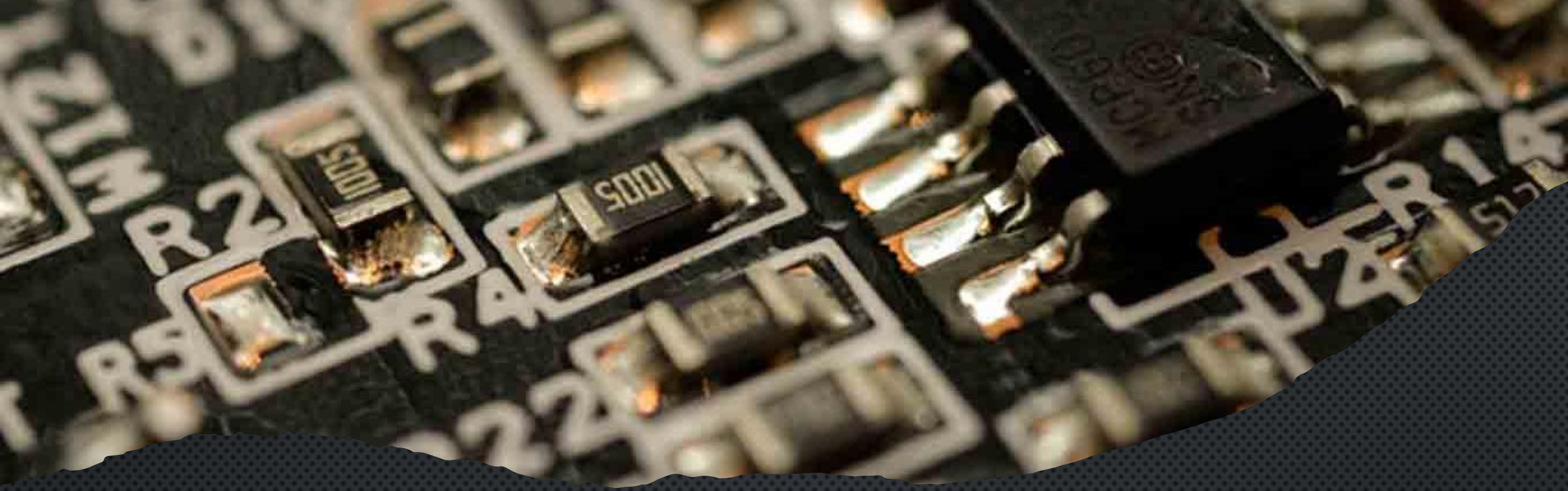
RISC-V PROCESSOR WITH SCOREBOARDING AND REGISTER RENAMING

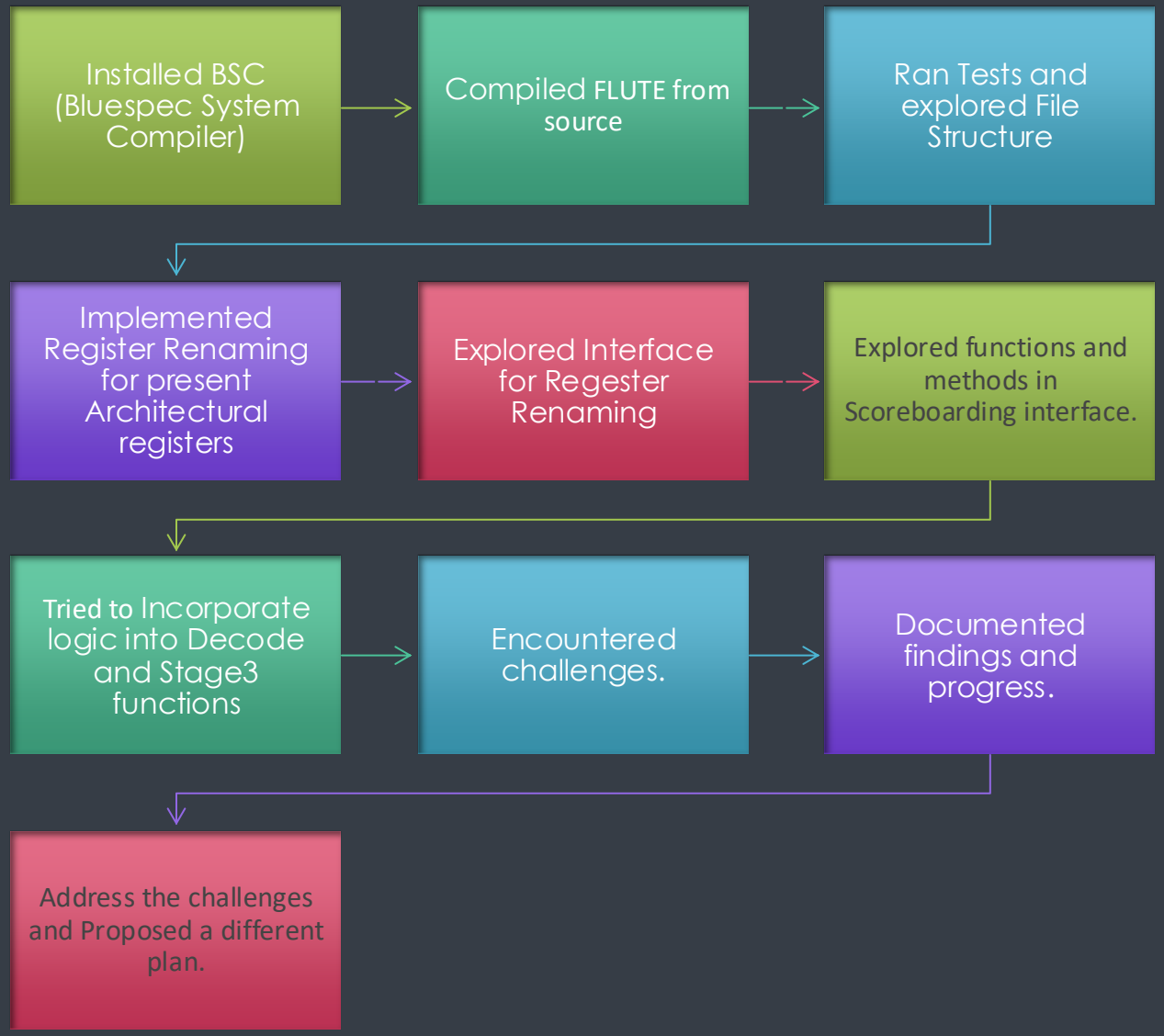
