Digo

Distributed Golang

Manager: Wenqian Yan (wy2249)
Language Guru: Yufan Chen (yc3858)
System Architect: Sida Huang (sh4081)
Test Designer: Hanxiao Lu (hl3424)

Table of Contents

Part I. Introduction

1.1 Motivation

Digo is an imperative, statically typed, compiled programming language inspired by Golang, but with support for distributed routine.

When it comes to building distributed systems, one of the challenges is that developers may need to think about the network response model, thread management, concurrency, and the resources shared by different threads. Digo is designed to make it easy for the programmer to write simple yet impactful code for the master-worker model without worrying about networks and concurrency.

With Digo, we can easily create a master-worker program where the master can distribute tasks to worker nodes. Users only need to give function annotations (async/ async remote) to run the function asynchronously either locally or on workers, and use await to wait for results if needed.

1.2 Digo Features

Type Scope	Static
Type Strength	Strongly Typed
Garbage Collection	Automatic
Type System	Statically typed
Evaluation	Strict evaluation
Type Inference	Yes
Syntax	Golang-like
Others	Distributed routine

Besides the above language features, to work for the master-worker distributed model, Digo also supports two modes to compile executables (master-mode and worker-mode) to specify when executing. An example to run with master mode: `./a.out --master 127.0.0.1:20001`. And to run code in the worker node, we need to specify its master's address and mode in args:`./a.out worker 127.0.0.1:20001 127.0.0.1:20002`.

A more detailed explanation on how digo works underlyingly with a diagram is in appendix (section 9.16). Highly recommend checking it out to learn more about digo!

Part II. Language Tutorial

2.1 Environment Setup

First, download the code for the compiler along with all of the tests from GitHub:

```
$ git clone https://github.com/wy2249/Digo.git
```

Then, make sure docker is installed on your end, and run:

```
$ docker run --rm -it -v `pwd`:/home/digo -w=/home/digo wy2249/plt
```

2.2 Compilation Guide

Build the compiler binary using make. Sometimes a clean is needed

```
$ make clean
$ make gen
```

To run all the test cases, under home working directory run:

```
$ cd digo-test
$ ./testall.sh
```

To write Digo source code, first create a file with a .digo extension. Inside that file, write a program following the rules outlined in this language reference manual. Then, save your file and run it following the instructions below. If you need inspiration for your first Digo program, feel free to copy any of the many examples provided throughout this manual.

To compile a Digo program into LLVM code, under home working directory run:

```
$ make print-llvm digo=<filename.digo>
```

To compile a Digo program, under home working directory run:

```
$ make build digo=<filename.digo> out=<executable_name>
```

To execute the Digo program with master mode, run:

```
$ ./<executable_name> --master 127.0.0.1:20001
```

To execute the Digo program with worker mode, run:

```
$ ./<executable_name> --worker 127.0.0.1:20001 127.0.0.1:20002
```

Note that in worker mode, the first ip address is master's address and the second one is worker's ip address.

2.3 Language Tutorial

Here is a walk-through for creating your first Digo program!

First, create a file called hello-world.digo and copy and paste the following code into it:

```
func hello() string {
    return "hello"
}

func world() string {
    s := "world"
    return s
}

func hello_world() string {
    var s string = hello()+" "+world()
    return s
}

func digo_main() void {
    println("%s", hello_world())
}
```

Then, compile and run this program with master mode

```
$ make build digo=hello-world.digo out=hello-world
$ ./hello-world --master 127.0.0.1:20001
```

Now, you are ready to become a Digoher! Feel free to refer to test codes and sample codes in the appendix (section 9.12 and 9.15) to get more practice with Digo.

Part III. Language Manual

3.1 Lexical Elements

3.1.1 Comment

Comments serve as program documentation. Digo supports two forms of comment:

1. ***Inline comment***: starts with the character sequence // and stops at the end of the line.
2. ***Comment block****:* starts with the character sequence /* and stops with the first subsequent character sequence */.

Comment cannot start inside a literal or inside a comment.

3.1.2 Tokens

Tokens form the vocabulary of Digo. There are five classes: *identifiers*, *keywords*, *operators and punctuation*, and *literals*.

Only white space that separates tokens would combine into a single token, and others (formed from spaces, horizontal tabs, carriage returns, and newlines) are ignored. While breaking the input into tokens, the next token is the longest sequence of characters that form a valid token. Digo hides semicolon delimiters. When the input is broken into tokens, a semicolon is automatically inserted into the token stream immediately after a line's final token.

3.1.3 Identifiers

Identifier in Digo is a case-sensitive ASCII sequence of one or more letters, digits and underscore '_'. The first character in an identifier must be a letter. Identifiers name program entities such as variables and types.

3.1.4 Keywords

Keywords are case-sensitive sequences of letters reserved for use in Digo.The following keywords are reserved and may not be used as identifiers.

```
for, if, else, func, return, await, async, remote,
var, string, int, bool, float, future, void
```

3.1.5 Operators and punctuation

The following character sequences represent operators and punctuation:

```
+    &&    =     !=    (    )
-    ||    <     <=    [    ]
*    ==    >     >=    {    }
/    ++    :=          ,    ;
%    --    !
```

3.1.6 Basic Literals

Literals in Digo are the representation of a value of some primitive types. Basic literals in Digo include integer literals, float literals, character literals, string literals, boolean literals. Digo also implements slice literal, which is introduced in the later part.

3.1.6.1 Integer literals

Sequence of one or more digits in decimal. For representing negative numbers, a negation operator is prefixed.

Example: 12

3.1.6.2 Float literals

Sequence that consists of an integer part, a decimal point and a fraction part. For representing negative numbers, a negation operator is prefixed.

Example: 3.14

3.1.6.3 String literals

Sequence of ASCII characters quoted by double quotes. A string can also be empty.

Example: “Digo is Great!”

3.1.6.4 Boolean literals

Logical true and false.

3.2 Functions

A function is a group of statements that together perform a task, which accepts a set of input and returns a set of output.

A function declaration tells the compiler about a function's name, return type, and parameters, and provides the actual body of the function.

3.3 Variables

A variable is a storage location for holding a *value*. The set of permissible values is determined by the variable's *type*.

A variable declaration reserves storage for a named variable.

A variable's value is retrieved by referring to the variable in an expression; it is the most recent value assigned to the variable. If a variable has not yet been assigned a value, its value is the zero value for its type.

3.4 Types

A type determines a set of values together with operations and methods specific to those values.

```
Type =  "string" | "float" | "bool" | "int" | "future" | SliceType .
```

3.4.1 Boolean type

The predeclared boolean type is bool. A *boolean type* represents the set of Boolean truth values denoted by keywords true and false.

3.4.2 Integer type

An integer type represents sets of integer values. The predeclared type is “int”.

3.4.3 Float type

A float type represents sets of floating numbers. The predeclared type is “float”.

3.4.4 Future type

A future type represents a job(function) that has been sent to a worker for execution. The predeclared type is “future”. A job is a function and its corresponding arguments for a worker to execute.

3.4.5 Slice type

A slice is a sequence container representing arrays that can change in size.

```
SliceType = "[" "]" Type .
```

The length of a slice s can be discovered by the built-in function `len`. The elements can be addressed by integer indices 0 through `len(s)-1`.

Semantic checking prohibited nested slices. In another word, A slice can only store elements with the type among Boolean type, Integer type, Float type and Future type.

If a slice is of future type, then the user should make sure future objects in this slice are of the same kind of job. Otherwise the behavior is undefined.

3.5. Properties of types and values

3.5.1 Type Identity

Two types are either identical or different. For `string`, `float`, `int`, `bool`, `future`, each of these types is always different from any other type.

For `slice` types, two slice types are identical if and only if they have identical element types.

3.5.2 Assignability

A value x is *assignable* to a variable of type T ("x is assignable to T") iff x's type is identical to T.

3.6. Declarations

A *declaration* binds a non-blank identifier to a variable or a function. Every identifier in a program must be declared.

3.6.1 Function declaration

A function declaration tells the compiler about a function's name, return type, and parameters, and provides the actual body of the function.

```
FunctionDecl = FunctionAnnotation "func"  FunctionName Signature
FunctionBody .
FunctionName = identifier .
FunctionBody = "{" StatementList "}"
Signature = Parameters [ Result ] .
Result = Type | "(" Type { "," Type } ")" .
Parameters = ParameterDecl { "," ParameterDecl } .
ParameterDecl = Identifier Type .
```

A function can only be declared in the global context. No function can be declared inside another function. No function identifier can be declared twice.

3.6.2 Variable declaration

A variable declaration creates one variable, binds the corresponding identifier to them, and gives it a type and an initial value.

A variable can only be declared inside a function. No variable identifier can be declared twice in the same function.

```
VarDecl = "var" VarSpec .
VarSpec = IdentifierList Type [ "=" ExpressionList ] .
```

Examples:

```
var i int = 0
var b string = a + b
var x, y int = 1 , 2
```

3.6.2.1 Short variable declaration

A *short variable declaration* uses the syntax:

```
ShortVarDecl = IdentifierList ":=" ExpressionList .
```

It is shorthand for a regular variable declaration with initializer expression but no type:

```
"var" IdentifierList = ExpressionList .
```

Examples:

```
I := 100
I, j := 0, 10
```

3.7. Expressions

An expression specifies the computation of a value by applying operators and functions to operands.

3.7.1 Operands

Operands denote the elementary values in an expression. An operand may be a literal, an identifier denoting a variable, or a parenthesized expression.

```
Operand     = Literal | OperandName | "(" Expression ")" .
Literal     = BasicLit | SliceLit .
BasicLit    = int_lit | float_lit | string_lit  |  bool_lit .
OperandName = identifier .
```

3.7.1.1 Slice literals

Slice literals construct values for slices.

```
SliceLit = SliceType LiteralValue .
LiteralValue  = "{" [ ElementList [ "," ] ] "}" .
ElementList   = Element { "," Element } .
Element       = Expression | LiteralValue .
```

Each element has an associated integer index marking its position in the slice. An element uses the previous element's index plus one. The first element's index is zero.

3.7.2 Expressions

Expressions are the operands for unary and binary expressions.

```
Expression =
       Operand |
       Expression Index |
       Expression Slice |
       Expression Arguments |
       "Await" Expression |
       Unary_op  |
```

```
        Expression binary_op Expression |
        EmptyExpression .

Index  = "[" Expression "]" .
Slice    = "[" [ Expression ] ":" [ Expression ] "]"
Arguments   = "(" [ ( ExpressionList | Type [ "," ExpressionList ] )
[ "..." ] [ "," ] ] ")" .
ExpressionList = Expression { "," Expression } .
```

Examples:

```
x
2
(s + ".txt")
f(3.1415, true)
s[i : j + 1]
p[10]
```

3.7.3 Empty Expression

Empty expression does nothing.

```
EmptyExpression = .
```

3.7.4 Index Expression

An expression of the form a[x] denotes the element of the slice. The following rules apply:

- if x is out of range at run time, the program sends an segment fault to itself and therefore exits.
- a[x] is the slice element at index x and the type of a[x] is the element type of S

3.7.5 Slice Expression

For a slice `s`, the expression `s[low : high]` constructs a subslice. The indices `low` and `high` select which elements of operand a appear in the result. The result has indices starting at 0 and length equal to `high` - `low`.

For convenience, one of the indices may be omitted. A missing `low` index defaults to zero; a missing `high` index defaults to the length of the sliced operand:

```
a[2:]  // same as a[2 : len(a)]
a[:3]  // same as a[0 : 3]
```

The indices are *in range* if 0 <= low <= high < len(a), otherwise they are *out of range*. If the indices are out of range at run time, a segment fault occurs.

3.7.6 Function Call

Given a function `f` of function prototype `F`,

```
f(a1, a2, ... an)
```

Calls `f` with arguments `a1, a2, ...an`. Arguments must be single-valued expressions assignable to the parameter types of F and are evaluated before the function is called. The type of the expression is the result type of F.

In a function call, the function value and arguments are evaluated in the usual order. After they are evaluated, if the types of the parameters are then they are passed by value to the function and the called function begins execution.

A special built-in function call is `await`. When calling the built-in function `await(future-obj)`, open should use `await future-obj` to call this built-in function, instead of the normal function call syntax.

3.7.7 Operators

Operators combine operands into expressions.

```
binary_op = "||" | "&&" | rel_op | add_op | mul_op .
rel_op = "==" | "!=" | "<" | "<=" | ">" | ""=" .
add_op = "+" | "-" .
mul_op = "*" | "/" | "%" .
assign_op = "=" .
unary_op = "-" | "!" .
```

The associativity of operators are as follows:

Operator	Name	Associativity
=	Assign	Right
==	Equal to	Left
!=	Unequal to	Left
>	Greater than	Left
>=	Greater than or equal	Left
<	Less than	Left
<=	Less than or equal to	Left
+	Addition	Left
-	Subtraction	Left
*	Multiplication	Left
/	Division	Left
%	Modulo	Left
&&	Logical and	Left
||	Logical or	Left
!	Logical not	Right

The precedence of operators is as follows:

1. * / %
2. + -
3. > >= < <=
4. == !=
5. !
6. &&
7. ||
8. ,
9. =

3.7.7.1 Arithmetic Operators

Arithmetic operators apply to numeric values and yield a result of the same type as the first operand. The four standard arithmetic operators (+, -, *, /) apply to integer, floating-point, and complex types; + also applies to strings.

```
+    sum                    integers, floats, strings
-    difference             integers, floats
*    product                integers, floats
/    quotient               integers, floats
%    remainder              integers
```

3.7.1.1.1 String Concatenation

Strings can be concatenated using the + operator:

```
s := "hi" + s2
```

3.7.7.2 Comparison Operators

Comparison operators compare two operands and yield an untyped boolean value.

```
==    equal
!=    not equal
<     less
<=    less or equal
>     greater
```

```
>=    greater or equal
```

3.7.7.3 Logical Operators

Logical operators apply to boolean values and yield a result of the same type as the operands. The right operand is evaluated conditionally.

```
&&    conditional AND    p && q  is  "if p then q else false"
||    conditional OR     p || q  is  "if p then true else q"
!     NOT                !p      is  "not p"
```

3.7.8 Order of Evaluation

when evaluating the operands of an expression, assignment, or return statement, all function calls, are evaluated in lexical left-to-right order.

3.8 Statements

Statements control execution.

```
Statement =
       FunctionDecl | VarDecl | EmptyStmt | Expression | ShortVarDecl |
       ReturnStmt | IfStmt | ForStmt  .
```

3.8.1 Terminating Statements

A *terminating statement* prevents execution of all statements that lexically appear after it in the same block. The following statements are terminating:

1. A "return" statement
2. An "if" statement in which the "else" branch is present, and both branches are terminating statements.
3. A "for" statement in which the loop condition is absent.

3.8.2 If Statements

"If" statements specify the conditional execution of two branches according to the value of a boolean expression. If the expression evaluates to true, the "if" branch is executed, otherwise, if present, the "else" branch is executed.

```
IfStmt = "if" "(" Expression ")" "{" StatementList "}" [ "else" ( IfStmt |
"{" StatementList "}" ) ] .
StatementList = { Statement ";" } .
```

Example:

```
If (x > max) {
      x = max
}
```

3.8.3 For Statements

A "for" statement specifies repeated execution of a statement list.

```
ForStmt = "for"  ForClause  "{" StatementList "}" .
ForClause = "(" Expression ";" Expression ";" Expression ")" .
```

The init statement is executed once before evaluating the condition for the first iteration; the post statement is executed after each execution of the a statement list (and only if the the statement list was executed). Any element of the ForClause may be empty but the semicolons are required unless there is only a condition. If the condition is absent, it is equivalent to the boolean value true.

3.8.4 Return Statements

A "return" statement in a function F terminates the execution of F, and optionally provides one or more result values.

```
ReturnStmt = "return" [ ExpressionList ] .
```

In a function without a result type, a "return" statement must not specify any result values.

```
func noResult() {
       return
```

```
}
```

One exception is that `digo_main` function must explicitly specify the result type as “void”

3.9 Built-in Functions

Function Prototype	Return Result
len(Slice) int	Returns the length of a slice object
len(String) int	Returns the length of a string object
append(Slice, element) Slice	Returns a new slice that contains elements in the first argument slice plus the element at the end of the new slice
await(Future) Any	Get the return result of a remote job corresponding to a future object
read(String) Slice	Return a slice of strings containing all the words in a file specified by a path specified by the string argument
pintln(sting, arg1, arg2, …)	Print a string with variable arguments to the terminal device.

3.10 Program Execution

3.10.1 Master

```
./executable --master ip:port
```

If the program is a master, the program starts listening to <ip:port> and waits for other workers to send a join request and then invoking the function `digo_main`. When that function invocation returns, the program exits.

3.10.2 Worker

```
./executable --worker MasterIP:MasterPort WorkerIP:WorkerPort
```

If the program is a worker, the program starts listening to <WorkerIP:WorkerPort>. Then the worker sends a join request to the master(the hostname and port of which are gotten from the cmdline arguments). Then the worker keeps alive and waits for remote tasks from the master program, executes it and returns the result back to the master.

Part IV. Project Plan

4.1 Process used for planning, specification, development and testing

4.1.1 Planning and specification

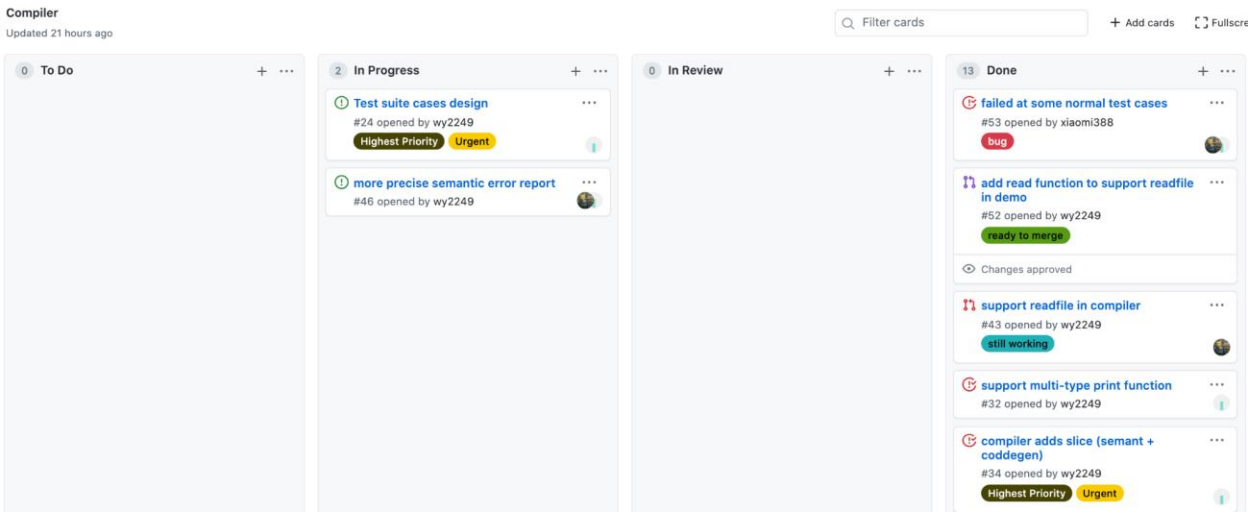
We meet once a week over zoom to check in each member's progress and make sure that everyone is on the right track and on the same page. During our team meeting, each of us provided an update on what they were working on currently, discussed design/implementation details and made decisions together.

After ensuring that we were constantly making forward progress on our language, we made plans on tasks to do for the next week. Wenqian, our manager, recorded all the tasks in the project timeline table, opened github issues for each task and assigned corresponding tasks to each member. During the week, members also updated their output and communicated under that github issue.

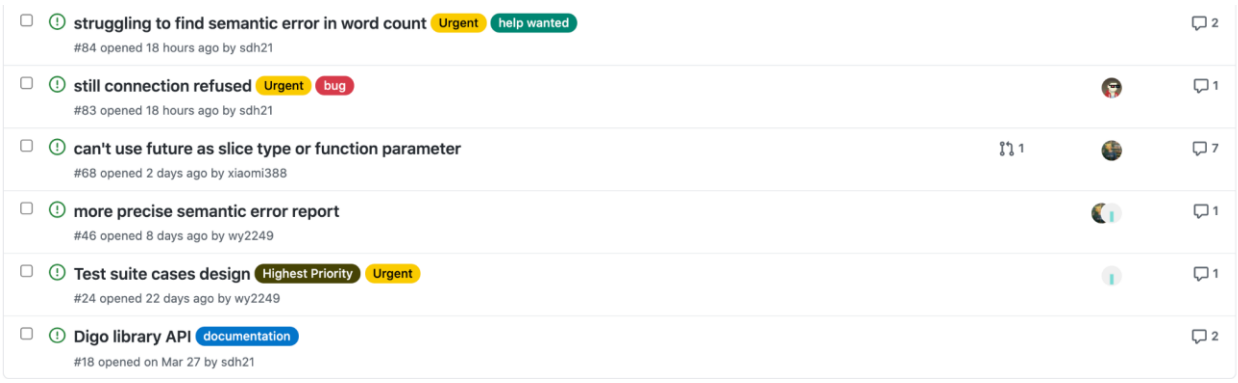
4.1.2 Development

To facilitate collaboration, we used github project board for agile development and updated progress for features/bugs on github issues. We also labeled issues to give them different priority levels and make it easy to track progress.

As digo has a pipelined architecture among compiler, linker and lib, we also helped each other troubleshoot any issues that we had and talked about options for resolving any bugs in our code. Finally, throughout the week, we communicated over wechat if we had any pressing concerns or questions about our language implementation.



Github project Board with 4 boards (To Do, In Progress, In Review, Done)

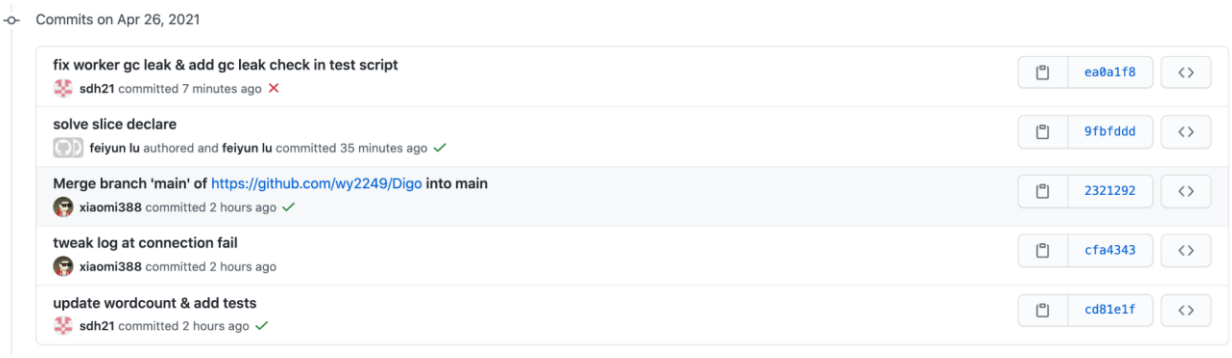


Github Issues with labels

4.1.3 Testing

We insisted that everyone should write test cases for all of the features they were working on to prove that everything was working as expected.

After most of the built-in functions are completed, we added an automated integration test with CircleCI to our github repo so that every commit can trigger a build and run all the tests automatically. This automated testing drastically improved test experience and brought convenience to visualize the effect of each commit.



circle ci automatically runs test on each commit and shows whether it passed on commit log

4.2 Programming style guide

The codebase that is used to implement Digo is mostly OCaml (for compiler) and C++ code (for library). We tried to write clean, readable, modular, OCaml code.

We often followed these general guidelines when writing our compiler:

1. Indent clearly. This is very important to make our ocaml code readable.
2. Commit frequently.
3. Use descriptive function names to make it easier to understand the code.
4. All AST types are camel case and all SAST types are the related AST type with a capital ‘S’ in front.
5. Remove all errors before submitting a pull request.
6. Write robust test cases for any code modifications and try to find both positive (passing test cases ie. if something should work, that it works) and negative (failure test cases ie. if something should fail, that it fails) test cases.
7. Write comments whenever things are unclear or if comments would be helpful to future readers.
8. Simplify programs when and if possible.

4.3 Project Timeline

Week	Task	Milestone
2/21-2/27	initial end-to-end scanner, parser, ast, printer	LMR, Parser
2/28-3/6	Solve issue "shift/reduce" conflict in parser Lib: Runtime library and interface	
3/7-3/13	Spring Break	

3/14-3/20	Compiler: ● initial end-to-end codegen ● initial end-to-end semant Lib: Runtime library and interface Linker: initial linker	
3/21-3/27	Semant: add expression consistency check Codegen: implement three print functions for Hello World	Hello World
3/28-4/3	Semant: add statement check Codegen: ● Control flow: if, for, return ● Expression: UnaryOp, BinaryOp, Function call	
4/4-4/10	Compiler: ● support declaration (not limited to beginning of function) ● support short declare ● support multi-value return from function call ● support built-in function related to string Lib: ● support string ● support slice Linker: ● multi-value return from function call ● Digo Object base class and GC APIs	
4/11-4/17	Compiler: ● built-in function-call related to future ● built-in function-call related to read ● built-in functions related to slice ● built-in print function (support all types) ● bug: return void (cannot be {void}) ● add digo.ml for compilation Lib: support file read Test suite: ● add test script ● add more test cases	
4/18-4/25	Compiler: ● fix bugs from tests and error cleanup ● remove all the warnings	

	Linker: finish GC Test suite: ● add more test cases ● add automated test CI (circle ci)	
4/26		Project Due

4.4 Member roles and responsibilities

Wenqian Yan, Manager
Sida Huang, System Architect
Yufan Chen, Language Guru
Hanxiao Lu, Test Designer

Every member of the team contributed to ast and parser in compiler, test suite as well as the writing of proposal, LRM and Final Report. After we planned on our whole architecture, we divided roles so that Wenqian and Hanxiao worked on compiler implementation, Yufan implemented libraries and Sida worked on linker. We collaborated on many features, doing different parts and helping each other out when stuck.

This was the general division-of-labor:

Yufan Chen:
- Network library for data transfer between workers and master
- Master-Worker layer for tasks distribution and execution
- Test Cases
- Slice and String implementation in library
- Continuous integration
- Read in library

Hanxiao Lu:
- Initial end-to-end codegen
- Control flow in compiler
- Slice in compiler
- Multi-return from function call in compiler
- Variable argument function call like Print function call in compiler
- Test cases

Sida Huang:
- Serialization library
- Async/async remote abstraction layer (Digo Linker)

- Reference count based GC
- Print function similar to printf; support built-in slice and string
- Test script

Wenqian Yan:

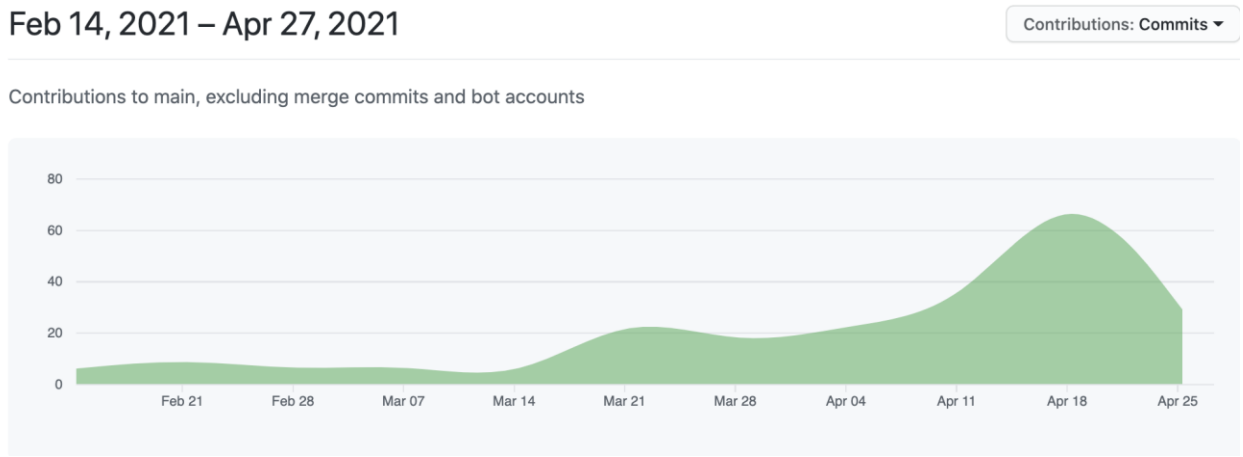
- Short declare in compiler
- String and its operations in compiler
- Future (including async/async remote functions and await) in compiler
- Slice of future in compiler
- Read in compiler
- Pretty print in ast and sast

4.5 Software development environment

- Operating Systems: MacOS, Ubuntu 20.04
- Languages: OCaml, C++
- Text Editor: VS Code
- Version Control: GitHub
- Documentation: Lark, Google Docs, Github Issue
- Agile Development: Github Project Board
- Continuous Integration: Circle CI

4.6 Project log

4.6.1 Commit overview



4.6.2 Commit log

We have over 300 commits. Please see appendix (section 9.1) for a complete commit log listing.

Part V. Architecture

5.1 Design

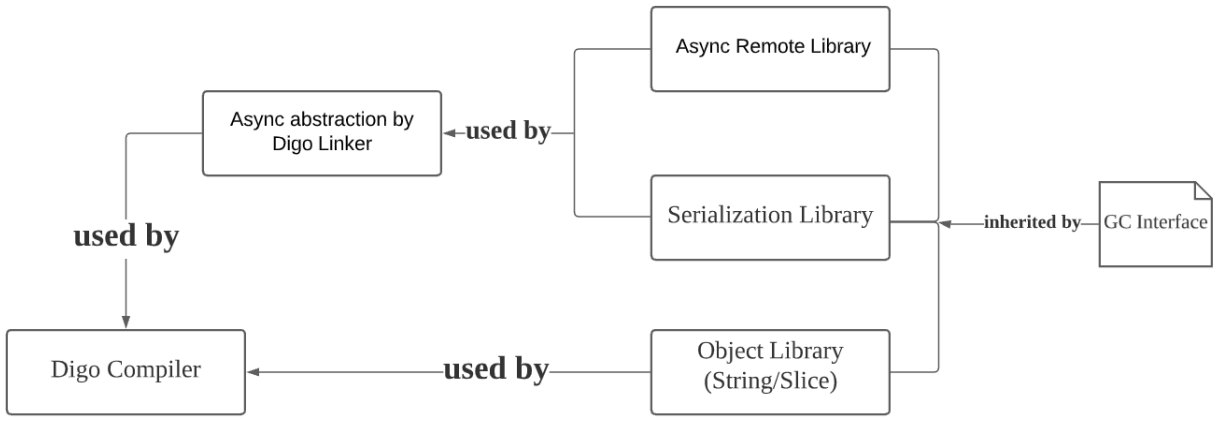
Theoretically, all things including string, slice, serialization could have been done in the code generator written in OCaml, as long as functions like malloc/free/pthread_create and network libraries are linked in.

However, if we actually did that, the code generator will be too large for us to debug.

Therefore, we want each component of our project to be light-weight. That is to say, the code generator written in OCaml will not be calling malloc, doing serialization, creating new threads and so on. Instead, many interfaces will be provided to this code generator, and the code generated by it can directly call these APIs as LLVM IR functions.

5.2 Main Components

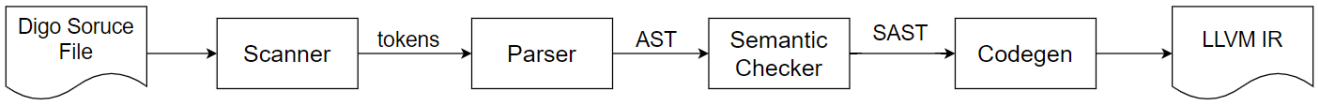
Our overall architecture is broken into multiple components.



5.2.1 Digo Compiler

(implemented by Wenqian Yan, Hanxiao Lu)

The Digo Compiler focuses on lexing, parsing, generating AST, doing semantic check and translating AST into LLVM IR.



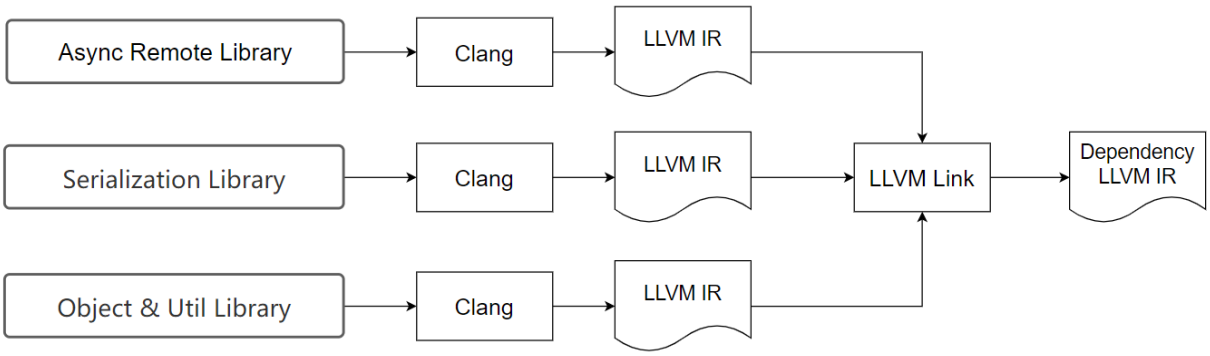
5.2.2 Digo Library

(implemented by Yufan Chen, Sida Huang)

There are several light-weight Digo libraries to provide different functionalities.

(1) The Digo Async Remote Library provides the async remote service, through which the caller can run a master or a worker, and send bytes between masters and workers.
(2) The Digo Object Library provides GC-compatible string/slice objects. All the objects in this library are inherited from a GC base class. This GC base class implements general reference count operations and also provides GC interfaces to other libraries and the Digo compiler.
(3) The Digo Serialization Library provides serialization methods for each Digo type.
(4) The Digo Util Library includes util functions like FileReader and our own Printf.

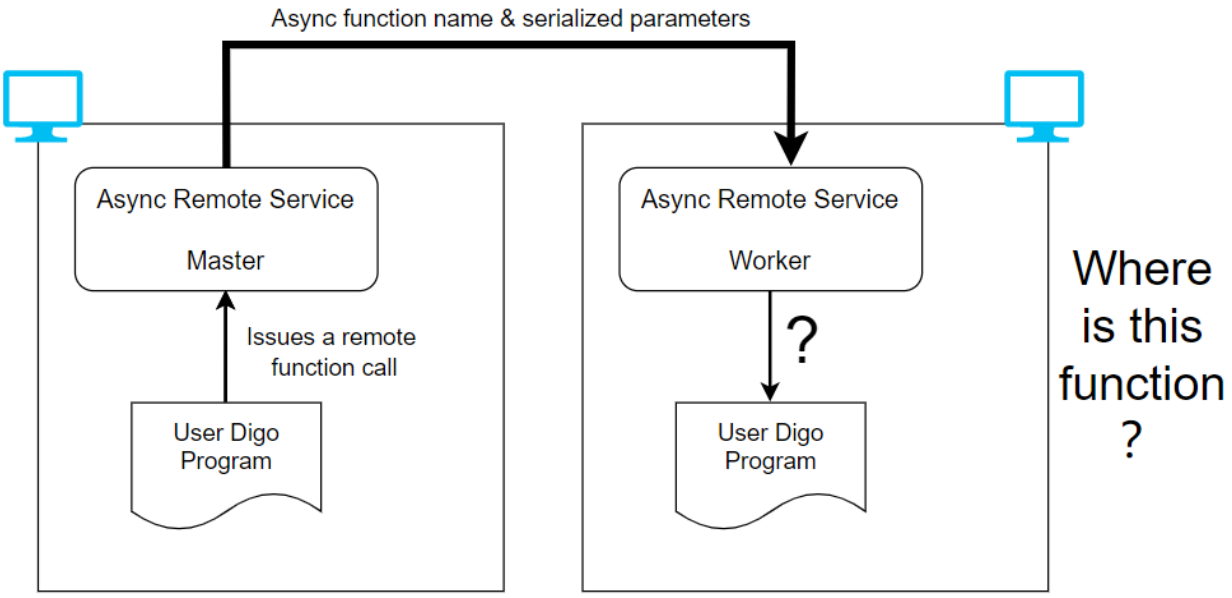
Digo libraries are implemented in C++, and the APIs are exported as LLVM IR functions.



5.2.3 Digo Linker

(implemented by Sida Huang)

The main purpose of the Digo Linker is to provide a function jump table, so the Async Remote Service can call a Digo user function by having its name and serialized parameters.



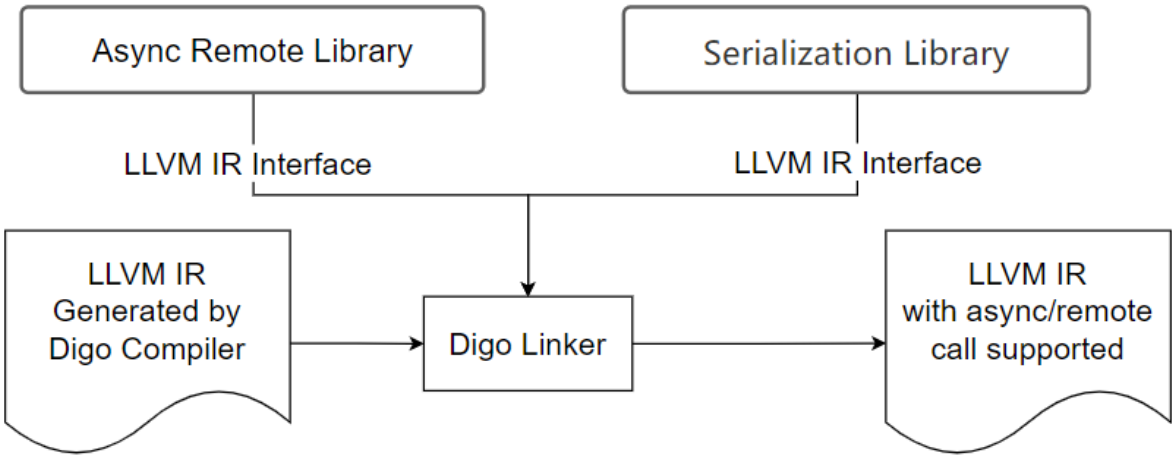
Consider this example: the (code generated by) Digo Compiler issues a remote function call, and on the remote machine, the Async Remote Library receives this request and has to call a Digo user function. How can it locate and call the Digo function by only having the function name and the serialized arguments? The Async Remote library implemented independently never had the idea about a Digo user function (in the form of LLVM IR) generated by Digo Compiler.

This interface could have been implemented in Digo Compiler at compile time; more specifically, the Digo Compiler could export an LLVM IR function to be called by the Async Remote Library to locate a Digo user function. However, this burdens the Digo Compiler, and the boundary is not clear.

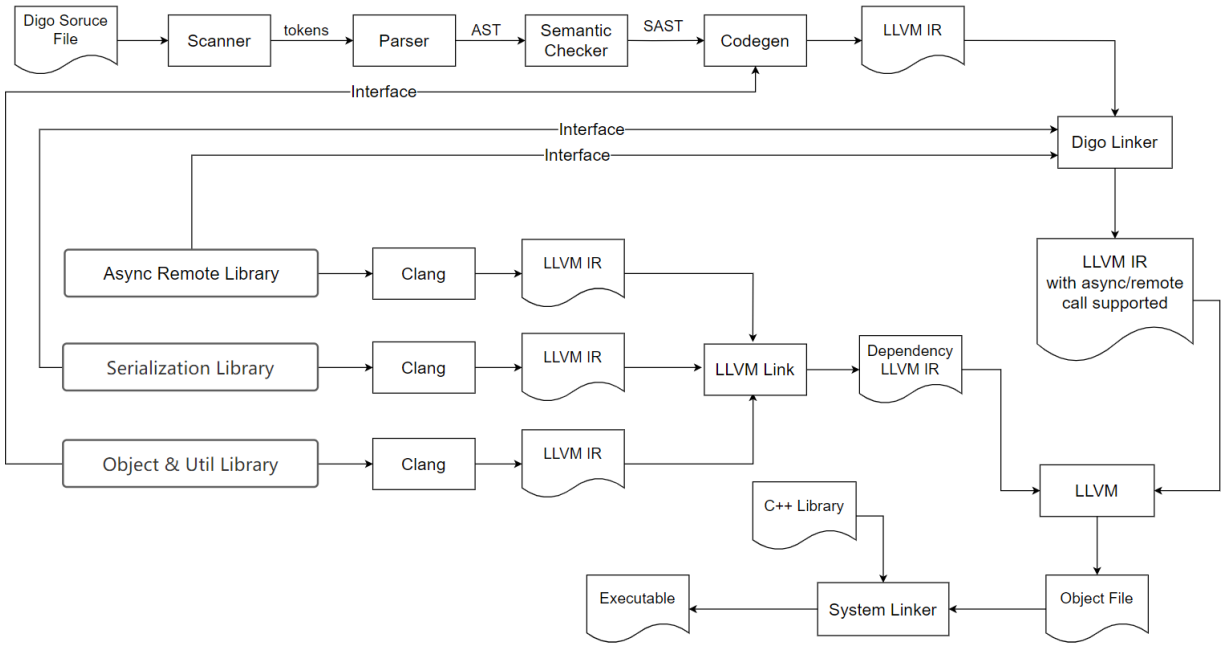
So here comes the Digo Linker. This idea is borrowed from LLVM IR Pass. The Digo Linker takes in the LLVM IR generated by Digo Compiler, and generates all the codes necessary for the above purpose.

But the Digo Linker does more. Serialization & deserialization is a complex task, so the Digo Linker hides all the serialization details behind calling a remote function, and exports simpler interfaces to the Async Remote Library, as well as the Digo Compiler. This is reasonable, because the Async Remote Library also does not know how to de-serialize the arguments and serialize return values, without having prototypes of Digo functions.

All of the above are hidden by two APIs: one for the compiler to issue a remote function call for a specific Digo function, and another one for the Async Remote Library to call a Digo function and get serialized return value.



5.3 The whole picture



Our project relies heavily on LLVM IR. LLVM IR is a wonderful intermediate language, and using LLVM IR brings us some obvious merits:

a) LLVM IR is strongly typed so we can find all type-inconsistency bugs at compile time.
b) It's human readable, so it's good for debugging.
c) It provides function call abstraction, so we can base our interfaces on LLVM IR. Each exported API described above is just a function in LLVM IR.

Part VI. Test Plan

6.1 Test Suite and Automation

6.1.1 Why do I choose the three examples in section 6.2?

The three examples chosen below (section 6.2) include the core functionality in digo language. To be specific, digo has index accessing, slicing, appending items and evaluating length on slicing objects in example 1. In example 2, digo has appending and evaluating length on string objects. In example 3, digo has multiple return and await functionality, which could handle the case that user code sends tasks to workers.

6.1.2 Test Suite explain

Our group divides tests into eight categories and each category defines a specific domain to be tested. These categories are Async, Basic, ControlFlow, GC, Remote, Semantic, Syntax and Utils. Async and Remote share some similarities which we are using to test our future object and built-in function type. Basic is testing whether the basic functionality like binary operation, unary operation, function call work. ControlFlow is testing if and for statements. GC is testing garbage collection. Semantic is composed of a number of files with semantic errors. Util is testing print and println functionality.

6.1.3 Test Automation explain

We have a lot of test files so we decide to write a test script run_all.sh, which not only runs all the tests files but also builds up a log file that compares the results of expected value with system output. If the test passes, it shows passed. If the test fails, it shows the reasons for failure and expected outputs.

A part of Failure running test in log file looks like this:

```
##### Testing test-add1
 ## Executable output:
42
diff -b test-add1.exec.output /home/microc/digo-test/tests/test-
add1.digo.pass.expected > test-add1.diff.output
 ## Diff output:
##### Test test-add1: Passed
```

```
##### Testing test-arith1
 ## Executable output:
42
diff -b test-arith1.exec.output /home/microc/digo-test/tests/test-
arith1.digo.pass.expected > test-arith1.diff.output
 ## Diff output:
##### Test test-arith1: Passed

##### Testing test-arith2
 ## Executable output:
11
diff -b test-arith2.exec.output /home/microc/digo-test/tests/test-
arith2.digo.pass.expected > test-arith2.diff.output
 ## Diff output:
##### Test test-arith2: Passed
```

A part of Success running test in log files looks like this:

```
##### Testing test-fib
 ## Build output:
Integer arithmetic operators only work with integral types!
  %tmp8 = add { i64 } %fib_result, %fib_result7
Invalid InsertValueInst operands!
  %mrv = insertvalue { i64 } undef, { i64 } %tmp8, 0
LLVM ERROR: Broken module found, compilation aborted!
 ## Compilation failed, but we cannot find /home/microc/digo-
test/tests/test-fib.digo.fail.expected
##### Test test-fib: Failed
##### Testing test-float1
 ## Executable output:
3.141593
diff -b test-float1.exec.output /home/microc/digo-test/tests/test-
float1.digo.pass.expected > test-float1.diff.output
FAILED test-float1.exec.output differs from /home/microc/digo-
test/tests/test-float1.digo.pass.expected
 ## Diff output:
1c1
< 3.141593
---
> 3.14159267
##### Test test-float1: Failed
```

```
##### Testing test-float2
 ## Executable output:
0.423313
diff -b test-float2.exec.output /home/microc/digo-test/tests/test-
float2.digo.pass.expected > test-float2.diff.output
FAILED test-float2.exec.output differs from /home/microc/digo-
test/tests/test-float2.digo.pass.expected
 ## Diff output:
1c1,2
< 0.423313
---
> 0.42331267
> 0.42331267
\ No newline at end of file
##### Test test-float2: Failed
```

Our test script code is shown in Appendix(see section 9.12.1).

