Will Lillis
DATE HERE

CONTEXT

This template was made to stylize the reports I wrote for the **Forta Foundation**.

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Template creation	2023/07/31	Apehex
1.0	Complete template	2023/10/04	Apehex
1.1	Various improvements	2024/01/11	Apehex

CONTACTS

CONTACT	MAIL	MORE
Will Lillis	**will.lillis@gmail.com**	**Github: WillLillis**

CREDITS

The template started from the **Legrand Orange template**.

The cover page started from the templates in **Latexdraw**.

The syntax highlighting for Solidity has been made by **Sergei Tikhomirov**.

PART 1

1. INTRODUCTION

1.0.1. Introduction

I started my programming journey much later than a lot of my peers. Coding was some scary topic better left to child prodigies and other geniuses, so why even both I reasoned. Nevertheless, following my first year of undergrad I landed an internship with my school's Physics department. Little did I know the internship was about 1% physics, and 99% learning C++. My professor worked in low-energy nuclear physics, and his group's experiments generated large amounts of data that needed to be processed. My first introduction with "real" coding was with C++, doing large-scale analysis. I spent my summer switching between online documentation and gedit, the text editor he showed me on my first day. Following this internship, I started to Little did I know, the internship was actually writing C++ code to perform analysis onlarge amounts of dataI was intimidated by the

Eventually a friend took pity on me and pointed me out to Visual Studio. I cannot understate how mindblowing things like autocomplete, jump to definition, and intelligent syntax highlighting was to me after months of default Notepad++. I wondered how these features worked, but understanding seemed beyond my grasp. If I'm struggling to even write code, then code that can understand other code has to be on some elevated plane of complexity.

Eventually I stumbled upon The Primeagen by accident on YouTube one day. This man fundamentally changed my outlook on CS. After months of listening to him shill for Neovim and Rust, I decided to spend a summer learning the language and editor. These things scared me, but I had finally gotten over this fear and just decided to try my best and see where I got.

I *cannot* understate what a great decision this was. Coding has never been more fun. After reading through the Rust book, I set out to complete the 2021 Advent of Code calendar. Working through those exercises (and sometimes referring to Reddit for help), I slowly figured things out.

After a few weeks, I was comfortable enough to start making small modifications to the Lua configuration I had copied from Prime's setup video.

I eventually stumbled onto asm-lsp, an open-source assembly LSP written in Rust. The project had hover support for instructions, but that was about it. I found a small `TODO` comment in the source code, and after an hour of reading through the code, I thought I had an idea of how to solve the problem. I opened a PR, and to my surprise it was accepted. I started digging into the code a bit more, and found another thing to fix. And then I found a feature to add. And then another. And another. Over the course of a few months, I got very comfortable with the LSP's code, and in turn also found myself fairly comfortable with the LSP protocol. Working on the project had demysitified the "magic" code that understood code, and I found myself enjoying the rabbit hole I had unwitteningly dived down.

This learning journey was incredibly valuable to me, and I'd like to help others discover this magic.

1.0.2. What is this book?

This book walks you through building an LSP server for x86-64 assembly.

"Why Assembly? Nobody actually codes in that anymore."

Fair enough, almost nobody actually hand rolls assembly anymore. Those damned compiler people have done too good of a job! But in all seriousness, I've chosen assembly for its *simplicity*. Assembly is simple enough that the majority of one's effort can be spent learning the main item, LSPs! Assembly's flat structure makes things just simple enough so that lots of features can be doable without weeks of studying a specific language's features. The obvious alternative is to use a toy language, which in my opinion sucks all the fun out.

We'll start with some general LSP information and project setup. While I won't skip any steps, I'll do my best to speed through the non-LSP parts to keep things interesting. While I'm sure there's lots to be learned about things like XML parsing, serialization, and deserialization, I'm not the right person and this isn't the right book. After these initial steps, we'll have a working program that can connect to an LSP client, load some relevant information from the disk...and that's about it. But don't despair! The very next thing on our list is to add hover support for instructions and registers. After this feature is in place, we'll sidetrack briefly to add some configuration options for our users (of which there will be many thousands, so we can also practice asking the product manager "but how will this scale?"). Next we'll add autocomplete!

Doing so will require a couple different steps, but this is when things truly start to get fun. When I first added autocomplete to asm-lsp, things really started to click. After autocomplete, we'll add a bunch of cool (but relatively siloed) capabilities.

2. GETTING STARTED

All of the finished code for each chapter can be found at https://github.com/WillLillis/LSPBook. With that being said, let's get started! As with nearly any other Rust project, we can start by creating a new project with cargo: `cargo new assembly-lsp`. We'll add some dependencies next.