ARYAN | ABHIRAM



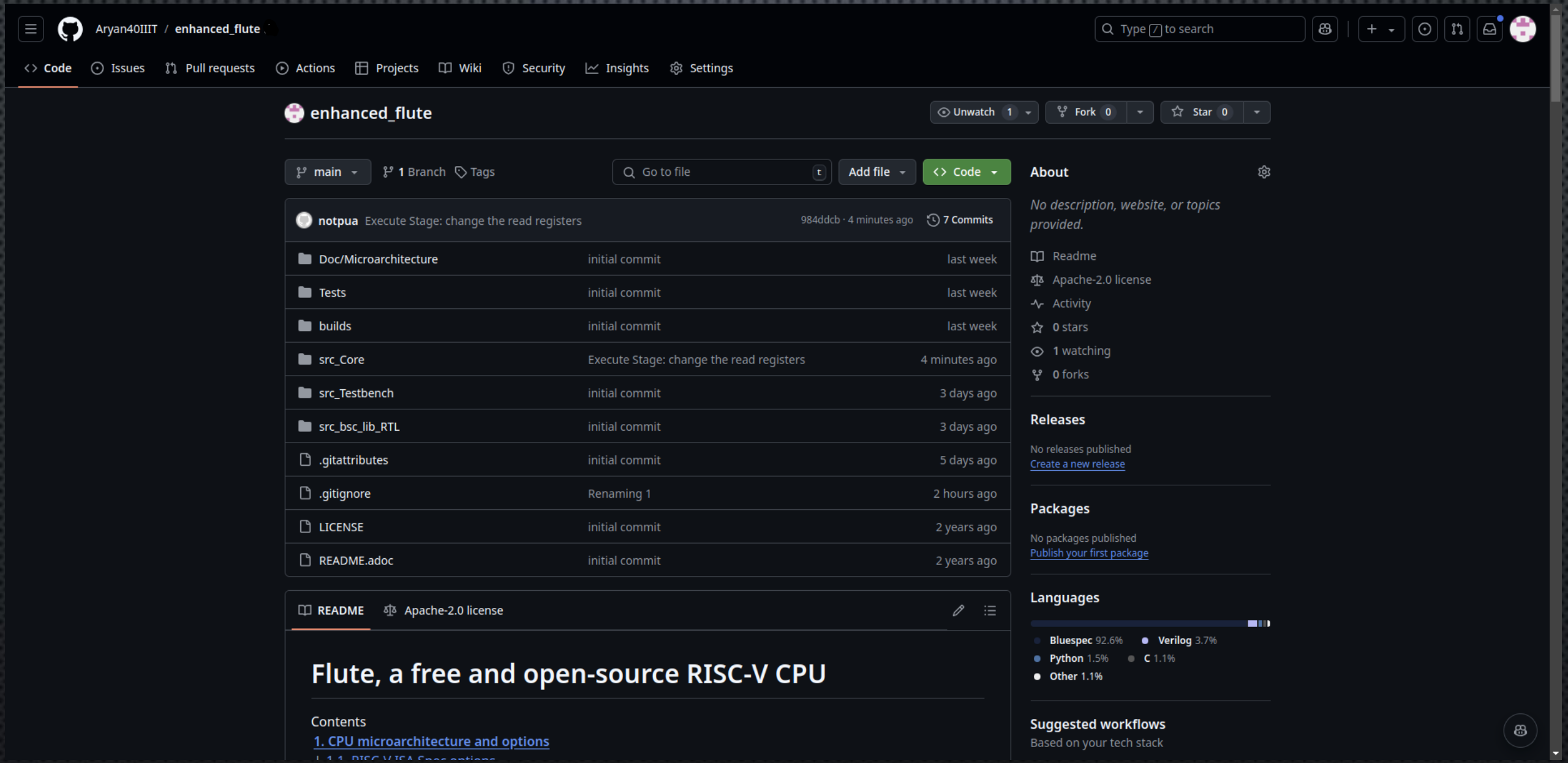
PROJECT DESCRIPTION

- To implement Scoreboarding and Register renaming in Risc-V core.
- To evaluate Performance based on metrics like instruction throughput, execution latency.

