

Democritus Language Final Report

Amy Xu
xx2152
Project Manager

Emily Pakulski
enp2111
Language Guru

Amarto Rajaram
aar2160
System Architect

Kyle Lee
kpl2111
Tester

May 11, 2016

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Product Goals	2
	Native Concurrency and Atomicity	2
	Portability	2
	Flexibility	2
2	Language Tutorial	3
2.1	Setup and Installation	3
2.2	Compiling Your Code	3
2.3	Writing Code	4
2.4	Getting Started	4
	Declarations	4
	Types	4
	Primitives	4
	Strings	4
	Structs	5
	Operators	5
	Binary Operators:	5
	Unary Operators:	5
2.5	Pointers	6
2.6	Control Flow	6
	Conditional Branching	6
	Loops and Iteration	6
2.7	Multithreading and Atomicity	7
2.8	Miscellaneous	7
	Malloc	7
	File I/O	8
	Sockets API	8
3	Language Reference Manual	9
3.1	Introduction	9
3.2	Structure of a Democritus Program	9
3.3	Data types	11
	Primitive Types	11
	int	11
	float	11
	boolean	11
	pointer	11
	Complex Types	11

	string	11
	struct	11
3.4	Lexical Conventions	12
	Identifiers	12
	Reserved Keywords	12
	Literals	12
	Integer Literals	12
	Boolean Literals	12
	String Literals	12
	All Democritus Tokens	13
	Punctuation	14
	Semicolon	14
	Curly Brackets	14
	Parentheses	14
	Comments	14
3.5	Variable Declarations	14
	Variable Declaration	14
	Struct Declaration	15
3.6	Expressions and Operators	15
	Expressions	15
	Literal	15
	Identifier	15
	Binary Operation	15
	Unary Operation	15
	Struct Access	15
	Struct Assignment	15
	Function Call	15
	Variable Assignment	16
	Parenthisization	16
	Binary and Unary Operations	16
	Arithmetic Operations	17
	Addition and Subtraction	17
	Multiplication and Division	17
	Modulo	17
	Boolean Expressions	17
	Equality	17
	Negation	17
	Comparison	17
	Chained Expressions	17
	Parentheses	17
	Function Calls	18
	Pointers and References	18
	Operator Precedence and Associativity	18
3.7	Statements	18
	Expressions	19
	Return Statements	19
	Nested Blocks	19
	Conditional Statements	19
	Conditional Loops	19
3.8	Functions	20
	Overview and Grammar	20

	Calling and Recursion	20
3.9	Concurrency	21
	Overview	21
	Spawning Threads	21
4	Project Plan	22
4.1	Planning	22
4.2	Workflow	22
4.3	Team Member Responsibilities	22
4.4	Git Logs	23
	master branch	23
	threads-2 branch	33
	nested-structs branch	38
	linkedList-and-stack branch	44
	fix-malloc branch	55
	build-malloc-attempt branch	63
	Final Report	73
5	Architecture Overview	75
5.1	Compiler Overview	75
	The Scanner	75
	The Parser	75
	The Semantic Analyzer	75
	The Code Generator	75
6	Testing	76
6.1	Integration Testing	76
	Development and Testing Process	76
	Aside: Unit Testing	77
6.2	The Test Suite and Automated Regression Testing	77
7	Lessons Learned	78
7.1	Amy	78
7.2	Emily	78
7.3	Amarto	78
7.4	Kyle	78
8	Code Listing	79
8.1	democritus.ml	79
8.2	scanner.mll	79
8.3	parser.mly	81
8.4	semant.ml	83
8.5	ast.ml	90
8.6	codegen.ml	92
8.7	bindings.c	102

1. Introduction

Democritus is a programming language with a static type system and native support for concurrent programming via its `atomic` keyword, with facilities for both imperative and functional programming. Democritus is compiled to the LLVM (Low Level Virtual Machine) intermediate form, which can then be optimized to machine-specific assembly code. Democritus' syntax draws inspiration from contemporary languages, aspiring to emulate Go and Python in terms of focusing on use cases familiar to the modern software engineer, emphasizing readability, and having “one – and preferably only one – obvious way to do it”¹.

1.1 Motivation

The main motivation behind Democritus was to create a lower level imperative language that supported concurrency ‘out-of-the-box’. The race condition arising from threading would be solved by the language’s `atomic` keyword, which provides a native locking system for variables and data. Users would then be able to write and run simple multi-threaded applications relatively quickly.

1.2 Product Goals

Native Concurrency and Atomicity

Users should be able to easily and quickly thread their program with minimal worry about race conditions. Their development process should not be hindered by the use of multithreading, nor should they have to define special threading classes as is common in some other languages.

Portability

Developed under the LLVM IR, code written in Democritus can be compiled and run on any machine that LLVM can run on. As an industrial-level compiler, LLVM offers robustness and portability as the compiler back-end of Democritus.

Flexibility

Though Democritus is not an object-oriented language, it seeks to grant users flexibility in functionality, supporting structures, standard primitive data types, and native string support.

¹<http://c2.com/cgi/wiki?PythonPhilosophy>

2. Language Tutorial

Democritus is a statically-typed, imperative language with standard methods for conditional blocks, iteration, variable assignment, and expression evaluation. In this chapter, we will cover environment configuration as well as utilizing both Democritus' basic and advanced features.

2.1 Setup and Installation

To set up the Democritus compiler, OCaml and LLVM must be installed. Testing and development was done in both native Ubuntu 15.04 and Ubuntu 14.04 running on a virtual machine.

```
1 sudo apt-get install m4 clang-3.7 clang clang-3.7-doc libclang-common-3.7-dev libclang-3.7-dev libclang1-3.7 libclang1-3.7-dbg libllvm-3.7-ocaml-dev libllvm3.7 libllvm3.7-dbg lldb-3.7 llvm-3.7 llvm-3.7-dev llvm-3.7-doc llvm-3.7-examples llvm-3.7-runtime clang-modernize-3.7 clang-format-3.7 python-clang-3.7 lldb-3.7-dev liblldb-3.7-dbg opam llvm-runtime
```

For Ubuntu 15.04, we need the matching LLVM 3.6 OCaml Library.

```
1 sudo apt-get install -y ocaml m4 llvm opam
2 opam init
3 opam install llvm.3.6 ocamlfind
4 eval `opam config env`
```

For Ubuntu 14.04:

```
1 sudo apt-get install m4 llvm software-properties-common
2
3 sudo add-apt-repository --yes ppa:avsm/ppa
4 sudo apt-get update -qq
5 sudo apt-get install -y opam
6 opam init
7
8 eval `opam config env`
9
10 opam install llvm.3.4 ocamlfind
```

After setting up the environment, clone the git repository into your desired installation directory:

```
1 git clone https://github.com/DemocritusLang/Democritus.git
```

2.2 Compiling Your Code

To build the compiler, cd into the Democritus repository, and run **make**.

If building fails, try running `eval 'opam config env'`, which should update your local environment use OPAM packages and compilers. It's recommended to add the above command to your shell's configuration file if you plan on developing with Democritus.

To compile code, simply run

```
1 ./Democritus < filename.dem > outfile.lll
```

To run compiled code, call `lll` on the output:

```
1 lli outfile.lll
```

2.3 Writing Code

Code can be written in any text file, but Democritus source files should have the `.dem` extension by convention. Democritus programs consist of global function, struct, and variable declarations. Only the code inside `main()` will be executed at runtime. At this time, linking is not included in the Democritus compiler; all code should be written and compiled from a single `.dem` source file.

2.4 Getting Started

Declarations

Functions are declared with the `<function func_name(a type, b type) return_type>` syntax. Variables are declared with the `<let var_name var_type;>` syntax. Statements are terminated with the semicolon `;`. Note that all variable declarations must happen before statements (including assignments) in any given function.

```
1 function triangle_area(base int, height int) int{
2     return base*height/2;
3 }
```

Types

Primitives

Primitive types in Democritus include booleans and integers. The void type is also used for functions.

Strings

Strings are built-in to Democritus. String literals are added to global static memory at runtime, and string variables point to the literals. These literals are automatically null-terminated.

```
1 function main() int{
2     let s string;
3     let foo int;
4     let bar bool;
5
6     bar = true;
7     s = "Hello, World!";
8     foo = 55;
9     bar = false;
10
11     return 0;
12 }
```

Structs

Structs are declared at the global level with the `<struct struct_type { named fields }>` syntax. Struct declarations may also be nested.

```
1 struct Person{
2     let name string;
3     let age int;
4     let info struct Info;
5 }
6
7 struct Info{
8     let education string;
9     let salary int;
10 }
11
12
13 function main() int{
14     let p struct Person;
15
16     p.name = "Joe";
17     p.age = 30;
18     p.info.education = "Bachelor's";
19     p.info.salary = 99999;
20
21     print(p.name);
22     print(" earns: ");
23     print_int(p.info.salary);
24
25     return 0;
26 }
```

Operators

Democritus includes the ‘standard’ set of operators, defined as follows:

Binary Operators:

- arithmetic: `+`, `-`, `*`, `/`, `%`
- logical: `==`, `!=`, `<`, `<=`, `>`, `>=`, `&&` (and), `||` (or)

Unary Operators:

- arithmetic: `-`
- logical: `!` (not)
- addressing: `&` (reference), `*` (reference)

Logical expressions return a boolean value.

The expressions on each side of a binary operation must be of the same type. The `&&`, `||`, and `!` operators must be called on boolean expressions.

References can only be called on addressable fields (such as variables, or struct fields). Dereferences can only be called on pointer types.

2.5 Pointers

2.6 Control Flow

As an imperative language, Democritus executes statements sequentially from the top of any given function to the bottom. Branching and iteration is done similarly to many other imperative languages.

Conditional Branching

Conditional branching is done with:

```
1 if(boolean expression)
2 {
3     /* do something here */
4 }
5
6 else
7 {
8     /* do alternative here */
9 }
```

Here is an example of conditional branching in Democritus:

```
1 struct Person{
2     let education string;
3     let name string;
4     let age int;
5     let working bool;
6 }
7
8
9 function main() int{
10     let p Struct person;
11     p.name = "Joe"
12     p.education = "Bachelor's";
13     p.age = 25;
14     p.working = false;
15
16     if(p.working){
17         print(p.name);
18         print(" works.\n");
19     }else{
20         print(p.name);
21         print(" is looking for work.\n");
22     }
23
24     return 0;
25 }
```

This program prints “Joe is looking for work.”

Loops and Iteration

Iteration can be done either via a `for` loop. A `for(e1; e2; e3)` loop may take three expressions; `e1` is called prior to entering the loop, `e2` is a boolean conditional statement for the loop, and `e3` is called after

each iteration. Both **e1** and **e3** are optional; omitting both converts the **for** loop into the conditional **while** used in other languages.

```
1 function main() int{
2     let i int;
3     for(i = 0; i<42; i=i+1){
4         i = i+1;
5     }
6
7     for(;false;){
8         // This block will never be reached
9     }
10
11    for(true){
12        print_int(i);
13    }
14    return 0;
15 }
```

This program will print 42 forever.

2.7 Multithreading and Atomicity

Democritus supports threading with the **thread()** function call, which then calls the underlying **pthread** function in C. Any defined function can be called with multiple threads. The calling syntax is as follows:

```
1 thread("functionname", <comma separated args>, #threads);
```

Multithreaded functions must take a ***void** type as input and return a **starvoid** to conform with C's calling convention. An example of a multithreaded program:

```
1 function multiprint(noop *void) *void{
2     let x starvoid;
3     print("Hello, World!\n");
4     return x;
5 }
6
7 function main() int{
8     thread("multiprint", 0, 6); /* "Hello, World!" will be printed six times. */
9     return 0;
10 }
```

2.8 Miscellaneous

Besides threading, a couple of other functions from C have been bound to Democritus.

Malloc

malloc(size) may be called, returning a pointer to a newly heap-allocated block of *size* bytes. These pointers may be bound to strings, which themselves are pointers to string literals. An example utilizing malloc will be included with file I/O.

File I/O

Files may be opened with `open()`. This call returns an integer, which may be then bound as a file descriptor. C functions such as `write()`, `read()`, or `lseek()` may then be called on the file descriptor.

- `open(filename, fd, fd2)`: opens a file. `filename` is a string referring to a file to be opened. `fd` and `fd2` are file descriptors used for `open`.
- `write(fd, text, length)`: writes to a file. `fd` refers to the file descriptor of an open file. `text` is a string representing text to be written. `length` is an integer specifying the number of bytes to be written.
- `read(fd, buf, length)`: reads from a file. `fd` refers to the file descriptor of an open file. `buf` is a pointer to malloc'd or allocated space. `length` is an integer representing amount of data to be read. Buffers should be malloc'd before reading.
- `lseek(fd, offset, whence)`: sets a file descriptors cursor position. `fd` is the file descriptor to an open file. `offset` is an integer describing how many bytes the cursor should be offset by, and `whence` is an integer describing how `offset` should be applied: as an absolute location, relative location to the current cursor, or relative location to the end of the file.

For more detailed information on these calls, run `man function.name`.

```
1 function main() int{
2
3     let fd int;
4     let malloced string;
5
6     fd = open("tests/HELLOOOOOO.txt", 66, 384);    /* Open this file */
7     write(fd, "hellooo!\n", 10);                  /* Write these 10 bytes */
8
9     malloced = malloc(10);                        /* Allocate space for the data and null terminator */
10    lseek(fd, 0, 0);                               /* Jump to the front of the file */
11    read(fd, malloced, 10);                        /* Read the data we just wrote into the buffer */
12
13    print(malloced);                               /* Prints "hellooo!\n" */
14    return 0;
15 }
```

Sockets API

Democritus provides support for networking functionality through use of the C sockets API. A bound C function, `request_from_server`, allows a user to retrieve the contents of a webpage, written to a file, as follows:

```
1 function main() int
2 {
3     let fd int; //the file descriptor
4     request_from_server("www.xkcd.com/index.html"); //write the content of the page to
        "index.html"
5     exec_prog("/bin/cat", "cat", "tests/index.html"); //cat the file to stdout, used
        for testing
6     return 0;
7 }
```

3. Language Reference Manual

3.1 Introduction

In this language reference manual, Democritus, its syntax, and underlying operating mechanisms will be documented. In the grammars shown in this reference manual, all terminals are expressed in uppercase and all nonterminals are kept lowercase. The Lexical Conventions section will detail terminals (also known as tokens).

3.2 Structure of a Democritus Program

A basic Democritus program reduces to a list of global variable, struct, and function declarations. Code ‘to be executed’ should be written in functions. These declarations are accessible and usable from any scope in a Democritus program. At runtime, the function `main()` will be executed.

The full grammar of a program is as follows:

```
program:
    decls EOF

decls:
    /* nothing */
    | decls vdecl
    | decls fdecl
    | decls sdecl

fdecl:
    FUNCTION ID LPAREN formals_opt RPAREN typ LBRACE vdecl_list stmt_list RBRACE

formals_opt:
    /* nothing */
    | formal_list

formal_list:
    ID typ
    | formal_list COMMA ID typ

typ:
    INT
    | FLOAT
    | BOOL
    | VOID
```

```

| STRTYPE
| STRUCT ID
| VOIDSTAR
| STAR %prec POINTER typ

vdecl_list:
    /* nothing */
    | vdecl_list vdecl

vdecl:
    LET ID typ SEMI

sdecl:
    STRUCT ID LBRACE vdecl_list RBRACE

stmt_list:
    /* nothing */
    | stmt_list stmt

stmt:
    expr SEMI
    | RETURN SEMI
    | RETURN expr SEMI
    | LBRACE stmt_list RBRACE
    | IF LPAREN expr RPAREN stmt %prec NOELSE
    | IF LPAREN expr RPAREN stmt ELSE stmt
    | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
    | FOR LPAREN expr RPAREN stmt

expr_opt:
    /* nothing */
    | expr

expr:
    LITERAL
    | FLOATLITERAL
    | TRUE
    | FALSE
    | ID
    | STRING
    | expr PLUS expr
    | expr MINUS expr
    | expr STAR expr
    | expr DIVIDE expr
    | expr MOD expr
    | expr EQ expr
    | expr NEQ expr
    | expr LT expr
    | expr LEQ expr
    | expr GT expr
    | expr GEQ expr

```

```

| expr AND      expr
| expr OR       expr
| expr DOT      ID
| expr DOT      ID ASSIGN expr
| MINUS expr     %prec NEG
| STAR expr     %prec Deref
| REF expr
| NOT expr
| ID ASSIGN expr
| ID LPAREN actuals_opt RPAREN
| LPAREN expr RPAREN

```

```

actuals_opt:
    /* nothing */
| actuals_list

```

```

actuals_list:
    expr
| actuals_list COMMA expr

```

3.3 Data types

Primitive Types

int

A standard 32-bit two's-complement signed integer. It can take any value in the inclusive range (-2147483648, 2147483647).

float

A 64-bit floating precision number, represented in the IEEE 754 format.

boolean

A 1-bit true or false value.

pointer

A 64-bit pointer that holds the value to a location in memory; pointers may be passed and dereferenced.

Complex Types

string

An immutable array of characters, implemented as a native data type in Democritus. String variables are 8-bit pointers to the location of the string literal in the global static memory.

struct

A struct is a simple user-defined data structure that holds various data types, such as primitives, other structs, or pointers.

3.4 Lexical Conventions

In this subsection, we will cover the standard lexical conventions for Democritus. Lexical elements are scanned as ‘tokens,’ which are then parsed into a valid Democritus program. Democritus is a free-format language, discarding all whitespace characters such as ‘ ’, `\t`, and `\n`.

Identifiers

Identifiers for Democritus will be defined as follows: any sequence of letters and numbers without whitespaces and not a keyword will be parsed as an identifier. Identifiers must start with a letter, but they may contain any lowercase or uppercase ASCII letter, numbers, and the underscore ‘_’. Identifiers are case-sensitive, so ‘var1’ and ‘Var1’ would be deemed separate and unique. Identifiers are used to identify named elements, such as variables, struct fields, and functions. Note that identifiers cannot begin with a number. The following is a regular expression for identifiers:

```
ID = "[ 'a'-'z' 'A'-'Z' ] [ 'a'-'z' 'A'-'Z' '0'-'9' '_' ] *"
```

Reserved Keywords

The following is a list of reserved Democritus keywords:

<code>if</code>	<code>else</code>	<code>for</code>	<code>return</code>	<code>int</code>
<code>bool</code>	<code>void</code>	<code>true</code>	<code>false</code>	<code>string</code>
<code>struct</code>	<code>*void</code>	<code>function</code>	<code>let</code>	

Literals

Literals are used to represent various values or constants within the language.

Integer Literals

Integer literals are simply a sequence of ASCII digits, represented in decimal.

```
INT = "[ '0'-'9' ] +"
```

Boolean Literals

Boolean literals represent the two possible values that boolean variables can take, `true` or `false`. These literals are represented in lowercase.

```
BOOLEAN = "true|false"
```

String Literals

String literals represent strings of characters, including escaped characters. String literals are automatically null-terminated. Strings are opened and closed with double quotations. A special OCaml `lexbuf` was used to parse string literals.

(Taken from <http://realworldocaml.org/v1/en/html/parsing-with-ocamllex-and-menhir.html>)

```
read_string buf = parse
```

```

| '"'          { STRING (Buffer.contents buf) }
| '\\\' '/'    { Buffer.add_char buf '/' ; read_string buf lexbuf }
| '\\\' '\\\'   { Buffer.add_char buf '\\\' ; read_string buf lexbuf }
| '\\\' 'b'     { Buffer.add_char buf 'b' ; read_string buf lexbuf }
| '\\\' 'f'     { Buffer.add_char buf '\012' ; read_string buf lexbuf }
| '\\\' 'n'     { Buffer.add_char buf 'n' ; read_string buf lexbuf }
| '\\\' 'r'     { Buffer.add_char buf 'r' ; read_string buf lexbuf }
| '\\\' 't'     { Buffer.add_char buf 't' ; read_string buf lexbuf }
| [^ '"' '\\\' ]+ { Buffer.add_string buf (Lexing.lexeme lexbuf);
                  read_string buf lexbuf
                }
}

```

All Democritus Tokens

The list of tokens used in Democritus are as follows:

```

| "/"          { comment lexbuf }
| "/*"        { multicomment lexbuf }
| '('         { LPAREN }
| ')'         { RPAREN }
| '{'         { LBRACE }
| '}'         { RBRACE }
| ';'         { SEMI }
| ','         { COMMA }
| '+'         { PLUS }
| '-'         { MINUS }
| '*'         { STAR }
| '&'         { REF }
| '.'         { DOT }
| '/'         { DIVIDE }
| '='         { ASSIGN }
| "=="        { EQ }
| "!="        { NEQ }
| '<'         { LT }
| "<="        { LEQ }
| ">"         { GT }
| ">="        { GEQ }
| "&&"        { AND }
| "||"        { OR }
| "!"         { NOT }
| "if"        { IF }
| "else"      { ELSE }
| "for"       { FOR }
| "return"    { RETURN }
| "int"       { INT }
| "bool"      { BOOL }
| "void"      { VOID }
| "true"      { TRUE }
| "string"    { STRTYPE }
| "struct"    { STRUCT }
| "*void"     { VOIDSTAR }

```



```

| "false"    { FALSE }
| "function" { FUNCTION }
| "let"      { LET }
| ['0'-'9']+ as lxm { LITERAL(int_of_string lxm) }
| ['a'-'z' 'A'-'Z'] ['a'-'z' 'A'-'Z' '0'-'9' '_' ]* as lxm { ID(lxm) }
| '"'       { read_string (Buffer.create 17) lexbuf } (* refer to read_string above *)
| eof { EOF }

```

Punctuation

Semicolon

The semicolon ‘;’ is required to terminate any statement in Democritus.

```
statement SEMI
```

Curly Brackets

In order to keep the language free-format, curly braces are used to delineate separate and nested blocks. These braces are required even for single-statement conditional and iteration loops.

```
LBRACE statements RBRACE
```

Parentheses

To assert precedence, expressions may be encapsulated within parentheses to guarantee order of operations.

```
LPAREN expression RPAREN
```

Comments

Comments may either be single-line, initialized with two backslashes, or multi-line, enclosed by `*` and `*\`.

```
COMMENT = ("// [^\n']* \n") | ("/*" [^ "*/"]* "*/")
```

3.5 Variable Declarations

In Democritus, local variables must be declared at the top of each function, before being later assigned.

Variable Declaration

Democritus requires all named variables to be declared with its type at the top of each function. Named variables are declared with the `let [ID] type` syntax. Assignment to these variables may then be done with `=`.

The grammar for variable declarations is as follows:

```

vdecl:
  LET ID typ SEMI

```

```

typ:
  INT

```

```
| BOOL
| VOID
| STRTYPE
| STRUCT ID
| VOIDSTAR
| STAR %prec POINTER typ
```

Struct Declaration

Structs are defined at the global scope, and can then be declared as variables. The global definitions are as follows:

```
sdecl:
    STRUCT ID LBRACE vdecl_list RBRACE

vdecl_list:
    | vdecl_list vdecl
```

3.6 Expressions and Operators

Expressions

Expressions may be any of the following:

Literal

A literal of any type, as detailed in the lexical conventions section.

Identifier

An identifier for a variable.

Binary Operation

A binary operation between an expression and another expression.

Unary Operation

A unary operation acting on the expression appearing on the immediate right of the operator.

Struct Access

An expression of a **struct** type accessing an identifier field with the **dot** (.) operator.

Struct Assignment

An expression of a **struct** type assigning a value to one of its fields (accessed with the **dot** (.) operator) using the = operator.

Function Call

A call to a function along with its formal arguments.

Variable Assignment

An identifier being assigned a value with the = operator.

Parenthisization

Another expression nested within parentheses.

The grammar for expressions is as follows:

```
expr:
  LITERAL
| FLOATLITERAL
| TRUE
| FALSE
| ID
| STRING
| expr PLUS      expr      (* expr TERMINAL expr are binary operations *)
| expr MINUS     expr
| expr STAR      expr
| expr DIVIDE    expr
| expr MOD       expr
| expr EQ        expr
| expr NEQ       expr
| expr LT        expr
| expr LEQ       expr
| expr GT        expr
| expr GEQ       expr
| expr AND       expr
| expr OR        expr
| expr DOT ID    (* struct access *)
| expr DOT ID ASSIGN expr (* struct assign *)
| MINUS expr %prec NEG   (* unary arith negate *)
| STAR expr %prec Deref  (* unary deref *)
| REF expr              (* unary ref *)
| NOT expr              (* unary log negate *)
| ID ASSIGN expr
| ID LPAREN actuals_opt RPAREN (* function call *)
| LPAREN expr RPAREN          (* paren'd expr *)
```

Binary and Unary Operations

A binary operation operates on the two expressions on the left and right side of the operator. Binary operations may be:

- an addition, subtraction, mult., division, or modulo on two arithmetic expressions (+, -, *, /, %). Modulo only works on integer types.
- equality or inequality expression between boolean expressions (==, !=, <, <=, >, >=, &&, ||)

A unary operation operates on the expression on the operator's right side:

- a negation of an arithmetic expression (-)

- a dereference of a pointer type (*)
- an address reference of a variable or field within a struct (&)
- a negation of a boolean expression (!)

Arithmetic Operations

Democritus supports all the arithmetic operations standard to most general-purpose languages, documented below. Automatic casting has not been included in the language, and the compiler will throw an error in the case that arithmetic operations are performed between the same types of expressions.

Addition and Subtraction

Addition works with the + character, behaving as expected. Subtraction is called with -.

Multiplication and Division

Multiplication is called with *, and division with /. Division between integers discards the fractional part of the division.

Modulo

The remainder of an integer division operation can be computed via the modulo % operator.

Boolean Expressions

Democritus features all standard logical operators, utilizing ! for negation, and && and || for **and** and **or**, respectively. Each expression will return a boolean value of true or false.

Equality

Equality is tested with the == operator. Inequality is tested with !=. Equality may be tested on both boolean and arithmetic expressions.

Negation

Negation is done with !, a unary operation for boolean expressions.

Comparison

Democritus also features the <, <=, >, and >= operators. These represent less than, less than or equal to, greater than, and greater than or equal to, respectively. These operators are called on arithmetic expressions and return a boolean value.

Chained Expressions

Boolean expressions can be chained with && and ||, representing **and** and **or**. These operators have lower precedence than any of the other boolean operators described above. The **and** operator has a higher precedence than **or**.

Parentheses

Parentheses are used to group expressions together, since they have the highest order of precedence. Using parentheses will ensure that whatever is encapsulated within will be evaluated first.

Function Calls

Function calls are treated as expressions with a type equal to their return type. As an applicative-order language, Democritus evaluates function arguments first before passing them to the function. The grammar for function calls is as follows:

```
expr:
  .
  .
  .
  | ID LPAREN actuals_opt RPAREN    (* Function call *)

actuals_opt:
  /* nothing */
  | actuals_list

actuals_list:
  expr
  | actuals_list COMMA expr
```

Pointers and References

Referencing and dereferencing operations are used to manage memory and addressing in Democritus. The unary operator `&` gives a variable or struct field's address in memory, and the operator `*` dereferences a pointer type.

