Macro Expansions

To ease the burden of writing repetitive code prior to high-level programming languages, many assemblers provided sophisticated **marco expansion** facilities. Consider the task of loading an element of a two-dimensional array from memory into a register. This operation can easily require half a dozen instructions, with details depending on the hardware instruction set; the size of the array elements; and whether the indices are constants, values in memory, or values in registers. In many early assemblers, one could define a macro that would that would replace an expression like `ld2d(target_reg, array_name, row, column, row_size, element_size)` with the appropriate multi-instruction sequence. In a numeric program containing hundreds or thousands of array access operations, this macro could prove extremely useful.

Macros like `#define LINE_LEN 80` avoided the need to support named constants in C. More importantly parameterized macros are more efficent in equivalent C functions. They avoid the overhead of the subroutine call mechanism (including register saves and restores), and the code they generated could be integrated into any code improvements that the compiler was able to effect in the code surrounding the call.