Cahners
MICROPROCESSOR REPORT
www.MPRonline.com
THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

EEMBC 1.0 Scores, Part 2

Analysis & Mysteries

By Markus Levy {9/25/00-02}

In the first article of this two-part series (see *MPR 8/14/00-01*), MDR opened the Pandora's box of the EEMBC benchmarks and examined the finer details of the benchmark scores published on the EEMBC Web site. We presented the processor and compiler vendors with our observations on their products (Table 1). This exploration revealed interesting architectural and compiler differences. Some of these differences were easily explained; some remain a mystery.

Thorough comprehension of the results requires analysis at four overlapping levels. The first level, which was performed by most of the vendors, is a high-level analysis at

Processor Name	Clock rate (MHz)	Compiler	Native Data Type	HW FPU	I/D Cache	I/D Cache Type	L2 Cache Clock	Bus Width	Bus Freq. (MHz)
AMD ElanSC520	133	CAD-UL I381G1X0	32	Yes	16K/16K	2-way SA	n/a	32	66
AMD K6-2	450	CAD-UL I381G1X0	32	Yes	32K/32K	2-way SA	66	32	66
IBM PowerPC 750CX	500	Wind River Diab 4.3b	32	Yes	32K/32K	8-way SA	500	64	66
IDT79RC32364	100 and 150	Algorithmics SDE4.0B	32	No	8K/2K	2-way SA	n/a	32	50
IDT79RC64575	250	Algorithmics SDE4.0B	64	Yes	32K/32K	2-way SA	n/a	64	50
Infineon TriCore TC10GP	80	Tasking V1.1r1	32	No	16K/16K	2-way SA	n/a	32	80
Mitsubishi M16C/62A	16	Mitsubishi, NC30, V3.20 R.1	16	No	n/a	n/a	n/a	16	16
Mitsubishi M16C/80	20	Mitsubishi NC308WAV2.00 R.1	16	No	n/a	n/a	n/a	16	20
Mitsubishi M32R/E	40	Wind River Diab Rel4.3 Rev f	32	No	40K	On-chip RAM	n/a	32	40
National Geode GX-1	200	Microsoft MSVC 6.0	32	Yes	16K Unified	4-way SA	n/a	64	66
NEC V832	143	Green Hills V1.8.9	32	No	4K/4K (write-through)	Direct-mapped (plus 4k data RAM)	n/a	32	47.6
NEC VR5000	250	Green Hills Multi(r)2000 V2.0	64	Yes	32K/32K	2-way SA	100	64	100
NEC VR5432	167	Apogee Software V4.1	64	Yes	32K/32K	2-way SA	100	32	100
Toshiba TMPR3927	133	Green Hills Multi2000 V2.0	32	No	8K/4K	2-way SA	n/a	32	66

Table 1. Comparison of the basic features and compilers of the processors discussed in this article. (n/a = not applicable)

the benchmark's main function and its relationship to the capabilities of the processor architecture. For example, a benchmark that includes floating-point operations will have the best results on a processor with hardware floating-point capability. However, one would expect even better results on a processor such as NEC's VR5432, which has dual pipelines that let the core execute any combination of integer/integer, integer/FP, or FP/FP instructions. To enable each pipeline to handle both integer and FP operations, NEC split up the FP operations: the fraction goes through the integer portion of the pipe, and the exponent goes through a separate 12-bit ALU. However, this level of analysis provides somewhat of a theoretical guess.

The second level of analysis requires a performance profiling, although it appears that few vendors have done this. Performance profiling would point out the "hot spots" in the benchmark, so that the most important sections of code could be examined. However, a code section may be a "hot spot" for one processor but not necessarily for another. Although this explanation is overly simplified, an example would be a comparison of the execution of a floating-point benchmark on a processor with hardware floating-point support versus a processor that performed floating-point operations in software. Another example might be related to cache size and would depend on the number of cache misses.

This discussion leads to the third, and most time-consuming, level of analysis, which requires an examination of the compiler's output and resource scheduling to determine how efficiently the processor is being used. For example, does the compiler schedule instructions to adequately take advantage of the processor's dual pipelines? Alternatively, is the processor performing a multiply and add in sequence, or is it able to use its single-cycle MAC operation? Part of a processor's efficiency depends on the type and complexity of the benchmark. A benchmark that lends itself to parallelism will potentially show good results on a superscalar processor. On the other hand, a benchmark that has many consecutive data dependencies will bring that same superscalar processor to its knees.