Operator Precedence and Associativity

Precedence	Operator	Description	Associativity
1	()	Parenthesis	Left-to-right
2	()	Function call	Left-to-right
3	*	Dereference	Right-to-left
	&	Address-of	
	!	Negation	
	-	Unary minus	
4	*	Multiplication	Left-to-right
	/	Division	
	%	Modulo	
5	+	Addition	Left-to-right
	-	Subtraction	
6	>> =	For relational > and ≥ respectively	Left-to-right
	<<=	For relational < and ≤ respectively	
7	== !=	For relational = and ≠ respectively	Left-to-right
8	&&	Logical and	Left-to-right
9		Logical or	Left-to-right
10	=	Assignment	Right-to-left

3.7 Statements

Statements written in a Democritus program are run from the top to bottom, sequentially. Statements can reduce to the following:

Expressions

An expression statement consists of an expression followed by a semicolon. Expressions in expression statements will be evaluated, with their values calculated.

Return Statements

A return statement is either a `RETURN SEMI` or `RETURN expr SEMI`. They are used as endpoints of a function, and control from a function returns to the original caller when a return statement is executed. Returns may be empty or return a type, though non-void functions must return an expression of their type.

Nested Blocks

A nested block is another statement list encapsulated within braces `{}`.

Conditional Statements

Conditional statements follow the `IF (boolean expr) stmt1 ELSE stmt2` format. When the `expr` evaluates to true, `stmt1` is run. Otherwise, if an `ELSE` and `stmt2` have been specified, `stmt2` is run.

Conditional Loops

A conditional loop is similar to a conditional statement, except in that it will loop or run repeatedly until its given boolean expression evaluates to `false`. In the case that the expression never evaluates to `false`, an infinite loop will occur.

Democritus eliminates the `while` keyword sometimes used in conditional iteration. Conditional loops follow the `FOR (expr1;boolean expr2;expr3) stmt` format where `expr1` and `expr3` are optional expressions to be evaluated prior to entering the `for` loop and upon each loop completion, respectively. Prior to entering or re-entering the loop, `expr2` is evaluated; control only transfers to `stmt` if this evaluation returns true. If both `expr1` and `expr3` are omitted, a simpler `for` loop can be written of the form `FOR (boolean expr) stmt`.

The full grammar for statements is as follows:

```
stmt_list:
    /* nothing */
    | stmt_list stmt

stmt:
    expr SEMI
    | RETURN SEMI
    | RETURN expr SEMI
    | LBRACE stmt_list RBRACE
    | IF LPAREN expr RPAREN stmt %prec NOELSE
    | IF LPAREN expr RPAREN stmt ELSE stmt
    | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
    | FOR LPAREN expr RPAREN stmt
```

3.8 Functions

Overview and Grammar

Functions can be defined in Democritus to return one or no data type. Functions are evaluated via eager (applicative-order) evaluation and the function implementation must directly follow the function header. The grammar for function declarations is as follows:

```
fdecl:
    FUNCTION ID LPAREN formals_opt RPAREN typ LBRACE vdecl_list stmt_list RBRACE

formals_opt:
    /* nothing */
    | formal_list

formal_list:
    ID typ
    | formal_list COMMA ID typ
```

All functions require **return** statements at the end, and must return an expression of the same type as the function. **void** functions may simply terminate with an empty **return** statement.

```
1     }
2 function function_name([formal_arg type, ... ]) type_r {
3
4     [function implementation]
5     return [variable of type type_r]
6 }
```

Calling and Recursion

Functions may be recursive and call themselves:

```
1 function recursive_func(i int) void {
2
3     if (i < 0) {
4         return;
5     } else {
6         print(  h i  );
7         recursive_func(i-1);    // Call ourselves again.
8     }
9 }
```

Functions may be called within other functions:

```
1
2 function main() void{
3     recursive_func(3);
4     return;                // Return nothing for void.
5 }
```

3.9 Concurrency

Overview

Democritus intends to cater to modern software engineering use cases. Developments in the field are steering us more and more towards highly concurrent programming as the scale at which software is used trends upward.

Spawning Threads

To spawn threads, Democritus uses a wrapper around the C-language **pthread** family of functions.

The *thread_t* data type wraps *pthread_t*.

To spawn a thread, the thread function takes a variable number of arguments where the first argument is a function and the remaining optional arguments are the arguments for that function. It returns an error code.

The **detach** boolean determines whether or not the parent thread will be able to join on the thread or not.

```
1      {  
2      function thread(f function, arg *void, arg, nthreads int) *void;  
3      }
```


4. Project Plan

4.1 Planning

Much of the planning was facilitated by our weekly meetings with David Watkins, our TA. He very clearly explained what the requirements of each milestone entailed, and helped keep expectations transparent. Since Professor Edwards had emphasized the need for vertical development of features instead of horizontal building of each compiler layer, we quickly identified the key features that would be required to enable the key functionality of our language. Two of the most important components were structs and threads.

Our initial plan, which we largely followed, was to complete the Language Reference Manual, experiment with the layers of the compiler and get `Hello, World!` working, and then use what we had learned to begin implementing the more crucial aspects of the language.

4.2 Workflow

Workflow was facilitated by Git and GitHub, which allowed for the team to easily work on multiple features simultaneously and (usually) merge together features without overlapping conflicts. The Git workflow reached an optimal point by the conclusion of the project; new features would be developed, tested, and finalized in separate branches, and the commits for that feature would be squashed down until a single commit representing the new feature would be merged into the master branch.

Features were developed individually or through paired programming, depending on the scope and complexity of the feature. GitHub and division of labor allowed for many team members to work independently, at different times of the day per their own schedule. Branching allowed for one person to quickly deploy buggy code to another in hopes of resolving the issue, without any modification or bad commits to the master branch. GroupMe was used extensively for inter-team communication.

4.3 Team Member Responsibilities

Team Member	Responsibilities	GitHub Handle
Amy Xu	structs, nested structs, pointers, malloc	axxu
Emily Pakulski	C-bindings, reformatting tests, atomicity	ohEmily
Amarto Rajaram	C-bindings, pthread, file I/O, malloc	Amarto
Kyle Lee	structs, LRM, Final Report, debug and assist Amy	kyle-lee

4.4 Git Logs

Note that ‘PLT Student’ was team member Amy Xu.

master branch

```
1 commit 4ecbc629057cc19c839d5e0f5f9224b88710ac8e
2 Merge: d0a23f0 cb6b509
3 Author: Amy Xin Xu <axxu3795@gmail.com>
4 Date: Wed May 11 18:04:44 2016 -0400
5
6 Merge pull request #24 from DemocritusLang/linkedlist_and_stack
7
8 Final linked list demo
9
10 commit cb6b5091e6812c3f3a60869bdb39bdd6900c201e
11 Author: PLT Student <axxu3795@gmail.com>
12 Date: Wed May 11 17:35:32 2016 -0400
13
14 Final linked list demo
15
16 Fixed merge accident
17
18 Fixed malloc size
19
20 Added to demo folder
21
22 commit d0a23f079f4a9828c6d2724141c5b7de1a126958
23 Author: Emily Pakulski <enp2111@columbia.edu>
24 Date: Wed May 11 17:52:08 2016 -0400
25
26 Added simple threads test.
27
28 commit e113aa58cc30af12cad615d07ed57b99905e0edf
29 Author: Emily <ohEmily@users.noreply.github.com>
30 Date: Wed May 11 16:26:44 2016 -0400
31
32 Multithreading and networking working together. Added concurrent comic download.
33 (#23)
34
35 * Added multithreaded test— getting parse error. Added strcat and int to string
36 wrappers
37
38 * fixed parse error.
39
40 * Added memset to fix test—multithreaded—sockets.
41
42 * Fix memset bug
43
44 * Updated test file so sockets test passes
45
46 * Fixed thread function signature
47
48 * Added failing test.
```

```

48      * Fixed bug in init_thread() and added test that passes string into thread().
49
50      * possibly fixed request bug.
51
52      * Fixed binary file reading bugs.
53
54      * Moved code into demo directory.
55
56      * Fixed merge conflicts after rebase.
57
58  commit d4972e0ff8b5ba850bf39528cc0b113bc1912ee5
59  Merge: dbb06cc 232belf
60  Author: Amy Xin Xu <axxu3795@gmail.com>
61  Date:   Wed May 11 16:10:39 2016 -0400
62
63      Merge pull request #22 from DemocritusLang/build_malloc_attempt
64
65      Malloc and simple linked lists working
66
67  commit 232belf27fb7ffa7110e5184c6016b81e2da94ff
68  Author: PLT Student <axxu3795@gmail.com>
69  Date:   Wed May 11 03:45:50 2016 -0400
70
71      Malloc and simple linked lists working
72
73      Fixed shift reduce errors
74
75      temp commit
76
77      CASTING AND MALLOC WORK
78
79      Cleaned up warnings
80
81      Cleaned up codegen warnings
82
83      Fixed (*a). dotops and halfassed addnode
84
85      Half way linked lists
86
87      Linked lists with add and print_list functions
88
89  commit dbb06cc9f0f2f608583946158d48d8d841d8dc62
90  Author: Emily Pakulski <enp2111@columbia.edu>
91  Date:   Wed May 11 13:30:05 2016 -0400
92
93      Added check in tester for whether code was already compiled.
94
95  commit 36aee48817bdeec1553b1c16963e5abcc7aaaa6b
96  Merge: 0371f67 2402f74
97  Author: Amarto <aar2160@columbia.edu>
98  Date:   Wed May 11 06:10:49 2016 -0400
99
100      Merge pull request #21 from DemocritusLang/sockets_finished
101
102      Sockets finished
103

```

```

104 commit 2402f740bec59d6ed4912b2ee302d9b9b7bb5480
105 Author: Amarto <aar2160@columbia.edu>
106 Date: Wed May 11 05:45:05 2016 -0400
107
108     Added free(), execl wrapper, and corrected output reference for socket test.
        Changed tests to use free() after malloc. Refactored the weblink downloading
        method to only take one param so it matches the signature for a thread function
109
110 commit 170645297d4578f9551f690b08b82718beebf033
111 Author: Emily Pakulski <enp2111@columbia.edu>
112 Date: Wed May 11 01:44:23 2016 -0400
113
114     Changed up get request impl a bit.
115
116 commit 5226ad21d850b6622e09388a4337d0b515123f76
117 Author: Amarto <aar2160@columbia.edu>
118 Date: Tue May 10 17:46:11 2016 -0400
119
120     Added basic socket impl and loading files. Need to handle tests
121
122 commit 0371f67bac394c08ae72cb372491fd3264839f65
123 Merge: a6ce096 a784099
124 Author: Amy Xin Xu <axxu3795@gmail.com>
125 Date: Wed May 11 02:57:57 2016 -0400
126
127     Merge pull request #20 from DemocritusLang/add_float_and_mod
128
129     Added modulo and floats
130
131 commit a78409901632a6f70e889fee8f5ccff2bfe12989
132 Author: PLT Student <axxu3795@gmail.com>
133 Date: Tue May 10 23:26:55 2016 -0400
134
135     Added modulo and floats
136
137     Mod done
138
139     Working on floats
140
141     fixed floating print issue
142
143     Working floats
144
145     Added floats in struct test
146
147 commit a6ce096863beb92e5b5251b0802613edab31cc76
148 Author: Emily <ohEmily@users.noreply.github.com>
149 Date: Tue May 10 23:09:23 2016 -0400
150
151     Added sleep function and test. (#18)
152
153 commit ff330840be2c13aeed6f2a6d87a69ce153f29421
154 Merge: a63b40f fdcadf4
155 Author: Amy Xin Xu <axxu3795@gmail.com>
156 Date: Tue May 10 15:28:57 2016 -0400
157

```

```

158 Merge pull request #17 from DemocritusLang/add_pointers
159
160 Pointers done
161
162 commit fdcadf4fa032354c8a9ec96f41cecb76b94e66f0
163 Author: PLT Student <axxu3795@gmail.com>
164 Date: Tue May 10 02:02:43 2016 -0400
165
166 Pointers done
167
168 Dereference syntax there, need to clean warning
169
170 Added ref op and semantic checking
171
172 Working pointers for ints, need to test rest
173
174 Modified test-pointer-int.dem for clarity and wrote test-pointer-bool, passing
175
176 Added single and multilevel struct tests, passing
177
178 Linkedlist test not working
179
180 Added hypothetical linkedlist tests (not working
181
182 Linked of list proof of concept
183
184 Changed type* to *type to reflect Go syntax
185
186 commit a63b40fb618149ec65e57ea3ac986bdccf9f4ac4
187 Author: Kyle Lee <kylelee.contact@gmail.com>
188 Date: Tue May 10 01:11:44 2016 -0400
189
190 Added singleline comments
191
192 commit 901814668aa2d7513fe74b45fa4390d82635ac01
193 Author: Amarto <aar2160@columbia.edu>
194 Date: Tue May 10 00:46:58 2016 -0400
195
196 Fixed void pointer notation to match Go syntax, fixed test
197
198 commit 0ed94930f8362cb6eb322ec6a9570043660aabb5
199 Merge: b662f67 d9b467b
200 Author: Amy Xin Xu <axxu3795@gmail.com>
201 Date: Tue May 10 00:43:18 2016 -0400
202
203 Merge pull request #15 from DemocritusLang/clean.nested.structs
204
205 Working nested structs
206
207 commit d9b467b80ce152696875d6ec1d3d2f1ec6ea77e6
208 Author: PLT Student <axxu3795@gmail.com>
209 Date: Mon May 9 22:47:07 2016 -0400
210
211 Working nested structs
212
213 Added nested struct test

```

```

214
215     Fixed mistyped identifier
216
217     nested structs working
218
219     Fixed typo in test-structs-nested.out and added another test
220
221     Edited test to be more informative of functionality
222
223     test-struct-nested1
224
225 commit b662f676ae12fbb27eedaf5af6ae990d76f423bc
226 Author: Emily Pakulski <enp2111@columbia.edu>
227 Date:   Mon May 9 20:34:05 2016 -0400
228
229     Finished file I/O. lseek also implemented.
230
231 commit 84c1fc11bc2a2e59b8fec9d68937db8205f1b5d9
232 Author: Amarto <aar2160@columbia.edu>
233 Date:   Sun May 8 20:30:58 2016 -0400
234
235     Added malloc and started file I/O.
236
237 commit 8b3944051cfde07be958214aae56bf47988fb803
238 Author: Emily <ohEmily@users.noreply.github.com>
239 Date:   Mon May 9 11:15:23 2016 -0400
240
241     Updated all instances of MicroC to Democritus and added 'make all' target (#12)
242
243     * Changed MicroC -> Democritus and added make all target.
244
245     * Changed file extension for democrituslang files from .mc to .dem.
246
247 commit ed27ce5f8a31a740f3eb0e5ad3ff3cfcf7a838f9
248 Author: Amarto <aar2160@columbia.edu>
249 Date:   Sun May 8 19:31:43 2016 -0400
250
251     Fixed warnings resulting from merge
252
253 commit c6cbdf15fd8854a02fb695fa9aa41b50966431a7
254 Author: Amarto <aar2160@columbia.edu>
255 Date:   Sat Apr 30 15:16:57 2016 -0400
256
257     Added multithreading and void pointers, and added calling bound C functions
258     Added declaration of thread() function to codegen. Everything compiles
259
260     Added basic threads checking to semant.ml. Need to wait until arguments for
        pthread are passed in
261
262     Working on codegen.ml, but getting compiler warning. Working on threading test,
        but need NULL keyword?
263
264     Added tests for threading and modified codegen and semant
265
266     Baby steps. Still not working. (temp commit).
267

```

```

268     Oops. But still not working.
269
270     Fixed some things in test case. Pretty sure function name should be passed in as a
        string. (temp commit.)
271
272     Temp commit. More debug info. Maybe fixed some bugs but same error.
273
274     temp commit – fixed compiler warning but old tests are failing
275
276     Fixed old tests, fixed compiler warning
277
278     Added correct(?) invocation of args in thread_init. Still not_found exception
279
280     It was failing to match on [e]. Changed to e, and now it's giving a broken module
        error: params don't match
281
282     Still not working (broken module) but now using lookup_function and pattern
        matching to remove option
283
284     Added a void ptr type for thread (kinda hacky for now but it's for testing threads
        ). Also it's now finding the function from the string name
285
286     Added thread testing script
287
288     THREADS NOW WORK IN SCRIPT!!!
289
290     Passing threading test
291
292     Fixed compiler warnings from pattern matching in codegen
293
294     commit bca9388fld5b7011fde7461b2f1055562f1c7561
295     Author: PLT Student <axxu3795@gmail.com>
296     Date:   Thu May 5 00:59:11 2016 -0400
297
298     Clean compilation without warnings
299
300     commit ecf06799e7b2a68a08ef9603a4b9eacfdcf7b3ce
301     Author: Kyle Lee <kylelee.contact@gmail.com>
302     Date:   Wed May 4 13:28:00 2016 -0400
303
304     Removed codegen warnings, and some semant warnings
305
306     commit 08a4e105a2267891a38e76b4f280a4631bbe3413
307     Author: Kyle Lee <kylelee.contact@gmail.com>
308     Date:   Wed May 4 00:54:52 2016 -0400
309
310     fixed struct tests for let format
311
312     commit 152ab95f7e0c087cc914c0a5c2b176951b77a1d3
313     Merge: b4f812b 116094b
314     Author: Kyle Lee <kylelee.contact@gmail.com>
315     Date:   Wed May 4 00:36:07 2016 -0400
316
317     Merge add_structs
318
319     commit 116094b8ee508fd191c6d793cb14b9b5d6955c2a

```

```

320 Author: PLT Student <axxu3795@gmail.com>
321 Date: Sat Apr 30 20:57:04 2016 -0400
322
323 Semantic checking to disallow circularly dependent structs
324
325 commit 490aa96cafcf6e93d9a5b981f37244cc5b0cb6c6
326 Author: PLT Student <axxu3795@gmail.com>
327 Date: Sat Apr 30 17:03:42 2016 -0400
328
329 Fixed the stack overflow problem and updated tests
330
331 commit 41cb475f79c1d6baf22baf68b02219be8a9a49b2
332 Author: Kyle Lee <kylelee.contact@gmail.com>
333 Date: Sat Apr 30 14:19:54 2016 -0400
334
335 struct access works (messy)
336
337 commit eef6eb9fc000a844957f73dde0a56980a3b44ee0
338 Author: Kyle Lee <kylelee.contact@gmail.com>
339 Date: Sat Apr 30 14:14:10 2016 -0400
340
341 Structs reach llvm failure point. need to clean up exception catching and matches.
342
343 commit b4f812b37b0788d6f4a6d09495f41bc1515488ec
344 Author: Emily Pakulski <enp2111@columbia.edu>
345 Date: Sat Apr 30 13:29:07 2016 -0400
346
347 Flattened built-in function declarations so we don't need extra variables.
348
349 commit 2c0b9cba13e157863e8dcc7fb2f3346a6262c5b1
350 Author: PLT Student <axxu3795@gmail.com>
351 Date: Sat Apr 30 02:06:26 2016 -0400
352
353 changed to named structs
354
355 commit aa095775c0b41e8776214758f2af8d31e142dlea
356 Author: Kyle Lee <kylelee.contact@gmail.com>
357 Date: Fri Apr 29 22:21:28 2016 -0400
358
359 added semant for struct field assignment
360
361 commit c90388e0f0d0eed30323be990eb29ec089fef474
362 Author: PLT Student <axxu3795@gmail.com>
363 Date: Wed Apr 27 21:19:46 2016 -0400
364
365 Created struct field index list
366
367 commit ef7a1054b5250ff47bd60f9fbd2a6e13396e1796
368 Author: PLT Student <axxu3795@gmail.com>
369 Date: Wed Apr 27 03:07:09 2016 -0400
370
371 ltype_of_type now includes struct types so structs can be allocated
372
373 commit e1b6f98760055b9f39c4f0a03606f06d94c2fc8b
374 Author: PLT Student <axxu3795@gmail.com>
375 Date: Tue Apr 26 18:48:27 2016 -0400

```



```

376
377     Cleaned up some warnings, still not sure what 42 is
378
379 commit 24ec2afb38e4bad5f7684ef663aa6d6993116dce
380 Author: PLT Student <axxu3795@gmail.com>
381 Date:   Mon Apr 25 13:38:02 2016 -0400
382
383     Working error checking for struct
384
385 commit 95a5222e09300e7f5037e22c78bd4282cba9929f
386 Author: Kyle Lee <kylelee.contact@gmail.com>
387 Date:   Mon Apr 25 13:01:23 2016 -0400
388
389     Added struct tests
390
391 commit 995258d61bd5f0db54369a7fc65dd6f188e6d415
392 Author: Kyle Lee <kylelee.contact@gmail.com>
393 Date:   Mon Apr 25 12:57:47 2016 -0400
394
395     Working struct semant (throws not found exception)
396
397 commit 037886b737edccbe231b780897b7b31e67e36046
398 Author: Kyle Lee <kylelee.contact@gmail.com>
399 Date:   Sat Apr 23 19:10:56 2016 -0400
400
401     match struct compiles
402
403 commit 6fa5581d2255d28b23d83efafd00b0a868b96740
404 Author: Kyle Lee <kylelee.contact@gmail.com>
405 Date:   Sat Apr 23 18:49:31 2016 -0400
406
407     added broken struct accessor method
408
409 commit e91042b38c3dea600325c9239ddd42b7cbfebf6a
410 Author: PLT Student <axxu3795@gmail.com>
411 Date:   Sat Apr 23 17:47:47 2016 -0400
412
413     Adding check_access, still need to match inside
414
415 commit 3940c80078342f00879360619d0d5f5ad0ba1c57
416 Author: Kyle Lee <kylelee.contact@gmail.com>
417 Date:   Sat Apr 23 17:09:42 2016 -0400
418
419     Prepared to start adding structs to semant.
420
421 commit 450a12b335d46566822e314cbe3030fdc240a17c
422 Author: PLT Student <axxu3795@gmail.com>
423 Date:   Sat Apr 23 16:26:47 2016 -0400
424
425     Gave struct types a string to hold for struct type name
426
427 commit e870131767f7edd529e0a3fbb2b1e9a3ff366bdc
428 Merge: a37ba16 38d78d3
429 Author: Amarto <aar2160@columbia.edu>
430 Date:   Tue Apr 19 23:40:52 2016 -0400
431

```

```

432 Merge pull request #7 from DemocritusLang/change_syntax_order
433
434 Change syntax order with tests
435
436 commit ca8356e47677421467fad358a65bbd16809b4b37
437 Author: PLT Student <axxu3795@gmail.com>
438 Date: Tue Apr 19 21:49:46 2016 -0400
439
440 Added dot operator syntax as a binop
441
442 commit 38d78d3708f6dd5058345b5de776ae035f123240
443 Author: Emily Pakulski <enp2111@columbia.edu>
444 Date: Mon Apr 18 00:23:08 2016 -0400
445
446 Fixed bad string tests.
447
448 commit 9523521d6768e94f504ff983a1deb4738870f897
449 Author: PLT Student <axxu3795@gmail.com>
450 Date: Mon Apr 18 00:01:30 2016 -0400
451
452 I forgot to make clean the last commit b/c i'm dumb
453
454 commit b699bb84863eb86e006f95e82f687f3e367586de
455 Author: PLT Student <axxu3795@gmail.com>
456 Date: Sun Apr 17 23:58:44 2016 -0400
457
458 Compiles with the third struct list
459
460 commit 34076bdcc2f6a519691555482261913623bfd97d
461 Author: Kyle Lee <kylelee.contact@gmail.com>
462 Date: Sun Apr 17 23:36:54 2016 -0400
463
464 Initial addition of struct to parsing
465
466 commit d2221587f8c81155a1ca8f9e2ba50b0a83a89684
467 Author: Amarto <aar2160@columbia.edu>
468 Date: Sun Apr 17 23:36:06 2016 -0400
469
470 Fixed test-helloworld-assign declaration order
471
472 commit 12301820c5bbd32b55b29dfbd5a99068e62ee6b5
473 Author: Emily Pakulski <enp2111@columbia.edu>
474 Date: Sun Apr 17 23:14:02 2016 -0400
475
476 Changed tests to add let keyword.
477
478 commit 3a626ec31e042cfa3bcb8fc5410dd666fae12bea
479 Author: Amarto <aar2160@columbia.edu>
480 Date: Wed Apr 13 01:45:56 2016 -0400
481
482 Changed parser and scanner with LET keyword. Still working on tests
483
484 commit c6ecb302808b192e6e6f51360537556119a867ec
485 Author: Amarto <aar2160@columbia.edu>
486 Date: Tue Mar 15 00:33:01 2016 -0400
487

```

```

488     Temp commit — tried to change variable order but got SR error.
489
490 commit a37ba16c593be6be0d9980aec71b1d2b93eaf69e
491 Author: Kyle Lee <kylelee.contact@gmail.com>
492 Date:   Mon Apr 11 13:05:53 2016 -0400
493
494     Added tentative install instructions (needs testing on Ubuntu 14.x and before)
495
496 commit 605b8bdlf6b6a1612d588ce0c5a52f107292d609
497 Merge:  caa0380 0f67850
498 Author: Amy Xin Xu <axxu3795@gmail.com>
499 Date:   Tue Apr 5 18:36:38 2016 -0400
500
501     Merge pull request #6 from DemocritusLang/strings_2
502
503     HelloWorld checkpoint!
504     :pizza:
505
506 commit 0f678507385ebacba0c05a41eac72ada0d9df015
507 Author: Kyle Lee <kylelee.contact@gmail.com>
508 Date:   Tue Apr 5 18:29:24 2016 -0400
509
510     fixed failing function call test (semant only checks for print())
511
512 commit 22a7dal415ce11b8dbb75ea0a8766e50a48bac38
513 Author: PLT Student <axxu3795@gmail.com>
514 Date:   Tue Apr 5 18:26:25 2016 -0400
515
516     Fixed missing printb
517
518 commit cb55b59f5981f69e28a347525e35ef1071020447
519 Author: = <kylelee.contact@gmail.com>
520 Date:   Tue Apr 5 18:12:31 2016 -0400
521
522     Fixed tarball makefile builder for helloworld
523
524 commit 67ddab48071f203f447caf80f6906870af14e510
525 Author: = <kylelee.contact@gmail.com>
526 Date:   Tue Apr 5 18:02:03 2016 -0400
527
528     Added test case for string assignment and printing.
529
530 commit daceabeb68dbd482000334162a0f797788554616
531 Author: Amarto <aar2160@columbia.edu>
532 Date:   Mon Apr 4 13:49:14 2016 -0400
533
534     Print hello world working. Tests that use printb() are failing, because i had to
        remove it from semant.ml temporarily.
535
536 commit f121b87bdc8a6afac89d6374a9eb2705529d123
537 Author: Amarto <aar2160@columbia.edu>
538 Date:   Mon Apr 4 02:22:46 2016 -0400
539
540     Compiling, but not passing tests.
541
542 commit 93833564906d28a36e6cd8303241e69a764ac2f0

```

```

543 Author: Emily Pakulski <enp2111@columbia.edu>
544 Date: Sun Apr 3 18:11:16 2016 -0400
545
546 Tried moving strings to types...
547
548 commit de2ba3fd50270ec7894abbb7f8c9a1bb0efd8ca3
549 Author: Emily Pakulski <enp2111@columbia.edu>
550 Date: Sun Apr 3 17:56:06 2016 -0400
551
552 Added partial implementation of string literal.
553
554 commit 2fd8d9408fc9ceb78e03d3ebdc2195aee4ad7403
555 Author: Emily Pakulski <enp2111@columbia.edu>
556 Date: Sun Apr 3 16:53:52 2016 -0400
557
558 Added test and function for helloworld. Need string literal implementation.
559
560 commit 0471926a19d8626bab3140b6a12abcf588288620
561 Author: Emily Pakulski <enp2111@columbia.edu>
562 Date: Sun Apr 3 16:27:19 2016 -0400
563
564 Changed Edwards' print to be print_int to avoid confusion with our print
    implementation.
565
566 commit caa0380101c0ac9f657eb626b1930c5ca72bfd5e
567 Author: Emily Pakulski <enp2111@columbia.edu>
568 Date: Mon Mar 14 22:47:26 2016 -0400
569
570 Added function keyword to function declarations.
571
572 commit de3f696465ef9a95a737282d1affa7d1d812cad0
573 Author: Amarto Rajaram <aar2160@columbia.edu>
574 Date: Mon Mar 14 21:58:20 2016 -0400
575
576 Removed while keyword; replaced functionality with for.
577
578 commit d0829835a72243f858bbe2426ab81b04792ed883
579 Author: Amarto Rajaram <amarto.rajaram@gmail.com>
580 Date: Mon Mar 14 21:03:09 2016 -0400
581
582 Added Edwards' tests back in.
583
584 commit 798f67d953e965dae1282e679f6b0e5373982058
585 Author: Emily Pakulski <enp2111@columbia.edu>
586 Date: Fri Feb 26 11:45:05 2016 -0500
587
588 Edwards' MicroC code with our README.

