Lab Six

Emily Doran

Emily.Doran1@Marist.edu

April 13, 2021

CRAFTING A COMPILER

8.1 (BINARY SEARCH TREES AND HASH TABLES)

The two data structures most commonly used to implement symbol tables in production compilers are binary search trees and hash tables. What are the advantages and disadvantages of using each of these data structures for symbol tables?

An advantage of using a binary search tree for a symbol tree is it's simple, widely known implementation. They have a fairly average-case performance, which makes them very popular for implementing symbol tables. Also, each node in the BST is actually a stack of currently active scopes that declare the name. A disadvantage of using BSTs for symbol tables is that their average-case performance doesn't always hold for symbol tables. Since the identifier names are not chosen at random, the name lookup process could take O(n) time.

Hash tables are the most common mechanism for managing symbols because of their excellent performance. Even with a sufficiently large table, with a good hash function, and appropriate collision-handling, insertion and retrieval can be performed in constant time, regardless of the number of entries in the table. Although hash tables allow for quick searches, a disadvantage is that these can be complicated to implement.