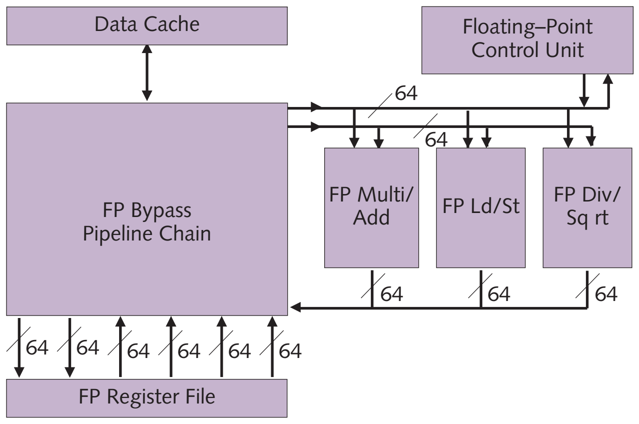

Figure 1. The floating-point unit in NEC's VR5000 demonstrates incredible floating-point performance, owing to its two- to four-cycle latency and five-stage pipeline.

Another part of understanding a processor's efficiency requires separation of the processor architecture from the compiler, the biggest mystery we have encountered. Most of EEMBC's benchmark results to date have been generated using the "out-of-the-box" method—the source code cannot be altered. This implies that a benchmark score could be far below a processor's potential and therefore, one must carefully examine the compiler's output and resource scheduling. On the other hand, until the vendors run these benchmarks using EEMBC's full-fury (hand-optimized) method, it will be difficult to determine whether the compiler is the processor's limiting factor.

Last is a system-level analysis. Things to look for include cache size and number of cache misses, cache policies (write-through or write-back), and speed and width of the memory bus. A high-level observation will reveal whether a benchmark and its associated data fit into the caches. For those benchmarks that do not fit into the caches, performance profiling will reveal the number of cache misses and the corresponding benefits or limitations of the memory subsystem.

VR5000 Floating Point Blows Doors Off Others

For most of the EEMBC automotive/industrial benchmarks, we observed that AMD's 450MHz K6-2 beats out NEC's 250MHz VR5000 by 40–90%. This is what we would expect, based on the different clock speeds balanced out with the VR5000's 64-bit architecture and 64-bit external data bus. Additionally, the VR5000's bus is running at 100MHz compared with the K6-2's at 66MHz. But for the floating-point and matrix-arithmetic benchmarks (which are floating-point intensive), the K6-2 is 35% and 59% slower, respectively. Another indicator of the VR5000's strong floating-point capability appeared when we compared this processor with IBM's 500MHz PowerPC 750CX, which executes three times faster on the benchmarks in this suite, except for the floating-point and matrix-arithmetic benchmarks, where the 750CX was only 1.66 and 2.22 times faster, respectively. A guess may lead one to credit the VR5000's performance on the floating-point capability of the MIPS IV architecture. But we also observed that for the floating-point and matrix math benchmarks, the VR5000 is about 2.5 times faster than NEC's 167MHz VR5432, which is also based on the MIPS IV architecture (the other benchmark scores were fairly close).

In short, the K6 floating-point performance lags the VR5000 because the former is a 32-bit architecture with 32-bit registers, and FP operations use two instructions to achieve the 64-bit results. In addition, the K6 executes floating-point instructions in a minimum of two processor clocks, whereas the maximum is significantly higher. IBM's 750CX contains a 64-bit FPU that can pipeline up to three instructions. Most of its floating-point instructions execute with three- or four-cycle latency and one- or two-cycle throughput. The VR5432, also with a 64-bit FPU, has many two- to six-cycle latency instructions with one- to five-cycle throughput. A multicycle FPU instruction iterates in the execute stage of the pipeline until it completes, and—since the FPU and CPU instructions share the same issue logic, data path, and pipeline stages—instruction execution stalls. On the other hand, most FPU instructions in the VR5000 have a latency of only two- to four-cycles, with a throughput of one cycle; this performance is caused by the processor's five-stage floating-point pipeline (Figure 1).

Poor Floating-Point Performance; Blame the Compiler

In Part 1 we pointed out that Infineon's TriCore superscalar architecture gave the TriCore an average 34% advantage over the 133MHz ElanSC520 for many of the automotive/industrial benchmarks. But the Elan beat the 80MHz TriCore by a factor of only two on the floating-point benchmark, although the TriCore processor performs floating-point operations in software. In general, Infineon said it expected the TriCore to outperform the x86 platform because of TriCore's tuned instruction-set architecture and its cache efficiency, and because it is a multiple-issue machine. However, we expected much better floating-point performance from Elan with hardware support versus TriCore with floating-point emulation.