```

threads-2 branch

```

1 commit c298c8ae9920959dd8f6cb25aa76b5b3a25275ac
2 Author: Amarto <aar2160@columbia.edu>
3 Date: Sun May 8 17:35:31 2016 -0400
4
5 THREADS NOW WORK IN SCRIPTgit status!
6

```

```

7  commit f131bac05e4816370e8892d53bd5e57ea996fe3a
8  Author: Amarto <aar2160@columbia.edu>
9  Date:   Sun May 8 14:41:25 2016 -0400
10
11     Added thread testing script
12
13  commit c6f48061c066b532f9cff0d87ed0218fd44d231a
14  Author: Amarto <aar2160@columbia.edu>
15  Date:   Sun May 8 02:52:26 2016 -0400
16
17     Added a void ptr type for thread (kinda hacky for now but it's for testing threads
18         ). Also it's now finding the function from the string name
19
20  commit 2a17bc340b7fca03844fe5db10c2fb2f635ed41c
21  Author: Amarto <aar2160@columbia.edu>
22  Date:   Sat May 7 12:54:29 2016 -0400
23
24     Still not working (broken module) but now using lookup_function and pattern
25     matching to remove option
26
27  commit 97ee5e9b35cdc5bf306e37fc037a41b318f211d6
28  Author: Amarto <aar2160@columbia.edu>
29  Date:   Sat May 7 03:20:03 2016 -0400
30
31     It was failing to match on [e]. Changed to e, and now it's giving a broken module
32     error: params don't match
33
34  commit 81792eeaa686912c0116005af65341b927e69db
35  Author: Amarto <aar2160@columbia.edu>
36  Date:   Sat May 7 00:04:54 2016 -0400
37
38     Added correct(?) invocation of args in thread_init. Still not-found exception
39
40  commit 1faeb7d59482ecda134d22f208d88b914b8bd2e4
41  Author: Amarto <aar2160@columbia.edu>
42  Date:   Thu May 5 02:23:59 2016 -0400
43
44     Fixed old tests, fixed compiler warning
45
46  commit 012414249fc2aee0873d38f67cb2f2a68fb06793
47  Author: Amarto <aar2160@columbia.edu>
48  Date:   Thu May 5 02:01:45 2016 -0400
49
50     temp commit - fixed compiler warning but old tests are failing
51
52  commit 4adb64624bacd11b5d94dbe9a0ac5a6eb5b8d9da
53  Author: Emily Pakulski <enp2111@columbia.edu>
54  Date:   Wed May 4 22:06:33 2016 -0400
55
56     Temp commit. More debug info. Maybe fixed some bugs but same error.
57
58  commit 7e18b4917c27092a34390e173a013feac87a06b0
59  Author: Emily Pakulski <enp2111@columbia.edu>
60  Date:   Wed May 4 21:32:25 2016 -0400
61
62     Fixed some things in test case. Pretty sure function name should be passed in as a

```

```

        string. (temp commit.)
60
61 commit f8b057052c0cba41dlb22f2a4c26bf56a4b0de53
62 Author: Emily Pakulski <enp2111@columbia.edu>
63 Date:   Wed May 4 20:58:01 2016 -0400
64
65     Oops. But still not working.
66
67 commit ec286074732024096574ccb3fb704e4569049325
68 Author: Emily Pakulski <enp2111@columbia.edu>
69 Date:   Wed May 4 19:36:05 2016 -0400
70
71     Baby steps. Still not working. (temp commit).
72
73 commit 583818a84f8f7097075417620fbebela31ccd911
74 Author: Amarto <aar2160@columbia.edu>
75 Date:   Wed May 4 18:38:40 2016 -0400
76
77     Added tests for threading and modified codegen and semant
78
79 commit f5157bee06030e89dbaabdfc842678a67a90e683
80 Author: Amarto <aar2160@columbia.edu>
81 Date:   Wed May 4 17:10:34 2016 -0400
82
83     Working on codegen.ml, but getting compiler warning. Working on threading test,
      but need NULL keyword?
84
85 commit 2e76483930fdbdaa4b216cbd1609da6e70bc44b7
86 Author: Amarto <aar2160@columbia.edu>
87 Date:   Sat Apr 30 15:47:54 2016 -0400
88
89     Added basic threads checking to semant.ml. Need to wait until arguments for
      pthread are passed in
90
91 commit c8d806e00f51e5ca4a8ced271e70e2c46461e68a
92 Author: Amarto <aar2160@columbia.edu>
93 Date:   Sat Apr 30 15:16:57 2016 -0400
94
95     Added declaration of thread() function to codegen. Everything compiles
96
97 commit b4f812b37b0788d6f4a6d09495f41bc1515488ec
98 Author: Emily Pakulski <enp2111@columbia.edu>
99 Date:   Sat Apr 30 13:29:07 2016 -0400
100
101     Flattened built-in function declarations so we don't need extra variables.
102
103 commit e870131767f7edd529e0a3fbb2b1e9a3ff366bdc
104 Merge:  a37ba16 38d78d3
105 Author: Amarto <aar2160@columbia.edu>
106 Date:   Tue Apr 19 23:40:52 2016 -0400
107
108     Merge pull request #7 from DemocritusLang/change_syntax_order
109
110     Change syntax order with tests
111
112 commit 38d78d3708f6dd5058345b5de776ae035f123240

```

```

113 Author: Emily Pakulski <enp2111@columbia.edu>
114 Date: Mon Apr 18 00:23:08 2016 -0400
115
116 Fixed bad string tests.
117
118 commit d2221587f8c81155a1ca8f9e2ba50b0a83a89684
119 Author: Amarto <aar2160@columbia.edu>
120 Date: Sun Apr 17 23:36:06 2016 -0400
121
122 Fixed test-helloworld-assign declaration order
123
124 commit 12301820c5bbd32b55b29dfbd5a99068e62ee6b5
125 Author: Emily Pakulski <enp2111@columbia.edu>
126 Date: Sun Apr 17 23:14:02 2016 -0400
127
128 Changed tests to add let keyword.
129
130 commit 3a626ec31e042cfa3bcb8fc5410dd666fael2bea
131 Author: Amarto <aar2160@columbia.edu>
132 Date: Wed Apr 13 01:45:56 2016 -0400
133
134 Changed parser and scanner with LET keyword. Still working on tests
135
136 commit c6ecb302808b192e6e6f51360537556119a867ec
137 Author: Amarto <aar2160@columbia.edu>
138 Date: Tue Mar 15 00:33:01 2016 -0400
139
140 Temp commit — tried to change variable order but got SR error.
141
142 commit a37ba16c593be6be0d9980aec71b1d2b93eaf69e
143 Author: Kyle Lee <kylelee.contact@gmail.com>
144 Date: Mon Apr 11 13:05:53 2016 -0400
145
146 Added tentative install instructions (needs testing on Ubuntu 14.x and before)
147
148 commit 605b8bd1f6b6a1612d588ce0c5a52f107292d609
149 Merge: caa0380 0f67850
150 Author: Amy Xin Xu <axxu3795@gmail.com>
151 Date: Tue Apr 5 18:36:38 2016 -0400
152
153 Merge pull request #6 from DemocritusLang/strings.2
154
155 HelloWorld checkpoint!
156 :pizza:
157
158 commit 0f678507385ebacba0c05a41eac72ada0d9df015
159 Author: Kyle Lee <kylelee.contact@gmail.com>
160 Date: Tue Apr 5 18:29:24 2016 -0400
161
162 fixed failing function call test (semant only checks for print())
163
164 commit 22a7da1415ce11b8dbb75ea0a8766e50a48bac38
165 Author: PLT Student <axxu3795@gmail.com>
166 Date: Tue Apr 5 18:26:25 2016 -0400
167
168 Fixed missing printb

```

```

169
170 commit cb55b59f5981f69e28a347525e35ef1071020447
171 Author: = <kylelee.contact@gmail.com>
172 Date: Tue Apr 5 18:12:31 2016 -0400
173
174 Fixed tarball makefile builder for helloworld
175
176 commit 67ddab48071f203f447caf80f6906870af14e510
177 Author: = <kylelee.contact@gmail.com>
178 Date: Tue Apr 5 18:02:03 2016 -0400
179
180 Added test case for string assignment and printing.
181
182 commit daceabeb68dbd482000334162a0f797788554616
183 Author: Amarto <aar2160@columbia.edu>
184 Date: Mon Apr 4 13:49:14 2016 -0400
185
186 Print hello world working. Tests that use printb() are failing, because i had to
    remove it from semant.ml temporarily.
187
188 commit f121b87bdc8a6afac89d6374a9eb2705529d123
189 Author: Amarto <aar2160@columbia.edu>
190 Date: Mon Apr 4 02:22:46 2016 -0400
191
192 Compiling, but not passing tests.
193
194 commit 93833564906d28a36e6cd8303241e69a764ac2f0
195 Author: Emily Pakulski <enp2111@columbia.edu>
196 Date: Sun Apr 3 18:11:16 2016 -0400
197
198 Tried moving strings to types...
199
200 commit de2ba3fd50270ec7894abbb7f8c9a1bb0efd8ca3
201 Author: Emily Pakulski <enp2111@columbia.edu>
202 Date: Sun Apr 3 17:56:06 2016 -0400
203
204 Added partial implementation of string literal.
205
206 commit 2fd8d9408fc9ceb78e03d3ebdc2195aee4ad7403
207 Author: Emily Pakulski <enp2111@columbia.edu>
208 Date: Sun Apr 3 16:53:52 2016 -0400
209
210 Added test and function for helloworld. Need string literal implementation.
211
212 commit 0471926a19d8626bab3140b6a12abcf588288620
213 Author: Emily Pakulski <enp2111@columbia.edu>
214 Date: Sun Apr 3 16:27:19 2016 -0400
215
216 Changed Edwards' print to be print.int to avoid confusion with our print
    implementation.
217
218 commit caa0380101c0ac9f657eb626b1930c5ca72bfd5e
219 Author: Emily Pakulski <enp2111@columbia.edu>
220 Date: Mon Mar 14 22:47:26 2016 -0400
221
222 Added function keyword to function declarations.

```



```

223
224 commit de3f696465ef9a95a737282d1affa7d1d812cad0
225 Author: Amarto Rajaram <aar2160@columbia.edu>
226 Date: Mon Mar 14 21:58:20 2016 -0400
227
228     Removed while keyword; replaced functionality with for.
229
230 commit d0829835a72243f858bbe2426ab81b04792ed883
231 Author: Amarto Rajaram <amarto.rajaram@gmail.com>
232 Date: Mon Mar 14 21:03:09 2016 -0400
233
234     Added Edwards' tests back in.
235
236 commit 798f67d953e965dae1282e679f6b0e5373982058
237 Author: Emily Pakulski <enp2111@columbia.edu>
238 Date: Fri Feb 26 11:45:05 2016 -0500
239
240     Edwards' MicroC code with our README.

```

nested-structs branch

```

1 commit 2545a29fbe4a106506f61cd21d00e68024c2d23e
2 Author: PLT Student <axxu3795@gmail.com>
3 Date: Mon May 9 01:06:25 2016 -0400
4
5     IT WORKS
6
7 commit b91958665b48ea59c0cd3b74fc531007c9a59e98
8 Author: PLT Student <axxu3795@gmail.com>
9 Date: Mon May 9 00:21:41 2016 -0400
10
11     IT WORKS... v messy
12
13 commit b39e230a6670c3e04f95b08d17fd7e8b054004f9
14 Author: PLT Student <axxu3795@gmail.com>
15 Date: Sun May 8 23:50:39 2016 -0400
16
17     Throwing error for comparison
18
19 commit a67b43313123361da7e0d7b533ad6a2f025dedf5
20 Author: PLT Student <axxu3795@gmail.com>
21 Date: Thu May 5 17:28:12 2016 -0400
22
23     Access working, not assigns
24
25 commit ba05136aa87a6c755ab85df7aaf95010aachbf86b
26 Author: PLT Student <axxu3795@gmail.com>
27 Date: Thu May 5 04:17:04 2016 -0400
28
29     .lli finally stores
30
31 commit 394caf29ae741057ab618ea52cba773c23e222c0
32 Author: PLT Student <axxu3795@gmail.com>
33 Date: Thu May 5 00:48:16 2016 -0400
34
35     fixed segfault but need to move to new branch

```

```

36
37 commit 2762e837b1d8cbd30920fae7f6e255bfd143b42f
38 Author: PLT Student <axxu3795@gmail.com>
39 Date: Wed May 4 18:43:52 2016 -0400
40
41 still stuck on structs
42
43 commit b845912ac5b327e52a11f7f8c01145e05bee5211
44 Author: PLT Student <axxu3795@gmail.com>
45 Date: Wed May 4 14:07:01 2016 -0400
46
47 How to make pointer to value...
48
49 commit 7428989b6dfe8861b7f4f506281ef5c51d69cd5c
50 Author: PLT Student <axxu3795@gmail.com>
51 Date: Wed May 4 13:38:40 2016 -0400
52
53 I forgot to clean again...
54
55 commit 5e0ab5887107f0f4019caa9b2d5e0c6d391195bb
56 Author: PLT Student <axxu3795@gmail.com>
57 Date: Wed May 4 13:38:19 2016 -0400
58
59 Working on struct type problem still
60
61 commit 1f130f56f0e9d84748f2a2e553f7445bba01b9ec
62 Author: PLT Student <axxu3795@gmail.com>
63 Date: Wed May 4 12:13:12 2016 -0400
64
65 Why is assigning Color to Color failing semant
66
67 commit 2078208599c55277b70ca4f9e41d0913239a30a0
68 Author: PLT Student <axxu3795@gmail.com>
69 Date: Wed May 4 03:21:13 2016 -0400
70
71 nested structs so far
72
73 commit 08a4e105a2267891a38e76b4f280a4631bbe3413
74 Author: Kyle Lee <kylelee.contact@gmail.com>
75 Date: Wed May 4 00:54:52 2016 -0400
76
77 fixed struct tests for let format
78
79 commit 152ab95f7e0c087cc914c0a5c2b176951b77a1d3
80 Merge: b4f812b 116094b
81 Author: Kyle Lee <kylelee.contact@gmail.com>
82 Date: Wed May 4 00:36:07 2016 -0400
83
84 Merge add.structs
85
86 commit 116094b8ee508fd191c6d793cb14b9b5d6955c2a
87 Author: PLT Student <axxu3795@gmail.com>
88 Date: Sat Apr 30 20:57:04 2016 -0400
89
90 Semantic checking to disallow circularly dependent structs
91

```

```

92 commit 490aa96cafcf6e93d9a5b981f37244cc5b0cb6c6
93 Author: PLT Student <axxu3795@gmail.com>
94 Date: Sat Apr 30 17:03:42 2016 -0400
95
96 Fixed the stack overflow problem and updated tests
97
98 commit 41cb475f79c1d6baf22baf68b02219be8a9a49b2
99 Author: Kyle Lee <kylelee.contact@gmail.com>
100 Date: Sat Apr 30 14:19:54 2016 -0400
101
102 struct access works (messy)
103
104 commit eef6eb9fc000a844957f73dde0a56980a3b44ee0
105 Author: Kyle Lee <kylelee.contact@gmail.com>
106 Date: Sat Apr 30 14:14:10 2016 -0400
107
108 Structs reach llvm failure point. need to clean up exception catching and matches.
109
110 commit b4f812b37b0788d6f4a6d09495f41bc1515488ec
111 Author: Emily Pakulski <enp2111@columbia.edu>
112 Date: Sat Apr 30 13:29:07 2016 -0400
113
114 Flattened built-in function declarations so we don't need extra variables.
115
116 commit 2c0b9cba13e157863e8dcc7fb2f3346a6262c5b1
117 Author: PLT Student <axxu3795@gmail.com>
118 Date: Sat Apr 30 02:06:26 2016 -0400
119
120 changed to named structs
121
122 commit aa095775c0b41e8776214758f2af8d31e142dlea
123 Author: Kyle Lee <kylelee.contact@gmail.com>
124 Date: Fri Apr 29 22:21:28 2016 -0400
125
126 added semant for struct field assignment
127
128 commit c90388e0f0d0eed30323be990eb29ec089fef474
129 Author: PLT Student <axxu3795@gmail.com>
130 Date: Wed Apr 27 21:19:46 2016 -0400
131
132 Created struct field index list
133
134 commit ef7a1054b5250ff47bd60f9fbd2a6e13396e1796
135 Author: PLT Student <axxu3795@gmail.com>
136 Date: Wed Apr 27 03:07:09 2016 -0400
137
138 ltype_of.type now includes struct types so structs can be allocated
139
140 commit e1b6f98760055b9f39c4f0a03606f06d94c2fc8b
141 Author: PLT Student <axxu3795@gmail.com>
142 Date: Tue Apr 26 18:48:27 2016 -0400
143
144 Cleaned up some warnings, still not sure what 42 is
145
146 commit 24ec2afb38e4bad5f7684ef663aa6d6993116dce
147 Author: PLT Student <axxu3795@gmail.com>

```

```

148 Date:   Mon Apr 25 13:38:02 2016 -0400
149
150     Working error checking for struct
151
152 commit 95a5222e09300e7f5037e22c78bd4282cba9929f
153 Author: Kyle Lee <kylelee.contact@gmail.com>
154 Date:   Mon Apr 25 13:01:23 2016 -0400
155
156     Added struct tests
157
158 commit 995258d61bd5f0db54369a7fc65dd6f188e6d415
159 Author: Kyle Lee <kylelee.contact@gmail.com>
160 Date:   Mon Apr 25 12:57:47 2016 -0400
161
162     Working struct semant (throws not found exception)
163
164 commit 037886b737edccbe231b780897b7b31e67e36046
165 Author: Kyle Lee <kylelee.contact@gmail.com>
166 Date:   Sat Apr 23 19:10:56 2016 -0400
167
168     match struct compiles
169
170 commit 6fa5581d2255d28b23d83efafd00b0a868b96740
171 Author: Kyle Lee <kylelee.contact@gmail.com>
172 Date:   Sat Apr 23 18:49:31 2016 -0400
173
174     added broken struct accessor method
175
176 commit e91042b38c3dea600325c9239ddd42b7cbfebf6a
177 Author: PLT Student <axxu3795@gmail.com>
178 Date:   Sat Apr 23 17:47:47 2016 -0400
179
180     Adding check-access, still need to match inside
181
182 commit 3940c80078342f00879360619d0d5f5ad0ba1c57
183 Author: Kyle Lee <kylelee.contact@gmail.com>
184 Date:   Sat Apr 23 17:09:42 2016 -0400
185
186     Prepared to start adding structs to semant.
187
188 commit 450a12b335d46566822e314cbe3030fdc240a17c
189 Author: PLT Student <axxu3795@gmail.com>
190 Date:   Sat Apr 23 16:26:47 2016 -0400
191
192     Gave struct types a string to hold for struct type name
193
194 commit e870131767f7edd529e0a3fbb2b1e9a3ff366bdc
195 Merge: a37ba16 38d78d3
196 Author: Amarto <aar2160@columbia.edu>
197 Date:   Tue Apr 19 23:40:52 2016 -0400
198
199     Merge pull request #7 from DemocritusLang/change_syntax_order
200
201     Change syntax order with tests
202
203 commit ca8356e47677421467fad358a65bbd16809b4b37

```

```

204 Author: PLT Student <axxu3795@gmail.com>
205 Date: Tue Apr 19 21:49:46 2016 -0400
206
207 Added dot operator syntax as a binop
208
209 commit 38d78d3708f6dd5058345b5de776ae035f123240
210 Author: Emily Pakulski <enp2111@columbia.edu>
211 Date: Mon Apr 18 00:23:08 2016 -0400
212
213 Fixed bad string tests.
214
215 commit 9523521d6768e94f504ff983a1deb4738870f897
216 Author: PLT Student <axxu3795@gmail.com>
217 Date: Mon Apr 18 00:01:30 2016 -0400
218
219 I forgot to make clean the last commit b/c i'm dumb
220
221 commit b699bb84863eb86e006f95e82f687f3e367586de
222 Author: PLT Student <axxu3795@gmail.com>
223 Date: Sun Apr 17 23:58:44 2016 -0400
224
225 Compiles with the third struct list
226
227 commit 34076bdcc2f6a519691555482261913623bfd97d
228 Author: Kyle Lee <kylelee.contact@gmail.com>
229 Date: Sun Apr 17 23:36:54 2016 -0400
230
231 Initial addition of struct to parsing
232
233 commit d2221587f8c81155a1ca8f9e2ba50b0a83a89684
234 Author: Amarto <aar2160@columbia.edu>
235 Date: Sun Apr 17 23:36:06 2016 -0400
236
237 Fixed test-helloworld-assign declaration order
238
239 commit 12301820c5bbd32b55b29dfbd5a99068e62ee6b5
240 Author: Emily Pakulski <enp2111@columbia.edu>
241 Date: Sun Apr 17 23:14:02 2016 -0400
242
243 Changed tests to add let keyword.
244
245 commit 3a626ec31e042cfa3bcb8fc5410dd666fael2bea
246 Author: Amarto <aar2160@columbia.edu>
247 Date: Wed Apr 13 01:45:56 2016 -0400
248
249 Changed parser and scanner with LET keyword. Still working on tests
250
251 commit c6ecb302808b192e6e6f51360537556119a867ec
252 Author: Amarto <aar2160@columbia.edu>
253 Date: Tue Mar 15 00:33:01 2016 -0400
254
255 Temp commit — tried to change variable order but got SR error.
256
257 commit a37ba16c593be6be0d9980aec71b1d2b93eaf69e
258 Author: Kyle Lee <kylelee.contact@gmail.com>
259 Date: Mon Apr 11 13:05:53 2016 -0400

```

```

260
261     Added tentative install instructions (needs testing on Ubuntu 14.x and before)
262
263 commit 605b8bdlf6b6a1612d588ce0c5a52f107292d609
264 Merge: caa0380 0f67850
265 Author: Amy Xin Xu <axxu3795@gmail.com>
266 Date: Tue Apr 5 18:36:38 2016 -0400
267
268     Merge pull request #6 from DemocritusLang/strings_2
269
270     HelloWorld checkpoint!
271     :pizza:
272
273 commit 0f678507385ebacba0c05a41eac72ada0d9df015
274 Author: Kyle Lee <kylelee.contact@gmail.com>
275 Date: Tue Apr 5 18:29:24 2016 -0400
276
277     fixed failing function call test (semant only checks for print())
278
279 commit 22a7da1415ce11b8dbb75ea0a8766e50a48bac38
280 Author: PLT Student <axxu3795@gmail.com>
281 Date: Tue Apr 5 18:26:25 2016 -0400
282
283     Fixed missing printb
284
285 commit cb55b59f5981f69e28a347525e35ef1071020447
286 Author: = <kylelee.contact@gmail.com>
287 Date: Tue Apr 5 18:12:31 2016 -0400
288
289     Fixed tarball makefile builder for helloworld
290
291 commit 67ddab48071f203f447caf80f6906870af14e510
292 Author: = <kylelee.contact@gmail.com>
293 Date: Tue Apr 5 18:02:03 2016 -0400
294
295     Added test case for string assignment and printing.
296
297 commit daceabeb68dbd482000334162a0f797788554616
298 Author: Amarto <aar2160@columbia.edu>
299 Date: Mon Apr 4 13:49:14 2016 -0400
300
301     Print hello world working. Tests that use printb() are failing, because i had to
        remove it from semant.ml temporarily.
302
303 commit f121b87bdc8a6afac89d6374a9eb2705529d123
304 Author: Amarto <aar2160@columbia.edu>
305 Date: Mon Apr 4 02:22:46 2016 -0400
306
307     Compiling, but not passing tests.
308
309 commit 93833564906d28a36e6cd8303241e69a764ac2f0
310 Author: Emily Pakulski <enp2111@columbia.edu>
311 Date: Sun Apr 3 18:11:16 2016 -0400
312
313     Tried moving strings to types...
314

```

```

315 commit de2ba3fd50270ec7894abbb7f8c9a1bb0efd8ca3
316 Author: Emily Pakulski <enp2111@columbia.edu>
317 Date: Sun Apr 3 17:56:06 2016 -0400
318
319 Added partial implementation of string literal.
320
321 commit 2fd8d9408fc9ceb78e03d3ebdc2195aee4ad7403
322 Author: Emily Pakulski <enp2111@columbia.edu>
323 Date: Sun Apr 3 16:53:52 2016 -0400
324
325 Added test and function for helloworld. Need string literal implementation.
326
327 commit 0471926a19d8626bab3140b6a12abcf588288620
328 Author: Emily Pakulski <enp2111@columbia.edu>
329 Date: Sun Apr 3 16:27:19 2016 -0400
330
331 Changed Edwards' print to be print_int to avoid confusion with our print
    implementation.
332
333 commit caa0380101c0ac9f657eb626b1930c5ca72bfd5e
334 Author: Emily Pakulski <enp2111@columbia.edu>
335 Date: Mon Mar 14 22:47:26 2016 -0400
336
337 Added function keyword to function declarations.
338
339 commit de3f696465ef9a95a737282d1affa7d1d812cad0
340 Author: Amarto Rajaram <aar2160@columbia.edu>
341 Date: Mon Mar 14 21:58:20 2016 -0400
342
343 Removed while keyword; replaced functionality with for.
344
345 commit d0829835a72243f858bbe2426ab81b04792ed883
346 Author: Amarto Rajaram <amarto.rajaram@gmail.com>
347 Date: Mon Mar 14 21:03:09 2016 -0400
348
349 Added Edwards' tests back in.
350
351 commit 798f67d953e965dae1282e679f6b0e5373982058
352 Author: Emily Pakulski <enp2111@columbia.edu>
353 Date: Fri Feb 26 11:45:05 2016 -0500
354
355 Edwards' MicroC code with our README.