- `cargo add crossbeam-channel`
  - Handles sending and receiving data across channels so our LSP server can talk to our editor's LSP client
- `cargo add flexi_logger`
  - Allows for some easy structured logging over `stderr`.
- `cargo add log`
  - Another logging crate we'll use
- `cargo add lsp-server`
  - Handles the majority of LSP-related things for us. This crates lets us focus on features, rather than implementing a spec.
- `cargo add lsp-types`
  - The LSP spec includes a fair number of type defitinitions, which are (unfortunately) in Typescript. This crate provides Rust versions of the spec's types that adhere to the spec.
- `cargo add serde --features derive`
  - This amazing crate is part of nearly every Rust project, including ours too.
- `cargo add serde_json`
  - LSP servers and clients communicate over JSON RPC. Naturally this crate will be of use.

Now that the scaffolding is in place, we can finally get to coding. A great template can be found in the rust-analyzer (the absolutely excellent Rust LSP) **repo**. Their example starts out with some code for the go to definition capability. We'll get there eventually, but our starting point is different. Here's the stripped down code:

```rust
use log::{error, info};
use lsp_server::{Connection, ExtractError, Message, Request, RequestId};
use lsp_types::{InitializeParams, ServerCapabilities};

fn main() -> anyhow::Result<()> {
    // Set up logging. Because `stdio_transport` gets a lock on stdout and stdin, we must have our
    // logging only write out to stderr.
    flexi_logger::Logger::try_with_str("info")?.start()?;

    info!("Starting assembly-lsp");

    // Create the transport. Includes the stdio (stdin and stdout) versions but this could
    // also be implemented to use sockets or HTTP.
    let (connection, io_threads) = Connection::stdio();

    // Run the server and wait for the two threads to end (typically by trigger LSP Exit event).
    let server_capabilities = serde_json::to_value(&ServerCapabilities {
```

```
        ..Default::default()
    })
    .unwrap();

    let initialization_params = connection.initialize(server_capabilities)?;

    main_loop(connection, initialization_params)?;
    io_threads.join()?;

    // Shut down gracefully.
    error!("Shutting down assembly-lsp");
    Ok(())
}

fn main_loop(connection: Connection, params: serde_json::Value) -> anyhow::Result<()> {
    let _params: InitializeParams = serde_json::from_value(params).unwrap();
    eprintln!("starting example main loop");
    for msg in &connection.receiver {
        eprintln!("got msg: {msg:?}");
        match msg {
            Message::Request(req) => {
                if connection.handle_shutdown(&req)? {
                    return Ok(());
                }
                error!("Got request: {req:?}");
            }
            Message::Response(resp) => {
                error!("Got response: {resp:?}");
            }
            Message::Notification(notif) => {
                error!("Got notification: {notif:?}");
            }
        }
    }
    Ok(())
}

fn cast_req<R>(req: Request) -> Result<(RequestId, R::Params), ExtractError<Request>>
where
    R: lsp_types::request::Request,
    R::Params: serde::de::DeserializeOwned,
{
    req.extract(R::METHOD)
}
```

3. IPSUM

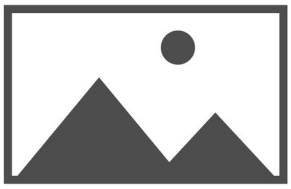
3.1. PROIN NEC METUS VENENATIS

Condimentum quam sed, auctor diam. Fusce at bibendum nunc. Vestibulum ut justo volutpat, placerat turpis at, ultrices lacus. Phasellus blandit fermentum tincidunt.

Quisque eget est ullamcorper massa vestibulum facilisis eget non quam. Pellentesque eleifend augue vel elit mattis, ut porta risus sodales. Suspendisse vitae odio facilisis, venenatis leo ut, consequat turpis.

3.1.1. Integer magna quam

In hac habitasse platea dictumst. Pellentesque a hendrerit mauris, ac congue erat. Maecenas in lorem non velit euismod volutpat. Morbi et arcu quam. Mauris id scelerisque est. Cras a nunc in justo blandit dictum nec ut tortor.



Maecenas varius orci vitae eros euismod placerat

3.1.2. Convallis quis elementum feugiat

Rutrum in ligula. Nulla ut semper ligula, sed lobortis enim. Vivamus nulla eros, faucibus ac convallis in, ornare at elit. Aliquam erat odio, finibus at purus ut, porta ullamcorper odio. Vivamus ac augue tincidunt, semper magna volutpat, condimentum sem. Praesent eget neque vitae odio volutpat vulputate.

