HOW IS RASTEROP IMPLEMENTED EFFICIENTLY?

Rasterop is implemented by shifting and masking operations that use the low-level bit arithmetic operations available to the C compiler. These include the binary bit logical operations (|, &, ^), the unary bit negation operation (~), and the bit shifting operations (<< and >>). There are three basic things one must do to make an efficient and flexible rasterop function.

1. **Pack the image data.** The pixels must be bit-contiguous within words. For example, for binary images, which have 1 bit/pixel (1 *bpp*), 32 pixels are put in each 32-bit word.

2. **Access the data by word.** The word today is typically 32 bits. Using word access allows the maximum number of pixels to be affected by each machine operation. If and when 64-bit registers become the standard ``word'' size, the routines should be altered to handle 8 bytes at a time.

3. **Order the image data.** The pixels, ordered from left to right, must be placed in bytes with the *most significant byte* (MSB) in each word to the left. This is required so that pixels within each word shift properly across byte boundaries. For big-endian machines (e.g., Sun) the byte order from left to right is 0123; for little-endian machines (e.g., Intel) the byte order is 3210. The CPUs are internally wired so that 32 bit words shift properly from MSB <--> LSB with the << and >> bit shift operators.

Using 32-bit operations, the speed of a general rasterop is approximately 2 binary pixels/machine cycle. With a 1 GHz processor, you can expect to operate on 2×10^9 destination pixels/second!

Source: http://www.leptonica.com/rasterops.html