Since the benchmark results are derived from EEMBC's out-of-the-box method, we shouldn't necessarily assume any architectural deficiencies related to the x86 CPU. However, we still don't know whether to blame Elan's mediocre floating-point behavior on the CAD-UL compiler or TriCore's good performance on the hand-coded Tasking library. Tasking's hand-coded library was necessary to take advantage of the pack and unpack instructions that Infineon added to the ISA to accelerate floating-point emulation. CAD-UL claims that it ran its own floating-point benchmarks with its compiler on 486 devices and found no inefficiencies in that area. It also claims that its compiler uses instructions, such as FSIN, as built-in functions, and that it was able to reach better performance than its competitors. Of course, therein lies the problem with nonstandard benchmarks.

The same 80MHz TriCore platform also beat NEC's 143MHz V832 by an average of 23%, except for the FIR filter and IDCT benchmarks, where the V832 was 50% and 12% faster, respectively. Infineon surmises that the 23% better performance is related to TriCore's multi-issue architecture—where it averages an issue rate of 1.5 instructions/clock. Another factor is that this TriCore's caches are four times the size of the V832's caches (16KB versus 4KB), and its memory bus speed has a 1:1 ratio with the core (compared with 3:1 for the V832). In addition, TriCore also has special instructions optimized for control applications; the company claims these instructions help to avoid using a longer sequence of operations to accomplish the same task.

Control applications, which are typically branch intensive, often involve frequent tests on specific bit values. These applications may also perform tests and branches on complex combinations of condition flags set in memory, or the program may branch according to the relative values of control variables. Although not unique, TriCore supports full relational "compare and branch" instructions between two arbitrary values in data registers. This feature does distinguish it from those architectures that require a separate compare instruction followed by a branch on the compare outcome. TriCore's JZ.T and JNZ.T instructions (the.T suffix standing for "bit") support direct branching on the zero/not-zero values of specific bits in a register. This helps to eliminate shifting and/or masking to isolate a bit value for branch control. Furthermore, TriCore's "accumulating logical" and "accumulating compare" instructions can reduce the number of instructions required to evaluate complex logical expressions by nearly a factor of two. These are three-operand instructions, with one of the operands serving as both the source and the destination for the accumulated logical result. There are accumulating forms for OR as well as AND, combined with all six integer relational compares or with the bit logical operations AND.T, ANDN.T, OR.T, and NOR.T.

Infineon's TriCore runs the automotive benchmarks an average of only 2.8 times faster than Mitsubishi's 40MHz M32R/E. This appears to coincide with processor frequencies, as well as with the dual-issue design of TriCore's architecture. But again, the floating-point benchmark is an exception, because TriCore runs it 20 times faster than the M32R/E. A similar situation exists between the M32R/E and NEC's V832, where the NEC processor is almost nine times faster for floating point. Again pointing to the compiler, we note that the M32R/E using Wind River's Diab compiler seems to deliver poor results; this is the difference between a hand-written floating-point library and one written in C. Mitsubishi claims that its own CC32R compiler yielded almost twice the performance for this benchmark. (However, since the CC32R-related scores are not certified, actual numbers cannot be printed.)

Mitsubishi also published certified scores that allowed MDR to compare the 16MHz M16C/62A and next-generation 20MHz M16C/80 16-bit microcontrollers. The newer microcontroller averaged 70% faster, due to an instruction set with more single-cycle execution instructions and 24-bit addressing capability to eliminate the overhead associated with using far pointers. The M16C/80 allocates one-byte instructions for frequently used 16-bit instructions. Mitsubishi also enhanced the instruction set for 32-bit operations on add, subtract, compare, move, push, and shift instructions.

Windows Handicaps National's 200MHz Geode GX-1

After certifying and publishing its scores for the consumer benchmarks, National realized that its unique benchmarking environment (in which host and target are one) penalized the Geode GX-1's performance. National tried to minimize the Windows overhead by closing all programs and running the benchmark code in DOS mode. However, National's analysis indicated that the results were different if the system booted in safe mode versus DOS mode. Further investigation revealed that the interrupts caused by peripherals (e.g., network card, mouse) and Windows system timer (which controls task switches and the software timers) ate up the benchmark performance. National also noticed that the memory timings of the PC legacy mode were suboptimal, and the integrated graphics unit shared the memory bandwidth. Experiments that are run with interrupts disabled, optimized memory timing, and an add-in graphics card indicate that out-of-the-box scores for the Geode GX1 could be improved 20–75%. Further experiments using alternate compilers proved that Microsoft's MSVC 6.0 compiler limits this processor's performance for these benchmarks.