DOLOR SIT AMET

4. HENDRERIT SAPIEN

4.1. SED ULTRICES

4.1.1. Donec pellentesque

Tempus ipsum, vitae condimentum nisi efficitur id. In velit mauris, auctor eget sapien nec, viverra mattis neque. Nunc vel commodo nunc, eget cursus ex.

1. Nunc maximus consequat tristique.
2. Praesent luctus ex aliquam rhoncus consectetur.
3. Suspendisse mattis velit ante, vitae mollis est pharetra a.
4. Duis tincidunt, urna id auctor imperdiet, odio dui ullamcorper nisi.
5. Integer tincidunt enim vitae nulla iaculis, in varius metus blandit.
6. Nunc quam arcu, fermentum non dapibus condimentum, condimentum eget nunc.

4.1.2. Euismod sodales

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

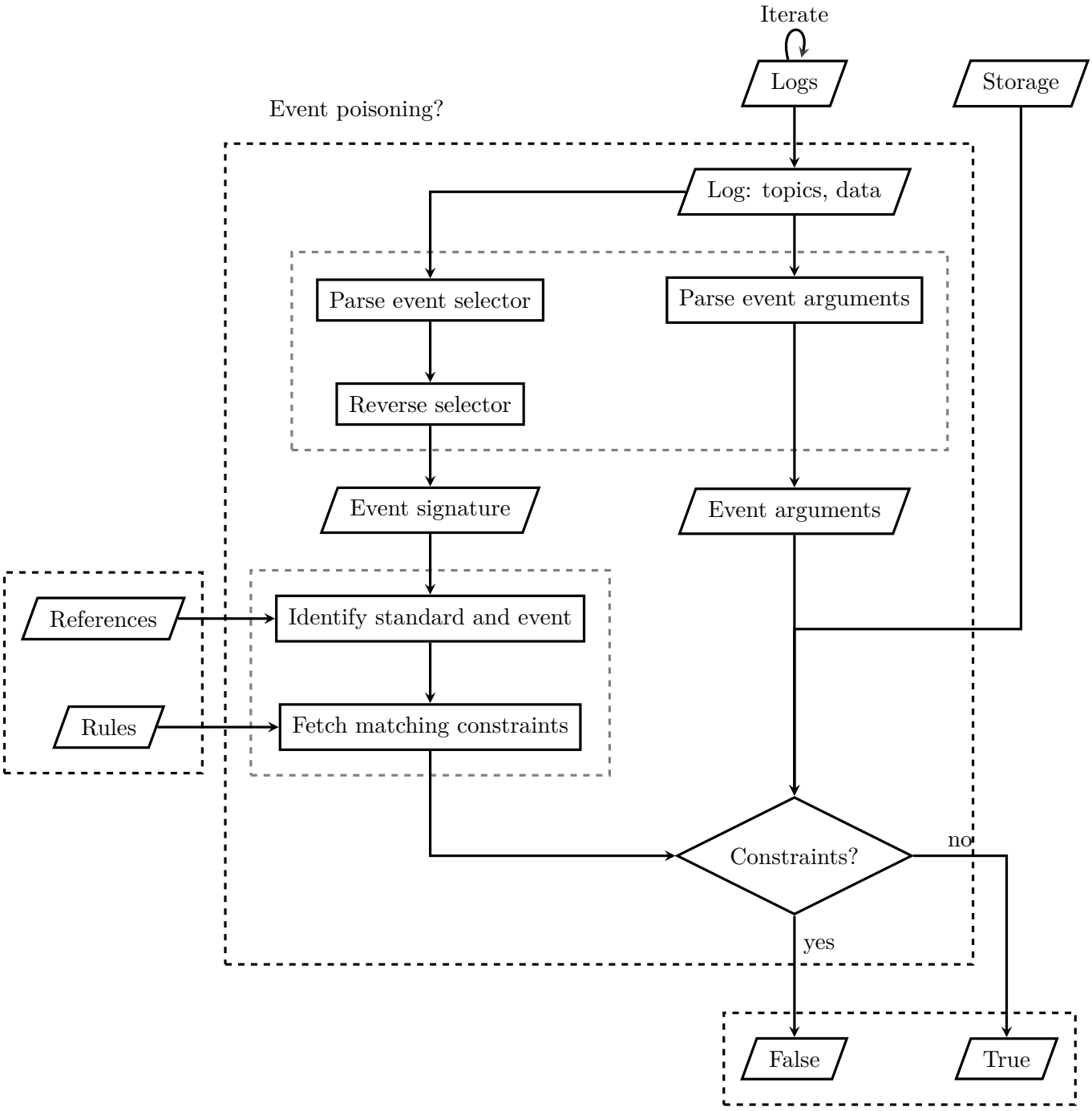
4.1.3. Pellentesque a nulla

```
1  contract KingOfTheHill is Ownable {
2      address public owner; // different from the owner in Ownable
3
4      function () public payable {
5          if(msg.value > jackpot) owner = msg.sender; // local owner
6          jackpot += msg.value;
7      }
8      function takeAll () public onlyOwner { // contract creator
9          msg.sender.transfer(this.balance);
10         jackpot = 0;
11     }
12 }
```

4.2. PHASELLUS TRISTIQUE

Lacus ac turpis semper fringilla. Mauris fermentum varius neque, vel congue elit tempus quis. Aliquam a nunc in nunc consectetur hendrerit ut quis felis.

