Technical Report - Bytecode VM

Architecture of the VM

The virtual machine follows a stack-based execution model. All computations are performed using an operand stack of 32-bit signed integers. A separate return stack is used to manage function calls and returns. The program counter (PC) tracks the current bytecode instruction. A global memory array provides persistent storage accessible via LOAD and STORE instructions.

Instruction Dispatch Strategy

The VM uses a fetch–decode–execute loop. At each step, one opcode byte is fetched and dispatched using a switch-case construct.

Instructions may be fixed-length or variable-length depending on whether they include operands. The PC is updated accordingly to ensure correct sequential execution.

Call Frames and Return Mechanism

Function calls are implemented using CALL and RET instructions. CALL saves the return address on a dedicated return stack before jumping to the function entry point. RET restores the saved address.

This design supports nested calls but does not implement local variables or activation records.

Limitations and Possible Enhancements

Possible future enhancements include support for local stack frames, function arguments, debugging and tracing facilities, and performance optimizations such

- as instruction caching or just-in-time compilation.