6.2 Source To Target

6.2.1 Source program example 1

```
func digo_main() void {
  a := []int {1,2,3,4}
  b := append(a,10)
  c := len(b)
  var d []int
  d = b
  println("%d",b[0])
  println("%l",b[1:2])
  println("%d",c)
  println("%l",d)
}
```

Target result:

```
ModuleID = 'Digo'
source_filename = "Digo"
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

@createstr_ptr = private unnamed_addr constant [3 x i8] c"%d\00"
@createstr_ptr.1 = private unnamed_addr constant [3 x i8] c"%l\00"
@createstr_ptr.2 = private unnamed_addr constant [3 x i8] c"%d\00"
@createstr_ptr.3 = private unnamed_addr constant [3 x i8] c"%l\00"

declare void @print(i8*, ...)

declare void @println(i8*, ...)

declare i8* @CreateString(i8*)

declare i8* @CreateEmptyString()

declare i8* @AddString(i8*, i8*)

declare i64 @CompareString(i8*, i8*)

declare i8* @CloneString(i8*)

declare i64 @GetStringSize(i8*)

declare i8* @CreateSlice(i64)

declare i8* @SliceAppend(i8*, ...)

declare i8* @SliceSlice(i8*, i64, i64)

declare i64 @GetSliceSize(i8*)

declare double @SetSliceIndexDouble(i8*, i64, double)

declare i8* @SetSliceIndexFuture(i8*, i64, i8*)

declare i8* @SetSliceIndexString(i8*, i64, i8*)

declare i64 @SetSliceIndexInt(i8*, i64, i64)

declare double @GetSliceIndexDouble(i8*, i64)
```

```
declare i8* @GetSliceIndexFuture(i8*, i64)

declare i8* @GetSliceIndexString(i8*, i64)

declare i64 @GetSliceIndexInt(i8*, i64)

declare i8* @CloneSlice(i8*)

declare i8* @ReadFile(i8*)

declare i8* @__GC_CreateTraceMap()

declare void @__GC_Trace(i8*, i8*)

declare void @__GC_NoTrace(i8*, i8*)

declare void @__GC_ReleaseAll(i8*)

define void @digo_main() {
entry:
  %gc_trace_map_obj = call i8* @__GC_CreateTraceMap()
  %a = alloca i8*
  %createslice = call i8* @CreateSlice(i64 3)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createslice)
  %initslicen = call i8* (i8*, ...) @SliceAppend(i8* %createslice, i64 1)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %initslicen)
  %initslicen1 = call i8* (i8*, ...) @SliceAppend(i8* %initslicen, i64 2)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %initslicen1)
  %initslicen2 = call i8* (i8*, ...) @SliceAppend(i8* %initslicen1, i64 3)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %initslicen2)
  %initslicen3 = call i8* (i8*, ...) @SliceAppend(i8* %initslicen2, i64 4)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %initslicen3)
  %cloneslice = call i8* @CloneSlice(i8* %initslicen3)
  store i8* %cloneslice, i8** %a
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %cloneslice)
  %b = alloca i8*
  %a4 = load i8*, i8** %a
  %slice_append = call i8* (i8*, ...) @SliceAppend(i8* %a4, i64 10)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %slice_append)
  %cloneslice5 = call i8* @CloneSlice(i8* %slice_append)
  store i8* %cloneslice5, i8** %b
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %cloneslice5)
```

```
  %c = alloca i64
  %b6 = load i8*, i8** %b
  %slicelen = call i64 @GetSliceSize(i8* %b6)
  store i64 %slicelen, i64* %c
  %d = alloca i8*
  %b7 = load i8*, i8** %b
  %cloneslice8 = call i8* @CloneSlice(i8* %b7)
  store i8* %cloneslice8, i8** %d
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %cloneslice8)
  %createstr = call i8* @CreateString(i8* getelementptr inbounds ([3 x i8],
[3 x i8]* @createstr_ptr, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr)
  %b9 = load i8*, i8** %b
  %findsliceindexn = call i64 @GetSliceIndexInt(i8* %b9, i64 0)
  call void (i8*, ...) @println(i8* %createstr, i64 %findsliceindexn)
  %createstr10 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.1, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr10)
  %b11 = load i8*, i8** %b
  %SliceSlice = call i8* @SliceSlice(i8* %b11, i64 1, i64 2)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %SliceSlice)
  call void (i8*, ...) @println(i8* %createstr10, i8* %SliceSlice)
  %createstr12 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.2, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr12)
  %c13 = load i64, i64* %c
  call void (i8*, ...) @println(i8* %createstr12, i64 %c13)
  %createstr14 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.3, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr14)
  %d15 = load i8*, i8** %d
  call void (i8*, ...) @println(i8* %createstr14, i8* %d15)
  call void @__GC_ReleaseAll(i8* %gc_trace_map_obj)
  ret void
}
```

6.2.2 Source program example 2

```
func digo_main() void {
      a := "abc"
      var b string
```

```
	b = a
	c := len(b)
	d := a+b
	println("%d",c)
	println("%s",d)
	println("%d",a == d)
}
```

Target result:

```
ModuleID = 'Digo'
source_filename = "Digo"
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

@createstr_ptr = private unnamed_addr constant [4 x i8] c"abc\00"
@createstr_ptr.1 = private unnamed_addr constant [3 x i8] c"%d\00"
@createstr_ptr.2 = private unnamed_addr constant [3 x i8] c"%s\00"
@createstr_ptr.3 = private unnamed_addr constant [3 x i8] c"%d\00"

declare void @print(i8*, ...)

declare void @println(i8*, ...)

declare i8* @CreateString(i8*)

declare i8* @CreateEmptyString()

declare i8* @AddString(i8*, i8*)

declare i64 @CompareString(i8*, i8*)

declare i8* @CloneString(i8*)

declare i64 @GetStringSize(i8*)

declare i8* @CreateSlice(i64)

declare i8* @SliceAppend(i8*, ...)

declare i8* @SliceSlice(i8*, i64, i64)
```

```
declare i64 @GetSliceSize(i8*)

declare double @SetSliceIndexDouble(i8*, i64, double)

declare i8* @SetSliceIndexFuture(i8*, i64, i8*)

declare i8* @SetSliceIndexString(i8*, i64, i8*)

declare i64 @SetSliceIndexInt(i8*, i64, i64)

declare double @GetSliceIndexDouble(i8*, i64)

declare i8* @GetSliceIndexFuture(i8*, i64)

declare i8* @GetSliceIndexString(i8*, i64)

declare i64 @GetSliceIndexInt(i8*, i64)

declare i8* @CloneSlice(i8*)

declare i8* @ReadFile(i8*)

declare i8* @__GC_CreateTraceMap()

declare void @__GC_Trace(i8*, i8*)

declare void @__GC_NoTrace(i8*, i8*)

declare void @__GC_ReleaseAll(i8*)

define void @digo_main() {
entry:
  %gc_trace_map_obj = call i8* @__GC_CreateTraceMap()
  %a = alloca i8*
  %createstr = call i8* @CreateString(i8* getelementptr inbounds ([4 x i8],
[4 x i8]* @createstr_ptr, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr)
  %clonestr = call i8* @CloneString(i8* %createstr)
  store i8* %clonestr, i8** %a
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %clonestr)
  %b = alloca i8*
  %a1 = load i8*, i8** %a
  %clonestr2 = call i8* @CloneString(i8* %a1)
```

```
  store i8* %clonestr2, i8** %b
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %clonestr2)
  %c = alloca i64
  %b3 = load i8*, i8** %b
  %str_len = call i64 @GetStringSize(i8* %b3)
  store i64 %str_len, i64* %c
  %d = alloca i8*
  %a4 = load i8*, i8** %a
  %b5 = load i8*, i8** %b
  %addstr = call i8* @AddString(i8* %a4, i8* %b5)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %addstr)
  %clonestr6 = call i8* @CloneString(i8* %addstr)
  store i8* %clonestr6, i8** %d
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %clonestr6)
  %createstr7 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.1, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr7)
  %c8 = load i64, i64* %c
  call void (i8*, ...) @println(i8* %createstr7, i64 %c8)
  %createstr9 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.2, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr9)
  %d10 = load i8*, i8** %d
  call void (i8*, ...) @println(i8* %createstr9, i8* %d10)
  %createstr11 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.3, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr11)
  %a12 = load i8*, i8** %a
  %d13 = load i8*, i8** %d
  %cmpstr = call i64 @CompareString(i8* %a12, i8* %d13)
  %cmpstr_bool = icmp eq i64 %cmpstr, 0
  %booleantoint = select i1 %cmpstr_bool, i64 1, i64 0
  call void (i8*, ...) @println(i8* %createstr11, i64 %booleantoint)
  call void @__GC_ReleaseAll(i8* %gc_trace_map_obj)
  ret void
}
```

6.2.3 Source program example 3

```
func digo_main() void {
    s := []future{}
    a := 10
```

```
    b := 20
    c := try_return_one(a)
    e := try_return_one(b)
    s = append(s, c)
    s = append(s, e)

    fu := s[1]
    g := await fu

    println("%d", g)

    fu = s[0]
    g1 := await fu
    println("%d", g1)

}

async func try_return_one(a int) int {
    return a+10
}
```

Target result:

```
ModuleID = 'Digo'
source_filename = "Digo"
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

@createstr_ptr = private unnamed_addr constant [3 x i8] c"%d\00"
@createstr_ptr.1 = private unnamed_addr constant [3 x i8] c"%f\00"

declare void @print(i8*, ...)

declare void @println(i8*, ...)

declare i8* @CreateString(i8*)

declare i8* @CreateEmptyString()

declare i8* @AddString(i8*, i8*)
```

```
declare i64 @CompareString(i8*, i8*)

declare i8* @CloneString(i8*)

declare i64 @GetStringSize(i8*)

declare i8* @CreateSlice(i64)

declare i8* @SliceAppend(i8*, ...)

declare i8* @SliceSlice(i8*, i64, i64)

declare i64 @GetSliceSize(i8*)

declare double @SetSliceIndexDouble(i8*, i64, double)

declare i8* @SetSliceIndexFuture(i8*, i64, i8*)

declare i8* @SetSliceIndexString(i8*, i64, i8*)

declare i64 @SetSliceIndexInt(i8*, i64, i64)

declare double @GetSliceIndexDouble(i8*, i64)

declare i8* @GetSliceIndexFuture(i8*, i64)

declare i8* @GetSliceIndexString(i8*, i64)

declare i64 @GetSliceIndexInt(i8*, i64)

declare i8* @CloneSlice(i8*)

declare i8* @ReadFile(i8*)

declare i8* @__GC_CreateTraceMap()

declare void @__GC_Trace(i8*, i8*)

declare void @__GC_NoTrace(i8*, i8*)

declare void @__GC_ReleaseAll(i8*)

define void @digo_main() {
```

```
entry:
  %gc_trace_map_obj = call i8* @__GC_CreateTraceMap()
  %a = alloca i64
  store i64 10, i64* %a
  %b = alloca double
  store double 1.000000e+00, double* %b
  %c = alloca i8*
  %a1 = load i64, i64* %a
  %try_return_one_result = call i8*
@digo_linker_async_call_func_try_return_one(i64 %a1)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %try_return_one_result)
  store i8* %try_return_one_result, i8** %c
  %e = alloca i8*
  %a2 = load i64, i64* %a
  %b3 = load double, double* %b
  %try_return_two_result = call i8*
@digo_linker_async_call_func_try_return_two(i64 %a2, double %b3)
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %try_return_two_result)
  store i8* %try_return_two_result, i8** %e
  %f = alloca i64
  %g = alloca double
  %e4 = load i8*, i8** %e
  %await_try_return_two_result = call { i64, double }
@digo_linker_await_func_try_return_two(i8* %e4)
  %extracted_value = extractvalue { i64,
double } %await_try_return_two_result, 0
  store i64 %extracted_value, i64* %f
  %extracted_value5 = extractvalue { i64,
double } %await_try_return_two_result, 1
  store double %extracted_value5, double* %g
  %createstr = call i8* @CreateString(i8* getelementptr inbounds ([3 x i8],
[3 x i8]* @createstr_ptr, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr)
  %f6 = load i64, i64* %f
  call void (i8*, ...) @println(i8* %createstr, i64 %f6)
  %createstr7 = call i8* @CreateString(i8* getelementptr inbounds ([3 x
i8], [3 x i8]* @createstr_ptr.1, i32 0, i32 0))
  call void @__GC_Trace(i8* %gc_trace_map_obj, i8* %createstr7)
  %g8 = load double, double* %g
  call void (i8*, ...) @println(i8* %createstr7, double %g8)
  call void @__GC_ReleaseAll(i8* %gc_trace_map_obj)
  ret void
}
```

```
define { i64 } @try_return_one(i64 %a) {
entry:
  %gc_trace_map_obj = call i8* @__GC_CreateTraceMap()
  %a1 = alloca i64
  store i64 %a, i64* %a1
  %a2 = load i64, i64* %a1
  %tmp = add i64 %a2, 10
  call void @__GC_ReleaseAll(i8* %gc_trace_map_obj)
  %mrv = insertvalue { i64 } undef, i64 %tmp, 0
  ret { i64 } %mrv
}

define { i64, double } @try_return_two(i64 %a, double %b) {
entry:
  %gc_trace_map_obj = call i8* @__GC_CreateTraceMap()
  %a1 = alloca i64
  store i64 %a, i64* %a1
  %b2 = alloca double
  store double %b, double* %b2
  %a3 = load i64, i64* %a1
  %tmp = add i64 %a3, 10
  %b4 = load double, double* %b2
  %tmp5 = fadd double %b4, 2.000000e+01
  call void @__GC_ReleaseAll(i8* %gc_trace_map_obj)
  %mrv = insertvalue { i64, double } undef, i64 %tmp, 0
  %mrv6 = insertvalue { i64, double } %mrv, double %tmp5, 1
  ret { i64, double } %mrv6
}

declare { i64 } @digo_linker_await_func_try_return_one(i8*)

declare i8* @digo_linker_async_call_func_try_return_one(i64)

declare { i64, double } @digo_linker_await_func_try_return_two(i8*)

declare i8* @digo_linker_async_call_func_try_return_two(i64, double)
```

Part VII. Lessons Learned

7.1 Yufan

What this project taught me are not only techniques within the world of PLT, but also more things about how to build a real world project.

For PLT things, as the language guru in this project, I learned about the importance of rigorously semantic and syntax. I had to admit that two month ago when I first made the draft of the language reference manual, I didn't realize how many ambiguous syntaxes in the manual, which results in lots of shift/reduce errors when we were actually implementing the parser. Besides, semantics are also easy to cause traps. An ambiguous semantic definition may result in repeating change of implementation when doing the actual code generation. It would have been better if we made them as clear as possible at the first beginning. But after all, even though many obstacles we met during the development processes, many new things have been learned as well. I also learned the semantics of many data structures in Golang, since we were trying to follow Golang's syntax and semantics.

If I gained much with regard to PLT, then I would say I learned tremendous knowledge and skills outside the PLT world from my excellent and talented teammates. Our project manager introduced tools like Lark, Github, CircleCI to keep our project running smoothly, which surprised me because I have been in many groups in which no one tracked the project process, bugs were everywhere in the master branch, documents were always missing, etc. With the help of our project manager I started to feel like I was working on a real world project in a team of a big N tech company. Our system architect also taught me a lot regarding the implementation of our language. Without the system architecture he brought to us, I believed that we would never be able to finish such complex features inside Digo. But what made me feel amazed was not only his technical skills, but also that he was always willing to answer questions for us, when we had a lot of confusion regarding this architecture and questions regarding Ocaml and LLVM. Even when he was busy doing his own stuff, he was still going to take some time to share his thoughts to us. I believe cultivating this kind of attitude is important to an engineer. It is unrelated to technique, but it should be an important thing to learn if we want to build an excellent real world project by cooperating with others.

In short, I think this may be the actual meaning of this project: learning PLT, and also learning things beyonds PLT. As for advice for future teams, starting early and considering this project as a real world product, then you will learn things more than PLT.

7.2 Hanxiao

I learnt a lot from this project ．The project itself covers the whole picture of what a compiler is doing. By going through them in detail, I know what role each part of the compiler is playing. Besides, although ocaml is not a friendly language to use, I am getting used to it through this project. In addition, I could use github more efficiently after this project.

I also learnt a lot from my teammate. Communication skill is an important part of this project. My teammate showed me how good communication could facilitate the project going forward. Besides, trying to be patient is always the best way to step out of a dilemma. Instead of giving up instantly when stucking in some problems, diving into the question and thinking problem patiently is helping me solve the problem.

7.3 Wenqian

The whole learning and programming process is stunning and enlightening! It was my first time diving into designing a language and building a compiler from scratch to what Digo is right now. This journey drastically deepened my understanding of programming languages in general: how they scan my code, parse into ast, check semantically and finally translate to the machine.

Ocaml is also amazing, especially when I found that a few lines of code can be so powerful and can do such magic. At the very beginning, I was unused with pattern matching and felt lost with multiple levels of indentation in ocaml, but after I wrote more codes in it, pattern matching is such a great design! I am also amazed by the nature of ocaml that as long as I can compile my code, it’s usually not far away from the success.

Besides these technical takeaways, I also learned a lot being a team manager. Throughout the project, I learned how vital a manager has to be to keep everyone in the team on the right track and on the same page. It’s a bit difficult at first under the current special circumstances that everything has to be conducted online over zoom/wechat, but I am appreciative that all my teammates are super supportive, collaborative and considerate. All of us really put lots of effort into building digo and making it work.

My advice for future teams is to plan ahead, divide work clearly and make constant progress. Also, if you are feeling upset with Ocaml, my suggestion is to grab your fav drink (either starbucks or bubble tea, probably not wine or whiskey to ensure your brain can still function) and sit down for a whole afternoon to code your compiler in Ocaml. Of course you may get stuck with errors and warnings many times, but please don’t give up and do insist on solving them this afternoon. I suppose after three hours you will get used to Ocaml, and hopefully you would love it the other day. Good luck kids!

7.4 Sida

Designing the architecture of a long-term group project is not easy. There are so many things to take into consideration, from details like how to implement a language feature, to abstractions like how to make an interface fixed (not subject to implementation changes).

LLVM IR functions are pretty similar to C functions. It might seem to be difficult to use object-oriented concepts in LLVM IR functions at first. However, we can use a void*(or i8* in IR) to indicate an unknown class or wrapper; this is good, because objects should be transparent, i.e., the caller should not care about the details in this void*.

This pattern is widely used in our interface design. For example, the APIs exported from our serialization library is something like:

i8* SW_CreateWrapper() to create a new wrapper;
void SW_AddSlice(i8* wrapper, i8* slice) to serialize a slice and append into wrapper
void SW_AddInt64(i8* wrapper, int64 num) to serialize int64 and append into wrapper
void SW_GetBytes(i8* wrapper, i8** buf, i8* len) to get the serialization result

Another example is the GC interface:

void IncRef(i8* obj)
void DecRef(i8* obj)

The obj can be any objects inherited from the GC base class, including Future Object, Slice Object, and String Object.

You might have noticed that this is similar to C++. And yes, our libraries are all implemented in C++. When the APIs get exported, we use “extern c” to make the exported name the same as the function name.

Our programming language involves many features. It might be simple to cut them into separate parts, but it seems tricky to bring them back together. Good interface design and proper abstraction always helps.

Given that we do not treat basic types like int, double as objects, I did not realize to merge the serialization library into the object library. AddSlice and AddString are actually the same. If they override virtual functions such as ToBytes and FromBytes from base class, we can have a general API like SW_AddObject(i8* wrapper, i8* obj).

Another thing I want to mention is that at first we failed to notice it is ambiguous to pass a `future` as a function parameter. By only having a future object, we do not know the corresponding function; in other words, this function can be any function if a future object appears as one of the parameters; then, the return value of awaiting this future object cannot be determined at compile time. We do not have struct and use aggregated return value in IR, so it is impossible for us to generate strong-typed IR in this circumstance. A solution to this problem might be using something like future<Func> to explicitly state the corresponding function. Then we can also have slice<future<Func>>. And this is again similar to C++. Another alternative is

providing runtime information. We can just report an exception if the type of return value is not consistent. This is complex and you might have to figure out a way to store your return value somewhere instead of using strong-typed aggregated value in LLVM IR. Hope this will help future groups working on such a distributed language.

Part VIII. Acknowledgements

Special thanks to all the team members in Digo! We made it! Thanks to Professor Edwards for providing MicroC and for helping us achieve functional enlightenment. We referenced MicroC, the M/S project, the Shoo's final report, the Fly's final project, and most commonly, LLVM.moe.

