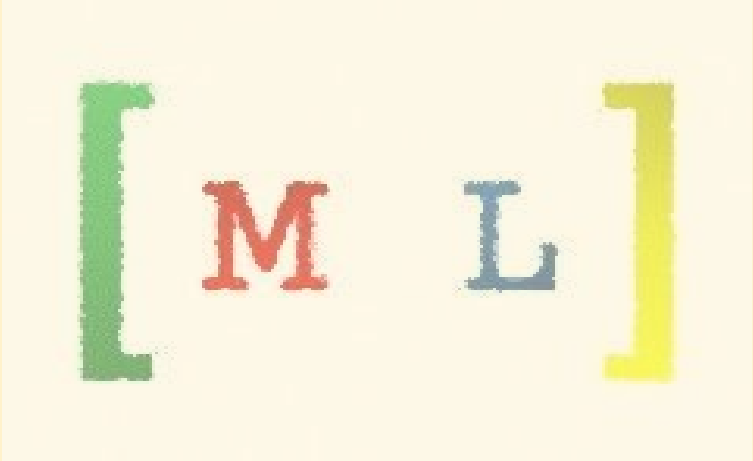
Matrix Language



Alex Barkume (ajb2233) • Jared Greene (jmg2227) • Kyle Jackson (kdj2109) • Caroline Trimble (cdt2132) • Jessica Valarezo (jgv2108)

Motivation

- Write an imperative language specifically designed for matrix manipulation
- Simplicity of treating an image as a large matrix
- Matrix datatype for matrix functions and manipulation, as well as image processing
- C-like syntax and structure of ML gives users freedom and control
- Large standard library provides both image processing and matrix functions

Overview

- C-like syntax
- Main Datatype: Matrix
- Easily import picture (PPM) - automatically becomes matrix declaration
- Print functions to output a PPM
- Large matrix-oriented standard library
- Compiles to the Low Level Virtual Machine (LLVM)

Basics

Primitive Types:
Int, Float, Bool, Char

Datatypes:
Tuple, Matrix

Declaration/Initialization:

```
int a;
a = 3;

float[2:2] b;
b = [|2.4, 5.3| 8.2, 100.2|];
```

Function Declaration:

```
void addIntTuples(int[] x, int[] y, int len)
{
    //function
}
```

```
#include <stdlib.mxl>;

m1 = open("pic.ppm");
//declares and initializes
//matrix of pic.ppm
```

File Extension: .mxl

Basics (Continued)

Operators:

Standard C arithmetic and equality operators --> primitive types

[] Tuple access
[:] Matrix access

@, @@, @@@ Access pointer to first element

$ Dereference pointer

.+ Increment pointer

Control Flow:

```
if(x > 0) {
    print(x);}
else{
    print(y);}

while(y == false){
    prints("hi");}

for(i=0; i < len ; i = i+1 ){
    x = tup[i];
    print(x);
}
```

(Very) Brief Tutorial

```
1 #include <stdlib.mxl>;
2 int main() {
3         int[3] a;
4         int[3] b;
5         int i;
6
7         a = [1, 2, 3];
8         b = [1, 2, 3];
9
10    addIntTuples(@a, @b, a.length); // Add b to a -> change a
11
12    printIntTuple(@a, a.length);
13
14        return 0; }
```

main function

tuple declaration

tuple initialization

function call

pointer

function arguments

return statement

(Very) Brief Tutorial

function name

expecting a pointer

parameters

```
1 void addIntTuples(int[] x, int[] y, int len) {
2          int i;
3
4          for (i = 0; i < len; i = i + 1) {
5                    $x = $x + $y;
6                    x = x.+;
7                    y = y.+;
8          }
9}
```

