OABench™ Version 1.1 — Benchmark Name: Image Rotation

Highlights

- **Benchmarks Potential Performance of a Printer Application.**
- **Uses a Bitmap Rotation Algorithm to Perform a Clockwise, 90° Rotation on a Binary Image.**
- **Tests Bit Manipulation, Comparison and Indirect Reference Capabilities.**
- **Largely Logical Compares/Branches and Integer Addition/Subtraction**

Application

The Image Rotation Benchmark is representative of color and monochrome printer applications that must rotate an arbitrary binary image 90 degrees, for example, to switch between portrait and landscape modes. This benchmark uses a bitmap rotation algorithm to perform a clockwise, 90-degree rotation on a binary image. Rotated images are assumed to be a complete image (i.e. not rotating a bitmap within a larger image), with rows padded out to byte boundaries.

Benchmark Description

The bitmap rotation algorithm is primarily aimed at testing the bit manipulation, comparison and indirect reference capabilities of the microprocessor. The algorithm uses a series of indirect references and bit masks to check and set individual bits in a data buffer representing a binary image. The implementation supports 8-, 16- and 32-bit data as well as little and big Endian memory architectures. Two buffers are used, one for input and one for output, rather than trying to rotate the image in place.

There are multiple input data buffers available to debug the benchmark, but the "Medium" image must be used in the certified benchmark. This image is 295 wide and 345 bits high, or about 12K. The input buffer is included in the benchmark as statically initialized data and the output buffer is created by calling the test harness memory allocation routine, th_malloc(). After the timed iterations have been completed, the test is run one additional time so that the results can be checked by calculating a CRC check of the output buffer.

The C library routine memset() is called at the beginning of each iteration to set the output buffer to zeroes.

Analysis of Computing Resources

The benchmark effectively stresses the bit manipulation capabilities of the target CPU.

This benchmark uses an instruction mix of Compare/Branch instructions (45%), integer Add/Subtract instructions (25%) and Loads/Stores (12%). The percentages are approximate and may vary across architectures. The C library function memset() is called once per iteration to initialize the 12K output buffer to zeroes. No floating-point calculations are used. The code size is small and the data size is moderate.

Special Notes:

The Image Rotation Benchmark is part of the EEMBC OA_{mark}™ score.