Part IX. Appendix

9.1 Project Log

```
commit 15f27d9b91f83915306c10339def00da64b137bb
Merge: 9a97634 be2c35d
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Mon Apr 26 21:25:58 2021 +0800

    Merge pull request #89 from wy2249/fix-semant-w

    fixed all warnings in semant

commit be2c35d6e1401ff60a8e962b23596a348a7719c6
Author: wy2249 <wy2249@columbia.edu>
Date:   Mon Apr 26 13:23:17 2021 +0000

    fixed all warnings in semant

commit 9a9763414bf57583efbc24ee9d77b985da91ea68
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Mon Apr 26 18:52:24 2021 +0800

    add builder after gc_inject

commit 8ea1df2bbc790cfa9d27b779faa160eea723ed50
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Mon Apr 26 18:46:08 2021 +0800
```

```
    fix most of warnings in codegen

commit 25abf216bc6be0f64d2b601fc8e114e79d2d12f2
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 26 18:27:53 2021 +0800

    Update run-worker.sh

commit f6b376373751fe511e6ec0b3697ee78f6117b542
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 26 18:21:36 2021 +0800

    add demo

commit 5e58e71adf5f0be1a480669354b6fdb7213e389b
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 26 12:58:39 2021 +0800

    fix bugs in gc

commit 3c860973939efbec59862494f2916a6200c69797
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Mon Apr 26 02:59:56 2021 +0800

    finally fix slice decl

commit 4045f7dac70071d5db5d0eec7306c402073be09d
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 26 02:28:38 2021 +0800

    update tests

commit ea0a1f8ff4930ec7de05ef99d224c23531e58e3e
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 26 01:44:37 2021 +0800

    fix worker gc leak & add gc leak check in test script

commit 9fbfddd3a42e81d4dbe346a5bd4db7ef6b468038
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Mon Apr 26 01:16:15 2021 +0800
```

```
    solve slice declare

commit 23212926046945ee4ccb9ea08c7c4dfec99fa065
Merge: cfa4343 cd81e1f
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 25 09:11:28 2021 -0700

    Merge branch 'main' of https://github.com/wy2249/Digo into main

commit cfa43434edd53c78ad6c2041b3099fbdf4569764
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 25 09:11:05 2021 -0700

    tweak log at connection fail

commit cd81e1f2d8cd46f67436b6c3c7d929e6f8049666
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 26 00:04:54 2021 +0800

    update wordcount & add tests

commit 0b100daf36f4e4bc52ef00a55589dd0bb61a1d13
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 25 22:03:07 2021 +0800

    updated failure info per updated semant

commit 3d00dda5eadf8746c6f7259b7346c5eb7f4415eb
Merge: dcbbc0b 4f51595
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 25 21:59:41 2021 +0800

    Merge pull request #86 from wy2249/future_slice

    remove fussy print

commit 4f515958dbc0ff42c7f9e4f31bc57e25267793db
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 13:57:55 2021 +0000

    remove fussy print

commit dcbbc0b655b6d1c36310cf231846388cffb7efa1
```

```
Merge: 049757a 1100faa
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 25 21:52:02 2021 +0800

    Merge pull request #85 from wy2249/future_slice

    hot fix for word-count

commit 1100faa4194d2632c42ebec8764013d9565a543f
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 13:49:09 2021 +0000

    hot fix for word-count

commit 049757a4c53e2a198915a732be8ee8f9a38d93d6
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 20:36:17 2021 +0800

    update wordcount

commit afd751e99bb8c10c449d1ca73848479d88cda323
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 20:03:52 2021 +0800

    add wordcount

commit 498709f26a6e0accc4c265af494c846dd0049f59
Merge: fccb5c1 be262d6
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 25 17:08:38 2021 +0800

    Merge pull request #82 from wy2249/future_slice

    Fix#68 future slice

commit be262d60e40222760c69ff5bf9cb224320c5643a
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 08:59:26 2021 +0000

    fix for broken case in circelci

commit ea5c393defd0df90556bb4a2551bed061ec96f6c
Author: wy2249 <wy2249@columbia.edu>
```

```
Date:   Sun Apr 25 08:17:01 2021 +0000

    fix check same type function

commit ba271d56df0e2c63cd8ef3f85eb9b0e4f7fde534
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 07:46:57 2021 +0000

    remove al fussy print

commit fccb5c1cfbbed6877a26efa8c1fb6dcf9341dd3c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 15:37:06 2021 +0800

    fix typo in test script

commit 3f764b0c8351ede176626fbe1db91932aabbcdbe
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 15:31:06 2021 +0800

    update test script

commit c53a5712d16c9a7a017af32b01009cd09ec1d949
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 15:16:33 2021 +0800

    update test script

commit 7ecda12ba2bdd256baf3f2391e3b69fee4e94d73
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 07:16:31 2021 +0000

    codegen able to genrate llrm

commit e740c2a9cab5882c0baaf117e1201fb136238cf3
Merge: c5afce8 af427cf
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 25 00:15:18 2021 -0700

    Merge pull request #81 from wy2249/async-remote-lib

    fix worker retry
```

```
commit af427cfe7db0b87a068153f868a655b2d2cd493b
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 25 00:13:18 2021 -0700

    fix worker retry

commit 3275e9c8356d65584d8f6944482cf50d934bbf64
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 07:10:52 2021 +0000

    add logic to extract future function from future slice

commit ee1ed363dd153f7dc9ece40a2b7c9e846fc177ff
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 07:10:16 2021 +0000

    fix slice index

commit 1e443f5a1deca1178795db9192b60e28f136a87a
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 25 07:09:40 2021 +0000

    addd pretty print for expr

commit c5afce8b12c84f04377a60f91fdfa29810b4cdc5
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 23:23:38 2021 -0700

    Delete test-string2.digo.pass.expected

commit cbbc152ac21be297f69e8c9ea4ae180671cb5c0b
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 23:23:27 2021 -0700

    Delete test-string2.digo

commit 7b2bc07b1b633fab7b59859bc7c7142170bfdf48
Merge: 378f062 5a0bd24
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 23:19:10 2021 -0700

    Merge pull request #80 from wy2249/testcases
```

```
    (WIP) add several test cases

commit 378f06270fe9960d57a141ebe7e4e882736bc88c
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Sun Apr 25 14:10:40 2021 +0800

    index fix

commit a8a2ca13179fe54e7aec5cf120b8c6fc43d1943c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 13:26:28 2021 +0800

    add gc trace for readfile; fix ioutil gc leak

commit 478a79991397d4f7b02c310caa1519908c95b7d3
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 13:15:36 2021 +0800

    fix typo

commit 5a0bd24ada85ce14de0cdc452684b6dccbd9afd5
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 22:05:37 2021 -0700

    add several cases

commit 3b84292e13c4d0f5644ff518858c6ad832de9343
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 25 13:03:59 2021 +0800

    update test script to log debug info

commit e9e444d67832aa448cfc109b291a1f1fe011883c
Merge: 3e6eb31 84ddb54
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 15:53:24 2021 -0700

    Merge pull request #79 from wy2249/async-remote-lib

    fix segment fault

commit 84ddb54423e813097ab05d38a7908762a49b1c96
Author: xiaomi388 <xiaomi388@gmail.com>
```

```
Date:   Sat Apr 24 15:50:02 2021 -0700

    fix segment fault

commit 3e6eb31ceb77a08fbaa096c3f468d9a8aa4edc48
Merge: 4ae5000 6b8cb4c
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 11:29:13 2021 -0700

    Merge pull request #78 from wy2249/async-remote-lib

    Async remote lib

commit 6b8cb4cfcf430ee88382b052cbb0209795a44f78
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 11:26:07 2021 -0700

    fix read

commit ece91aded6da6e2a9fa666c1fd8e73205b3687d2
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 11:15:31 2021 -0700

    use at in vector

commit 07ab699576168d3ab07ef078654e1df545a223db
Merge: fca2726 4ae5000
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 10:42:00 2021 -0700

    Merge branch 'main' of https://github.com/wy2249/Digo into async-
remote-lib

commit fca27264af9fea9c26b10ba7622f03737fab848b
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 10:40:24 2021 -0700

    fix #77

commit 4ae50007ceb608efd4b88a5a25105730ebd05f65
Merge: 503fef0 14c1c88
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 10:11:52 2021 -0700
```

```
    Merge pull request #76 from wy2249/testcases

    add test for read

commit 14c1c88377ee0eb7936a4f843e666677d983f6a8
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 24 10:09:55 2021 -0700

    add test for read

commit ce56a459192b677962d4fb472bee421e79c73158
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 24 16:17:08 2021 +0000

    add pretty print for expr,sexpr for debug

commit 503fef0dcadb819911a85c607cf65a44f6fbbf15
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 23:43:29 2021 +0800

    Update config.yml

commit 45af95f19f5df8853b3ce15ee8aff066399f67bd
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 23:39:13 2021 +0800

    support test group & fix semant

commit d0d9300c872f95eec6e39c00e51728a05265353a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 20:04:07 2021 +0800

    add no-master option

commit 84f339eed0b30649d471555450985479ba3dbfc8
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 20:01:57 2021 +0800

    fix slice gc bugs

commit d2455324219d774ea72322445cae7060f99ec2b4
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
```

```
Date:   Sat Apr 24 17:42:46 2021 +0800

    update tests & fix bugs

commit 535fb4014ae9055c647114ed193ff8338f4cac43
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 14:50:17 2021 +0800

    fix typo in codegen

commit 92da8a46f510db23768b32c67f174d2cd43c8f83
Merge: 2266b71 b0de17a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 14:44:43 2021 +0800

    Merge pull request #75 from wy2249/fix-future

    avoid extract func call async llvm

commit b0de17a9068b1304b5381a1f1b58b6cdc5fab22a
Merge: 4d2fe5a 2266b71
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sat Apr 24 14:43:00 2021 +0800

    Merge branch 'main' into fix-future

commit 2266b71b1bb74f1bf7e8d0b078d3050037fa6544
Merge: 02f181c 5bffc3e
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 14:32:44 2021 +0800

    Merge pull request #74 from wy2249/async-remote-lib

    add gc support for slice string and future

commit 5bffc3e86f7126a2218b928155488496ec5f22b2
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 23:17:48 2021 -0700

    tweak

commit 02f181c7118263445122399b0b36c05a5456086c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
```

```
Date:   Sat Apr 24 14:16:30 2021 +0800

    fix gc in codegen

commit 0348fb7924b812635a2b5a36df0e34a968b9287a
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 23:15:34 2021 -0700

    add gc support for slice string and future

commit cb7c8d4e271b36f0eba0a9faabfda6a9a56494c2
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 13:55:35 2021 +0800

    gc finished

commit adb6ffcc460d7ccc904e5bd27756d53d2c0aa7b9
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 22:43:07 2021 -0700

    Updated config.yml

commit 4d2fe5a90b7da9bfd9a855f7bb1ce8235f26cf2a
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 24 05:32:26 2021 +0000

    avoid extract func call async llvm

commit 2c5d493c73bcca2753aa974437972689f45283f8
Merge: 46ae036 3e5c0ab
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sat Apr 24 13:14:57 2021 +0800

    Merge pull request #72 from wy2249/circleci-project-setup

    Circleci project setup

commit 46ae03660c3e485a0ef8b150b3ecf8783ee3e1b5
Merge: b23fa02 134d233
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 13:12:44 2021 +0800

    Merge pull request #63 from wy2249/semant
```

```
    Semant

commit b23fa025ddebf6692f709ab6e727ec6e9f304b81
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 12:58:07 2021 +0800

    fix Makefile

commit 3e5c0abaaf20bb5c7ec5b31f048259bc0f0181d0
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 21:55:57 2021 -0700

    Updated config.yml

commit a1c10fe0dec17140ceb4d2418d4e0913e3025887
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sat Apr 24 12:28:03 2021 +0800

    Add .circleci/config.yml

commit 08ca579ef2d180a6eee6105adfa0352064bc016a
Merge: 9c6155a b542f12
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 11:42:23 2021 +0800

    Merge pull request #71 from wy2249/gc-in-compiler

    Gc in compiler

commit b542f12e56757207c5ca8104a9a01f9299ce4d73
Merge: 8781972 9c6155a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 11:41:53 2021 +0800

    Merge branch 'main' into gc-in-compiler

commit 878197259478e70735a914c4eef1035738e204c0
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 11:22:21 2021 +0800

    check nullptr in wrapper
```

```
commit 9c6155ab71636b1e265c06c04c86c9401ba23aab
Merge: 24a2c52 582a9d8
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 17:15:53 2021 -0700

    Merge pull request #70 from wy2249/testcases

    add cases for digo features

commit 24a2c520dd0bf3f4d3424586504b6cd40ff47ce1
Merge: 8f05d10 ce6a42d
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 17:15:40 2021 -0700

    Merge pull request #69 from wy2249/async-remote-lib

    fix slice append

commit ce6a42d9ecb8bf8e8e689ea887707131223f7c45
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 17:15:15 2021 -0700

    fix slice append

commit 582a9d8dd50251d248f1cf2d37ae907d96036e60
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 17:05:37 2021 -0700

    add cases for digo features

commit 8f05d10a232596a828adc07065f1cc3b99751c3b
Merge: 6ccd9a9 d2c8057
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 15:40:35 2021 -0700

    Merge pull request #67 from wy2249/testcases

    tweak folders name

commit d2c80574bc4f65bb3bcd407d0ade50ab99d0730e
Merge: 864c066 6ccd9a9
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 15:40:21 2021 -0700
```

```
    Merge branch 'main' of https://github.com/wy2249/Digo into testcases

commit 864c066da1b07304b04549b02e3742029ec69983
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 15:38:09 2021 -0700

    tweak folders name

commit 6ccd9a9f9e02734886a7b17d2098bf172ab43531
Merge: 10bfd21 c0334f6
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 15:36:56 2021 -0700

    Merge pull request #66 from wy2249/testcases

    add semantic test expected result

commit c0334f6a94b6d542aa0d0300eea2b91023f657de
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 15:36:21 2021 -0700

    add semantics expected result

commit 10bfd21f7e59781fc3f9924ae9fbea77642d6298
Merge: a75450d 0b85d0e
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 24 03:27:08 2021 +0800

    Merge branch 'hot-fix' into main

commit 0b85d0ea878f5e177719e00dbeac9afb07f34548
Merge: 8da2eba a75450d
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 24 03:26:57 2021 +0800

    Merge branch 'main' into hot-fix

commit a75450d82492189cd59bf7fff642ea9b365c4a03
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sat Apr 24 03:05:22 2021 +0800

    replace docker image
```

```
commit 967722e98a7f7884e4c336560792a0e61e768d14
Merge: f013b57 3b8884d
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 23 11:32:27 2021 -0700

    Merge branch 'main' of https://github.com/wy2249/Digo into testcases

commit 3b8884df7b73ddd13fe30aaa2ea2a83b73a4e44e
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 01:32:47 2021 +0800

    add test config

commit 39e081b7edf880439f243b0571688e4420e8b3d0
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 24 01:31:06 2021 +0800

    clean tests

commit 12c5ebe9908785624c868ced5f710f57bb79d463
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Fri Apr 23 23:28:28 2021 +0800

    Delete .travis.yml

commit 1ed3b5d56571d5bfc2bd297bc35903ebfde7be18
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Fri Apr 23 22:38:16 2021 +0800

    Update .travis.yml

commit d722ef350e22c42e501591e60f10e3c0ed66b5a1
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Fri Apr 23 22:30:30 2021 +0800

    add initial travis file

    test for ci with travis

commit 134d23390588dd2cd4238ae8139eaf88d179ffe2
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 14:07:46 2021 +0000
```

```
    fix #60: add default val for implicit function return

commit 2c8dd56ade404bc3ffaf575a098c359ca9b3d3b2
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 12:19:49 2021 +0000

    add return type and length check

commit c200c80369831259917b9d008b872e14660e8d67
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Fri Apr 23 18:21:40 2021 +0800

    remove additional tests dir

commit 33b2f4b8d66ff7fd87c3cfc280d0a346cebe82b8
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Fri Apr 23 18:10:50 2021 +0800

    classify test cases

commit 8da2ebab245160e305f4994d12b73462aa0d5f38
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 10:09:04 2021 +0000

    add all pass log

commit ac0f42b984c68194834f5988fa2f5eabab4efe5e
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 10:08:17 2021 +0000

    add more ways to assign add(52,10) in test-func4

commit aff51c7a49872ba944f5e1f62babfbeec7a5f73f
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 10:07:03 2021 +0000

    revert test output change made by Hanxiao

commit 2a67c222c33b7d6f608ed2bbe4664288ba855582
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 10:06:22 2021 +0000
```

```
    fix void return type not able to retrieve

commit 83a8db972cd1f1b2907635c513b0640b1860fbcb
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 10:00:33 2021 +0000

    remove duplicate call and store for short decl one val

commit eb42bb0512bbef3b33118aa011062e214c2540d2
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 09:42:19 2021 +0000

    fix conflict of short decl retrieve multi val

commit 4b294c9b09a9faf0bfa19461eb897d3c29b923ce
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 09:23:21 2021 +0000

    fix test-fib: retrieve single type from function call

commit 48849296a86bad9920afa6d0e4d2d780f3c6dcbf
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 23 08:59:51 2021 +0000

    remove unnecassry check in declare

commit 865822b59eaf29ec714114709be3d6049459dc0b
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Apr 23 13:05:25 2021 +0800

    fix bugs in gc codegen

commit dfc7644888c3a3ebe67ab33a686324ec71de935c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Apr 23 00:38:41 2021 +0800

    add gc tracing for most objects

commit 79fd7e710a15dd31c6e0c2b65e74317b63934104
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Apr 22 21:27:34 2021 +0800

    gc almost finished in compiler
```

```
commit 2aeca34c8ea7d51172651bdea1e81caa7bf572b7
Merge: 1c7fb9d 618ad3e
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Apr 22 14:50:20 2021 +0800

    Merge branch 'main' into gc-in-compiler

commit 1c7fb9d9de2605154fa76c3c150c2e4589bbe436
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Apr 22 14:48:42 2021 +0800

    add debug interfaces for gc

commit f013b577d99039ad2c61c799c30e18741f674e40
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Apr 21 21:56:00 2021 -0700

    Delete fail-assign1.digo

commit dc41ebc31a602d279af60e6383a5e6a8249d070d
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Apr 21 21:54:01 2021 -0700

    add two tests for return

commit 9a10ffdb8d9bd236a5f8d549424ac9529354703c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Apr 22 12:07:03 2021 +0800

    simplified gc

commit 618ad3e2bfe1f682d3232289869090d289e74fa1
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Thu Apr 22 03:41:57 2021 +0800

    add additional test

commit 85bc92b6e4e631893039d020ef9b59d7184b9406
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Thu Apr 22 00:29:28 2021 +0800

    fix float and for
```

```
commit 833f0b7522be46039b76325f41fc83c9714350f7
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Apr 21 22:15:48 2021 +0800

    Update readme

commit c3aba1f03449f915cc7c895e40fbc542d25eecf2
Merge: a7ab6bd 3ff1c5f
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 21:57:06 2021 -0700

    Merge pull request #59 from wy2249/testcases

    tweak tests

commit 3ff1c5ff31db7a51031542df7b5b8ebbf9b00c7f
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 21:55:33 2021 -0700

    tweak tests

commit a7ab6bd012550261c47c6d152fe0efe04a82afcc
Merge: ed7f203 ec40403
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 21:46:45 2021 -0700

    Merge pull request #58 from wy2249/hot-fix

    fixed: no new line before else

commit ec4040362e8b7a8965dbba690305de8be04919a2
Author: wy2249 <wy2249@columbia.edu>
Date:   Tue Apr 20 04:46:03 2021 +0000

    fix test-if6

commit 20432cd8b1e51fe3259acca04aacc951f26f3910
Author: wy2249 <wy2249@columbia.edu>
Date:   Tue Apr 20 04:42:04 2021 +0000

    fixed: no new line before else
```

```
commit ed7f203c677193bdba0538bd140be192052af6a4
Merge: 4676266 c8a8231
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 21:20:02 2021 -0700

    Merge pull request #57 from wy2249/testcases

    tweak tests

commit c8a82311716dd7cb22cc1cb12bf42fd7d6874ae9
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 21:19:24 2021 -0700

    tweak tests

commit 46762660245dc56dbc40e77eb1e87c6e7ac963a4
Merge: 5952925 7aa7fe4
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 20:56:52 2021 -0700

    Merge pull request #56 from wy2249/testcases

    tweak some testcases

commit 7aa7fe434f81ccbe4bb2c66db5b40e214705f516
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 20:55:53 2021 -0700

    tweak test

commit 599837c932025e17a3da4bc3a4f6675e03721d8e
Merge: 7002322 5952925
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 19 20:46:55 2021 -0700

    Merge branch 'main' of https://github.com/wy2249/Digo into testcases

commit 5952925ac533f727ec7a66616ea7cdfc27e16bc1
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 19 21:54:20 2021 +0800

    Update README.md
```

```
commit f954d0a88aff93f3e7a5f98563f4583bfd1a3c85
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 19 21:40:49 2021 +0800

    improve logs for testall

commit 7002322ff6c08c8498e5ca2f8314c6899ba001a5
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 19:36:52 2021 -0700

    fix two cases

commit aac1e61d130af0178d0d06578c455ab1cb1b129b
Merge: 344221a 2966a5d
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 19:18:12 2021 -0700

    Merge pull request #54 from wy2249/fix-void

    fixed: void type return

commit 344221af4118bdc6a9a3595d8c522baac568cd94
Merge: 4a05a0d cbb8afa
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 19:07:42 2021 -0700

    Merge pull request #55 from wy2249/testcases

    Testcases

commit 2966a5d09340652d28edc0a36bfbf227fa080b2c
Author: wy2249 <wy2249@columbia.edu>
Date:   Mon Apr 19 01:51:54 2021 +0000

    fixed: void type return

commit 4a05a0d80c143ba43910c6b453ce7f72f972c754
Merge: e574df7 9cdb59b
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Mon Apr 19 09:29:16 2021 +0800

    Merge pull request #52 from wy2249/dev-reafile
```

```
    add read function to support readfile in demo

commit 9cdb59b54aa89d4b08cce0c79d08fc9a8a7d56ca
Merge: 04ce4c5 e574df7
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Mon Apr 19 09:28:58 2021 +0800

    Merge branch 'main' into dev-reafile

commit cbb8afa9e5f60b5b41f48a471438b97d3dff5254
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 11:57:06 2021 -0700

    add normal tests

commit de93c8ccbd529cf29ee7c09573e87df7e9deacf9
Merge: d007854 e574df7
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 10:03:39 2021 -0700

    Merge branch 'main' of https://github.com/wy2249/Digo into testcases

commit e574df7ef198dd2a90a1b14e2d8e236cc8a13359
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 19 00:55:36 2021 +0800

    fix bugs in testall.sh

commit d00785431cf2b9c6f0100281dc137978861524e8
Merge: 59ec7c4 40f7087
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 09:49:26 2021 -0700

    Merge branch 'main' of https://github.com/wy2249/Digo into testcases

commit 59ec7c42056d48e382d769e0046a3e97aad57188
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 18 09:49:21 2021 -0700

    add exec permission

commit 40f7087a5e83a8b95c82e7255cc665d199346f93
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
```

```
Date:   Mon Apr 19 00:47:25 2021 +0800

    fix bugs

commit 720e5a34fdbcf2049a16738cceba2dd76238db81
Merge: 4d30f77 5194b9a
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Mon Apr 19 00:05:49 2021 +0800

    Merge branch 'main' of https://github.com/wy2249/Digo into main

commit 4d30f7728e0be1c1517a6fa899cb20c44e47cf95
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Mon Apr 19 00:03:59 2021 +0800

    fix print and slice

commit 5194b9a69158773682874d7a6c54718195417d34
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 23:05:53 2021 +0800

    add ioutil to Makefile

commit 04ce4c52e9a631dc340a9f0d2c7ee5e4cf0beacd
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 14:40:30 2021 +0000

    change return type to slice(string); addd test+output

commit 328dc0bda3407c28719c6a35658b39f9414040ff
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 14:27:55 2021 +0000

    add read in semant

commit c35b5e800d786dcdf930aa88a320661b024150f2
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 14:26:37 2021 +0000

    add read in scanner, parser

commit ee762f1dfcc12bb4fd6a891d4c9a5370b12f417b
Merge: 06ff53e 7b78fb3
```

```
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 22:16:25 2021 +0800

    Merge branch 'main' of https://github.com/wy2249/Digo into main

commit 06ff53e27b19beb77899b13a4767b275b7070873
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 22:16:13 2021 +0800

    update tests

commit 7b78fb3ec2a9205270ff4410a832efad8017693a
Merge: 94621ec f75e3b2
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 18 22:13:07 2021 +0800

    Merge pull request #50 from wy2249/fix-len

    fixedd: len for slice and string

commit f75e3b27cd197c6fa62c163b7645af9c6dced7a0
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 14:06:40 2021 +0000

    remove fussy print; addd test output

commit 0bd52b76db048df7de4c94e47725f1be624dfac1
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 14:03:22 2021 +0000

    fixedd: len for slice and string

commit 94621ecb96b7beffb917d04e2f60ef4b010dcbf6
Merge: db36fe2 bd07f20
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 21:23:33 2021 +0800

    Merge pull request #49 from wy2249/new_slice_print

    add slice and print; need to work on len later

commit bd07f203782f14a91986b4bf12e0e6e65088a7b9
Merge: c929d61 db36fe2
```

```
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 18 21:22:13 2021 +0800

    Merge branch 'main' into new_slice_print

commit c929d61a3b128fcc9570aaf8189ae12858c9fb34
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 13:19:42 2021 +0000

    add slice and print; need to work on len later

commit db36fe209a27ee0044084323c597c73ee58fdf01
Merge: b69a198 9d00fb9
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 19:29:10 2021 +0800

    Merge branch 'main' of https://github.com/wy2249/Digo into main

commit b69a19834fae976cfafa75f725ba8a6cb694d0ff
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 19:29:06 2021 +0800

    update test sh

commit 9d00fb919767258b704999a7a32f2bd68459efbf
Merge: 53a3139 f4dbef6
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 18 19:27:36 2021 +0800

    Merge pull request #47 from wy2249/compiler-fix-async

    add target datalayout, target triple

commit f4dbef6a8c7730bba6fee50931b1680430bfd584
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 11:26:35 2021 +0000

    add target datalayout, target triple

commit 53a3139ea0fefc4e3e5284865d95ce43901cb138
Merge: 49ad4a1 efcfe54
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 18:40:05 2021 +0800
```

```
    Merge pull request #45 from wy2249/compiler-fix-async

    fix async related bugs found

commit 49ad4a16f051a8d658f9a638e73524d90646b084
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 17:18:50 2021 +0800

    update Makefile

commit efcfe549b33e83a8e0902ad8b61a69a10ca2f156
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 09:06:52 2021 +0000

    fixed: sematic check digo_main

commit 1a8fb37a399a9e2907a8726b627d007b276bccb0
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 08:44:58 2021 +0000

    remove fussy print_string

commit c395f0ae12993957eb239ba4a9baddd16022010f
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 08:43:08 2021 +0000

    async remote works; addd output in test

commit e6555801d1e9c4153195ad375d6c997925af5666
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 16:32:50 2021 +0800

    Update README.md

commit 26e3102304ca921a8e0a79fb4d176a3d7414b145
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 08:28:48 2021 +0000

    fixed: cannot await a short declared future object

commit 2a9e4f4320cf71736f5a0861ebdfabe68e008750
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
```

```
Date:   Sun Apr 18 16:23:21 2021 +0800

    fix interface inconsistency in compiler

commit d672948c3655f466d94fbbdaa3b9b6cac7d110fc
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 18 16:20:55 2021 +0800

    fix bugs in linker; make gc exported

commit d9dc6cf18f1f1d3f1824d9f209679b15d8a440af
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 18 06:57:51 2021 +0000

    fix: Incorrect number of arguments passed to called function!

commit cb6800937aa67cf6658746c490160484630d6886
Merge: 4acc5b7 81cb073
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 17 23:06:10 2021 -0700

    Merge pull request #44 from wy2249/async-remote-lib

    add support for setsliceindex

commit 81cb073f0516c64cf1bcb9c1e792bb2242c9c361
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 17 22:31:58 2021 -0700

    add support for setsliceindex

commit 4acc5b7d8322ca8701f2038359429b7501274dfe
Merge: df69e9e 1756acb
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sat Apr 17 21:44:25 2021 +0800

    Merge pull request #39 from wy2249/async-remote-lib

    finish ReadFile in lib

commit df69e9ea993ba97303224b0afbf2ceb83ec3a940
Merge: 2dd26bf 32adb3f
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
```

```
Date:   Sat Apr 17 19:24:04 2021 +0800

    Merge pull request #41 from wy2249/compiler-void

    fix void function decl type

commit 2dd26bf6d1e47f60bb9b371f2dbd32c9821f4db5
Merge: b299b5b 5690e80
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 17 19:23:09 2021 +0800

    Merge pull request #40 from wy2249/compiler-future

    future-related:  async and await

commit 32adb3f21d0048c35c3d16875f9f0e44dfc9df57
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 08:53:53 2021 +0000

    remove all print_string; digo.ml compile

commit 0bd209a66b76e3297954e672acdcf99d0b5c9b89
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 08:49:46 2021 +0000

    digo.ml for compile otion

commit 8eb3af41dc87afed3f8c7d70419736e46363e973
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 08:25:15 2021 +0000

    move compile from semant to digo.ml; more needd to do

commit dd76afeee8fdd94f25160d551b914a4138600129
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 08:12:00 2021 +0000

    fix for all void type func

commit 8ad8709c05846962d60e0f50148d4289b9ffb569
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 08:04:34 2021 +0000
```

```
    add ouput for test

commit efe510979304b0674c3485d6196de7015cfe019a
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 08:03:01 2021 +0000

    fix digo_main void return

commit 5690e80deb291f4d16dffd7479c5a5b16313e2e5
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 07:34:29 2021 +0000

    fix semant/codegen error for multi-return from await; lgtm

commit 17c25c19ad3eda2f749540a251610f1771c9eacb
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 07:12:10 2021 +0000

    await short ddecl done!

commit f531b007b88137fe550ebc8c49c60566e1395778
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 06:52:59 2021 +0000

    able to declare await

commit a9cf34ef59999135b1fbc36a29a182a91bdc768b
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 06:32:14 2021 +0000

    need to change stringmap to hashtbl; commit for backup

commit 387fb3367b1cd6157b2a81fa147990136c9f0af2
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 05:30:14 2021 +0000

    declare both async_call and await_func in async func call

commit c46c418d2465cf5103d9bdc96e34ce6d3b3ee505
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 17 04:57:00 2021 +0000

    await in semant
```

```
commit 41abae0c5e2dfc92bed21973be6de8b58ed25327
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 16 04:10:10 2021 +0000

    add test-future-decl output

commit 75d72ff785b0f500b4916de965297a64246df198
Author: wy2249 <wy2249@columbia.edu>
Date:   Fri Apr 16 04:09:18 2021 +0000

    fix furture short decl

commit 1756acbe106ce08ab611e048749e8708c5f00112
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 12 22:00:38 2021 -0700

    add one more test

commit 18f234d0566c25398a2413cb7823c75b37f6aad1
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Apr 12 21:53:12 2021 -0700

    finish readfile

commit b299b5b74abbab64053d43e8ae556f5b2fb0b215
Merge: 1b51883 9c15872
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 12 22:40:54 2021 +0800

    Merge pull request #38 from wy2249/compiler-string

    compiler string

commit 1b518836866a1686b5a8f4c32e3d5d20aaf0049f
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 12 20:52:29 2021 +0800

    move declaration out of linker to avoid repetition

commit eee50123d647e187eee3aa7db40ad930200ad3b7
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 12 20:32:55 2021 +0800
```

```
    Update README.md

commit caccd5a76113a1b0527b22735b79c24534bdec02
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 12 20:32:08 2021 +0800

    fix bugs in linker & add metadata gen

commit fac131afef380782f3405381effe58122c1d942a
Author: wy2249 <wy2249@columbia.edu>
Date:   Mon Apr 12 11:23:42 2021 +0000

    future decl, need to do short decl

commit 9c15872d193757f9ee98e73f371c68b59d99169f
Author: wy2249 <wy2249@columbia.edu>
Date:   Mon Apr 12 07:49:23 2021 +0000

    fix short decl len err; working now!

commit fed8587ee212c77835dd35c18946cd92f237a813
Author: wy2249 <wy2249@columbia.edu>
Date:   Mon Apr 12 07:05:37 2021 +0000

    add string compare and string clone

commit 0e9f4b62d128310e56239e810d5d1a5c1a4550fa
Merge: 3c2fbca 48d8b39
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 11 18:18:15 2021 -0700

    Merge pull request #31 from wy2249/async-remote-lib

    add get slice index api

commit 48d8b397829a8331ad66a8896a4c7e5790f6513e
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 10 23:47:37 2021 -0700

    add get slice api

commit 209b8ba56c8215f6ce7b57a03eb32700e0bdcc41
```

```
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 11 05:26:44 2021 +0000

    backup string

commit 3c2fbca30f506f0aac89fde9f20b4a770792f6a2
Merge: d085cc9 6e98a41
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 11 12:59:13 2021 +0800

    Merge pull request #30 from wy2249/compiler

    fix#23 compiler decl & short decl & multi val return

commit 8acb60ccf01dec8c7bf626039c4f60c0b85a6a80
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 11 04:48:26 2021 +0000

    str len; almost ddone except short decl cannot take len return (need to
fix)

commit 6c25d9bb0a40a965d0f431e810b16551b0120d08
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 11 03:56:43 2021 +0000

    str add

commit 6e98a41aed7585a34c09251a5259ca4c9fe4c62b
Author: wy2249 <wy2249@columbia.edu>
Date:   Sun Apr 11 03:41:41 2021 +0000

    create string

commit 9a42637a1911588f37adb97f058d0307092e590a
Merge: 4d49fca d05d84d
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 11 02:21:33 2021 +0800

    Merge pull request #29 from wy2249/compiler-fix

    fix semant and codegen checking order err

commit d05d84df16a87075ccf394c4afe08395d9f99888
```

```
Merge: d81e60e 4d49fca
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 11 02:21:21 2021 +0800

    Merge branch 'compiler' into compiler-fix

commit d81e60eb46f220a51a8111423efb8fb262b5cd65
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 18:13:02 2021 +0000

    fix semant and codegen checking order err

commit 4d49fca18c8884bec13ae2fc53fec0649a8376a0
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Sun Apr 11 01:57:08 2021 +0800

    fix multi return

commit d085cc9f3bccb7fa9553446a22b73c14139e470b
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 11 01:03:16 2021 +0800

    lib & linker finished

commit 298fec60dfc651d81db2d870a2d4fd3065237453
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 14:54:29 2021 +0000

    multi val

commit ab46c06bfa0d71664497a59f80f4c157e1f3b8de
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 12:19:22 2021 +0000

    short decl done

commit 9eb4c862ebca0fd9a91623e55271d60b1b1c0448
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 11:52:33 2021 +0000

    fix minor error for decl

commit 13c3f1382278e5cd8e82bb3aa7bbc22c1fe35697
```

```
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 11:06:54 2021 +0000

    error: decl should be right but semant checks in reverse order, so
weird

commit 5e4fae1dc82c8e902fe22d46f846214dde67dec0
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 10:17:02 2021 +0000

    fixedd; decl can work

commit e3b57509f12e9d635056a393ee7ed8c8f460db8d
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 09:58:04 2021 +0000

    error need to fix: decl cannot add to stringmap (commit for backup)

commit 4136bfb2ccf2747cb9b67860ac40d0f7705a966c
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 06:52:22 2021 +0000

    remove all about slocal (for backup)

commit cec822110b09c45812dc9db31a628fe2bf2e5938
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 06:32:49 2021 +0000

    add decl and short decl in ast,sat,semant

commit 129b4b798c5e0c9dc32a716747772b46d15623d4
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 04:40:54 2021 +0000

    fix error from ocamlbuild

commit 1711dba08ea2bb4535c8b113fd3e9793607517f6
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 10 04:18:30 2021 +0000

    fix parser for decl; no shift-reduce err

commit e04c75cb7fc16e398b045dcd1c8ef9436977ce8a
```

```
Merge: 0d81606 de9d6ae
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 10 12:11:28 2021 +0800

    Merge pull request #27 from wy2249/linker

    add linker-level multi return value support

commit 0d8160670cdbe5de2a3f526574df4022e590e592
Merge: a40d343 4ea7a08
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 10 12:10:19 2021 +0800

    Merge pull request #28 from wy2249/async-remote-lib

    digoslice

commit 4ea7a08c0ecbea6becb347741056b7360931a65b
Merge: 85949e6 00cf7a8
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 9 21:02:51 2021 -0700

    Merge branch 'async-remote-lib' of https://github.com/wy2249/Digo into
async-remote-lib

commit 85949e683887325c51ffd91be597f4c261f59581
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 9 21:02:28 2021 -0700

    delete sliceslice

commit 00cf7a82036b0b4753ffc7218e39b254b418b233
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 9 20:50:37 2021 -0700

    Delete .DS_Store

commit 3329ef003aa36b0314d724f90bf655268285ae8a
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 9 12:18:21 2021 -0700

    add extern c
```

```
commit a33d50d2999158fc2e1777302c0fd82dcaa608df
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 9 12:04:10 2021 -0700

    tweak gc

commit a5030ad488a74881b8e7b2bc850769674f408f18
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Fri Apr 9 12:03:28 2021 -0700

    finish slice

commit de9d6ae3b05230786c2a5859319052799c7e433b
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 10 01:48:34 2021 +0800

    add linker-level multi return value support

commit a40d343fa9bf66fe4c39dfe2fcfc3fc6983b9cdb
Merge: b9c4162 5678cd4
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 10 00:53:12 2021 +0800

    Merge pull request #26 from wy2249/linker

    Uses DObject instead of RefWrapper

commit 5678cd484d439c87201a1026d8afbea93e298dc4
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Apr 9 23:03:37 2021 +0800

    Uses DObject instead of RefWrapper

commit b9c4162d0d080f8fbe8d9f9f6e31613261a47316
Merge: 52f264a 683d766
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Tue Apr 6 16:19:52 2021 +0800

    Merge branch 'main' of https://github.com/wy2249/Digo into main

commit 52f264a413a8d19ea5b7ffc0b88e859b6c8ad28a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Tue Apr 6 16:19:45 2021 +0800
```

```
    Update type definition

commit 683d7668165e612c070c1fe0362893073c98a14a
Merge: 3f1bc70 0189acd
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Apr 5 14:10:51 2021 +0800

    Merge pull request #25 from wy2249/async-remote-lib

    Finish DigoString

commit 0189acd2a5fea5528a152ecb00732e6438cfe7df
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 4 23:03:19 2021 -0700

    remove attribute((noinline)) in dstring

commit 08703e00e5ebc701fd0da1a7c38571d1131fda59
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 4 10:51:54 2021 -0700

    finish test

commit dc75f2dbea946997183768ff7a63e03db6633556
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Apr 4 10:14:53 2021 -0700

    add support for digostring

commit 3f1bc70780b7a93a25282d6e0e22b4be9f8f8348
Merge: 7389e86 53f1f16
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 4 11:27:01 2021 +0800

    Merge pull request #22 from wy2249/issue-19

    fix vector bug

commit 53f1f1666f61c3a31a0ef57faa8780ec59afa795
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 3 20:22:23 2021 -0700
```

```
    fix vector bug

commit 7389e86e0dd3338f94d9e93153ae709893a472c1
Merge: a5b87c3 488158a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 4 11:20:35 2021 +0800

    Merge pull request #21 from wy2249/compiler

    Compiler

commit 488158a4f9144e70883239e60ec0dd2c0ee9ae7c
Merge: a3df1ea a5b87c3
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Sun Apr 4 11:19:23 2021 +0800

    Merge branch 'main' into compiler

commit a5b87c35e309f6280289a86872117d302817371c
Merge: a6f1bbb c3fd9a6
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Apr 4 11:06:22 2021 +0800

    Merge pull request #20 from wy2249/issue-19

    fix #19

commit c3fd9a6ec8db3a5cab1d64eb8578a03d3a532caf
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 3 14:30:40 2021 -0700

    tweak func name

commit 83b4040d23c56cf802b0bc9373b12788f6c9a518
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 3 14:29:35 2021 -0700

    rm

commit a23030a127c6c235961499839c9f61b5f39371a9
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 3 14:28:06 2021 -0700
```

```
    rm

commit 9c43d6b9ee899da72089c8136b0f7f2b97075779
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sat Apr 3 14:27:00 2021 -0700

    fix #19

commit a3df1eafc3ce72dc92de9ea40aa3452f27c04590
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 3 19:48:47 2021 +0000

    fix the last shit/reduce one; to discuss: multiple vals assignment

commit 7748ba36fc647af9968b5a144c4ab431897e71b2
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 3 19:08:09 2021 +0000

    fix shift/reduce conflict; still one need to go

commit 1fedb0543c92e5f2ba7adc55dc6efdc92d675a20
Author: wy2249 <wy2249@columbia.edu>
Date:   Sat Apr 3 17:52:42 2021 +0000

    fix bugs (able to generate ll)

commit a6f1bbba0d9a38f7a64d088e31c66e663a3956c5
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 3 17:16:21 2021 +0800

    Update README.md

commit e66a04895f70c21ddf8ab36ec6265e2f3075594a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 3 17:09:51 2021 +0800

    linker v1.1 with remote support

commit d1fffd25622f41f3fff137fdcdbca7738dc1b3aa
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Apr 3 12:05:37 2021 +0800

    add test framework; compiler failed; cannot make progress
```

```
commit 793727d32be3dffd5fcaef9400295cb09cc20c41
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Apr 2 18:34:18 2021 +0800

    update auto gc

commit c6bd28b6064eb527b395f1b0047a72cf9d18c483
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Apr 2 18:14:22 2021 +0800

    add auto gc

commit 5ea5d4567e8bc8952e9d0db92179908cb2260d7e
Author: Hanxiao <m13056350538@163.com>
Date:   Fri Apr 2 14:07:50 2021 +0800

    needs debug

commit 91165942306c9b45bd7a25c7bfadb21c7278dc77
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Mar 27 23:30:59 2021 +0800

    Update README.md

commit 6ebf118697bcb25d90ebd65d1d328f1310fd42fb
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Mar 27 23:29:10 2021 +0800

    Update README.md

commit 33a059f457502236bf5824f585223e8a9682c1d3
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Mar 27 23:22:58 2021 +0800

    full compile chain completed

commit 30ad12c2732711305b5fb86731490258511c7ef6
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Mar 27 22:05:40 2021 +0800

    linker v1.0 finished
```

```
commit a87bef4d9f1f49d0d4d56946ce52e4a875380fb4
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Mar 26 23:34:44 2021 +0800

    JobDecRef should be in the same scope with job creator

commit a414dcb9b09a2ee12e0221b7095cc00f91b1a9a7
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Mar 26 23:13:59 2021 +0800

    linker v0.3 with template ll

commit 6cb34c49a8b193be929777e52aeba516770c1798
Merge: 8e77a97 6054f60
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Thu Mar 25 00:53:34 2021 +0800

    Merge pull request #13 from wy2249/async-remote-lib

    fix printstring

commit 6054f6034cb091749c5148cfb75f69931502cfe7
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Thu Mar 25 00:53:00 2021 +0800

    fix printstring

commit 8e77a97a4897c21ae4f97114585e28eae0e0a1c8
Merge: 61ffa51 191deac
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Thu Mar 25 00:46:55 2021 +0800

    Merge pull request #12 from wy2249/async-remote-lib

    fix printstring

commit 191deace4d7c49d06edd0fcf386bb4d259c9846c
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Thu Mar 25 00:46:06 2021 +0800

    fix printstring

commit 61ffa51e72351fb774b0bce902169ec1faec1bf3
```

```
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Wed Mar 24 23:45:26 2021 +0800

    Update README.md

commit 86ac930740c9da23d39708f6baee5365d1151eff
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Wed Mar 24 23:20:23 2021 +0800

    Update README.md

commit 87e46b06bc1aa2917ede3f94c90e2aaf7f2dee69
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Wed Mar 24 23:19:36 2021 +0800

    Update README.md

commit b0171e5de89370a0f2c8027b61950bad46c8d26d
Merge: e39bb7e 201c261
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Mar 24 23:04:30 2021 +0800

    Merge branch 'main' of https://github.com/wy2249/Digo into main

commit e39bb7e562749a442aed4e1da5e3aa902178bd46
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Mar 24 23:03:44 2021 +0800

    makefile

commit 201c261ff7068441bcbfe40b424c0d0981df3a57
Merge: 1aff31e 1fb58d9
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Mar 24 22:56:43 2021 +0800

    Merge pull request #11 from wy2249/async-remote-lib

    tweak makefile

commit 1fb58d92e3f4f73c2f99553d611e4eed44c9a545
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Mar 24 22:55:50 2021 +0800
```

```
    tweak float to double

commit f57673a084086bc14cce7b913d1680169bfb15aa
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Mar 24 22:48:38 2021 +0800

    tweak makefile

commit 1aff31ecf8906eb8b332dbc4a48cc616dce5e2ed
Merge: c89d4a7 99de614
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Mar 24 22:33:10 2021 +0800

    Merge pull request #10 from wy2249/async-remote-lib

    add print

commit 99de61416b925f74059ece5bdaf901faf1b05ce4
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Mar 24 22:22:45 2021 +0800

    fix share ptr in byte

commit 7aba8debf0dbd17ce2c022b6e41210dfeeef2b7c
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Mar 24 22:04:41 2021 +0800

    add print

commit c89d4a72653bf73fa8d5743b23b43dedf6828e88
Merge: 638a1d8 e2d7b5a
Author: Wenqian Yan <60408875+wy2249@users.noreply.github.com>
Date:   Wed Mar 24 22:01:22 2021 +0800

    Merge pull request #9 from wy2249/compiler

    Compiler (hello world)

commit e2d7b5ae6819bbb775f4245b4a92f7e3a8d30662
Author: lhx <m13056350538@163.com>
Date:   Wed Mar 24 13:34:34 2021 +0000

    hello world is good
```

```
commit e285f73da07df979423ece16484790c4f601fcd9
Author: lhx <m13056350538@163.com>
Date:   Wed Mar 24 11:05:09 2021 +0000

    good except hello world

commit c99b8411fdc7a02c0092150a9ae63a3a9fd32889
Author: lhx <m13056350538@163.com>
Date:   Wed Mar 24 09:36:36 2021 +0000

    could compile code

commit 72830cfd9b5c918081fc426f926b71fb21178cc6
Author: lhx <m13056350538@163.com>
Date:   Wed Mar 24 09:24:01 2021 +0000

    partially fix codgen.ml

commit 9bc3821066e4fd7214049a05a8928821f4b8a1e7
Author: lhx <m13056350538@163.com>
Date:   Wed Mar 24 03:16:16 2021 +0000

    add codegen.ml

commit b42e905d13bec59f79a5652ef7afb22f310ca1a1
Author: wy2249 <wy2249@columbia.edu>
Date:   Wed Mar 24 08:56:55 2021 +0800

    bugs need to fix; just for backup

commit cd3158ae8ea28f7503341b4b6c38fa1b43473c20
Author: wy2249 <wy2249@columbia.edu>
Date:   Wed Mar 24 00:56:05 2021 +0800

    semantic check duplicate function and param

commit 638a1d812357beb04e523e61586341ab7da2d2c1
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Mar 21 21:36:48 2021 +0800

    linker v0.2 with metadata parser
```

```
commit faeecf957e91de69963463e88f0e23bc48277da6
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Mar 21 18:48:03 2021 +0800

    Update serialization_wrapper.cpp

commit 6ba2f3211629e81fcc29d73a88addba37a2ea981
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Mar 21 18:43:53 2021 +0800

    Update serializer_template.ll

commit e7ad8a00ca0c0314b0d54766e60579753d85b721
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Mar 21 18:37:12 2021 +0800

    no exception across boundary

commit 24a1e80cb3953b5f2b5df7806dc6450227581c63
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Mar 21 18:27:19 2021 +0800

    linker v0.1

commit abcee0ebda42b57470ae82d3d36d1689232092c7
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Mar 21 17:03:10 2021 +0800

    serialization v0.1

commit 28bb36b5c704113df8d211f86b6acc244110c30d
Merge: 145dfc1 7754007
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Tue Mar 9 23:59:03 2021 +0800

    Merge pull request #6 from wy2249/async-remote-lib

    finished async-remote-lib w/o the linker part

commit 77540075c4036add9d3b349e248426f94fa7e7d1
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 23:41:14 2021 +0800
```

```
    add readme & create local test

commit 32cc2d4c657e5c3c3a822b5369e495b2e285f3b3
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 23:36:49 2021 +0800

    fix unittest not passed

commit f6bb9ee8a40f7e2dae7a1d4ad4e13f9b1307216d
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 22:31:02 2021 +0800

    add test for async

commit 578c708fd79d1f4cd4bc540054d283597ced67ee
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 20:21:23 2021 +0800

    add unitest for remote async

commit 33c33edb475f85db971a9d38f6c0c65d4d3a5b84
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 18:00:29 2021 +0800

    finish master & worker & unittest

commit 35a9645193ff7606fd40be3d8b75ac0a06ad0a72
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 12:33:58 2021 +0800

    finish client & unittest

commit 486571a40cb3d129925f0861c9d18e6d97fa07f0
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 8 11:36:57 2021 +0800

    finish server & unittest

commit f9d7dced56a327b8fe514fd5ff9508079ad18fac
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Sun Mar 7 20:56:41 2021 +0800

    add test framework & finish network::Server::Start
```

```
commit 145dfc194895b30005a938188725222b8932c15b
Merge: 71f64df f08c6a9
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Mon Mar 1 21:12:43 2021 +0800

    Merge pull request #5 from wy2249/bugfix-shift-reduce

    Bugfix shift reduce

commit f08c6a99b2d479d680290684cf76f6d2f6ec28ad
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 1 20:47:24 2021 +0800

    tweak

commit 4b5d9c26d5d7ac9a256055e09fecb349d1c4b476
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 1 20:45:44 2021 +0800

    fix

commit 71f64df70a876783ec25e82c84ac0fdcf11f0f16
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Mon Mar 1 20:33:46 2021 +0800

    update ignore

commit 7f228ee2a8972be366f3d9f5245a9a8ef480871a
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Feb 28 21:29:51 2021 +0800

    generate executable directly from llvm ir

commit a7c97e90ddc447845db8ea10847382ccc715b379
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sun Feb 28 18:20:17 2021 +0800

    template for async remote library

commit 09492959776453af22dea42f49b72bed1a2746fa
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Feb 27 22:07:25 2021 +0800
```

```
    change directory structure

commit 6a03ab704f6b62d236b443a819a3249fff2eced3
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Sat Feb 27 21:59:26 2021 +0800

    support declare of multiple values

commit 86761d296ad8535f7d44ad40ea0d162fe194cc49
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Feb 25 01:35:12 2021 +0800

    unary op, builtin functions, await

commit 4318cbb711a48ccc4fa10d5a801fff8600bfe7ad
Author: feiyun lu <feiyunlu@feiyundeMacBook-Pro.local>
Date:   Wed Feb 24 23:26:48 2021 +0800

    add binary operation and test2

commit eddf012ef8e4fe1daf93a3f931922bb62edde789
Merge: c8c5fba 15ec023
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Feb 24 13:13:21 2021 +0800

    Merge pull request #3 from wy2249/feature-comment

    Feature comment

commit 15ec0230051ed7fa6fa3ce26447895389926d662
Merge: e0e70d2 c8c5fba
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Feb 24 13:13:11 2021 +0800

    Merge branch 'main' into feature-comment

commit c8c5fbab40eb0a7e0e26d8255d0b1ba612f09de7
Merge: 82d3ef0 dd6806c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Feb 24 13:06:12 2021 +0800

    Merge pull request #2 from wy2249/feature-literal
```

```
    Feature literal

commit dd6806c549cd54140b3150902e12004ff930cbfa
Merge: 5ffd8da 82d3ef0
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Feb 24 13:04:42 2021 +0800

    Merge branch 'main' into feature-literal

commit 82d3ef0f6c4bac8996c010b46cb093cf41731ace
Merge: 7b18dfb 730d796
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Feb 24 13:00:13 2021 +0800

    Merge pull request #1 from wy2249/feature-slice

    Feature slice

commit 730d796950c7e64252aaefbc9fc561f292cd0747
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Feb 24 11:54:42 2021 +0800

    tweak filename

commit afb56b296b0b40fb16d993a467908f36a999850a
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Feb 24 11:26:32 2021 +0800

    modify test

commit 97363bac1dae024a00c50393310d08e47c98cc02
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Feb 24 11:24:41 2021 +0800

    finish slice operations

commit 00e9c6bb975468a35358c4d7ad8b570e118b9ccc
Merge: 18e5697 7b18dfb
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Feb 24 11:05:08 2021 +0800

    finish slice type & literal
```

```
commit 18e5697f060861daade9fe62fa41a9681c097279
Author: xiaomi388 <xiaomi388@gmail.com>
Date:   Wed Feb 24 10:05:33 2021 +0800

    save

commit e0e70d2495d24583a4c3d8aee1f773b9c5f91339
Author: wy2249 <wy2249@columbia.edu>
Date:   Wed Feb 24 02:03:47 2021 +0800

    add two types comment

commit 5ffd8da5883b4d6b161d20c1c2744008e51e2393
Author: wy2249 <wy2249@columbia.edu>
Date:   Wed Feb 24 01:18:21 2021 +0800

    add bool literal and test

commit db53b5669d158b1d8a45bf28e27f26c7819e8155
Author: wy2249 <wy2249@columbia.edu>
Date:   Wed Feb 24 01:02:58 2021 +0800

    add float literal and test

commit 7b18dfbe91b5615b565a762fd41335daed95c40f
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Feb 19 22:11:28 2021 +0800

    support (expr)

commit 1b1139cfd71d8ca1ede3b64af6e2ac12d1e9788c
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Fri Feb 19 21:59:37 2021 +0800

    support multiple return value

commit 4dab48ba2765323e2460a484fb80d662077ec1f1
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Feb 18 00:14:24 2021 +0800

    Update Makefile
```

```
commit c676dfe224519c0f45d1e1ebfec4577c0869c715
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Thu Feb 18 00:06:40 2021 +0800

    Update alpha version

commit e6a88a15dcba816a63251885971201f22822eff5
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Wed Feb 17 20:27:46 2021 +0800

    Alpha version

commit ae0ac6cfb9faa80fe7cb08c1c61c0d52f68d5cb7
Author: sdh21 <78904432+sdh21@users.noreply.github.com>
Date:   Tue Feb 16 17:21:46 2021 +0800

    Create README.md
```

9.2 digo-compiler

All of the four members got involved in scanner, parser and ast when designing language. Afterwards, digo-compiler was mostly finished and modified by Wenqian and Hanxiao. Sida did metadata_gen.ml.

9.2.1 digo.ml

```
(* Top-level of the Digo compiler: scan & parse the input,
   check the resulting AST and generate an SAST from it, generate LLVM IR,
   and dump the module *)

   type action = Ast | Sast | LLVM_IR | Compile

   let () =
     let action = ref Compile in
     let set_action a () = action := a in
     let speclist = [
       ("-a", Arg.Unit (set_action Ast), "Print the AST");
       ("-s", Arg.Unit (set_action Sast), "Print the SAST");
       ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM
```

```
IR");
       ("-c", Arg.Unit (set_action Compile),
         "Check and print the generated LLVM IR (default)");
     ] in
     let usage_msg = "usage: ./digo.native [-a|-s|-l|-c] [file.digo]" in
     let channel = ref stdin in
     Arg.parse speclist (fun filename -> channel := open_in filename)
usage_msg;

     let lexbuf = Lexing.from_channel !channel in
     let ast = Parser.functions Scanner.tokenize lexbuf in
     match !action with
       Ast -> ()
     | _ -> let sast = Semant.check ast in
       match !action with
         Ast     -> () (*print_string (Ast.string_of_program ast) *)
       | Sast    -> () (*print_string (Sast.string_of_sprogram sast) *)
       | LLVM_IR -> print_string (Llvm.string_of_llmodule
(Codegen.translate sast))
       | Compile -> let m = Codegen.translate sast in
     Llvm_analysis.assert_valid_module m;
     print_string (Llvm.string_of_llmodule m)
  (*

  let _ =
    let lexbuf = Lexing.from_channel stdin in
    let ast = Parser.functions Scanner.tokenize lexbuf in

    let sast = Semant.check ast in
    let m = Codegen.translate sast in
    Llvm_analysis.assert_valid_module m;
    print_string(Llvm.string_of_llmodule m)
*)
```

9.2.2 scanner.mll

```
{ open Parser }

rule tokenize = parse
  [' ' '\t' ] { tokenize lexbuf }
| ['\r' '\n'] { NEWLINE }
```

```
| "//"     { comment_line lexbuf }
| "/*"     { comment_block lexbuf}

| '+' { PLUS }
| '-' { MINUS }
| '*' { TIMES }
| '/' { DIVIDE }
| "||" { LOGICAL_OR }
| "&&" { LOGICAL_AND }
| "==" { IS_EQUAL }
| "!=" { IS_NOT_EQUAL }
| "<" { IS_LESS_THAN }
| ">" { IS_GREATER_THAN }
| ">=" {IS_GREATER_EQUAL}
| "<=" {IS_LESS_EQUAL}

| '!' { LOGICAL_NOT  }

| '{' { LEFT_BRACE }
| '}' { RIGHT_BRACE }
| '[' { LEFT_BRACKET }
| ']' { RIGHT_BRACKET }
| '(' { LEFT_PARENTHE }
| ')' { RIGHT_PARENTHE }

| ',' { COMMA }

| '=' { ASSIGNMENT }
| ":=" { ASSIGNNEW }   (* TODO *)
| ':' {COLON}
| ';' { SEMICOLON }
| '%' {  MODSIGN }
| eof { EOF }

| "for"     { KEYWORD_FOR     }
| "if"      { KEYWORD_IF      }
| "else"    { KEYWORD_ELSE    }
| "func"    { KEYWORD_FUNC    }
| "return"  { KEYWORD_RETURN  }
| "await"   { KEYWORD_AWAIT   }
| "async"   { KEYWORD_ASYNC   }
| "remote"  { KEYWORD_REMOTE  }
| "var"     { KEYWORD_VAR     }
```

```
| "string"  { KEYWORD_STRING  }
| "int"     { KEYWORD_INT     }
| "float"   { KEYWORD_FLOAT   }
| "bool"    { KEYWORD_BOOL    }
| "continue"{ KEYWORD_CONTINUE}
| "break"   { KEYWORD_BREAK   }

| "len"     { KEYWORD_LEN     }
| "append"  { KEYWORD_APPEND  }
| "read"    { KEYWORD_READ    }

| "future"  { KEYWORD_FUTURE  }
| "void"    { KEYWORD_VOID    }

| "true" | "false" as boollit { BOOLEAN_LITERAL(bool_of_string boollit)}
| ['0'-'9']+ as lit { INT_LITERAL(int_of_string lit) }
| ['a'-'z' 'A'-'Z'] ['a'-'z' 'A'-'Z' '0'-'9' '_']* as var { VARIABLE(var) }
| '"' ([^ '"' ]*) '"' as str { STRING_LITERAL(String.sub str 1
(String.length str-2)) }
| (['0'-'9']+)'.'(['0'-'9']+) as lxm { FLOAT_LITERAL(float_of_string lxm)}

and comment_line = parse
  '\n'  { tokenize lexbuf }
  | _   { comment_line lexbuf }

and comment_block = parse
  "*/"  { tokenize lexbuf }
  | _   { comment_block lexbuf }
```

9.2.3 ast.ml

```
type binary_operator =
  Add | Sub | Mul | Div | Mod
| LessThan | LessEqual | GreaterThan | GreaterEqual
| IsEqual | IsNotEqual
| LogicalAnd | LogicalOr

type unary_operator = LogicalNot | Negative

type builtin_type =
IntegerType
| FloatType
```

```
| StringType
| SliceType of builtin_type
| BoolType
| FutureType
| VoidType

type expr =
  EmptyExpr
| Integer of int
| Float of float
| String of string
| Bool of bool
| BinaryOp of expr * binary_operator * expr
| UnaryOp  of unary_operator * expr
| AssignOp of expr * expr
| FunctionCall of string * expr list
| NamedVariable of string
| SliceLiteral of builtin_type * int * expr list
| SliceIndex of expr * expr
| SliceSlice of expr * expr * expr
| Len of expr
| Append of expr list
| Await of string
| Read of expr


type statement =
  EmptyStatement
| IfStatement of expr * statement * statement
   (* condition expression; statements if true; statements if false  *)
| ForStatement of expr * expr * expr * statement
    (*  for ssmt1; expr2; ssmt3 {  statements   }  *)
| Break
| Continue
| Return of expr list
| Expr of expr
| Declare of string list * builtin_type * expr list
| ShortDecl of string list * expr list
| Block of statement list

type bind = builtin_type * string

type func_annotation =
```

```
  FuncNormal
| FuncAsync
| FuncAsyncRemote

type func_decl = {
ann : func_annotation;
fname : string;
typ : builtin_type list;
formals : bind list;
body : statement list;
}

type functions = func_decl list


let string_of_op = function
Add -> "+"
| Sub -> "-"
| Mul -> "*"
| Div -> "/"
| LessThan -> "<"
| LessEqual -> "<="
| GreaterThan -> ">"
| GreaterEqual -> ">="
| IsEqual -> "=="
| IsNotEqual -> "!="
| LogicalAnd -> "&&"
| LogicalOr -> "||"
| Mod -> "%"

let string_of_uop = function
LogicalNot -> "!"
| Negative -> "-"

let rec string_of_typ = function
IntegerType -> "int"
| FloatType -> "float"
| StringType -> "string"
| SliceType(typ) ->"Slice(" ^ string_of_typ typ ^ ")"
| BoolType -> "bool"
| FutureType -> "future"
| VoidType -> "void"
```

```
let rec string_of_ann = function
FuncNormal -> "normal"
| FuncAsync -> "async"
| FuncAsyncRemote -> "async remote"

let rec string_of_expr = function
EmptyExpr -> "empty"
| Integer(x) -> string_of_int x
| Float(x) -> string_of_float x
| String(x) -> x
| Bool(x) -> string_of_bool x
| BinaryOp(e1,op,e2) -> string_of_expr e1 ^ " " ^ string_of_op op ^ " " ^
string_of_expr e2
| UnaryOp(op,e) -> string_of_uop op ^ string_of_expr e
| AssignOp(e1,e2)-> string_of_expr e1 ^ " = " ^ string_of_expr e2
| FunctionCall(f,el)->f ^ "(" ^ String.concat ", " (List.map string_of_expr
el) ^ ")"
| NamedVariable(x) -> x
| SliceLiteral(t,int,el) -> "[]"^string_of_typ t ^ "{"^ String.concat ", "
(List.map string_of_expr el) ^ "}"
| SliceIndex(e1,e2)-> string_of_expr e1 ^ "[" ^ string_of_expr e2^ "]"
| SliceSlice (e1,e2,e3)->string_of_expr e1 ^ "[" ^ string_of_expr e2 ^":"^
string_of_expr e3^ "]"
| Len(e1)->"len("^string_of_expr e1 ^ ")"
| Append(el)->"append("^ String.concat ", " (List.map string_of_expr el) ^
")"
| Await(e)->"await "^e
| Read(e)->"read " ^string_of_expr e
```

9.2.4 parser.mly

```
%{ open Ast %}

%token NEWLINE PLUS MINUS TIMES DIVIDE MODSIGN
%token LOGICAL_OR LOGICAL_AND IS_EQUAL IS_NOT_EQUAL IS_LESS_THAN
IS_GREATER_THAN
%token IS_LESS_EQUAL IS_GREATER_EQUAL LOGICAL_NOT
%token LEFT_BRACE RIGHT_BRACE LEFT_BRACKET RIGHT_BRACKET
%token LEFT_PARENTHE RIGHT_PARENTHE
%token ASSIGNMENT ASSIGNNEW SEMICOLON COLON EOF COMMA
%token <int> INT_LITERAL
```

```
%token <string> STRING_LITERAL
%token <float> FLOAT_LITERAL
%token <bool> BOOLEAN_LITERAL
%token <string> VARIABLE

%token KEYWORD_FOR KEYWORD_IF KEYWORD_ELSE KEYWORD_FUNC
%token KEYWORD_RETURN KEYWORD_AWAIT KEYWORD_ASYNC
%token KEYWORD_REMOTE KEYWORD_VAR KEYWORD_STRING
%token KEYWORD_INT KEYWORD_FLOAT KEYWORD_BOOL KEYWORD_FUTURE
%token KEYWORD_CONTINUE KEYWORD_BREAK
%token KEYWORD_LEN KEYWORD_APPEND KEYWORD_READ KEYWORD_VOID

%nonassoc NOELSE
%nonassoc KEYWORD_ELSE
%nonassoc COLON
%nonassoc COMMA
%nonassoc ASSIGNNEW
%nonassoc LEFT_PARENTHE
%right ASSIGNMENT
%left LOGICAL_OR
%left LOGICAL_AND
%right LOGICAL_NOT
%left IS_EQUAL IS_NOT_EQUAL
%left IS_GREATER_THAN IS_LESS_THAN IS_GREATER_EQUAL IS_LESS_EQUAL
%left PLUS MINUS
%nonassoc LEFT_BRACKET
%left TIMES DIVIDE MODSIGN
%right NEGATIVE

%start functions
%type <Ast.functions> functions

%%

functions:
  p_functions EOF { $1 }

p_functions:
| { [] }
| NEWLINE p_functions  { $2 }
| p_function_decl p_functions { $1::$2 }
/*| p_function p_functions  { $1::$2 } */
```

```
p_function_annotation:
  { FuncNormal }
| KEYWORD_ASYNC { FuncAsync }
| KEYWORD_ASYNC KEYWORD_REMOTE { FuncAsyncRemote }

p_function_decl:
  /*   the variable here is actually an ID     */
  /* 1. func FuncName(parameters) retType */
  p_function_annotation KEYWORD_FUNC VARIABLE LEFT_PARENTHE p_parameters
RIGHT_PARENTHE
  p_type LEFT_BRACE NEWLINE p_statements RIGHT_BRACE
  /*{ FunctionProto($1, $3, [$7], $5) }*/
  { { ann = $1;
    fname = $3;
    typ = [$7];
    formals = $5;
    body = List.rev $10 } }
| /* 2. func FuncName(parameters)  */
  p_function_annotation KEYWORD_FUNC VARIABLE LEFT_PARENTHE p_parameters
RIGHT_PARENTHE
  LEFT_BRACE NEWLINE p_statements RIGHT_BRACE
  /* { FunctionProto($1, $3, [], $5) } */
  { { ann = $1;
    fname = $3;
    typ = [];
    formals = $5;
    body = List.rev $9 } }
| /* 3. func FuncName(parameters)  (retType1, retType2, ...)  */
  p_function_annotation KEYWORD_FUNC VARIABLE LEFT_PARENTHE p_parameters
RIGHT_PARENTHE
  LEFT_PARENTHE p_type_list RIGHT_PARENTHE
  LEFT_BRACE NEWLINE p_statements RIGHT_BRACE
  /* { FunctionProto($1, $3, $8, $5) } */
  { { ann = $1;
    fname = $3;
    typ = $8;
    formals = $5;
    body = List.rev $12 } }


/*p_function_impl:*/
  /*LEFT_BRACE NEWLINE p_statements RIGHT_BRACE   {  FunctionImpl($3) }*/
```

```
/*p_function: */
/*  p_function_prototype p_function_impl  {  Function($1, $2)   }  */


p_type_list:
  /* empty type list is not allowed  */
| p_type    {  [$1]  }
| p_type COMMA p_type_list  {  $1::$3  }

p_variable_list:
  /* empty variabie list is not allowed  */
| VARIABLE    {  [$1]  }
| VARIABLE COMMA p_variable_list  {  $1::$3  }

p_parameters:
  { [] }
| p_parameter    {  [$1]  }
| p_parameter COMMA p_parameters  {  $1::$3  }

p_parameter:
  VARIABLE p_type  {  ($2, $1)  }

p_expr_list:
  { [] }
| p_expr_list_required   {  $1  }

/*   at least one p_expr in the list  */
p_expr_list_required:
| p_expr   {  [$1]  }
| p_expr COMMA p_expr_list_required { $1::$3 }

p_expr:
  p_expr PLUS   p_expr { BinaryOp($1, Add, $3) }
| p_expr MINUS  p_expr { BinaryOp($1, Sub, $3) }
| p_expr TIMES  p_expr { BinaryOp($1, Mul, $3) }
| p_expr DIVIDE p_expr { BinaryOp($1, Div, $3) }
| p_expr MODSIGN p_expr { BinaryOp($1, Mod, $3) }
| p_expr IS_LESS_THAN p_expr { BinaryOp($1, LessThan, $3) }
| p_expr IS_GREATER_THAN p_expr {BinaryOp($1, GreaterThan, $3)}
| p_expr IS_NOT_EQUAL p_expr {BinaryOp($1, IsNotEqual, $3)}
| p_expr IS_EQUAL p_expr {BinaryOp($1, IsEqual, $3)}
| p_expr IS_LESS_EQUAL p_expr {BinaryOp($1, LessEqual, $3)}
| p_expr IS_GREATER_EQUAL p_expr {BinaryOp($1, GreaterEqual, $3)}
```

```
| p_expr LOGICAL_AND p_expr {BinaryOp($1, LogicalAnd, $3)}
| p_expr LOGICAL_OR p_expr {BinaryOp($1, LogicalOr, $3)}

| MINUS p_expr %prec NEGATIVE {  UnaryOp(Negative, $2) }
| LOGICAL_NOT p_expr {  UnaryOp(LogicalNot, $2) }

/* p_expr_list_required will cause reduce/reduce conflict   */
/*  FIXME or do not support a, b = b, a */
| p_expr ASSIGNMENT p_expr { AssignOp($1, $3) }

/* literals */
| INT_LITERAL     { Integer($1) }
| STRING_LITERAL  { String($1)  }
| FLOAT_LITERAL   { Float($1)   }
| BOOLEAN_LITERAL { Bool($1)    }

/* identifier */
| VARIABLE        { NamedVariable($1) }

/*  the variable here is actually an ID (for a function)  */
| VARIABLE LEFT_PARENTHE p_expr_list RIGHT_PARENTHE { FunctionCall($1,
$3)  }

/* built-in functions */
| KEYWORD_AWAIT  VARIABLE {  Await($2)  }
| KEYWORD_LEN    LEFT_PARENTHE p_expr RIGHT_PARENTHE { Len($3)  }
| KEYWORD_APPEND LEFT_PARENTHE p_expr_list RIGHT_PARENTHE { Append($3)  }
| KEYWORD_READ LEFT_PARENTHE p_expr RIGHT_PARENTHE {Read($3)}

| LEFT_PARENTHE p_expr RIGHT_PARENTHE { $2 }

| p_slice_type LEFT_BRACE p_expr_list RIGHT_BRACE { SliceLiteral($1,
List.length $3, $3) }
/* index */
| p_expr LEFT_BRACKET p_expr RIGHT_BRACKET { SliceIndex($1, $3) }
/* slice */
| p_expr LEFT_BRACKET p_expr COLON p_expr RIGHT_BRACKET { SliceSlice($1,
$3, $5) }
| p_expr LEFT_BRACKET COLON p_expr RIGHT_BRACKET { SliceSlice($1,
EmptyExpr, $4) }
| p_expr LEFT_BRACKET p_expr COLON RIGHT_BRACKET { SliceSlice($1, $3,
EmptyExpr) }
```

```
p_slice_type:
  LEFT_BRACKET RIGHT_BRACKET p_type { SliceType($3) }

p_type:
  KEYWORD_STRING {  StringType  }
| KEYWORD_INT    {  IntegerType }
| KEYWORD_FLOAT  {  FloatType   }
| KEYWORD_BOOL   {  BoolType    }
| KEYWORD_FUTURE {  FutureType  }
| KEYWORD_VOID   {  VoidType    }
| p_slice_type   {  $1 }


p_statements:
  { [] }
| p_statements NEWLINE     { $1 }
| p_statements p_statement  { $2::$1 }


p_if_statement:
  KEYWORD_IF LEFT_PARENTHE p_expr RIGHT_PARENTHE p_statement KEYWORD_ELSE
p_statement { IfStatement($3,$5, $7) }
| KEYWORD_IF LEFT_PARENTHE p_expr RIGHT_PARENTHE p_statement %prec NOELSE
{IfStatement($3, $5,EmptyStatement) }

/*  Declare:
    var a int = 5
    var a int
    Simple declare:
    a := 5
 */

/*p_simple_statement: */
/* need to check with local variables, works latter*/
/*  p_expr              { Expr($1) }
*/
/*| KEYWORD_VAR p_variable_list p_type_list ASSIGNMENT p_expr_list_required
{ SimpleDeclare($3, $2, $5)  }*/
/*| KEYWORD_VAR p_variable_list p_type_list { SimpleDeclare($3, $2,
[EmptyExpr])  }*/
/*| p_variable_list ASSIGNNEW p_expr_list_required { SimpleShortDecl($1,
$3)  }*/
```

```
p_statement:
  p_expr                 NEWLINE    { Expr($1) }
// Return
| KEYWORD_RETURN p_expr_list  NEWLINE    { Return($2) }
// If
| p_if_statement                   { $1 }
// Declare
| KEYWORD_VAR p_variable_list p_type ASSIGNMENT p_expr_list_required
NEWLINE  { Declare($2, $3, $5)  }
| KEYWORD_VAR p_variable_list p_type NEWLINE  { Declare($2, $3, [])  }
| p_variable_list ASSIGNNEW p_expr_list_required NEWLINE  { ShortDecl($1,
$3)}
// For loop
| KEYWORD_FOR LEFT_PARENTHE p_expr SEMICOLON p_expr SEMICOLON p_expr
RIGHT_PARENTHE p_statement {ForStatement($3, $5, $7, $9)}
| KEYWORD_FOR LEFT_PARENTHE SEMICOLON p_expr SEMICOLON RIGHT_PARENTHE
p_statement {  ForStatement(EmptyExpr, $4, EmptyExpr, $7)}
| KEYWORD_FOR LEFT_PARENTHE SEMICOLON SEMICOLON RIGHT_PARENTHE p_statement
{ForStatement(EmptyExpr, EmptyExpr, EmptyExpr, $6)  }
| KEYWORD_FOR LEFT_PARENTHE SEMICOLON p_expr SEMICOLON p_expr
RIGHT_PARENTHE p_statement {ForStatement(EmptyExpr, $4, $6 , $8)}
| KEYWORD_FOR LEFT_PARENTHE p_expr SEMICOLON p_expr SEMICOLON
RIGHT_PARENTHE p_statement {ForStatement($3, $5, EmptyExpr, $8)}
// Break
| KEYWORD_BREAK   NEWLINE        {  Break  }
// Continue
| KEYWORD_CONTINUE   NEWLINE     {  Continue  }
// Block
| LEFT_BRACE NEWLINE p_statements RIGHT_BRACE {Block(List.rev $3)}
```