dereference

pointer increment

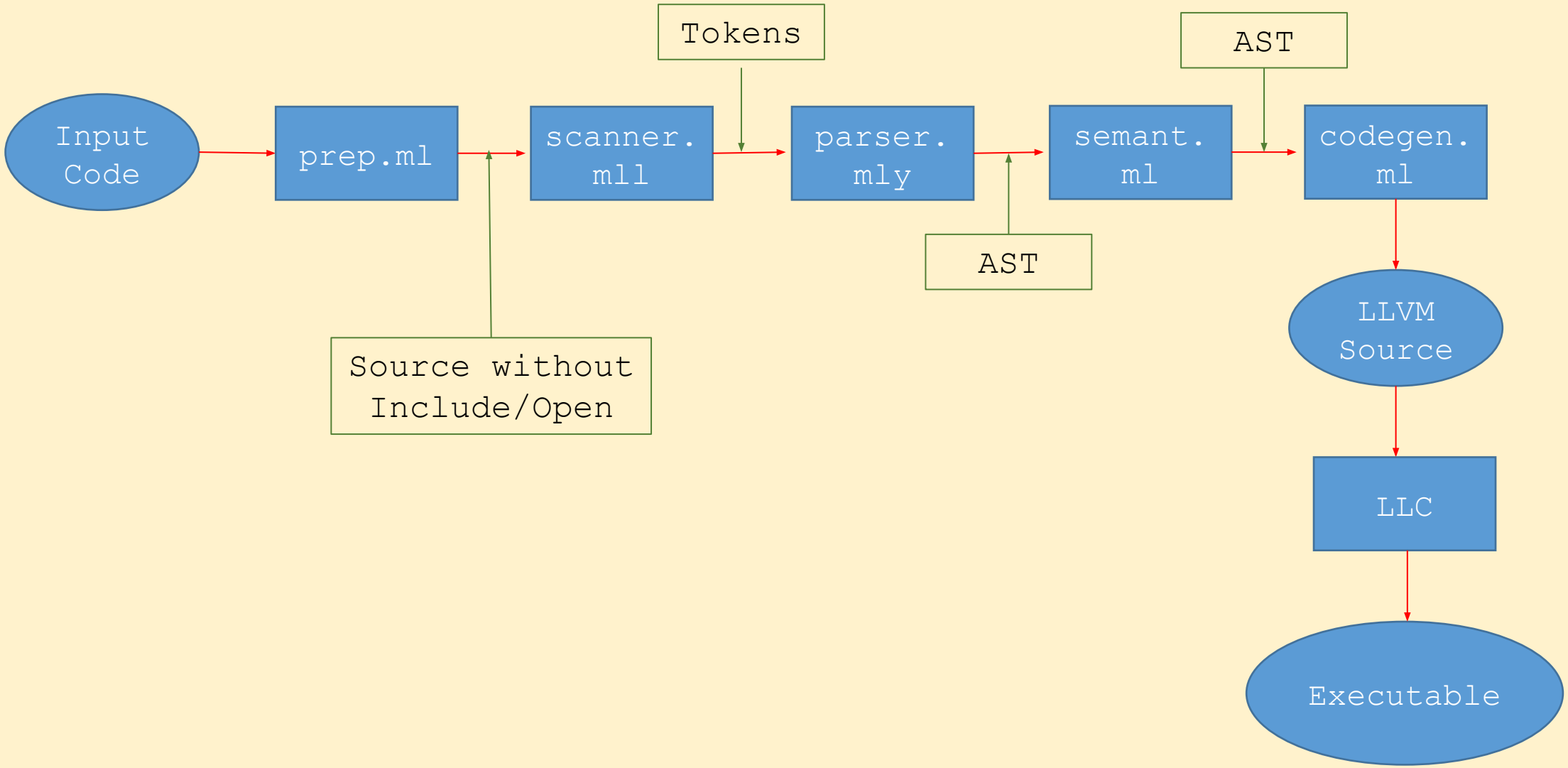
Hello, World!

```
1  int main() {
2      prints("Hello, World!");
3
4      return 0;
5  }
```

Generated Code

```
1 ; ModuleID = 'ML'
2
3  @fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
4  @fmt1 = private unnamed_addr constant [4 x i8] c"%f\0A\00"
5  @fmt2 = private unnamed_addr constant [3 x i8] c"%d\00"
6  @fmt3 = private unnamed_addr constant [3 x i8] c"%f\00"
7  @fmt4 = private unnamed_addr constant [4 x i8] c"%c\0A\00"
8  @fmt5 = private unnamed_addr constant [3 x i8] c"%c\00"
9  @.str = private unnamed_addr constant [15 x i8] c"Hello, World!\0A\00"
10
11 declare i32 @printf(i8*, ...)
12
13 define i32 @main() {
14 entry:
15   %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([15 x i8]
     * @.str, i32 0, i32 0))
16   ret i32 0
17 }
```

Structure of the Compiler



Structure of a Program

```
type program = var_dec list * func_decl list

type func_decl =
{ datatype : datatype;
  fname : string;
  formals : var_dec list;
  locals : var_dec list;
  body : stmt list;
}
```

Testing

- Integration Testing
- Automation of Test Suite
  - called by ./testall.sh
- Tests on all types, operators, element access and re-assignment, exceptions
- Each function added to stdlib.mxl has a test
- As of commit 6cc8fe8, 105 tests total

Demo