6.823 Computer System Architecture
Scoreboarding for In-Order Issues

http://csg.csail.mit.edu/6.823/

Detection and resolution of hazards in a complex pipeline is a challenging problem for computer architects. Even in the relatively simple in-order pipeline shown in Figure H8-A, the problem is not trivial. Because every functional unit takes a different number of cycles to complete an execution, instructions write back to the register file *out-of-order*, and Write-After-Read (WAR) or Write-After-Write (WAW) hazards may result. We should also handle Read-After-Write (RAW) hazards appropriately as in the simple 5-stage MIPS pipeline. Equalizing all pipeline depths and bypassing could solve the problem, but they often impose a performance penalty, increase hardware cost, or both.

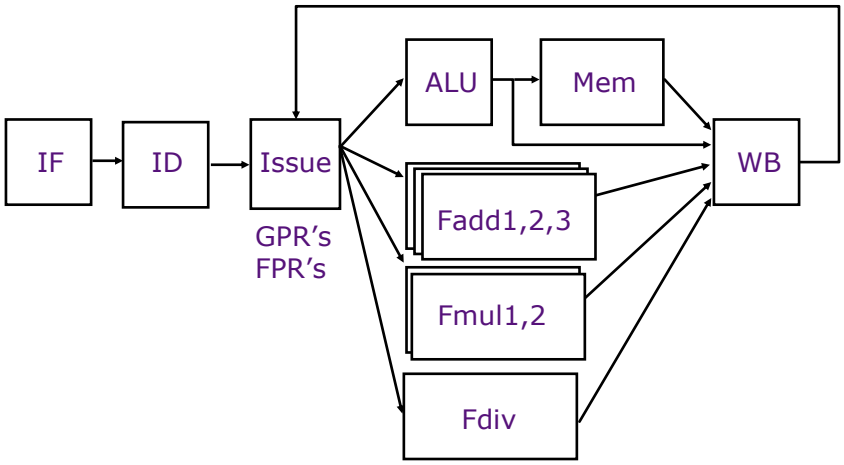

Figure H8-A: In-order MIPS Pipeline with Floating-Point Units

Scoreboarding is a hardware data structure to detect such hazards dynamically (originally introduced in CDC 6600 in 1964). In this handout, we present a simplified form of it to illustrate its operations in the MIPS pipeline shown above. A scoreboard data structure is given in Figure H8-B.

Name	*Busy*	*Op*	*Dest*	*Src1*	*Src2*
Int Mem					
Add1 Add2 Add3					
Mult1 Mult2					
Div					

Figure H8-B: Scoreboard Example

Here we define two bit vectors extracted from this data structure.

Busy [FU#]	A bit vector to indicate each functional unit's availability. (FU := Int | Mem | Add1 | Add2 | Add3 | Mult1 | Mult2 | Div)
WP [Reg#] (Write Pending)	A bit vector to record the registers for which writes are pending. These bits are set to true by the Issue stage, and set to false by the WB stage.

Before getting issued to a functional unit, an instruction in the Issue stage looks up the scoreboard to make sure that the instruction is safe to issue. If any hazard is detected for the instruction, dispatching is delayed until the condition is resolved (and the next instructions are also blocked). The issue logic checks the instruction (opcode dest src1 src2) against the scoreboard (Busy and WP) to detect each type of hazards as follows.

- **Structural Hazard:** This hazard can be detected by checking the Busy bit in the functional unit (FUx) that the instruction is to be issued to. If Busy[FUx] is true, the instruction dispatch is delayed until it is cleared.

- **RAW Hazard:** A RAW hazard exists if any entry in the Dest column in Figure H8-B matches either of the *source* registers of the current instruction (i.e., WP[src1] or WP[src] is true.).

- **WAR Hazard:** This hazard cannot arise here, because the pipeline issues instructions in program order and operands are read on issue.

- **WAW Hazard:** There may be a WAW hazard if any of the Dest column in Figure H8-B matches the *destination* register of the current instruction (i.e., WP[dest] is true).