Exceptions

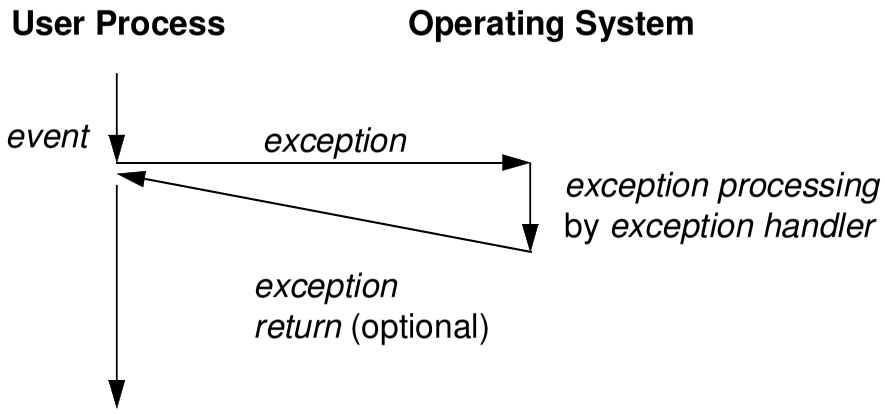
March 7, 1996

Topics

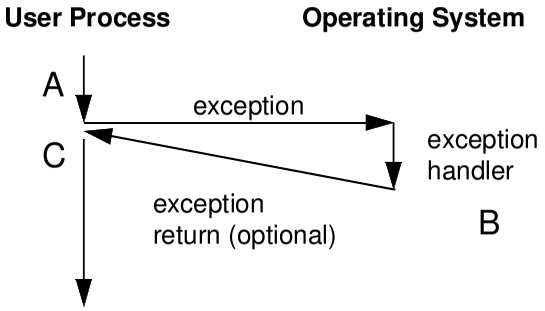
- **basic issues**
- **simple implementation**

Exceptions

An *exception* is a transfer of control to the OS in response to some *event* (i.e. change in processor state)



Issues with exceptions



A1: What kinds of events can cause an exception?

A2: When does the exception occur?

B1: How does the handler determine the location and cause of the exception?

B2: Are exceptions allowed within exception handlers?

C1: Can the user process restart?

C2: If so, where?

Internal (CPU) exceptions

Internal exceptions occur as a result of events generated by executing instructions.

Execution of a SYSCALL instruction.

- **allows a program to ask for OS services (e.g., timer updates)**

Execution of a BREAK instruction

- **used by debuggers**

Errors during instruction execution

- **arithmetic overflow, address error, parity error, undefined instruction**

Events that require OS intervention

- **virtual memory page fault**

External (I/O) exceptions

External exceptions occur as a result of events generated by devices external to the processor.

I/O interrupts

- **hitting ^C at the keyboard**
- **arrival of a packet**
- **arrival of a disk sector**

Hard reset interrupt

- **hitting the reset button**

Soft reset interrupt

- **hitting ctl-alt-delete on a PC**

Exceptions vs interrupts

Much confusion of these terms in the industry.

MIPS terminology:

- **an *exception* refers to *any* control transfer from a user process to the CPU caused by internal or external events.**
- **an *interrupt* refers to an external exception only.**

Exception handling (hardware tasks)

Recognize event(s)

Associate one event with one instruction.

- **external event: pick any instruction**
- **muliple internal events: typically choose the earliest instruction.**
- **multiple external events: prioritize**
- **multiple internal and external events: prioritize**

Abort the instruction.

- **make sure no side effects occur**

Flush any following instructions from the pipeline.

