Constructive Computer Architecture

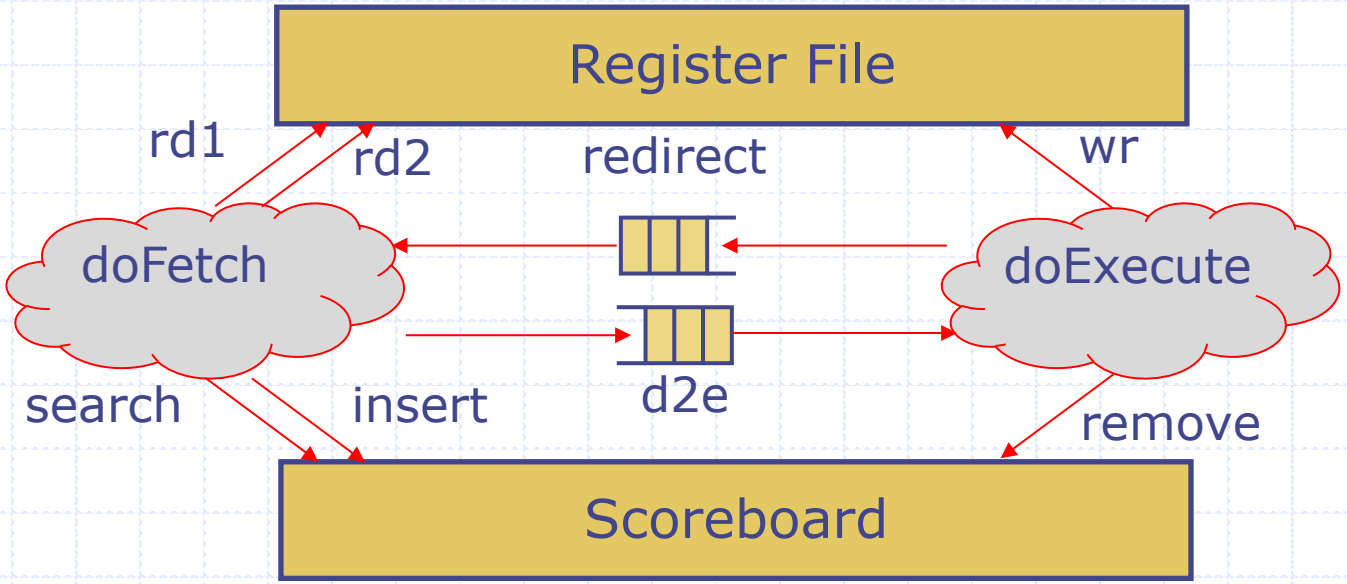
Tutorial 5: Overcoming SMIPS Scheduling Errors

Andy Wright
6.S195 TA

Introduction

- Going to talk about the SMIPS 2 stage pipeline connected by FIFOs
- In the process of speeding it up, we are going to run into some scheduling errors
  - To learn more about these errors we are going to use tools built into the Bluespec GUI
- After we know where the scheduling errors are coming from, we are going to fix them
- Fixing these scheduling errors will be very relevant to lab5!

SMIPS Fetch/Execute Division



Two rules connected by fifos.