```

linkedlist-and-stack branch

```

1 commit cb6b5091e6812c3f3a60869bdb39bdd6900c201e
2 Author: PLT Student <axxu3795@gmail.com>
3 Date: Wed May 11 17:35:32 2016 -0400
4
5 Final linked list demo
6
7 Fixed merge accident
8
9 Fixed malloc size
10
11 Added to demo folder

```

```

12
13 commit d0a23f079f4a9828c6d2724141c5b7de1a126958
14 Author: Emily Pakulski <enp2111@columbia.edu>
15 Date: Wed May 11 17:52:08 2016 -0400
16
17     Added simple threads test.
18
19 commit e113aa58cc30af12cad615d07ed57b99905e0edf
20 Author: Emily <ohEmily@users.noreply.github.com>
21 Date: Wed May 11 16:26:44 2016 -0400
22
23     Multithreading and networking working together. Added concurrent comic download.
24     (#23)
25
26     * Added multithreaded test- getting parse error. Added strcat and int to string
27     wrappers
28
29     * fixed parse error.
30
31     * Added memset to fix test-multithreaded-sockets.
32
33     * Fix memset bug
34
35     * Updated test file so sockets test passes
36
37     * Fixed thread function signature
38
39     * Added failing test.
40
41     * Fixed bug in init_thread() and added test that passes string into thread().
42
43     * possibly fixed request bug.
44
45     * Fixed binary file reading bugs.
46
47     * Moved code into demo directory.
48
49     * Fixed merge conflicts after rebase.
50
51 commit d4972e0ff8b5ba850bf39528cc0b113bc1912ee5
52 Merge: dbb06cc 232belf
53 Author: Amy Xin Xu <axxu3795@gmail.com>
54 Date: Wed May 11 16:10:39 2016 -0400
55
56     Merge pull request #22 from DemocritusLang/build_malloc_attempt
57
58     Malloc and simple linked lists working
59
60 commit 232belf27fb7ffa7110e5184c6016b81e2da94ff
61 Author: PLT Student <axxu3795@gmail.com>
62 Date: Wed May 11 03:45:50 2016 -0400
63
64     Malloc and simple linked lists working
65
66     Fixed shift reduce errors

```



```

66      temp commit
67
68      CASTING AND MALLOC WORK
69
70      Cleaned up warnings
71
72      Cleaned up codegen warnings
73
74      Fixed (*a). dotops and halfassed addnode
75
76      Half way linked lists
77
78      Linked lists with add and print_list functions
79
80  commit dbb06cc9f0f2f608583946158d48d8d841d8dc62
81  Author: Emily Pakulski <enp2111@columbia.edu>
82  Date:   Wed May 11 13:30:05 2016 -0400
83
84      Added check in tester for whether code was already compiled.
85
86  commit 36aee48817bdeec1553b1c16963e5abcc7aaaa6b
87  Merge: 0371f67 2402f74
88  Author: Amarto <aar2160@columbia.edu>
89  Date:   Wed May 11 06:10:49 2016 -0400
90
91      Merge pull request #21 from DemocritusLang/sockets.finished
92
93      Sockets finished
94
95  commit 2402f740bec59d6ed4912b2ee302d9b9b7bb5480
96  Author: Amarto <aar2160@columbia.edu>
97  Date:   Wed May 11 05:45:05 2016 -0400
98
99      Added free(), execl wrapper, and corrected output reference for socket test.
      Changed tests to use free() after malloc. Refactored the weblink downloading
      method to only take one param so it matches the signature for a thread function
100
101  commit 170645297d4578f9551f690b08b82718beebf033
102  Author: Emily Pakulski <enp2111@columbia.edu>
103  Date:   Wed May 11 01:44:23 2016 -0400
104
105      Changed up get request impl a bit.
106
107  commit 5226ad21d850b6622e09388a4337d0b515123f76
108  Author: Amarto <aar2160@columbia.edu>
109  Date:   Tue May 10 17:46:11 2016 -0400
110
111      Added basic socket impl and loading files. Need to handle tests
112
113  commit 0371f67bac394c08ae72cb372491fd3264839f65
114  Merge: a6ce096 a784099
115  Author: Amy Xin Xu <axxu3795@gmail.com>
116  Date:   Wed May 11 02:57:57 2016 -0400
117
118      Merge pull request #20 from DemocritusLang/add.float.and.mod
119

```

```

120     Added modulo and floats
121
122 commit a78409901632a6f70e889fee8f5ccff2bfe12989
123 Author: PLT Student <axxu3795@gmail.com>
124 Date:   Tue May 10 23:26:55 2016 -0400
125
126     Added modulo and floats
127
128     Mod done
129
130     Working on floats
131
132     fixed floating print issue
133
134     Working floats
135
136     Added floats in struct test
137
138 commit a6ce096863beb92e5b5251b0802613edab31cc76
139 Author: Emily <ohEmily@users.noreply.github.com>
140 Date:   Tue May 10 23:09:23 2016 -0400
141
142     Added sleep function and test. (#18)
143
144 commit ff330840be2c13aeed6f2a6d87a69ce153f29421
145 Merge: a63b40f fdcadf4
146 Author: Amy Xin Xu <axxu3795@gmail.com>
147 Date:   Tue May 10 15:28:57 2016 -0400
148
149     Merge pull request #17 from DemocritusLang/add_pointers
150
151     Pointers done
152
153 commit fdcadf4fa032354c8a9ec96f41cecb76b94e66f0
154 Author: PLT Student <axxu3795@gmail.com>
155 Date:   Tue May 10 02:02:43 2016 -0400
156
157     Pointers done
158
159     Dereference syntax there, need to clean warning
160
161     Added ref op and semantic checking
162
163     Working pointers for ints, need to test rest
164
165     Modified test-pointer-int.dem for clarity and wrote test-pointer-bool, passing
166
167     Added single and multilevel struct tests, passing
168
169     Linkedlist test not working
170
171     Added hypothetical linkedlist tests (not working
172
173     Linked of list proof of concept
174
175     Changed type* to *type to reflect Go syntax

```

```

176
177 commit a63b40fb618149ec65e57ea3ac986bdccf9f4ac4
178 Author: Kyle Lee <kylelee.contact@gmail.com>
179 Date: Tue May 10 01:11:44 2016 -0400
180
181 Added singleline comments
182
183 commit 901814668aa2d7513fe74b45fa4390d82635ac01
184 Author: Amarto <aar2160@columbia.edu>
185 Date: Tue May 10 00:46:58 2016 -0400
186
187 Fixed void pointer notation to match Go syntax, fixed test
188
189 commit 0ed94930f8362cb6eb322ec6a9570043660aabb5
190 Merge: b662f67 d9b467b
191 Author: Amy Xin Xu <axxu3795@gmail.com>
192 Date: Tue May 10 00:43:18 2016 -0400
193
194 Merge pull request #15 from DemocritusLang/clean.nested.structs
195
196 Working nested structs
197
198 commit d9b467b80ce152696875d6ec1d3d2f1ec6ea77e6
199 Author: PLT Student <axxu3795@gmail.com>
200 Date: Mon May 9 22:47:07 2016 -0400
201
202 Working nested structs
203
204 Added nested struct test
205
206 Fixed mistyped identifier
207
208 nested structs working
209
210 Fixed typo in test-structs-nested.out and added another test
211
212 Edited test to be more informative of functionality
213
214 test-struct-nested1
215
216 commit b662f676ae12fbb27eedaf5af6ae990d76f423bc
217 Author: Emily Pakulski <enp2111@columbia.edu>
218 Date: Mon May 9 20:34:05 2016 -0400
219
220 Finished file I/O. lseek also implemented.
221
222 commit 84c1fc11bc2a2e59b8fec9d68937db8205f1b5d9
223 Author: Amarto <aar2160@columbia.edu>
224 Date: Sun May 8 20:30:58 2016 -0400
225
226 Added malloc and started file I/O.
227
228 commit 8b3944051cfde07be958214aae56bf47988fb803
229 Author: Emily <ohEmily@users.noreply.github.com>
230 Date: Mon May 9 11:15:23 2016 -0400
231

```

```

232 Updated all instances of MicroC to Democritus and added 'make all' target (#12)
233
234 * Changed MicroC -> Democritus and added make all target.
235
236 * Changed file extension for democrituslang files from .mc to .dem.
237
238 commit ed27ce5f8a31a740f3eb0e5ad3ff3cf7a838f9
239 Author: Amarto <aar2160@columbia.edu>
240 Date: Sun May 8 19:31:43 2016 -0400
241
242 Fixed warnings resulting from merge
243
244 commit c6cbdf15fd8854a02fb695fa9aa41b50966431a7
245 Author: Amarto <aar2160@columbia.edu>
246 Date: Sat Apr 30 15:16:57 2016 -0400
247
248 Added multithreading and void pointers, and added calling bound C functions
249 Added declaration of thread() function to codegen. Everything compiles
250
251 Added basic threads checking to semant.ml. Need to wait until arguments for
    pthread are passed in
252
253 Working on codegen.ml, but getting compiler warning. Working on threading test,
    but need NULL keyword?
254
255 Added tests for threading and modified codegen and semant
256
257 Baby steps. Still not working. (temp commit).
258
259 Oops. But still not working.
260
261 Fixed some things in test case. Pretty sure function name should be passed in as a
    string. (temp commit.)
262
263 Temp commit. More debug info. Maybe fixed some bugs but same error.
264
265 temp commit - fixed compiler warning but old tests are failing
266
267 Fixed old tests, fixed compiler warning
268
269 Added correct(?) invocation of args in thread_init. Still not-found exception
270
271 It was failing to match on [e]. Changed to e, and now it's giving a broken module
    error: params don't match
272
273 Still not working (broken module) but now using lookup_function and pattern
    matching to remove option
274
275 Added a void ptr type for thread (kinda hacky for now but it's for testing threads
    ). Also it's now finding the function from the string name
276
277 Added thread testing script
278
279 THREADS NOW WORK IN SCRIPT!!!
280
281 Passing threading test

```

```

282
283     Fixed compiler warnings from pattern matching in codegen
284
285 commit bca9388f1d5b7011fde7461b2f1055562f1c7561
286 Author: PLT Student <axxu3795@gmail.com>
287 Date:   Thu May 5 00:59:11 2016 -0400
288
289     Clean compilation without warnings
290
291 commit ecf06799e7b2a68a08ef9603a4b9eacfdcf7b3ce
292 Author: Kyle Lee <kylelee.contact@gmail.com>
293 Date:   Wed May 4 13:28:00 2016 -0400
294
295     Removed codegen warnings, and some semant warnings
296
297 commit 08a4e105a2267891a38e76b4f280a4631bbe3413
298 Author: Kyle Lee <kylelee.contact@gmail.com>
299 Date:   Wed May 4 00:54:52 2016 -0400
300
301     fixed struct tests for let format
302
303 commit 152ab95f7e0c087cc914c0a5c2b176951b77a1d3
304 Merge: b4f812b 116094b
305 Author: Kyle Lee <kylelee.contact@gmail.com>
306 Date:   Wed May 4 00:36:07 2016 -0400
307
308     Merge add.structs
309
310 commit 116094b8ee508fd191c6d793cb14b9b5d6955c2a
311 Author: PLT Student <axxu3795@gmail.com>
312 Date:   Sat Apr 30 20:57:04 2016 -0400
313
314     Semantic checking to disallow circularly dependent structs
315
316 commit 490aa96cafcf6e93d9a5b981f37244cc5b0cb6c6
317 Author: PLT Student <axxu3795@gmail.com>
318 Date:   Sat Apr 30 17:03:42 2016 -0400
319
320     Fixed the stack overflow problem and updated tests
321
322 commit 41cb475f79c1d6baf22baf68b02219be8a9a49b2
323 Author: Kyle Lee <kylelee.contact@gmail.com>
324 Date:   Sat Apr 30 14:19:54 2016 -0400
325
326     struct access works (messy)
327
328 commit eef6eb9fc000a844957f73dde0a56980a3b44ee0
329 Author: Kyle Lee <kylelee.contact@gmail.com>
330 Date:   Sat Apr 30 14:14:10 2016 -0400
331
332     Structs reach llvm failure point. need to clean up exception catching and matches.
333
334 commit b4f812b37b0788d6f4a6d09495f41bc1515488ec
335 Author: Emily Pakulski <enp2111@columbia.edu>
336 Date:   Sat Apr 30 13:29:07 2016 -0400
337

```

```

338     Flattened built-in function declarations so we don't need extra variables.
339
340 commit 2c0b9cba13e157863e8dcc7fb2f3346a6262c5b1
341 Author: PLT Student <axxu3795@gmail.com>
342 Date:   Sat Apr 30 02:06:26 2016 -0400
343
344     changed to named structs
345
346 commit aa095775c0b41e8776214758f2af8d31e142dlea
347 Author: Kyle Lee <kylelee.contact@gmail.com>
348 Date:   Fri Apr 29 22:21:28 2016 -0400
349
350     added semant for struct field assignment
351
352 commit c90388e0f0d0eed30323be990eb29ec089fef474
353 Author: PLT Student <axxu3795@gmail.com>
354 Date:   Wed Apr 27 21:19:46 2016 -0400
355
356     Created struct field index list
357
358 commit ef7a1054b5250ff47bd60f9fbd2a6e13396e1796
359 Author: PLT Student <axxu3795@gmail.com>
360 Date:   Wed Apr 27 03:07:09 2016 -0400
361
362     ltype_of.type now includes struct types so structs can be allocated
363
364 commit e1b6f98760055b9f39c4f0a03606f06d94c2fc8b
365 Author: PLT Student <axxu3795@gmail.com>
366 Date:   Tue Apr 26 18:48:27 2016 -0400
367
368     Cleaned up some warnings, still not sure what 42 is
369
370 commit 24ec2afb38e4bad5f7684ef663aa6d6993116dce
371 Author: PLT Student <axxu3795@gmail.com>
372 Date:   Mon Apr 25 13:38:02 2016 -0400
373
374     Working error checking for struct
375
376 commit 95a5222e09300e7f5037e22c78bd4282cba9929f
377 Author: Kyle Lee <kylelee.contact@gmail.com>
378 Date:   Mon Apr 25 13:01:23 2016 -0400
379
380     Added struct tests
381
382 commit 995258d61bd5f0db54369a7fc65dd6f188e6d415
383 Author: Kyle Lee <kylelee.contact@gmail.com>
384 Date:   Mon Apr 25 12:57:47 2016 -0400
385
386     Working struct semant (throws not found exception)
387
388 commit 037886b737edccbe231b780897b7b31e67e36046
389 Author: Kyle Lee <kylelee.contact@gmail.com>
390 Date:   Sat Apr 23 19:10:56 2016 -0400
391
392     match struct compiles
393

```

```

394 commit 6fa5581d2255d28b23d83efafd00b0a868b96740
395 Author: Kyle Lee <kylelee.contact@gmail.com>
396 Date: Sat Apr 23 18:49:31 2016 -0400
397
398     added broken struct accessor method
399
400 commit e91042b38c3dea600325c9239ddd42b7cbfebf6a
401 Author: PLT Student <axxu3795@gmail.com>
402 Date: Sat Apr 23 17:47:47 2016 -0400
403
404     Adding check.access, still need to match inside
405
406 commit 3940c80078342f00879360619d0d5f5ad0balc57
407 Author: Kyle Lee <kylelee.contact@gmail.com>
408 Date: Sat Apr 23 17:09:42 2016 -0400
409
410     Prepared to start adding structs to semant.
411
412 commit 450a12b335d46566822e314cbe3030fdc240a17c
413 Author: PLT Student <axxu3795@gmail.com>
414 Date: Sat Apr 23 16:26:47 2016 -0400
415
416     Gave struct types a string to hold for struct type name
417
418 commit e870131767f7edd529e0a3fbb2ble9a3ff366bdc
419 Merge: a37ba16 38d78d3
420 Author: Amarto <aar2160@columbia.edu>
421 Date: Tue Apr 19 23:40:52 2016 -0400
422
423     Merge pull request #7 from DemocritusLang/change.syntax.order
424
425     Change syntax order with tests
426
427 commit ca8356e47677421467fad358a65bbd16809b4b37
428 Author: PLT Student <axxu3795@gmail.com>
429 Date: Tue Apr 19 21:49:46 2016 -0400
430
431     Added dot operator syntax as a binop
432
433 commit 38d78d3708f6dd5058345b5de776ae035f123240
434 Author: Emily Pakulski <enp2111@columbia.edu>
435 Date: Mon Apr 18 00:23:08 2016 -0400
436
437     Fixed bad string tests.
438
439 commit 9523521d6768e94f504ff983a1deb4738870f897
440 Author: PLT Student <axxu3795@gmail.com>
441 Date: Mon Apr 18 00:01:30 2016 -0400
442
443     I forgot to make clean the last commit b/c i'm dumb
444
445 commit b699bb84863eb86e006f95e82f687f3e367586de
446 Author: PLT Student <axxu3795@gmail.com>
447 Date: Sun Apr 17 23:58:44 2016 -0400
448
449     Compiles with the third struct list

```

```

450
451 commit 34076bdcc2f6a519691555482261913623bfd97d
452 Author: Kyle Lee <kylelee.contact@gmail.com>
453 Date: Sun Apr 17 23:36:54 2016 -0400
454
455 Initial addition of struct to parsing
456
457 commit d2221587f8c81155a1ca8f9e2ba50b0a83a89684
458 Author: Amarto <aar2160@columbia.edu>
459 Date: Sun Apr 17 23:36:06 2016 -0400
460
461 Fixed test-helloworld-assign declaration order
462
463 commit 12301820c5bbd32b55b29dfbd5a99068e62ee6b5
464 Author: Emily Pakulski <enp2111@columbia.edu>
465 Date: Sun Apr 17 23:14:02 2016 -0400
466
467 Changed tests to add let keyword.
468
469 commit 3a626ec31e042cfa3bcb8fc5410dd666fael2bea
470 Author: Amarto <aar2160@columbia.edu>
471 Date: Wed Apr 13 01:45:56 2016 -0400
472
473 Changed parser and scanner with LET keyword. Still working on tests
474
475 commit c6ecb302808b192e6e6f51360537556119a867ec
476 Author: Amarto <aar2160@columbia.edu>
477 Date: Tue Mar 15 00:33:01 2016 -0400
478
479 Temp commit — tried to change variable order but got SR error.
480
481 commit a37ba16c593be6be0d9980aec71b1d2b93eaf69e
482 Author: Kyle Lee <kylelee.contact@gmail.com>
483 Date: Mon Apr 11 13:05:53 2016 -0400
484
485 Added tentative install instructions (needs testing on Ubuntu 14.x and before)
486
487 commit 605b8bdlf6b6a1612d588ce0c5a52f107292d609
488 Merge: caa0380 0f67850
489 Author: Amy Xin Xu <axxu3795@gmail.com>
490 Date: Tue Apr 5 18:36:38 2016 -0400
491
492 Merge pull request #6 from DemocritusLang/strings_2
493
494 HelloWorld checkpoint!
495 :pizza:
496
497 commit 0f678507385ebacba0c05a41eac72ada0d9df015
498 Author: Kyle Lee <kylelee.contact@gmail.com>
499 Date: Tue Apr 5 18:29:24 2016 -0400
500
501 fixed failing function call test (semant only checks for print())
502
503 commit 22a7da1415ce11b8dbb75ea0a8766e50a48bac38
504 Author: PLT Student <axxu3795@gmail.com>
505 Date: Tue Apr 5 18:26:25 2016 -0400

```



```

506
507     Fixed missing printb
508
509 commit cb55b59f5981f69e28a347525e35ef1071020447
510 Author: = <kylelee.contact@gmail.com>
511 Date:   Tue Apr 5 18:12:31 2016 -0400
512
513     Fixed tarball makefile builder for helloworld
514
515 commit 67ddab48071f203f447caf80f6906870af14e510
516 Author: = <kylelee.contact@gmail.com>
517 Date:   Tue Apr 5 18:02:03 2016 -0400
518
519     Added test case for string assignment and printing.
520
521 commit daceabeb68dbd482000334162a0f797788554616
522 Author: Amarto <aar2160@columbia.edu>
523 Date:   Mon Apr 4 13:49:14 2016 -0400
524
525     Print hello world working. Tests that use printb() are failing, because i had to
        remove it from semant.ml temporarily.
526
527 commit f121b87bdc8a6afac89d6374a9eb2705529d123
528 Author: Amarto <aar2160@columbia.edu>
529 Date:   Mon Apr 4 02:22:46 2016 -0400
530
531     Compiling, but not passing tests.
532
533 commit 93833564906d28a36e6cd8303241e69a764ac2f0
534 Author: Emily Pakulski <enp2111@columbia.edu>
535 Date:   Sun Apr 3 18:11:16 2016 -0400
536
537     Tried moving strings to types...
538
539 commit de2ba3fd50270ec7894abbb7f8c9a1bb0efd8ca3
540 Author: Emily Pakulski <enp2111@columbia.edu>
541 Date:   Sun Apr 3 17:56:06 2016 -0400
542
543     Added partial implementation of string literal.
544
545 commit 2fd8d9408fc9ceb78e03d3ebdc2195aee4ad7403
546 Author: Emily Pakulski <enp2111@columbia.edu>
547 Date:   Sun Apr 3 16:53:52 2016 -0400
548
549     Added test and function for helloworld. Need string literal implementation.
550
551 commit 0471926a19d8626bab3140b6a12abcf588288620
552 Author: Emily Pakulski <enp2111@columbia.edu>
553 Date:   Sun Apr 3 16:27:19 2016 -0400
554
555     Changed Edwards' print to be print_int to avoid confusion with our print
        implementation.
556
557 commit caa0380101c0ac9f657eb626b1930c5ca72bfd5e
558 Author: Emily Pakulski <enp2111@columbia.edu>
559 Date:   Mon Mar 14 22:47:26 2016 -0400

```

```

560
561     Added function keyword to function declarations.
562
563 commit de3f696465ef9a95a737282d1affa7d1d812cad0
564 Author: Amarto Rajaram <aar2160@columbia.edu>
565 Date:   Mon Mar 14 21:58:20 2016 -0400
566
567     Removed while keyword; replaced functionality with for.
568
569 commit d0829835a72243f858bbe2426ab81b04792ed883
570 Author: Amarto Rajaram <amarto.rajaram@gmail.com>
571 Date:   Mon Mar 14 21:03:09 2016 -0400
572
573     Added Edwards' tests back in.
574
575 commit 798f67d953e965dae1282e679f6b0e5373982058
576 Author: Emily Pakulski <enp2111@columbia.edu>
577 Date:   Fri Feb 26 11:45:05 2016 -0500
578
579     Edwards' MicroC code with our README.

