esa estec

european space research
and technology centre

Ref: WDI/JG/2105/NL
Issue 3, 26-10-1994

Benchmarking of 32-bit processors for space applications

Prepared by J.Gaisler

On-Board Data Division
Automation and Informatics Department
ESA/ESTEC

Keplerlan 1 - Noordwijk - Netherlands
Mail address: Postbus 299 - 2200 AG - Noordwijk - The Netherlands

1 Introduction

In 1992, the European Space Agency begun the development of an 32-bit processing core for embedded space applications. The processing core, ERC32, is based on the SPARC V7 architecture and is developed by a consortium consisting of Matra MHS, Matra Marconi Space and Saab Ericsson Space. At about the same time, Saab Ericsson Space developed a second, proprietary 32-bit processor called THOR. The THOR is a stack based processor also targeted towards embedded applications, and includes hardware support for ADA tasking. Various performance claims have been made but very few actual tests has been performed until now. This evaluation has therefore been performed to measure and verify the performance of the two processors.

The evaluation was performed with the following objectives:

- To measure the computing performance using benchmarks with realistic work-loads, representative of on-board applications.
- To measure the code size of the benchmarks
- To perform the measurements using hardware and software configurations representative of on-board applications

The major conclusions of the evaluation can be summarized in the following points:

- The ERC32 delivers between two and three times higher system performance than THOR.
- The codesize of the benchmarks were the same for ERC32 and THOR.
- The usage of a optimising state-of-the-art compiler is crucial for optimum system performance.

The following paragraps describe the tested systemsresults in detail and

2 Target systems

2.1 ERC32

The tests have been run on a ERC32 simulator, based on sparcsim from Sun Microsystems. The IU and FPU has been modelled to execute the instructions in the same number of clock cycles as defined in the ERC32 data sheets [MHS 94]. The execution time of the floating-point instructions is in some cases data-dependent, and in these cases the typical values have been used. To provide a reference, the benchmarks were also run on the on a Sparcstation 1. The table below summarises the two systems:

Name	Description	CPU/FPU	CACHE	MEMORY
SS1	Sparcstation 1	L64801/W3170 @ 20 MHz	32 KByte	16 MB @ 80 ns
ERC32	ERC32 simulator	90C601E/90C602E @ 14 MHz	0	1 MB @ 40 ns

Table 1: Benchmarked SPARC systems

Two ADA compilers were available for the SPARC architecture; Telesoft ADA and Gnu ADA (Gnat). The Telesoft compiler is targeted to the Sun4 architecture and could not be used as a cross-compiler for the ERC32. The Gnat however, provides object files which can be linked with libraries for the ERC32 and can therefore be used as a cross-compiler. The Gnat supports suppression of ADA run-time checks, and the benchmarks were run with the checks both enabled and disabled. Some of the benchmarks were also available in C, and were run using GNU C-compiler (gcc) and SunOs C (cc). The following table shows the compile switches:

Compiler	Version	Switches	Comments
GNU ADA	1.81	-O2	Safe optimization, run-time checks enabled
GNU ADA	1.81	-O2 -gnatp	Safe optimization, run-time checks disabled
GNU C	2.5.8	-O2	Safe optimization
Telesoft ADA	4.1	--O prIA -i a	Safe optimization, suppress checks
SunOs C	4.1.3	-O4	Maximum optimization

Table 2: Compilers and switches - SPARC

2.2 THOR

The current version of the THOR processor is characterised up to 10 MHz operations. With 40 ns memories and ACT address and data buffers, zero-waitstate operation can be achieved at 8 MHz, operation at 10 MHz will require one waitstate. THOR includes a 256-byte data cache on-chip. The cache is not protected by parity or EDAC checksums and therefore vulnerable for SEU's. The cache may also affect the predictability of the real-time performance. The following THOR configurations have therefore been used:

Name	Description	CPU	CACHE	MEMORY
THOR08	SAAB THOR	THOR @ 8 MHz	256 Byte	0.5MB @40 ns
THOR08NC	SAAB THOR	THOR @ 8 MHz, no cache	0	0.5MB @40 ns
THOR10WS	SAAB THOR	THOR @ 10 MHz, one waitstate	256 Byte	0.5MB @40 ns
THOR10WSNC	SAAB THOR	THOR @ 10 MHz, one waitstate, no cache	0	0.5MB @40 ns

Table 3: THOR configurations

An evaluation board for THOR was available but the test were run on the THOR simulator, since some benchmarks did not run on the board. The only available ADA compiler for THOR is Oden ADA, developed by Saab Ericsson Space. There are no optional optimization switches, partial run-time checking is always generated. No C-compilers were available.

