Office Automation Subcommittee — Benchmark Name: Text Processing

Highlights

- **Benchmarks Potential Performance of a Printer Interpretive Control Language.**
- **Parses Boolean Expressions Made Up Of Text Strings.**
- **Tests bit manipulation, comparison and indirect reference capabilities.**
- **Largely Shift/Rotates with Integer Math and Logical Compares/Branches**

Application

The Text Processing Benchmark is representative of a printer application where an interpretive control language is parsed. The algorithm parses boolean expressions represented as text lines made up of variables, constants and operators. The variables are space separated words, from 1 to 64 characters long, the constants are single character "T" or "F" and the operators may either be single character symbols (& | !) or their phonetic equivalents (and, or, not). Standard precedence rules for expression parsing apply.

Benchmark Description

Input to the benchmark consists of a statically declared array of variable length strings. The strings consist of variables, constants and operators separated by spaces. For example:

```
"sss and fred implies ( red & blue ) or fred"
```

The expression is broken down into a binary tree structure, with each branch on the tree being an operand (a single variable, or a constant, or a reference to yet another tree node representing another expression). Unary operators are stored as modifiers to each of the branches. The resulting structure is then traversed to evaluate the value of the expression.

The benchmark avoids calling a memory allocation routine by statically declaring and managing a 1000 node buffer.

After the timed iterations have been completed, the test is run one additional time and a CRC is calculated for the binary tree to be used for checking for correct operation.

Analysis of Computing Resources

This benchmark exercises the byte manipulation, pointer comparison, indirect reference handling and stack manipulation capabilities of a processor.

This benchmark uses an instruction mix of Compare/Branch instructions (35%), Load/Store instructions (30%), Add/Subtract instructions (20%) and Logical/Shift instructions (8%). About 30% of the memory accesses are for characters or strings. The percentages are approximate and may vary across architectures. The C library functions strcmp() and strncpy() are used extensively by this benchmark. No floating-point calculations are used. The code size and the data size are moderate.

Special Notes: The Text Processing Benchmark is part of the EEMBC OA_{mark}™ score.

Optimizations Allowed

Out of the Box / Standard C	Full Fury / Optimized				
▪ The C code must not be changed unless it must be modified to get it to compile. All changes must be documented and must not have a performance impact. ▪ Additional hardware can be used if it does not require code changes. ▪ All optimized libraries must be part of the standard compiler package, and/or available to all customers. ▪ The EEMBC Test Harness Regular or Test Harness Lite may be used. Test harness changes may be made for portability reasons if they do not impact performance.	*ASM*	*Opt. Libs*	*Inlining*	*Re-write Alg.*	*Hard-ware*
	Yes	**Yes**	**Yes**	**No**	**Yes**
	▪ The basic algorithm may not be rewritten. ▪ The code may be rewritten in assembler, as long as it doesn't change the algorithm. ▪ Optimized libraries can be used if they are publicly available. ▪ Hardware assist can be used if it is on the same processor as that being benchmarked. ▪ In-lining is allowed. ▪ Additional data files may be used by ECL during certification to ensure the correctness of the optimized benchmark.				