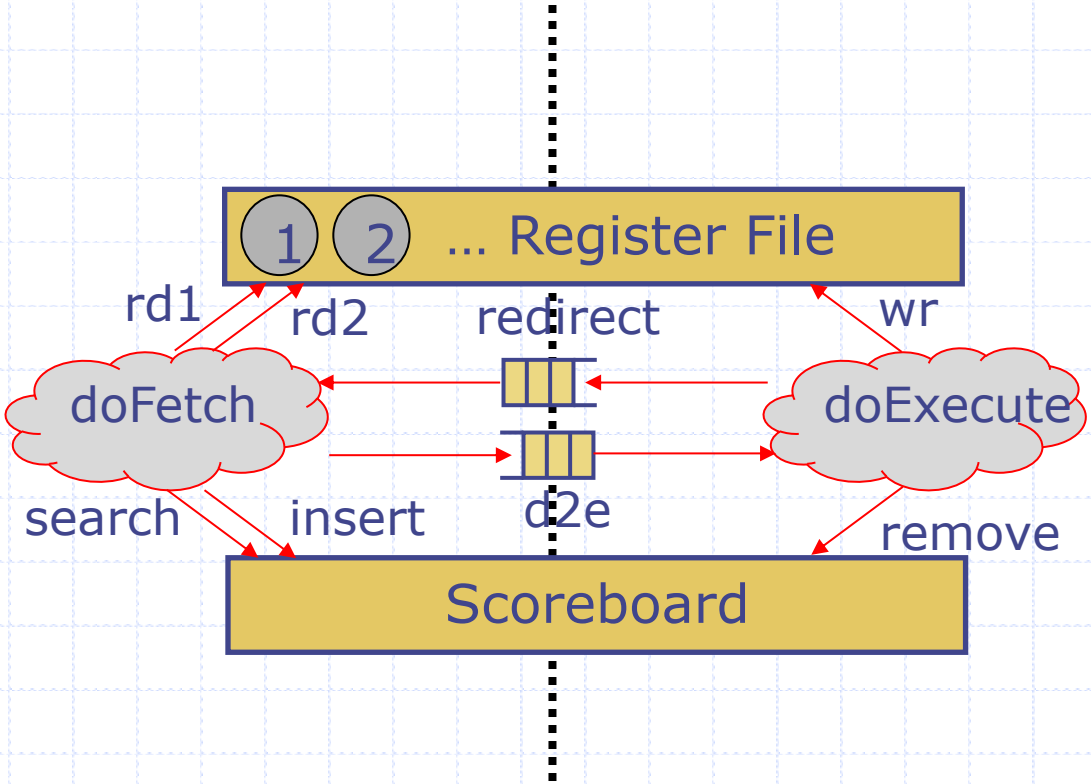
Not very fast, but can be sped up with a few additions.

SMIPS 2 Cycle Pipeline

- Two sources of slowdowns:
  - Control hazards
    - Branch misprediction
    - Seen in Lab 4
    - Can be sped up with bypass redirect fifo or a better branch predictor
  - Read after write (RAW) hazards
    - Scoreboard forces stalling
    - Can be sped up with a bypass register file