Compiler	Version	Switches	Comments
THOR Oden	0.4.1	-	Run-time checks enabled

Table 4: Oden ADA compiler

3 Benchmarks and results

The benchmarks are divided in three groups, SAAB MIX, ESTEC-B and JIAWG. Not all benchmarks were available in C, in these cases only the ADA results are given. The results are given as execution times for one iteration of the benchmark, even if more iteration have been run in order to obtain a better resolution of the execution time.

3.1 SAAB benchmarks

The six SAAB benchmarks are presented in [Saab 90] and [Gomez 91] and will therefore not be described here. The following table summarises the benchmark results:

System	RVD	PILOT	DAIS	FFT	AOCS	COMM	GEO. MEAN
Sparcstation1, Telesoft ADA	0.12	1.4	0.14	0.80	0.08	3.1	**0.41**
Sparcstation1, GNU ADA, checks disabled	0.12	0.88	0.16	0.76	0.11	2.8	**0.40**
Sparcstation1, GNU C	0.088	0.78	0.16	0.69	0.13	2.4	**0.36**
Sparcstation1, SunOs C	0.21	1.5	0.22	1.6	0.20	2.5	**0.61**
ERC32, GNU ADA, checks disabled	0.10	0.90	0.19	0.67	0.12	2.8	**0.39**
ERC32, GNU ADA, checks enabled	0.10	0.95	0.19	0.73	0.12	3.5	**0.42**
ERC32, GNU C	0.087	0.80	0.16	0.64	0.13	2.8	**0.37**
THOR @ 8 MHz, cache enabled	0.15	1.9	0.24	1.4	0.21	11.4	**0.78**
THOR @ 8 MHz, cache disabled	0.22	3.4	0.42	2.6	0.39	18.5	**1.35**
THOR @ 10 MHz, 1 ws, cache enabled	0.16	2.1	0.24	1.5	0.18	13.6	**0.81**
THOR @ 10 MHz, 1 ws, cache disabled	0.30	4.65	0.57	3.6	0.49	25.6	**1.81**

Table 5: Execution time in milliseconds - SAAB mix

With the exception of COMM, all SAAB MIX benchmarks are very floating-point intensive. This can also be seen from the fact that the execution time for ERC32 with GNU ADA is the same regardless if run-time checking is enabled or not, most of the time is simply spent in floating-point calculations. In the COMM benchmark, run-time checking increases the execution time with 25%, typically by checking array indexes. The ERC32 with run-time checking is 86% faster than the best THOR configuration (8 MHz, cache enabled). The only significant difference between THOR and ERC32 in these benchmarks can be found in COMM, which executes three times faster on ERC32.

3.2 JIAWG

A few avionics benchmarks are available from the JIAWG group. The ones selected for theses test are **Kalman**, **Cap**, **Inmult, Flmult** and **Des1**. The Kalman benchmark performs kalman filtering, Cap performs target acquisition and tracking, Inmult and Flmult perform multiplications of large matrixes and Des1 performs the DES encryption algorithm. Again, the Oden compiler produced code which failed during execution, and only results for the **Kalman**, **Inmult** and **Flmult** can be reported for THOR.

System	Kalman	Cap	Inmult	Flmult	Des1	GEO. MEAN
Sparcstation 1, Telesoft ADA	84.2	1.64	230	69.0	13.0	**48.9**
Sparcstation 1, GNU ADA, checks disabled	49.6	1.36	163	64.6	18.5	**42.1**
ERC32, GNU ADA, checks disabled	59.8	1.53	218	76.6	21.8	**50.7**
ERC32, GNU ADA, checks enabled	90.8	1.74	263	76.6	23.3	**59.4**
THOR, Oden ADA, 8 MHz, cache enabled	410	failed	486	496	failed	failed
THOR, Oden ADA, 8 MHz, cache disabled	761	failed	1,245	1,100	failed	failed
THOR, Oden ADA, 10 MHz, 1 ws, cache enabled	457	failed	522	535	failed	failed
THOR, Oden ADA, 10 MHz, 1 ws, cache disabled	1,045	failed	1,732	1,791	failed	failed

Table 6: JIAWG benchmark results in milliseconds

The results of Inmult is rather surprising; since SPARC V7 performs integer multiplication by iteration in software, low performance was expected for ERC32 in this benchmark. Nevertheless, it still outperforms THOR with a factor of two or more. In the matrix multiplications, THOR seems to be very dependent on the cache; it is a factor of three between the performance with and without cache. The Flmult result for THOR is unexpectedly low, the ERC32 runs five times faster than the fastest THOR configuration. Since the floating-point performance of THOR is assumed to be equal to SPARC, this result can come from the fact that handling large arrays yields low data cache hit rates, or slow array indexing. The performance degradation of run-time checking can also be significant, kalman filtering runs 50% faster without checks. Floating-point intensive benchmarks, such as Flmult, show little or no degradation due to run-time checking.

