matrices, a task that is parallelizable. So, it comes to no surprise that most benchmarks exercise this kind of computation. In the following section, we describe the four following benchmarks for supercomputers:

- LINPACK
- LAPACK
- Lawrence Livermore Loops
- NAS Parallel Benchmarks

4.1 LINPACK

The LINPACK benchmark was developed gradually from the LINPACK software project. The software project was a collection of linear algebra routines that solve vector and matrix problems. To give the user an

idea of how long it takes to solve certain matrix problems, the appendix of the user's guide listed the times for some reference machines. The guide was published in 1979 [LINPACK-FAQ]. The LINPACK benchmark measures the floating-point rate of execution. It is determined by executing the included programs which spent most of their time in the so called Basic Linear Algebra Subroutines (BLAS). BLAS have a high percentage of floating point additions and multiplication. One of the most popular variants solves a dense n by n system of linear equations which results in 2/3*n^3+n^2+O(n) floating point operations [LINPACK-ppf]. LINPACK was originally written in FORTRAN, but is also available in C.

A report entitled "Performance of Various Computers Using Standard Linear Equations Software" by Jack Dongarra can be found at http://www.netlib.org/benchmark/performance.ps. This report describes some background and delivers nearly 90 pages of benchmark results from the beginning of the LINPACK benchmark to now. A list of the fastest 500 computers in the world based on LINPACK can be found at [Top500]. LINPACK has been largely superseded by LAPACK, which runs more efficiently on modern computer architectures.

4.2 LAPACK

LAPACK [LAPACK], short for Linear Algebra PACKage, is a collection of FORTRAN 77 routines used to measure the performance of vector processors, super-scalar processors, and shared memory multiprocessors. LAPACK was designed to replace LINPACK by restructuring the software to take advantage of new hardware features and by using some new and improved algorithms. Another difference between LAPACK and LINPACK is that LINPACK uses mostly level 1 BLAS (vector-vector operations) while LAPACK uses level 3 BLAS (matrix-matrix operations). ScaLAPACK is a version of LAPACK that was designed to run efficiently on distributed systems.

4.3 Lawrence Livermore Loops

Lawrence Livermore Loops (LLL) are a collection of 24 kernels from many diverse scientific applications that are largely vectorizable. The kernels were extracted from FORTRAN applications and exercise mostly FP operations. It seems that the LLL are outdated because there is very little information about them listed online. Nevertheless we believe that it was once an important benchmark that should not be omitted.

4.4 NAS Parallel Benchmarks

The NAS Parallel Benchmark (NPB) [NPB] has been developed by the NASA Advanced Supercomputing Division to determine the performance of highly parallel supercomputers. These benchmarks consist of five parallel kernels and three simulated application benchmarks. They are designed to mimic the computation and data movement characteristics of applications that compute fluid dynamics.

There is also a grid version called GridNPB which consists of four applications designed to mimic the computation and data movement characteristics of scientific grid applications.

Back to Table of Contents

---

5. Benchmarks for Graphic Processors

Graphic processors (GPU) are specialized processors that are usually found on graphic cards. They have a high throughput for FP operations but usually do not implement more general and time intensive instructions such branches. Many GPUs also have dedicated memory attached. Since GPUs are specialized, they require

their own benchmarks to make useful comparisons between them. Most, if not all, GPU benchmarks also include the performance of the CPU and memory in the final score since those entities are partially responsible for the GPU performance. In this section, we present the following three benchmarks:

- 3Dmark06
- SPEC viewperf 9.0
- 3D games

5.1 3Dmark06

3Dmark is published by Futuremark (formerly MadOnion) which claims to be the "worldwide standard in advanced 3D game performance benchmarking" [3Dmark]. The benchmark is intended to measure the performance of and compare different visual processing units. The measurements mainly include the amount of floating point operations of the GPU as well as the support of special features such as pixel shaders and DirectX extensions, but also CPU and memory performance. The current version of 3Dmark is 3Dmark06 where 06 indicates the year. A new version of 3Dmark is usually released a few months before the start of its title year to give graphic card manufactures the opportunity report current scores for new graphic cards. New versions of 3Dmark contain newly developed features and report lower scores than the previous version. The basic versions of the benchmark can be downloaded free of charge from the companies website. A special version of the 3Dmark benchmark is also available for mobile devices.