```

fix-malloc branch

```

1  commit c7f317b4ddddd1e0cd4a2152b83e4a65feaadd128
2  Author: PLT Student <axxu3795@gmail.com>
3  Date:   Wed May 11 01:28:24 2016 -0400
4
5      commented out check.assign, test-pointer-malloc replicating malloc problem in
        codegen
6
7  commit a6ce096863beb92e5b5251b0802613edab31cc76
8  Author: Emily <ohEmily@users.noreply.github.com>
9  Date:   Tue May 10 23:09:23 2016 -0400
10
11     Added sleep function and test. (#18)
12
13 commit ff330840be2c13aeed6f2a6d87a69ce153f29421
14 Merge: a63b40f fdcadf4
15 Author: Amy Xin Xu <axxu3795@gmail.com>
16 Date:   Tue May 10 15:28:57 2016 -0400
17
18     Merge pull request #17 from DemocritusLang/add_pointers
19
20     Pointers done
21
22 commit fdcadf4fa032354c8a9ec96f41cecb76b94e66f0
23 Author: PLT Student <axxu3795@gmail.com>
24 Date:   Tue May 10 02:02:43 2016 -0400
25
26     Pointers done
27
28     Dereference syntax there, need to clean warning
29
30     Added ref op and semantic checking
31
32     Working pointers for ints, need to test rest

```

```

33
34     Modified test-pointer-int.dem for clarity and wrote test-pointer-bool, passing
35
36     Added single and multilevel struct tests, passing
37
38     Linkedlist test not working
39
40     Added hypothetical linkedlist tests (not working
41
42     Linked of list proof of concept
43
44     Changed type* to *type to reflect Go syntax
45
46 commit a63b40fb618149ec65e57ea3ac986bdccf9f4ac4
47 Author: Kyle Lee <kylelee.contact@gmail.com>
48 Date:   Tue May 10 01:11:44 2016 -0400
49
50     Added singleline comments
51
52 commit 901814668aa2d7513fe74b45fa4390d82635ac01
53 Author: Amarto <aar2160@columbia.edu>
54 Date:   Tue May 10 00:46:58 2016 -0400
55
56     Fixed void pointer notation to match Go syntax, fixed test
57
58 commit 0ed94930f8362cb6eb322ec6a9570043660aabb5
59 Merge: b662f67 d9b467b
60 Author: Amy Xin Xu <axxu3795@gmail.com>
61 Date:   Tue May 10 00:43:18 2016 -0400
62
63     Merge pull request #15 from DemocritusLang/clean.nested.structs
64
65     Working nested structs
66
67 commit d9b467b80ce152696875d6ecd3d2f1ec6ea77e6
68 Author: PLT Student <axxu3795@gmail.com>
69 Date:   Mon May 9 22:47:07 2016 -0400
70
71     Working nested structs
72
73     Added nested struct test
74
75     Fixed mistyped identifier
76
77     nested structs working
78
79     Fixed typo in test-structs-nested.out and added another test
80
81     Edited test to be more informative of functionality
82
83     test-struct-nested1
84
85 commit b662f676ae12fbb27eedaf5af6ae990d76f423bc
86 Author: Emily Pakulski <enp2111@columbia.edu>
87 Date:   Mon May 9 20:34:05 2016 -0400
88

```

```

89     Finished file I/O. lseek also implemented.
90
91 commit 84c1fc11bc2a2e59b8fec9d68937db8205f1b5d9
92 Author: Amarto <aar2160@columbia.edu>
93 Date:    Sun May 8 20:30:58 2016 -0400
94
95     Added malloc and started file I/O.
96
97 commit 8b3944051cfde07be958214aae56bf47988fb803
98 Author: Emily <ohEmily@users.noreply.github.com>
99 Date:    Mon May 9 11:15:23 2016 -0400
100
101     Updated all instances of MicroC to Democritus and added 'make all' target (#12)
102
103     * Changed MicroC -> Democritus and added make all target.
104
105     * Changed file extension for democrituslang files from .mc to .dem.
106
107 commit ed27ce5f8a31a740f3eb0e5ad3ff3cf7a838f9
108 Author: Amarto <aar2160@columbia.edu>
109 Date:    Sun May 8 19:31:43 2016 -0400
110
111     Fixed warnings resulting from merge
112
113 commit c6cbdf15fd8854a02fb695fa9aa41b50966431a7
114 Author: Amarto <aar2160@columbia.edu>
115 Date:    Sat Apr 30 15:16:57 2016 -0400
116
117     Added multithreading and void pointers, and added calling bound C functions
118     Added declaration of thread() function to codegen. Everything compiles
119
120     Added basic threads checking to semant.ml. Need to wait until arguments for
        pthread are passed in
121
122     Working on codegen.ml, but getting compiler warning. Working on threading test,
        but need NULL keyword?
123
124     Added tests for threading and modified codegen and semant
125
126     Baby steps. Still not working. (temp commit).
127
128     Oops. But still not working.
129
130     Fixed some things in test case. Pretty sure function name should be passed in as a
        string. (temp commit.)
131
132     Temp commit. More debug info. Maybe fixed some bugs but same error.
133
134     temp commit - fixed compiler warning but old tests are failing
135
136     Fixed old tests, fixed compiler warning
137
138     Added correct(?) invocation of args in thread_init. Still not_found exception
139
140     It was failing to match on [e]. Changed to e, and now it's giving a broken module
        error: params don't match

```

```

141
142     Still not working (broken module) but now using lookup_function and pattern
      matching to remove option
143
144     Added a void ptr type for thread (kinda hacky for now but it's for testing threads
      ). Also it's now finding the function from the string name
145
146     Added thread testing script
147
148     THREADS NOW WORK IN SCRIPT!!!
149
150     Passing threading test
151
152     Fixed compiler warnings from pattern matching in codegen
153
154     commit bca9388f1d5b7011fde7461b2f1055562f1c7561
155     Author: PLT Student <axxu3795@gmail.com>
156     Date:   Thu May 5 00:59:11 2016 -0400
157
158     Clean compilation without warnings
159
160     commit ecf06799e7b2a68a08ef9603a4b9eacfd7b3ce
161     Author: Kyle Lee <kylelee.contact@gmail.com>
162     Date:   Wed May 4 13:28:00 2016 -0400
163
164     Removed codegen warnings, and some semant warnings
165
166     commit 08a4e105a2267891a38e76b4f280a4631bbe3413
167     Author: Kyle Lee <kylelee.contact@gmail.com>
168     Date:   Wed May 4 00:54:52 2016 -0400
169
170     fixed struct tests for let format
171
172     commit 152ab95f7e0c087cc914c0a5c2b176951b77a1d3
173     Merge:  b4f812b 116094b
174     Author: Kyle Lee <kylelee.contact@gmail.com>
175     Date:   Wed May 4 00:36:07 2016 -0400
176
177     Merge add.structs
178
179     commit 116094b8ee508fd191c6d793cb14b9b5d6955c2a
180     Author: PLT Student <axxu3795@gmail.com>
181     Date:   Sat Apr 30 20:57:04 2016 -0400
182
183     Semantic checking to disallow circularly dependent structs
184
185     commit 490aa96caf6e93d9a5b981f37244cc5b0cb6c6
186     Author: PLT Student <axxu3795@gmail.com>
187     Date:   Sat Apr 30 17:03:42 2016 -0400
188
189     Fixed the stack overflow problem and updated tests
190
191     commit 41cb475f79c1d6baf22baf68b02219be8a9a49b2
192     Author: Kyle Lee <kylelee.contact@gmail.com>
193     Date:   Sat Apr 30 14:19:54 2016 -0400
194

```

```

195     struct access works (messy)
196
197 commit eef6eb9fc000a844957f73dde0a56980a3b44ee0
198 Author: Kyle Lee <kylelee.contact@gmail.com>
199 Date:   Sat Apr 30 14:14:10 2016 -0400
200
201     Structs reach llvm failure point. need to clean up exception catching and matches.
202
203 commit b4f812b37b0788d6f4a6d09495f41bc1515488ec
204 Author: Emily Pakulski <enp2111@columbia.edu>
205 Date:   Sat Apr 30 13:29:07 2016 -0400
206
207     Flattened built-in function declarations so we don't need extra variables.
208
209 commit 2c0b9cba13e157863e8dcc7fb2f3346a6262c5b1
210 Author: PLT Student <axxu3795@gmail.com>
211 Date:   Sat Apr 30 02:06:26 2016 -0400
212
213     changed to named structs
214
215 commit aa095775c0b41e8776214758f2af8d31e142dlea
216 Author: Kyle Lee <kylelee.contact@gmail.com>
217 Date:   Fri Apr 29 22:21:28 2016 -0400
218
219     added semant for struct field assignment
220
221 commit c90388e0f0d0eed30323be990eb29ec089fef474
222 Author: PLT Student <axxu3795@gmail.com>
223 Date:   Wed Apr 27 21:19:46 2016 -0400
224
225     Created struct field index list
226
227 commit ef7a1054b5250ff47bd60f9fbd2a6e13396e1796
228 Author: PLT Student <axxu3795@gmail.com>
229 Date:   Wed Apr 27 03:07:09 2016 -0400
230
231     ltype-of.type now includes struct types so structs can be allocated
232
233 commit e1b6f98760055b9f39c4f0a03606f06d94c2fc8b
234 Author: PLT Student <axxu3795@gmail.com>
235 Date:   Tue Apr 26 18:48:27 2016 -0400
236
237     Cleaned up some warnings, still not sure what 42 is
238
239 commit 24ec2afb38e4bad5f7684ef663aa6d6993116dce
240 Author: PLT Student <axxu3795@gmail.com>
241 Date:   Mon Apr 25 13:38:02 2016 -0400
242
243     Working error checking for struct
244
245 commit 95a5222e09300e7f5037e22c78bd4282cba9929f
246 Author: Kyle Lee <kylelee.contact@gmail.com>
247 Date:   Mon Apr 25 13:01:23 2016 -0400
248
249     Added struct tests
250

```

```

251 commit 995258d61bd5f0db54369a7fc65dd6f188e6d415
252 Author: Kyle Lee <kylelee.contact@gmail.com>
253 Date: Mon Apr 25 12:57:47 2016 -0400
254
255     Working struct semant (throws not found exception)
256
257 commit 037886b737edccbe231b780897b7b31e67e36046
258 Author: Kyle Lee <kylelee.contact@gmail.com>
259 Date: Sat Apr 23 19:10:56 2016 -0400
260
261     match struct compiles
262
263 commit 6fa5581d2255d28b23d83efafd00b0a868b96740
264 Author: Kyle Lee <kylelee.contact@gmail.com>
265 Date: Sat Apr 23 18:49:31 2016 -0400
266
267     added broken struct accessor method
268
269 commit e91042b38c3dea600325c9239ddd42b7cbfebf6a
270 Author: PLT Student <axxu3795@gmail.com>
271 Date: Sat Apr 23 17:47:47 2016 -0400
272
273     Adding check_access, still need to match inside
274
275 commit 3940c80078342f00879360619d0d5f5ad0balc57
276 Author: Kyle Lee <kylelee.contact@gmail.com>
277 Date: Sat Apr 23 17:09:42 2016 -0400
278
279     Prepared to start adding structs to semant.
280
281 commit 450a12b335d46566822e314cbe3030fdc240a17c
282 Author: PLT Student <axxu3795@gmail.com>
283 Date: Sat Apr 23 16:26:47 2016 -0400
284
285     Gave struct types a string to hold for struct type name
286
287 commit e870131767f7edd529e0a3fbb2b1e9a3ff366bdc
288 Merge: a37ba16 38d78d3
289 Author: Amarto <aar2160@columbia.edu>
290 Date: Tue Apr 19 23:40:52 2016 -0400
291
292     Merge pull request #7 from DemocritusLang/change.syntax.order
293
294     Change syntax order with tests
295
296 commit ca8356e47677421467fad358a65bbd16809b4b37
297 Author: PLT Student <axxu3795@gmail.com>
298 Date: Tue Apr 19 21:49:46 2016 -0400
299
300     Added dot operator syntax as a binop
301
302 commit 38d78d3708f6dd5058345b5de776ae035f123240
303 Author: Emily Pakulski <enp2111@columbia.edu>
304 Date: Mon Apr 18 00:23:08 2016 -0400
305
306     Fixed bad string tests.

```

```

307
308 commit 9523521d6768e94f504ff983alde4738870f897
309 Author: PLT Student <axxu3795@gmail.com>
310 Date: Mon Apr 18 00:01:30 2016 -0400
311
312 I forgot to make clean the last commit b/c i'm dumb
313
314 commit b699bb84863eb86e006f95e82f687f3e367586de
315 Author: PLT Student <axxu3795@gmail.com>
316 Date: Sun Apr 17 23:58:44 2016 -0400
317
318 Compiles with the third struct list
319
320 commit 34076bdcc2f6a519691555482261913623bfd97d
321 Author: Kyle Lee <kylelee.contact@gmail.com>
322 Date: Sun Apr 17 23:36:54 2016 -0400
323
324 Initial addition of struct to parsing
325
326 commit d2221587f8c81155a1ca8f9e2ba50b0a83a89684
327 Author: Amarto <aar2160@columbia.edu>
328 Date: Sun Apr 17 23:36:06 2016 -0400
329
330 Fixed test-helloworld-assign declaration order
331
332 commit 12301820c5bbd32b55b29dfbd5a99068e62ee6b5
333 Author: Emily Pakulski <enp2111@columbia.edu>
334 Date: Sun Apr 17 23:14:02 2016 -0400
335
336 Changed tests to add let keyword.
337
338 commit 3a626ec31e042cfa3bcb8fc5410dd666fael2bea
339 Author: Amarto <aar2160@columbia.edu>
340 Date: Wed Apr 13 01:45:56 2016 -0400
341
342 Changed parser and scanner with LET keyword. Still working on tests
343
344 commit c6ecb302808b192e6e6f51360537556119a867ec
345 Author: Amarto <aar2160@columbia.edu>
346 Date: Tue Mar 15 00:33:01 2016 -0400
347
348 Temp commit — tried to change variable order but got SR error.
349
350 commit a37ba16c593be6be0d9980aec71b1d2b93eaf69e
351 Author: Kyle Lee <kylelee.contact@gmail.com>
352 Date: Mon Apr 11 13:05:53 2016 -0400
353
354 Added tentative install instructions (needs testing on Ubuntu 14.x and before)
355
356 commit 605b8bdlf6b6a1612d588ce0c5a52f107292d609
357 Merge: caa0380 0f67850
358 Author: Amy Xin Xu <axxu3795@gmail.com>
359 Date: Tue Apr 5 18:36:38 2016 -0400
360
361 Merge pull request #6 from DemocritusLang/strings_2
362

```



```

363     HelloWorld checkpoint!
364     :pizza:
365
366 commit 0f678507385ebacba0c05a41eac72ada0d9df015
367 Author: Kyle Lee <kylelee.contact@gmail.com>
368 Date:   Tue Apr 5 18:29:24 2016 -0400
369
370     fixed failing function call test (semant only checks for print())
371
372 commit 22a7da1415ce11b8dbb75ea0a8766e50a48bac38
373 Author: PLT Student <axxu3795@gmail.com>
374 Date:   Tue Apr 5 18:26:25 2016 -0400
375
376     Fixed missing printb
377
378 commit cb55b59f5981f69e28a347525e35ef1071020447
379 Author: = <kylelee.contact@gmail.com>
380 Date:   Tue Apr 5 18:12:31 2016 -0400
381
382     Fixed tarball makefile builder for helloworld
383
384 commit 67ddab48071f203f447caf80f6906870af14e510
385 Author: = <kylelee.contact@gmail.com>
386 Date:   Tue Apr 5 18:02:03 2016 -0400
387
388     Added test case for string assignment and printing.
389
390 commit daceabeb68dbd482000334162a0f797788554616
391 Author: Amarto <aar2160@columbia.edu>
392 Date:   Mon Apr 4 13:49:14 2016 -0400
393
394     Print hello world working. Tests that use printb() are failing, because i had to
395         remove it from semant.ml temporarily.
396
397 commit f121b87bdc8a6afac89d6374a9eb2705529d123
398 Author: Amarto <aar2160@columbia.edu>
399 Date:   Mon Apr 4 02:22:46 2016 -0400
400
401     Compiling, but not passing tests.
402
403 commit 93833564906d28a36e6cd8303241e69a764ac2f0
404 Author: Emily Pakulski <enp2111@columbia.edu>
405 Date:   Sun Apr 3 18:11:16 2016 -0400
406
407     Tried moving strings to types...
408
409 commit de2ba3fd50270ec7894abbb7f8c9a1bb0efd8ca3
410 Author: Emily Pakulski <enp2111@columbia.edu>
411 Date:   Sun Apr 3 17:56:06 2016 -0400
412
413     Added partial implementation of string literal.
414
415 commit 2fd8d9408fc9ceb78e03d3ebdc2195aee4ad7403
416 Author: Emily Pakulski <enp2111@columbia.edu>
417 Date:   Sun Apr 3 16:53:52 2016 -0400

```

```

418     Added test and function for helloworld. Need string literal implementation.
419
420 commit 0471926a19d8626bab3140b6a12abcf588288620
421 Author: Emily Pakulski <enp2111@columbia.edu>
422 Date:    Sun Apr 3 16:27:19 2016 -0400
423
424     Changed Edwards' print to be print_int to avoid confusion with our print
         implementation.
425
426 commit caa0380101c0ac9f657eb626b1930c5ca72bfd5e
427 Author: Emily Pakulski <enp2111@columbia.edu>
428 Date:    Mon Mar 14 22:47:26 2016 -0400
429
430     Added function keyword to function declarations.
431
432 commit de3f696465ef9a95a737282d1affa7d1d812cad0
433 Author: Amarto Rajaram <aar2160@columbia.edu>
434 Date:    Mon Mar 14 21:58:20 2016 -0400
435
436     Removed while keyword; replaced functionality with for.
437
438 commit d0829835a72243f858bbe2426ab81b04792ed883
439 Author: Amarto Rajaram <amarto.rajaram@gmail.com>
440 Date:    Mon Mar 14 21:03:09 2016 -0400
441
442     Added Edwards' tests back in.
443
444 commit 798f67d953e965dae1282e679f6b0e5373982058
445 Author: Emily Pakulski <enp2111@columbia.edu>
446 Date:    Fri Feb 26 11:45:05 2016 -0500
447
448     Edwards' MicroC code with our README.

```

build-malloc-attempt branch

```

1 commit 232belf27fb7ffa7110e5184c6016b81e2da94ff
2 Author: PLT Student <axxu3795@gmail.com>
3 Date:    Wed May 11 03:45:50 2016 -0400
4
5     Malloc and simple linked lists working
6
7     Fixed shift reduce errors
8
9     temp commit
10
11     CASTING AND MALLOC WORK
12
13     Cleaned up warnings
14
15     Cleaned up codegen warnings
16
17     Fixed (*a). dotops and halfassed addnode
18
19     Half way linked lists
20
21     Linked lists with add and print_list functions

```

```

22
23 commit dbb06cc9f0f2f608583946158d48d8d841d8dc62
24 Author: Emily Pakulski <enp2111@columbia.edu>
25 Date: Wed May 11 13:30:05 2016 -0400
26
27 Added check in tester for whether code was already compiled.
28
29 commit 36aee48817bdeec1553b1c16963e5abcc7aaaa6b
30 Merge: 0371f67 2402f74
31 Author: Amarto <aar2160@columbia.edu>
32 Date: Wed May 11 06:10:49 2016 -0400
33
34 Merge pull request #21 from DemocritusLang/sockets_finished
35
36 Sockets finished
37
38 commit 2402f740bec59d6ed4912b2ee302d9b9b7bb5480
39 Author: Amarto <aar2160@columbia.edu>
40 Date: Wed May 11 05:45:05 2016 -0400
41
42 Added free(), execl wrapper, and corrected output reference for socket test.
    Changed tests to use free() after malloc. Refactored the weblink downloading
    method to only take one param so it matches the signature for a thread function
43
44 commit 170645297d4578f9551f690b08b82718beebf033
45 Author: Emily Pakulski <enp2111@columbia.edu>
46 Date: Wed May 11 01:44:23 2016 -0400
47
48 Changed up get request impl a bit.
49
50 commit 5226ad21d850b6622e09388a4337d0b515123f76
51 Author: Amarto <aar2160@columbia.edu>
52 Date: Tue May 10 17:46:11 2016 -0400
53
54 Added basic socket impl and loading files. Need to handle tests
55
56 commit 0371f67bac394c08ae72cb372491fd3264839f65
57 Merge: a6ce096 a784099
58 Author: Amy Xin Xu <axxu3795@gmail.com>
59 Date: Wed May 11 02:57:57 2016 -0400
60
61 Merge pull request #20 from DemocritusLang/add_float_and_mod
62
63 Added modulo and floats
64
65 commit a78409901632a6f70e889fee8f5ccff2bfe12989
66 Author: PLT Student <axxu3795@gmail.com>
67 Date: Tue May 10 23:26:55 2016 -0400
68
69 Added modulo and floats
70
71 Mod done
72
73 Working on floats
74
75 fixed floating print issue

```

```

76
77     Working floats
78
79     Added floats in struct test
80
81     commit a6ce096863beb92e5b5251b0802613edab31cc76
82     Author: Emily <ohEmily@users.noreply.github.com>
83     Date:   Tue May 10 23:09:23 2016 -0400
84
85     Added sleep function and test. (#18)
86
87     commit ff330840be2c13aeed6f2a6d87a69ce153f29421
88     Merge: a63b40f fdcadf4
89     Author: Amy Xin Xu <axxu3795@gmail.com>
90     Date:   Tue May 10 15:28:57 2016 -0400
91
92     Merge pull request #17 from DemocritusLang/add_pointers
93
94     Pointers done
95
96     commit fdcadf4fa032354c8a9ec96f41cecb76b94e66f0
97     Author: PLT Student <axxu3795@gmail.com>
98     Date:   Tue May 10 02:02:43 2016 -0400
99
100    Pointers done
101
102    Dereference syntax there, need to clean warning
103
104    Added ref op and semantic checking
105
106    Working pointers for ints, need to test rest
107
108    Modified test-pointer-int.dem for clarity and wrote test-pointer-bool, passing
109
110    Added single and multilevel struct tests, passing
111
112    Linkedlist test not working
113
114    Added hypothetical linkedlist tests (not working)
115
116    Linked of list proof of concept
117
118    Changed type* to *type to reflect Go syntax
119
120    commit a63b40fb618149ec65e57ea3ac986bdccf9f4ac4
121    Author: Kyle Lee <kylelee.contact@gmail.com>
122    Date:   Tue May 10 01:11:44 2016 -0400
123
124    Added singleline comments
125
126    commit 901814668aa2d7513fe74b45fa4390d82635ac01
127    Author: Amarto <aar2160@columbia.edu>
128    Date:   Tue May 10 00:46:58 2016 -0400
129
130    Fixed void pointer notation to match Go syntax, fixed test
131

```

```

132 commit 0ed94930f8362cb6eb322ec6a9570043660aabb5
133 Merge: b662f67 d9b467b
134 Author: Amy Xin Xu <axxu3795@gmail.com>
135 Date: Tue May 10 00:43:18 2016 -0400
136
137 Merge pull request #15 from DemocritusLang/clean.nested.structs
138
139 Working nested structs
140
141 commit d9b467b80ce152696875d6ec1d3d2f1ec6ea77e6
142 Author: PLT Student <axxu3795@gmail.com>
143 Date: Mon May 9 22:47:07 2016 -0400
144
145 Working nested structs
146
147 Added nested struct test
148
149 Fixed mistyped identifier
150
151 nested structs working
152
153 Fixed typo in test-structs-nested.out and added another test
154
155 Edited test to be more informative of functionality
156
157 test-struct-nested1
158
159 commit b662f676ae12fbb27eedaf5af6ae990d76f423bc
160 Author: Emily Pakulski <enp2111@columbia.edu>
161 Date: Mon May 9 20:34:05 2016 -0400
162
163 Finished file I/O. lseek also implemented.
164
165 commit 84c1fc11bc2a2e59b8fec9d68937db8205f1b5d9
166 Author: Amarto <aar2160@columbia.edu>
167 Date: Sun May 8 20:30:58 2016 -0400
168
169 Added malloc and started file I/O.
170
171 commit 8b3944051cfde07be958214aae56bf47988fb803
172 Author: Emily <ohEmily@users.noreply.github.com>
173 Date: Mon May 9 11:15:23 2016 -0400
174
175 Updated all instances of MicroC to Democritus and added 'make all' target (#12)
176
177 * Changed MicroC -> Democritus and added make all target.
178
179 * Changed file extension for democrituslang files from .mc to .dem.
180
181 commit ed27ce5f8a31a740f3eb0e5ad3ff3cfcf7a838f9
182 Author: Amarto <aar2160@columbia.edu>
183 Date: Sun May 8 19:31:43 2016 -0400
184
185 Fixed warnings resulting from merge
186
187 commit c6cbdf15fd8854a02fb695fa9aa41b50966431a7

```

```

188 Author: Amarto <aar2160@columbia.edu>
189 Date: Sat Apr 30 15:16:57 2016 -0400
190
191 Added multithreading and void pointers, and added calling bound C functions
192 Added declaration of thread() function to codegen. Everything compiles
193
194 Added basic threads checking to semant.ml. Need to wait until arguments for
    pthread are passed in
195
196 Working on codegen.ml, but getting compiler warning. Working on threading test,
    but need NULL keyword?
197
198 Added tests for threading and modified codegen and semant
199
200 Baby steps. Still not working. (temp commit).
201
202 Oops. But still not working.
203
204 Fixed some things in test case. Pretty sure function name should be passed in as a
    string. (temp commit.)
205
206 Temp commit. More debug info. Maybe fixed some bugs but same error.
207
208 temp commit - fixed compiler warning but old tests are failing
209
210 Fixed old tests, fixed compiler warning
211
212 Added correct(?) invocation of args in thread_init. Still not-found exception
213
214 It was failing to match on [e]. Changed to e, and now it's giving a broken module
    error: params don't match
215
216 Still not working (broken module) but now using lookup_function and pattern
    matching to remove option
217
218 Added a void ptr type for thread (kinda hacky for now but it's for testing threads
    ). Also it's now finding the function from the string name
219
220 Added thread testing script
221
222 THREADS NOW WORK IN SCRIPT!!!
223
224 Passing threading test
225
226 Fixed compiler warnings from pattern matching in codegen
227
228 commit bca9388fld5b7011fde7461b2f1055562f1c7561
229 Author: PLT Student <axxu3795@gmail.com>
230 Date: Thu May 5 00:59:11 2016 -0400
231
232 Clean compilation without warnings
233
234 commit ecf06799e7b2a68a08ef9603a4b9eacfdcf7b3ce
235 Author: Kyle Lee <kylelee.contact@gmail.com>
236 Date: Wed May 4 13:28:00 2016 -0400
237

```

```

238     Removed codegen warnings, and some semant warnings
239
240 commit 08a4e105a2267891a38e76b4f280a4631bbe3413
241 Author: Kyle Lee <kylelee.contact@gmail.com>
242 Date:   Wed May 4 00:54:52 2016 -0400
243
244     fixed struct tests for let format
245
246 commit 152ab95f7e0c087cc914c0a5c2b176951b77a1d3
247 Merge: b4f812b 116094b
248 Author: Kyle Lee <kylelee.contact@gmail.com>
249 Date:   Wed May 4 00:36:07 2016 -0400
250
251     Merge add.structs
252
253 commit 116094b8ee508fd191c6d793cb14b9b5d6955c2a
254 Author: PLT Student <axxu3795@gmail.com>
255 Date:   Sat Apr 30 20:57:04 2016 -0400
256
257     Semantic checking to disallow circularly dependent structs
258
259 commit 490aa96cafcf6e93d9a5b981f37244cc5b0cb6c6
260 Author: PLT Student <axxu3795@gmail.com>
261 Date:   Sat Apr 30 17:03:42 2016 -0400
262
263     Fixed the stack overflow problem and updated tests
264
265 commit 41cb475f79c1d6baf22baf68b02219be8a9a49b2
266 Author: Kyle Lee <kylelee.contact@gmail.com>
267 Date:   Sat Apr 30 14:19:54 2016 -0400
268
269     struct access works (messy)
270
271 commit eef6eb9fc000a844957f73dde0a56980a3b44ee0
272 Author: Kyle Lee <kylelee.contact@gmail.com>
273 Date:   Sat Apr 30 14:14:10 2016 -0400
274
275     Structs reach llvm failure point. need to clean up exception catching and matches.
276
277 commit b4f812b37b0788d6f4a6d09495f41bc1515488ec
278 Author: Emily Pakulski <enp2111@columbia.edu>
279 Date:   Sat Apr 30 13:29:07 2016 -0400
280
281     Flattened built-in function declarations so we don't need extra variables.
282
283 commit 2c0b9cba13e157863e8dcc7fb2f3346a6262c5b1
284 Author: PLT Student <axxu3795@gmail.com>
285 Date:   Sat Apr 30 02:06:26 2016 -0400
286
287     changed to named structs
288
289 commit aa095775c0b41e8776214758f2af8d31e142d1ea
290 Author: Kyle Lee <kylelee.contact@gmail.com>
291 Date:   Fri Apr 29 22:21:28 2016 -0400
292
293     added semant for struct field assignment