3.3 ESTEC-B

The Estec-B benchmark is an image compression program based on the Estec-B algorithm. It performs compression and decompression of a small image. The ADA version is less efficiently coded then the C version; this is due to the lack of bit-manipulation routines in ADA. Direct comparison of ADA and C results are therefore misleading.

System	DCT	R&Q	BIT	IBIT	IR&Q	IDCT	TOTAL
Sparcstation1, Telesoft ADA	1.30	0.24	0.69	0.84	0.27	1.38	**4.72**
Sparcstation1, GNU ADA, checks dis.	0.22	0.24	0.95	1.1	0.22	0.36	**3.0**
Sparcstation1, GNU C	0.24	0.25	0.30	0.23	0.15	0.33	**1.37**
Sparcstation1, SunOs C	0.43	0.22	0.28	0.22	0.13	0.40	**1.67**
ERC32, GNU ADA, checks disabled	0.27	0.31	1.2	1.2	0.32	0.45	**3.8**
ERC32, GNU ADA, checks enabled	0.31	0.36	1.3	1.4	0.35	0.47	**4.2**
ERC32, GNU C	0.31	0.26	0.33	0.27	0.19	0.41	**1.77**
THOR	-	-	-	-	-	-	**failed**

Table 7: Execution time in milliseconds - Estec-B (Q-step = 8)

After some modifications of the code, the Oden compiler managed to compile the benchmark; however the compiled units could not be linked due to a linker error. No result for THOR can therefore be reported.

3.4 Code size measurement

Special precautions were made when the code size was measured. To remove any size differences in the TEXT_IO library, a version without TEXT_IO was derived of each benchmark. Since different compilers provide various run-time support, a “null” program was first compiled and the size was then subtracted from the code size of the benchmarks. With this approach, only the size of the benchmark code is measured, excluding all run-time support. This is especially important for small benchmarks, such as the SAAB mix, which are often smaller than the run-time library. In a real application, the overhead of the run-time library is less noticeable.

The SPARC V7 architecture does not include a multiplication or division instruction. Instead, these routines are performed in software. The code size overhead of the multiplication and division routines can be significant in small programs, such as the SAAB MIX, but is negligible in programs with realistic size. When the code size of the SAAB MIX was measured, the individual procedures were therefore combined in one program.