9.2.5 sast.ml

```
open Ast

type sexpr = builtin_type list * sx
and sx =
    SEmptyExpr
  | SInteger of int
  | SFloat of float
```

```
  | SString of string
  | SBool of bool
  | SBinaryOp of sexpr * binary_operator * sexpr
  | SUnaryOp  of unary_operator * sexpr
  | SAssignOp of sexpr * sexpr
  | SFunctionCall of string * sexpr list
  | SNamedVariable of string
  | SLen of sexpr
  | SAppend of sexpr list
  | SSliceLiteral of builtin_type * int * sexpr list
  | SSliceIndex of sexpr * sexpr
  | SSliceSlice of sexpr * sexpr * sexpr
  | SAwait of string
  | SRead of sexpr


type sstatement =
    SEmptyStatement
  | SIfStatement of sexpr * sstatement * sstatement
  | SForStatement of sexpr * sexpr * sexpr * sstatement
  | SBreak
  | SContinue
  | SDeclare of string list * builtin_type * sexpr list
  | SShortDecl of string list * sexpr list
  | SReturn of sexpr list
  | SExpr of sexpr
  | SBlock of sstatement list

type sfunc_decl = {
  sann : func_annotation;
  sfname : string;
  styp : builtin_type list;
  sformals : bind list;
  sbody : sstatement list;
}

type functions = sfunc_decl list

let rec string_of_sexpr (tl, e) =
  "(" ^ String.concat ", " (List.map string_of_typ tl) ^ " : " ^ (match e
with
  SEmptyExpr -> "empty"
| SInteger(x) -> string_of_int x
```

```
| SFloat(x) -> string_of_float x
| SString(x) -> x
| SBool(x) -> string_of_bool x
| SBinaryOp(e1,op,e2) -> string_of_sexpr e1 ^ " " ^ string_of_op op ^ " " ^
string_of_sexpr e2
| SUnaryOp(op,e) -> string_of_uop op ^ string_of_sexpr e
| SAssignOp(e1,e2)-> string_of_sexpr e1 ^ " = " ^ string_of_sexpr e2
| SFunctionCall(f,el)->f ^ "(" ^ String.concat ", " (List.map
string_of_sexpr el) ^ ")"
| SNamedVariable(x) -> x
| SSliceLiteral(t,int,el) -> "[]"^string_of_typ t ^ "{"^ String.concat ", "
(List.map string_of_sexpr el) ^ "}"
| SSliceIndex(e1,e2)-> string_of_sexpr e1 ^ "[" ^ string_of_sexpr e2^ "]"
| SSliceSlice (e1,e2,e3)->string_of_sexpr e1 ^ "[" ^ string_of_sexpr e2
^":"^ string_of_sexpr e3^ "]"
| SLen(e1)->"len("^string_of_sexpr e1 ^ ")"
| SAppend(el)->"append("^ String.concat ", " (List.map string_of_sexpr el)
^ ")"
| SAwait(e)->"await "^e
| SRead(e)->"read " ^string_of_sexpr e
) ^ ")"
```

9.2.6 semant.ml

```
(* Semantic checking for the Digo compiler *)

open Ast
open Sast
open Llvm
open Llvm_analysis

module StringMap = Map.Make(String)

let check (functions) =

  (* Collect function declarations for built-in functions: no bodies. *)
  let built_in_decls =
    let builtins =
      [
        (* string related*)
        (* Accepts a C-layout string and returns a wrapped Digo String
```

```
object *)
        ("CreateString", {ann = FuncNormal; fname = "CreateString"; typ =
[StringType];
        formals = [(StringType,"string")] ; body=[]});
        (* Returns an empty Digo String object *)
        ("CreateEmptyString", {ann = FuncNormal; fname =
"CreateEmptyString"; typ = [StringType];
        formals = [] ; body=[]});
        (* concat(Digo String A + Digo String B) *)
        ("AddString", {ann = FuncNormal; fname = "AddString"; typ =
[StringType];
        formals = [(StringType,"string"); (StringType,"string")] ;
body=[]});
        (* the second parameter is a C-style string *)
        ("AddCString", {ann = FuncNormal; fname = "AddCString"; typ =
[StringType];
        formals = [(StringType,"string"); (StringType,"string")] ;
body=[]});
        (* Get a copy of Digo String *)
        ("CloneString", {ann = FuncNormal; fname = "CloneString"; typ =
[StringType];
        formals = [(StringType,"string")] ; body=[]});
        (* Returns strcmp(strA, strB) *)
        ("CompareString", {ann = FuncNormal; fname = "CompareString"; typ =
[IntegerType];
        formals = [(StringType,"string"); (StringType,"string")] ;
body=[]});
        (* Returns strlen *)
        ("GetStringSize", {ann = FuncNormal; fname = "GetStringSize"; typ =
[IntegerType];
        formals = [(StringType,"string")] ; body=[]});
        (* Get C-style string *)
        ("GetCStr", {ann = FuncNormal; fname = "GetCStr"; typ =
[StringType];
        formals = [(StringType,"string")] ; body=[]});

        (*could be any type at this point*)
        ("CreateSlice",{ann = FuncNormal; fname = "CreateSlice"; typ = [];
        formals = [] ; body = [] });
        ("SliceSlice",{ann = FuncNormal; fname = "SliceSlice"; typ = [];
        formals = [] ; body = [] });
        ("SliceAppends",{ann = FuncNormal; fname = "SliceAppend"; typ = [];
        formals = [] ; body = [] });
```

```
        ("SliceAppendn",{ann = FuncNormal; fname = "SliceAppend"; typ = [];
        formals = [] ; body = [] });
        ("SliceAppendf",{ann = FuncNormal; fname = "SliceAppend"; typ = [];
        formals = [] ; body = [] });
        ("SliceAppendF",{ann = FuncNormal; fname = "SliceAppend"; typ = [];
        formals = [] ; body = [] });
        ("CloneSlice",{ann = FuncNormal; fname = "CloneSlice"; typ = [];
        formals = [] ; body = [] });
        ("GetSliceSize",{ann = FuncNormal; fname = "CreateSliceSize"; typ =
[];
        formals = [] ; body = [] });

        ("SetSliceIndexDouble",{ann = FuncNormal; fname =
"SetSliceIndexDouble"; typ = [];
        formals = [] ; body = [] });
        ("SetSliceIndexFuture",{ann = FuncNormal; fname =
"SetSliceIndexFuture"; typ = [];
        formals = [] ; body = [] });
        ("SetSliceIndexString",{ann = FuncNormal; fname =
"SetSliceIndexString"; typ = [];
        formals = [] ; body = [] });
        ("SetSliceIndexInt",{ann = FuncNormal; fname = "SetSliceIndexInt";
typ = [];
        formals = [] ; body = [] });
        ("GetSliceIndexString",{ann = FuncNormal; fname =
"GetSliceIndexString"; typ = [];
        formals = [] ; body = [] });
        ("GetSliceIndexInt",{ann = FuncNormal; fname = "GetSliceIndexInt";
typ = [];
        formals = [] ; body = [] });
        ("GetSliceIndexDouble",{ann = FuncNormal; fname =
"GetSliceIndexDouble"; typ = [];
        formals = [] ; body = [] });
        ("GetSliceIndexFuture",{ann = FuncNormal; fname =
"GetSliceIndexFuture"; typ = [];
        formals = [] ; body = [] })

      ]
    in
    let add_bind map (name, ty) = StringMap.add name ty map
    in
      List.fold_left add_bind StringMap.empty builtins
    in
```

```
  (*
    This part is
    1. Add function name to symbol table
    2. Check for duplciate function naming and naming same with built-in
functions.
  *)
  let add_func map fd =
    let built_in_err = "Semant Err: function " ^ fd.fname ^ " may not be
defined"
    and dup_err = "Semant Err: duplicate function " ^ fd.fname
    and make_err er = raise (Failure er)
    and n = fd.fname (* Name of the function *)

    in match fd with (* No duplicate functions or redefinitions of built-
ins *)
          _ when StringMap.mem n built_in_decls -> make_err built_in_err
        | _ when StringMap.mem n map -> make_err dup_err
        | _ ->  StringMap.add n fd map
  in

  (* This part is
    1. Collect all function names into one symbol table
  *)
  let function_decls = List.fold_left add_func built_in_decls functions
  (* pass here *)
  in

  (* Find_func finds a function from our symbol table. It can be used:
    1. check if every program main/master function
    2. check if function existed for function call
  *)
  let find_func s =
    try StringMap.find s function_decls
    with Not_found -> raise (Failure ("unrecognized function " ^ s))
  in

  (* Ensure "main" is defined *)
  let _ = find_func "digo_main" in

  (* Check semantic in function *)
  let check_function func =
```

```
    let _ =
      (* check digo_main is valid *)
      match func.fname with
      "digo_main" ->
          if (List.length func.formals) > 0
          then raise(Failure "Semant Err: digo_main should be no-
arugment.");
          (match func.typ with
            [VoidType] -> ignore()
            | _ -> raise(Failure "Semant Err: digo_main should be void
type.")
          )
      | _ -> ignore()
    in
    let check_binds kind binds =
      List.iter(function
        (VoidType, b) -> raise(Failure("illegal VoidType in " ^ kind ^" :
"^b))
        | _-> ()
      ) binds;
      let rec dup_check = function
        [] -> []
      | ((_,n1)::(_,n2)::_) when n1 = n2 ->
          raise(Failure("duplicate in "^ kind ^ " : " ^ n1))
      | _::vdecls -> dup_check vdecls
      in
        dup_check (List.sort (fun (_,n1)(_,n2)-> compare n1 n2) binds)
    in
    ignore(check_binds "argument" func.formals);

    (* Type of each variable *)
    let symbols = Hashtbl.create 500 in (* symbol: hastbl for variables *)
    let futures = Hashtbl.create 500 in (* symbol: hastbl for future
variables *)
    let future_slice = Hashtbl.create 500 in
    let _ = List.iter (fun (t, n) -> Hashtbl.add symbols n t) func.formals
in
    let type_of_identifier n =
      if Hashtbl.mem symbols n then Hashtbl.find symbols n
      else raise (Failure("Sement Err: undeclared identifier " ^ n))
    in
    let check_assign lvaluet rvaluet err =
```

```
      if lvaluet = rvaluet then lvaluet else raise (Failure err)
    in
    let func_of_future n =
      let fname = if Hashtbl.mem futures n then Hashtbl.find futures n
      else raise (Failure("Semant Err: undeclared future object " ^ n)) in
      find_func fname
    in
    let get_type_in_slicetype stp = match stp with
          SliceType(StringType)      ->  StringType
        | SliceType(IntegerType)     ->  IntegerType
        | SliceType(FloatType)       ->  FloatType
        | SliceType(FutureType)      ->  FutureType
        | _                          ->  raise(Failure("invalide slice
type"))
    in
    (* Return a semantically-checked expression, i.e., with a type *)
    let rec expr e = match e with
        Integer(x)  -> ([IntegerType], SInteger(x))
      | Float(x) -> ([FloatType], SFloat(x))
      | Bool(x)  -> ([BoolType], SBool(x))
      | String (x)   ->
        ([StringType],SString(x))
      | EmptyExpr -> ([VoidType],SEmptyExpr)
      | NamedVariable s  ->
        ([type_of_identifier s], SNamedVariable(s))
      | AssignOp(var, e) ->
        (* this part check left side is valid: only accept
nameddvariable/slice index *)
        (match var with
        | SliceIndex(e1,e2)  ->
          (match e1 with
          | NamedVariable(s) ->
            let (var_typ,_) = expr var
            and (ret_typ,e') = expr e in
            let err = "illegal assignment" in
            ([check_assign (List.hd var_typ) (List.hd ret_typ) err],
              SAssignOp((expr var),(ret_typ, e')))
          | _  -> raise(Failure("AssignOp error: left hand side is
invalid"))
          )
        | NamedVariable(x)   ->
          (* check named variable type macth with expression type *)
          let var_typ = type_of_identifier x
```

```
          and (ret_typ,e') = expr e in
          let err = "illegal assignment " in
          let _ = match var_typ with
            FutureType -> (match e' with
                SFunctionCall(fname,_) -> Hashtbl.add futures x fname
              | SSliceIndex((_,SNamedVariable(slice_name)),_) ->
                  Hashtbl.replace futures x (Hashtbl.find futures
slice_name)
              | SNamedVariable(slice_name) -> Hashtbl.replace futures x
(Hashtbl.find
                                                futures slice_name)
              | _ -> raise(Failure("AssignOp error: left hand side "
^string_of_expr(e)
                           ^" is invalid"))
              )
            | SliceType(FutureType) ->
              (match e with
                NamedVariable(slice_name) -> Hashtbl.replace futures x
(Hashtbl.find
                                                futures slice_name)
              | Append(expl) ->
                ( match (List.hd expl) with
                NamedVariable(slice_name)-> Hashtbl.replace futures x
(Hashtbl.find
                                                  futures slice_name)
                | _ -> raise(Failure("AssignOp error: left hand side "
^string_of_expr(e)
                    ^" is invalid"))
                )
              | _ -> raise(Failure("AssignOp error: left hand side "
^string_of_expr(e) ^
                    " is invalid"))
              )
            | _ -> ignore()
          in
            ([check_assign var_typ (List.hd ret_typ) err], SAssignOp(expr
var,(ret_typ, e')))
        | _ ->
          raise(Failure("AssignOp error: left hand side is invalid"))
        )
      | UnaryOp(op, e)   ->
        let (ret_typl,e') = expr e in
        let op_typ = match op with
```

```
              Negative when (List.hd ret_typl) = IntegerType || (List.hd
ret_typl) = FloatType -> ret_typl
            | LogicalNot when (List.hd ret_typl) = BoolType -> ret_typl
            | _ -> raise (Failure ("Semant Err: illegal unary operator "
(*^ stringify_unary_operator op e ^
              " expression type "^stringify_builtin_type ret_typ*)  )) in
        (op_typ,SUnaryOp(op,(ret_typl,e')))

      | BinaryOp(e1, op, e2)  ->
        let (ret_typl1,e1') = expr e1 and (ret_typl2,e2') = expr e2 in
          (* All binary operators require operands of the same type *)
        let same = ret_typl1 = ret_typl2 in
          (* Determine expression type based on operator and operand types
*)
        let op_typ = match op with
            Add | Sub | Mul | Div | Mod when same && ((List.hd ret_typl1) =
IntegerType) -> IntegerType
          | Add | Sub | Mul | Div | Mod when same && ((List.hd ret_typl1) =
FloatType) -> FloatType
          | IsEqual | IsNotEqual  when same -> BoolType
          | LessThan | LessEqual | GreaterThan | GreaterEqual
            when same && ((List.hd ret_typl1) = IntegerType || (List.hd
ret_typl1) = FloatType || (List.hd
            ret_typl1) = StringType) -> BoolType
          | LogicalAnd | LogicalOr when same && ((List.hd ret_typl1) =
BoolType) -> BoolType
          | Add when same && (List.hd ret_typl1) == StringType ->
StringType
          | _ -> raise ( Failure ("Semant Err: illegal binary operator "
^(string_of_op op)^" for type "^
          (String.concat ", " (List.map string_of_typ ret_typl1)) ^ " and
type "^ (String.concat ", "(List.map
          string_of_typ ret_typl2))^ " in "^ string_of_expr e)) in
        ([op_typ],SBinaryOp((ret_typl1,e1'),op,(ret_typl2,e2')))
      | FunctionCall(fname, args) ->
      (match fname with
      | "print" -> ([VoidType],SFunctionCall(fname,(List.map expr args)))
      | "println" -> ([VoidType],SFunctionCall(fname,(List.map expr args)))
      | _ ->
        let fd = find_func fname in
        let param_length = List.length fd.formals in
        if List.length args != param_length then
          raise (Failure ("Semant Err: different number of aruguments
```

```
passed. Expected "
                  ^ string_of_int param_length ^ " aruguments but " ^
string_of_int
                  (List.length args) ^" aruguments provided in function " ^
fname))
        else
          let check_call (ft, _) e =
            let (ret_typl,e') = expr e in
            let err = "Semant Err: illegal argument found" in
          ([check_assign ft (List.hd ret_typl) err],e')
          in
          let args' = List.map2 check_call fd.formals args in
          let tpy' = match fd.ann with
            FuncNormal -> fd.typ
          | _ -> [FutureType]
          in (tpy',SFunctionCall(fname,args'))
      )
      | Append(exl) ->
        let exlen = List.length exl in
        if exlen = 2
        then
          let (ret_typl1,e1') = expr (List.hd exl) in
          let (ret_typl2,e2') = expr (List.nth exl 1) in
          ( match ret_typl1 with
          | [SliceType(x)] ->
            if x = (List.hd ret_typl2)
            then
              (* check future object type when appending. if it's the first
one, add (slice, func of future) *)
              let _ = match x with
                  FutureType ->
                    (match e1' with
                        SNamedVariable(slice_name)->
                          let check_eq a b =
                            if a = b then () else raise (Failure ( "Semant
Err: cannot append future with async
                            function " ^b ^" to a slice of future object
with function " ^ a)) in
                            ( match e2' with
                                SNamedVariable(future_object)->
                                (* let SNamedVariable(future_object) =
e2'*)
                                let future_func = if Hashtbl.mem futures
```

```
future_object then Hashtbl.find futures
                                future_object
                                else raise (Failure ( "Semant Err:
undeclared future object " ^ future_object)) in
                                ignore(if Hashtbl.mem futures slice_name
then check_eq (Hashtbl.find futures
                                slice_name) future_func else
ignore(Hashtbl.add futures slice_name future_func))
                              | SFunctionCall(future_func,_)->
                                ignore(if Hashtbl.mem futures slice_name
then check_eq (Hashtbl.find futures slice_name)
                                future_func else ignore(Hashtbl.add futures
slice_name future_func))
                              | _ -> ignore()
                            )
                         | _ -> raise (Failure ( "Semant Err: left hand side
in " ^ string_of_expr(e) ^ "
                                  should be a variable"))
                      )
                  | _ -> ignore()
              in (ret_typl1,SAppend([(ret_typl1,e1');(ret_typl2,e2')]))
            else
              raise(Failure("Semantic Err: cannot append type
"^(string_of_typ (List.hd ret_typl2))^ " to "
                            ^ (string_of_typ (List.hd ret_typl1)) ^ " in
expression " ^ (string_of_expr e)))
          | _  -> raise(Failure("Built-in Append being called on non-slice
object"))
          )
        else raise(Failure("Append needs two objects"))
      | Len(ex)              ->
            let (ret_typl,e') = expr ex in
            ( match ret_typl with
            | [SliceType(_)] -> ([IntegerType],SLen((ret_typl,e')))
            | [StringType] -> ([IntegerType],SLen((ret_typl,e')))
            | _  -> raise(Failure("Built-in Len being called on non-slice
object " ^ string_of_expr(ex)))
            )
    | Read(e) ->
            (* only string type*)
          let (ret_typ,e') = expr e in
          let ck = match (List.hd ret_typ) with
            StringType -> ([SliceType(StringType)],SRead((ret_typ,e')))
```

```
            |_ -> raise (Failure ("error: read is not supported for "^
string_of_typ (List.hd ret_typ)))
          in ck
    | SliceLiteral(btyp, slice_len , expl)  ->
      let rt_typ = get_type_in_slicetype btyp in
      let check_type e_ =
      let (ret_typl,e') = expr e_ in
      let err = "illegal slice type found" in
      ([check_assign rt_typ (List.hd ret_typl) err],e')
      in
      let sexpl = List.map check_type expl in
      ([btyp],SSliceLiteral(btyp, slice_len , sexpl))
    | SliceIndex(e1,e2)                      ->
      let (ret_typl1,e1') = expr e1 and (ret_typl2,e2') = expr e2 in
      let _ = match ret_typl1 with
        [SliceType(_)]  -> ()
      | _             -> raise(Failure("illgal slice indexing on non-slice
object"))
      in
      let _ = match ret_typl2 with
      | [IntegerType] -> ()
      | _ -> raise(Failure("illgal slice index: non-integer index"))
      in
      let rt_typ = get_type_in_slicetype (List.hd ret_typl1) in
      ([rt_typ],SSliceIndex((ret_typl1,e1'),(ret_typl2,e2')))
    | SliceSlice(e1,e2,e3)                   ->
      let (ret_typl1,e1') = expr e1 and (ret_typl2,e2') = expr e2 and
(ret_typl3,e3') = expr e3 in
      let _ = match ret_typl1 with
        [SliceType(_)]  -> ()
      | _             -> raise(Failure("illgal slice slicing on non-slice
object"))
      in
      let _ = match ret_typl2 with
      | [IntegerType] -> ()
      | [VoidType]    -> ()
      | _ -> raise(Failure("illgal slice slice: non-integer index"))
      in
      let _ = match ret_typl3 with
      | [IntegerType] -> ()
      | [VoidType]    -> ()
      | _ -> raise(Failure("illgal slice slice: non-integer index"))
      in
```

```
(ret_typl1,SSliceSlice((ret_typl1,e1'),(ret_typl2,e2'),(ret_typl3,e3')))
      | Await(n) ->
            (* await futureVar *)
            (* return [list of returned aysn val types, SAwait(n)]*)
          let fd = func_of_future n in
          (fd.typ, SAwait(n))

    in

    let check_bool_expr e =
      let (ret_typl,e') = expr e in
      if ((List.hd ret_typl) != BoolType && e != EmptyExpr)
      then raise (Failure ("expected Boolean expression"))
      else ([List.hd ret_typl],e')
    in

    let rec check_stmt = function
        EmptyStatement                    ->  SEmptyStatement
      | IfStatement(e,st1,st2)            ->  SIfStatement(check_bool_expr
e, check_stmt st1, check_stmt st2)
      | ForStatement(e1,e,e2,st3)         ->  SForStatement(expr e1,
check_bool_expr e, expr e2, check_stmt st3)
      | Break                             ->  SBreak
      | Continue                          ->  SContinue
      | Declare(nl,t,el) ->
        let check_dup_var n =
          if Hashtbl.mem symbols n then raise (Failure ("Semant Err:
duplicate local variable declarations ( " ^ n
          ^ " )")) else  ignore(Hashtbl.add symbols n t)
        in List.iter check_dup_var nl;
        let ck = match el with
          [] -> SDeclare(nl, t, [([VoidType],SEmptyExpr)])
          | _ ->
            if List.length nl != List.length el then raise (Failure
("assignment mismatch: "^string_of_int (List.length
            nl) ^" variables but "^ string_of_int (List.length el) ^ "
values"));
            let ret_list = List.map (fun e -> expr e) el in
            let _ = List.iter (fun (rt,_) -> ignore(check_assign t (List.hd
rt) "illegal assignment type")) ret_list
            in SDeclare(nl, t, ret_list)
        in ck
```

```
      | ShortDecl(nl,el) ->
        let ret_list = List.map (fun e -> expr e) el in
        let _ = match (List.hd ret_list) with
            (tyl,SFunctionCall(_,_)) | (tyl, SAwait(_)) ->
            if List.length nl != List.length tyl then raise (Failure
("short decl assignment mismatch: "^
            string_of_int (List.length nl) ^" variables but "^
string_of_int (List.length tyl) ^ " values"));
          | _ ->
            if List.length nl != List.length el then raise (Failure ("short
decl assignment mismatch: "^
            string_of_int (List.length nl) ^" variables but "^
string_of_int (List.length el) ^ " values"));
        in
        let check_dup_var n (rt,_) =
          if Hashtbl.mem symbols n then raise (Failure ("Semant Err:
duplicate local variable declarations ( "
          ^ n ^ " )")) else  ignore(Hashtbl.add symbols n (List.hd rt))
        in
        let check_dup_var_function n rt =
          if Hashtbl.mem symbols n then raise (Failure ("Semant Err:
duplicate local variable declarations ( "
          ^ n ^ " )")) else  ignore(Hashtbl.add symbols n rt)
        in
        let _ = match (List.hd ret_list) with
          ([FutureType],SFunctionCall(fname,_)) ->
            List.iter (fun n -> if Hashtbl.mem symbols n
            then raise (Failure ("Semant Err: duplicate local variable
declarations ( " ^ n ^ " )"))
            else ignore(Hashtbl.add symbols n FutureType)) nl;
            List.iter (fun n -> if Hashtbl.mem futures n
            then raise (Failure ("Semant Err: duplicate future variable
declarations ( " ^ n ^ " )"))
            else  ignore(Hashtbl.add futures n fname)) nl;
          | ([FutureType], SSliceIndex((_,SNamedVariable(slice_name)),_))
->
            let add_futures n ret =
              ( match ret with
                  (_, SSliceIndex((_,SNamedVariable(slice_name)),_)) ->
                      Hashtbl.replace futures n (Hashtbl.find futures
slice_name)
                  | _ -> ignore()
              ) in
```

```
            List.iter2 add_futures nl ret_list;
            List.iter2 check_dup_var nl ret_list

          | (etl,SFunctionCall(_,_)) | (etl, SAwait(_))   ->
              List.iter2 check_dup_var_function nl etl
          | ([SliceType(FutureType)], SSliceLiteral(_,_,_)) ->
              (* add (slice_name, []) to future_slice. this is to
initialize a list of functions*)
              List.iter (fun n->Hashtbl.add future_slice n []) nl;
              List.iter2 check_dup_var nl ret_list
          | _ -> List.iter2 check_dup_var nl ret_list
        in
        SShortDecl(nl, ret_list)
      | Expr(e)                           ->  SExpr(expr e)
      | Return(el)                        ->
            (match func.typ with
            [VoidType] -> if (List.length el) > 0
                          then raise (Failure ("Semant Err: too many
arguments to return in a void function "^
                          func.fname)) else raise (Failure ("Semant Err:
"))
            | _ ->
              (match (List.length func.typ - List.length el) with
              0 -> let ret_list = List.map (fun e -> expr e) el in
                let _ = List.iter2 (fun (rt,_) ft -> ignore(check_assign
(List.hd rt) ft ("cannot use "^(string_of_typ
                (List.hd rt)) ^" in return argument of function "
^func.fname))) ret_list func.typ
                in SReturn(ret_list)
              | _ when (List.length func.typ - List.length el)> 0 ->
                    raise (Failure ("Semant Err: not enough arguments to
return in function "^func.fname))
              | _ -> raise (Failure ("Semant Err: too many arguments in
function "^func.fname))
              ))
      | Block(stl)                        ->
        let rec check_stmt_list = function
          [Return _ as s] -> [check_stmt s]
        | Return _ :: _       -> raise(Failure ("Statements appear after
Return"))
        | Block b::ss     -> check_stmt_list (b@ss)
        | s::ss           -> let a = check_stmt s in a :: check_stmt_list
ss
```

```
        | []              ->   [SEmptyStatement]
        in
        SBlock(check_stmt_list stl)
    in

    let rec generate_default_return_value func_typ =
      (
        match func_typ with
          [VoidType] -> []
          | _ ->
            let rec make_arr = function
              [] -> []
              | IntegerType :: ar -> Integer(0) :: make_arr ar
              | FloatType :: ar -> Float(0.0) :: make_arr ar
              | StringType :: ar -> String("") :: make_arr ar
              | _ :: ar -> make_arr ar
            in
            let new_return = make_arr func_typ in
            [check_stmt (Return(new_return))]
      )

    in (* body of check_function *)
    { sann = func.ann;
      styp = func.typ;
      sfname = func.fname;
      sformals = func.formals;
      sbody =
        match check_stmt (Block func.body) with
        SBlock(stl) ->
          (match List.hd (List.rev stl) with
          SReturn(x) ->  stl
          | _  -> let default_return = generate_default_return_value
func.typ in
                    stl@default_return
          )
      | _      -> raise(Failure("function body does not form"))
    }
  in  List.map check_function functions
;;
```

9.2.7 metadata_gen.ml

```
open Ast

module StringMap = Map.Make(String)

let rec stringify_builtin_type = function
IntegerType -> "int"
| FloatType -> "double"
| StringType -> "string"
| SliceType(typ) -> "slice"
| BoolType   ->  "int"
| FutureType ->  "future"
| VoidType -> "void"
;;

let rec stringify_builtin_types = function
[]             ->   ""
| (typ :: typs)  ->   stringify_builtin_type typ ^ ", " ^
stringify_builtin_types typs
;;

let rec stringify_builtin_type_list l =
  let s = stringify_builtin_types l in
    String.sub s 0 (String.length s-2)
;;

let rec stringify_parameter (bind : builtin_type * string) : string =
     let (typ, name) = bind in
       stringify_builtin_type typ
;;

let rec stringify_parameters = function
   []           ->      ""
 | (par :: pars) ->      stringify_parameter par ^ ", " ^
stringify_parameters pars
;;

let rec stringify_parameter_list l =
  let s = stringify_parameters l in
    if String.length s == 0 then
      ""
    else
```

```
    String.sub s 0 (String.length s-2)
;;

let rec stringify_func_metadata f (ann_str : string) : string =
  "; FUNC DECLARE BEGIN\n" ^ "; FUNC_NAME = '" ^ f.fname ^ "'\n; FUNC_ANNOT
= '" ^ ann_str ^
  "'\n; PARAMETERS = '" ^ stringify_parameter_list f.formals ^ "'\n" ^ ";
RETURN_TYPE = '" ^
  stringify_builtin_type_list f.typ ^ "'\n; FUNC DECLARE END\n"
;;

let rec print_metadata f =
  match f.ann with
  | FuncNormal -> ()
  | FuncAsync -> print_endline (stringify_func_metadata f "async")
  | FuncAsyncRemote -> print_endline (stringify_func_metadata f "async
remote")
;;

let rec print_mul_metadata = function
    []  ->   print_endline ""
  | (func :: funcs)  ->   print_metadata func ; print_mul_metadata funcs
;;

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let functions = Parser.functions Scanner.tokenize lexbuf in
  print_endline "; DIGO Async Function Metadata BEGIN\n\n; VERSION = 1\n";
  print_mul_metadata functions;
  print_endline "\n; DIGO Async Function Metadata END\n";
```