Nevertheless, the Geode GX-1 was 76–78% faster than the V832 for the JPEG and high-pass filter benchmarks, although it was 21–45% slower for the color-conversion benchmarks. NEC credits the compiler for the V832 results and assumed that the compiler generated code that had fewer memory accesses for the color-conversion benchmarks. NEC also thinks the cache structures are involved. The write-through cache of the V832 limits its performance on the data-intensive compression and decompression routines. National also pointed out that none of the compilers it tested were able to use the Geode GX-1's MMX instructions; this ability would have allowed the processor to better handle the extensive number of multiplies integral to the consumer benchmarks.

The high-pass filter benchmark raised another interesting discussion point for NEC's VR5000 when we compared it with the VR5432. Why is the VR5000 20–55% faster for the JPEG and conversion benchmarks but 8% slower for the gray-scale filter? The answer, according to NEC, is that the VR5000's wider bus (64 bits versus 32 bits) and faster bus access (100MHZ versus 83.5MHz) are important for JPEG. On the other hand, the VR5432's integer unit, with its true dual pipeline and greater number of execution units, is 50% faster than that of the VR5000.

The Pipeline Tells the Truth

The Bezier-curve benchmark in the office automation suite also presented the VR5000 (Green Hills compiler) and VR5432 (Apogee compiler) with an interesting challenge. The processor's scores were almost equal, and again this is related to the true dual-pipeline advantage of the VR5432. NEC disabled the compiler optimization for the dual pipeline and discovered that the VR5432's score dropped by almost 40%. As with the gray-scale filter score, the VR5432 beat the VR5000 on the text processing benchmark by 10%. In an experiment with the VR5000, NEC used Apogee's compiler (instead of the one from Green Hills) for the text processing benchmark and was able to beat the VR5432.

Toshiba's 133MHz 32-bit TMPR3927F, also a MIPS-based processor, is 20–40% faster than the V832 for the office automation benchmarks, except for text processing, where it is 12% slower. According to Toshiba, the V832 and TMPR3927 have the same data cache sizes, so the text processing result is proportional to the frequency ratio. However, this is a questionable response, because the V832's cache is write-through, which certainly degrades its performance. (Another factor may be that the TMPR3927's data cache is two-way set-associative compared with the V832's direct-mapped implementation.) Toshiba also believes that the text processing benchmark extensively exercises data transfers and stack manipulations; thus, any additional cache or scratch-pad memory may help improve performance. We don't know whether the compiler can automatically take advantage of the V832's extra 4KB RAM, which could be used for the constants and global variables in the text processing benchmark (this may prevent some cache thrashing).

As for the Bezier-curve and dithering benchmarks, Toshiba has confirmed that the Green Hills compiler takes advantage of the TMPR3927's multiply-accumulate instruction. Again, the verdict is still out on whether the same is true for the V832 and its single-cycle MAC.

Benefits of the Write-Back Cache

In our Part 1 observations with the telecomm benchmarks, we compared the ElanSC520 with IDT's 100MHz 79RC-32364 and the V832. We noted that the IDT device was an average of 2.2 times faster than the Elan chip. After looking more closely at the benchmarking setup, we believe that this relationship cannot be attributed to cache sizes, bus speeds, or operating frequency. It's still a mystery whether the CAD-UL compiler (used for the AMD benchmark execution) is or isn't able to extract the full potential out of the SC520, but it's also reasonable to assume that the MIPS-based 79RC32364 is a more efficient architecture for this benchmark. And the benchmark scores improved for the 79RC32364, because IDT discovered that its original scores were derived with the processor's cache in the write-through mode. Two weeks ago, the company sent its platform to ECL (EEMBC Certification Labs) for recertification with the cache in the write-back mode, and the scores increased by up to a factor of two, depending on the benchmark (some telecomm benchmarks did not change appreciably).

We also saw the same trend for IDT with the networking benchmarks. Recertification with the cache in write-back mode demonstrated a 20–40% performance increase, depending on the benchmark. However, the route lookup benchmark scores were identical for write-through and write-back, indicating that this algorithm was more computation intensive than data intensive.

We made another observation for the VR5000 with the Green Hills compiler versus IDT's 79RC64575 with the IDT/c 7.2.1 GNU compiler. Although both processors have the same architecture, the VR5000 ran the OSPF and packet-flow benchmarks (the 512K versions) 36% and 9% faster, respectively. But, in the other three networking benchmarks, the VR5000 ran 9–18% slower.

Analyzing processors and associated compilers using EEMBC benchmark scores, or any other benchmarks, is a complicated business. This two-part series has helped to expose some intricate architectural details and has raised many questions. Equally important, we persuaded the vendors to look under the hood as they tried to understand how their architectures interacted with the benchmarks. ◇