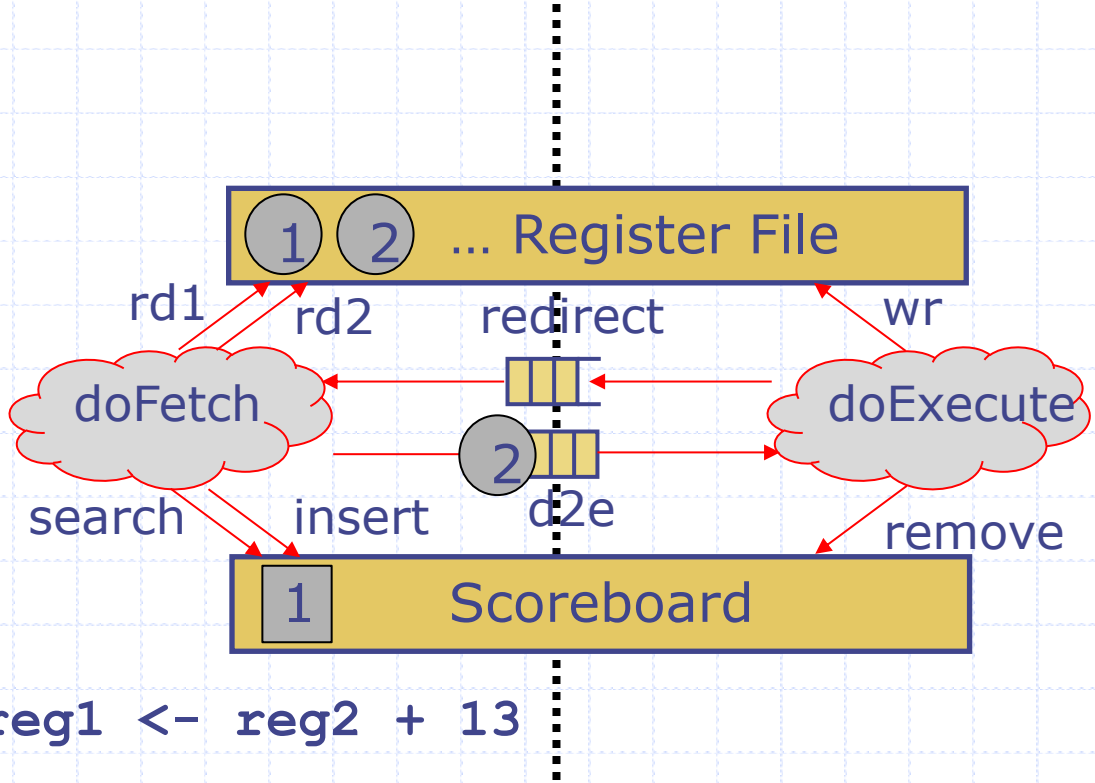
SMIPS Example – RAW

```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



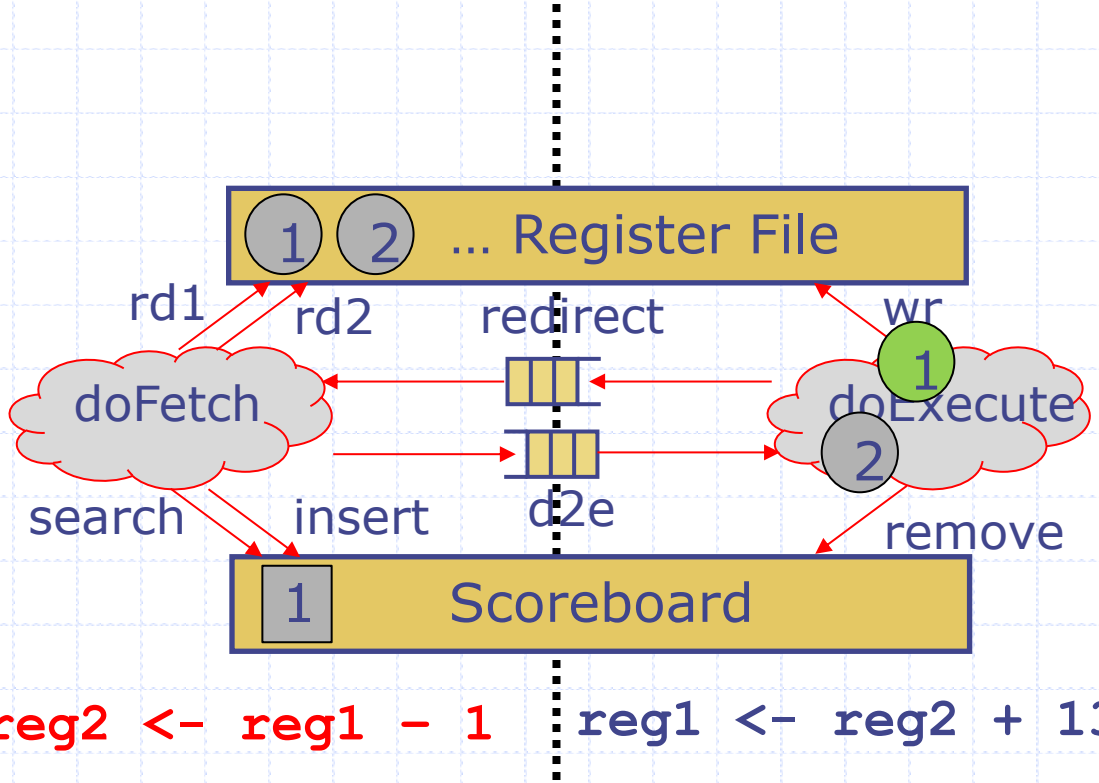
SMIPS Example – Cycle 1