```

```

294
295 commit c90388e0f0d0eed30323be990eb29ec089fef474
296 Author: PLT Student <axxu3795@gmail.com>
297 Date: Wed Apr 27 21:19:46 2016 -0400
298
299 Created struct field index list
300
301 commit ef7a1054b5250ff47bd60f9fbd2a6e13396e1796
302 Author: PLT Student <axxu3795@gmail.com>
303 Date: Wed Apr 27 03:07:09 2016 -0400
304
305 ltype_of.type now includes struct types so structs can be allocated
306
307 commit e1b6f98760055b9f39c4f0a03606f06d94c2fc8b
308 Author: PLT Student <axxu3795@gmail.com>
309 Date: Tue Apr 26 18:48:27 2016 -0400
310
311 Cleaned up some warnings, still not sure what 42 is
312
313 commit 24ec2afb38e4bad5f7684ef663aa6d6993116dce
314 Author: PLT Student <axxu3795@gmail.com>
315 Date: Mon Apr 25 13:38:02 2016 -0400
316
317 Working error checking for struct
318
319 commit 95a5222e09300e7f5037e22c78bd4282cba9929f
320 Author: Kyle Lee <kylelee.contact@gmail.com>
321 Date: Mon Apr 25 13:01:23 2016 -0400
322
323 Added struct tests
324
325 commit 995258d61bd5f0db54369a7fc65dd6f188e6d415
326 Author: Kyle Lee <kylelee.contact@gmail.com>
327 Date: Mon Apr 25 12:57:47 2016 -0400
328
329 Working struct semant (throws not found exception)
330
331 commit 037886b737edccbe231b780897b7b31e67e36046
332 Author: Kyle Lee <kylelee.contact@gmail.com>
333 Date: Sat Apr 23 19:10:56 2016 -0400
334
335 match struct compiles
336
337 commit 6fa5581d2255d28b23d83efafd00b0a868b96740
338 Author: Kyle Lee <kylelee.contact@gmail.com>
339 Date: Sat Apr 23 18:49:31 2016 -0400
340
341 added broken struct accessor method
342
343 commit e91042b38c3dea600325c9239ddd42b7cbfebf6a
344 Author: PLT Student <axxu3795@gmail.com>
345 Date: Sat Apr 23 17:47:47 2016 -0400
346
347 Adding check_access, still need to match inside
348
349 commit 3940c80078342f00879360619d0d5f5ad0balc57

```



```

350 Author: Kyle Lee <kylelee.contact@gmail.com>
351 Date: Sat Apr 23 17:09:42 2016 -0400
352
353     Prepared to start adding structs to semant.
354
355 commit 450a12b335d46566822e314cbe3030fdc240a17c
356 Author: PLT Student <axxu3795@gmail.com>
357 Date: Sat Apr 23 16:26:47 2016 -0400
358
359     Gave struct types a string to hold for struct type name
360
361 commit e870131767f7edd529e0a3fbb2b1e9a3ff366bdc
362 Merge: a37ba16 38d78d3
363 Author: Amarto <aar2160@columbia.edu>
364 Date: Tue Apr 19 23:40:52 2016 -0400
365
366     Merge pull request #7 from DemocritusLang/change_syntax_order
367
368     Change syntax order with tests
369
370 commit ca8356e47677421467fad358a65bbd16809b4b37
371 Author: PLT Student <axxu3795@gmail.com>
372 Date: Tue Apr 19 21:49:46 2016 -0400
373
374     Added dot operator syntax as a binop
375
376 commit 38d78d3708f6dd5058345b5de776ae035f123240
377 Author: Emily Pakulski <enp2111@columbia.edu>
378 Date: Mon Apr 18 00:23:08 2016 -0400
379
380     Fixed bad string tests.
381
382 commit 9523521d6768e94f504ff983alde4738870f897
383 Author: PLT Student <axxu3795@gmail.com>
384 Date: Mon Apr 18 00:01:30 2016 -0400
385
386     I forgot to make clean the last commit b/c i'm dumb
387
388 commit b699bb84863eb86e006f95e82f687f3e367586de
389 Author: PLT Student <axxu3795@gmail.com>
390 Date: Sun Apr 17 23:58:44 2016 -0400
391
392     Compiles with the third struct list
393
394 commit 34076bdcc2f6a519691555482261913623bfd97d
395 Author: Kyle Lee <kylelee.contact@gmail.com>
396 Date: Sun Apr 17 23:36:54 2016 -0400
397
398     Initial addition of struct to parsing
399
400 commit d2221587f8c81155a1ca8f9e2ba50b0a83a89684
401 Author: Amarto <aar2160@columbia.edu>
402 Date: Sun Apr 17 23:36:06 2016 -0400
403
404     Fixed test-helloworld-assign declaration order
405

```

```

406 commit 12301820c5bbd32b55b29dfbd5a99068e62ee6b5
407 Author: Emily Pakulski <enp2111@columbia.edu>
408 Date: Sun Apr 17 23:14:02 2016 -0400
409
410     Changed tests to add let keyword.
411
412 commit 3a626ec31e042cfa3bcb8fc5410dd666fael2bea
413 Author: Amarto <aar2160@columbia.edu>
414 Date: Wed Apr 13 01:45:56 2016 -0400
415
416     Changed parser and scanner with LET keyword. Still working on tests
417
418 commit c6ecb302808b192e6e6f51360537556119a867ec
419 Author: Amarto <aar2160@columbia.edu>
420 Date: Tue Mar 15 00:33:01 2016 -0400
421
422     Temp commit — tried to change variable order but got SR error.
423
424 commit a37ba16c593be6be0d9980aec71b1d2b93eaf69e
425 Author: Kyle Lee <kylelee.contact@gmail.com>
426 Date: Mon Apr 11 13:05:53 2016 -0400
427
428     Added tentative install instructions (needs testing on Ubuntu 14.x and before)
429
430 commit 605b8bdlf6b6a1612d588ce0c5a52f107292d609
431 Merge: caa0380 0f67850
432 Author: Amy Xin Xu <axxu3795@gmail.com>
433 Date: Tue Apr 5 18:36:38 2016 -0400
434
435     Merge pull request #6 from DemocritusLang/strings.2
436
437     HelloWorld checkpoint!
438     :pizza:
439
440 commit 0f678507385ebacba0c05a41eac72ada0d9df015
441 Author: Kyle Lee <kylelee.contact@gmail.com>
442 Date: Tue Apr 5 18:29:24 2016 -0400
443
444     fixed failing function call test (semant only checks for print())
445
446 commit 22a7da1415ce11b8dbb75ea0a8766e50a48bac38
447 Author: PLT Student <axxu3795@gmail.com>
448 Date: Tue Apr 5 18:26:25 2016 -0400
449
450     Fixed missing printb
451
452 commit cb55b59f5981f69e28a347525e35ef1071020447
453 Author: = <kylelee.contact@gmail.com>
454 Date: Tue Apr 5 18:12:31 2016 -0400
455
456     Fixed tarball makefile builder for helloworld
457
458 commit 67ddab48071f203f447caf80f6906870af14e510
459 Author: = <kylelee.contact@gmail.com>
460 Date: Tue Apr 5 18:02:03 2016 -0400
461

```

```

462     Added test case for string assignment and printing.
463
464 commit daceabeb68dbd482000334162a0f797788554616
465 Author: Amarto <aar2160@columbia.edu>
466 Date:   Mon Apr 4 13:49:14 2016 -0400
467
468     Print hello world working. Tests that use printb() are failing, because i had to
         remove it from semant.ml temporarily.
469
470 commit f121b87bdc8a6afac89d6374a9eb2705529d123
471 Author: Amarto <aar2160@columbia.edu>
472 Date:   Mon Apr 4 02:22:46 2016 -0400
473
474     Compiling, but not passing tests.
475
476 commit 93833564906d28a36e6cd8303241e69a764ac2f0
477 Author: Emily Pakulski <enp2111@columbia.edu>
478 Date:   Sun Apr 3 18:11:16 2016 -0400
479
480     Tried moving strings to types...
481
482 commit de2ba3fd50270ec7894abbb7f8c9a1bb0efd8ca3
483 Author: Emily Pakulski <enp2111@columbia.edu>
484 Date:   Sun Apr 3 17:56:06 2016 -0400
485
486     Added partial implementation of string literal.
487
488 commit 2fd8d9408fc9ceb78e03d3ebdc2195aee4ad7403
489 Author: Emily Pakulski <enp2111@columbia.edu>
490 Date:   Sun Apr 3 16:53:52 2016 -0400
491
492     Added test and function for helloworld. Need string literal implementation.
493
494 commit 0471926a19d8626bab3140b6a12abcf588288620
495 Author: Emily Pakulski <enp2111@columbia.edu>
496 Date:   Sun Apr 3 16:27:19 2016 -0400
497
498     Changed Edwards' print to be print_int to avoid confusion with our print
         implementation.
499
500 commit caa0380101c0ac9f657eb626b1930c5ca72bfd5e
501 Author: Emily Pakulski <enp2111@columbia.edu>
502 Date:   Mon Mar 14 22:47:26 2016 -0400
503
504     Added function keyword to function declarations.
505
506 commit de3f696465ef9a95a737282d1affa7d1d812cad0
507 Author: Amarto Rajaram <aar2160@columbia.edu>
508 Date:   Mon Mar 14 21:58:20 2016 -0400
509
510     Removed while keyword; replaced functionality with for.
511
512 commit d0829835a72243f858bbe2426ab81b04792ed883
513 Author: Amarto Rajaram <amarto.rajaram@gmail.com>
514 Date:   Mon Mar 14 21:03:09 2016 -0400
515

```

516 Added Edwards' tests back in.
517
518 commit 798f67d953e965dae1282e679f6b0e5373982058
519 Author: Emily Pakulski <enp2111@columbia.edu>
520 Date: Fri Feb 26 11:45:05 2016 -0500
521
522 Edwards' MicroC code with our README.

Final Report

1 commit 7c0f7a5ed8540cf9cb43f628a03db1c82e626dee
2 Author: Emily Pakulski <enp2111@columbia.edu>
3 Date: Wed May 11 21:48:51 2016 -0400
4
5 Making including easier.
6
7 commit 197d46b63145a1aef3b3675c19fb9231549ee74b
8 Author: Emily Pakulski <enp2111@columbia.edu>
9 Date: Wed May 11 21:41:01 2016 -0400
10
11 Added script that should allow us to dynamically generate source code files.
12
13 commit f62eb16d6f1faa62b2ddd1ba9eebc15d378b29af
14 Author: Emily Pakulski <enp2111@columbia.edu>
15 Date: Wed May 11 20:51:45 2016 -0400
16
17 Tiny changes.
18
19 commit 89bb5c883803ea47bf5c1ac32ba671f92c4a260c
20 Author: Kyle Lee <kylelee.contact@gmail.com>
21 Date: Wed May 11 16:28:39 2016 -0400
22
23 Did testing chapter
24
25 commit 73839a25ced0849181664738054bcb79ff67230e
26 Author: Kyle Lee <kylelee.contact@gmail.com>
27 Date: Wed May 11 15:23:16 2016 -0400
28
29 finished up LRM
30
31 commit 1bf1b7f84d1a54e1c9fa34a4dd9df3de5afecdd7
32 Author: Kyle Lee <kylelee.contact@gmail.com>
33 Date: Wed May 11 14:48:34 2016 -0400
34
35 LRM basically fully done
36
37 commit d1d930bc3c77ab377132ddc3c75b37bc1d3425d7
38 Author: Kyle Lee <kylelee.contact@gmail.com>
39 Date: Wed May 11 03:46:38 2016 -0400
40
41 finished expressions and operators
42
43 commit a946ec6f6af1b9782ba5c1c895cb4fa1c0d74b1d
44 Author: Kyle Lee <kylelee.contact@gmail.com>
45 Date: Wed May 11 01:16:41 2016 -0400
46

47 started updating LRM; fixed grammars and such
48
49 commit 6aa26ab70e7e8f52bf3579f373638796750759b9
50 Author: Kyle Lee <kylelee.contact@gmail.com>
51 Date: Tue May 10 03:24:54 2016 -0400
52
53 added plan, fixed some rendering issues
54
55 commit 95add631e41738269f470110aa807994286e1586
56 Author: Kyle Lee <kylelee.contact@gmail.com>
57 Date: Tue May 10 00:48:11 2016 -0400
58
59 Added more tutorial stuff, started architecture and other sections
60
61 commit 4f72336d4ef7f06f536d4f9c1febcdcd3dc55d56c
62 Author: Kyle Lee <kylelee.contact@gmail.com>
63 Date: Mon May 9 03:25:32 2016 -0400
64
65 added more introduction page info
66
67 commit 7468774d0ac3345921d545057fcf6214433824ef
68 Author: Kyle Lee <kylelee.contact@gmail.com>
69 Date: Sun May 8 20:04:33 2016 -0400
70
71 Added tutorial section; added sections to chapters
72
73 commit e016dac39ce5ac68ff4f7862eee25b514a637a78
74 Author: Kyle Lee <kylelee.contact@gmail.com>
75 Date: Fri May 6 19:43:37 2016 -0400
76
77 initial commit for final report; added more chapters
78
79 commit 57517095899f2a1a4a30bb39c94339d5f5e5cfc6
80 Author: kyle—lee <kylelee.contact@gmail.com>
81 Date: Fri May 6 18:41:55 2016 -0400
82
83 Initial commit

5. Architecture Overview

Democritus' compiler is built off of Professor Stephen Edwards' `MicroC` compiler.

5.1 Compiler Overview

Several files make up the source code of the compiler. These include:

- `scanner.mll`: the OCamllex scanner.
- `ast.ml`: the abstract syntax tree, summarizing the overall structure of a Democritus program.
- `parser.mly`: the Ocamlyacc parser. Tokens from the scanner are parsed into the abstract syntax tree in the parser.
- `semant.ml`: the semantic analyzer.
- `codegen.ml`: the LLVM IR code generator.
- `democritus.ml`: the overarching OCaml program that calls the four main steps of the compiler.
- `bindings.c`: a C file that provides facilitates low-level operations that interact with the OS through C functions, such as for threads, which is then compiled to LLVM bytecode.

The Scanner

The scanner is simply a text scanner that parses text into various tokens, to then be interpreted by the parser. The regular expressions used by the scanner are listed in the language reference chapter.

The Parser

The parser is a token scanner that converts the tokens read into a valid abstract syntax tree of the program. If the program follows valid syntax, it will be parsed accordingly. Otherwise, compilation of code will yield a parse error. The structure of the program is as follows:

The Semantic Analyzer

The semantic analyzer checks the consistency and correctness of user programs. For example, it will check whether variables are defined within a scope, whether types of expressions match their uses in definitions and function calls, and whether structs are used correctly (a large modification we made was semantic checking for circular struct definitions).

The Code Generator

The code generator then takes in a definition of a program and builds the equivalent LLVM IR.

6. Testing

As with any software project, extensive testing was required to verify that all the features being implemented were working properly.

6.1 Integration Testing

Development and Testing Process

Development of new features required making them pass through the scanner, parser, semantic analyzer, and then code generation, in that order. When envisioning or developing a new feature, the testing process would proceed as follows:

1. Write example code implementing and utilizing the desired feature. (E.g. writing a struct definition in a new test file).
2. Modify the scanner (if needed) to read new tokens required by the new feature.
3. Modify the parser (usually needed) to change the grammar of the program to accept the new feature and pass necessary information (E.g. struct field names) to the semantic analyzer.
4. Modify the example code and test it so that only the ‘correct’ implementation of the feature passes the parser. Modify the scanner and parser until this step passes.
5. Modify the semantic analyzer so that it detects possible semantic issues that could arise from utilization of the new feature (E.g. accessing an undefined field in a struct or an undefined struct).
6. Modify the example code and test it so that only the ‘correct’ implementation passes the semantic analyzer; try testing multiple cases that should cause the analyzer to raise an error. Modify the semantic analyzer until this step passes.
7. Modify the code generator so that it generates the appropriate LLVM IR representing your new feature (E.g., allocating the correct amount of memory for new structs, building a map of struct field indexes, calling `LLVM.build_struct_gep`, etc.).
8. Modify your example code to utilize your feature and produce some visible effect or output (E.g. assigning a struct field, doing arithmetic on it, then printing it).
9. Test the code and ensure that running the program produces the expected output or effect; continue working on code generation until it does.

The process of writing test code, compiling it, and observing its output after being run as LLVM IR was the integration testing method that the Democritus team utilized throughout development. It helped ensure that whole features were working properly, and that the language, built up from multiple features, was still functioning correctly. Integration testing was done on all new features added to the language, as well as the existing ones from MicroC (such as basic variable assignment, conditional iteration, etc).

Aside: Unit Testing

Unit testing was not overly utilized in this development process, besides for testing to ensure that new features could pass certain layers of the compiler while working towards a passing integration test. This is because unit tests can still pass, while whole features lose vertical integration in the process of building up a compiler. This is because new features may often conflict with each other and the successful introduction of one feature could very well mean the breaking of another. This leads us to the test suite and automated regression testing.

6.2 The Test Suite and Automated Regression Testing

Democritus' test suite was built upon MicroC's automated regression testing package. Within the `tests` directory, there are dozens of integration test files for various language features as well as their expected `stdout` output. Additionally, there are several 'fail' tests used for showing invalid Democritus code as well as their expected error outputs.

The automated regression testing suite was used to quickly test all major language features by compiling each test, writing the error thrown by compilation (if it was a failure) or output of running the LLVM file (if compilation was a success) to a temporary file, and comparing that output to the expected output of each test with `diff`. The automated test was a shell script, invoked with `./testall.sh` in the Democritus root directory.

The test suite was used frequently throughout development; while developing new features, team members would utilize the test suite to ensure that all major features of the language were still working. If a certain test in the suite failed, more verbose information about the test's failure could be accessed in the `testall.log` file generated by the testing suite. The automated regression testing was crucial in ensuring that the language stayed consistent and working, and that our master branch remained 'updated' and error-free.

7. Lessons Learned

7.1 Amy

Trying to force new code to match legacy code can be more effort than it's worth. It's always okay to branch and attempt a larger rewrite if it will make everyone's lives easier. Also, be sure to understand your own syntax when writing tests.

7.2 Emily

Remote teamwork can be tough. Writing tests that guarantee no regressions is surprisingly difficult, especially when testing against remote files.

7.3 Amarto

Debugging a compiler is like playing whack-a-mole – it's much easier to write a script to isolate the action you're trying to debug, and then gradually build it back into the compiler.

7.4 Kyle

In a team, try to play your strengths and figure out where you can help most effectively. If you think you can do something well or more efficiently than someone else, try to do it and save time - same thing works the other way (if pressed for time, let someone who knows how to do it manage it)

8. Code Listing

8.1 democritus.ml

```
1 (* Democritus, adapted from MicroC by Stephen Edwards Columbia University *)
2 (* Top-level of the MicroC compiler: scan & parse the input,
3    check the resulting AST, generate LLVM IR, and dump the module *)
4
5 type action = Ast | LLVM\_IR | Compile
6
7 let \_ =
8   let action = if Array.length Sys.argv > 1 then
9     List.assoc Sys.argv.(1) [ ("-a", Ast); (* Print the AST only *)
10      ("l", LLVM\_IR); (* Generate LLVM, don't check *)
11      ("-c", Compile) ] (* Generate, check LLVM IR *)
12   else Compile in
13   let lexbuf = Lexing.from\_channel stdin in
14   let ast = Parser.program Scanner.token lexbuf in
15   Semant.check ast;
16   match action with
17   | Ast -> print\_string (Ast.string\_of\_program ast)
18   | LLVM\_IR -> print\_string (Llvm.string\_of\_llmodule (Codegen.translate ast))
19   | Compile -> let m = Codegen.translate ast in
20     Llvm\_analysis.assert\_valid\_module m;
21     print\_string (Llvm.string\_of\_llmodule m)
```

8.2 scanner.mll

```
22 (* Democritus, adapted from MicroC by Stephen Edwards Columbia University *)
23 (* Ocamllex scanner *)
24
25 { open Parser }
26
27 rule token = parse
28   [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
29 | "//" { comment lexbuf } (* Comments *)
30 | "/*" { multicomment lexbuf } (* Multiline comments *)
31 | '(' { LPAREN }
32 | ')' { RPAREN }
33 | '{' { LBRACE }
34 | '}' { RBRACE }
35 | ';' { SEMI }
36 | ':' { COLON }
37 | ',' { COMMA }
38 | '+' { PLUS }
```

```

39 | '-'      { MINUS }
40 | '*'      { STAR  }
41 | '%'      { MOD   }
42 | '&'      { REF   }
43 | '.'      { DOT   }
44 | '/'      { DIVIDE }
45 | '='      { ASSIGN }
46 | "=="     { EQ    }
47 | "!="     { NEQ   }
48 | '<'      { LT    }
49 | "<="     { LEQ   }
50 | ">"      { GT    }
51 | ">="     { GEQ   }
52 | "&&"     { AND   }
53 | "||"     { OR    }
54 | "!"      { NOT   }
55 | "["      { LEFTBR }
56 | "]"      { RIGHTBR }
57 | "if"     { IF    }
58 | "else"   { ELSE   }
59 | "for"    { FOR    }
60 | "return" { RETURN }
61 | "int"    { INT    }
62 | "float"  { FLOAT  }
63 | "bool"   { BOOL   }
64 | "void"   { VOID   }
65 | "true"   { TRUE   }
66 | "string" { STRTYPE }
67 | "struct" { STRUCT }
68 | "*void"  {VOIDSTAR }
69 | "false"  { FALSE  }
70 | "function" { FUNCTION }
71 | "cast"   { CAST   }
72 | "to"     { TO     }
73 | "set"    { SET    }
74 | "let"    { LET    }
75 | ['0'-'9']+['.']['0'-'9']+ as lxm { FLOATLITERAL(float\_of\_string lxm) }
76 | ['0'-'9']+ as lxm { LITERAL(int\_of\_string lxm) }
77 | ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '\_']* as lxm { ID(lxm) }
78 | '"'      { read\_string (Buffer.create 17) lexbuf }
79 | eof { EOF }
80 | \_ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
81
82 and comment = parse
83   "\n" { token lexbuf }
84   | \_ { comment lexbuf }
85
86 and multicomment = parse
87   "*/" { token lexbuf }
88   | \_ { multicomment lexbuf }
89
90 (* From: realworldocaml.org/v1/en/html/parsing-with-ocamllex-and-menhir.html *)
91 and read\_string buf =
92   parse
93   | '"'      { STRING (Buffer.contents buf) }
94   | '\\\' '/' { Buffer.add\_char buf '/'; read\_string buf lexbuf }

```

```

95 | '\\ ' '\\ ' { Buffer.add\_char buf '\\'; read\_string buf lexbuf }
96 | '\\ ' 'b' { Buffer.add\_char buf '\\b'; read\_string buf lexbuf }
97 | '\\ ' 'f' { Buffer.add\_char buf '\\012'; read\_string buf lexbuf }
98 | '\\ ' 'n' { Buffer.add\_char buf '\\n'; read\_string buf lexbuf }
99 | '\\ ' 'r' { Buffer.add\_char buf '\\r'; read\_string buf lexbuf }
100 | '\\ ' 't' { Buffer.add\_char buf '\\t'; read\_string buf lexbuf }
101 | [^ '\" '\\']+
102 | { Buffer.add\_string buf (Lexing.lexeme lexbuf);
103 |   read\_string buf lexbuf
104 | }
105 | \_ { raise (Failure("Illegal string character: " ^ Lexing.lexeme lexbuf)) }
106 | eof { raise (Failure("String is not terminated")) }

```

8.3 parser.mly

```

107 /* Democritus, adapted from MicroC by Stephen Edwards Columbia University */
108 /* Ocamlyacc parser */
109
110 %{
111 open Ast;;
112
113 let first (a,\_,\_) = a;;
114 let second (\_,b,\_) = b;;
115 let third (\_,\_,c) = c;;
116 %}
117
118 %token LEFTBR RIGHTBR
119 %token COLON SEMI LPAREN RPAREN LBRACE RBRACE COMMA
120 %token PLUS MINUS STAR DIVIDE MOD ASSIGN NOT DOT Deref REF
121 %token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
122 %token LET RETURN IF ELSE FOR INT FLOAT BOOL VOID STRTYPE FUNCTION STRUCT VOIDSTAR
123   CAST TO SET
124 %token <string> STRING
125 %token <float> FLOATLITERAL
126 %token <int> LITERAL
127 %token <string> ID
128 %token EOF
129
130 %nonassoc NOELSE
131 %nonassoc ELSE
132 %right ASSIGN
133 %left OR
134 %left AND
135 %left EQ NEQ
136 %left LT GT LEQ GEQ
137 %left PLUS MINUS
138 %left STAR DIVIDE MOD
139 %right NOT NEG Deref REF
140 %left DOT
141
142 %start program
143 %type <Ast.program> program
144
145 %%

```

```

146
147 program:
148     decls EOF { $1 }
149
150 decls:
151     /* nothing */ { [], [], [] }
152 | decls vdecl { ($2 :: first $1), second $1, third $1 }
153 | decls fdecl { first $1, ($2 :: second $1), third $1 }
154 | decls sdecl { first $1, second $1, ($2 :: third $1) }
155
156 fdecl:
157     FUNCTION ID LPAREN formals\_opt RPAREN typ LBRACE vdecl\_list stmt\_list RBRACE
158     { { typ = $6;
159       fname = $2;
160       formals = $4;
161       locals = List.rev $8;
162       body = List.rev $9 } }
163
164 formals\_opt:
165     /* nothing */ { [] }
166 | formal\_list { List.rev $1 }
167
168 formal\_list:
169     ID typ { [($2,$1)] }
170 | formal\_list COMMA ID typ { ($4,$3) :: $1 }
171
172 typ:
173     INT { Int }
174 | FLOAT { Float }
175 | BOOL { Bool }
176 | VOID { Void }
177 | STRTYPE { MyString }
178 | STRUCT ID { StructType ($2) }
179 | VOIDSTAR { Voidstar }
180 | STAR %prec POINTER typ { PointerType ($2) }
181
182 vdecl\_list:
183     /* nothing */ { [] }
184 | vdecl\_list vdecl { $2 :: $1 }
185
186 vdecl:
187     LET ID typ SEMI { ($3, $2) }
188
189 sdecl:
190     STRUCT ID LBRACE vdecl\_list RBRACE
191     { { sname = $2;
192       sformals = $4;
193       } }
194
195 stmt\_list:
196     /* nothing */ { [] }
197 | stmt\_list stmt { $2 :: $1 }
198
199 stmt:
200     expr SEMI { Expr $1 }
201 | RETURN SEMI { Return Noexpr }

```

```

202 | RETURN expr SEMI { Return $2 }
203 | LBRACE stmt\_list RBRACE { Block(List.rev $2) }
204 | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
205 | IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) }
206 | FOR LPAREN expr\_opt SEMI expr SEMI expr\_opt RPAREN stmt
207 |   { For($3, $5, $7, $9) }
208 | FOR LPAREN expr RPAREN stmt { While($3, $5) }
209
210 expr\_opt:
211   /* nothing */ { Noexpr }
212 | expr { $1 }
213
214 expr:
215   LITERAL { Literal($1) }
216 | FLOATLITERAL { FloatLiteral($1) }
217 | TRUE { BoolLit(true) }
218 | FALSE { BoolLit(false) }
219 | ID { Id($1) }
220 | STRING { MyStringLit($1) }
221 | expr PLUS expr { Binop($1, Add, $3) }
222 | expr MINUS expr { Binop($1, Sub, $3) }
223 | expr STAR expr { Binop($1, Mult, $3) }
224 | expr DIVIDE expr { Binop($1, Div, $3) }
225 | expr MOD expr { Binop($1, Mod, $3) }
226 | expr EQ expr { Binop($1, Equal, $3) }
227 | expr NEQ expr { Binop($1, Neq, $3) }
228 | expr LT expr { Binop($1, Less, $3) }
229 | expr LEQ expr { Binop($1, Leq, $3) }
230 | expr GT expr { Binop($1, Greater, $3) }
231 | expr GEQ expr { Binop($1, Geq, $3) }
232 | expr AND expr { Binop($1, And, $3) }
233 | expr OR expr { Binop($1, Or, $3) }
234 | expr DOT ID { Dotop($1, $3) }
235 | expr LEFTBR LITERAL RIGHTBR { ArrayRef($1, $3) }
236 | CAST expr TO typ { Castop($4, $2) }
237 | MINUS expr %prec NEG { Unop(Neg, $2) }
238 | STAR expr %prec Deref { Unop(Deref, $2) }
239 | REF expr { Unop(Ref, $2) }
240 | NOT expr { Unop(Not, $2) }
241 | expr ASSIGN expr { Assign($1, $3) }
242 | ID LPAREN actuals\_opt RPAREN { Call($1, $3) }
243 | LPAREN expr RPAREN { $2 }
244
245 actuals\_opt:
246   /* nothing */ { [] }
247 | actuals\_list { List.rev $1 }
248
249 actuals\_list:
250   expr { [$1] }
251 | actuals\_list COMMA expr { $3 :: $1 }