9.2.8 codegen.ml

```
open Llvm
open Ast
open Sast

module StringMap = Map.Make(String)

let translate(functions) =
```

```
  let context = global_context () in
  let the_module = create_module context "Digo" in
  let _ = set_data_layout "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
the_module in
  let _ = set_target_triple "x86_64-pc-linux-gnu" the_module in

  let i64_t      = i64_type    context
    and i8_t       = i8_type     context
    and i1_t       = i1_type     context
    and float_t    = double_type context
    and void_t     = void_type   context in

    let ltype_of_typ = function
        IntegerType  -> i64_t
      | FloatType    -> float_t
      | BoolType     -> i1_t
      | StringType   -> pointer_type i8_t
      | SliceType(x)    -> pointer_type i8_t
      | FutureType   -> pointer_type i8_t   (*needs work*)
      | VoidType     -> void_t
    in

(* print built-in function *)
  let printf_t =
      var_arg_function_type void_t [|(pointer_type i8_t)|]  in
  let printf =
      declare_function "print" printf_t the_module in
  let printfln_t =
      var_arg_function_type void_t [|(pointer_type i8_t)|]  in
  let printfln =
      declare_function "println" printfln_t the_module in

(* String related functions *)
  let createString_t=
      function_type (pointer_type i8_t) [|pointer_type i8_t|] in
  let createString=
      declare_function "CreateString" (createString_t) the_module in
  let addString_t=
      function_type (pointer_type i8_t) [|pointer_type i8_t; pointer_type
i8_t|] in
  let addString=
      declare_function "AddString" (addString_t) the_module in
  let compareString_t=
```

```
        function_type (i64_t) [|pointer_type i8_t; pointer_type i8_t|] in
  let compareString=
        declare_function "CompareString" (compareString_t) the_module in
  let cloneString_t=
        function_type (pointer_type i8_t) [|pointer_type i8_t|] in
  let cloneString=
        declare_function "CloneString" (cloneString_t) the_module in
  let lenString_t=
      function_type (i64_t) [|pointer_type i8_t|] in
  let lenString=
      declare_function "GetStringSize" (lenString_t) the_module in

(* slice related built-in functions *)
    let createSlice_t =
        function_type (pointer_type i8_t) [|i64_t|] in
    let createSlice =
        declare_function "CreateSlice" (createSlice_t) the_module in
    let sliceAppend_t =
        var_arg_function_type (pointer_type i8_t) [|(pointer_type i8_t)|]
in
    let sliceAppend =
        declare_function "SliceAppend" sliceAppend_t the_module in
    let sliceSlice_t =
        function_type (pointer_type i8_t) [|(pointer_type
i8_t);i64_t;i64_t|] in
    let sliceSlice =
        declare_function "SliceSlice" (sliceSlice_t) the_module in
    let getSliceSize_t =
        function_type i64_t [|(pointer_type i8_t)|] in
    let getSliceSize =
        declare_function "GetSliceSize" (getSliceSize_t) the_module in
    let setSliceIndexDouble_t =
        function_type float_t [|(pointer_type i8_t);i64_t;float_t|] in
    let setSliceIndexDouble =
        declare_function "SetSliceIndexDouble" (setSliceIndexDouble_t)
the_module in
    let setSliceIndexFuture_t =
        function_type (pointer_type i8_t) [|(pointer_type
i8_t);i64_t;(pointer_type i8_t)|] in
    let setSliceIndexFuture =
        declare_function "SetSliceIndexFuture" (setSliceIndexFuture_t)
the_module in
    let setSliceIndexString_t =
```

```
        function_type (pointer_type i8_t) [|(pointer_type
i8_t);i64_t;(pointer_type i8_t)|] in
    let setSliceIndexString =
        declare_function "SetSliceIndexString" (setSliceIndexString_t)
the_module in
    let setSliceIndexInt_t =
        function_type i64_t [|(pointer_type i8_t);i64_t;i64_t|] in
    let setSliceIndexInt =
        declare_function "SetSliceIndexInt" (setSliceIndexInt_t) the_module
in
    let getSliceIndexDouble_t =
        function_type float_t [|(pointer_type i8_t);i64_t|] in
    let getSliceIndexDouble =
        declare_function "GetSliceIndexDouble" (getSliceIndexDouble_t)
the_module in
    let getSliceIndexFuture_t =
        function_type (pointer_type i8_t) [|(pointer_type i8_t);i64_t|] in
    let getSliceIndexFuture =
        declare_function "GetSliceIndexFuture" (getSliceIndexFuture_t)
the_module in
    let getSliceIndexString_t =
        function_type (pointer_type i8_t) [|(pointer_type i8_t);i64_t|] in
    let getSliceIndexString =
        declare_function "GetSliceIndexString" (getSliceIndexString_t)
the_module in
    let getSliceIndexInt_t =
        function_type i64_t [|(pointer_type i8_t);i64_t|] in
    let getSliceIndexInt =
        declare_function "GetSliceIndexInt" (getSliceIndexInt_t) the_module
in
    let cloneSlice_t =
        function_type (pointer_type i8_t) [|(pointer_type i8_t)|] in
    let cloneSlice =
        declare_function "CloneSlice" (cloneSlice_t) the_module in

(* red function *)
    let readFile_t=
          function_type (pointer_type i8_t) [|pointer_type i8_t|] in
    let readFile=
          declare_function "ReadFile" (readFile_t) the_module in

(* gc related functions*)
    let gc_create_trace_map_t =
```

```
          function_type (pointer_type i8_t) [| |] in
    let gc_create_trace_map =
          declare_function "__GC_CreateTraceMap" (gc_create_trace_map_t)
the_module in
    let gc_trace_t =
            function_type (void_t) [| (pointer_type i8_t); (pointer_type
i8_t) |] in
    let gc_trace =
            declare_function "__GC_Trace" (gc_trace_t) the_module in
    let gc_no_trace_t =
              function_type (void_t) [| (pointer_type i8_t); (pointer_type
i8_t) |] in
    let gc_no_trace =
              declare_function "__GC_NoTrace" (gc_no_trace_t) the_module in
    let gc_release_all_t =
              function_type (void_t)  [| (pointer_type i8_t) |] in
    let gc_release_all =
              declare_function "__GC_ReleaseAll" (gc_release_all_t)
the_module in

    let rec gc_gen_notrace_from_retval builder trace_map eexpr =
      match eexpr with
        []     ->    builder
        | ((e, e_typl) :: el)  ->  (match e_typl with
          1 -> ignore(build_call gc_no_trace [| trace_map; e |] ""
builder);
          gc_gen_notrace_from_retval builder trace_map el
        | _ -> gc_gen_notrace_from_retval builder trace_map el
        )
    in


(* usr functions *)
  let function_decls = Hashtbl.create 5000 in
  let function_delc fdecl=
      let name = fdecl.sfname
      and argument_types =
        Array.of_list (List.map (fun (t,_) -> ltype_of_typ t)
fdecl.sformals) in
      let stype =
        match fdecl.styp with
        [VoidType] -> void_t
```

```
        | _ ->
        struct_type context (Array.of_list (List.map ltype_of_typ
fdecl.styp))
      in
      let ftype = function_type stype argument_types
      in
      Hashtbl.replace function_decls name (define_function name ftype
the_module,fdecl) in
  let _ = List.iter function_delc functions in

  let find_func s =
      if Hashtbl.mem function_decls s then Hashtbl.find function_decls s
      else raise (Failure ("unrecognized function " ^ s))
    in

  let build_function_body fdecl=
      let (the_function,_) = find_func fdecl.sfname
      in
      let builder =
        builder_at_end context (entry_block the_function) in
      (*  GC: create the tracing map  *)
      let gc_trace_map_obj =
        build_call gc_create_trace_map [| |] "gc_trace_map_obj" builder in
      let futures = Hashtbl.create 5000 in
      let func_of_future n =
        let fname = if Hashtbl.mem futures n then Hashtbl.find futures n
        else raise (Failure("Err: undeclared future object " ^ n)) in
        find_func ("digo_linker_await_func_"^fname)
      in
      let local_vars = Hashtbl.create 5000 in
      let add_formal (t, n) p =
        set_value_name n p;
        let local = build_alloca (ltype_of_typ t) n builder in ignore
(build_store p local builder);
        Hashtbl.replace local_vars n local
      in
      ignore (List.iter2 add_formal fdecl.sformals (Array.to_list (params
the_function)));

      let add_var_decl id llvalue = Hashtbl.add local_vars id llvalue
      in
      let lookup n = if Hashtbl.mem local_vars n then Hashtbl.find
local_vars n
```

```
        else raise (Failure("cannot find symbol in local_vars"))
      in
      let get_type_in_slicetype = function
            SliceType(StringType)      ->  StringType
          | SliceType(IntegerType)     ->  IntegerType
          | SliceType(FloatType)       ->  FloatType
          | SliceType(FutureType)      ->  FutureType
          | _                          ->  raise(Failure("invalide slice
type"))
      in
      let get_slice_argument_number = function
        StringType  -> const_int i64_t 1
      | IntegerType -> const_int i64_t 3
      | FloatType   -> const_int i64_t 4
      | FutureType  -> const_int i64_t 6
      | _           -> raise(Failure("invalide slice type"))
      in

      (*   Now we just infer from the i8* type.
      Obsolete:
      function expr only returns the llvalue here,
        but the GC wants to know the type of this llvalue in the
        form of Digo type.

        Since GC is introduced after this part of code is written,
           to avoid changing the prototype of `expr`,
        I have to use a ref value here to store whether
           the evaluation result is a Digo Object.
      *)

      let expr_is_lvalue_obj = ref 0 in

      (*  This function takes the return value of a function call;
          adds it to GC tracing map; and sets the is_lvalue_obj to 1
indicating
          it is a Digo Object  *)
      let rec gc_inject func_ret_val builder =
        ignore(build_call gc_trace [|gc_trace_map_obj; func_ret_val|] ""
builder);
        expr_is_lvalue_obj := 1;
        func_ret_val
      in
```

```
      let rec expr builder (e_typl,e) =
        expr_is_lvalue_obj := 0;
        match e with
          SEmptyExpr  ->  const_int i1_t 1      (*cannot changed since for
loop needs boolean value*)
        | SBinaryOp(ex1,op,ex2) when List.hd e_typl = FloatType
->
          let e1 = expr builder ex1
          and e2 = expr builder ex2 in
          (match op with
          Add       ->  build_fadd
          | Sub       ->  build_fsub
          | Mul       ->  build_fmul
          | Div       ->  build_fdiv
          | Mod       ->  build_frem
          | _         ->  raise(Failure("binary operation is invalid and
should be rejected in semant"))
          ) e1 e2 "tmp" builder
        | SBinaryOp(ex1,op,ex2) when List.hd e_typl = IntegerType
->
          let e1 = expr builder ex1
          and e2 = expr builder ex2 in
          (match op with
          Add       ->  build_add
          | Sub       ->  build_sub
          | Mul       ->  build_mul
          | Div       ->  build_sdiv
          | Mod       ->  build_srem
          | _         ->  raise(Failure("binary operation is invalid and
should be rejected in semant"))
          ) e1 e2 "tmp" builder
        | SBinaryOp(ex1,op,ex2) when List.hd e_typl = BoolType
->
          let e1 = expr builder ex1
          and e2 = expr builder ex2 in
          (match ex1 with
            ([FloatType],_)                                    ->
              let condition = (match op with
                LessThan    ->  build_fcmp Fcmp.Olt
              | LessEqual   ->  build_fcmp Fcmp.Ole
              | GreaterThan   ->  build_fcmp Fcmp.Ogt
              | GreaterEqual  ->  build_fcmp Fcmp.Oge
              | IsEqual     ->  build_fcmp Fcmp.Oeq
```

```
              | IsNotEqual  ->  build_fcmp Fcmp.One
              | _ -> raise(Failure("invalid binary operation: should be
rejected in semant"))
              ) e1 e2 "tmp" builder in
              build_select condition (const_float float_t 1.0) (const_float
float_t 0.0) "booleantofloat" builder
          | ([IntegerType],_)                                    ->
              let condition = (match op with
                LessThan    ->  build_icmp Icmp.Slt
              | LessEqual   ->  build_icmp Icmp.Sle
              | GreaterThan   ->  build_icmp Icmp.Sgt
              | GreaterEqual  ->  build_icmp Icmp.Sge
              | IsEqual     ->  build_icmp Icmp.Eq
              | IsNotEqual  ->  build_icmp Icmp.Ne
              | _ -> raise(Failure("invalid binary operation: should be
rejected in semant"))
              ) e1 e2 "tmp" builder in
              build_select condition (const_int i64_t 1) (const_int i64_t
0) "booleantoint" builder
          | ([BoolType],_)                                       ->
              let condition = (match op with
              | LogicalAnd  ->  build_and
              | LogicalOr   ->  build_or
              | _ -> raise(Failure("invalid binary operation: should be
rejected in semant"))
              ) e1 e2 "tmp" builder in
              build_select condition (const_int i64_t 1) (const_int i64_t
0) "booleantoint" builder
          | ([StringType],_)                                      ->
              let cmpllvm = build_call compareString [|e1; e2|] "cmpstr"
builder in
              let condition = (match op with
                IsEqual  ->
                build_icmp Icmp.Eq
              | LessThan  ->
                build_icmp Icmp.Slt
              | LessEqual ->
                build_icmp Icmp.Sle
              | GreaterThan  ->
                build_icmp Icmp.Sgt
              | GreaterEqual  ->
                build_icmp Icmp.Sge
              | _         ->  raise(Failure("binary operation is invalid
```

```
and should be rejected in semant"))
              ) cmpllvm (const_int i64_t 0) "cmpstr_bool" builder in
              build_select condition (const_int i64_t 1) (const_int i64_t
0) "booleantoint" builder
          | _ -> raise(Failure("invalid binary operation: should be
rejected in semant"))
          )
        | SBinaryOp(ex1,op,ex2) when (List.hd e_typl) = StringType
->
          let e1 = expr builder ex1
          and e2 = expr builder ex2 in
          (match op with
            Add ->  let call_ret = build_call addString [|e1; e2|] "addstr"
builder in
                    (*  GC injection for AddString  *)
                    gc_inject call_ret builder
          | _   ->  raise(Failure("codegen error: semant should reject any
operation between string except add"))
          )
        | SUnaryOp(op,((ex1_typl,_) as ex1))
->
          let e_ = expr builder ex1 in
          (match op with
            LogicalNot                                  ->  build_not
          | Negative when ex1_typl = [IntegerType]   ->  build_neg
          | Negative when ex1_typl = [FloatType]     ->  build_fneg
          | _ -> raise (Failure("unary operation is invalid and should be
rejected in semant"))
          ) e_ "tmp" builder
        | SAssignOp(var,ex1)  ->
          let e_ = expr builder ex1 in
          (match var with
              ([IntegerType],SSliceIndex(e1,e2))  ->
              (match e1 with
              | (_,SNamedVariable(s)) -> build_call setSliceIndexInt
[|(expr builder e1);(expr builder e2)
                                          ;e_|] "setsliceidxn" builder
              | _ -> raise(Failure("SAssignOp error: should be rejected in
semant"))
              )
            | ([FloatType],SSliceIndex(e1,e2))    ->
              (match e1 with
              | (_,SNamedVariable(s)) -> build_call setSliceIndexDouble
```

```
[|(expr builder e1);(expr builder e2)
                                          ;e_|] "setsliceidxf" builder
              | _ -> raise(Failure("SAssignOp error: should be rejected in
semant"))
              )
            | ([FutureType],SSliceIndex(e1,e2))   ->
              (match e1 with
              | (_,SNamedVariable(s)) -> build_call setSliceIndexFuture
[|(expr builder e1);(expr builder e2);
                                          e_|] "setsliceidxF" builder
              | _ -> raise(Failure("SAssignOp error: should be rejected in
semant"))
              )
            | ([StringType],SSliceIndex(e1,e2))    ->
              (match e1 with
              | (_,SNamedVariable(s)) ->  build_call setSliceIndexString
[|(expr builder e1);(expr builder e2);e_|]
                                            "setsliceidxs" builder
              | _ -> raise(Failure("SAssignOp error: should be rejected in
semant"))
              )
            | (_,SNamedVariable(s)) ->
              let (typl, ex1_) = ex1 in let from_llvm = e_ in
              ( match List.hd typl with
                  StringType ->
                    let clonellvm = build_call cloneString [|from_llvm|]
"clonestr" builder in
                    ignore(build_store clonellvm (lookup s) builder);
                    (*  GC Injection for CloneString  *)
                    gc_inject clonellvm builder

                | FutureType ->
                  let _ = (match ex1_ with
                      SFunctionCall(fname,_) -> Hashtbl.add futures s fname
                      | SSliceIndex((_,SNamedVariable(slice_name)),_) ->
                          Hashtbl.replace futures s (Hashtbl.find futures
slice_name)
                    | _ -> raise(Failure("AssignOp error: left hand side "
^string_of_sexpr(ex1) ^" is invalid"))
                  )
                  in
                  (*  No GC here; no new object is actually created; it is
just a reference  *)
```

```
                ignore(build_store e_ (lookup s) builder); e_

              | SliceType(x) ->
                let clonellvm = build_call cloneSlice [|from_llvm|]
"cloneslice" builder in
                ignore(build_store clonellvm (lookup s) builder);
                let _ = match x with
                  FutureType->
                  let (typl, ex1_) = ex1 in
                  (match ex1_ with
                      SNamedVariable(slice_name) -> Hashtbl.replace
futures s (Hashtbl.find futures slice_name)
                    | SAppend(expl) ->
                      (
                        match List.hd expl with
                        (_,SNamedVariable(slice_name)) -> Hashtbl.replace
futures s (Hashtbl.find futures slice_name)
                        | _ -> ignore()
                      )
                    | _ -> ignore()
                  )
                  | _ ->ignore()
                in
                (* MERGE CONFLICT FIXME TODO: return e_ ??*)
                 (*  GC Injection for CloneSlice  *)
                ignore(gc_inject clonellvm builder); e_

              | _ ->
                ignore(build_store from_llvm (lookup s) builder); e_
            )
          | _ -> raise(Failure("SAssignOp error: should be rejected in
semant"))
        )
      | SLen (e) ->
          let (typl, _) = e in
          let e_ = expr builder e in
          (match typl with
          [SliceType(x)] -> build_call getSliceSize [|e_|] "slicelen"
builder
          | _ -> build_call lenString [| e_ |] "str_len" builder
          )
      | SAwait(s)                                                    ->
        let (await_llvm,fd) = func_of_future s in
```

```
          let future_arg = build_load (lookup s) s builder  in
          let result = "await_"^fd.sfname^"_result" in
          build_call await_llvm (Array.of_list [future_arg]) result builder

        | SRead(e) ->
            let e_ = expr builder e in
            let call_ret =
              build_call readFile [| e_ |] "read_file" builder in
             (*   GC Injection for Read  *)
              gc_inject call_ret builder
        | SFunctionCall("print",el)
->
            let exarr= Array.of_list (List.map (fun x -> expr builder x)
el) in
            build_call printf exarr "" builder
        | SFunctionCall("println",el)
->
            let exarr= Array.of_list (List.map (fun x -> expr builder x)
el) in
            build_call printfln exarr "" builder
        | SAppend(args)                                    ->
            let e1 = expr builder (List.hd args) in
            let e2 = expr builder (List.nth args 1) in
            let (ret_typl1,e1') = (List.hd args) in
            let (ret_typl2,e2') = (List.nth args 1) in
            (* e.g. s = append(s, c)*)
            let _ = match ret_typl2 with
            [FutureType] ->
                (*let SNamedVariable(slice_name) = e1' in*)
                let slice_name = match e1' with
                  SNamedVariable(x) -> x
                | _ -> raise(Failure("SAppend error: should be rejected in
semant"))
                in
                let check_eq a b = if a=b then ignore() else raise (Failure
( "Codegen Err: only support
                    add future objects with same async function in one
slice")) in
                ( match e2' with
                | SNamedVariable(future_object)->
                    (* let SNamedVariable(future_object) = e2'*)
                    let future_func =
                        if Hashtbl.mem futures future_object then
```

```
Hashtbl.find futures future_object
                        else raise (Failure ( "Codegen Err: undeclared
future object " ^ future_object)) in
                    ignore(if Hashtbl.mem futures slice_name then check_eq
(Hashtbl.find futures slice_name)
                    future_func else ignore(Hashtbl.add futures slice_name
future_func))
                | SFunctionCall(future_func,_)->
                    ignore(if Hashtbl.mem futures slice_name then check_eq
(Hashtbl.find futures slice_name)
                    future_func else ignore(Hashtbl.add futures slice_name
future_func))
                | _ -> raise(Failure("SAppend error: should be rejected in
semant"))
                )
                (* print_string(string_of_bool(Hashtbl.mem futures
slice_name)) *)
            | _ -> ignore()
            in
            let call_ret =
            build_call sliceAppend [|e1;e2|] "slice_append" builder in
            (*   GC Injection for SliceAppend  *)
            gc_inject call_ret builder
        | SFunctionCall(f_name,args) ->
          let (fdef,fd) = find_func f_name in
          let llargs = List.map (expr builder) args in
          let result = (match fd.styp with [VoidType] -> "" | _ -> f_name ^
"_result") in
          ( match fd.sann with
            FuncNormal ->
              let build_llvm = build_call fdef (Array.of_list llargs)
result builder
              in ( match fd.styp with
                  [IntegerType] | [FloatType] | [BoolType] ->
                    build_extractvalue build_llvm 0 "extracted_value"
builder
                | [StringType] | [SliceType(_)] | [FutureType] ->
                    let extracted_val = build_extractvalue build_llvm 0
"extracted_value" builder in
                    (*  GC Injection for function return  *)
                    gc_inject extracted_val builder
                | _ -> build_llvm
              )
```

```
            | _ ->
              let return_types = Array.of_list (List.map (fun t ->
ltype_of_typ t) fd.styp) in
              let stype = struct_type context return_types in
              let await_ftyp_t = function_type stype [|(pointer_type
i8_t)|] in
              let await_llvm = declare_function
("digo_linker_await_func_"^f_name) await_ftyp_t the_module in
              Hashtbl.add function_decls ("digo_linker_await_func_"^f_name)
(await_llvm,fd);

              let argument_types = Array.of_list (List.map (fun (t,_) ->
ltype_of_typ t) fd.sformals) in
              let new_ftyp_t = function_type (pointer_type i8_t)
argument_types in
              let new_fdef = declare_function
("digo_linker_async_call_func_"^f_name) new_ftyp_t the_module in
              let call_ret =
                build_call new_fdef (Array.of_list llargs) result builder
             (*   GC Inject the future object returned by
digo_linker_async_call_func_  *)
              in gc_inject call_ret builder
          )

        | SInteger(ex)  ->  const_int i64_t ex
        | SFloat(ex) ->  const_float float_t ex
        | SString(ex)  ->
        (* build_global_stringptr ex "str" builder *)
          let current_ptr = build_global_stringptr ex "createstr_ptr"
builder in
          let call_ret =
            build_call createString [|current_ptr|] "createstr" builder in
            (*  GC Injection for CreateString  *)
          gc_inject call_ret builder
        | SBool(ex)                              ->  const_int i1_t (if
ex then 1 else 0)
        | SNamedVariable(n)                      ->  build_load (lookup
n) n builder
        | SSliceLiteral(built_typ,len,e1_l)        ->
        let tp = get_type_in_slicetype built_typ in
        let arg_sn = get_slice_argument_number tp in
        let empty_slice = build_call createSlice [|arg_sn|] "createslice"
builder in
```

```
        (
          (*  GC Injection for CreateSlice  *)
          ignore(gc_inject empty_slice builder);
        match tp  with
          StringType  ->
            let append_string slc e1 =
            let e = expr builder e1 in
            (*  GC Injection for sliceAppend  *)
            let eval_result =
                build_call sliceAppend [|slc;e|] "initslices" builder
            in
                gc_inject eval_result builder
            (*   the SliceAppend will IncRef the object being appended.
Slice will DecRef objects when appropriate.   *)
            in
            List.fold_left append_string empty_slice e1_l
        | IntegerType ->
            let append_integer slc e1 =
            let e = expr builder e1 in
            let eval_result =
                build_call sliceAppend [|slc;e|] "initslicen" builder
            in
                gc_inject eval_result builder
            in
            List.fold_left append_integer empty_slice e1_l
        | FloatType   ->
            let append_float slc e1 =
            let e = expr builder e1 in
            let eval_result =
                build_call sliceAppend [|slc;e|] "initslicef" builder
            in
                gc_inject eval_result builder
            in
            List.fold_left append_float empty_slice e1_l
        | FutureType  ->
            let append_future slc e1 =
            let e = expr builder e1 in
            let eval_result =
                build_call sliceAppend [|slc;e|] "initslicenF" builder
            in
                gc_inject eval_result builder
            in
            List.fold_left append_future empty_slice e1_l
```

```
        | _           -> raise(Failure("invalide slice type"))
        )
      | SSliceIndex(ex1,ex2) ->
        let rt = List.hd e_typl in
        let ex1' = expr builder ex1 in
        let ex2' = expr builder ex2 in
        (match rt with
          StringType  -> let call_ret = build_call getSliceIndexString
[|ex1';ex2'|]
                           "findsliceindexs" builder in
                         gc_inject call_ret builder
        | IntegerType ->  build_call getSliceIndexInt [|ex1';ex2'|]
"findsliceindexn" builder
        | FloatType ->  build_call getSliceIndexDouble [|ex1';ex2'|]
"findsliceindexf" builder
        | FutureType  -> let call_ret = build_call getSliceIndexFuture
[|ex1';ex2'|]
                           "findsliceindexF" builder in
                         gc_inject call_ret builder
        | _         ->  raise(Failure("sliceindex error: should be rejected
in semant"))
        )
      | SSliceSlice(ex1,ex2,ex3)  ->
        let ex1' = expr builder ex1 in
        let start_idx = match ex2 with
        | ([IntegerType],_)  -> expr builder ex2
        | ([VoidType],_)     -> const_int i64_t 0
        | _ -> raise(Failure("sliceslice error: should be rejected in
semant"))
        in
        let end_idx = match ex3 with
        | ([IntegerType],_)  -> expr builder ex3
        | ([VoidType],_)     ->
           let slen = build_call getSliceSize [|ex1'|] "totallen" builder
in
           build_sub slen (const_int i64_t 1) "getlastindex" builder
        | _ -> raise(Failure("sliceslice error: should be rejected in
semant"))
        in let call_ret =
        build_call sliceSlice [|ex1';start_idx;end_idx|] "SliceSlice"
builder
         (*  GC Injection for SliceSlice  *)
        in gc_inject call_ret builder
```

```
      | _ -> raise(Failure("invalid expr: should be rejected in semant"))
    in

      let add_terminal builder instr  =
        match block_terminator (insertion_block builder) with
          Some _ -> ()
        | None -> ignore (instr builder) in

      let rec stmt builder = function
        SEmptyStatement  ->  builder
      | SBlock(sl)  ->
        List.fold_left stmt builder sl
      | SIfStatement(ex, s1, s2) ->
        let bool_val =
        (match ex with
          ([BoolType],SBinaryOp(e1,op,e2)) ->
           (match e1 with
             ([FloatType],_)   ->
            let rt = expr builder ex in
            build_fcmp Fcmp.Oeq rt (const_float float_t 1.0) "judgeif"
builder
           | ([IntegerType],_) ->
            let rt = expr builder ex in
            build_icmp Icmp.Eq rt (const_int i64_t 1) "judgeif" builder
           | ([BoolType],_)    ->
            let rt = expr builder ex in
            build_icmp Icmp.Eq rt (const_int i64_t 1) "judgeif" builder
           | ([StringType],_)  ->
            let rt = expr builder ex in
            build_icmp Icmp.Eq rt (const_int i64_t 1) "judgeif" builder
           | _ -> raise(Failure("if statement error: should be rejected in
semant"))
           )
        | ([BoolType],_) -> expr builder ex
        | _ -> raise(Failure("not boolean on if condition"))
        ) in
        let merge_bb = append_block context "merge" the_function in
        let build_br_merge = build_br merge_bb in

        let then_bb = append_block context "then" the_function in
        add_terminal (stmt (builder_at_end context then_bb) s1)
build_br_merge;
```

```
        let else_bb = append_block context "else" the_function in
        add_terminal (stmt (builder_at_end context else_bb) s2)
build_br_merge;

        ignore(build_cond_br bool_val then_bb else_bb builder);
        builder_at_end context merge_bb

      | SForStatement(e1, ex, e2, stl) ->
        ignore(stmt builder (SExpr e1));

        let whole_body = SBlock[stl;SExpr(e2)] in
        let pred_bb = append_block context "for" the_function in
        ignore(build_br pred_bb builder);

        let body_bb = append_block context "for_body" the_function in
        add_terminal (stmt (builder_at_end context body_bb) whole_body)
(build_br pred_bb);

        let pred_builder = builder_at_end context pred_bb in
        let bool_val =
        (match ex with
        | ([BoolType],SBinaryOp(e1,op,e2)) ->
           (match e1 with
             ([FloatType],_)   ->
            let rt = expr pred_builder ex in
            build_fcmp Fcmp.Oeq rt (const_float float_t 1.0) "judgeif"
pred_builder
           | ([IntegerType],_) ->
            let rt = expr pred_builder ex in
            build_icmp Icmp.Eq rt (const_int i64_t 1) "judgeif"
pred_builder
           | ([BoolType],_)    ->
            let rt = expr pred_builder ex in
            build_icmp Icmp.Eq rt (const_int i64_t 1) "judgeif"
pred_builder
           | ([StringType],_)  ->
            let rt = expr builder ex in
            build_icmp Icmp.Eq rt (const_int i64_t 1) "judgeif" builder
           | _ -> raise(Failure("for statement error: should be rejected in
semant"))
           )
        | ([BoolType],_) -> expr builder ex
        | _ -> raise(Failure("not boolean on for condition"))
```

```
        )
        in
        let merge_bb = append_block context "merge" the_function in
        ignore(build_cond_br bool_val body_bb merge_bb pred_builder);
        builder_at_end context merge_bb
      | SDeclare(nl,ty,el) ->
        (match ty with
          SliceType(x) ->
          let add_slice_decl n =
          let arg_sn = get_slice_argument_number x in
          let empty_slice = build_call createSlice [|arg_sn|] "createslice"
builder in
          (*  DONOT DELETE THIS GC_INJECT  *)
          ignore(gc_inject empty_slice builder);
          let alloca = build_alloca (ltype_of_typ ty) n builder in
          let _ = build_store empty_slice alloca builder in
          add_var_decl n alloca
          in List.iter add_slice_decl nl
        | _ -> let add_decl n = ignore(add_var_decl n (build_alloca
(ltype_of_typ ty) n builder))
        in List.iter add_decl nl
        );
        let ck = match el with
          [([VoidType],_)] -> builder
          | _ ->
            let build_decll n e = expr builder
([ty],SAssignOp(([ty],SNamedVariable(n)),e))
            in ignore(List.map2 build_decll nl el);
            builder
        in ck

      | SShortDecl(nl,el) ->
        (match el with
        [([FutureType],SFunctionCall(_,_))] ->
          let add_decl n =
            ignore(add_var_decl n (build_alloca (ltype_of_typ FutureType) n
builder))
            in List.iter add_decl nl
        | [(etl,SFunctionCall(_,_))] | [(etl,SAwait(_))] ->
          let add_decl n et =
          ignore(add_var_decl n (build_alloca (ltype_of_typ et) n builder))
          in List.iter2 add_decl nl etl
        | _ ->
```

```
          let add_decl n e =
          let (et, _) = e in
          ignore(add_var_decl n (build_alloca (ltype_of_typ (List.hd et)) n
builder))
          in List.iter2 add_decl nl el);
        let check_func_call =
          let build_decll n e =
            let (et, _) = e in
            expr builder (et,SAssignOp((et,SNamedVariable(n)),e))
          in
          match (List.hd el) with
          ([FutureType],SFunctionCall(_,_))  ->  ignore(List.map2
build_decll nl el); builder
          | ([FutureType], SSliceIndex((_,SNamedVariable(slice_name)),_))
->
            let add_futures n ret =
              ( match ret with
                  (_, SSliceIndex((_,SNamedVariable(slice_name)),_)) ->
Hashtbl.replace futures
                       n (Hashtbl.find futures slice_name)
                  | _ -> ignore()
              ) in
            List.iter2 add_futures nl el;
            ignore(List.map2 build_decll nl el);
            builder
          | (_,SAwait(_)) ->
            let e_ = expr builder (List.hd el) in
                let rec apply_extractvaluef current_idx = function
                []          ->  ()
                | a::tl       ->
                (*   GC Injection of return value here   *)
                let extracted_llvalue = build_extractvalue e_ current_idx
"extracted_value" builder
                in
                (* GC Inject if the type is i8*  *)
                if type_of extracted_llvalue == pointer_type i8_t then
                  ignore(gc_inject extracted_llvalue builder);
                ignore(build_store extracted_llvalue (lookup a) builder);
                apply_extractvaluef (current_idx+1) tl
              in  ignore(apply_extractvaluef 0 nl);
              builder
          | (_,SFunctionCall(_,_)) ->
            let (ret_typ, _) = (List.hd el) in
```

```
            (match List.length ret_typ with
            1 -> ignore(List.map2 build_decll nl el); builder
            | _ ->
              let e_ = expr builder (List.hd el) in
              let rec apply_extractvaluef current_idx = function
                []          ->  ()
                | a::tl       ->
                (*   GC Injection of return value here   *)
                let extracted_llvalue = build_extractvalue e_ current_idx
"extracted_value" builder
                in
                (* GC Inject if the type is i8*  *)
                if type_of extracted_llvalue == pointer_type i8_t then
                  ignore(gc_inject extracted_llvalue builder);
                ignore(build_store extracted_llvalue (lookup a) builder);
                apply_extractvaluef (current_idx+1) tl

              in  ignore(apply_extractvaluef 0 nl);
              builder
            )
          | _ -> ignore(List.map2 build_decll nl el); builder
        in check_func_call
      | SBreak
->  builder
      | SContinue
->  builder
      | SReturn(el)
->
        let el_mapper e =
          let expr_llvalue = expr builder e in
           (*(expr_llvalue, !expr_is_lvalue_obj)*)
           if type_of expr_llvalue == pointer_type i8_t then
              (expr_llvalue, 1)
           else
              (expr_llvalue, 0)
        in
        let el_unmmaper (e2, _) = e2 in
        let agg_with_type = List.map el_mapper el in
        let agg = Array.of_list (List.map el_unmmaper agg_with_type) in
        ignore(gc_gen_notrace_from_retval builder gc_trace_map_obj
agg_with_type);
        ignore(build_call gc_release_all [|gc_trace_map_obj|] "" builder);
        ignore(build_aggregate_ret agg builder); builder
```

```
      | SExpr(ex) ->  ignore(expr builder ex); builder

      in

        let builder = stmt builder (SBlock fdecl.sbody) in
        let agg_ = [|const_int i64_t 0|] in
        match fdecl.styp with
        [VoidType] -> ignore(build_call gc_release_all [|gc_trace_map_obj|]
"" builder);
                      ignore(build_ret_void builder)
        | _ ->
              add_terminal builder (build_aggregate_ret agg_)

      in

        List.iter build_function_body functions;
        the_module
```

```
Usage() {
    echo "Usage: testall.sh [options] [.digo files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

MAKE_DIR='../'
# Generate depenencies and Digo Compiler
BuildCompiler() {
    echo "------------------ !Compiler not found ------------------------"
    echo "------------------ Generating Compiler ------------------------"
    echo "Please wait..... It may take 1-5 minutes to generate compiler"

    if ! make -C $MAKE_DIR clean &>/dev/null; then
      echo "clean failed "
      exit 1
    fi
```

```
    if ! make -C $MAKE_DIR generate-dependency &>/dev/null;
    then
      echo "build dependency failed"
      exit 1
    fi

    echo "Library compilation succeeds, now generating OCaml compiler..."
    if ! make -C $MAKE_DIR generate-digo-compiler &>/dev/null; then
      echo "build compiler failed"
      exit 1
    fi

    echo "------------------ Compiler Generated ------------------------"
}

global_test_error=0
global_test_error_any=0

global_log="testall.log"
global_llvm_ir_output_log="testall_ir.log"
rm -f $global_log
rm -f $global_llvm_ir_output_log

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile. Differences, if any, written to
difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 &>>"$global_log"
    diff -b "$1" "$2" > "$3" 2>&1 || {
        global_test_error=1
        echo "FAILED $1 differs from $2" >> "$global_log"
    }
}

WORKER_COUNT=0
GC_DEBUG=0
ENABLE_MASTER=0
MASTER_ADDR=()
WORKER_ADDR=()

LoadDefaultConfig() {
    WORKER_COUNT=0
```

```
    GC_DEBUG=0
    ENABLE_MASTER=0
    MASTER_ADDR=()
    WORKER_ADDR=()
}

# RunTest <digofile>
#    Compile, run, and check the output.
#    If the compilation fails, it will diff
#       the error reported by the compiler and the
$source_code.fail.expected
#    If the compilation succeeds, it will diff
#       the result produced by the executable and the
$source_code.pass.expected
RunTest() {
    global_test_error=0
    test_name=`echo $1 | sed 's/.*\\///
                             s/.digo//'`
    test_src="$1"

    dir_name=`dirname $test_src`

    echo -n "$test_name..."

    rm -f "../executable"

    ../digo-compiler/digo.native
"$test_src" > ../tmp.compiled.nometadata.ll 2>"$test_name.build.output"
    errorlevel=$?

    echo "" >> "$global_llvm_ir_output_log"
    echo "# -------------------- Testing $test_name" >>
"$global_llvm_ir_output_log"
    cat ../tmp.compiled.nometadata.ll >> $global_llvm_ir_output_log

    if [ $errorlevel -eq 0 ] ; then
        make -C $MAKE_DIR build-link-pass digo="$test_src" out=executable
&>"$test_name.linker.output"
        errorlevel=$?
    fi

    echo "" >> "$global_log"
    echo "# -------------------- Testing $test_name --------------------
```

```
" >> "$global_log"

    # load group config
    if [ -f "$dir_name/config.txt" ] ; then
        . "$dir_name/config.txt"
    fi

    # config override
    if [ -f "$test_src.cfg" ] ; then
        . "$test_src.cfg"
    fi

    echo "Config: WORKER_COUNT=$WORKER_COUNT; GC_DEBUG=$GC_DEBUG;
ENABLE_MASTER=$ENABLE_MASTER; " >> "$global_log"
    echo "        MASTER_ADDR=${MASTER_ADDR[*]};
WORKER_ADDR=${WORKER_ADDR[*]}" >> "$global_log"

    diff_output_file="$test_name.diff.output"

    if [ ! -f "../executable" ]; then
        errorlevel=1
    fi

    if [ $errorlevel -eq 0 ] ; then
        # success expected, so try to run the executable
        exec_output="$test_name.exec.output"
        exec_debug_output="$test_name.debug.output"
        expected_file="$test_src.pass.expected"

        if [ $ENABLE_MASTER -eq 0 ] ; then
            eval '../executable --no-master > "$exec_output"
2>"$exec_debug_output" &'
            master_pid=$!
        else
            # echo ../executable --master $MASTER_ADDR
            eval '../executable --master $MASTER_ADDR > "$exec_output"
2>"$exec_debug_output" &'
            master_pid=$!
        fi

        worker_pid=()
        if [ $WORKER_COUNT -gt 0 ] ; then
            for (( i = 0 ; i < $WORKER_COUNT ; i++ ))
```

```
            do
                command="../executable --worker $MASTER_ADDR
${WORKER_ADDR[$i]} &> \"$test_name.worker$i.output\" &"
                echo "Running worker $i: $command" >> "$global_log"
                pid=`(eval $command ; echo $!) 2>/dev/null`
                worker_pid[$i]=$pid
            done
        fi

        wait $master_pid

        kill -9 $master_pid &> /dev/null

        for pid in ${worker_pid[*]}; do
            kill -9 $pid &> /dev/null
        done

        echo " ## Executable output: " >> "$global_log"
        cat "$exec_output" >> "$global_log"

        echo " ## Executable debug output: " >> "$global_log"
        cat "$exec_debug_output" >> "$global_log"

        if [ $WORKER_COUNT -gt 0 ] ; then
            for (( i = 0 ; i < $WORKER_COUNT ; i++ ))
            do
                echo "" >> "$global_log"
                echo " ## Worker $i stdout&stderr output: " >>
"$global_log"
                cat "$test_name.worker$i.output" >> "$global_log"
                worker_pid[$i]=$!
            done
        fi

        if [ ! -f $expected_file ]; then
            echo " ## Compilation passed, but we cannot find
$expected_file" >> "$global_log"
            global_test_error=1
        else
            echo " ## Diff output: " >> "$global_log"
            Compare "$exec_output" "$expected_file" "$diff_output_file"

            cat "$diff_output_file" >> "$global_log"
```

```
        fi
    else
        # fail expected, so try to diff the fail file
        expected_file="$test_src.fail.expected"
        build_output="$test_name.build.output"

        echo " ## Build output: " >> "$global_log"
        cat "$build_output" >> "$global_log"

        if [ ! -f $expected_file ]; then
            echo " ## Compilation failed, but we cannot find
$expected_file" >> "$global_log"
            global_test_error=1
        else
            Compare "$build_output" "$expected_file" "$diff_output_file"

            echo " ## Diff output: " >> "$global_log"
            cat "$diff_output_file" >> "$global_log"
        fi
    fi

    if [ $global_test_error -eq 1 ] ; then
        global_test_error_any=1
    fi

    if [ $global_test_error -eq 1 ] ; then
        echo "##### Test $test_name: Failed" >> "$global_log"
        echo "FAIL"
    else
        echo "##### Test $test_name: Passed" >> "$global_log"
        echo "OK"
    fi

    echo "" >> "$global_log"

}

RunTestFiles() {
    test_files=$@
    for filename in $test_files; do
        RunTest "$(pwd)/$filename"
    done
}
```

```
# RunTestGroup <test_group_dir>
RunTestGroup() {
    echo "# --- Test Group: $1"

    echo "# ------------------------- Test Group: $1 --------------------
-" >> "$global_log"

    group_dir=$1
    LoadDefaultConfig
    # load config

    . "$group_dir/config.txt"

    RunTestFiles "$group_dir/*.digo"

    echo "" >> "$global_log"
    echo "" >> "$global_log"
}

Clean() {
    # make -C $MAKE_DIR clean &>/dev/null
    rm -f *.build.output
    rm -f *.exec.output
    rm -f *.diff.output
    rm -f *.linker.output
    rm -f *.worker*.output
    rm -f *.debug.output
}

# Set time limit for all operations
ulimit -t 30
keep=0

while getopts kdpsh c; do
    case $c in
      k) # Keep intermediate files
          keep=1
          ;;
      h) # Help
          Usage
          ;;
    esac
```

```
done

shift `expr $OPTIND - 1`


if [ ! -f ../digo-compiler/digo.native ]; then
    BuildCompiler
fi


if [ $# -ge 1 ]
then
    RunTestFiles $@
else
    RunTestGroup "Basic"
    RunTestGroup "Async"
    RunTestGroup "Syntax"
    RunTestGroup "Semantic"
    RunTestGroup "ControlFlow"
    RunTestGroup "GC"
    RunTestGroup "Remote"
    RunTestGroup "Utils"
fi


if [ $keep -eq 0 ] ; then
    echo "Cleaning..."
    Clean
fi

echo "---------"

if [ $global_test_error_any -eq 1 ] ; then
    echo "Some tests failed, see $global_log for details"
else
    echo "All tests passed"
fi

exit $global_test_error_any
```

9.2.9 Makefile

```
all : generate-compiler

semant : parser.cmo scanner.cmo semant.cmo
	ocamlc -o semant $^

printer : parser.cmo scanner.cmo printer.cmo
	ocamlc -o printer $^

metadata-generator : parser.cmo scanner.cmo metadata_gen.cmo
	ocamlc -o metadata_gen $^

%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
	ocamlc -c $<

scanner.ml : scanner.mll
	ocamllex $^

parser.ml parser.mli : parser.mly
	ocamlyacc $^

# Depedencies from ocamldep
semant.cmo : scanner.cmo parser.cmi ast.cmo
semant.cmx : scanner.cmx parser.cmx ast.cmo

printer.cmo : scanner.cmo parser.cmi ast.cmo
printer.cmx : scanner.cmx parser.cmx ast.cmo

parser.cmo : ast.cmo parser.cmi
parser.cmx : ast.cmo parser.cmi

scanner.cmo : parser.cmi
scanner.cmx : parser.cmx

.PHONY : remake

remake: clean printer

.PHONY : clean
```

```
clean :
	rm -rf *.cmi *.cmo parser.ml parser.mli scanner.ml printer.out
printer semant.out semant
	rm -rf _build
	rm -rf digo.native metadata_gen

generate-compiler: clean
	ocamlbuild -use-ocamlfind digo.native -package llvm,llvm.analysis
```

9.3 async-remote-lib

Async-remote-lib is mostly finished by Yufan.

9.3.1 async.cpp

```
#include "async.h"
#include "master_worker.h"

#include <functional>
#include <sstream>

using namespace std;

shared_ptr<Async> Async::CreateLocal(string digo_func_name, bytes
parameters) {
    shared_ptr<Async> async = make_shared<Async>();
    if (ASYNC_DEBUG) {
        stringstream f;
        f << "Local job created: " << async.get() << " func_name: " <<
          digo_func_name << " arg len: " << parameters.length << endl;
        cerr << f.str();
    }
    async->result_set_ = false;
    async->std_future_obj_ = std::async([=] {
        return CallFunctionByName(digo_func_name, parameters);
    });
    return async;
}

shared_ptr<Async> Async::CreateRemote(string digo_func_name, bytes
parameters) {
```

```
    shared_ptr<Async> async = make_shared<Async>();
    if (ASYNC_DEBUG) {
        stringstream f;
        f << "Remote job created: " << async.get() << " func_name: " <<
          digo_func_name << " arg len: " << parameters.length << endl;
        cerr << f.str();
    }
    async->result_set_ = false;
    async->std_future_obj_ = std::async([=] {
        auto ret = Master::GetInst()->CallRemoteFunctionByName(
                digo_func_name, to_vector(parameters));
        auto bs = to_bytes(ret);
        return bs;
    });
    return async;
}

bytes Async::Await() {
    std::lock_guard l(this->lock_);
    if (ASYNC_DEBUG) {
        stringstream f;
        f << "Awaiting on job " << this << endl;
        cerr << f.str();
    }
    if (result_set_) {
        return result_;
    }
    result_set_ = true;
    result_ = this->std_future_obj_.get();
    return result_;
}
```

9.3.2 async.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_ASYNC_H_
#define ASYNC_REMOTE_LIB_SRC_ASYNC_H_

#include <thread>
#include <memory>
#include <future>
```

```
#include <mutex>

#include "common.h"
#include "linker_common.h"

using std::shared_ptr;
using std::future;

#define ENABLE_ASYNC_DEBUG

#ifdef ENABLE_ASYNC_DEBUG
const bool ASYNC_DEBUG = true;
#else
const bool ASYNC_DEBUG = false;
#endif

class Async : public noncopyable {
 public:
  static shared_ptr<Async> CreateLocal(string digo_func_name, bytes
parameters);
  static shared_ptr<Async> CreateRemote(string digo_func_name, bytes
parameters);
  bytes Await();
 private:
  std::future<bytes> std_future_obj_;
  bool result_set_;
  bytes result_;
  std::mutex lock_;
};

#endif //ASYNC_REMOTE_LIB_SRC_ASYNC_H_
```

9.3.3 master_worker.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_MASTER_WORKER_H_
#define ASYNC_REMOTE_LIB_SRC_MASTER_WORKER_H_

#include <set>
#include <atomic>
```

```
#include "common.h"
#include "network.h"

vector<byte> to_vector(const bytes &bs);

bytes to_bytes(const vector<byte> &data);

class Master : public noncopyable {
 private:
  static shared_ptr<Master> master_;

 public:
  static shared_ptr<Master> GetInst();

  void Listen(string server_addr);

  void StopListen();

  void WaitForReady();

  vector<byte> CallRemoteFunctionByName(const string &digo_func_name, const
vector<byte> &parameters);

  void AddWorker(const string &worker_addr);

 private:
  shared_ptr<Server> srv;

  std::set<string> worker_pool;
};

class Worker : public noncopyable {
 private:
  static shared_ptr<Worker> worker_;

 public:
  static shared_ptr<Worker> GetInst();

  void Start(const string &server_addr, const string &client_addr);

  void Stop();

 private:
```

```
  shared_ptr<Server> srv;
};

#endif //ASYNC_REMOTE_LIB_SRC_MASTER_WORKER_H_
```

9.3.4 master_worker.cpp

```
#include <iostream>
#include <unistd.h>
#include <cstdio>
#include <cstdlib>
#include <cstring>

#include "common.h"
#include "network.h"
#include "master_worker.h"
#include "linker_common.h"

shared_ptr<Master> Master::master_ = nullptr;
shared_ptr<Worker> Worker::worker_ = nullptr;

vector<byte> to_vector(const bytes &bs) {
  return vector<byte>(bs.content.get(), bs.content.get()+bs.length);
}

bytes to_bytes(const vector<byte> &data) {
  bytes bs;
  byte *content = new byte[data.size()];
  memcpy(content, data.data(), data.size());
  bs.content = shared_ptr<byte>(content);
  bs.length = data.size();
  return bs;
}

map<string, Handler> master_handlers = {
    {"join", [](const vector<byte> &data) {
      auto master = Master::GetInst();
      master->AddWorker(string(data.begin(), data.end()));
      string ret = "success\n";
```

```
      return vector<byte>(ret.begin(), ret.end());
    }}
};

shared_ptr<Master> Master::GetInst() {
  if (master_ == nullptr) {
    master_ = std::make_shared<Master>();
  }
  return master_;
}

void Master::WaitForReady() {
  while (!this->srv || !this->srv->IsListening())
    std::this_thread::yield();
}

void Master::StopListen() {
  this->srv->Stop();
  this->srv = nullptr;
  this->worker_pool.clear();
}

void Worker::Stop() {
  this->srv->Stop();
  this->srv = nullptr;
}

void Master::Listen(const string server_addr) {
  if (this->srv) {
    this->StopListen();
  }
  this->srv = Server::Create(server_addr);
  this->srv->SetHandlers(master_handlers);
  this->srv->Start();
}

void Master::AddWorker(const string &worker_addr) {
  this->worker_pool.insert(worker_addr);
}

vector<byte> Master::CallRemoteFunctionByName(const string &digo_func_name,
                                              const vector<byte> &parameters) {
  do {
```

```
    // busy waiting when no workers
    while (!this->worker_pool.size()) {
        sleep(1);
    }

    int idx = rand() % this->worker_pool.size();
    auto it = this->worker_pool.begin();
    advance(it, idx);

    shared_ptr<Client> cli = Client::Create();
    vector<byte> resp;
    vector<byte> req(digo_func_name.begin(), digo_func_name.end());
    req.push_back(':');
    req.insert(req.end(), parameters.begin(), parameters.end());


    if (cli->Call(*it, "call", req, resp) != 0) {
      cerr << "call worker fail" << endl;
      sleep(3);
      continue;
    }
    return resp;
  } while (true);
}

shared_ptr<Worker> Worker::GetInst() {
  if (worker_ == nullptr) {
    worker_ = std::make_shared<Worker>();
  }
  return worker_;
}

map<string, Handler> worker_handlers = {
    {"call", [](const vector<byte> &data) {
      auto worker = Worker::GetInst();

      auto p = string(data.begin(), data.end()).find(':');
      if (p == -1) {
        string ret = "error: request format invalid\n";
        return vector<byte>(ret.begin(), ret.end());
      }

      string digo_func_name, params_str;
```

```
      digo_func_name = string(data.begin(), data.begin()+p);
      bytes params = to_bytes(vector<byte>(data.begin()+p+1, data.end()));
      bytes result = CallFunctionByName(digo_func_name, params);
      return to_vector(result);
    }}
};

void Worker::Start(const string &server_addr, const string &client_addr) {
  if (this->srv) {
    this->srv->Stop();
    this->srv = nullptr;
  }
  this->srv = Server::Create(client_addr);
  this->srv->SetHandlers(worker_handlers);
  auto f = std::async([&] { this->srv->Start(); });

  auto cli = Client::Create();

  vector<byte> resp;
  if (cli->Call(server_addr, "join", vector<byte>(client_addr.begin(),
      client_addr.end()), resp) != 0) {
    cerr << "join master fail" << endl;
    exit(EXIT_FAILURE);
  }

  if (string(resp.begin(), resp.end()) != "success\n") {
    cerr << "join master fail. reason: " << string(resp.begin(),
resp.end()) << endl;
    exit(EXIT_FAILURE);
  }

  f.wait();
}
```

9.3.5 network.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_NETWORK_H_
#define ASYNC_REMOTE_LIB_SRC_NETWORK_H_
```

```
#include <sys/socket.h>
#include <vector>
#include <functional>
#include <map>
#include <string>
#include <ctime>
#include <future>
#include <atomic>

#include "common.h"


#define DELIM "\r\n\r\n"
#define HEADER_LENGTH_SIZE 10

typedef std::function<vector<byte> (const vector<byte> &)> Handler;

class Client : public noncopyable {
 public:
  static shared_ptr<Client> Create();
  int Call(const string &server_addr,
           const string &rpc_name, const vector<byte> &data, vector<byte>
&resp);

 private:
  int socket_;
};

class Server : public noncopyable {
 public:
  static shared_ptr<Server> Create(const string &server_addr);

  void SetHandlers(const std::map<string, Handler> &);
  void Stop();
  void Start();

  ~Server() {this->Stop();}

  bool IsListening();

 private:
  void HandleConn(int fd, const string &client_addr);
```

```
  std::map<string, Handler> handlers_;
  int socket_;
  std::atomic<bool> listening_;
};

#endif //ASYNC_REMOTE_LIB_SRC_NETWORK_H_
```

9.3.6 network.cpp

```
#include <iostream>
#include <sys/socket.h>
#include <cstdlib>
#include <cstdio>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>
#include <thread>
#include <string>
#include <cstring>
#include <cmath>
#include <vector>

#include "network.h"


vector<byte> PackData(const string &rpc_name, const vector<byte> &data) {
  string prefix = rpc_name + DELIM;
  string p_length = to_string(prefix.length() + data.size());
  p_length = string(10 - p_length.length(), '0') + p_length;

  vector<byte> ret;
  ret.insert(ret.end(), p_length.begin(), p_length.end());
  ret.insert(ret.end(), prefix.begin(), prefix.end());
  ret.insert(ret.end(), data.begin(), data.end());
  return ret;
}

int ReadUnpack(int fd, string &rpc_name, vector<byte> &data) {
  int n;
  char buf[1024];
  memset(buf, 0, sizeof(buf));
```

```
  data.clear();

  // read the first 10 bytes and convert it to the total length
  if ((n = read(fd, buf, HEADER_LENGTH_SIZE)) != HEADER_LENGTH_SIZE) {
    return -1;
  }

  auto packet_length = strtoul(buf, nullptr, 10);
  size_t read_left = packet_length;
  while (data.size() != packet_length && read_left > 0 &&
      (n = read(fd, buf, std::min(read_left, sizeof(buf)))) > 0) {
    data.insert(data.end(), buf, buf+n);
    read_left -= n;
  }

  if (n < 0) {
    return -1;
  }

  int pos = string(data.begin(), data.end()).find(DELIM);
  if (pos == -1) {
    cerr << "the request format is incorrect" << endl;
    return -1;
  }
  rpc_name = string(data.begin(), data.begin()+pos);
  data = vector<byte>(data.begin()+pos+sizeof(DELIM)-1, data.end());
  return 0;
}

int ParseAddr(const string &addr, string &hostname, unsigned short &port) {
  auto p = addr.find(':');

  if (p == -1) {
    return -1;
  }
  hostname = addr.substr(0, p);
  string port_str = addr.substr(p + 1);
  port = stoul(port_str);
  return 0;
}

shared_ptr<Client> Client::Create() {
  shared_ptr<Client> client = make_shared<Client>();
```

```
  client->socket_ = socket(AF_INET, SOCK_STREAM, 0);
  return client;
}

int Client::Call(const string &server_addr, const string &rpc_name,
                 const vector<byte> &data, vector<byte> &resp) {
  string hostname;
  unsigned short port;
  if (ParseAddr(server_addr, hostname, port) != 0) {
    // TODO: better error handling
    cerr << "parse server address failed" << endl;
    exit(EXIT_FAILURE);
  }

  struct sockaddr_in address{};
  memset(&address, 0, sizeof(address));
  address.sin_port = htons(port);
  address.sin_family = AF_INET;
  address.sin_addr.s_addr = inet_addr(hostname.c_str());

  int retry_cnt = 10;
  while (connect(this->socket_, (struct sockaddr *) &address,
sizeof(address)) < 0) {
    perror("connect");
    if (retry_cnt-- <= 0) {
      cerr << "retried 10 times but still failed, exit now." << endl;
      exit(EXIT_FAILURE);
    }
    cerr << "retry after 1 second..." << endl;
    sleep(1);

    this->socket_ = socket(AF_INET, SOCK_STREAM, 0);
    memset(&address, 0, sizeof(address));
    address.sin_port = htons(port);
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = inet_addr(hostname.c_str());
  }

  vector<byte> req = PackData(rpc_name, data);
  if (send(this->socket_, req.data(), req.size(), 0) < 0) {
    perror("send");
    exit(EXIT_FAILURE);
  }
```

