OABench™ Version 1.1 Benchmark Name: Dithering

Highlights

- **Benchmarks Potential Performance of a Printer Application**
- **Uses Floyd-Steinberg Error Diffusion Dithering Algorithm (1975)**
- **Converts 8bpp Grayscale Image to Binary**
- **Largely Integer Math with Shifts and Logical Compares**

Application

The Dithering Benchmark is representative of color and monochrome printer applications. The algorithm converts a grayscale image into a form ready for printing using the Floyd-Steinberg Error Diffusion dithering algorithm. This algorithm propagates an error quantity from image row to image row, effectively diffusing errors from the rendering calculations and preventing unwanted printing artifacts, such as banding.

References: Robert Ulichney (1987); Digital Halftoning, The MIT Press, Cambridge, Massachusetts; pp. 239-242

Benchmark Description

The benchmark changes a 64K byte grayscale 8bpp image to a 8K binary image, using a Floyd-Steinberg Error Diffusion dithering algorithm. It uses two image buffers (one for the source image and a second for the generated output), and two line buffers to hold error data.

Two “error” arrays are used - one for saving the errors from the current row (used to dither the next row) and one from the previous row, used to diffuse the errors from that row to the current pixel. This array must be zeroed out first thing before the first row, to ensure that no spurious data is left there.

The error array is created such that there is one extra int at either end. This eliminates special processing at the start and end of each row (but requires zeroing the additional columns).

Each pixel of the input image file is processed as follows:

1. Calculate an “error” value using the history buffer (weighted values of
2. surrounding pixels).
3. Calculate a binary output pixel value and store.
4. Store "error" value to next line history buffer.

Analysis of Computing Resources

The benchmark effectively stresses four areas of the target CPU:

- Its indirect references used for managing internal buffers.
- Its manipulation of large data sets, since large images will stress the cache.
- Its ability to manipulate packed-byte quantities, which are used to hold grayscale pixel information.
- Its ability to perform four byte-wide multiply-accumulate operations per pixel.

This benchmark uses an instruction mix of integer Add/Subtract instructions (35%), Compare/Branch instructions (25%), Loads/Stores (20%), Shift/Rotate instructions (10%) and integer Multiply instructions (5%). The percentages are approximate and may vary across architectures. The C library function memset() is called twice per iteration, once for 8196 bytes and again for 2064 bytes. No floating-point calculations are used. The code size is small and the data size is large.

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