Exceptions and Interrupts

"Unexpected" events such as exceptions and interrupts require altering control flow. Exceptions occur within the CPU, e.g. undefined opcodes or overflow, while interrupts arise from external I/O controllers. For example, an I/O controller might send an interrupt when a disk finishes reading data, prompting the CPU to switch context to a waiting process

In MIPs, exceptions are handled by the System Control Coprocessor (CP0), saving the program counter in the Exception Program Counter (EPC), indicating problems in the status register.

The exception handling process involves reading the cause, transferring control to the appropriate handler, and determining required actions. If restartable, corrective measures are taken; otherwise, the program is terminated, using the EPC to correct errors.

Pipeline exceptions introduce another form of control hazard. For example, overflow during the execution stage requires preventing register overwrites, completing previous instructions and transferring control to a handler. This approach is similar to handling mispredicted branches. With multiple instructions overlapping in pipelines, simultanes exceptions are possible. Addressing the earliest instruction exception and flushing subsequent instructions is a typical strategy.

Challenges arise when instructions depend on eachother, creating data hazards. Pitfalls in pipelining usually come from poorly designed ISAs, when they are overly complex it makes pipelining harder due to issues like complicated memory access methods, instructions that unexpectedly change registers, and instructions that depend on memory.