```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



SMIPS Example – Cycle 2

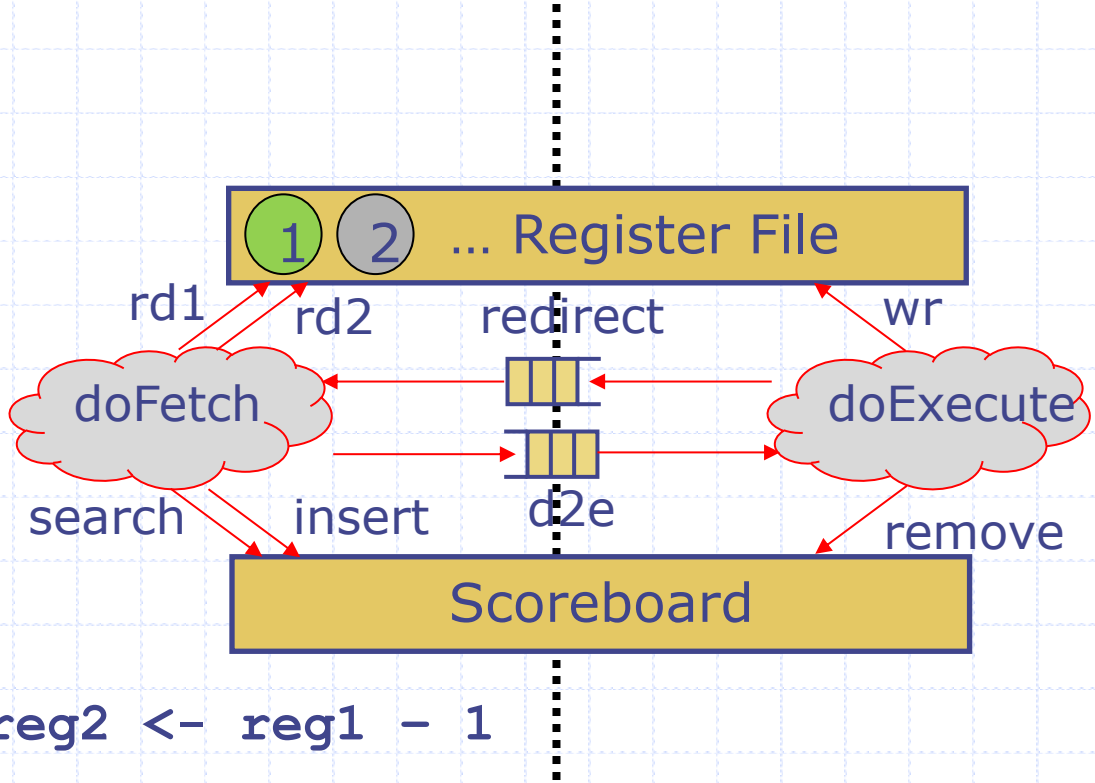
```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



Stall due to Scoreboard!

SMIPS Example – Cycle 3

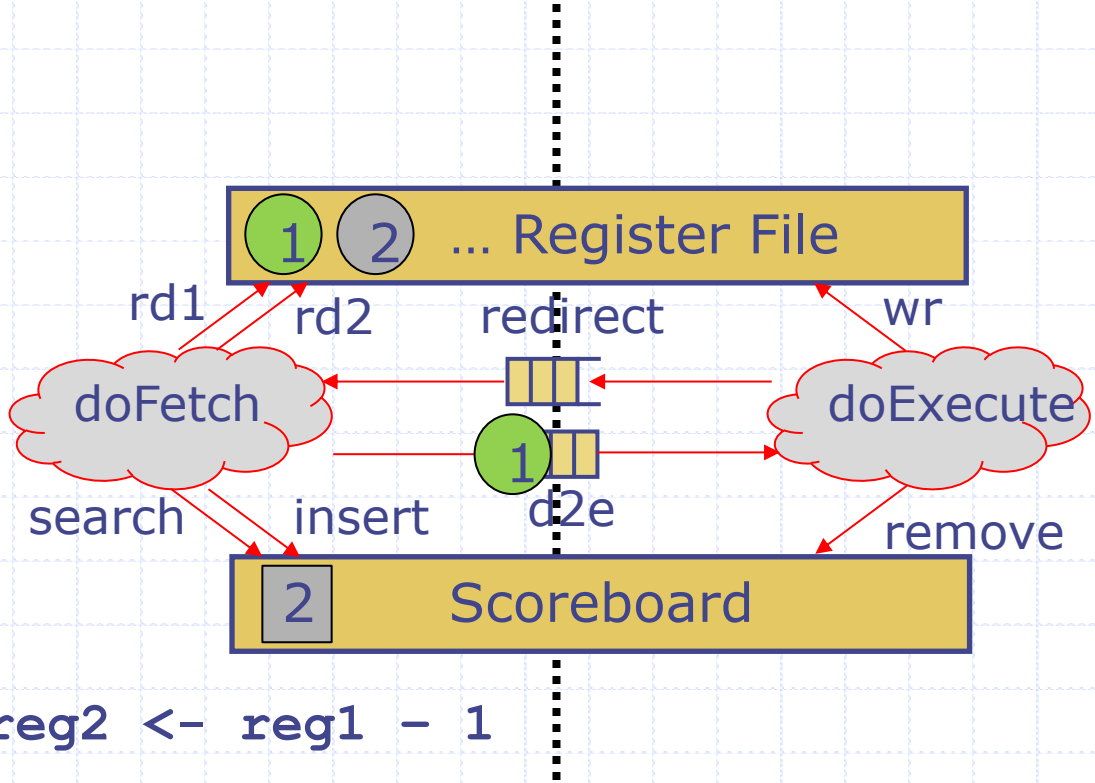
```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