```
  string whatever;
  if (ReadUnpack(this->socket_, whatever, resp) == -1) {
    close(this->socket_);
    return -1;
  }
  close(this->socket_);
  return 0;
}

shared_ptr<Server> Server::Create(const string &server_addr) {
  shared_ptr<Server> server = make_shared<Server>();

  string hostname;
  unsigned short port;
  if (ParseAddr(server_addr, hostname, port) != 0) {
    cerr << "parse address failed" << endl;
    exit(-1);
  }

  struct sockaddr_in address{};
  memset(&address, 0, sizeof(sockaddr_in));
  address.sin_family = AF_INET;
  address.sin_port = htons(port);
  address.sin_addr.s_addr = inet_addr(hostname.c_str());

  int listen_fd;

  if ((listen_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    perror("socket failed");
    exit(EXIT_FAILURE);
  }

  int opt = 1;
  if (setsockopt(listen_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt))) {
    perror("setsockopt");
    exit(EXIT_FAILURE);
  }

  if (bind(listen_fd, (struct sockaddr *) &address, sizeof(address)) < 0) {
    perror("bind");
    exit(EXIT_FAILURE);
  }
```

```
  server->socket_ = listen_fd;
  return server;
}

void Server::SetHandlers(const map<string, Handler> &hs) {
  this->handlers_ = hs;
}

void Server::Stop() {
  close(this->socket_);
  this->listening_ = false;
}

bool Server::IsListening() {
  return this->listening_;
}

void Server::Start() {
  if (listen(this->socket_, 128) < 0) {
    perror("server::start::listen");
    exit(EXIT_FAILURE);
  }

  this->listening_ = true;

  int accept_fd;
  socklen_t addr_len;
  struct sockaddr_in addr_in{};

  while (true) {
    if (fcntl(this->socket_, F_GETFD) == -1 && errno == EBADF) {
      return;
    }

    accept_fd = accept(this->socket_,
                       (struct sockaddr *) &addr_in, (socklen_t *)
&addr_len);
    if (accept_fd < 0) {
      perror("accept");
      continue;
    }
```

```
    // parse addr
    char client_addr_cstr[addr_len];
    inet_ntop(AF_INET, &(addr_in.sin_addr), client_addr_cstr, addr_len);

    string client_addr = client_addr_cstr;
    client_addr += ":" + to_string(ntohs(addr_in.sin_port));
    std::thread([=] { this->HandleConn(accept_fd,
client_addr); }).detach();
  }
}

void Server::HandleConn(int fd, const string &client_addr) {
  /* 4bytes: <packet_length> */

  string rpc_name;
  vector<byte> data;

  if (ReadUnpack(fd, rpc_name, data) != 0) {
    close(fd);
    return;
  }

  vector<byte> resp = this->handlers_[rpc_name](data);
  resp = PackData(rpc_name, resp);
  send(fd, resp.data(), resp.size(), 0);
  close(fd);

}
```

9.3.7 dslice.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_DSLICE_H_
#define ASYNC_REMOTE_LIB_SRC_DSLICE_H_

#include "common.h"
#include "dstring.h"
#include "../../digo-linker/src/common.h"

class TypeCellArray;
```

```
class DigoSlice : public DObject {
public:
    explicit DigoSlice(digo_type t);

    ~DigoSlice() override;

    std::tuple<vector<TypeCell> &, size_t &, size_t &> Data();

    [[nodiscard]] int64_t Size() const;

    [[nodiscard]] digo_type Type() const;

    [[nodiscard]] DigoSlice *Append(const TypeCell &tv) const;

    [[nodiscard]] DigoSlice *Slice(int64_t begin, int64_t end) const;

    [[nodiscard]] DigoSlice *Clone() const;

    [[nodiscard]] TypeCell &Index(int64_t idx) const;

    const char *name() override {
        return "Slice Object";
    }

private:
    shared_ptr<TypeCellArray> raw_data_;

    digo_type type = TYPE_UNDEFINED;
    size_t begin_, end_;

};

#include "../../digo-linker/src/gc.h"

extern "C" {

void *CreateSlice(int64_t type);
void *SliceSlice(void *obj, int64_t begin, int64_t end);
void *CloneSlice(void *obj);
void *SliceAppend(void *obj, ...);
int64_t GetSliceSize(void *obj);

void *GetSliceIndexString(void *obj, int64_t idx);
```

```
int64_t GetSliceIndexInt(void *obj, int64_t idx);
double GetSliceIndexDouble(void *obj, int64_t idx);
void *GetSliceIndexFuture(void *obj, int64_t idx);

void *SetSliceIndexString(void *obj, int64_t idx, void *val);
int64_t SetSliceIndexInt(void *obj, int64_t idx, int64_t val);
double SetSliceIndexDouble(void *obj, int64_t idx, double val);
void *SetSliceIndexFuture(void *obj, int64_t idx, void *val);
}


#endif //ASYNC_REMOTE_LIB_SRC_RESOURCE_H_
```

9.3.8 dslice.cpp

```
#include "dslice.h"
#include <iterator>
#include <cstdarg>

class TypeCellArray {
public:
    vector<TypeCell> arr_;
    digo_type type_;
    explicit TypeCellArray(digo_type type) {
        type_ = type;
    }

    vector<TypeCell> * get() {
        return &arr_;
    }

    virtual ~TypeCellArray() {
        switch (type_) {
            case TYPE_STR:
                for (auto & tc : arr_) {
                    ((DObject *)tc.str_obj)->DecRef();
                }
                break;
            case TYPE_FUTURE_OBJ:
                for (auto & tc : arr_) {
```

```
                    ((DObject *)tc.future_obj)->DecRef();
                }
                break;
            default:
                break;
        }
    }
};

DigoSlice::DigoSlice(digo_type t) {
    this->raw_data_ = make_shared<TypeCellArray>(t);
    this->type = t;
    this->begin_ = this->end_ = 0;
}

std::tuple<vector<TypeCell> &, size_t &, size_t &> DigoSlice::Data() {
    return std::make_tuple(std::ref(*this->raw_data_->get()),
                           std::ref(this->begin_), std::ref(this->end_));
}

int64_t DigoSlice::Size() const {
    return static_cast<int64_t>(this->end_ - this->begin_);
}

digo_type DigoSlice::Type() const {
    return this->type;
}

TypeCell &DigoSlice::Index(int64_t idx) const {
    return this->raw_data_->get()->at(this->begin_ + idx);
}

DigoSlice *DigoSlice::Slice(int64_t begin, int64_t end) const {
    auto sub_slice = new DigoSlice(this->type);
    sub_slice->begin_ = begin;
    sub_slice->end_ = end;
    sub_slice->raw_data_ = this->raw_data_;
    return sub_slice;
}

DigoSlice *DigoSlice::Append(const TypeCell &tv) const {
    /*  tv's ref count is already incremented */
    auto ret = new DigoSlice(this->type);
```

```
    ret->begin_ = this->begin_;
    ret->end_ = this->end_;
    ret->raw_data_ = this->raw_data_;

    if (ret->end_ < ret->raw_data_->get()->size()) {
        if (this->type == TYPE_STR) {

((DObject*)((*ret->raw_data_->get())[ret->end_].str_obj))->DecRef();
        } else if (this->type == TYPE_FUTURE_OBJ) {

((DObject*)((*ret->raw_data_->get())[ret->end_].future_obj))->DecRef();
        }
        (*ret->raw_data_->get())[ret->end_] = tv;
    } else {
        ret->raw_data_->get()->push_back(tv);
    }
    ret->end_ += 1;
    return ret;
}

DigoSlice *DigoSlice::Clone() const {
    auto ret = new DigoSlice(this->type);
    ret->begin_ = this->begin_;
    ret->end_ = this->end_;
    ret->raw_data_ = this->raw_data_;
    return ret;
}

DigoSlice::~DigoSlice() {

}

void *CreateSlice(int64_t type) {
    return new DigoSlice(digo_type(type));
}

void *SliceSlice(void *obj,
                 int64_t begin, int64_t end) {
    return ((DigoSlice *) obj)->Slice(begin, end);
}

void *CloneSlice(void *obj) {
    return ((DigoSlice *) obj)->Clone();
```

```
}

void *SliceAppend(void *vobj, ...) {
    va_list valist;
    va_start(valist, vobj);

    auto obj = (DigoSlice *) vobj;
    TypeCell tv;
    tv.type = obj->Type();

    switch (tv.type) {
        case TYPE_INT64:
            tv.num64 = va_arg(valist, int64_t);
            break;
        case TYPE_STR:
            tv.str_obj = va_arg(valist, void*);
            ((DObject*)tv.str_obj)->IncRef();
            break;
        case TYPE_FUTURE_OBJ:
            tv.future_obj = va_arg(valist, void*);
            ((DObject*)tv.future_obj)->IncRef();
            break;
        case TYPE_DOUBLE:
            tv.num_double = va_arg(valist, double);
            break;
        default:
            cerr << "Error: type undefined" << endl;
    }
    va_end(valist);
    return obj->Append(tv);
}

int64_t GetSliceSize(void *obj) {
    return ((DigoSlice *) obj)->Size();
}

void *GetSliceIndexString(void *obj, int64_t idx) {
    auto s = ((DigoSlice *) obj)->Index(idx).str_obj;
    ((DObject*)s)->IncRef();
    return s;
}

int64_t GetSliceIndexInt(void *obj, int64_t idx) {
```

```
    return ((DigoSlice *) obj)->Index(idx).num64;
}

double GetSliceIndexDouble(void *obj, int64_t idx) {
    return ((DigoSlice *) obj)->Index(idx).num_double;
}

void *GetSliceIndexFuture(void *obj, int64_t idx) {
    auto s = ((DigoSlice *) obj)->Index(idx).future_obj;
    ((DObject*)s)->IncRef();
    return s;
}

void *SetSliceIndexString(void *obj, int64_t idx, void *val) {
    TypeCell tv;
    tv.type = TYPE_STR;
    tv.str_obj = val;
    ((DigoString *)val)->IncRef();
    ((DigoString *)((DigoSlice *) obj)->Index(idx).str_obj)->DecRef();
    ((DigoSlice *) obj)->Index(idx) = tv;
    return val;
}

int64_t SetSliceIndexInt(void *obj, int64_t idx, int64_t val) {
    TypeCell tv;
    tv.type = TYPE_INT64;
    tv.num64 = val;
    ((DigoSlice *) obj)->Index(idx) = tv;
    return val;
}

double SetSliceIndexDouble(void *obj, int64_t idx, double val) {
    TypeCell tv;
    tv.type = TYPE_DOUBLE;
    tv.num_double = val;
    ((DigoSlice *) obj)->Index(idx) = tv;
    return val;
}

void *SetSliceIndexFuture(void *obj, int64_t idx, void *val) {
    TypeCell tv;
    tv.type = TYPE_FUTURE_OBJ;
    tv.future_obj = val;
```

```
    ((DObject*)tv.future_obj)->IncRef();
    ((DObject *)((DigoSlice *) obj)->Index(idx).future_obj)->DecRef();
    ((DigoSlice *) obj)->Index(idx) = tv;
    return val;
}
```

9.3.9 dstring.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_DSTRING_H_
#define ASYNC_REMOTE_LIB_SRC_DSTRING_H_

#include "common.h"
#include <memory>
#include <cstdint>
#include "../../digo-linker/src/gc.h"

class DigoString : public DObject {
public:
    DigoString() = default;
    explicit DigoString(const char *);
    explicit DigoString(const string &);

    DigoString operator+(const DigoString &) const;
    DigoString operator+(const char *) const;

    const string &Data() const;
    int64_t Compare(const DigoString &) const;
    int64_t Size() const;

    const char *name() override {
        return "String Object";
    }

private:
    string raw_data_;

};

extern "C" {

void *CreateString(const char *);
void *CreateEmptyString();
```

```
void *AddString(void *, void *);
void *AddCString(void *, const char *);

void *CloneString(void *);
int64_t CompareString(void *, void *);
int64_t GetStringSize(void *);
const char *GetCStr(void *);

}


#endif //ASYNC_REMOTE_LIB_SRC_RESOURCE_H_
```

9.3.10 dstring.cpp

```
#include <cstring>

#include "dstring.h"

void *CreateString(const char *src) {
    return new DigoString(src);
}

void *CreateEmptyString() {
    return new DigoString();
}

void *AddString(void *vl,
                void *vr) {
    auto l = (DigoString *) vl;
    auto r = (DigoString *) vr;
    return new DigoString(
            *l + *r);
}

void *AddCString(void *vl, const char *r) {
    auto l = (DigoString *) vl;
    return new DigoString(
            *l + r);
}
```

```
void *CloneString(void *vsrc) {
    return new DigoString(((DigoString *) vsrc)->Data());
}

int64_t CompareString(void *vl,
                      void *vr) {
    auto l = (DigoString *) vl;
    auto r = (DigoString *) vr;
    return l->Compare(*r);
}

int64_t GetStringSize(void *vs) {
    auto s = (DigoString *) vs;
    return s->Size();
}

const char *GetCStr(void *vs) {
    auto s = (DigoString *) vs;
    return s->Data().c_str();
}

DigoString::DigoString(const char *src) {
    raw_data_ = src;
}

DigoString::DigoString(const string &src) {
    raw_data_ = src;
}

DigoString DigoString::operator+(const DigoString &r) const {
    return DigoString(this->raw_data_ + r.raw_data_);
}

DigoString DigoString::operator+(const char *r) const {
    return DigoString(this->raw_data_ + r);
}

int64_t DigoString::Compare(const DigoString &r) const {
    return strcmp(this->raw_data_.c_str(), r.raw_data_.c_str());
}

int64_t DigoString::Size() const {
```

```
    return this->raw_data_.length();
}

const string &DigoString::Data() const {
    return this->raw_data_;
}
```

9.3.11 ioutil.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_IOUTIL_H_
#define ASYNC_REMOTE_LIB_SRC_IOUTIL_H_

#include <iostream>

using std::istream;
using std::ifstream;

void* ReadStream(istream &in);

extern "C" {
  void *ReadFile(void* path);
};

#endif //ASYNC_REMOTE_LIB_SRC_IOUTIL_H_
```

9.3.12 ioutil.cpp

```
#include <fstream>

#include "ioutil.h"
#include "dslice.h"
#include "dstring.h"

void* ReadStream(istream &in) {
  auto d_sli = (DigoSlice*)CreateSlice(TYPE_STR);
  string word;
  while (in >> word) {
      auto tmp_str = (DigoString*)CreateString(word.c_str());
    auto next_d_sli = (DigoSlice*)SliceAppend(d_sli, tmp_str);
    d_sli->DecRef();
    tmp_str->DecRef();
    d_sli = next_d_sli;
```

```
  }
  return d_sli;
}

void* ReadFile(void* path) {
  auto path_cstr = ((DigoString *)path)->Data().c_str();
  ifstream file(path_cstr);
  return ReadStream(file);
}
```

9.3.13 linker_common.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_LINKER_COMMON_H_
#define ASYNC_REMOTE_LIB_SRC_LINKER_COMMON_H_

#include "common.h"
#include "../../digo-linker/src/wrapper.h"

extern "C" {
__attribute__((noinline)) void linker_call_function(int32_t func_id, byte
*content, int32 length,
                                                    byte **result, int32
*result_length);
}

inline bytes CallFunctionByName(string digo_func_name, bytes parameters) {
  byte *result;

  int32 result_length = 0;
  linker_call_function(ASYNC_GetFunctionId(digo_func_name.c_str()),
parameters.content.get(), parameters.length,
                       &result, &result_length);
  return bytes{shared_ptr<byte>(result), result_length};
}

#endif //ASYNC_REMOTE_LIB_SRC_LINKER_COMMON_H_
```

9.3.14 common.h

```
#ifndef ASYNC_REMOTE_LIB_SRC_COMMON_H_
#define ASYNC_REMOTE_LIB_SRC_COMMON_H_
```

```
#include <string>
#include <memory>
#include <map>
#include <iostream>
#include <vector>

using std::string;
using std::shared_ptr;
using std::map;
using std::to_string;
using std::cout;
using std::cerr;
using std::vector;
using std::endl;
using std::make_shared;

typedef int32_t int32;
typedef int future_obj_ptr;
typedef unsigned char byte;

typedef struct bytes {
  // FIXED: memory leak when destructing content
  // using delete instead of delete[] to free an array of POD types
(trivial destructor)
  // is safe because it is not necessary to destruct each element of the
array, and
  // freeing the memory allocated is sufficient.
  shared_ptr<byte> content = nullptr;
  int32 length = 0;
} bytes;

class noncopyable {
 protected:
  noncopyable() = default;
  ~noncopyable() = default;

 public:
  noncopyable(const noncopyable &) = delete;
  noncopyable &operator=(const noncopyable &) = delete;
};

#endif //ASYNC_REMOTE_LIB_SRC_COMMON_H_
```

9.3.15 Makefile

```
all: generate-executable

.PHONY: generate-llvm
generate-llvm: clean
	clang -stdlib=libc++ -std=c++2a -O3 -S -emit-
llvm ./src/async.cpp ./src/master_worker.cpp ./src/network.cpp ./src/dstrin
g.cpp ./src/dslice.cpp ./src/ioutil.cpp

.PHONY: link-all-llvm
link-all-llvm: generate-llvm
	llvm-link -S -v -o allinone.ll *.ll

.PHONY: generate-executable
generate-executable: link-all-llvm
	clang++ -stdlib=libc++ -pthread allinone.ll -o executable

.PHONY: clean
clean:
	rm -f *.ll
	rm -f executable
```

9.4 digo-linker

Digo-linker is finished by Sida.

9.4.1 exported_api.ll

```
declare dso_local i8* @CreateString(i8*)
declare dso_local i8* @CreateEmptyString()
declare dso_local i8* @AddString(i8*, i8*)
declare dso_local i8* @AddCString(i8*, i8*)
declare dso_local i8* @CloneString(i8*)
declare dso_local i64 @CompareString(i8*, i8*)
declare dso_local i64 @GetStringSize(i8*)
```

```
declare dso_local i8* @GetCStr(i8*)

declare dso_local void @print(i8*, ...)
declare dso_local void @println(i8*, ...)

declare dso_local i8* @CreateSlice(i64)
declare dso_local i8* @SliceSlice(i8*, i64, i64)
declare dso_local i8* @SliceAppend(i8*, ...)
declare dso_local i8* @CloneSlice(i8*)
declare dso_local i64 @GetSliceSize(i8*)
```

9.4.2 builtin_types.h

```
/* This file includes the headers of Digo Objects for Digo Serialization
Library.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#ifndef DIGO_LINKER_BUILTIN_TYPES_H
#define DIGO_LINKER_BUILTIN_TYPES_H

#include "../../async-remote-lib/src/dstring.h"
#include "../../async-remote-lib/src/dslice.h"

#endif //DIGO_LINKER_BUILTIN_TYPES_H
```

9.4.3 common.h

```
/* This file provides common type definition, for the Digo libraries.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#ifndef DIGO_LINKER_COMMON_H_
#define DIGO_LINKER_COMMON_H_

#include <string>
```

```cpp
#include <memory>
#include <map>
#include <iostream>
#include <utility>
#include <vector>

#include "gc.h"

using std::string;
typedef unsigned char byte;

enum digo_type {
    TYPE_UNDEFINED = 0,
    TYPE_STR = 1,
    TYPE_INT32 = 2,
    TYPE_INT64 = 3,
    TYPE_DOUBLE = 4,
    TYPE_SLICE = 5,
    /*  future object cannot be serialized,
     *  but may be stored in Slice
     */
    TYPE_FUTURE_OBJ = 6,
};

class TypeCell {
public:
    TypeCell() = default;
    explicit TypeCell(string s) : str(std::move(s)), type(TYPE_STR) {}
    explicit TypeCell(int32_t num) : num32(num), type(TYPE_INT32) {}
    explicit TypeCell(int64_t num) : num64(num), type(TYPE_INT64) {}
    explicit TypeCell(double num) : num_double(num), type(TYPE_DOUBLE) {}
    explicit TypeCell(std::vector<TypeCell> a, digo_type t) :
arr(std::move(a)), arr_slice_type(t), type(TYPE_SLICE) {}

    digo_type type = TYPE_UNDEFINED;
    int32_t  num32 = 0;
    int64_t  num64 = 0;
    double   num_double = 0.0;
    void*    slice_obj = nullptr;
    /*  use str_obj for a digo string object  */
    void*    str_obj = nullptr;
    void*    future_obj = nullptr;
```

```
    /*   serializer reserved  */
    string   str;
    std::vector<TypeCell> arr;
    digo_type arr_slice_type = TYPE_UNDEFINED;
};

namespace Linker {

    class noncopyable {
    protected:
        noncopyable() = default;

        virtual ~noncopyable() = default;

    public:
        noncopyable(const noncopyable &) = delete;

        noncopyable &operator=(const noncopyable &) = delete;
    };

}

#endif //DIGO_LINKER_COMMON_H_
```

9.4.4 gc.h

```
/* gc.h provides a C++ base class for Digo objects that need reference
count gc.
 *
 * Digo Slice and Digo String are wrapped in this gc template.
 *
 * Author: sh4081
 * Date: 2021/3/25
 */

#ifndef DIGO_LINKER_GC_H
#define DIGO_LINKER_GC_H

// #define ENABLE_GC_DEBUG

#ifdef ENABLE_GC_DEBUG
```

```
const bool GC_DEBUG = true;

#else

const bool GC_DEBUG = false;

#endif

#include <mutex>
#include <memory>

/*  A Digo object with ref count GC support  */
class DObject {
protected:
    std::mutex ref_lock;
    int ref_cnt = 0;

public:
    virtual ~DObject() = default;

    /*   overrides this function to provide the name of the object;
     *   for debugging purpose.
     */
    virtual const char* name() {
        return "Base Object";
    }

    DObject();
    virtual void IncRef() final;
    virtual void DecRef() final;
};


extern "C" {
void __GC_DecRef(void* obj);
void* __GC_CreateTraceMap();
void  __GC_Trace(void* map, void* obj);
void  __GC_NoTrace(void* map, void* obj);
void  __GC_ReleaseAll(void* map);

void  __GC_DEBUG_COLLECT_LEAK_INFO();
}
```

```
#endif //DIGO_LINKER_GC_H
```

9.4.5 gc.cpp

```
/* This file exports the GC_DecRef API for reference count GC.
 * The compiler may trace Digo Objects allocated in heap, and releases them
 * using this API, if the objects do not escape their scope.
 *
 * Author: sh4081
 * Date: 2021/4/14
 */

#include "gc.h"
#include "builtin_types.h"
#include <unordered_set>
#include <unordered_map>

using namespace std;

/* All Digo Objects are inherited from the base class DObject, so
 * we can have a general IncRef/DecRef wrapper.
 *
 * Since there is no `real` async function in our project
 * (all async functions goes through the serialization),
 * IncRef can be omitted.
 *
 * To simplify the implementation in the codegen (in Digo-Compiler),
 * we provide some helper functions here:
 *
 * 1. void* __GC_CreateTraceMap() to create a tracing map.
 * 2. void  __GC_Trace(void* map, void* obj) to add an object to the
tracing map.
 * 3. void  __GC_NoTrace(void* map, void* obj) to remove an object from the
tracing map.
 * 4. void  __ReleaseAll(void* map) to release(DecRef) all the objects in
the tracing map,
 *                                                  and delete the tracing
map itself.
 */

void __GC_DecRef(void* obj) {
    if (obj == nullptr) {
```

```
        return;
    }
    auto o = (DObject*)obj;
    o->DecRef();
}

struct TraceMap {
    unordered_map<void*, int> ref;
};

void* __GC_CreateTraceMap() {
    return new TraceMap();
}

void  __GC_Trace(void* map, void* obj) {
    auto m = (TraceMap*)map;
    m->ref[obj]++;
}

void  __GC_NoTrace(void* map, void* obj) {
    auto m = (TraceMap*)map;
    // fprintf(stderr, "GC NO_TRACE map: %p, obj: %p\n", map, obj);
    m->ref[obj]--;
}

void  __GC_ReleaseAll(void* map) {
    auto m = (TraceMap*)map;
    for (auto obj : m->ref) {
        // fprintf(stderr, "GC ReleaseAll map: %p, obj: %p, time: %d\n",
map, obj.first, obj.second);
        for (int i = 0; i < obj.second; i++) {
            __GC_DecRef(obj.first);
        }
        if (obj.second < 0) {
            for (int i = 0; i < -(obj.second); i++) {
                ((DObject*)obj.first)->IncRef();
            }
        }
    }
    delete m;
}

TraceMap GC_DEBUG_AllocatedObjects;
```

```
void  __GC_DEBUG_COLLECT_LEAK_INFO() {
    if (GC_DEBUG) {
        for (auto r : GC_DEBUG_AllocatedObjects.ref) {
            if (r.second != 0) {
                fprintf(stderr, "GC Leak: %p\n", r.first);
            }
        }
    }
}

DObject::DObject() {
    ref_cnt = 1;
    if (GC_DEBUG) {
        fprintf(stderr, "GC Debug: %p is created\n", this);
        GC_DEBUG_AllocatedObjects.ref[this] = 1;
    }
}

void DObject::IncRef() {
    this->ref_lock.lock();
    this->ref_cnt++;
    if (GC_DEBUG) {
        fprintf(stderr, "GC Debug: ref cnt of %s, %p is incremented
to %d\n", name(), this, this->ref_cnt);
        GC_DEBUG_AllocatedObjects.ref[this]++;
    }
    this->ref_lock.unlock();
}

void DObject::DecRef() {
    // fprintf(stderr, "You are visiting GC System: %p\n", this);
    this->ref_lock.lock();
    this->ref_cnt--;
    if (GC_DEBUG) {
        fprintf(stderr, "GC Debug: ref cnt of %s, %p is decremented
to %d\n", name(), this, this->ref_cnt);
        GC_DEBUG_AllocatedObjects.ref[this]--;
    }
    if (this->ref_cnt < 0) {
        fprintf(stderr, "using an already released object\n");
        exit(1);
    }
```

```
    if (this->ref_cnt == 0) {
        this->ref_lock.unlock();
        delete this;
    } else {
        this->ref_lock.unlock();
    }
}
```

9.4.6 linker_main.cpp

```
/* This file is the entry of the Digo Linker.
 *
 * Author: sh4081
 * Date: 2021/3/25
 */

#include "metadata.h"
#include <iostream>
#include <fstream>

using namespace std;

int generate_async_call_entry(const string& input_file, const string&
output_file) {
    fstream s;
    s.open(input_file, ios::in);
    string ir;
    string tmp;
    while (getline(s, tmp)) {
        ir += tmp + "\n";
    }
    Metadata metadata;
    metadata.ParseFuncMetadataFromLLIR(ir);

    fstream output;
    output.open(output_file, ios::out);

    output << R"XXXXX(
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
    )XXXXX";
```

```
    output << metadata.GenerateDeclare();
    output << metadata.GenerateAsyncCalls();
    output << metadata.GenerateJumpTable();
    output << metadata.GenerateEntry();

    return 0;
}

int main(int argc,char *argv[]) {
    if (argc != 4) return 1;
    string command = argv[1];
    string input_file = argv[2];
    string output_file = argv[3];
    if (command == "async") {
        generate_async_call_entry(input_file, output_file);
    } else {
        return 1;
    }
    return 0;
}
```

9.4.7 main.cpp

```
/* This file is for testing purpose, and is not compiled as part of the
Digo Linker.
 *
 * Author: sh4081
 * Date: 2021/3/25
 */

#include <iostream>
#include <fstream>
#include <cmath>
#include "serialization.h"
#include "metadata.h"
#include "wrapper.h"
#include "serialization_wrapper.h"
#include "builtin_types.h"

using namespace std;

int test_serialization();
```

```
int generate_async_call_entry(const string& input_file, const string&
output_file);

int main() {

    return 0;

    string command = "async";
    string input_file = "../../digo-linker/test/test-async-2.ll";
    string output_file = "../../digo-linker/test/test-async-2.ll.out";
    if (command == "async") {
        generate_async_call_entry(input_file, output_file);
    } else {
        return 1;
    }
    return 0;
}

int generate_async_call_entry(const string& input_file, const string&
output_file) {
    fstream s;
    s.open(input_file, ios::in);
    string ir;
    string tmp;
    while (getline(s, tmp)) {
        ir += tmp + "\n";
    }
    Metadata metadata;
    metadata.ParseFuncMetadataFromLLIR(ir);

    fstream output;
    output.open(output_file, ios::out);

    output << metadata.GenerateDeclare();
    output << metadata.GenerateAsyncCalls();
    output << metadata.GenerateJumpTable();
    output << metadata.GenerateEntry();

    return 0;
}

void print_slice(const vector<TypeCell> & arr) {
```

```
    std::cout << " slice with size " << arr.size() << ": ";
    for (int i = 0; i < arr.size(); i++) {
        auto cell = arr[i];
        switch(cell.type) {
            case TYPE_STR:
                std::cout << cell.str;
                break;
            case TYPE_INT32:
                std::cout << cell.num32;
                break;
            case TYPE_INT64:
                std::cout << cell.num64;
                break;
            case TYPE_DOUBLE:
                std::cout << cell.num_double;
                break;
            default:
                std::cout << "error!";
        }
        std::cout << " | ";
    }
}

int test_serialization() {
    Serialization s;
    s.AddInt32(100);
    s.AddInt64(200);
    auto str = "12345-- -- -- && TEST -- --- /*9*---*
**d//s*;;;;;;;;~~`1````1";
    auto str2 = "22345-- -- -- && TEST -- --- /*9*---*
**d//s*;;;;;;;;~~`1````1";
    s.AddString(str);
    s.AddString(str2);
    s.AddInt32(INT32_MAX);
    s.AddInt32(INT32_MIN);
    s.AddInt64(INT64_MAX);
    s.AddInt64(INT64_MIN);
    s.AddDouble(10.403);
    s.AddDouble(sqrt(-1));
    s.AddDouble(log(-1));
    s.AddDouble(1 / 0.0);
    s.AddDouble(-1 / 0.0);
    s.AddDouble(-10.0);
```

```
    s.AddSlice({TypeCell("1234-EST -- --- /*9*---* **d//s*;;;;;;;;~~`1"),
                TypeCell("2234-EST -- --- /*9*---*
**d//s*;;;223;;;;;~~`1"),
                TypeCell("1234")}, TYPE_STR);
    /*  mixed type is only for test purpose  */
    s.AddSlice({TypeCell(-23.403),
                TypeCell("2234-"),
                TypeCell(100203033)}, TYPE_STR);
    s.AddDouble(-102222222.1);

    auto serialized = s.Get();

    Serialization s_ext;
    s_ext.Extract(serialized);

    for (const auto& cell : s_ext.GetResult().extracted_cells) {
        std::cout << "TypeCell: " << cell.type << " content: ";
        switch(cell.type) {
            case TYPE_STR:
                std::cout << cell.str;
                break;
            case TYPE_INT32:
                std::cout << cell.num32;
                break;
            case TYPE_INT64:
                std::cout << cell.num64;
                break;
            case TYPE_DOUBLE:
                std::cout << cell.num_double;
                break;
            case TYPE_SLICE:
                print_slice(cell.arr);
                break;
            default:
                std::cout << "error!";
        }
        std::cout << std::endl;
    }

    return 0;
}
```

9.4.8 metadata.cpp

```
/* The IR pass in Digo Linker.
 * It parses the function metadata, generates serialization code for each
async (local/remote)
 * function, and provides simple APIs for the Digo Compiler and Digo Async
Remote Library.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#include "metadata.h"
#include <regex>
#include <sstream>
#include <utility>

#define FMT_HEADER_ONLY
#include "../third-party/fmt/format.h"

using namespace std;

const string metadata_begin = "; DIGO Async Function Metadata BEGIN";
const string metadata_end = "; DIGO Async Function Metadata END";

const regex version_regex("; VERSION = (\\d+)");

const regex func_decl_regex("; FUNC DECLARE BEGIN\n; FUNC_NAME = '(.+)'\n"
                            "; FUNC_ANNOT = '(.+)'\n"
                            "; PARAMETERS = '(.*)'\n; RETURN_TYPE =
'(.*)'\n"
                            "; FUNC DECLARE END\n");


class WrongVersionException: public exception {
public:
    const char * what() const noexcept override {
        return "metadata version != 1";
    }
};

class IncorrectMetadataException: public exception {
public:
```

```
    IncorrectMetadataException(): msg("incorrect metadata") {}
    IncorrectMetadataException(string m): msg(std::move(m)) {}
    const char * what() const noexcept override {
        return msg.c_str();
    }
private:
    string msg;
};

class IncorrectIRException: public exception {
public:
    const char * what() const noexcept override {
        return "incorrect ir";
    }
};

vector<string> split(const string &s, char delim) {
    vector<string> result;
    stringstream ss (s);
    string item;

    while (getline (ss, item, delim)) {
        result.push_back(item);
    }

    return result;
}

void Metadata::ParseFuncMetadataFromLLIR(const string &ir) {
    vector<FuncPrototype> ret;

    auto pos_begin = ir.find(metadata_begin) + metadata_begin.size();
    if (pos_begin == string::npos) {
        throw IncorrectMetadataException();
    }
    auto pos_end = ir.find(metadata_end, pos_begin);
    if (pos_end == string::npos) {
        throw IncorrectMetadataException();
    }

    auto metadata_part = ir.substr(pos_begin, pos_end - pos_begin + 1);

    regex multiple_newline(R"((\n|\r\n|\r)+)");
```

```
    metadata_part = regex_replace(metadata_part, multiple_newline, "\n");

    auto iter = sregex_iterator(metadata_part.begin(),
                                    metadata_part.end(), version_regex);

    for (; iter != sregex_iterator() ; iter++) {
        string ver = iter->str(1);
        if (stoi(ver) != 1) {
            throw WrongVersionException();
        }
    }

    iter = sregex_iterator(metadata_part.begin(), metadata_part.end(),
                           func_decl_regex);

    for (; iter != sregex_iterator() ; iter++) {
        string func_name = iter->str(1);
        string func_annotation = iter->str(2);
        string parameters = iter->str(3);
        string return_type = iter->str(4);
        if (func_name.empty()) {
            throw IncorrectMetadataException();
        }
        regex multiple_space("\\s+");
        parameters = regex_replace(parameters, multiple_space, "");
        return_type = regex_replace(return_type, multiple_space, "");

        FuncPrototype prototype;
        if (func_annotation == "async") {
            prototype.is_remote = 0;
        } else if (func_annotation == "async remote") {
            prototype.is_remote = 1;
        } else {
            throw IncorrectMetadataException("async/async remote
expected");
        }

        prototype.func_name = func_name;
        vector<string> p = split(parameters, ',');
        for(const string& str : p) {
            if (str == "int") {
                prototype.parameters.push_back(TYPE_INT64);
            } else if (str == "string") {
```

```
                prototype.parameters.push_back(TYPE_STR);
            } else if (str == "double") {
                prototype.parameters.push_back(TYPE_DOUBLE);
            } else if (str == "slice") {
                prototype.parameters.push_back(TYPE_SLICE);
            } else {
                throw IncorrectMetadataException("wrong parameter type");
            }
        }

        p = split(return_type, ',');

        for(const string& str : p) {
            if (str == "int") {
                prototype.return_type.push_back(TYPE_INT64);
            } else if (str == "string") {
                prototype.return_type.push_back(TYPE_STR);
            } else if (str == "double") {
                prototype.return_type.push_back(TYPE_DOUBLE);
            } else if (str == "slice") {
                prototype.return_type.push_back(TYPE_SLICE);
            } else {
                throw IncorrectMetadataException("wrong return type");
            }
        }
        ret.push_back(prototype);
    }

    functions_prototype_ = ret;
}

string Metadata::GenerateJumpTable() {
    string jump_template = R"XXXXX(

define i32 @linker_call_function(i32 %func_id, i8* %arg, i32 %arg_len,
i8** %result, i32* %result_len) {
  call void @Debug_Real_LinkerCallFunction(i32 %func_id, i32 %arg_len)

  %wrapper = call i8* @SW_CreateWrapper()
  %extractor = call i8* @SW_CreateExtractor(i8* %arg, i32 %arg_len)

  switch i32 %func_id, label %if.nomatch [
```

```
)XXXXX";

    for (int i = 0 ; i < functions_prototype_.size(); i++) {
        jump_template += "    i32 " + to_string(i) + ", label %if.func" +
to_string(i) + "\n";
    }

    jump_template += R"XXXXX(

  ]

#<labels>#

if.nomatch:
  call void @NoMatchExceptionHandler(i32 %func_id)
  ret i32 0

if.end:

  call void @SW_DestroyExtractor(i8* %extractor)

  ret i32 0
}

)XXXXX";

    jump_template = regex_replace(jump_template, regex("\\{"), "{{");
    jump_template = regex_replace(jump_template, regex("\\}"), "}}");
    jump_template = regex_replace(jump_template, regex("#<([a-z_]+)>#"),
"{$1}");

    string labels;

    for (int i = 0; i < functions_prototype_.size(); i++) {
        labels += GenerateJumpLabel(i, functions_prototype_.at(i));
    }

    auto result = fmt::format(jump_template, fmt::arg("labels", labels));

    return result;
}

string Metadata::GenerateJumpLabel(int id, const FuncPrototype &proto) {
```

```
    string label_template = R"XXXXX(
if.func#<id>#:
  #<arg_extractor>#
  %aggResult#<id># = call #<ret_type># @#<func_name>#(#<arguments>#)
#<ret_serializer>#
  call void @SW_GetAndDestroy(i8* %wrapper, i8** %result, i32* %result_len)
#<dec_ref>#
  br label %if.end
)XXXXX";

    label_template = regex_replace(label_template, regex("\\{"), "{{");
    label_template = regex_replace(label_template, regex("\\}"), "}}");
    label_template = regex_replace(label_template, regex("#<([a-z_]+)>#"),
"{$1}");

    /*  We do not need the digo function result after serializing them.
     *  So we have to do GC_DecRef.
     */
    string dec_ref;

    for (int i = 0; i < proto.return_type.size(); i++) {
        auto t = proto.return_type[i];
        if (t == TYPE_STR || t == TYPE_SLICE) {
            dec_ref += "  call void @__GC_DecRef(i8* %aggResult" +
to_string(id) + "_tmp_" + to_string(i) + ")\n";
        }
    }

    /*  We also do not need the digo objects extracted by
#<arg_extractor>#.
     *  So we have to do GC_DecRef.
     */
    for (int i = 0; i < proto.parameters.size(); i++) {
        auto t = proto.parameters[i];
        if (t == TYPE_STR || t == TYPE_SLICE) {
            dec_ref += "  call void @__GC_DecRef(i8* %arg" + to_string(id)
+ "_" + to_string(i) + ")\n";
        }
    }

    string ret_type = "{ " + GenerateArgumentsType(proto.return_type) +
" }";
```

```
    auto result = fmt::format(label_template, fmt::arg("id", id),
                              fmt::arg("func_name", proto.func_name),
                              fmt::arg("ret_type", ret_type),
                              fmt::arg("arguments",
GenerateArgumentsDef(proto.parameters,

"arg" + to_string(id) + "_")),
                              fmt::arg("arg_extractor",
GenerateExtractor(proto.parameters,

"arg" + to_string(id) + "_")),
                              fmt::arg("ret_serializer",

GenerateSerializerAggregated(proto.return_type,

"%aggResult" + to_string(id))),
                              fmt::arg("dec_ref", dec_ref));
    return result;
}

// serialize %<prefix>0, 1, ...
string Metadata::GenerateSerializer(const vector<digo_type> & types, const
string & prefix) {
    string result;
    for (int i = 0; i < types.size(); i++) {
        auto type = types[i];
        if (type == TYPE_INT32) {
            result += R"(
  call void @SW_AddInt32(i8* %wrapper, i32 %)" + prefix + to_string(i) +
")";
        } else if (type == TYPE_INT64) {
            result += R"(
  call void @SW_AddInt64(i8* %wrapper, i64 %)" + prefix + to_string(i) +
")";
        } else if (type == TYPE_STR) {
            result += R"(
  call void @SW_AddString(i8* %wrapper, i8* %)" + prefix + to_string(i) +
")";
        } else if (type == TYPE_SLICE) {
            result += R"(
  call void @SW_AddSlice(i8* %wrapper, i8* %)" + prefix + to_string(i) +
")";
        } else if (type == TYPE_DOUBLE) {
```

```
            result += R"(
  call void @SW_AddDouble(i8* %wrapper, double %)" + prefix + to_string(i)
+ ")";
        }
    }
    return result;
}

// serialize an aggregate type, agg_name should include "%"
string Metadata::GenerateSerializerAggregated(const vector<digo_type> &
types, const string & agg_name) {
    string result;
    int auto_inc_reg = 0;
    string agg_type = "{ " + GenerateArgumentsType(types) + " }";

    string extract_value_template =
            "  {to_reg} = extractvalue {agg_type} " + agg_name + ", {i}\n";

    for (int i = 0; i < types.size(); i++) {
        auto type = types[i];
        string reg = agg_name + "_tmp_" + to_string(auto_inc_reg++);
        if (type == TYPE_INT32) {
            result += fmt::format(extract_value_template,
                                  fmt::arg("to_reg", reg),
                                  fmt::arg("agg_type", agg_type),
                                  fmt::arg("i", i));
            result += R"(
  call void @SW_AddInt32(i8* %wrapper, i32 )" + reg + ")\n";
        } else if (type == TYPE_INT64) {
            result += fmt::format(extract_value_template,
                                  fmt::arg("to_reg", reg),
                                  fmt::arg("agg_type", agg_type),
                                  fmt::arg("i", i));
            result += R"(
  call void @SW_AddInt64(i8* %wrapper, i64 )" + reg + ")\n";
        } else if (type == TYPE_STR) {
            result += fmt::format(extract_value_template,
                                  fmt::arg("to_reg", reg),
                                  fmt::arg("agg_type", agg_type),
                                  fmt::arg("i", i));
            result += R"(
  call void @SW_AddString(i8* %wrapper, i8* )" + reg + ")\n";
        } else if (type == TYPE_SLICE) {
```

```
            result += fmt::format(extract_value_template,
                                  fmt::arg("to_reg", reg),
                                  fmt::arg("agg_type", agg_type),
                                  fmt::arg("i", i));
            result += R"(
  call void @SW_AddSlice(i8* %wrapper, i8* )" + reg + ")\n";
        } else if (type == TYPE_DOUBLE) {
            result += fmt::format(extract_value_template,
                                  fmt::arg("to_reg", reg),
                                  fmt::arg("agg_type", agg_type),
                                  fmt::arg("i", i));
            result += R"(
  call void @SW_AddDouble(i8* %wrapper, double )" + reg + ")\n";
        }
    }
    return result;
}

// each item extracted is stored in %<padding>0, 1, ...
string Metadata::GenerateExtractor(const vector<digo_type> & types, const
string& padding) {
    string result;
    for (int i = 0; i < types.size(); i++) {
        auto type = types[i];
        if (type == TYPE_INT32) {
            result += R"(
  %)" + padding + to_string(i) +  R"( = call i32
@SW_ExtractInt32(i8* %extractor)
)";
        } else if (type == TYPE_INT64) {
            result += R"(
  %)" + padding + to_string(i) +  R"( = call i64
@SW_ExtractInt64(i8* %extractor)
)";
        } else if (type == TYPE_STR) {
            result += R"(
  %)" + padding + to_string(i) +  R"( = call i8*
@SW_ExtractString(i8* %extractor)
)";
        } else if (type == TYPE_SLICE) {
            result += R"(
  %)" + padding + to_string(i) +  R"( = call i8*
@SW_ExtractSlice(i8* %extractor)
```

```
)";
        } else if (type == TYPE_DOUBLE) {
            result += R"(
  %)" + padding + to_string(i) +  R"( = call double
@SW_ExtractDouble(i8* %extractor)
)";
        }
    }
    return result;
}

// type list. e.g. i8, i32, i8* ....
string Metadata::GenerateArgumentsType(const vector<digo_type> &types) {
    string result;
    if (types.empty()) return result;
    for (int i = 0; i < types.size(); i++) {
        auto type = types[i];
        if (type == TYPE_INT32) {
            result += R"(i32, )";
        } else if (type == TYPE_INT64) {
            result += R"(i64, )";
        } else if (type == TYPE_STR) {
            result += R"(i8*, )";
        } else if (type == TYPE_SLICE) {
            result += R"(i8*, )";
        } else if (type == TYPE_DOUBLE) {
            result += R"(double, )";
        }
    }
    result = result.substr(0, result.length() - 2);
    return result;
}

// each argument is named %padding0, %padding1, ...
string Metadata::GenerateArgumentsDef(const vector<digo_type> &types, const
string & padding) {
    string result;
    if (types.empty()) return result;
    for (int i = 0; i < types.size(); i++) {
        auto type = types[i];
        if (type == TYPE_INT32) {
            result += R"(i32 %)" + padding + to_string(i) + ", ";
        } else if (type == TYPE_INT64) {
```