Quisque id dui magna. Curabitur posuere nisl at magna vehicula aliquam 4.1. Duis et suscipit felis. Mauris non justo quis diam gravida placerat condimentum ut sapien. Praesent imperdiet libero id est tincidunt ullamcorper.



Maecenas hendrerit congue faucibus

APPENDICES

5. VESTIBULUM COMMODO

INTEGER SEMPER DICTUM TELLUS

Neque eu cursus faucibus, ipsum magna tincidunt dui, vel scelerisque urna nibh ut nisl:

Duis sit amet mattis magna.
Curabitur sit amet fermentum mi.
Morbi convallis purus eu fermentum accumsan, mauris felis consequat ipsum.
Ut sollicitudin sit amet tellus et mollis.

A PORTTITOR ORCI FAUCIBUS SIT AMET

Vivamus et sapien vitae lacus ornare suscipit vitae id mauris:

Vivamus in dui arcu.
Morbi vitae dolor libero.
Tellus est, pellentesque at ultrices non, consectetur sit amet augue.
Suspendisse elementum mollis nisl at aliquam.

6. SOLLICITUDIN

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

// github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/token/ERC20/IERC20.sol
interface IERC20 {
    function totalSupply() external view returns (uint);

    function balanceOf(address account) external view returns (uint);

    function transfer(address recipient, uint amount) external returns (bool);

    function allowance(address owner, address spender) external view returns (uint);

    function approve(address spender, uint amount) external returns (bool);

    function transferFrom(
        address sender,
        address recipient,
        uint amount
    ) external returns (bool);

    event Transfer(address indexed from, address indexed to, uint value);
    event Approval(address indexed owner, address indexed spender, uint value);
}
```

```
{
    "blockHash": "0xfff25d13f3e37982e6a380771a52ea79d60d4592fb13f24a157a96bd48cb823a",
    "blockNumber": "0x011367d1",
    "from": "0x7d9bc45a9abda926a7ce63f78759dbfa9ed72e26",
    "gas": "0x01724b",
    "gasPrice": "0x030305bec8",
    "maxFeePerGas": "0x03f6b8baa1",
    "maxPriorityFeePerGas": "0x05f5e100",
    "hash": "0xe0a98402cb9b9536b4b308458ba46f23d41a51dfd6ee4102c118230a44529cbd",
    "input": "0xa9059cbb000000000000000000000000e897c0f9443785f8d4f0fa6e...",
    "nonce": "0x17",
    "to": "0xdac17f958d2ee523a2206206994597c13d831ec7",
    "transactionIndex": "0xd5",
    "value": "0x00",
    "type": "0x02",
    "accessList": "0x",
    "chainId": "0x01",
    "v": "0x01",
    "r": "0x66ffefa83e6ce55d11ee5d9eb5e0534b9b8579d2e354f8ea72d8f91250fc5893",
    "s": "0x328dc78a37989de1c52bd4b5f24c1a7f9796981e6eb8ac520a5945b06888fef4",
    "yParity": "0x01"
}
```

7. ETIAM FACILISIS

7.1. IPSUM PRIMIS

In faucibus orci luctus et ultrices posuere cubilia curae; Aliquam nunc ipsum, sollicitudin ut tempus consectetur, fermentum at lectus.

Nullam molestie viverra **??** augue sit amet gravida.

7.1.1. Mauris pellentesque

Massa sagittis malesuada

Symbol	Unit	Description
g	m/s^2	nisi in mollis gravida

7.1.2. Nulla massa

Quis imperdiet

Symbol	Unit	Description
Q_p	t/h	consectetur tortor

Magna lectus

Symbol	Unit	Description
τ_a	$^\circ C$	integer mollis bibendum gravida
f_s	$/jour$	vestibulum porta urna at ex dignissim tincidunt

7.1.3. Nam vitae

Symbol	Unit	Description
X_n	m	venenatis nisi
Y_n	m	integer malesuada eu ligula nec pharetra
Z_n	m	fusce non elit lectus

LIST OF FIGURES

LIST OF TABLES