Now this instruction
is free to execute

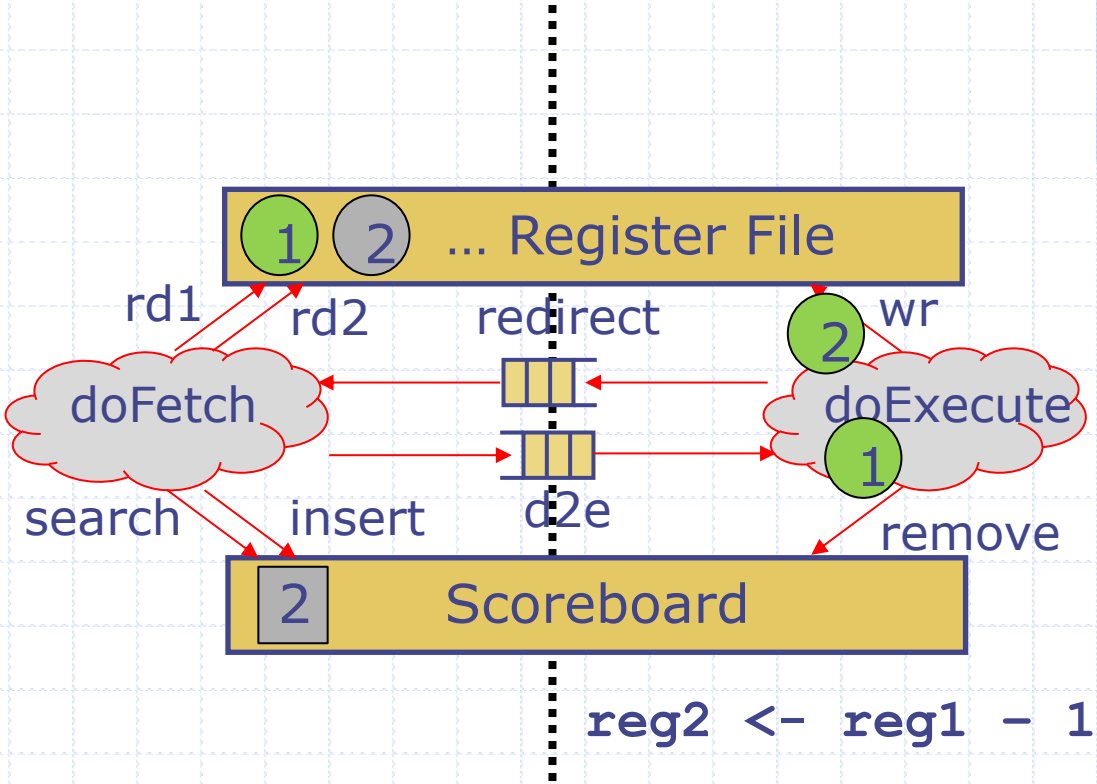
SMIPS Example – Cycle 3

```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



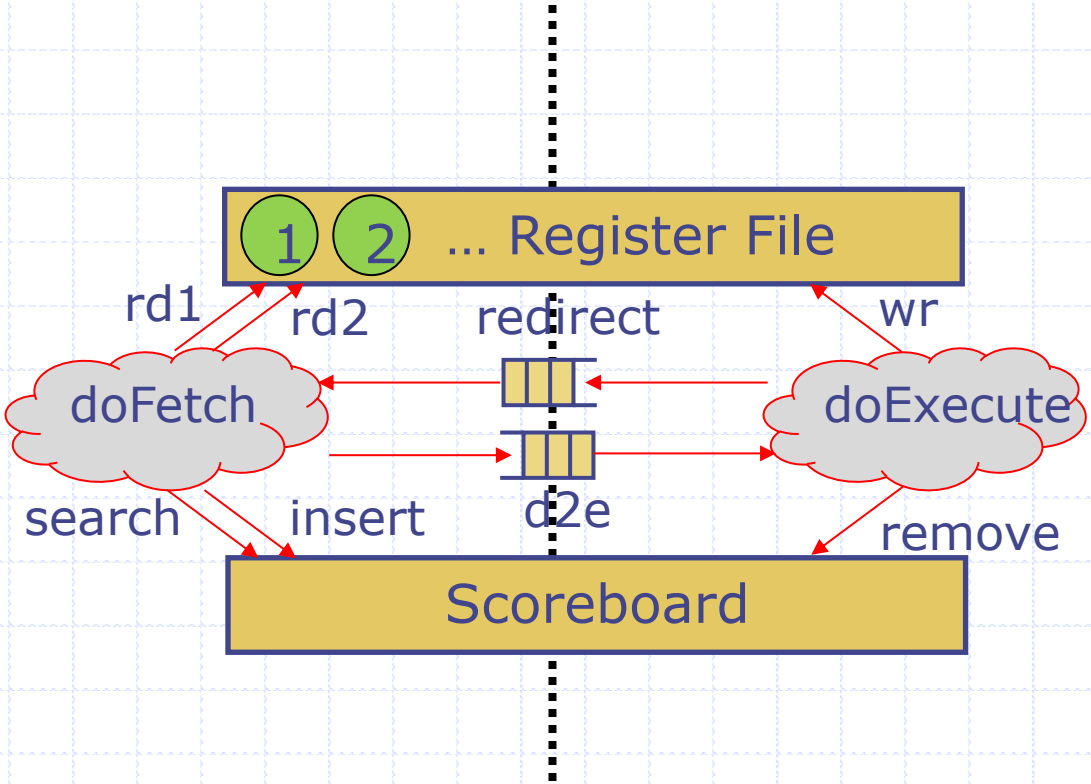
SMIPS Example – Cycle 4

