The C4 compiler is a small, C compiler that compiles source code into a bytecode instruction set executed by an embedded virtual machine (VM). The compiler is intended to demonstrate the main phases of compilation, lexical analysis, parsing, code generation, and execution.

In lexical analysis, the compiler scans the source code and breaks it into tokens such as keywords, operators, and identifiers. It ignores whitespace characters, newlines, and comments, and the compiler utilizes functions (such as next()) to classify each symbol into a category accordingly. Tokenization allows parsing afterward to be feasible in a more organized manner.

Instead of constructing a full Abstract Syntax Tree (AST) as with traditional compilers, however, C4 uses recursive descent parsing and emits bytecode directly. Arithmetic and logical expressions are handled by the expr() routine, taking care of precedence rules, and if, while, return, and brace-delimited code blocks are handled by stmt(). Code generation is focused on directly, which simplifies compiler design.

Once the bytecode is generated, it runs on a stack-based VM with a program counter, base pointer, stack pointer, and an accumulator to manage the flow of execution. Calls, locals, and parameters are placed on the stack, and globals in a separate data segment. The VM reads, decodes, and executes the instructions in a loop with code-data separation always present.

C4 employs stack allocation for local variables and function calls and heap allocation for dynamic data in memory management. Heap memory is managed by functions like malloc and free (or equivalents). In adopting both stack and heap use, C4 benefits from a balanced approach that is simple to implement and efficient.

In total, the C4 compiler is a simple yet instructive example of how a C-like language can be implemented. Compiling directly to bytecode and running on a custom VM, it shows how fundamental compiler pieces like lexical analysis, parsing, code generation, and memory management can be simple but effective.