```
            result += R"(i64 %)" +padding + to_string(i) + ", ";
        } else if (type == TYPE_STR) {
            result += R"(i8* %)" + padding + to_string(i) + ", ";
        } else if (type == TYPE_SLICE) {
            result += R"(i8* %)" + padding + to_string(i) + ", ";
        } else if (type == TYPE_DOUBLE) {
            result += R"(double %)" + padding + to_string(i) + ", ";
        }
    }
    result = result.substr(0, result.length() - 2);
    return result;
}

string Metadata::GenerateAsyncAsLLVMIR(int id, const FuncPrototype &proto)
{
    string async_template = R"XXXXX(
define i8* @digo_linker_async_call_func_#<func_name>#(#<arg_def>#) {
  %call = call i8* @digo_linker_async_call_id_#<id>#(#<arg_def>#)
  ret i8* %call
}

declare dso_local #<ret_type_list># @#<func_name>#(#<arg_def>#)

define i8* @digo_linker_async_call_id_#<id>#(#<arg_def>#) {
entry:
  %wrapper = call i8* @SW_CreateWrapper()

  #<arg_serialization>#

  %result = alloca i8*, align 8
  %len = alloca i32, align 4

  call void @SW_GetAndDestroy(i8* %wrapper, i8** %result, i32* %len)

  %result_in = load i8*, i8** %result, align 8
  %len_in = load i32, i32* %len, align 4

  %future_obj = call i8* @#<job_call>#(i32 #<id>#, i8* %result_in,
i32 %len_in)

  ret i8* %future_obj
}
```

```
define #<ret_type_list>#
@digo_linker_await_func_#<func_name>#(i8* %arg_future_obj) {
  %call = call #<ret_type_list>#
@digo_linker_await_id_#<id>#(i8* %arg_future_obj)
  ret #<ret_type_list># %call
}

define #<ret_type_list># @digo_linker_await_id_#<id>#(i8* %arg_future_obj)
{
  %result = alloca i8*, align 8
  %len = alloca i32, align 4
  call void @AwaitJob(i8* %arg_future_obj, i8** %result, i32* %len)

  %result_in = load i8*, i8** %result, align 8
  %len_in = load i32, i32* %len, align 4

  %extractor = call i8* @SW_CreateExtractor(i8* %result_in, i32 %len_in)

  #<ret_extractor>#

  call void @SW_DestroyExtractor(i8* %extractor)

#<ret_formation>#

  ret #<ret_type_list># %aggRet
}
)XXXXX";

    async_template = regex_replace(async_template, regex("\\{"), "{{");
    async_template = regex_replace(async_template, regex("\\}"), "}}");
    async_template = regex_replace(async_template, regex("#<([a-z_]+)>#"),
"{$1}");

    string ret_formation;
    string ret_type = "{ " + GenerateArgumentsType(proto.return_type) +
" }";

    string ret_formation_template = "  {this_var} = insertvalue {ret_type}
{last_var}, {this_type} %ret{i}, {i}\n";

    for (int i = 0; i < proto.return_type.size(); i++) {
        string last_var = "undef";
        if (i > 0) last_var = "%ret_agg_" + to_string(i - 1);
```

```
        string this_var = "%ret_agg_" + to_string(i);
        if (i == (int)proto.return_type.size() - 1) this_var = "%aggRet";

        string this_type;
        auto type = proto.return_type[i];
        if (type == TYPE_INT32) {
            this_type = "i32";
        } else if (type == TYPE_INT64) {
            this_type = "i64";
        } else if (type == TYPE_STR) {
            this_type = "i8*";
        } else if (type == TYPE_SLICE) {
            this_type = "i8*";
        } else if (type == TYPE_DOUBLE) {
            this_type = "double";
        }

        ret_formation += fmt::format(ret_formation_template,
fmt::arg("this_var", this_var),
                                     fmt::arg("ret_type", ret_type),
                                     fmt::arg("last_var", last_var),
                                     fmt::arg("this_type", this_type),
                                     fmt::arg("i", i));
    }

    auto result = fmt::format(async_template, fmt::arg("id", id),
    fmt::arg("arg_serialization", GenerateSerializer(proto.parameters,
"arg")),
    fmt::arg("arg_def", GenerateArgumentsDef(proto.parameters, "arg")),
    fmt::arg("ret_type_list", ret_type),
    fmt::arg("ret_extractor", GenerateExtractor(proto.return_type, "ret")),
    fmt::arg("func_name", proto.func_name),
    fmt::arg("job_call", proto.is_remote ? "CreateRemoteJob" :
"CreateAsyncJob"),
    fmt::arg("ret_formation", ret_formation));

    return result;
}

string Metadata::GenerateAsyncCalls() {
    string result;

    for (int i = 0; i < functions_prototype_.size(); i++) {
```

```
        result += GenerateAsyncAsLLVMIR(i, functions_prototype_[i]);
    }

    return result;
}

string Metadata::GenerateDeclare() {
    string declare_template = R"XXXXX(

declare dso_local void @digo_main()
declare dso_local void @AwaitJob(i8*, i8**, i32*)
declare dso_local void @JobDecRef(i8*)
declare dso_local i8* @CreateAsyncJob(i32, i8*, i32)
declare dso_local i8* @CreateRemoteJob(i32, i8*, i32)

declare dso_local i8* @SW_CreateWrapper()
declare dso_local void @SW_AddString(i8*, i8*)
declare dso_local void @SW_AddInt32(i8*, i32)
declare dso_local void @SW_AddInt64(i8*, i64)
declare dso_local void @SW_AddDouble(i8*, double)
declare dso_local void @SW_AddSlice(i8*, i8*)
declare dso_local void @SW_GetAndDestroy(i8*, i8**, i32*)

declare dso_local i8* @SW_CreateExtractor(i8*, i32)
declare dso_local i32 @SW_ExtractInt32(i8*)
declare dso_local i64 @SW_ExtractInt64(i8*)
declare dso_local double @SW_ExtractDouble(i8*)
declare dso_local i8* @SW_ExtractString(i8*)
declare dso_local i8* @SW_ExtractSlice(i8*)
declare dso_local void @SW_DestroyExtractor(i8*)

declare dso_local void @NoMatchExceptionHandler(i32 %func_id)
declare dso_local void @ASYNC_AddFunction(i32, i8*)

declare dso_local i32 @entry(i32, i8**)

declare dso_local void @Debug_Real_LinkerCallFunction(i32, i32)
declare dso_local void @__DIGO_RUNTIME_OnExit()

declare dso_local void @__GC_DecRef(i8*)

)XXXXX";
```

```
    return declare_template;
}

std::tuple<string, string> Metadata::GenerateFuncNameIdMap(int id, const
FuncPrototype &proto) {
    string str_name = "@.str.digo.linker.async.func.name" + to_string(id);
    string str_bound = "[" + to_string(proto.func_name.size()+1) + " x
i8]";
    string str = str_name +
            " = private unnamed_addr constant " + str_bound +
            " c\"" + proto.func_name + "\\00\", align 1";
    string mapadd = "call void @ASYNC_AddFunction(i32 " + to_string(id) +
            ", i8* getelementptr inbounds (" + str_bound +
            ", " + str_bound + "* " + str_name + ", i64 0, i64 0))";
    return {str, mapadd};
}

string Metadata::GenerateEntry() {
    string str_def;
    string result = R"XXXXX(

define void @init_async_function_table() {
  )XXXXX";

    for (int i = 0; i < functions_prototype_.size(); i++) {

        auto [str, mapadd] = GenerateFuncNameIdMap(i,
functions_prototype_[i]);

        str_def += str + "\n";
        result += mapadd + "\n";
    }

    result +=
  R"XXXXX(
  ret void
}

define i32 @main(i32 %argc, i8** %argv) {
entry:
  call void @init_async_function_table()

  %morw = call i32 @entry(i32 %argc, i8** %argv)
```

```
  switch i32 %morw, label %if.worker [
    i32 1, label %if.master
    i32 2, label %if.worker
  ]

if.master:
  call void @digo_main()
  call void @__DIGO_RUNTIME_OnExit()
  ret i32 0

if.worker:
  call void @__DIGO_RUNTIME_OnExit()
  ret i32 0
}
)XXXXX";
    return str_def + "\n\n" + result;
}
```

9.4.9 metadata.h

```
/* See metadata.cpp for details.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#ifndef DIGO_LINKER_METADATA_H
#define DIGO_LINKER_METADATA_H

#include "common.h"

#include <vector>

using std::vector;

class FuncPrototype {
public:
    string              func_name;
    int                 is_remote;
    vector<digo_type>   parameters;
```

```
    vector<digo_type>   return_type;
};

class FuncClosure {
public:
    string              func_name;
    vector<TypeCell>    parameters;
};

class Metadata: public Linker::noncopyable {
public:
    void    ParseFuncMetadataFromLLIR(const string & ir);

    string  GenerateJumpTable();
    string  GenerateAsyncCalls();

    string  GenerateDeclare();
    string  GenerateEntry();

private:

    string  GenerateAsyncAsLLVMIR(int id, const FuncPrototype & proto);

    string GenerateSerializer(const vector<digo_type> & types, const string
& prefix);
    string GenerateSerializerAggregated(const vector<digo_type> & types,
const string & agg_name);

    string GenerateExtractor(const vector<digo_type> & types, const string&
padding);
    string GenerateArgumentsDef(const vector<digo_type> & types, const
string &padding);
    string GenerateArgumentsType(const vector<digo_type> &types);

    string GenerateJumpLabel(int id, const FuncPrototype & proto);
    vector<FuncPrototype> functions_prototype_;

    std::tuple<string, string> GenerateFuncNameIdMap(int id, const
FuncPrototype & proto);
};

#endif //DIGO_LINKER_METADATA_H
```

9.4.10 print_funcs.cpp

```
/* This file provides print & println built-in Digo functions.
 *
 * Author: sh4081
 * Date: 2021/4/10
 */

#include <string>
#include <sstream>

#include <cstdarg>

#include "builtin_types.h"

namespace Print {
    template<typename T>
    string ToString(T num) {
        return std::to_string(num);
    }
    template<>
    string ToString<DigoString*> (DigoString * obj) {
        string ret = obj->Data();
        return ret;
    }
    template<>
    string ToString<void*> (void * obj) {
        std::ostringstream buf;
        buf << "Object: " << obj;
        return buf.str();
    }
    template<>
    string ToString<DigoSlice*> (DigoSlice * obj) {
        auto [underlying_arr, begin, end] = obj->Data();
        auto sliceType = obj->Type();
        if (underlying_arr.empty() || begin == end) {
            return "[]";
        }
        string ret = "[";
        for (auto i = begin; i < end; i++) {
            switch (sliceType) {
                case TYPE_DOUBLE:
                    ret += ToString(underlying_arr[i].num_double);
```

```
                    break;
                case TYPE_INT32:
                    ret += ToString(underlying_arr[i].num32);
                    break;
                case TYPE_INT64:
                    ret += ToString(underlying_arr[i].num64);
                    break;
                case TYPE_STR:
                    ret +=
ToString((DigoString*)(underlying_arr[i].str_obj));
                    break;
                case TYPE_SLICE:
                    ret += "Error: Nested Slice";
                    break;
                case TYPE_UNDEFINED:
                    ret += "Error: Type undefined";
                    break;
                case TYPE_FUTURE_OBJ:
                    ret += ToString(underlying_arr[i].future_obj);
                    break;
            }
            ret += ", ";
        }
        ret.pop_back();
        ret.pop_back();
        ret.push_back(']');
        return ret;
    }


    /*  our printf, %d => int, %s => string(obj), %x => future(obj),
     *  %f => double, %l => slice(obj)
     */
    string ToStringV(const string & format, va_list va) {
        string result;
        for (int i = 0; i < (int)format.size(); i++) {
            if (format[i] != '%') {
                result += format[i];
            } else {
                i++;
                if (i >= (int)format.size()) {
                    result += "Invalid Format";
                    break;
```

```
                }
                switch (format[i]) {
                    case '%':
                        result += "%";
                        break;
                    case 'd':
                        result += ToString(va_arg(va, int64_t));
                        break;
                    case 'f':
                        result += ToString(va_arg(va, double));
                        break;
                    case 'x':
                        result += ToString(va_arg(va, void*));
                        break;
                    case 's':
                        result += ToString(va_arg(va, DigoString*));
                        break;
                    case 'l':
                        result += ToString(va_arg(va, DigoSlice*));
                        break;
                    default:
                        result += "Invalid Format";
                        break;
                }
            }
        }
        return result;
    }
}

extern "C" {
    /*  we only have two exported functions: print & println   */
    void print(void * format, ...);
    void println(void * format, ...);
}

void print(void * format, ...) {
    auto raw_format = GetCStr(format);
    va_list va;
    va_start(va, format);
    string str = Print::ToStringV(raw_format, va);
    va_end(va);
    cout << str;
```

```
}

void println(void * format, ...) {
    auto raw_format = GetCStr(format);
    va_list va;
    va_start(va, format);
    string str = Print::ToStringV(raw_format, va) + "\n";
    va_end(va);
    cout << str;
}
```

9.4.11 serialization.cpp

```
/* The Digo Serialization Library.
 * It provides serialization interface for each Digo type.
 * Compiler does not use these interfaces directly.
 * Instead, the details of serialization is hidden in the interfaces
provided by Digo Linker.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#include "serialization.h"

Serialization::Serialization() {

}

void Serialization::Extract(byte *stream, int len) {
    if (stream == nullptr || len == 0) {
        throw EmptyStreamException();
    }
    vector<byte> tmp;
    tmp.reserve(len);
    for (int i = 0; i < len; i++) {
        tmp.push_back(stream[i]);
    }
    Extract(tmp);
}

void Serialization::Extract(vector<byte> & stream) {
```

```
    // extract header to get type info
    ExtractionResult result;
    int iter = 0;
    for ( ; ; ) {
        // 4 bytes header indicating the type
        auto type = (digo_type)ExtractInt32NoHeader(stream, &iter);
        switch (type) {
            case TYPE_STR: {
                auto str = (ExtractStringNoHeader(stream, &iter));
                result.extracted_cells.emplace_back(str);
                break;
            }
            case TYPE_INT32: {
                auto num = ExtractInt32NoHeader(stream, &iter);
                result.extracted_cells.emplace_back(num);
                break;
            }
            case TYPE_INT64: {
                auto num = ExtractInt64NoHeader(stream, &iter);
                result.extracted_cells.emplace_back(num);
                break;
            }
            case TYPE_DOUBLE: {
                auto num = ExtractDoubleNoHeader(stream, &iter);
                result.extracted_cells.emplace_back(num);
                break;
            }
            case TYPE_SLICE: {
                auto slice = ExtractSliceNoHeader(stream, &iter);
                result.extracted_cells.emplace_back(std::get<0>(slice),
std::get<1>(slice));
                break;
            }
            case TYPE_FUTURE_OBJ:
                throw NotSerializableException("future_obj cannot be
serialized");
            case TYPE_UNDEFINED:
                throw NotSerializableException("type undefined");
        }
        if (iter == stream.size()) {
            break;
        }
    }
```

```
    this->extraction_result_ = result;
    this->extraction_ptr_ = 0;
}

TypeCell Serialization::ExtractOne() {
    int idx = this->extraction_ptr_;
    int size = this->extraction_result_.extracted_cells.size();
    if (idx >= size) {
        throw ExtractionWrongIndexException(idx, size);
    }
    auto ret =
this->extraction_result_.extracted_cells.at(this->extraction_ptr_);
    this->extraction_ptr_++;
    return ret;
}

void Serialization::AddSlice(const vector<TypeCell> && arr, digo_type
sliceType) {
    return AddSlice(arr, sliceType);
}

void Serialization::AddSlice(const vector<TypeCell> & arr, digo_type
sliceType) {
    AddHeader(TYPE_SLICE);
    AddInt32NoHeader(arr.size());
    AddInt32NoHeader((int32_t)sliceType);
    for (int i = 0; i < arr.size(); i++) {
        digo_type type = arr[i].type;
        auto cell = arr[i];
        switch (type) {
            case TYPE_STR: {
                AddString(cell.str);
                break;
            }
            case TYPE_INT32: {
                AddInt32(cell.num32);
                break;
            }
            case TYPE_INT64: {
                AddInt64(cell.num64);
                break;
            }
            case TYPE_DOUBLE: {
```

```
                AddDouble(cell.num_double);
                break;
            }
            case TYPE_SLICE: {
                throw NotSerializableException("are you sure to support
nested slice?");
            }
            case TYPE_FUTURE_OBJ:
                throw NotSerializableException("future_obj cannot be
serialized");
            case TYPE_UNDEFINED:
                throw NotSerializableException("type undefined");
        }
    }
}

std::tuple<vector<TypeCell>, digo_type>
Serialization::ExtractSliceNoHeader(vector<byte> & stream, int *iter) {
    auto size = ExtractInt32NoHeader(stream, iter);
    auto sliceType = static_cast<digo_type>(ExtractInt32NoHeader(stream,
iter));
    vector<TypeCell> result;
    for (int32_t i = 0; i < size; i++) {
        auto type = (digo_type)ExtractInt32NoHeader(stream, iter);
        switch (type) {
            case TYPE_STR: {
                auto str = (ExtractStringNoHeader(stream, iter));
                result.emplace_back(str);
                break;
            }
            case TYPE_INT32: {
                auto num = ExtractInt32NoHeader(stream, iter);
                result.emplace_back(num);
                break;
            }
            case TYPE_INT64: {
                auto num = ExtractInt64NoHeader(stream, iter);
                result.emplace_back(num);
                break;
            }
            case TYPE_DOUBLE: {
                auto num = ExtractDoubleNoHeader(stream, iter);
                result.emplace_back(num);
```

```
                break;
            }
            case TYPE_SLICE: {
                throw NotSerializableException("are you sure to support
nested slice? (ext)");
            }
            case TYPE_FUTURE_OBJ:
                throw NotSerializableException("future_obj cannot be
extracted");
            case TYPE_UNDEFINED:
                throw NotSerializableException("type undefined (ext)");
        }
    }
    return std::make_tuple(result, sliceType);
}

void Serialization::AddString(const string & str) {
    AddHeader(TYPE_STR);
    AddInt32NoHeader(str.size());
    for (int i = 0; i < str.size(); i++) {
        AddToEnd({static_cast<byte>(str[i])});
    }
}

string Serialization::ExtractStringNoHeader(vector<byte> & stream, int
*iter) {
    int32_t cnt = ExtractInt32NoHeader(stream, iter);
    string ret;
    for (int i = 0; i < cnt; i++) {
        ret += stream.at(*iter);
        (*iter)++;
    }
    return ret;
}

void Serialization::AddInt32(int32_t num) {
    AddHeader(TYPE_INT32);
    AddInt32NoHeader(num);
}

void Serialization::AddInt64(int64_t num) {
    AddHeader(TYPE_INT64);
    AddInt64NoHeader(num);
```

```
}

void Serialization::AddDouble(double num) {
    AddHeader(TYPE_DOUBLE);
    byte* ptr = (byte*)(&num);
    for (int i = 0; i < sizeof(double); i++) {
        AddToEnd({ptr[i]});
    }
};

double Serialization::ExtractDoubleNoHeader(vector<byte> & stream, int
*iter) {
    double result;
    byte* ptr = (byte*)(&result);
    for (int i = 0; i < sizeof(double); i++) {
        ptr[i] = stream.at((*iter));
        (*iter)++;
    }
    return result;
}

void Serialization::AddHeader(digo_type type) {
    AddInt32NoHeader(type);
    // AddToEnd(PaddingFront({(byte)type}, 2));
}

vector<byte> && Serialization::PaddingFront(vector<byte> && bytes, int
length) {
    int n = bytes.size();
    if (n % length != 0) {
        bytes.insert(bytes.begin(), n % length, 0);
    }
    return std::move(bytes);
}

void Serialization::AddToEnd(vector<byte> && in) {
    content_.insert(content_.end(), in.begin(), in.end());
}

void Serialization::AddInt32NoHeader(int32_t num) {
    for (int i = 0; i < 4; i++) {
        AddToEnd({static_cast<byte>((num >> ((3-i) * 8)) & 0xff)});
    }
```

```
}

int32_t Serialization::ExtractInt32NoHeader(vector<byte> & stream, int
*iter) {
    int32_t num = 0;
    for (int i = 0; i < 4; i++) {
        num = (num << 8) + stream.at((*iter));
        (*iter)++;
    }
    return num;
}

void Serialization::AddInt64NoHeader(int64_t num) {
    for (int i = 0; i < 8; i++) {
        AddToEnd({static_cast<byte>((num >> ((7-i) * 8)) & 0xff)});
    }
}

int64_t Serialization::ExtractInt64NoHeader(vector<byte> & stream, int
*iter) {
    int64_t num = 0;
    for (int i = 0; i < 8; i++) {
        num = (num << 8) + stream.at(*iter);
        (*iter)++;
    }
    return num;
}

vector<byte> Serialization::Get() {
    return content_;
}

byte *Serialization::GetBytes() {
    byte* bytes = new byte[content_.size()];
    int i = 0;
    for (byte b : content_) {
        bytes[i++] = b;
    }
    return bytes;
}

int Serialization::GetSize() {
    return content_.size();
```

```
}
```

9.4.12 serialization.h

```
/* See serialization.cpp for details.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#ifndef DIGO_LINKER_SERIALIZATION_H
#define DIGO_LINKER_SERIALIZATION_H

#include "common.h"

#include <utility>
#include <vector>
#include <exception>

using std::exception;
using std::vector;

class ExtractionResult {
public:
    ExtractionResult() = default;
    int err_number = 0;
    string err_info;

    vector<TypeCell> extracted_cells;

};

class Serialization {
public:
    Serialization();
    void Extract(vector<byte> & stream);
    void AddString(const string &);
    void AddInt32(int32_t);
    void AddInt64(int64_t);
    void AddDouble(double);
    void AddSlice(const vector<TypeCell> & arr, digo_type sliceType);
    void AddSlice(const vector<TypeCell> && arr, digo_type sliceType);
```

```
    vector<byte> Get();

    void Extract(byte* stream, int len);

    const ExtractionResult& GetResult() {
        return extraction_result_;
    }

    TypeCell ExtractOne();

    byte* GetBytes();
    int GetSize();

protected:
    void AddHeader(digo_type);
    vector<byte> && PaddingFront(vector<byte> &&, int length);
    void AddToEnd(vector<byte> &&);
    void AddInt32NoHeader(int32_t);
    void AddInt64NoHeader(int64_t);
    string ExtractStringNoHeader(vector<byte> &, int *iter);
    int32_t ExtractInt32NoHeader(vector<byte> &, int *iter);
    int64_t ExtractInt64NoHeader(vector<byte> &, int *iter);
    double ExtractDoubleNoHeader(vector<byte> &, int *iter);

    std::tuple<vector<TypeCell>, digo_type>
ExtractSliceNoHeader(vector<byte> &, int *iter);

    vector<byte> content_;

    ExtractionResult extraction_result_;

    int extraction_ptr_ = 0;

};

class EmptyStreamException: public exception {
public:
    const char * what() const noexcept override {
        return "empty stream";
    }
};

class ExtractionWrongIndexException: public exception {
```

```
public:
    ExtractionWrongIndexException(int i, int len) {
        msg = "Index: ";
        msg += std::to_string(i);
        msg += ", out of bound ";
        msg += std::to_string(len);
    }
    const char * what() const noexcept override {
        return msg.c_str();
    }
private:
    string msg;
};

class NotSerializableException: public exception {
public:
    NotSerializableException(string err) {
        msg = std::move(err);
    }
    const char * what() const noexcept override {
        return msg.c_str();
    }
private:
    string msg;
};

class NullPointerException: public exception {
public:
    NullPointerException(string err) {
        msg = std::move(err);
    }
    const char * what() const noexcept override {
        return msg.c_str();
    }
private:
    string msg;
};

#endif //DIGO_LINKER_SERIALIZATION_H
```

9.4.13 serialization_wrapper.cpp

```
/* The LLVM IR boundary wrapper for the Digo Serialization Library.
 * It provides serialization interface for each Digo type.
 * Compiler does not use these interfaces directly.
 * Instead, the details of serialization is hidden in the interfaces
provided by Digo Linker.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#include <unistd.h>
#include "serialization.h"
#include "serialization_wrapper.h"

#include "gc.h"
#include "builtin_types.h"

static void SerializationExceptionHandler(const std::string& func,
std::exception & e) noexcept {
    std::cerr << "Serialization Error: wrapper exception: " <<
typeid(e).name() << " " << e.what() << " caught in function " << func <<
std::endl;
    sleep(5);
    exit(1);
}

static void ExtractorExceptionHandler(const std::string& func,
std::exception & e) noexcept {
    std::cerr << "Serialization Error: extractor exception: " <<
typeid(e).name() << " " << e.what() << " caught in function " << func  <<
std::endl;
    sleep(5);
    exit(1);
}

void* SW_CreateWrapper() {
    try {
        return (void*)new Serialization();
    }
    catch (std::exception & e) {
        SerializationExceptionHandler("SW_CreateWrapper", e);
```

```
    }
    return nullptr;
}

void SW_AddInt32(void* w, int32_t n) {
    try {
        ((Serialization *) w)->AddInt32(n);
    }
    catch (std::exception & e) {
        SerializationExceptionHandler("SW_AddInt32", e);
    }
}

void SW_AddInt64(void* w, int64_t n) {
    try {
        ((Serialization*)w)->AddInt64(n);
    }
    catch (std::exception & e) {
        SerializationExceptionHandler("SW_AddInt64", e);
    }
}

void SW_AddDouble(void* w, double n) {
    try {
        ((Serialization*)w)->AddDouble(n);
    }
    catch (std::exception & e) {
        SerializationExceptionHandler("SW_AddDouble", e);
    }
}

void SW_AddString(void* wi, void* wj) {
    Serialization * w = (Serialization*)wi;
    DigoString * strWrapper = (DigoString*)wj;
    try {
        /*   unwrap here  */
        w->AddString(strWrapper->Data());
    }
    catch (std::exception & e) {
        SerializationExceptionHandler("SW_AddString", e);
    }
}
```

```
void SW_AddSlice(void* wi, void* wj) {
    Serialization * w = (Serialization*)wi;
    DigoSlice * sliceWrapper = (DigoSlice*)wj;
    try {
        if (sliceWrapper == nullptr) {
            // throw NullPointerException("Add a null slice");
            fprintf(stderr, "adding a null slice, seg fault !\n");
        }
        /*   unwrap here  */
        /*   generates an array with necessary data only   */
        vector<TypeCell> arr;
        auto [underlying_arr, begin, end] = sliceWrapper->Data();
        digo_type sliceType = sliceWrapper->Type();
        arr.reserve(end - begin);
        for (auto i = begin; i < end; i++) {
            if (i < 0 || i >= underlying_arr.size()) {
                printf("%lu in %lu:%lu out of range!!\n", i, begin, end);
            }
            TypeCell cell = underlying_arr.at(i);
            if (sliceType == TYPE_STR) {
                if (cell.str_obj == nullptr) {
                    cell.str = "";
                } else {
                    cell.str = ((DigoString*)(cell.str_obj))->Data();
                }
            }
            cell.type = sliceType;
            arr.push_back(cell);
        }
        w->AddSlice(arr, sliceType);
    }
    catch (std::exception & e) {
        SerializationExceptionHandler("SW_AddSlice", e);
    }
}

void SW_GetAndDestroy(void* w, byte** out_bytes, int32_t* out_length) {
    try {
        *out_bytes = ((Serialization*)w)->GetBytes();
        *out_length = ((Serialization*)w)->GetSize();
        delete (Serialization*)w;
    }
    catch (std::exception & e) {
```

```
        SerializationExceptionHandler("SW_GetAndDestroy", e);
    }
}

void SW_FreeArray(const byte* b) {
    // delete never throws exceptions
    delete[] b;
}

void* SW_CreateExtractor(byte* stream, int len) {
    try {

        auto s = new Serialization();
        s->Extract(stream, len);

        return s;
    }
    catch (std::exception & e) {
        ExtractorExceptionHandler("SW_CreateExtractor", e);
    }
    return nullptr;
}

int32_t SW_ExtractInt32(void* s) {
    try {
        auto se = (Serialization*)s;
        auto cell = se->ExtractOne();
        if (cell.type != TYPE_INT32) {
            printf("Wrong extraction type\n");
            return -1;
        }
        return cell.num32;
    }
    catch (std::exception & e) {
        ExtractorExceptionHandler("SW_ExtractInt32", e);
    }
    return -1;
}

int64_t SW_ExtractInt64(void* s) {
    try {
        auto se = (Serialization*)s;
        auto cell = se->ExtractOne();
```

```
        if (cell.type != TYPE_INT64) {
            printf("Wrong extraction type\n");
            return -1;
        }
        return cell.num64;
    }
    catch (std::exception & e) {
        ExtractorExceptionHandler("SW_ExtractInt64", e);
    }
    return -1;
}

double SW_ExtractDouble(void* s) {
    try {
        auto se = (Serialization*)s;
        auto cell = se->ExtractOne();
        if (cell.type != TYPE_DOUBLE) {
            printf("Wrong extraction type\n");
            return -1;
        }
        return cell.num_double;
    }
    catch (std::exception & e) {
        ExtractorExceptionHandler("SW_ExtractDouble", e);
    }
    return -1;
}

void* SW_ExtractString(void* s) {
    try {
        auto se = (Serialization*)s;
        auto cell = se->ExtractOne();
        if (cell.type != TYPE_STR) {
            printf("Wrong extraction type\n");
            return nullptr;
        }
        return CreateString(cell.str.c_str());
    }
    catch (std::exception & e) {
        ExtractorExceptionHandler("SW_ExtractString", e);
    }
    return nullptr;
}
```

```
void* SW_ExtractSlice(void* s) {
    try {
        auto se = (Serialization*)s;
        auto cell = se->ExtractOne();
        if (cell.type != TYPE_SLICE) {
            printf("Wrong extraction type\n");
            return nullptr;
        }
        /*  wraps it to Digo Object  */
        auto sliceObj = new DigoSlice(cell.arr_slice_type);
        /*  directly set data  */
        auto [underlying_arr, begin, end] = sliceObj->Data();
        begin = 0; end = 0;
        for (int i = 0; i < (int)cell.arr.size(); i++) {
            auto nested_cell = cell.arr.at(i);
            if (cell.arr_slice_type == TYPE_STR) {
                /* wraps nested cell to Digo Object */
                nested_cell.str_obj =
CreateString(nested_cell.str.c_str());
                nested_cell.str = "";
            }
            nested_cell.type = cell.arr_slice_type;
            underlying_arr.push_back(nested_cell);
            end++;
        }
        return sliceObj;
    }
    catch (std::exception & e) {
        ExtractorExceptionHandler("SW_ExtractString", e);
    }
    return nullptr;
}

void SW_DestroyExtractor(void* s) {
    auto se = (Serialization*)s;
    delete se;
}
```

9.4.14 serialization_wrapper.h

```
/* The interface definition for the Digo Serialization Library.
 * It provides serialization interface for each Digo type.
 * Compiler does not use these interfaces directly.
 * Instead, the details of serialization is hidden in the interfaces
provided by Digo Linker.
 *
 * Author: sh4081
 * Date: 2021/3/21
 */

#ifndef DIGO_LINKER_SERIALIZATION_WRAPPER_H
#define DIGO_LINKER_SERIALIZATION_WRAPPER_H

#include <cstdint>
typedef unsigned char byte;

extern "C" {
    void* SW_CreateWrapper();
    void SW_AddInt32(void*, int32_t);
    void SW_AddInt64(void*, int64_t);
    void SW_AddDouble(void*, double);
    void SW_AddString(void*, void* strWrapper);
    void SW_AddSlice(void*, void*);
    void SW_GetAndDestroy(void* w, byte** out_bytes, int32_t* out_length);
    void SW_FreeArray(const byte*);

    void* SW_CreateExtractor(byte*, int);
    int32_t SW_ExtractInt32(void*);
    int64_t SW_ExtractInt64(void*);
    double SW_ExtractDouble(void*);
    void* SW_ExtractString(void*);
    void* SW_ExtractSlice(void*);
    void SW_DestroyExtractor(void*);
}

#endif //DIGO_LINKER_SERIALIZATION_WRAPPER_H
```

9.4.15 wrapper.cpp

```
/* This file provides async API - LLVM IR boundaries.
 * However, these interfaces are not directly used by Compiler.
 * Instead, the Digo Linker uses these APIs and provides simpler
 * interfaces for the Compiler and Async Library.
 *
 * This file also provides an entry for the whole program.
 *
 * Author: sh4081
 * Date: 2021/3/25
 */

#include "../../async-remote-lib/src/async.h"
#include "../../async-remote-lib/src/master_worker.h"

#include "wrapper.h"
#include "gc.h"
#include "serialization_wrapper.h"

#include <memory>
#include <unordered_map>
#include <string>
#include <utility>
#include <unistd.h>

using std::unordered_map;
using std::string;

class DigoFuture : public DObject {
public:
    explicit DigoFuture(shared_ptr<Async> ptr) : p(std::move(ptr)) {}
    shared_ptr<Async> Get() {
        return p;
    }
    const char* name() override {
        return "Future Object";
    }
private:
    shared_ptr<Async> p;
};

unordered_map<int, string> ASYNC_FUNC_ID2NAME;
```

```
unordered_map<string, int> ASYNC_FUNC_NAME2ID;

static void WrapperExceptionHandler(const std::string& func, std::exception
& e) noexcept {
    std::cerr << "Library Wrapper Error: exception: " << typeid(e).name()
<< " " << e.what() << " caught in function " << func << std::endl;
    sleep(5);
    exit(1);
}

struct RuntimeInfo {
    shared_ptr<Master> master;
};

RuntimeInfo g_runtime_info;

__attribute__((noinline)) int entry(int argc, char* argv[]) {
    string usage = "usage: --worker MasterIP:MasterPort
WorkerIP:WorkerPort\n   or --master ip:port\n";
    try {
        if (argc < 3) {
            if (argc == 2) {
                if(string(argv[1]) == "--no-master") {
                    return 1;
                }
            }
            std::cerr << "Wrong arguments, " + usage;
            exit(1);
        }
        if (string(argv[1]) == "--master") {
            if (argc != 3) {
                std::cerr << "Wrong master arguments, " + usage;
                exit(1);
            }
            auto master = Master::GetInst();
            /*  master listens for new workers in another thread  */
            std::thread([=]{master->Listen(argv[2]);}).detach();
            master->WaitForReady();
            cerr << string("Debug Info: Master ready on ") + argv[2] +
"\n";
            g_runtime_info.master = master;
            /*  here the Entry() returns 1 indicating that
             *  the argument is --master, and
```

```
             *  control flow will go to digo_main() defined by Digo
Compiler
             */
            return 1;
        } else if (string(argv[1]) == "--worker") {
            if (argc != 4) {
                std::cerr << "Wrong worker arguments, " + usage;
                exit(1);
            }
            auto worker = Worker::GetInst();
            /*  worker has to block here  */
            cerr << string("Debug Info: Worker ready on ") + argv[3] + ",
master: "
                    + argv[2] + ", my pid: " + to_string(getpid())  + "\n";
            std::thread([=]{ worker->Start(argv[2], argv[3]);})
            .join();
            return 2;
        } else {
            std::cerr << "--master/--worker expected, " + usage;
            exit(1);
        }
    }
    catch (std::exception & e) {
        WrapperExceptionHandler("Entry", e);
    }
    exit(1);
    /*  unreachable  */
    return 0;
}

__attribute__((noinline)) void __DIGO_RUNTIME_OnExit() {
    // if (g_runtime_info.master)
        // g_runtime_info.master->StopListen();
    __GC_DEBUG_COLLECT_LEAK_INFO();
}

__attribute__((noinline)) void* CreateAsyncJob(int32 func, byte* args,
int32 arg_len) {
    try {
        auto future_obj = Async::CreateLocal(ASYNC_FUNC_ID2NAME[func],
                  bytes{.content=shared_ptr<byte>(args), .length=arg_len});
        return new DigoFuture(future_obj);
    }
```

```
    catch (std::exception & e) {
        WrapperExceptionHandler("CreateAsyncJob", e);
    }
    return nullptr;
}

__attribute__((noinline)) void* CreateRemoteJob(int32 func, byte* args,
int32 arg_len) {
    try {
        auto future_obj = Async::CreateRemote(ASYNC_FUNC_ID2NAME[func],
                  bytes{.content=shared_ptr<byte>(args), .length=arg_len});
        return new DigoFuture(future_obj);
    }
    catch (std::exception & e) {
        WrapperExceptionHandler("CreateRemoteJob", e);
    }
    return nullptr;
}

__attribute__((noinline)) void AwaitJob(void* future_obj, byte** result,
int32* len) {
    try {
        if (future_obj == nullptr) {
            throw std::bad_function_call();
        }
        auto r = ((DigoFuture*)future_obj)->Get()->Await();
        // the result will not be deconstructed because
        // we have a reference in Async->result_
        *result = r.content.get();
        *len = r.length;
    }
    catch (std::exception & e) {
        WrapperExceptionHandler("AwaitJob", e);
    }
}

__attribute__((noinline)) void ASYNC_AddFunction(int32 id, char* func_name)
{
    try {
        cerr << "Async function " << id << ": " << func_name << " is
added\n";
        ASYNC_FUNC_ID2NAME[id] = string(func_name);
        ASYNC_FUNC_NAME2ID[string(func_name)] = id;
```

```
    }
    catch (std::exception & e) {
        WrapperExceptionHandler("ASYNC_AddFunction", e);
    }
}

__attribute__((noinline)) const char* ASYNC_GetFunctionName(int32 id) {
    try {
        return ASYNC_FUNC_ID2NAME[id].c_str();
    }
    catch (std::exception & e) {
        WrapperExceptionHandler("ASYNC_GetFunctionName", e);
    }
    return nullptr;
}

__attribute__((noinline)) int32 ASYNC_GetFunctionId(const char* func_name)
{
    try {
        return ASYNC_FUNC_NAME2ID[func_name];
    }
    catch (std::exception & e) {
        WrapperExceptionHandler("ASYNC_GetFunctionId", e);
    }
    return -1;
}

__attribute__((noinline)) void Debug_Real_LinkerCallFunction(int32_t id,
int32_t arg_len) {
    cerr << "Digo Linker Info: the digo linker is calling func with id " +
to_string(id) + " and arg len "
     + to_string(arg_len) + "\n";
}

__attribute__((noinline)) void NoMatchExceptionHandler(int32_t id) {
    cerr << "Digo Linker Error: linker call function, NoMatchException, " +
to_string(id) + " not valid\n";
    sleep(5);
    exit(1);
}
```

9.4.16 wrapper.h

```
/* See wrapper.cpp for description.
 *
 * Author: sh4081
 * Date: 2021/3/25
 */

#ifndef ASYNC_REMOTE_LIB_WRAPPER_H
#define ASYNC_REMOTE_LIB_WRAPPER_H

typedef int32_t int32;

extern "C" {

__attribute__((noinline)) int entry(int argc, char* argv[]);

__attribute__((noinline)) void* CreateAsyncJob(int32, byte*, int32);
__attribute__((noinline)) void* CreateRemoteJob(int32, byte*, int32);
__attribute__((noinline)) void AwaitJob(void*, byte**, int32*);

// we maintain a name->id map and an id->name map in memory
__attribute__((noinline)) void ASYNC_AddFunction(int32 id, char*
func_name);
__attribute__((noinline)) const char* ASYNC_GetFunctionName(int32 id);
__attribute__((noinline)) int32 ASYNC_GetFunctionId(const char* func_name);

__attribute__((noinline)) void Debug_Real_LinkerCallFunction(int32_t id,
int32_t arg_len);
__attribute__((noinline)) void NoMatchExceptionHandler(int32_t id);

__attribute__((noinline)) void __DIGO_RUNTIME_OnExit();
}

#endif //ASYNC_REMOTE_LIB_WRAPPER_H
```

9.4.17 Makefile

```
all: build-all
```

```
.PHONY: build-all
build-all: clean generate-digo-dependency generate-digo-linker

.PHONY: generate-digo-dependency
generate-digo-dependency:
	clang -stdlib=libc++ -std=c++2a -O3 -S -emit-llvm
src/serialization.cpp src/wrapper.cpp src/serialization_wrapper.cpp
src/print_funcs.cpp src/gc.cpp
	llvm-link -S -v -o allinone.ll serialization.ll wrapper.ll
serialization_wrapper.ll print_funcs.ll gc.ll

.PHONY: generate-digo-linker
generate-digo-linker:
	clang -stdlib=libc++ -lstdc++ -std=c++2a -O3 src/metadata.cpp
src/linker_main.cpp -o digo-linker

.PHONY: clean
clean:
	rm -f serialization.ll wrapper.ll serialization_wrapper.ll
print_funcs.ll gc.ll
	rm -f allinone.ll digo-linker
```

9.5 digo-test

Everyone in the team contributed to test cases.
Hanxiao split test cases into multiple categories, and Sida as well as Yufan did the test script.
Yufan and Wenqian brought up the automated test build with circle-ci.

9.5.1 testall.sh

```
#!/bin/bash

# Testing script for Digo

# Reference: the regression testing script in MicroC

# This script first generates the Digo Library, Digo Linker, and Digo
Compiler,
```

```
#    if the Digo Compiler does not exist.