```

8.4 semant.ml

```

252 (* Democritus, adapted from MicroC by Stephen Edwards Columbia University *)
253 (* Semantic checking for compiler *)

```

```

254
255 open Ast
256
257 module StringMap = Map.Make(String)
258 module StringSet = Set.Make(String)
259
260 (* Semantic checking of a program. Returns void if successful,
261    throws an exception if something is wrong.
262
263    Check each global variable, then check each function *)
264
265 let check (globals, functions, structs) =
266
267   (* Raise an exception if the given list has a duplicate *)
268   let report\duplicate exceptf list =
269     let rec helper = function
270       n1 :: n2 :: \_ when n1 = n2 -> raise (Failure (exceptf n1))
271       | \_ :: t -> helper t
272       | [] -> ()
273     in helper (List.sort compare list)
274   in
275
276   (*Raise an exception if there is a recursive struct dependency*)
277
278   let find\sdecl\_from\_sname struct\_type\_name =
279     try List.find (fun s-> s.sname= struct\_type\_name) structs
280     with Not\_found -> raise (Failure("Struct of name " ^ struct\_type\_name ^ "not
281                                     found."))
282   in
283   let rec check\_recursive\_struct\_helper sdecl seen\_set =
284     let check\_if\_repeat struct\_type\_name =
285       let found = StringSet.mem struct\_type\_name seen\_set in
286       if found then raise (Failure ("recursive struct definition"))
287       else check\_recursive\_struct\_helper (find\sdecl\_from\_sname struct\_type\_
288                                         _name) (StringSet.add struct\_type\_name seen\_set)
289     in
290     let is\_struct\_field = function
291       (StructType s, \_) -> check\_if\_repeat s
292       | \_ -> ()
293     in
294     List.iter (is\_struct\_field) sdecl.sformals
295   in
296   let check\_recursive\_struct sdecl =
297     check\_recursive\_struct\_helper sdecl StringSet.empty
298   in
299   let \_ = List.map check\_recursive\_struct structs
300   in
301   (* Raise an exception if a given binding is to a void type *)
302   let check\_not\_void exceptf = function
303     (Void, n) -> raise (Failure (exceptf n))
304     | \_ -> ()
305   in
306   (* Raise an exception of the given rvalue type cannot be assigned to
307      the given lvalue type *)
308   let check\_assign lvaluet rvaluet err =

```

```

308 if (String.compare (string\_of\_typ lvaluet) (string\_of\_typ rvaluet)) == 0
309 then lvaluet
310 else raise err
311   (*if lvaluet == rvaluet then lvaluet else raise err*)
312 in
313
314 let match\_struct\_to\_accessor a b =
315   let s1 = try List.find (fun s-> s.sname=a) structs
316   with Not\_found -> raise (Failure("Struct of name " ^ a ^ "not found.")) in
317   try fst( List.find (fun s-> snd(s)=b) s1.sformals) with
318 Not\_found -> raise (Failure("Struct " ^ a ^ " does not have field " ^ b))
319 in
320
321 let check\_access lvaluet rvalues =
322   match lvaluet with
323   StructType s -> match\_struct\_to\_accessor s rvalues
324   | \_ -> raise (Failure(string\_of\_typ lvaluet ^ " is not a struct"))
325
326 in
327
328 (**** Checking Global Variables ****)
329
330 List.iter (check\_not\_void (fun n -> "illegal void global " ^ n)) globals;
331
332 report\_duplicate (fun n -> "duplicate global " ^ n) (List.map snd globals);
333
334 (**** Checking Functions ****)
335
336 if List.mem "append\_strings" (List.map (fun fd -> fd.fname) functions)
337 then raise (Failure ("function append\_strings may not be defined")) else ();
338
339 if List.mem "int\_to\_string" (List.map (fun fd -> fd.fname) functions)
340 then raise (Failure ("function int\_to\_string may not be defined")) else ();
341
342 if List.mem "print" (List.map (fun fd -> fd.fname) functions)
343 then raise (Failure ("function print may not be defined")) else ();
344
345 if List.mem "thread" (List.map (fun fd -> fd.fname) functions)
346 then raise (Failure ("function thread may not be defined")) else ();
347
348 if List.mem "exec\_prog" (List.map (fun fd -> fd.fname) functions)
349 then raise (Failure ("function exec\_prog may not be defined")) else ();
350
351 if List.mem "free" (List.map (fun fd -> fd.fname) functions)
352 then raise (Failure ("function free may not be defined")) else ();
353
354 if List.mem "malloc" (List.map (fun fd -> fd.fname) functions)
355 then raise (Failure ("function malloc may not be defined")) else ();
356
357 if List.mem "open" (List.map (fun fd -> fd.fname) functions)
358 then raise (Failure ("function open may not be defined")) else ();
359
360 if List.mem "close" (List.map (fun fd -> fd.fname) functions)
361 then raise (Failure ("function close may not be defined")) else ();
362
363 if List.mem "read" (List.map (fun fd -> fd.fname) functions)

```



```

364 then raise (Failure ("function read may not be defined")) else ();
365
366 if List.mem "write" (List.map (fun fd -> fd.fname) functions)
367 then raise (Failure ("function write may not be defined")) else ();
368
369 if List.mem "lseek" (List.map (fun fd -> fd.fname) functions)
370 then raise (Failure ("function lseek may not be defined")) else ();
371
372 if List.mem "sleep" (List.map (fun fd -> fd.fname) functions)
373 then raise (Failure ("function sleep may not be defined")) else ();
374
375 if List.mem "request\_from\_server" (List.map (fun fd -> fd.fname) functions)
376 then raise (Failure ("function request\_from\_server may not be defined")) else ();
377
378 if List.mem "memset" (List.map (fun fd -> fd.fname) functions)
379 then raise (Failure ("function memset may not be defined")) else ();
380
381 report\_duplicate (fun n -> "duplicate function " ^ n)
382   (List.map (fun fd -> fd.fname) functions);
383
384 (* Function declaration for a named function *)
385 let built\_in\_decls\_funcs = [
386   { typ = Void; fname = "print\_int"; formals = [(Int, "x")];
387     locals = []; body = [] };
388
389   { typ = Void; fname = "printb"; formals = [(Bool, "x")];
390     locals = []; body = [] };
391
392   { typ = Void; fname = "print\_float"; formals = [(Float, "x")];
393     locals = []; body = [] };
394
395   { typ = Void; fname = "thread"; formals = [(MyString, "func"); (MyString, "arg")
396     ; (Int, "nthreads")]; locals = []; body = [] };
397
398   { typ = MyString; fname = "malloc"; formals = [(Int, "size")]; locals = []; body
399     = [] };
400
401   (* { typ = DerefAndSet; fname = "malloc"; formals = [(Int, "size")]; locals = [];
402     body = [] }; *)
403
404   { typ = Int; fname = "open"; formals = [(MyString, "name"); (Int, "flags"); (Int
405     , "mode")]; locals = []; body = [] };
406
407   { typ = Int; fname = "close"; formals = [(Int, "fd")]; locals = []; body = [] };
408
409   { typ = Int; fname = "read"; formals = [(Int, "fd"); (MyString, "buf"); (Int, "
410     count")]; locals = []; body = [] };
411
412   { typ = Int; fname = "write"; formals = [(Int, "fd"); (MyString, "buf"); (Int,
413     "count")]; locals = []; body = [] };
414
415   { typ = Int; fname = "lseek"; formals = [(Int, "fd"); (Int, "offset"); (Int, "
416     whence")]; locals = []; body = [] };
417
418   { typ = Int; fname = "sleep"; formals = [(Int, "seconds")]; locals = []; body =
419     [] };

```

```

412
413     { typ = Int; fname = "memset"; formals = [(MyString, "s"); (Int, "val"); (Int,
414       "size")]; locals = []; body = [] };
415
416     { typ = MyString; fname = "request\_from\_server"; formals = [(MyString, "link")
417       ]; locals = []; body = [] }
418 ;
419
420     { typ = Int; fname = "exec\_prog"; formals = [(MyString, "arg1"); (MyString, "
421       arg2"); (MyString, "arg3") ]; locals = []; body = [] };
422
423     { typ = Void; fname = "free"; formals = [(MyString, "tofree")]; locals = [];
424       body = [] }
425 ;
426
427     { typ = Void; fname = "append\_strings"; formals = [(MyString, "str1"); (
428       MyString, "str2")]; locals = []; body = [] };
429
430     { typ = Void; fname = "int\_to\_string"; formals = [(Int, "n"); (MyString, "buf
431       ")]; locals = []; body = [] }
432 ]
433
434 in
435
436 let built\_in\_decls\_names = [ "print\_int"; "printb"; "print\_float"; "thread"; "
437   malloc"; "open"; "close"; "read"; "write"; "lseek"; "sleep"; "memset"; "request\
438   _from\_server"; "exec\_prog"; "free"; "append\_strings"; "int\_to\_string" ]
439
440 in
441
442 let built\_in\_decls = List.fold\_right2 (StringMap.add)
443   built\_in\_decls\_names
444   built\_in\_decls\_funcs
445   (StringMap.singleton "print"
446     { typ = Void; fname = "print"; formals = [(MyString, "
447       x")];
448       locals = []; body = [] })
449
450 in
451
452 let function\_decls = List.fold\_left (fun m fd -> StringMap.add fd.fname fd m)
453   built\_in\_decls functions
454
455 in
456
457 let function\_decl s = try StringMap.find s function\_decls
458   with Not\_found -> raise (Failure ("unrecognized function " ^ s))
459
460 in
461
462 let \_ = function\_decl "main" in (* Ensure "main" is defined *)
463
464 let check\_function func =
465
466   List.iter (check\_not\_void (fun n -> "illegal void formal " ^ n ^
467     " in " ^ func.fname)) func.formals;

```

```

459
460   report\_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^ func.fname)
461     (List.map snd func.formals);
462
463   List.iter (check\_not\_void (fun n -> "illegal void local " ^ n ^
464     " in " ^ func.fname)) func.locals;
465
466   report\_duplicate (fun n -> "duplicate local " ^ n ^ " in " ^ func.fname)
467     (List.map snd func.locals);
468
469   (* Type of each variable (global, formal, or local *)
470   let symbols = List.fold\_left (fun m (t, n) -> StringMap.add n t m)
471   StringMap.empty (globals @ func.formals @ func.locals )
472   in
473
474   let type\_of\_identifier s =
475     try StringMap.find s symbols
476     with Not\_found -> raise (Failure ("undeclared identifier " ^ s))
477   in
478
479   (* Return the type of an expression or throw an exception *)
480   let rec expr = function
481   Literal \_ -> Int
482     | FloatLiteral \_ -> Float
483     | BoolLit \_ -> Bool
484     | MyStringLit \_ -> MyString
485     | Id s -> type\_of\_identifier s
486     | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2 in
487   (match op with
488     | Add | Sub | Mult | Div when t1 = Int && t2 = Int -> Int
489     | Add | Sub | Mult | Div when t1 = Float && t2 = Float -> Float
490     | Mod when t1 = Int && t2 = Int -> Int
491     | Equal | Neq when t1 = t2 -> Bool
492     | Less | Leq | Greater | Geq when t1 = Int && t2 = Int -> Bool
493     | And | Or when t1 = Bool && t2 = Bool -> Bool
494     | \_ -> raise (Failure ("illegal binary operator " ^
495       string\_of\_typ t1 ^ " " ^ string\_of\_op op ^ " " ^
496       string\_of\_typ t2 ^ " in " ^ string\_of\_expr e))
497     )
498     | Dotop(e1, field) -> let lt = expr e1 in
499     check\_access (lt) (field)
500     | Castop(t, \_) -> (*check later*) t
501     | ArrayRef (e, idx) -> let t = expr e in
502     (match t with
503       | PointerType s -> s
504     | \_ -> raise (Failure("cannot dereference a " ^ string\_of\_typ t)) )
505     | Unop(op, e) as ex -> let t = expr e in
506   (match op with
507     | Neg when t = Int -> Int
508     | Not when t = Bool -> Bool
509     | Deref -> (match t with
510       | PointerType s -> s
511     | \_ -> raise (Failure("cannot dereference a " ^ string\_of\_typ t)) )
512     | Ref -> PointerType(t)
513     | \_ -> raise (Failure ("illegal unary operator " ^ string\_of\_uop op ^
514       string\_of\_typ t ^ " in " ^ string\_of\_expr ex)))

```

```

515 | Noexpr -> Void
516 | Call(fname, actuals) as call -> let fd = function\_decl fname in
517
518     if List.length actuals != List.length fd.formals then
519         raise (Failure ("expecting " ^ string\_of\_int
520             (List.length fd.formals) ^ " arguments in " ^ string\_of\_expr call))
521     else
522         List.iter2 (fun (ft, \_) e -> let et = expr e in
523             ignore (check\_assign ft et
524                 (Failure ("illegal actual argument found " ^ string\_of\_typ et ^
525                     " expected " ^ string\_of\_typ ft ^ " in " ^ string\_of\_expr e))))
526             fd.formals actuals;
527         fd.typ
528 | Assign(e1, e2) as ex ->
529 (match e1 with
530 | Id s ->
531     let lt = type\_of\_identifier s and rt = expr e2 in
532     check\_assign (lt) (rt) (Failure ("illegal assignment " ^ string\_of\_typ lt
533         ^ " = " ^
534             string\_of\_typ rt ^ " in " ^ string\_of\_expr ex))
535 | Unop(op, \_) ->
536     (match op with
537     | Deref -> expr e2
538     | \_ -> raise (Failure("whatever")))
539 | Dotop (\_, \_) -> expr e2
540 | \_ -> raise (Failure("whatever")))
541 )
542
543 in
544
545 let check\_bool\_expr e = if expr e != Bool
546 then raise (Failure ("expected Boolean expression in " ^ string\_of\_expr e))
547 else () in
548
549 (* Verify a statement or throw an exception *)
550 let rec stmt = function
551 Block s1 -> let rec check\_block = function
552     [Return \_ as s] -> stmt s
553     | Return \_ :: \_ -> raise (Failure "nothing may follow a return")
554     | Block s1 :: ss -> check\_block (s1 @ ss)
555     | s :: ss -> stmt s ; check\_block ss
556     | [] -> ()
557 in check\_block s1
558 | Expr e -> ignore (expr e)
559 | Return e -> let t = expr e in if t = func.typ then () else
560     raise (Failure ("return gives " ^ string\_of\_typ t ^ " expected " ^
561         string\_of\_typ func.typ ^ " in " ^ string\_of\_expr e))
562
563 | If(p, b1, b2) -> check\_bool\_expr p; stmt b1; stmt b2
564 | For(e1, e2, e3, st) -> ignore (expr e1); check\_bool\_expr e2;
565     ignore (expr e3); stmt st
566 | While(p, s) -> check\_bool\_expr p; stmt s
567 in
568
569 stmt (Block func.body)

```

```

570
571   in
572   List.iter check\_function functions

```

8.5 ast.ml

```

573 (* Democritus, adapted from MicroC by Stephen Edwards Columbia University *)
574 (* Abstract Syntax Tree and functions for printing it *)
575
576 type op = Add | Sub | Mult | Div | Mod | Equal | Neq | Less | Leq | Greater | Geq |
577         And | Or
578
579 type uop = Neg | Not | Deref | Ref
580
581 type typ = Int | Float | Bool | Void | MyString | StructType of string | Voidstar |
582         PointerType of typ
583
584 type bind = typ * string
585
586 type expr =
587   Literal of int
588   | FloatLiteral of float
589   | BoolLit of bool
590   | MyStringLit of string
591   | Id of string
592   | ArrayRef of string * int
593   | Binop of expr * op * expr
594   | Dotop of expr * string
595   | Castop of typ * expr
596   | Unop of uop * expr
597   | Assign of expr * expr
598   | Call of string * expr list
599   | Noexpr
600
601 type stmt =
602   Block of stmt list
603   | Expr of expr
604   | Return of expr
605   | If of expr * stmt * stmt
606   | For of expr * expr * expr * stmt
607   | While of expr * stmt
608
609 type func\_decl = {
610   typ : typ;
611   fname : string;
612   formals : bind list;
613   locals : bind list;
614   body : stmt list;
615 }
616
617 type struct\_decl = {
618   sname: string;
619   sformals: bind list;
620 }

```

```

621
622 type program = bind list * func\_decl list * struct\_decl list
623
624 (* Pretty-printing functions *)
625
626 let string\_of\_op = function
627   Add -> "+"
628   | Sub -> "-"
629   | Mult -> "*"
630   | Div -> "/"
631   | Mod -> "%"
632   | Equal -> "=="
633   | Neq -> "!="
634   | Less -> "<"
635   | Leq -> "<="
636   | Greater -> ">"
637   | Geq -> ">="
638   | And -> "&&"
639   | Or -> "||"
640
641 let string\_of\_uop = function
642   Neg -> "-"
643   | Not -> "!"
644   | Deref -> "*"
645   | Ref -> "&"
646
647 let rec string\_of\_typ = function
648   Int -> "int"
649   | Float -> "float"
650   | Bool -> "bool"
651   | Void -> "void"
652   | MyString -> "string"
653   | StructType(s) -> "struct" ^ s
654   | Voidstar -> "voidstar"
655   | PointerType(s) -> "pointerof" ^ (string\_of\_typ s)
656
657 let rec string\_of\_expr = function
658   Literal(l) -> string\_of\_int l
659   | FloatLiteral(l) -> string\_of\_float l
660   | BoolLit(true) -> "true"
661   | BoolLit(false) -> "false"
662   | MyStringLit(s) -> s
663   | Id(s) -> s
664   | ArrayRef(s, l) -> s ^ "[" ^ string\_of\_int l ^ "]"
665   | Binop(e1, o, e2) ->
666     string\_of\_expr e1 ^ " " ^ string\_of\_op o ^ " " ^ string\_of\_expr e2
667   | Unop(o, e) -> string\_of\_uop o ^ string\_of\_expr e
668   | Dotop(e1, e2) -> string\_of\_expr e1 ^ "." ^ e2
669   | Castop(t, e) -> "(" ^ string\_of\_typ t ^ ")" ^ string\_of\_expr e
670   | Assign(v, e) -> string\_of\_expr v ^ " = " ^ string\_of\_expr e
671   | Call(f, el) ->
672     f ^ "(" ^ String.concat ", " (List.map string\_of\_expr el) ^ ")"
673   | Noexpr -> ""
674
675 let rec string\_of\_stmt = function
676   Block(stmts) ->

```

```

677   "{\n" ^ String.concat "" (List.map string\_of\_stmt stmts) ^ "}\n"
678 | Expr(expr) -> string\_of\_expr expr ^ ";\n";
679 | Return(expr) -> "return " ^ string\_of\_expr expr ^ ";\n";
680 | If(e, s, Block([])) -> "if (" ^ string\_of\_expr e ^ ")\n" ^ string\_of\_stmt s
681 | If(e, s1, s2) -> "if (" ^ string\_of\_expr e ^ ")\n" ^
682   string\_of\_stmt s1 ^ "else\n" ^ string\_of\_stmt s2
683 | For(e1, e2, e3, s) ->
684   "for (" ^ string\_of\_expr e1 ^ " ; " ^ string\_of\_expr e2 ^ " ; " ^
685   string\_of\_expr e3 ^ ") " ^ string\_of\_stmt s
686 | While(e, s) -> "while (" ^ string\_of\_expr e ^ ") " ^ string\_of\_stmt s
687
688 let string\_of\_vdecl (t, id) = string\_of\_typ t ^ " " ^ id ^ ";\n"
689
690 let string\_of\_fdecl fdecl =
691   string\_of\_typ fdecl.typ ^ " " ^
692   fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals) ^
693   ")\n{\n" ^
694   String.concat "" (List.map string\_of\_vdecl fdecl.locals) ^
695   String.concat "" (List.map string\_of\_stmt fdecl.body) ^
696   "}\n"
697
698 let string\_of\_sdecl sdecl = sdecl.sname
699
700 let string\_of\_program (vars, funcs, structs) =
701   String.concat "" (List.map string\_of\_vdecl vars) ^ "\n" ^
702   String.concat "\n" (List.map string\_of\_fdecl funcs) ^ "\n" ^
703   String.concat "\n" (List.map string\_of\_sdecl structs)

```

8.6 codegen.ml

```

704 module L = Lllvm
705 module A = Ast
706 module StringMap = Map.Make(String)
707
708 let translate (globals, functions, structs) =
709   let context = L.global\_context () in
710   let the\_module = L.create\_module context "Democritus"
711   and i32\_t = L.i32\_type context
712   (* and i8\_t = L.i8\_type context *)
713   and i1\_t = L.i1\_type context
714   and void\_t = L.void\_type context
715   and ptr\_t = L.pointer\_type (L.i8\_type (context))
716   and float\_t = L.double\_type context
717   in
718
719
720   let struct\_types:(string, L.lltype) Hashtbl.t = Hashtbl.create 50 in
721
722   let add\_empty\_named\_struct\_types sdecl =
723     let struct\_t = L.named\_struct\_type context sdecl.A.sname in
724     Hashtbl.add struct\_types sdecl.A.sname struct\_t
725   in
726   let \_ =
727     List.map add\_empty\_named\_struct\_types structs
728   in

```

```

729
730 let rec ltype\_of\_typ = function
731   A.Int -> i32\_t
732 |   A.Float -> float\_t
733 |   A.Bool -> i1\_t
734 |   A.Void -> void\_t
735 |   A.StructType s -> Hashtbl.find struct\_types s
736 |   A.MyString -> ptr\_t
737 |   A.Voidstar -> ptr\_t
738 |   A.PointerType t -> L.pointer\_type (ltype\_of\_typ t) in
739 let populate\_struct\_type sdecl =
740   let struct\_t = Hashtbl.find struct\_types sdecl.A.sname in
741   let type\_list = Array.of\_list (List.map (fun(t, \_) -> ltype\_of\_typ t) sdecl.A.
     sforms) in
742   L.struct\_set\_body struct\_t type\_list true
743 in
744   ignore(List.map populate\_struct\_type structs);
745
746 let string\_option\_to\_string = function
747   None -> ""
748 |   Some(s) -> s
749 in
750
751 (*struct\_field\_index is a map where key is struct name and value is another map*)
752 (*in the second map, the key is the field name and the value is the index number*)
753 let struct\_field\_index\_list =
754 let handle\_list m individual\_struct =
755   (*list of all field names for that struct*)
756   let struct\_field\_name\_list = List.map snd individual\_struct.A.sforms in
757   let increment n = n + 1 in
758   let add\_field\_and\_index (m, i) field\_name =
759     (*add each field and index to the second map*)
760     (StringMap.add field\_name (increment i) m, increment i) in
761   (*struct\_field\_map is the second map, with key = field name and value = index*)
762   let struct\_field\_map =
763     List.fold\_left add\_field\_and\_index (StringMap.empty, -1) struct\_field\_name
       \_list
764   in
765     (*add field map (the first part of the tuple) to the main map*)
766     StringMap.add individual\_struct.A.sname (fst struct\_field\_map) m
767   in
768   List.fold\_left handle\_list StringMap.empty structs
769 in
770   (* Declare each global variable; remember its value in a map *)
771 let global\_vars =
772   let global\_var m (t, n) =
773     let init = L.const\_int (ltype\_of\_typ t) 0
774     in StringMap.add n (L.define\_global n init the\_module) m in
775   List.fold\_left global\_var StringMap.empty globals in
776
777 let append\_strings\_t = L.function\_type void\_t [| ptr\_t; ptr\_t |] in
778 let append\_strings\_func = L.declare\_function "append\_strings" append\_strings\_t
  the\_module in
779
780 let int\_to\_string\_t = L.function\_type void\_t [| i32\_t; ptr\_t |] in
781 let int\_to\_string\_func = L.declare\_function "int\_to\_string" int\_to\_string\_t

```



```

the\_module in
782
783 let printf\_t = L.var\_arg\_function\_type i32\_t [| ptr\_t |] in
784 let printf\_func = L.declare\_function "printf" printf\_t the\_module in
785
786 let execl\_t = L.var\_arg\_function\_type i32\_t [| ptr\_t |] in
787 let execl\_func = L.declare\_function "exec\_prog" execl\_t the\_module in
788
789 let free\_t = L.function\_type void\_t [| ptr\_t |] in
790 let free\_func = L.declare\_function "free" free\_t the\_module in
791
792 let malloc\_t = L.function\_type ptr\_t [| i32\_t |] in
793 let malloc\_func = L.declare\_function "malloc" malloc\_t the\_module in
794
795 let request\_from\_server\_t = L.function\_type ptr\_t [| ptr\_t |] in
796 let request\_from\_server\_func = L.declare\_function "request\_from\_server"
    request\_from\_server\_t the\_module in
797
798 let memset\_t = L.function\_type ptr\_t [| ptr\_t; i32\_t; i32\_t |] in
799 let memset\_func = L.declare\_function "memset" memset\_t the\_module in
800
801 (* File I/O functions *)
802 let open\_t = L.function\_type i32\_t [| ptr\_t; i32\_t; i32\_t |] in
803 let open\_func = L.declare\_function "open" open\_t the\_module in
804
805 let close\_t = L.function\_type i32\_t [| i32\_t |] in
806 let close\_func = L.declare\_function "close" close\_t the\_module in
807
808 let read\_t = L.function\_type i32\_t [| i32\_t; ptr\_t; i32\_t |] in
809 let read\_func = L.declare\_function "read" read\_t the\_module in
810
811 let write\_t = L.function\_type i32\_t [| i32\_t; ptr\_t; i32\_t |] in
812 let write\_func = L.declare\_function "write" write\_t the\_module in
813
814 let lseek\_t = L.function\_type i32\_t [| i32\_t; i32\_t; i32\_t |] in
815 let lseek\_func = L.declare\_function "lseek" lseek\_t the\_module in
816
817 let sleep\_t = L.function\_type i32\_t [| i32\_t |] in
818 let sleep\_func = L.declare\_function "sleep" sleep\_t the\_module in
819
820 let default\_t = L.function\_type ptr\_t [|ptr\_t|] in
821 let default\_func = L.declare\_function "default\_start\_routine" default\_t the\_
    module in
822
823 let param\_ty = L.function\_type ptr\_t [| ptr\_t |] in (* a function that returns
    void\_star and takes as argument void\_star *)
824 let param\_ptr = L.pointer\_type param\_ty in
825 let thread\_t = L.function\_type void\_t [| param\_ptr; ptr\_t; i32\_t |] in (*a
    function that returns void and takes (above) and a voidstar and an int *)
826 let thread\_func = L.declare\_function "init\_thread" thread\_t the\_module in
827
828
829 (* Define each function (arguments and return type) so we can call it *)
830 let function\_decls =
831     let function\_decl m fdecl =
832         let name = fdecl.A.fname

```

```

833     and formal\_types =
834     Array.of\_list (List.map (fun (t,\_) -> ltype\_of\_typ t) fdecl.A.formals)
835     in let ftype = L.function\_type (ltype\_of\_typ fdecl.A.type) formal\_types in
836     StringMap.add name (L.define\_function name ftype the\_module, fdecl) m in
837     List.fold\_left function\_decl StringMap.empty functions in
838
839     (* Fill in the body of the given function *)
840     let build\_function\_body fdecl =
841         let (the\_function, \_) = StringMap.find fdecl.A.fname function\_decls in
842         let builder = L.builder\_at\_end context (L.entry\_block the\_function) in
843
844         let int\_format\_str = L.build\_global\_stringptr "%d\n" "fmt" builder in
845         let float\_format\_str = L.build\_global\_stringptr "%f\n" "fmt" builder in
846
847         (* Construct the function's "locals": formal arguments and locally
848            declared variables. Allocate each on the stack, initialize their
849            value, if appropriate, and remember their values in the "locals" map *)
850         let local\_vars =
851             let add\_formal m (t, n) p = L.set\_value\_name n p;
852             let local = L.build\_alloca (ltype\_of\_typ t) n builder in
853             ignore (L.build\_store p local builder);
854             StringMap.add n local m in
855
856         let add\_local m (t, n) =
857             let local\_var = L.build\_alloca (ltype\_of\_typ t) n builder
858             in StringMap.add n local\_var m in
859
860         let formals = List.fold\_left2 add\_formal StringMap.empty fdecl.A.formals
861             (Array.to\_list (L.params the\_function)) in
862         List.fold\_left add\_local formals fdecl.A.locals in
863
864         (* Return the value for a variable or formal argument *)
865         let lookup n = try StringMap.find n local\_vars
866             with Not\_found -> try StringMap.find n global\_vars
867             with Not\_found -> raise (Failure ("undeclared variable " ^ n))
868         in
869
870         (* Construct code for an expression; return its value *)
871         let rec llvalue\_expr\_getter builder = function
872             A.Id s -> lookup s
873         | A.Dotop(e1, field) ->
874             (match e1 with
875             A.Id s -> let etype = fst(
876                 try List.find (fun t->snd(t)=s) fdecl.A.locals
877                 with Not\_found -> raise (Failure("Unable to find" ^ s ^ "in dotop")))
878             in
879                 (try match etype with
880                     A.StructType t->
881                         let index\_number\_list = StringMap.find t struct\_field\_index\_list in
882                         let index\_number = StringMap.find field index\_number\_list in
883                         let struct\_llvalue = lookup s in
884                         let access\_llvalue = L.build\_struct\_gep struct\_llvalue index\_number "
885                             dotop\_terminal" builder in
886                         access\_llvalue
887                     | \_ -> raise (Failure("No structype."))

