End-to-End EDM Framework

Technical Report

Lab 6 Submission

1. High-Level System Architecture

cite: 143

The EDM Framework is a layered execution system designed to manage the full lifecycle of a user program. It consists of four tightly coupled components:

- **Shell (Orchestrator)**
  Acts as the entry point, managing the Process Control Block (PCB) for each program. Handles command dispatching and state transitions: SUBMITTED → READY → RUNNING / PAUSED → TERMINATED.

- **Compiler (Parser & IR Gen)**
  Converts source code into an AST using Flex/Bison, then lowers it into a linear IR. Attaches debug metadata (source line numbers) to every IR instruction.

- **Virtual Machine (Execution Engine)**
  A stack-based VM that executes IR instructions. Supports a "Debug Mode" hook for single-stepping and breakpoint triggering before every instruction execution.

- **Memory Manager (GC)**
  A Mark-and-Sweep Garbage Collector integrated into the VM. Tracks all heap allocations; triggered manually via the shell or automatically during execution.

2. Component Interfaces and Data Flow

cite: 144

- **Shell → Compiler**
  The shell invokes compile_program(PID), which populates the Program struct with an IR object.

- **Compiler → VM**
  The Compiler resolves symbolic labels (Jumps) into concrete instruction indices via a Linker pass (ir_resolve_labels) before the VM receives the bytecode.

- **Parser → Debugger**
  Line numbers captured during parsing (node_with_line) are embedded in IRInstr structs. The Debugger maps user commands (e.g. break 3) to specific IR indices via this metadata.

3. Key Design Decisions & Trade-offs

cite: 145

- **Persistent VM State**
  Instead of creating a new VM for every run command, the Program struct maintains a persistent VM* pointer. This allows post-execution inspection (memstat) and state preservation for debug.

- **Global Root Tracking**
  The GC initially only scanned the stack. Extended to strictly track the Global Variable Table (vars[]) as part of the root set, ensuring globals persist across GC cycles while temporaries are reclaimed.

- **Label Resolution**

  To decouple IR generation from VM execution a late-binding resolution step was implemented. The IR generator emits Label IDs; a linear pass converts these to Program Counter (PC) targets.

4. Limitations and Known Issues

cite: 145

- **Scope**

  The current implementation supports global variables only.
  Function calls and local scoping are not fully implemented in the VM.

- **Types**

  The system is dynamically typed but primarily supports Integers.
  Type safety is enforced at runtime during operations.