ManiT: A PLT Adventure

ManiT Team

Akiva Dollin - Manager
Irwin Li - Language Guru
Seungmin Lee - Software Architect
Dong Hyeon (Paul) Seo - Tester

Overview

- ManiT is a multipurpose language that compiles into LLVM. The language itself is based on a series of statements which are executed sequentially.
  - NO NEED TO BOTHER WITH MAIN FUNCTIONS OR HEADER SYNTAX
- ManiT implements partial type inference and allows manipulation of structs and arrays.
- ManiT has a semi-robust standard library for file manipulation and system calls.

Dev Environment and Tools

- Git and Github
    - https://github.com/akdollin/ManiT
- Vim and Sublime
- VirtualBox and Ubuntu

Syntax: Functions, Loops, Partial Infer, Printing

```
def int foo(int a) {
    test = “Hello World”;
    counter = 0;
    for( counter = 0; counter < 3; counter = counter + 1) {
        print(counter);
    }
}
```

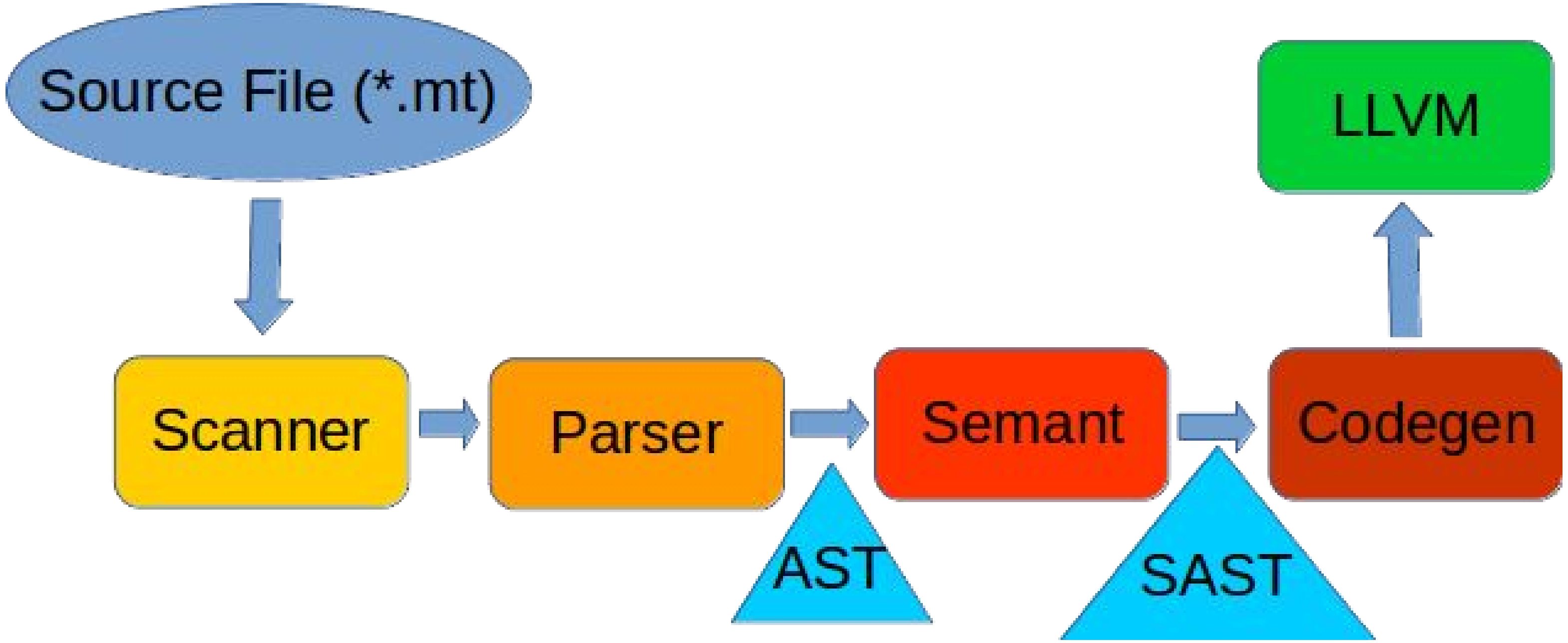
Syntax: Structs, Arrays

```
struct test {
        Int a; string b;
};
 struct test structTemp;
 structTemp.a = 4;
A = [1,2,3];
 A[0] = 2;
```

Syntax: Standard Library

- Files: Write, Open, Close, Put
- System Calls: fork(), execlp(), sleep()
- Helper: len()

Architecture



Compiler Overview

1. Manit.ml: Entry point for ManiT files.
2. Scanner.mll: Reads Characters and tokens.
3. Parser.mly: Creates AST from tokens.
4. Ast.ml: The AST tree.
5. Semant.ml: Checks AST and creates SAST.
6. Sast.ml: The SAST tree.
7. Codegen.ml: Converts SAST into LLVM code.
8. Execeptions.ml: A few error messages. Not a lot. Non-generic flags.
9. /Tests/: All the tests for ManiT.
   a. Testing covers all expressions, statements, and types defined by ManiT
   b. Total tests: ~80 tests

A Word On The Demo

Our demonstration illustrates what we think are the most important functionalities of ManiT.

1. Partial Type Inference
2. Structs/Struct Access
3. Fork/Exec/Files/Sleep
4. Loops/Functions

Demo, Let’s GOOO