```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



SMIPS Example – Cycle 5

```
reg1 <- reg2 + 13

reg2 <- reg1 -  1
```



Processor Enhancements

- 2 candidates:
  - Bypass register file
    - Reduces RAW hazard penalty, but it introduces a long combinational path
  - Bypass redirect fifo
    - Reduces misprediction penalty without increasing the critical path significantly
- Bypass redirect fifo is a good first step, so lets start there.

Variation 1 – Before improvements

- Standard CF fifos used for fetch to execute fifo and redirect fifo
- Results from simulation can be found in the **orig** folder in the included code

Variation 1 - Results

- Looking at smips.out
  - No scheduling warnings
  - Between 0.5 and 1.0 IPC
  - On average, 1 to 2 cycles per instruction
- Looking at simOut
  - Long misprediction penalty
  - Fetch and Execute fire at the same time

Variation 2 – Shorter Misprediction Penalty?

- This uses a bypass fifo for the redirect fifo in an attempt to reduce the misprediction penalty by a cycle.
- Results from simulation can be found in the **bad** folder in the included code

Variation 2 - Results

- Looking at smips.out
  - No scheduling warnings
  - Less than 0.5 IPC
  - More than 2 cycles per instruction on average
- Looking at simOut
  - Same misprediction penalty
  - Fetch and Execute don’t fire in the same cycle

Scheduling Analysis

- Look at scheduling analysis presentation to see how to use the Bluespec gui to get some additional scheduling information from the compiler.

Register File

- The current design of the register file forces read < write
- We want reads to happen functionally before writes, but we want writes to be scheduled before reads
- Let's modify the register file to be conflict free!

Variation 3 – Shorter Misprediction Penalty!

- This uses a CF register file to go along with the bypass fifo for the redirect fifo
- Results from simulation can be found in the **good** folder in the included code

Variation 3 - Results

- Looking at smips.out
  - No scheduling warnings
  - Between 0.5 and 1.0 IPC
  - Improvement over variations 1 and 2
- Looking at simOut
  - Reduced misprediction penalty
  - Fetch and Execute fire in the same cycle

Conclusion

- We had a plan to speed up the processor, but bluespec scheduling didn't work out
- We used the bluespec gui to debug our scheduling problem
- We resolved a conflicting module by making it conflict free using EHRs and a canonicalize rule
- We ended up getting a faster processor!

Questions?