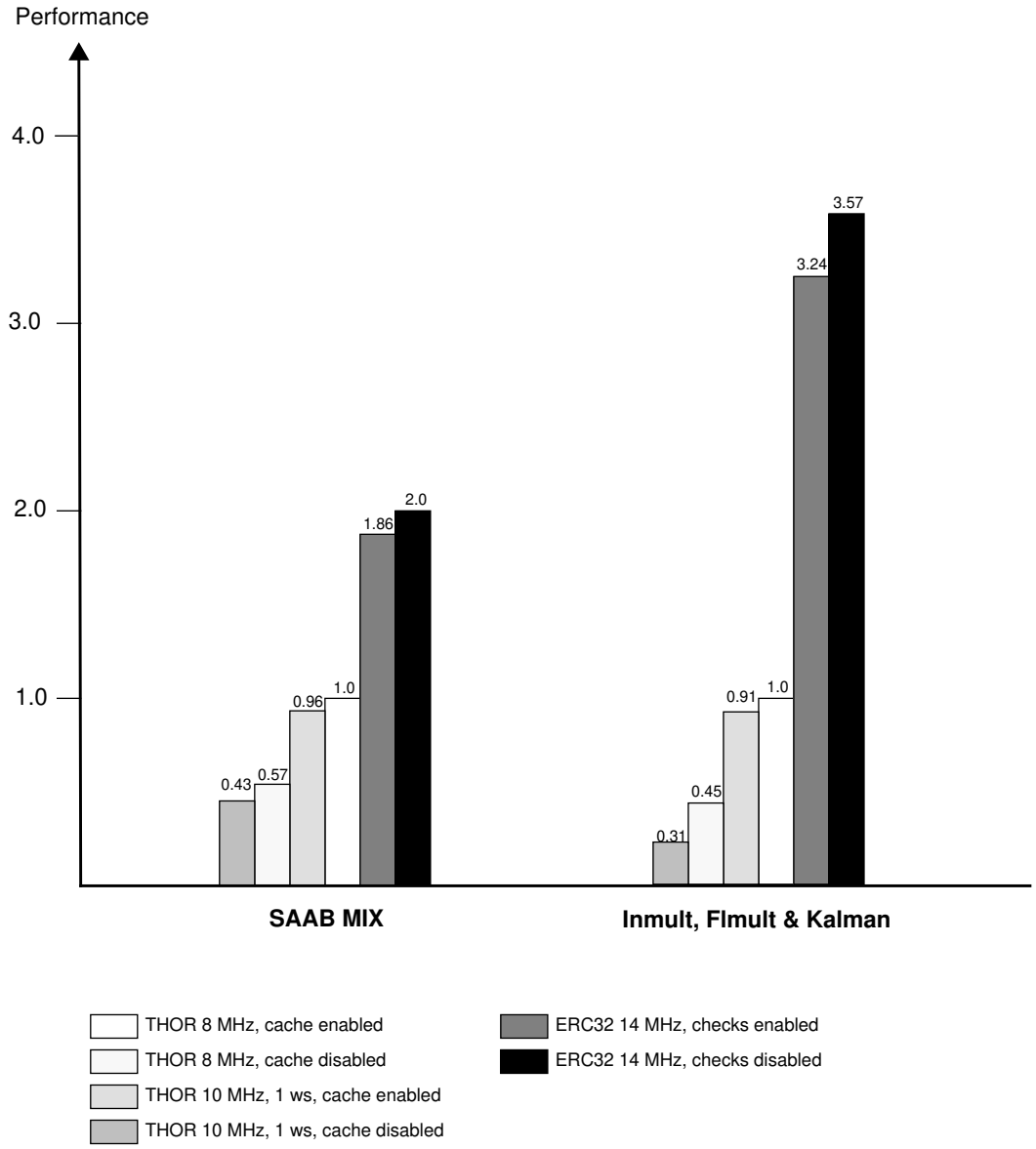
	Oden ADA (THOR)	**GNU ADA, no checks (SPARC)**	**GNU ADA, checks on (SPARC)**	**GNU C (SPARC)**
SAAB MIX	3,700	4,232	4,400	3,233
ESTEC-B	failed	9,208	10,832	7,256
Kalman	6,568	4,920	7,184	-
Cap	6,636	4,696	5,952	-
Inmult	17,448	17,592	18,200	-
Flmult	17,272	17,824	17,864	-
Des1	failed	8,672	10,088	-
Total (excluding ESTEC-B and Des1)	**51,624**	**48,264**	**53,600**	-

Table 8: Code size measurements

It can be seen from the table that GNU ADA produces approximately 4% larger code than Oden ADA. However, the routines for multiplication and division have then been counted five times in the SPARC code, since they are included in each program. Combining the various benchmarks into one program will decrease the code size for SPARC to be more or less equivalent to the code size for THOR. Note that all used benchmarks were purely sequential, i.e. no tasking was used. When tasking is used, a much larger part of the run-time kernel has to be included, and the total code size depends strongly on the implementation of the kernel.

4 Conclusions and comments

The following table summarizes the measured performance based on the SAAB MIX, Kalman, Inmult and Flmult; no other results were available for THOR. The results have been normalised relative to the best THOR value.



It is difficult to make a proper analysis of the performance of the two processors, since only a few benchmarks could be executed on THOR. Nevertheless, the performed test indicate a clear advantage for ERC32, which delivers up to 3 times higher system performance than THOR. The code size measurements indicate that the Oden and GNU ADA compilers generated approximately the same code size for a given program, run-time system excluded. It can be assumed that the

run-time system for THOR is smaller than for ERC32, since parts of the tasking mechanisms are in hardware. Whether this difference is significant compared to the size of a typical application is yet to be studied. It is also interesting to note the impact of the compiler on the overall performance; the difference between GNU C and SunOs C, or GNU ADA and Telesoft ADA was in some cases almost 100%. This shows the importance of selecting a state-of-the-art compiler; choosing a less performant compiler will require unnecessary costly and complex hardware to meet the performance requirements.

This evaluation is intended to be updated when an improved version of the Oden compiler is available. Plans are also made to include parts of the Eureka PPU software as a benchmark.

References

[Gomez 91] M. de Jong and F. Gomez-Molinero.
“Benchmarking of compilers for space embedded real-time systems”
ESA-STR-223, ESA/ESTEC 1991

[Saab 90] Saab Space AB.
“RISC Evaluation study - final report”
RIS/TRP/0010/SAAB, 1990

[MHS 90] Matra MHS SA.
“IU-RT device specification”
“FPU-RT device specification”
Issue 4, December 13, 1993