```

```

888     with Not\_found -> raise (Failure("unable to find" ^ s)) )
889 | \_ as e1\_expr -> let e1'\_llvalue = llvalue\_expr\_getter builder e1\_expr in
890     let loaded\_e1' = expr builder e1\_expr in
891     let e1'\_lltype = L.type\_of loaded\_e1' in
892     let e1'\_struct\_name\_string\_option = L.struct\_name e1'\_lltype in
893     let e1'\_struct\_name\_string = string\_option\_to\_string e1'\_struct\_name\_
      _string\_option in
894     let index\_number\_list = StringMap.find e1'\_struct\_name\_string struct\_field
      \_index\_list in
895     let index\_number = StringMap.find field index\_number\_list in
896     let access\_llvalue = L.build\_struct\_gep e1'\_llvalue index\_number "gep\_in\_
      _dotop" builder in
897     access\_llvalue )
898
899
900 | A.Unop(op, e) ->
901     (match op with
902     | A.Deref ->
903         let e\_llvalue = (llvalue\_expr\_getter builder e) in
904         let e\_loaded = L.build\_load e\_llvalue "loaded\_deref" builder in
905         e\_loaded
906     | \_ -> raise (Failure("nooo")))
907 )
908 | \_ -> raise (Failure ("in llvalue\_expr\_getter but not a dotop!"))
909 and
910 expr builder = function
911 | A.Literal i -> L.const\_int i32\_t i
912 (* | A.MyStringLit str -> L.const\_stringz context str *)
913 | A.FloatLiteral f -> L.const\_float float\_t f
914 | A.MyStringLit str -> L.build\_global\_stringptr str "tmp" builder
915 | A.BoolLit b -> L.const\_int i1\_t (if b then 1 else 0)
916 | A.Noexpr -> L.const\_int i32\_t 0
917 | A.Id s -> L.build\_load (lookup s) s builder
918 | A.ArrayRef (e, l) ->
919     let e\_llvalue = (llvalue\_expr\_getter builder e) in
920     let e\_loaded = L.build\_load e\_llvalue "loaded\_deref" builder in
921     e\_loaded
922 | A.Binop (e1, op, e2) ->
923     let e1' = expr builder e1
924     and e2' = expr builder e2 in
925     (match op with
926     | A.Add -> (let e1\_type\_string = L.string\_of\_lltype (L.type\_of e1') in
927         (match e1\_type\_string with
928         | "double" -> L.build\_fadd
929         | "i32" -> L.build\_add
930         | \_ -> raise(Failure("Can only add ints or floats"))) ))
931 | A.Sub -> (let e1\_type\_string = L.string\_of\_lltype (L.type\_of e1') in
932     (match e1\_type\_string with
933     | "double" -> L.build\_fsub
934     | "i32" -> L.build\_sub
935     | \_ -> raise(Failure("Can only subtract ints or floats"))) ))
936 | A.Mult -> (let e1\_type\_string = L.string\_of\_lltype (L.type\_of e1') in
937     (match e1\_type\_string with
938     | "double" -> L.build\_fmul
939     | "i32" -> L.build\_mul
940     | \_ -> raise(Failure("Can only multiply ints or floats"))) ))

```

```

941 | A.Div    -> (let e1\_type\_string = L.string\_of\_lltype (L.type\_of e1') in
942 |         (match e1\_type\_string with
943 |         "double" -> L.build\_fdiv
944 |         | "i32" -> L.build\_sdiv
945 |         | \_ -> raise(Failure("Can only divide ints or floats"))) ))
946 | A.Mod    -> L.build\_srem
947 | A.And    -> L.build\_and
948 | A.Or     -> L.build\_or
949 | A.Equal  -> L.build\_icmp L.Icmp.Eq
950 | A.Neq    -> L.build\_icmp L.Icmp.Ne
951 | A.Less   -> L.build\_icmp L.Icmp.Slt
952 | A.Leq    -> L.build\_icmp L.Icmp.Sle
953 | A.Greater -> L.build\_icmp L.Icmp.Sgt
954 | A.Geq    -> L.build\_icmp L.Icmp.Sge
955 | e1' e2' "tmp" builder
956 | A.Dotop(e1, field) -> let \_ = expr builder e1 in
957 (match e1 with
958 | A.Id s -> let etype = fst(
959 |         try List.find (fun t->snd(t)=s) fdecl.A.locals
960 |         with Not\_found -> raise (Failure("Unable to find" ^ s ^ "in dotop")))
961 | in
962 | (try match etype with
963 |     A.StructType t->
964 |         let index\_number\_list = StringMap.find t struct\_field\_index\_list in
965 |         let index\_number = StringMap.find field index\_number\_list in
966 |         let struct\_llvalue = lookup s in
967 |         let access\_llvalue = L.build\_struct\_gep struct\_llvalue index\_number "
968 |             dotop\_terminal" builder in
969 |         let loaded\_access = L.build\_load access\_llvalue "loaded\_dotop\_terminal"
970 |             builder in
971 |         loaded\_access
972 |         | \_ -> raise (Failure("No structype."))
973 |         with Not\_found -> raise (Failure("unable to find" ^ s)) )
974 | \_ as e1\_expr -> let e1'\_llvalue = llvalue\_expr\_getter builder e1\_expr in
975 |         let loaded\_e1' = expr builder e1\_expr in
976 |         let e1'\_lltype = L.type\_of loaded\_e1' in
977 |         let e1'\_struct\_name\_string\_option = L.struct\_name e1'\_lltype in
978 |         let e1'\_struct\_name\_string = string\_option\_to\_string e1'\_struct\_name\_
979 |             string\_option in
980 |         let index\_number\_list = StringMap.find e1'\_struct\_name\_string struct\_field\_
981 |             index\_list in
982 |         let index\_number = StringMap.find field index\_number\_list in
983 |         let access\_llvalue = L.build\_struct\_gep e1'\_llvalue index\_number "gep\_in\_
984 |             dotop" builder in
985 |         L.build\_load access\_llvalue "loaded\_dotop" builder )
986 | A.Unop(op, e) ->
987 let e' = expr builder e in
988 (match op with
989 | A.Neg    -> L.build\_neg e' "tmp" builder
990 | A.Not    -> L.build\_not e' "temp" builder
991 | A.Deref -> let e\_loaded = L.build\_load e' "loaded\_deref" builder in
992 |         e\_loaded
993 | A.Ref -> let e\_llvalue = (llvalue\_expr\_getter builder e) in
994 |         e\_llvalue
995 | )

```

```

992 | A.Castop(ast\_cast\_type, e) ->
993 let cast\_lltype = ltype\_of\_typ ast\_cast\_type in
994 let e\_llvalue = expr builder e in
995 L.build\_pointercast e\_llvalue cast\_lltype "plz" builder
996 | A.Assign (lhs, e2) -> let e2' = expr builder e2 in
997 (match lhs with
998 A.Id s -> ignore (L.build\_store e2' (lookup s) builder); e2'
999 | A.Dotop (e1, field) ->
1000 (match e1 with
1001
1002 A.Id s -> let eltyp = fst(
1003 try List.find (fun t -> snd(t) = s) fdecl.A.locals
1004 with Not\_found -> raise(Failure("unable to find" ^ s ^ "in Sassign")))
1005 in
1006 (match eltyp with
1007 A.StructType t -> (try
1008 let index\_number\_list = StringMap.find t struct\_field\_index\_list in
1009 let index\_number = StringMap.find field index\_number\_list in
1010 let struct\_llvalue = lookup s in
1011 let access\_llvalue = L.build\_struct\_gep struct\_llvalue index\_number
1012 field builder in
1013 (try (ignore(L.build\_store e2' access\_llvalue builder);e2')
1014 with Not\_found -> raise (Failure("unable to store " ^ t)) )
1015 with Not\_found -> raise (Failure("unable to find" ^ s)) )
1016 | \_ -> raise (Failure("StructType not found.")))
1017 |\_ as e1\_expr -> let e1'\_llvalue = llvalue\_expr\_getter builder e1\_expr
1018 in
1019 let loaded\_e1' = expr builder e1\_expr in
1020 let e1'\_lltype = L.type\_of loaded\_e1' in
1021 let e1'\_struct\_name\_string\_option = L.struct\_name e1'\_lltype in
1022 let e1'\_struct\_name\_string = string\_option\_to\_string e1'\_struct\_name
1023 \_string\_option in
1024 let index\_number\_list = StringMap.find e1'\_struct\_name\_string struct\_
1025 field\_index\_list in
1026 let index\_number = StringMap.find field index\_number\_list in
1027 let access\_llvalue = L.build\_struct\_gep e1'\_llvalue index\_number "gep\
1028 _in\_Sassign" builder in
1029 let \_ = L.build\_store e2' access\_llvalue builder in
1030 e2'
1031 )
1032
1033 | A.Unop(op, e) ->
1034 (match op with
1035 A.Deref ->
1036 let e\_llvalue = (llvalue\_expr\_getter builder e) in
1037 let e\_loaded = L.build\_load e\_llvalue "loaded\_deref" builder
1038 in
1039 let \_ = L.build\_store e2' e\_loaded builder in
1040 e2'
1041 |\_ -> raise (Failure("nooo"))
1042 )
1043 |\_ -> raise (Failure("can't match in assign"))
1044 )
1045 | A.Call ("print\_int", [e]) | A.Call ("printb", [e]) ->
1046 L.build\_call printf\_func [| int\_format\_str ; (expr builder e) |]
1047 "printf" builder

```

```

1042
1043 | A.Call ("print\_float", [e]) ->
1044 L.build\_call printf\_func [| float\_format\_str; (expr builder e) |] "printf"
    builder
1045
1046 | A.Call ("print", [e]) ->
1047 L.build\_call printf\_func [| (expr builder e) |] "printf" builder
1048
1049 | A.Call ("append\_strings", e) ->
1050 let evald\_expr\_list = List.map (expr builder) e in
1051 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1052 L.build\_call append\_strings\_func evald\_expr\_arr "" builder
1053
1054 | A.Call ("int\_to\_string", e) ->
1055 let evald\_expr\_list = List.map (expr builder) e in
1056 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1057 L.build\_call int\_to\_string\_func evald\_expr\_arr "" builder
1058
1059 | A.Call ("exec\_prog", e) ->
1060 let evald\_expr\_list = List.map (expr builder) e in
1061 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1062 L.build\_call execl\_func evald\_expr\_arr "exec\_prog" builder
1063
1064 | A.Call ("free", e) ->
1065 L.build\_call free\_func (Array.of\_list (List.map (expr builder) e)) "" builder
1066
1067 | A.Call ("malloc", e) ->
1068 let evald\_expr\_list = List.map (expr builder) e in
1069 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1070 L.build\_call malloc\_func evald\_expr\_arr "malloc" builder
1071
1072 | A.Call ("memset", e) ->
1073 let evald\_expr\_list = List.map (expr builder) e in
1074 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1075 L.build\_call memset\_func evald\_expr\_arr "memset" builder
1076
1077 (* File I/O functions *)
1078 | A.Call ("open", e) ->
1079 let evald\_expr\_list = List.map (expr builder) e in
1080 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1081 L.build\_call open\_func evald\_expr\_arr "open" builder
1082
1083 | A.Call ("close", e) ->
1084 let evald\_expr\_list = List.map (expr builder) e in
1085 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1086 L.build\_call close\_func evald\_expr\_arr "close" builder
1087
1088 | A.Call ("read", e) ->
1089 let evald\_expr\_list = List.map (expr builder) e in
1090 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1091 L.build\_call read\_func evald\_expr\_arr "read" builder
1092
1093 | A.Call ("write", e) ->
1094 let evald\_expr\_list = List.map (expr builder) e in
1095 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1096 L.build\_call write\_func evald\_expr\_arr "write" builder

```

```

1097
1098 | A.Call("lseek", e) ->
1099 let evald\_expr\_list = List.map (expr builder) e in
1100 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1101 L.build\_call lseek\_func evald\_expr\_arr "lseek" builder
1102
1103 | A.Call("sleep", e) ->
1104 let evald\_expr\_list = List.map (expr builder) e in
1105 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1106 L.build\_call sleep\_func evald\_expr\_arr "sleep" builder
1107
1108 | A.Call ("thread", e) ->
1109 (* L.build\_call printf\_func [| int\_format\_str ; L.const\_int i32\_t 8 |] "printf"
    builder *)
1110 let evald\_expr\_list = List.map (expr builder) e in
1111 (* let target\_func\_strptr = List.hd evald\_expr\_list in (* jsut get the string by
    doing List.hd on e *)
1112 let target\_func\_str = L.string\_of\_llvalue target\_func\_strptr in *)
1113 let get\_string v = match v with
1114 | A.MyStringLit i -> i
1115 | \_ -> "" in
1116 let target\_func\_str = get\_string (List.hd e) in
1117 (*let target\_func\_str = Option.default "" Some(target\_func\_str\_opt) in *)
1118 let target\_func\_llvalue\_opt = L.lookup\_function target\_func\_str the\_module in
1119 let deopt x = match x with
1120 | Some f -> f
1121 | None -> default\_func in
1122 let target\_func\_llvalue = deopt target\_func\_llvalue\_opt in
1123 let remaining\_list = List.tl evald\_expr\_list in
1124 let new\_arg\_list = target\_func\_llvalue :: remaining\_list in
1125 let new\_arg\_arr = Array.of\_list new\_arg\_list in
1126 L.build\_call thread\_func
1127 new\_arg\_arr
1128 "" builder
1129
1130 | A.Call ("request\_from\_server", e) ->
1131 let evald\_expr\_list = List.map (expr builder) e in
1132 let evald\_expr\_arr = Array.of\_list evald\_expr\_list in
1133 L.build\_call request\_from\_server\_func evald\_expr\_arr "request\_from\_server"
    builder
1134
1135 | A.Call (f, act) ->
1136 let (fdef, fdecl) = StringMap.find f function\_decls in
1137 let actuals = List.rev (List.map (expr builder) (List.rev act)) in
1138 let result = (match fdecl.A.typ with A.Void -> ""
1139 | \_ -> f ^ "\_result") in
1140 L.build\_call fdef (Array.of\_list actuals) result builder
1141 in
1142
1143 (* Invoke "f builder" if the current block doesn't already
1144 have a terminal (e.g., a branch). *)
1145 let add\_terminal builder f =
1146 match L.block\_terminator (L.insertion\_block builder) with
1147 Some \_ -> ()
1148 | None -> ignore (f builder) in
1149

```

```

1150     (* Build the code for the given statement; return the builder for
1151        the statement's successor *)
1152     let rec stmt builder = function
1153 A.Block s1 -> List.fold\_left stmt builder s1
1154   | A.Expr e -> ignore (expr builder e); builder
1155   | A.Return e -> ignore (match fdecl.A.typ with
1156     A.Void -> L.build\_ret\_void builder
1157   | \_ -> L.build\_ret (expr builder e) builder); builder
1158   | A.If (predicate, then\_stmt, else\_stmt) ->
1159     let bool\_val = expr builder predicate in
1160     let merge\_bb = L.append\_block context "merge" the\_function in
1161
1162     let then\_bb = L.append\_block context "then" the\_function in
1163     add\_terminal (stmt (L.builder\_at\_end context then\_bb) then\_stmt)
1164       (L.build\_br merge\_bb);
1165
1166     let else\_bb = L.append\_block context "else" the\_function in
1167     add\_terminal (stmt (L.builder\_at\_end context else\_bb) else\_stmt)
1168       (L.build\_br merge\_bb);
1169
1170     ignore (L.build\_cond\_br bool\_val then\_bb else\_bb builder);
1171     L.builder\_at\_end context merge\_bb
1172
1173   | A.While (predicate, body) ->
1174     let pred\_bb = L.append\_block context "while" the\_function in
1175     ignore (L.build\_br pred\_bb builder);
1176
1177     let body\_bb = L.append\_block context "while\_body" the\_function in
1178     add\_terminal (stmt (L.builder\_at\_end context body\_bb) body)
1179       (L.build\_br pred\_bb);
1180
1181     let pred\_builder = L.builder\_at\_end context pred\_bb in
1182     let bool\_val = expr pred\_builder predicate in
1183
1184     let merge\_bb = L.append\_block context "merge" the\_function in
1185     ignore (L.build\_cond\_br bool\_val body\_bb merge\_bb pred\_builder);
1186     L.builder\_at\_end context merge\_bb
1187
1188   | A.For (e1, e2, e3, body) -> stmt builder
1189     ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )
1190   in
1191
1192     (* Build the code for each statement in the function *)
1193     let builder = stmt builder (A.Block fdecl.A.body) in
1194
1195     (* Add a return if the last block falls off the end *)
1196     add\_terminal builder (match fdecl.A.typ with
1197       A.Void -> L.build\_ret\_void
1198     | t -> L.build\_ret (L.const\_int (ltype\_of\_typ t) 0))
1199   in
1200
1201
1202 List.iter build\_function\_body functions;
1203
1204 let llmem = LlvM.MemoryBuffer.of\_file "bindings.bc" in
1205 let llm = LlvM.bitreader.parse\_bitcode context llmem in

```



```

1206     ignore(Llvm\_linker.link\_modules the\_module llm Llvm\_linker.Mode.PreserveSource);
1207
1208     the\_module

```

8.7 bindings.c

```

1  #include <pthread.h>
2  #include <string.h>
3  #include <sys/socket.h>
4  #include <arpa/inet.h>
5  #include <stdio.h>
6  #include <errno.h>
7  #include <netdb.h>
8  #include <stdlib.h>
9  #include <unistd.h>
10 #include <string.h>
11 #define BUFSIZE 4096
12
13 void append\_strings(void *str1, void *str2)
14 {
15     strcat((char *)str1, (char *)str2);
16 }
17
18 void int\_to\_string(int n, void *buf)
19 {
20     sprintf(buf, "%d", n);
21 }
22
23 int exec\_prog(void *str1, void *str2, void *str3)
24 {
25
26     execl((char *)str1, (char *)str2, (char *)str3, NULL);
27     return 0;
28 }
29
30 /*
31  * Given a URL, send a GET request.
32  */
33 //void *get\_request(void *url, void *filePath)
34 void *request\_from\_server(void *urlVoid)
35 {
36     // www.xkcd.com/index.html
37     char *urlStr = (char *) urlVoid;
38     int idxslash = strchr(urlStr, '/') - urlStr;
39     char *url = malloc(idxslash + 1);
40     char *filePath = malloc(strlen(urlStr) - (idxslash) + 1);
41     memset(url, 0, idxslash - 1);
42     memset(filePath, 0, strlen(urlStr) - (idxslash));
43
44     strncat(url, urlStr, idxslash);
45     strncat(filePath, urlStr + idxslash, strlen(urlStr) - (idxslash));
46     char *fileName = strrchr(urlStr, '/') + 1;
47
48     char *serverIP;
49     int sock;    // socket we connect to remote on

```

```

50     struct sockaddr_in serverAddr;
51     struct hostent *he;
52     char recvbuf[BUFSIZE];
53
54     if ((he = gethostbyname((char *) url)) == NULL) {
55         fprintf(stderr, "gethostbyname() failed.");
56         exit(1);
57     }
58
59     sock = socket(AF_INET, SOCK_STREAM, 0);
60     if (sock < 0) {
61         fprintf(stderr, "socket() failed.");
62         exit(1);
63     }
64     serverIP = inet_ntoa(*(struct in_addr *)he->h_addr);
65     memset(&serverAddr, 0, sizeof(serverAddr));
66     serverAddr.sin_addr.s_addr = inet_addr(serverIP);
67     serverAddr.sin_family = AF_INET;
68     serverAddr.sin_port = htons(80);
69
70     int connected = connect(sock, (struct sockaddr *)&serverAddr, sizeof(serverAddr));
71     if(connected < 0) {
72         fprintf(stderr, "connect() failed.");
73         exit(1);
74     }
75
76     // send HTTP request
77     if (((char *) url)[strlen((char *) url) - 1] == '/') {
78         strcat(url, "index.html");
79     }
80
81     snprintf(recvbuf, sizeof(recvbuf),
82              "GET %s HTTP/1.0\r\n"
83              "Host: %s:%s\r\n"
84              "\r\n",
85              filePath, url, "80");
86     if (send(sock, recvbuf, strlen(recvbuf), 0) != strlen(recvbuf)) {
87         fprintf(stderr, "send() failed.");
88         exit(1);
89     }
90
91     // wrap the socket with a FILE* so that we can read the socket using fgets()
92     FILE *fd;
93     if ((fd = fdopen(sock, "rb")) == NULL) {
94         fprintf(stderr, "fdopen() failed.");
95         exit(1);
96     }
97
98     /* check header for valid protocol and status code */
99     if (fgets(recvbuf, sizeof(recvbuf), fd) == NULL) {
100         fprintf(stderr, "server terminated connection without response.");
101         exit(1);
102     }
103     if (strncmp("HTTP/1.0 ", recvbuf, 9) != 0 && strncmp("HTTP/1.1 ", recvbuf, 9) !=
104         0) {
105         fprintf(stderr, "unknown protocol response: %s.", recvbuf);

```

```

105     exit(1);
106     }
107     if (strncmp("200", recvbuf + 9, 3) != 0) {
108     fprintf(stderr, "request failed with status code %s.", recvbuf);
109     exit(1);
110     }
111     /* ignore remaining header lines */
112     do {
113         memset(recvbuf, 0, BUFSIZE);
114     if (fgets(recvbuf, sizeof(recvbuf), fd) == NULL) {
115         fprintf(stderr, "server terminated connection without sending file.");
116         exit(1);
117     }
118     } while (strcmp("\r\n", recvbuf) != 0);
119
120
121     char *filePathName = malloc(100);
122     memset(filePathName, 0, 100);
123     char *last_slash;
124     if ((last_slash = strrchr(filePath, '/')) != NULL) {
125         if (strlen(last_slash) == 1) {
126             strcpy(filePathName, "index.html");
127         } else {
128             strcpy(filePathName, last_slash + 1);
129         }
130     }
131
132     /* open and read into file */
133     printf("%s\n", filePathName);
134     FILE *outputFile = fopen(filePathName, "wb");
135     if (outputFile == NULL) {
136     fprintf(stderr, "fopen() failed.");
137     exit(1);
138     }
139
140     size_t n;
141     int total = 0;
142     memset(recvbuf, 0, BUFSIZE);
143     printf("buffer contents: %s\n", recvbuf);
144     while ((n = fread(recvbuf, 1, BUFSIZE, fd)) > 0) {
145     if (fwrite(recvbuf, 1, n, outputFile) != n) {
146         fprintf(stderr, "fwrite() failed.");
147         exit(1);
148     }
149     memset(recvbuf, 0, BUFSIZE);
150     total += n;
151     }
152     fprintf(outputFile, "\n");
153     fprintf(stderr, "total bytes written: %d\n", total);
154
155     if (ferror(fd)) {
156     fprintf(stderr, "fread() failed.");
157     exit(1);
158     }
159
160     fclose(outputFile);

```

```
161     fclose(fd);
162
163     return NULL;
164 }
165
166 void *default\_start\_routine(void *arg)
167 {
168     return arg;
169 }
170
171 void init\_thread(void *(*start\_routine) (void *), void *arg, int nthreads)
172 {
173     pthread\_t thread[nthreads];
174     int i;
175     for (i = 0; i < nthreads; i++) {
176         pthread\_create(&thread[i], NULL, start\_routine, arg);
177     }
178
179     for (i = 0; i < nthreads; i++) {
180         pthread\_join(thread[i], NULL);
181     }
182 }
```