# then it goes through a list of test groups,
# for each test group, we have a config.txt file which tells the script
that
#     how many workers the tests in this group need;
#     also, some options like GC_DEBUG can be enabled in this config.txt.
#    all tests in the same test group will use the same config.
# for each test,
#    compile, run, and check the output.
#    If the compilation fails, it will diff
#       the error reported by the compiler and the
$source_code.fail.expected
#    If the compilation succeeds, it will diff
#       the result produced by the executable and the
$source_code.pass.expected

Usage() {
    echo "Usage: testall.sh [options] [.digo files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

MAKE_DIR='../'
# Generate depenencies and Digo Compiler
BuildCompiler() {
    echo "------------------ !Compiler not found ------------------------"
    echo "------------------ Generating Compiler ------------------------"
    echo "Please wait..... It may take 1-5 minutes to generate compiler"

    if ! make -C $MAKE_DIR clean &>/dev/null; then
      echo "clean failed "
      exit 1
    fi

    if ! make -C $MAKE_DIR generate-dependency &>/dev/null;
    then
      echo "build dependency failed"
      exit 1
    fi

    echo "Library compilation succeeds, now generating OCaml compiler..."
```

```
    if ! make -C $MAKE_DIR generate-digo-compiler &>/dev/null; then
      echo "build compiler failed"
      exit 1
    fi

    echo "------------------ Compiler Generated ------------------------"
}

global_test_error=0
global_test_error_any=0

global_log="testall.log"
global_llvm_ir_output_log="testall_ir.log"
rm -f $global_log
rm -f $global_llvm_ir_output_log

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile. Differences, if any, written to
difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 &>>"$global_log"
    diff -b "$1" "$2" > "$3" 2>&1 || {
        global_test_error=1
        echo "FAILED $1 differs from $2" >> "$global_log"
    }
}

WORKER_COUNT=0
GC_DEBUG=0
ENABLE_MASTER=0
MASTER_ADDR=()
WORKER_ADDR=()

LoadDefaultConfig() {
    WORKER_COUNT=0
    GC_DEBUG=1
    ENABLE_MASTER=0
    MASTER_ADDR=()
    WORKER_ADDR=()
}

# RunTest <digofile>
```

```
#    Compile, run, and check the output.
#    If the compilation fails, it will diff
#       the error reported by the compiler and the
$source_code.fail.expected
#    If the compilation succeeds, it will diff
#       the result produced by the executable and the
$source_code.pass.expected
RunTest() {
    global_test_error=0
    test_name=`echo $1 | sed 's/.*\\///
                             s/.digo//'`
    test_src="$1"

    dir_name=`dirname $test_src`

    echo -n "$test_name..."

    rm -f "../executable"

    ../digo-compiler/digo.native
"$test_src" > ../tmp.compiled.nometadata.ll 2>"$test_name.build.output"
    errorlevel=$?

    echo "" >> "$global_llvm_ir_output_log"
    echo "# -------------------- Testing $test_name" >>
"$global_llvm_ir_output_log"
    cat ../tmp.compiled.nometadata.ll >> $global_llvm_ir_output_log

    if [ $errorlevel -eq 0 ] ; then
        make -C $MAKE_DIR build-link-pass digo="$test_src" out=executable
&>"$test_name.linker.output"
        errorlevel=$?
    fi

    echo "" >> "$global_log"
    echo "# -------------------- Testing $test_name --------------------
" >> "$global_log"

    # load group config
    if [ -f "$dir_name/config.txt" ] ; then
        . "$dir_name/config.txt"
    fi
```

```
    # config override
    if [ -f "$test_src.cfg" ] ; then
        . "$test_src.cfg"
    fi

    echo "Config: WORKER_COUNT=$WORKER_COUNT; GC_DEBUG=$GC_DEBUG;
ENABLE_MASTER=$ENABLE_MASTER; " >> "$global_log"
    echo "        MASTER_ADDR=${MASTER_ADDR[*]};
WORKER_ADDR=${WORKER_ADDR[*]}" >> "$global_log"

    diff_output_file="$test_name.diff.output"

    if [ ! -f "../executable" ]; then
        errorlevel=1
    fi

    if [ $errorlevel -eq 0 ] ; then
        # success expected, so try to run the executable
        exec_output="$test_name.exec.output"
        exec_debug_output="$test_name.debug.output"
        expected_file="$test_src.pass.expected"

        if [ $ENABLE_MASTER -eq 0 ] ; then
            eval '../executable --no-master > "$exec_output"
2>"$exec_debug_output" &'
            master_pid=$!
        else
            # echo ../executable --master $MASTER_ADDR
            eval '../executable --master $MASTER_ADDR > "$exec_output"
2>"$exec_debug_output" &'
            master_pid=$!
        fi

        worker_pid=()
        if [ $WORKER_COUNT -gt 0 ] ; then
            for (( i = 0 ; i < $WORKER_COUNT ; i++ ))
            do
                command="../executable --worker $MASTER_ADDR
${WORKER_ADDR[$i]} &> \"$test_name.worker$i.output\" &"
                echo "Running worker $i: $command" >> "$global_log"
                pid=`(eval $command ; echo $!) 2>/dev/null`
                worker_pid[$i]=$pid
            done
```

```
        fi

        wait $master_pid

        kill -9 $master_pid &> /dev/null

        for pid in ${worker_pid[*]}; do
            kill -9 $pid &> /dev/null
        done

        echo " ## Executable output: " >> "$global_log"
        cat "$exec_output" >> "$global_log"

        echo " ## Executable debug output: " >> "$global_log"
        cat "$exec_debug_output" >> "$global_log"

        if [ $GC_DEBUG -eq 1 ] ; then
            if grep -q "GC Leak:" "$exec_debug_output"; then
                echo " ### FAIL: GC Leak" >> "$global_log"
                global_test_error=1
            else
                echo " ### GC Check Passed" >> "$global_log"
            fi
        fi

        if [ $WORKER_COUNT -gt 0 ] ; then
            for (( i = 0 ; i < $WORKER_COUNT ; i++ ))
            do
                echo "" >> "$global_log"
                echo " ## Worker $i stdout&stderr output: " >>
"$global_log"
                cat "$test_name.worker$i.output" >> "$global_log"
                worker_pid[$i]=$!
            done
        fi

        if [ ! -f $expected_file ]; then
            echo " ### Compilation passed, but we cannot find
$expected_file" >> "$global_log"
            global_test_error=1
        else
            echo " ## Diff output: " >> "$global_log"
            Compare "$exec_output" "$expected_file" "$diff_output_file"
```

```
            cat "$diff_output_file" >> "$global_log"
        fi
    else
        # fail expected, so try to diff the fail file
        expected_file="$test_src.fail.expected"
        build_output="$test_name.build.output"

        echo " ## Build output: " >> "$global_log"
        cat "$build_output" >> "$global_log"

        if [ ! -f $expected_file ]; then
            echo " ### Compilation failed, but we cannot find
$expected_file" >> "$global_log"
            global_test_error=1
        else
            Compare "$build_output" "$expected_file" "$diff_output_file"

            echo " ## Diff output: " >> "$global_log"
            cat "$diff_output_file" >> "$global_log"
        fi
    fi

    if [ $global_test_error -eq 1 ] ; then
        global_test_error_any=1
    fi

    if [ $global_test_error -eq 1 ] ; then
        echo "### Test $test_name: Failed" >> "$global_log"
        echo "FAIL"
    else
        echo "### Test $test_name: Passed" >> "$global_log"
        echo "OK"
    fi

    echo "" >> "$global_log"

}

RunTestFiles() {
    test_files=$@
    for filename in $test_files; do
        RunTest "$(pwd)/$filename"
```

```
    done
}

# RunTestGroup <test_group_dir>
RunTestGroup() {
    echo "# --- Test Group: $1"

    echo "# ------------------------- Test Group: $1 ---------------------
-" >> "$global_log"

    group_dir=$1
    LoadDefaultConfig
    # load config

    . "$group_dir/config.txt"

    RunTestFiles "$group_dir/*.digo"

    echo "" >> "$global_log"
    echo "" >> "$global_log"
}

Clean() {
    # make -C $MAKE_DIR clean &>/dev/null
    rm -f *.build.output
    rm -f *.exec.output
    rm -f *.diff.output
    rm -f *.linker.output
    rm -f *.worker*.output
    rm -f *.debug.output
}

# Set time limit for all operations
ulimit -t 30
keep=0

while getopts kdpsh c; do
    case $c in
      k) # Keep intermediate files
          keep=1
          ;;
      h) # Help
          Usage
```

```
            ;;
    esac
done

shift `expr $OPTIND - 1`


if [ ! -f ../digo-compiler/digo.native ]; then
    BuildCompiler
fi


if [ $# -ge 1 ]
then
    RunTestFiles $@
else
    RunTestGroup "Basic"
    RunTestGroup "Async"
    RunTestGroup "Syntax"
    RunTestGroup "Semantic"
    RunTestGroup "ControlFlow"
    RunTestGroup "GC"
    RunTestGroup "Remote"
    RunTestGroup "Utils"
fi


if [ $keep -eq 0 ] ; then
    echo "Cleaning..."
    Clean
fi

echo "---------"

if [ $global_test_error_any -eq 1 ] ; then
    echo "Some tests failed, see $global_log for details"
else
    echo "All tests passed"
fi

exit $global_test_error_any
```

9.5.2 Async

test-future_slice.digo

```
func digo_main() void {
    s := []future{}
    a := 10
    b := 20
    c := try_return_one(a)
    e := try_return_one(b)
    s = append(s, c)
    s = append(s, e)

    fu := s[1]
    g := await fu

    println("%d", g)

    fu = s[0]
    g1 := await fu
    println("%d", g1)

}

async func try_return_one(a int) int {
    return a+10
}
```

test-future_slice.digo.pass.expected

```
30
20
```

test-local.digo

```
func digo_main() void {
    a := 10
    b := 1.0
    c := try_return_one(a)
    e := try_return_two(a,b)
    f, g := await e
```

```
    println("%d", f)
    println("%f", g)
}

async func try_return_one(a int) int {
    return a+10
}

async func try_return_two(a int,b float) (int,float) {

    return a+10,b+20.0
}
```

test-local.digo.pass.expected

```
20
21.000000
```

9.5.3 Basic

test-add1.digo

```
func add(x int, y int) int {
  return x + y
}

func digo_main() void {
  println("%d", add(17, 25))
}
```

test-add1.digo.pass.expected

```
42
```

test-arith1.digo

```
func digo_main() void {
  println("%d", 39 + 3)
}
```

test-arith1.digo.pass.expected

```
42
```

test-arith2.digo

```
func digo_main() void {
  println("%d", 1 + 2 * 3 + 4)
}
```

test-arith2.digo.pass.expected

```
11
```

test-arith3.digo

```
func foo(a int) int {
  return a
}

func digo_main() void {
  a := 42
  a = a + 5
  println("%d", a)
}
```

test-arith3.digo.pass.expected

```
47
```

test-fib.digo

```
func fib(x int) int {
  if (x < 2) {
   return 1
  }
  return fib(x-1) + fib(x-2)
}

func digo_main() void {
  println("%d", fib(0))
  println("%d", fib(1))
  println("%d", fib(2))
```

```
  println("%d", fib(3))
  println("%d", fib(4))
  println("%d", fib(5))
}
```

test-fib.digo.pass.expected

```
1
1
2
3
5
8
```

test-float1.digo

```
func digo_main() void {
  a := 3.14159267
  println("%f", a)
}
```

test-float1.digo.pass.expected

```
3.141593
```

test-float2.digo

```
func digo_main() void {
  var a, b, c float
  a = 3.14159267
  b = -2.71828
  c = a + b
  println("%f", c)
}
```

Test-float2.digo.pass.expected

```
0.423313
```

test-float3.digo

```
func testfloat(a float, b float) void {
  println("%f", a + b)
  println("%f", a - b)
  println("%f", a * b)
  println("%f", a / b)
```

```
  println("%f", a == b)
  println("%f", a == a)
  println("%f", a != b)
  println("%f", a != a)
  println("%f", a > b)
  println("%f", a >= b)
  println("%f", a < b)
  println("%f", a <= b)
}

func digo_main() void {
  c := 42.0
  d := 3.14159

  testfloat(c, d)
  testfloat(d, d)
}
```

test-float3.digo.pass.expected

```
45.141590
38.858410
131.946780
13.369027
0.000000
1.000000
1.000000
0.000000
1.000000
1.000000
0.000000
0.000000
6.283180
0.000000
9.869588
1.000000
1.000000
1.000000
0.000000
0.000000
0.000000
1.000000
0.000000
```

```
1.000000
```

test-func1.digo

```
func add(a int, b int) int {
  return a + b
}

func digo_main() void {
  a := add(39, 3)
  println("%d", a)
}
```

test-func1.digo.pass.expected

```
42
```

test-func2.digo

```
func fun(x int, y int) int {
  return 0
}

func digo_main() void {
  var i int
  i = 1
  fun(i = 2, i = i+1)
  println("%d", i)
}
```

test-func2.digo.pass.expected

```
3
```

test-func3.digo

```
func printem(a int, b int, c int, d int) void {
  println("%d %d %d %d", a, b, c, d)
}

func digo_main() void {
  printem(42,17,192,8)
}
```

test-func3.digo.pass.expected

```
42 17 192 8
```

test-func4.digo

```
func add(a int, b int) int {
  c := a + b
  return c
}

func digo_main() void {
  var x int
  x = add(52, 10)

  var y int = add(52, 10)

  z := add(52, 10)

  println("%d", x)
  println("%d", y)
  println("%d", z)
  println("%d", add(52, 10))
}
```

test-func4.digo.pass.expected

```
62
62
62
62
```

test-func5.digo

```
func foo(a int) int {
  return a
}

func digo_main() void {
}
```

test-func5.digo.pass.expected

```

```

test-func6.digo

```
func foo() void {
}

func bar(a int, b bool, c int) int {
    return a + c
}

func digo_main() void {
  println("%d", bar(17, false, 25))
}
```

test-func6.digo.pass.expected

```
42
```

test-func8.digo

```
func foo(a int) void {
  println("%d", a + 3)
}

func digo_main() void {
  foo(40)
}
```

test-func8.digo.pass.expected

```
43
```

test-func9.digo

```
func foo(a int) void {
  println("%d", a + 3)
}

func digo_main() void {
  foo(40)
}
```

test-func9.digo.pass.expected

```
43
```

test-func10.digo

```
func foo(a int) {
```

```
  println("%d", a + 3)
}

func digo_main() void {
  foo(40)
}
```

test-func10.digo.pass.expected

```
43
```

test-gcd.digo

```
func gcd(a int, b int) int {
  for (;a != b;) {
    if (a > b) {
        a = a - b
    } else {
        b = b - a
    }
  }
  return a
}

func digo_main() void {
  println("%d", gcd(2,14))
  println("%d", gcd(3,15))
  println("%d", gcd(99,121))
}
```

test-gcd.digo.pass.expected

```
2
3
11
```

test-gcd2.digo

```
func gcd(a int, b int) int {
  for (;a != b;) {
    if (a > b) {
     a = a - b
    } else {
     b = b - a
    }
```

```
  }
  return a
}

func digo_main() void {
  println("%d", gcd(14,21))
  println("%d", gcd(8,36))
  println("%d", gcd(99,121))
}
```

test-gcd2.digo.pass.expected

```
7
4
11
```

test-local1.digo

```
func foo(a int, b bool) int {
  c := a
  return c + 10
}

func digo_main() void {
 println("%d", foo(37, false))
}
```

test-local1.digo.pass.expected

```
47
```

test-ops1.digo

```
func digo_main() void {
  println("%d %d %d %d", 1 + 2, 1 - 2, 1 * 2, 100 /2)
  println("%d", 99)
  println("%d", 1 == 2)
  println("%d", 1 == 1)
  println("%d", 99)
  println("%d %d", 1 != 2, 1 != 1)
  println("%d %d %d %d", 1 < 2, 2 < 1, 1 <= 2, 1 <= 1, 2 <= 1)
}
```

test-ops1.digo.pass.expected

```
3 -1 2 50
99
0
1
99
1 0
1 0 1 1
```

test-ops2.digo

```
func digo_main() void {
  println("%d", true)
  println("%d", false)
  println("%d", true && true)
  println("%d", true && false)
  println("%d", false && true)
  println("%d", false && false)
  println("%d", true || true)
  println("%d", true || false)
  println("%d", false || true)
  println("%d", false || false)
  println("%d", !false)
  println("%d", !true)
  println("%d", -10)
  println("%d", 42)
}
```

test-ops2.digo.pass.expected

```
1
0
1
0
0
0
1
1
1
0
1
0
-10
42
```

test-return1.digo

```
func foo(x int) int {
  if (x == 1) {
    return 1
  }
  return 2
}

func digo_main() void {
  println("%d", foo(1))
}
```

test-return1.digo.pass.digo

```
1
```

test-return2.digo

```
func normal_function() int {
    a := 1
    b := 10.0
    if (a == 1) {
        c := try_return_one(a)
        return a

    } else if (b == 1.0) {
        e,f := try_return_two(a,b)
        return a

    } else {
        g02d := try_return_one(100)
    }
}

func try_return_one(a int) int {

    return a+10
}

func try_return_two(a int,b float) (int,float) {

    return a+10,b+20.0
```

```
}

func digo_main() void {
    println("%d", normal_function())
}
```

test-return2.digo.pass.expected

```
1
```

test-slice_func.digo

```
func foo(s []int) []int {
    s2 := append(s, 999)
    return s2
}

func digo_main() void {
  a := []int {1,2,3,4}
  b := foo(a)
  println("%d",len(a))
  println("%d",len(b))
}
```

test-slice_func.digo.pass.expected

```
4
5
```

test-slice_string_func.digo

```
func foo(s []string) []string {
    s2 := append(s, "haha")
    return s2
}

func digo_main() void {
  a := []string {"hello", "word"}
  b := foo(a)
  println("%d",len(a))
  println("%d",len(b))
  println("%s", a[1])
  println("%s", b[2])
}
```

test-slice_string_func.digo.pass.expected

```
2
3
word
haha
```

test-slice.digo

```
func digo_main() void {
  a := []int {1,2,3,4}
  b := append(a,10)
  c := len(b)
  var d []int
  d = b
  println("%d",b[0])
  println("%l",b[1:2])
  println("%d",c)
  println("%l",d)
  println("%l", b[:1])
  println("%l", b[2:])
}
```

test-slice.digo.pass.expected

```
1
[2]
5
[1, 2, 3, 4, 10]
[1]
[3, 4]
```

test-slice2.digo

```
func digo_main() void {
    s1 := []int{}
    s1 = append(s1, 1)
    var m int
    m = s1[0]
  println("%d", m)
}
```

test-slice2.digo.pass.expected

```
1
```

test-slice3.digo

```
func digo_main() void {
    s1 := []int{}
    var i int
    for (i = 0; i < 10; i = i + 1) {
        s1 = append(s1, i)
    }
    println("%l", s1)
    n := s1[1]
    println("%d", n)
}
```

test-slice3.digo.pass.expected

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
1
```

test-string_func.digo

```
func foo(s string) string {
    s = s + "haha"
    return s
}

func digo_main() void {
    s := "hello"
    s2 := foo(s)
      println("%s", s)
      println("%s", s2)
}
```

test-string_func.digo.pass.expected

```
hello
hellohaha
```

test-string.digo

```
func digo_main() void {
      a := "abc"
      var b string
```

```
      b = a
      c := len(b)
      d := a+b
      println("%d",c)
      println("%s",d)
      println("%d",a == d)
}
```

test-string.digo.pass.expected

```
3
abcabc
0
```

test-var_decl.digo

```
func digo_main() void {
    var a, b, c int = 3, 4, 5
    println("%d", a)
    println("%d", b)
    println("%d", c)
}
```

test-var_decl.digo.pass

```
3
4
5
```

test-var_short_decl.digo

```
func digo_main() void {
    a, b, c := 3, 4, 5
    println("%d", a)
    println("%d", b)
    println("%d", c)
}
```

test-var_short_decl.digo.pass.expected

```
3
4
5
```

9.5.4 ControlFlow

test-for1.digo

```
func digo_main() void {
  var i int
  for (i = 0 ; i < 5 ; i = i + 1) {
    println("%d", i)
  }
  println("%d", 42)
}
```

test-for1.digo.pass.expected

```
0
1
2
3
4
42
```

test-for2.digo

```
func digo_main() void {
  var i int
  i = 0
  for (;i < 5;) {
    println("%d", i)
    i = i + 1
  }
  println("%d", 42)
}
```

test-for2.digo.pass.expected

```
0
1
2
3
4
42
```

test-if1.digo

```
func digo_main() void {
```

```
  if (true) {
   println("%d", 42)
  }
  println("%d", 17)
}
```

Test-if1.digo.pass.expected

```
42
17
```

test-if2.digo

```
func digo_main() void {
  if (true) {
   println("%d", 42)
  } else {
    println("%d", 8)
  }
  println("%d", 17)
}
```

Test-if2.digo.pass.expected

```
42
17
```

test-if3.digo

```
func digo_main() void {
  if (false) {
   println("%d", 42)
  }
  println("%d", 17)
}
```

test-if3.digo.pass.expected

```
17
```

test-if4.digo

```
func digo_main() void {
  if (false) {
   println("42")
```

```
  } else {
   println("8")
  }
  println("%s", "17")
}
```

test-if4.digo.pass.expected

```
8
17
```

test-if5.digo

```
func cond(b bool) int {
  var x int
  if (b) {
    x = 42
  } else {
    x = 17
  }
  return x
}

func digo_main() void {
 println("%d", cond(true))
 println("%d", cond(false))
}
```

test-if5.digo.pass.expected

```
42
17
```

test-if6.digo

```
func cond(b bool) int {
  var x int
  x = 10
  if (b) {
    if (x == 10) {
      x = 42
    }
  } else {
    x = 17
```

```
  }
  return x
}

func digo_main() void {
 println("%d", cond(true))
 println("%d", cond(false))
}
```

test-if6.digo.pass.digo

```
42
17
```

9.5.5 GC

basic.digo

```
func digo_main() void {
    a := get_one_str()
    b := get_one_int()
    d1, d2, d3, d4, d5 := get_many()
    p1, p2, p3 := get_three2()
    println("%s %f %d", a + "343221" + p2, p3, p1)
}

func get_one_str() string {
    a := "2394239234"
    if (1 == 1) {
        return "12341235622" + a
    } else if (2==1) {
        return a + "123412334" + "222"
    } else {
        return "!234123"
    }

    return "invalid" + "12343" + "3"
}

func get_three2() (int,string,float) {

}
```

```
func get_many() (string, string, string, int, string) {
    return "12", "23", "454", 100, "4000"
}

func get_one_int() int {
    return 0
}
```

basic.digo.pass.expected

```
12341235622239423923434322 1 0.000000 0
```

future_slice_leak.digo

```
func digo_main() void {
    s := []future{}
    a := 10
    b := 20
    c := try_return_one(a)
    e := try_return_one(b)
    s = append(s, c)
    s = append(s, e)

    fu := s[1]
    g := await fu

    println("%d", g)

    fu = s[0]
    g1 := await fu
    println("%d", g1)

}

async func try_return_one(a int) int {
    return a+10
}
```

future_slice_leak.digo.pass.digo

```
30
20
```

null-slice.digo

```
func digo_main() void {
    var s1 []string
    var s2 []int

    println("%l %l", s1, s2)
}
```

null-slice.digo.pass.expected

```
[] []
```

null-slice2.digo

```
func digo_main() void {
    s1 := []string{}
    s2 := []int{}

    println("%l %l", s1, s2)
}
```

null-slice2.digo.pass.expected

```
[] []
```

remote.digo

```
func digo_main() void {
    a := []string{"1234", "4234fff", "da4dfa", "dfa4fda"}
    b := []string{"hedfa", "fdafedf", "fdafe", "hello"}
    c := []int{1, 2,3,4,5,6,7}

    future1 := f1(a, b, c)

    f, g, h := await future1

    println("%l", f)
    println("%l", g)
    println("%l", h)
}


async remote func f1(a []string, b []string, c []int) ([]string, []string,
```

```
[]int) {
    return b, a, c
}
```

remote.digo.pass.expected

```
[hedfa, fdafedf, fdafe, hello]
[1234, 4234fff, da4dfa, dfa4fda]
[1, 2, 3, 4, 5, 6, 7]
```

strinng_slice_leak.digo

```
func digo_main() void {
    s := []string{}

    c := "23123423"
    e := "324123423"
    s = append(s, c)
    s = append(s, e)

    fu := s[1]

    println("%s", fu)

    fu = s[0]
    println("%s", fu)

}
```

strinng_slice_leak.digo.pass.expected

```
324123423
23123423
```

9.5.6 Remote

test_remote_string.digo

```
func digo_main() void {
    a := "haha"
    b := "hehe"
    e := try_return_two(a,b)
    f, g := await e
```

```
    println("%s", f)
    println("%s", g)
}


async remote func try_return_two(a string,b string) (string,string) {
    return a+b,b+a
}
```

test_remote_string.digo.pass.expected

```
hahahehe
hehehaha
```

test-remote.digo

```
func digo_main() void {
    a := 10
    b := 1.0
    c := try_return_one(a)
    e := try_return_two(a,b)
    f, g := await e

    println("%d", f)
    println("%f", g)
}

async remote func try_return_one(a int) int {
    return a+10
}

async remote func try_return_two(a int,b float) (int,float) {

    return a+10,b+20.0
}
```

test-remote.digo.pass.expected

```
20
21.000000
```

test-wordcount.digo

```
//word-count
```

```
// The workers execute this `async remote` function.
// Calling an `async remote` function returns immediately
// after the job is dispatched to a worker. It does not
// wait for the job to finish.
// An `async remote` function does not return with return values
// defined in the function prototype.
// Instead, it returns a future object, on which
// user can call await/gather to get the actual return values.

async remote func remote_count_word(words []string) ([]string, []int) {
     // println("in remote_count_word: parameters: %l", words)
    // we first sorts the words.
    quick_sort(words, 0, len(words) - 1)

    resultWord := []string{}
    resultCount := []int{}
    if (len(words) == 0) {
        return resultWord, resultCount
    }
    word := words[0]
    resultWord = append(resultWord, word)
    resultCount = append(resultCount, 1)

    i := 1
    for (i = 1; i < len(words); i = i + 1) {
        if (words[i] == word) {
            resultCount[len(resultWord)-1] = resultCount[len(resultWord)-1]
+ 1
        } else {
            word = words[i]
            resultWord = append(resultWord, words[i])
            resultCount = append(resultCount, 1)
        }
    }

    /// println("remote_count_word: going to return: %l | %l", resultWord,
resultCount)
    return resultWord, resultCount
}

// The master executes this `async` function in a new thread.
// Calling an `async` function returns immediately with
```

```
// a implicit future object.
async func count_words_in_file(file string, taskCount int) ([]string,
[]int) {

    words := read(file)
    worker_word_cnt := len(words) / taskCount
    i := 0

    if (len(words) == 0) {
        println("warning!!!! len words = 0")
    }

    futures := []future{}
    for (i = 0; i < len(words); i = i + worker_word_cnt) {
        // Calling the `async remote` worker_count_word function
        // will automatically send the task to a worker.
        // It does not block.
        futures = append(futures, remote_count_word(words[i : min(i +
worker_word_cnt, len(words))]))
    }

    mergedWords := []string{}
    mergedCount := []int{}

    for (i = 0 ; i < len(futures) ; i = i + 1) {
        // Here, we are explicity waiting for the remote task
        // to finish.
        remote_task := futures[i]
        resultWord, resultCount := await remote_task

        w, c := merge_two_word_cnt(resultWord, resultCount, mergedWords,
mergedCount)
        // println("count_words_in_file: current: %l | %l", mergedWords,
mergedCount)
        mergedWords = w
        mergedCount = c
    }

    // println("count_words_in_file: going to return: %l | %l",
mergedWords, mergedCount)
    return mergedWords, mergedCount
}
```

```
// we merge wordcnt1 into wordcnt2
func merge_two_word_cnt(words1 []string, cnt1 []int, words2 []string, cnt2
[]int) ([]string, []int) {
    j := 0
    k := 0

    for (j = 0; j < len(words1); j = j + 1) {
        found := 0
        for (k = 0; k < len(words2); k = k + 1) {
            if (words1[j] == words2[k]) {
                if (found == 1) {
                    println("error")
                }
                found = 1
                cnt2[k] = cnt2[k] + cnt1[j]
            }
        }
        if (found == 0) {
            words2 = append(words2, words1[j])
            cnt2 = append(cnt2, cnt1[j])
        }
    }
    return words2, cnt2
}

func word_count_entry() {
    futures := []future{}

    futures = append(futures, count_words_in_file("./Remote/book1.txt", 4))
    futures = append(futures, count_words_in_file("./Remote/book2.txt", 4))
    futures = append(futures, count_words_in_file("./Remote/book3.txt", 4))

    // Calling `async`/`async remote` functions does not block unless
    // we expicitly wait for them to finish by calling await/gather.
    // So word_count_entry() does not block until we `gather` here:

    i := 0
    for (i = 0; i < len(futures); i = i + 1) {
        println("Word Count result of book%d: ", i + 1)
        f := futures[i]
        words, count := await f
        if (len(words) != len(count)) {
            println("error!")
```

```
        } else {
            var j int
            for (j = 0; j < len(words); j = j + 1) {
                println("word: %s, count: %d", words[j], count[j])
            }
        }
    }
}

// a quick sort function for string slice
func quick_sort(arr []string, p int, q int) void {
    if (p < q) {
        pivot := arr[p]
        i := p
        var j int
        for (j = p + 1; j <= q; j = j + 1) {
            if (arr[j] <= pivot) {
                i = i + 1
                swap(arr, i, j)
            }
        }
        swap(arr, p, i)
        quick_sort(arr, p, i - 1)
        quick_sort(arr, i + 1, q)
    }
}

func swap(arr []string, i int, j int) void {
    tmp := arr[i]
    arr[i] = arr[j]
    arr[j] = tmp
}

func min(p int, q int) int {
    if (p < q) {
        return p
    } else {
        return q
    }
}

func digo_main() void {
    word_count_entry()
```

```
}
```

test-wordcount.digo.pass.expected

```
Word Count result of book1:
word: a, count: 1
word: b, count: 1
word: c, count: 1
word: d, count: 1
word: e, count: 1
word: f, count: 1
Word Count result of book2:
word: a, count: 2
word: b, count: 2
word: c, count: 2
word: d, count: 2
word: e, count: 2
word: f, count: 2
Word Count result of book3:
word: df, count: 3
word: kk, count: 3
word: k1, count: 1
word: k2, count: 1
word: k3, count: 1
word: k4, count: 1
word: k5, count: 1
word: k6, count: 1
```

9.5.7 Semantic

fail-append_different_types.digo

```
func digo_main() void {
  a := []int {1,2,3,4}
  var j float
  b := append(a,j)
}
```

fail-append_different_types.digo.fail.expected

```
Fatal error: exception Failure("Semantic Err: cannot append type float to
Slice(int) in expression append(a, j)")
```

fail-append_nonslice_object.digo

```
func digo_main() void {
  var i int
  var j int
  a := append(i,j)
}
```

fail-append_nonslice_object.digo.fail.expected

```
Fatal error: exception Failure("Built-in Append being called on non-slice object")
```

fail-assign_different_types.digo

```
func digo_main() void {
  var i int
  var j float  = 2.0
  i = j
}
```

fail-assign_different_types.digo.fail.expected

```
Fatal error: exception Failure("illegal assignment ")
```

fail-assign_nonvariable.digo

```
func digo_main() void {
  var i int
  var j int
  i + j = 2
}
```

fail-assign_nonvariable.digo.fail.expected

```
Fatal error: exception Failure("AssignOp error: left hand side is invalid")
```

fail-digo_main_return_value.digo

```
func digo_main() int {
  return 0
}
```

fail-digo_main_return_value.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: digo_main should be void
type.")
```

fail-digo_main_undeclared.digo

```
func foo() void {

}
```

fail-digo_main_undeclared.digo.fail.expected

```
Fatal error: exception Failure("unrecognized function digo_main")
```

fail-digo_main_with_arguments.digo

```
func digo_main(a int,b float) void {

}
```

fail-digo_main_with_arguments.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: digo_main should be no-
arugment.")
```

fail-duplicate_arguments.digo

```
func digo_main() void {
  var i int = 1
  var j int = 2
  foo(i,j)
}

func foo(a int,a int) void {
  return 1
}
```

fail-duplicate_arguments.digo.fail.expected

```
Fatal error: exception Failure("duplicate in argument : a")
```

fail-duplicate_arguments.digo

```
func digo_main() void {
```

```
}

func foo() void {
  return 1
}

func foo() void{
  return 0
}
```

fail-duplicate_arguments.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: duplicate function foo")
```

fail-duplicate_local_var.digo

```
func digo_main() void {

}

func foo() void {
  return 1
}

func foo() void{
  return 0
}
```

fail-duplicate_local_var.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: duplicate local variable
declarations ( i )")
```

fail-duplicate_function_name.digo

```
func digo_main() void {

}

func foo() void {
  return 1
}
```

```
func foo() void{
  return 0
}
```

fail-duplicate_function_name.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: duplicate local variable
declarations ( i )")
```

fail-function_different_number_of_arguments.digo

```
func digo_main() void {

}

func foo() void {
  return 1
}

func foo() void{
  return 0
}
```

fail-function_different_number_of_arguments.digo.fail.expected

```
atal error: exception Failure("Semant Err: different number of aruguments
passed. Expected 2 aruguments but 1 aruguments provided in function foo")
```

fail-function_undefined.digo

```
func digo_main() void {
  foo()
}
```

fail-function_undefined.digo.fail.expected

```
Fatal error: exception Failure("unrecognized function foo")
```

fail-illegal_append.digo

```
func digo_main() void {
  a := []int {1,2,3,4}
  b := append(b,10,20)
}
```

```
```

fail-illegal_append.digo.fail.expected

```
Fatal error: exception Failure("Append needs two objects")
```

fail-illegal_declare.digo

```
func digo_main() void {
  var i int = 1,2
}
```

fail-illegal_declare.digo.fail.expected

```
Fatal error: exception Failure("assignment mismatch: 1 variables but 2
values")
```

fail-illegal_declare2.digo

```
func digo_main() void {
  var i int = 10.0
}
```

fail-illegal_declare2.digo.fail.expected

```
Fatal error: exception Failure("illegal assignment type")
```

fail-illegal_len.digo

```
func digo_main() void {
  var i int
  len(i)
}
```

fail-illegal_len.digo.fail.expected

```
Fatal error: exception Failure("Built-in Len being called on non-slice
object i")
```

fail-illegal_read.digo

```
func digo_main() void {
  var s int
  f := read(s)
}
```

fail-illegal_read.digo.fail.expected

```
Fatal error: exception Failure("error: read is not supported for int")
```

fail-illegal_shortdecl1.digo

```
func digo_main() void {
  i := 2,3
}
```

fail-illegal_shortdecl1.digo.fail.expected

```
Fatal error: exception Failure("short decl assignment mismatch: 1 variables
but 2 values")
```

fail-illegal_shortdecl2.digo

```
func digo_main() void {
  i := 2
  i := 3
}
```

fail-illegal_shortdecl2.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: duplicate local variable
declarations ( i )")
```

fail-illegal_slice_index.digo

```
func digo_main() void {
  a := []int {1,2,3,4}
  b := a[1.0]
}
```

fail-illegal_slice_index.digo.fail.expected

```
Fatal error: exception Failure("illgal slice index: non-integer index")
```

fail-statement_after_return.digo.digo

```
func digo_main() void {
  foo()
}

func foo() int {
  return 0
  return 10
}
```

fail-statement_after_return.digo.fail.expected

```
Fatal error: exception Failure("Statements appear after Return")
```

fail-undeclared_future_object.digo

```
func digo_main() void {
  f := await foo
}
```

fail-undeclared_future_object.digo.fail.expected

```
Fatal error: exception Failure("Semant Err: undeclared future object foo")
```

fail-undeclared_variable.digo

```
func digo_main() void {
  print("%d",i)
}
```

fail-undeclared_variable.digo.fail.expected

```
Fatal error: exception Failure("Sement Err: undeclared identifier i")
```

fail-void_type_argument.digo

```
func digo_main() void {
  var a void
  foo(a)
}

func foo(a void) void {
  return 1
}
```

fail-void_type_argument.digo.fail.expected

```
Fatal error: exception Failure("illegal VoidType in argument : a")
```

9.5.8 Syntax

test-comment.digo

```
func digo_main() void {
  a := 20
```

```
  // a = 10
  println("%d", a)
}
```

test-comment.digo.pass.expected

```
20
```

test-comment2.digo

```
func digo_main() void {
  a := 10
  /* this is a comment
   a = 20
   end of commment */
  println("%d", a)
}
```

test-comment2.digo.pas.expected

```
10
```

test-func1.digo

```
func add(a int, b int) int {
  return a + b
}

func digo_main() void {
  a := add(39, 3)
  println("%d", a)
}
```

test-func1.digo.pass.expected

```
42
```

test-id.digo

```
func digo_main() void {
  _a := 1
```

```
  println("%d", _a)
}
```

test-id.digo.fail.expected

```
Fatal error: exception Failure("lexing: empty token")
Test-id2.digo
func digo_main() void {
  1a := 1
  println("%d", 1a)
}
```

test-id2.digo

```
func digo_main() void {
  1a := 1
  println("%d", 1a)
}
```

test-id2.digo.fail.expected

```
Fatal error: exception Parsing.Parse_error
```

fail-duplicate_arguments.digo

```
func digo_main() void {
  var i int = 1
  var j int = 2
  foo(i,j)
}

func foo(a int,a int) void {
  return 1
}
```

fail-duplicate_arguments.digo.fail.expected

```
Fatal error: exception Failure("duplicate in argument : a")
```

9.5.9 Utils

test-print.digo

```
func digo_main() void {
```

```
  println("%d", 42)
  println("%f", 42.012)
  println("%s", "hello")
}
```

test-print.digo.pass.expected

```
42
42.012000
hello
```

test-read.digo

```
func digo_main() void {
  words := read("./Utils/words.txt")
  println("%d", len(words))
  println("%s", words[0])
}
```

test-read.pass.expected

```
11
hello,
```

9.13 Makefile

```
all: gen build

.PHONY: gen
gen: clean generate-dependency generate-digo-compiler

.PHONY: build-compiler-pass
build-compiler-pass:
	./digo-compiler/digo.native $(digo) > tmp.compiled.nometadata.ll

.PHONY: build
build: build-compiler-pass build-link-pass

.PHONY: print-llvm
print-llvm:
	./digo-compiler/digo.native $(digo)

.PHONY: build-link-pass
```

```
build-link-pass:
	./digo-compiler/metadata_gen < $(digo) > tmp.metadata.ll
	cat tmp.compiled.nometadata.ll tmp.metadata.ll > tmp.compiled.ll
	./digo-linker/digo-linker async tmp.compiled.ll tmp.async.linker.ll
	llvm-link -S -v -o tmp.async.ll tmp.compiled.ll tmp.async.linker.ll
	clang++ -stdlib=libc++ -pthread dependency.ll tmp.async.ll -o $(out)

.PHONY: generate-async-remote-lib
generate-async-remote-lib:
	$(MAKE) link-all-llvm -C async-remote-lib

.PHONY: generate-digo-linker
generate-digo-linker:
	$(MAKE) build-all -C digo-linker

.PHONY: generate-digo-compiler
generate-digo-compiler:
	$(MAKE) generate-compiler -C digo-compiler
	$(MAKE) metadata-generator -C digo-compiler

.PHONY: generate-dependency
generate-dependency: generate-async-remote-lib generate-digo-linker
	llvm-link -S -v -o dependency.ll async-remote-lib/allinone.ll digo-
linker/allinone.ll

.PHONY: clean
clean:
	rm -f dependency.ll executable
	rm -f tmp.compiled.ll tmp.async.ll tmp.metadata.ll
tmp.compiled.nometadata.ll tmp.async.linker.ll
	$(MAKE) clean -C digo-linker
	$(MAKE) clean -C async-remote-lib
	$(MAKE) clean -C digo-compiler
```

9.14 .CircleCI

9.14.1 config.yml

```
version: 2.1
```

```
jobs:
  build:
    docker:
     - image: wy2249/plt

    working_directory: /home/microc

    steps:
      - checkout
      - run: |
          eval `opam config env`
          cd ./digo-test
          ./testall.sh || ( cat testall.log && exit 1 )
```

9.15 Sample Code (Demo)

9.15.1 wordcount.digo

```
//  DEMO: word-count

// The workers will execute this `async remote` function.
// Calling an `async remote` function returns immediately
// after the job is dispatched to a worker. It does not
// wait for the job to finish.
//
// Calling an `async remote` function does not return with return values
// defined in the function prototype.
// Instead, it returns a future object, on which user can call await
// to wait for the job to finish and get the actual return values.
async remote func remote_count_word(words []string) ([]string, []int) {
    // println("in remote_count_word: parameters: %l", words)
    // we first sorts the words.
    quick_sort(words, 0, len(words) - 1)

    resultWord := []string{}
    resultCount := []int{}
    if (len(words) == 0) {
        return resultWord, resultCount
    }
    word := words[0]
    resultWord = append(resultWord, word)
```

```
    resultCount = append(resultCount, 1)

    i := 1
    for (i = 1; i < len(words); i = i + 1) {
        if (words[i] == word) {
            resultCount[len(resultWord)-1] = resultCount[len(resultWord)-1]
+ 1
        } else {
            word = words[i]
            resultWord = append(resultWord, words[i])
            resultCount = append(resultCount, 1)
        }
    }

    // println("remote_count_word: going to return: %l | %l", resultWord,
resultCount)
    return resultWord, resultCount
}

// The master executes this `async` function in a new thread.
// Calling an `async` function returns immediately with
// a future object.
async func count_words_in_file(file string, taskCount int) ([]string,
[]int) {

    words := read(file)
    worker_word_cnt := len(words) / taskCount
    i := 0

    if (len(words) == 0) {
        println("warning!!!! len words = 0")
    }

    futures := []future{}
    for (i = 0; i < len(words); i = i + worker_word_cnt) {
        // Calling the `async remote` remote_count_word function
        // will send the task to a worker and return immediately with a
future object.
        // It does not block.
        futures = append(futures, remote_count_word(words[i : min(i +
worker_word_cnt, len(words))]))
    }
```

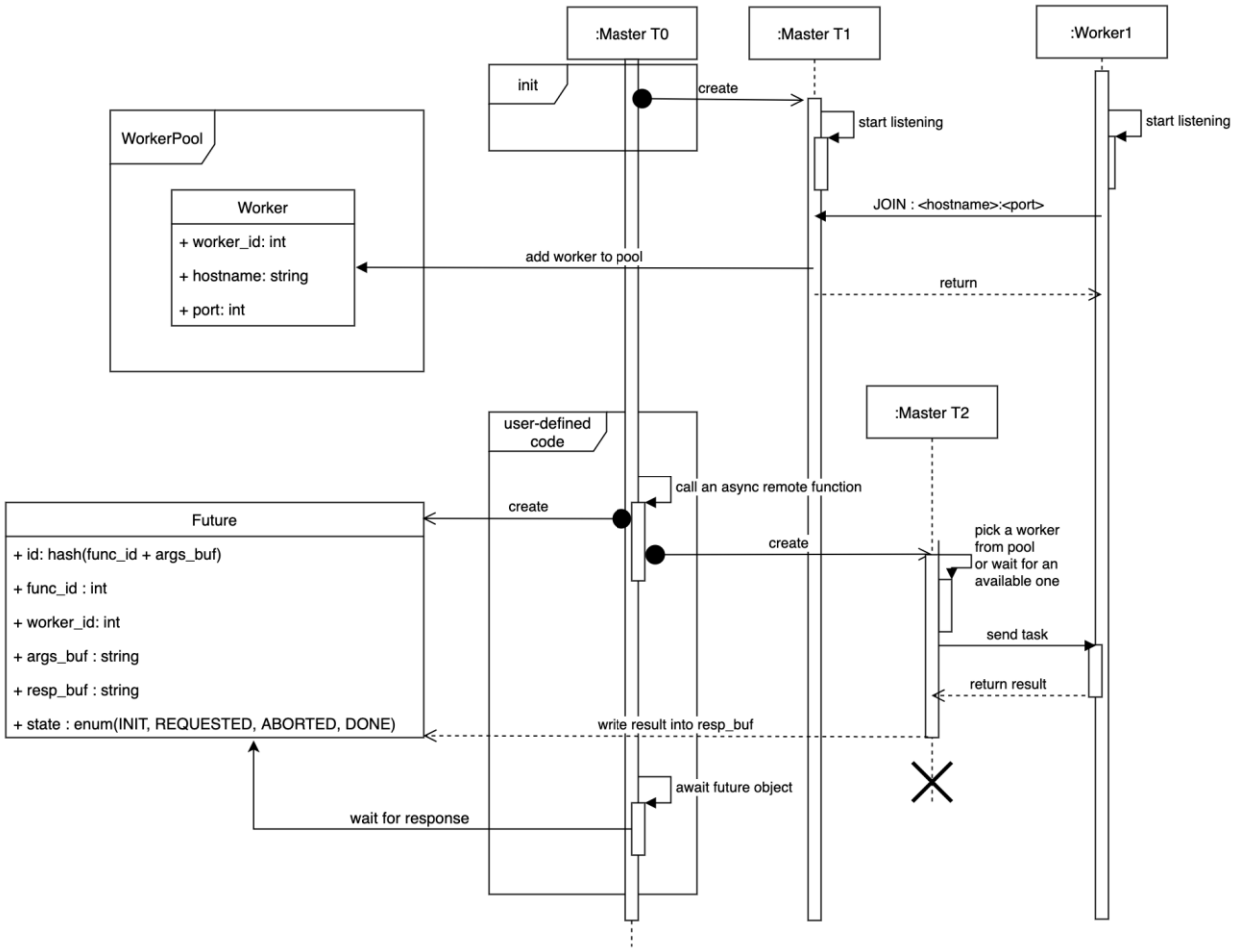
```
    mergedWords := []string{}
    mergedCount := []int{}

    for (i = 0 ; i < len(futures) ; i = i + 1) {
        // Here, we are explicity waiting for the remote task
        // to finish.
        remote_task := futures[i]
        resultWord, resultCount := await remote_task

        w, c := merge_two_word_cnt(resultWord, resultCount, mergedWords,
mergedCount)
        // println("count_words_in_file: current: %l | %l", mergedWords,
mergedCount)
        mergedWords = w
        mergedCount = c
    }

    // println("count_words_in_file: going to return: %l | %l",
mergedWords, mergedCount)
    return mergedWords, mergedCount
}

// we merge wordcnt1 into wordcnt2
func merge_two_word_cnt(words1 []string, cnt1 []int, words2 []string, cnt2
[]int) ([]string, []int) {
    j := 0
    k := 0

    for (j = 0; j < len(words1); j = j + 1) {
        found := 0
        for (k = 0; k < len(words2); k = k + 1) {
            if (words1[j] == words2[k]) {
                if (found == 1) {
                    println("error")
                }
                found = 1
                cnt2[k] = cnt2[k] + cnt1[j]
            }
        }
        if (found == 0) {
            words2 = append(words2, words1[j])
            cnt2 = append(cnt2, cnt1[j])
        }
```

```
    }
    return words2, cnt2
}

func word_count_entry() {
    futures := []future{}

    futures = append(futures, count_words_in_file("./Demo/book1.txt", 4))
    futures = append(futures, count_words_in_file("./Demo/book2.txt", 4))
    futures = append(futures, count_words_in_file("./Demo/book3.txt", 4))
    futures = append(futures,
count_words_in_file("./Demo/haskell_book_chapter2.txt", 6))

    // Calling `async`/`async remote` functions does not block unless
    // we expicitly wait for them to finish by calling await.

    i := 0
    for (i = 0; i < len(futures); i = i + 1) {
        println("Word Count result of book%d: ", i + 1)
        f := futures[i]
        // So count_words_in_file() does not block until we `await` here:
        words, count := await f
        if (len(words) != len(count)) {
            println("error!")
        } else {
            var j int
            for (j = 0; j < len(words); j = j + 1) {
                println("word: %s    count: %d", words[j], count[j])
            }
        }
    }
}

// a quick sort function for string slice
func quick_sort(arr []string, p int, q int) void {
    if (p < q) {
        pivot := arr[p]
        i := p
        var j int
        for (j = p + 1; j <= q; j = j + 1) {
            if (arr[j] <= pivot) {
                i = i + 1
                swap(arr, i, j)
```

```
            }
        }
        swap(arr, p, i)
        quick_sort(arr, p, i - 1)
        quick_sort(arr, i + 1, q)
    }
}

func swap(arr []string, i int, j int) void {
    tmp := arr[i]
    arr[i] = arr[j]
    arr[j] = tmp
}

func min(p int, q int) int {
    if (p < q) {
        return p
    } else {
        return q
    }
}

func digo_main() void {
    word_count_entry()
}
```

9.16 How Digo works with threads and networks, explained with diagram

The following plot tries to show the whole underlying process in digo:



As shown in the plot, the master keeps two major threads (T0 and T1): T0 is the main thread to execute the master function, and when it inits, it creates T1 to create and maintain a socket sitting there and waiting for any connections from newly-added workers. When a worker (worker1) is added and starts running, it firstly listens to a port and sends <hostname, port> to the master. After the master(T1) receives this information, it records <worker_id, hostname, port> in WorkerPool. The worker can close the connection after it receives ACK from the master.

When executing the master function, everytime a future object is to be created by calling an async remote function, a new thread (T2 in the diagram) is created with it to:

1. run scheduler to find a proper worker in the WorkerPool maintained by the master. If there is no available worker, this thread will be stuck in a loop until it gets one.
2. after it finds one worker to deliver the task to, it sends <function_id + serialized parameters + future_object_id> to the assigned worker.

Note that, by design, we only pass values instead of pointers to async remote functions and do not allow async tasks to access global variables.

When the worker receives a task request from the master, it starts a new thread to look for the function specified by the function_id it received, and then execute the task. After it finishes the function, it sends a response containing <result, future_object_id> back to the master. After the thread (T2) in the master receives the response from the worker, it copies the response to resp_buf in the corresponding future object. When the master executes await, the underlying function enters a loop to check if the resp_buf in the future object is filled, otherwise it waits for its response in a blocking way.