OVERALL PROGRESS

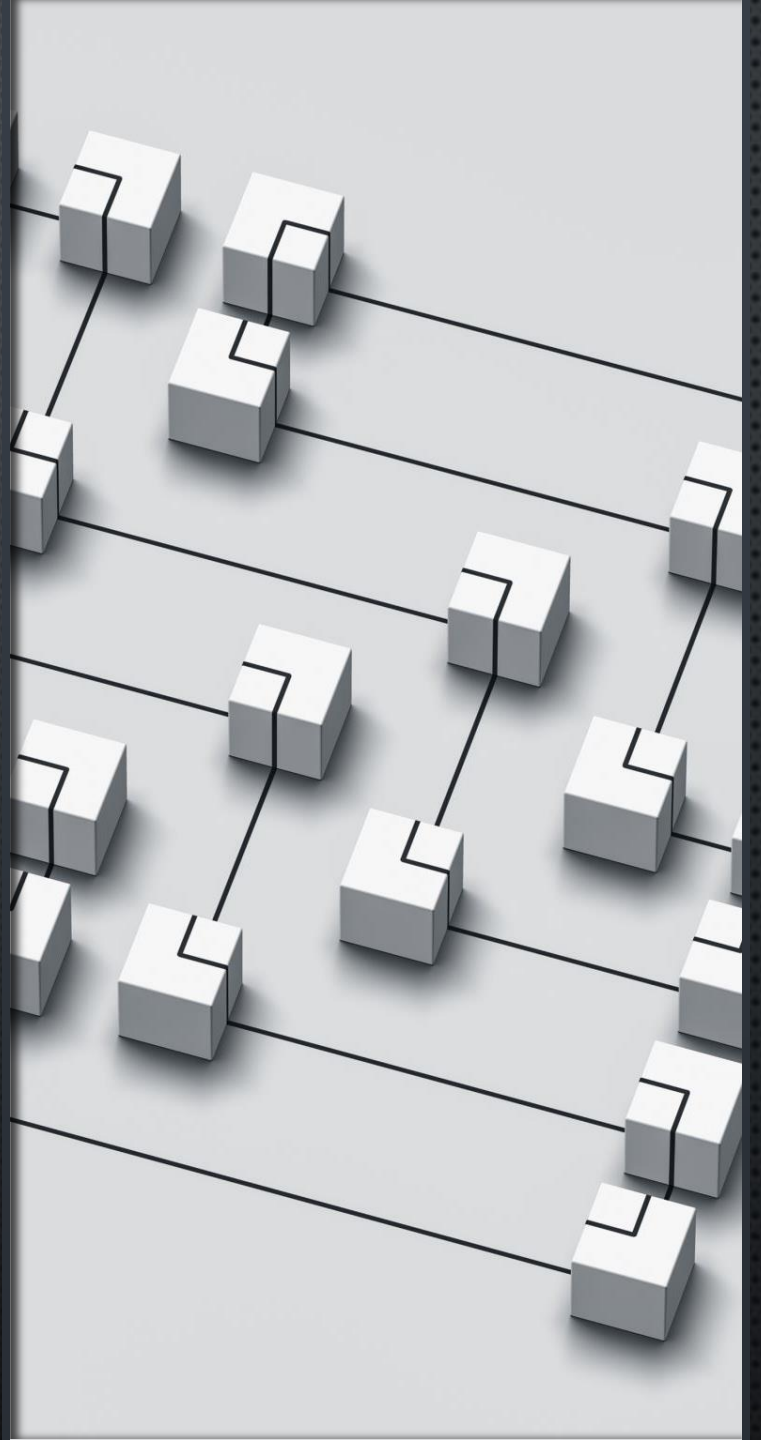


YOU CAN FIND OUR EFFORTS ON GITHUB...



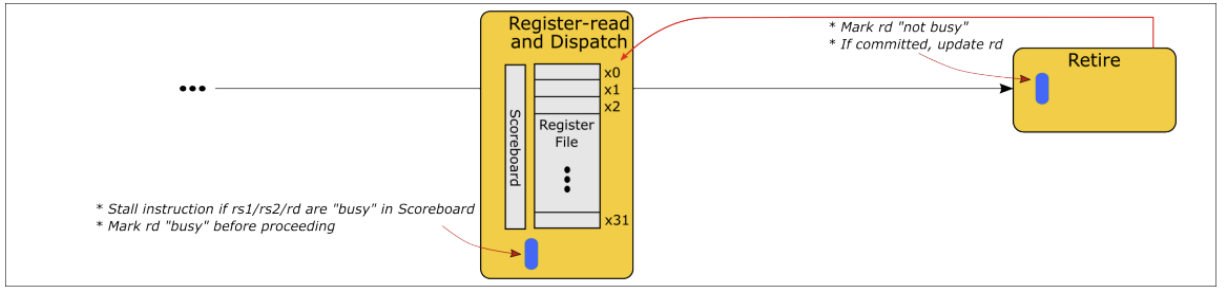
Scoreboarding Implementation

- Design:
  - Implemented as an array for every 1-bit register.
  - Prevents premature execution of instructions.
- Example:
  - Writing to register x7 sets scoreboard bit 7 as busy; subsequent reads stall until resolved.