5.2 SPEC viewperf 9.0

SPECviewperf [vp9] is an OpenGL benchmark program written in C. It was first developed by IBM and later extended by SPEC group members such a SGI, 3Dlabs, Compaq, and HP. The benchmark is platform independent, and runs under UNIX, Windows, and Linux. The benchmark receives a set of data files and command line arguments, parses them, and sets the rendering state so that OpenGL can render the data. The benchmark runs for a pre-specified amount of time or a number of frames. It then reports the number of frames per second. To keep the benchmark as close to the real world as possible, SPEC group members can create so called "viewsets". A viewset is a group of individual runs of SPECviewperf that attempt to characterize the graphics rendering portion of an application. Note that a viewset only exercises the graphics functionality of the given application and therefore reflects the performance of the graphics hardware (GPU and memory). Currently there are eight viewsets included in SPECviewperf. All viewsets consist of large datasets and challenge even high-end systems. In addition to the viewsets there exist over 100 command line arguments that can be used in combination with the viewsets.

5.3 3D games

The use of frame rates displayed in 3D computer games is a popular way among players to compare the performance of their system. Although it is adhoc and rather unscientific, it provides a quick and easy way to come up with a single number that reflects the performance of a system for this particular scenario and allows users to rate how suitable a computer is to play a particular game. Most often, the Quake or Unreal game engine is used to benchmark a system. This is likely the result of being popular games and having high quality engines that are used in other games as well. At this point, it should be noted that the frame rate does not only reflect the speed and capabilities of the GPU but also the speed of the CPU, main memory, graphics card memory, and, even to some extent, the motherboard's chipset. Some web pages [tomshardware] provide extensive descriptions of how the overall performance is impacted by each component. Again, use these explanations with caution as they are not scientifically evaluated.

In this section, we presented three benchmarks for graphics cards. Two of them, 3Dmark and SPEC viewperf, are developed by professional organizations and are scientifically evaluated. The third benchmark uses 3D

games as measurement tools. Although not validated by research, it is popular among game players because it is easy to perform and accurate for this particular use.

Back to Table of Contents

---

6. Benchmarks for Network Processors

Network processors (NP) are a special class of processors. They are designed to bridge the gap between ASICs and general purpose CPUs by providing a processor that is as fast as an ASIC, yet as programmable as a CPU. NPs often include instructions that are useful for fast processing of network packets such as counting the number of 1s in a binary field, while removing others that are seldom used and expensive to implement. New benchmarks are needed because NPs have a different instruction set and are used in a different area than CPUs. In this section, we present CommBench and its successor, PacketBench.

6.1 CommBench

CommBench is a network processor benchmark that focuses on two areas to measure performance: packet header processing and data stream processing. For both areas, the benchmark provides four programs that are representative. The four programs for the packet header processing are a deficit round robin scheduling algorithm, an application that fragments IP packet headers, a routing lookup algorithm, and a monitoring program that filters TCP packet headers. The four applications for the data stream processing are an implementation of the CAST-128 block cipher algorithm, a lossy compression algorithm for image data, an implementation of a Reed-Solomon Forward Error Correction algorithm, and an implementation of the Lempel-Ziv compression algorithm. Some of the provided programs are already implemented today in routers, while the others are expected to represent future needs.

All of these programs have compute intensive inner loops and perform only little I/O. The developer of CommBench calls these inner loop computational kernels. However, one should be aware that the benchmark itself is an application benchmark.

A paper describing CommBench was published in 2000, and a copy of the paper can be found at [Wolf00].

