# Lab 5: Mark–Sweep Garbage Collector

Performance and Evaluation Report

January 16, 2026

## 1 Introduction

Garbage collection is a fundamental memory management technique used to automatically reclaim heap memory that is no longer reachable during program execution. In this laboratory assignment, a **Mark–Sweep Garbage Collector** was implemented for a stack-based virtual machine (VM).

The objectives of this experiment are:

- To verify correct identification and collection of unreachable objects
- To ensure proper handling of transitive and cyclic references
- To evaluate performance under deep object graphs and high allocation stress
- To validate correctness during real program execution

This report presents performance results and analysis based on multiple correctness and stress tests.

## 2 Experimental Setup

- **Virtual Machine:** Stack-based VM implemented in C
- **Garbage Collection Algorithm:** Stop-the-world Mark–Sweep
- **Heap Model:** Linked list of heap-allocated objects
- **Roots:** VM stack, global memory, and closure environments
- **Metrics Collected:**
  - Execution time (milliseconds)
  - Number of heap objects freed

## 3 Performance Results

Table 1 summarizes the execution time and number of freed objects for each test case.

Table 1: Garbage Collection Performance Results

| Test ID | Test Name | Time (ms) | Objects Freed |
|---|---|---|---|
| 1 | Basic Reachability & Unreachable Collection | 0.122 | 1 |
| 2 | Transitive Reachability | 0.147 | 2 |
| 3 | Cyclic References | 0.182 | 2 |
| 4 | Deep Object Graph (Stress Test) | 15.544 | 501 |
| 5 | Closure Capture | 0.290 | 5 |
| 6 | Stress Allocation | 15.645 | 1000 |
| 7 | arithmetic.asm | 0.197 | 14 |
| 8 | factorial.asm | 0.310 | 24 |

# 4 Test Case Analysis

## 4.1 Basic Reachability & Unreachable Object Collection

This test verifies that objects which are no longer referenced by the VM stack are correctly reclaimed. The garbage collector successfully freed one unreachable object with negligible execution time, confirming correct root scanning and sweeping.

## 4.2 Transitive Reachability

This test ensures that objects reachable indirectly through other heap objects are preserved. The mark phase correctly traversed the object graph transitively, freeing only unreachable objects.

## 4.3 Cyclic References

Cyclic references pose a challenge for reference-counting collectors. In this test, the Mark–Sweep GC correctly handled cycles by using a marking flag, preventing infinite traversal and reclaiming unreachable cyclic structures.

## 4.4 Deep Object Graph Stress Test

This test evaluates the collector under a deeply nested object structure. A total of 501 unreachable objects were freed. Although execution time increased, the collector completed successfully without stack overflow or crashes.

## 4.5 Closure Capture

Closures capture variables from their surrounding environment. This test confirms that objects referenced by closures remain reachable while unused captured objects are correctly collected.

## 4.6 Stress Allocation Test

This test involves rapid allocation of a large number of objects. The garbage collector reclaimed 1000 unreachable objects, demonstrating linear sweep behavior and stable performance under heavy load.

## 4.7 Real Program Execution

Two real VM programs were executed:

- `arithmetic.asm`

- `factorial.asm`

The garbage collector operated correctly during execution, reclaiming unused objects without affecting program correctness.

# 5 Conclusion

The Mark–Sweep Garbage Collector implemented in this laboratory satisfies all functional and performance requirements. The evaluation demonstrates:

- Correct identification and reclamation of unreachable objects
- Safe handling of transitive and cyclic references
- Stable performance under deep object graphs and high allocation stress
- Seamless integration with the VM execution engine

Overall, the garbage collector is reliable, memory-safe, and suitable for managing dynamic memory in the virtual machine.