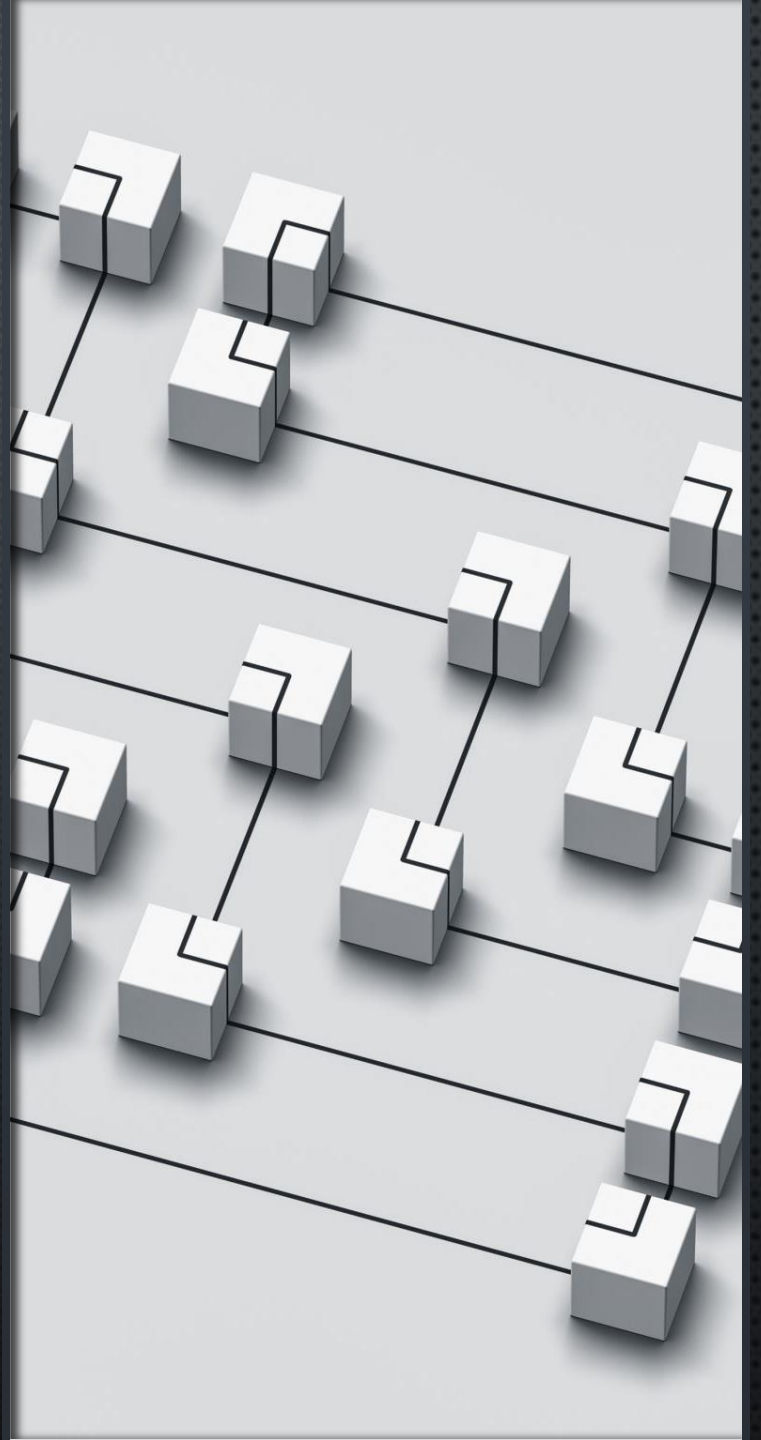
Scoreboarding Implementation

- Steps:
  - Instruction in **Register-Read** stage sets the scoreboard bit to `1` (register marked busy).
  - At the **Retire** stage, scoreboard bit resets to `0` (register free).
  - Dependent instructions stall until the register is free.
  - For All This We Made Changes In Decode Stage And Stage 3



Scoreboarding Implementation Example

- **Goal:**
  - Transition from in-order to out-of-order execution.
- **Current Status:**
  - Although we were unable to fully transition to out-of-order execution, we successfully demonstrated the implementation of proper scoreboarding.
- **Challenges:**
  - Requires additional components like reorder buffers.
  - Significant modifications needed in the execution and commit stages.

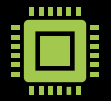


Tried to Add Some Custom Test Cases

- **Purpose:**
  - To test our code, we needed to test that units were being utilized multiple times.

```
+ #include "riscv_test.h"
+ #include "test_macros.h"
+
+ RVTEST_RV64UF
+ RVTEST_CODE_BEGIN
+
+ TEST_FP_OP2_D( 1,  fadd.d, 0,               3.5,        2.5,        1.0 );
+ TEST_FP_OP2_D( 2,  fadd.d, 1,             -1234,    -1235.1,        1.1 );
+ TEST_FP_OP2_D( 3,  fadd.d, 1,         3.14159266, 3.14159265, 0.00000001 );
+
+ TEST_PASSFAIL
+
+ RVTEST_CODE_END
+
+   .data
+ RVTEST_DATA_BEGIN
+
+   TEST_DATA
+
```

Learning Outcomes



Hardware Knowledge: Gained insights into CPU architecture and hardware description languages like Bluespec Verilog.

Pipeline Design: Deepened understanding of pipelining and hazard management.

Techniques: Learned the intricacies of register renaming and scoreboarding mechanisms.

Personal Growth: Enjoyed the challenge of tackling a real-world hardware project.

THINGS WE GOT STUCK IN

Implementing Out-of-Order Execution:

Despite significant efforts, we were unable to fully transition to out-of-order execution. This challenge stemmed from the complexity of modifying the pipeline to support dynamic instruction scheduling and reordering.

THANK YOU