6.2 PacketBench

In contrast to CommBench, PacketBench is a programming environment used to measure detailed statistics on the processing characteristics of networking applications. It provides a framework to develop new network applications on a PC, and then simulates them with the SimpleScalar processor simulator. The framework provides features such as reading packets from and writing packets to trace files, collecting statistics of resources used, and much more. Once an application is developed, it can be tested on different (simulated) network processors to measure its performance, but can also be used to compare different processors.

A paper describing PacketBench was published in 2003, and a copy of the paper can be found at [Ramaswamy03].

In this section, we covered two very different benchmarks for network processors. The first one was CommBench, and it uses a fixed set of applications to determine and compare the performance of network processors. The other benchmark was PacketBench, and it is a programming environment rather than a 'real' benchmark, but it is used to develop new network applications which in turn are used to compare network processors.

Back to Table of Contents

7. Benchmarks for Embedded Processors

Embedded processors are yet another kind of processor. They are typically designed for a specific task such as controlling a car's antilock brakes. Perceived by many as the poor cousin of general purpose processors, embedded processors usually run on a lower voltage and frequency, have fewer instructions, and have narrower instruction words. Recently however, embedded processors have become more and more powerful and their use in handheld computers blurs the line between themselves and general purpose processors. Nevertheless, embedded processors are different enough to justify their own benchmarks. In this section, rather than presenting a benchmark for each type of embedded processor, we present a consortium whose sole purpose is to create standard benchmarks for these processors.

Before application specific benchmarks appeared, it was common that the performance of embedded processors was measured with Dhrystone. But because embedded processors are designed for specific tasks and often with application specific instructions, it is clear that Dhrystone is not suitable for benchmarking embedded processors. As a matter of fact, it does not even make sense to compare two embedded processors designed for two different tasks as they have different requirements.

7.1 EEMBC Benchmarks

The Embedded Microprocessor Benchmark Consortium or EEMBC (pronounced embassy) was founded in 1997 by Markus Levy as a non-profit industry-standard consortium and initially had 12 members (mostly processor manufacturers). Today, it counts more than 50 member companies, ranging from processor manufactures to compiler vendors to real-time OS developers. The goal of EEMBC is to "develop meaningful performance benchmarks for the hardware and software used in embedded systems" [EEMBC].

The first certified benchmark scores were released in early 2000, and since then, many benchmarks for special areas have been developed. These areas include processors for automotive, consumer, digital entertainment, java, networking, telecommunication and many more. Each benchmark tests area specific operations (e.g. for telecommunication processors: convolution and Fast Fourier Transformation). For brevity, we do not go into details here as all algorithms/operations are listed at the EEMBC webpage [EEMBC].

The EEMBC benchmarks do not only measure the performance of the embedded processor, but also the quality of compilers. There exist two scoring methods. One is 'out-of-the-box' where testers can use a compiler of choice but are not allowed to modify the source code. The other one is called 'Full-Fury' where even hand-coding is allowed. It has been observed that different compilers can lead to a 40 percent difference in the final score. Because of that, it is common to produce a list of benchmark scores for a single embedded processor and multiple compilers.

In this section, we covered the benchmarks for embedded processors. Initially, Dhrystone was popular for benchmarking, but the growth in power and differentiation of the processor capabilities called for different benchmarks. The EEMBC was founded as a result of this, and today, it provides standardized benchmarks for embedded processors in many different areas.

Back to Table of Contents

8. Summary

In this paper, we surveyed benchmarks for five processor families: general purpose processors; graphics processors; network processors; embedded processors; and processors in supercomputers. We explained why it is difficult to truly measure the performance of processors and outlined how benchmarks developed from the beginning of benchmarking to today to reflect gained insights. We provided references to all surveyed benchmarks to allow interested readers to learn more about particular benchmarks.

Back to Table of Contents

---

9. References

Books:
[Jain02] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley- Interscience, New York, NY, April 1991.

[Hennessy03] John L. Hennessy and David A. Patterson, "Computer Architecture - A Quantitative Approach", Morgan Kaufmann Publishers, San Francisco, CA, 2003.

Papers:
[Wolf00] Tilman Wolf and Mark Franklin, "CommBench---A Telecommunications Benchmark for Network Processors", IEEE International Symposium on Performance Analysis of Systems and Software in Austin, TX, April 2000 - http://www.ecs.umass.edu/ece/wolf/nsl/software/cb/commbench.pdf

[Ramaswamy03] Ramaswamy Ramaswamy and Tilman Wolf, "PacketBench: A tool for workload characterization of network processing", in Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6), Austin, TX, Oct. 2003, pp. 42-50 - http://www.ecs.umass.edu/ece/wolf/pubs/2003/wwc2003.pdf

[Curnow76] H. J. Curnow and B. A. Wichmann, "A Synthetic Benchmark", The Computer Journal, 19(1), pp. 43-49, February 1976, http://freespace.virgin.net/roy.longbottom/whetstone.pdf

[Weiker84] R. P. Weicker, "Dhrystone: A Synthetic Systems Programming Benchmark", Communications of the ACM, vol. 27, Oct. 1984, http://portal.acm.org/citation.cfm?id=358283

URLs:
[wiki-MIPS] Wikipedia entry for MIPS: http://en.wikipedia.org/wiki/Million_instructions_per_second

[wiki-FLOPS] Wikipedia entry for FLOPS: http://en.wikipedia.org/wiki/FLOPS

[tomshardware] Webpage that lists benchmarks:
http://www.tomshardware.com/2000/07/04/3d_benchmarking_/index.html

[wiki-Whetstone] Wikipedia entry for Whetstone: http://en.wikipedia.org/wiki/Whetstone_(benchmark)

[longbottom] History of Whetstone: http://freespace.virgin.net/roy.longbottom/whetstone.htm

[wiki-Dhrystone] Wikipedia entry for Dhrystone: http://en.wikipedia.org/wiki/Dhrystone

[fhourstone] Developer webpage for Fhourstone: http://homepages.cwi.nl/~tromp/c4/fhour.html

[spec] SPEC homepage: http://www.spec.org/

[sisoft] SiSoft homepage: http://www.sisoftware.net/

[linpack] Developer webpage for LINPACK: http://www.netlib.org/LINPACK/

[linpack-FAQ] FAQ for LINPACK: http://cm.bell-labs.com/netlib/benchmark/faq-LINPACK.html

[linpack-ppf] LINPACK - Past, Present, and Future:
http://www.netlib.org/utk/people/JackDongarra/PAPERS/hpl.pdf

[top500] The 500 fastest computers: http://www.top500.org/

[lapack] Developer webpage for LAPACK: http://www.netlib.org/lapack/

[npb] Homepage of NAS Parallel Benchmark: http://www.nas.nasa.gov/Resources/Software/npb.html

[3dmark] Homepage of 3Dmark: http://www.futuremark.com/products/

[vp9] Homepage of SPEC viewperf 9.0: http://www.spec.org/gpc/opc.static/vp9info.html

[eembc] EEMBC homepage: http://www.eembc.org/

Back to Table of Contents

10.Acronymns

GPU - Graphics Processing Unit
LLL - Lawrence Livermore Loops
NPB - NAS Parallel Benchmark
EEMBC - Embedded Microprocessor Benchmark Consortium
FP - Floating Point
CPU - Central Processing Unit
MIPS - Million Instructions Per Second
FLOPS - Floating Point Operations Per Second
KWIPS - Kilo Whetstone Instructions Per Second
DIPS - Dhrystone Instructions Per Second
DMIPS - Dhrystone MIPS
SPEC - Standard Performance Evaluation Corporation

Back to Table of Contents

This report is available on-line at http://www.cse.wustl.edu/~jain/cse567-06/processor_workloads.htm
List of other reports in this series

Back to Raj Jain's home page