Exception handling (hardware tasks)

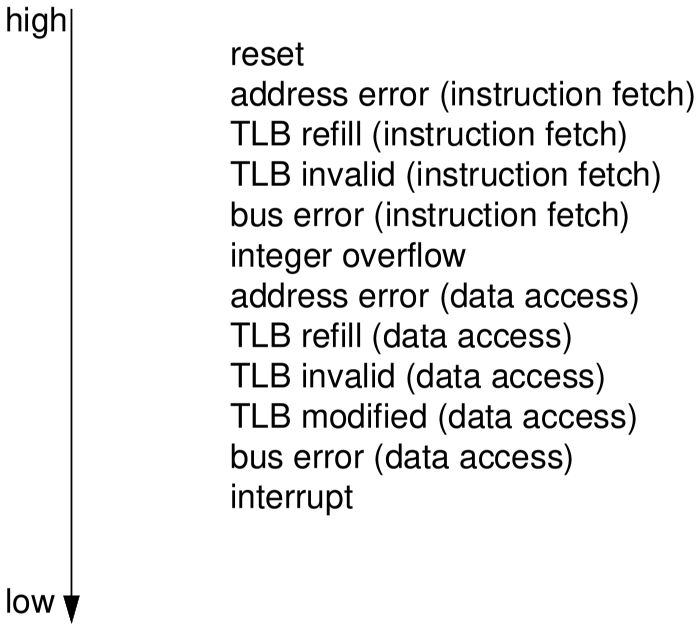
Set status registers

- **EPC register: exception program counter**
  - external exception: address of instruction about to be executed.
  - internal exception (not in the delay slot): address of instruction causing the exception
  - internal exception (in delay slot): address of preceding branch or jump
- **Cause register**
  - records the event that caused the exception
- **Others**
  - 15 other registers! Which get set depends on CPU and exception type

Disable interrupts and switch to kernel mode

Jump to common exception handler location

MIPS R200 exception priority order



Exception handling (software tasks)

Deal with event

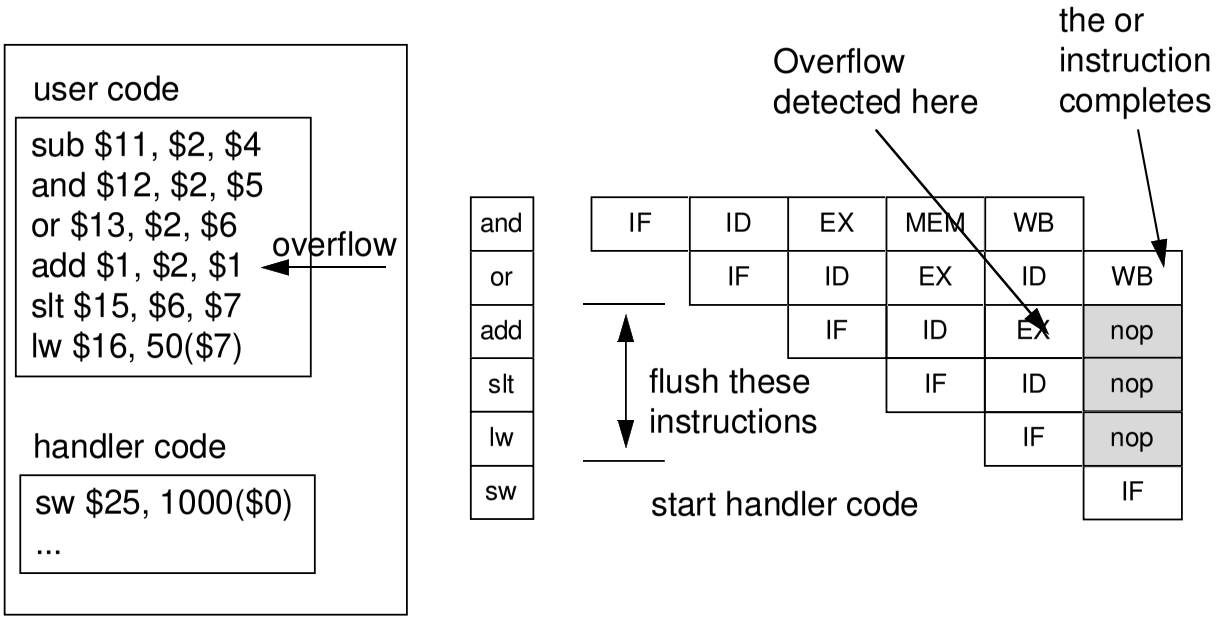
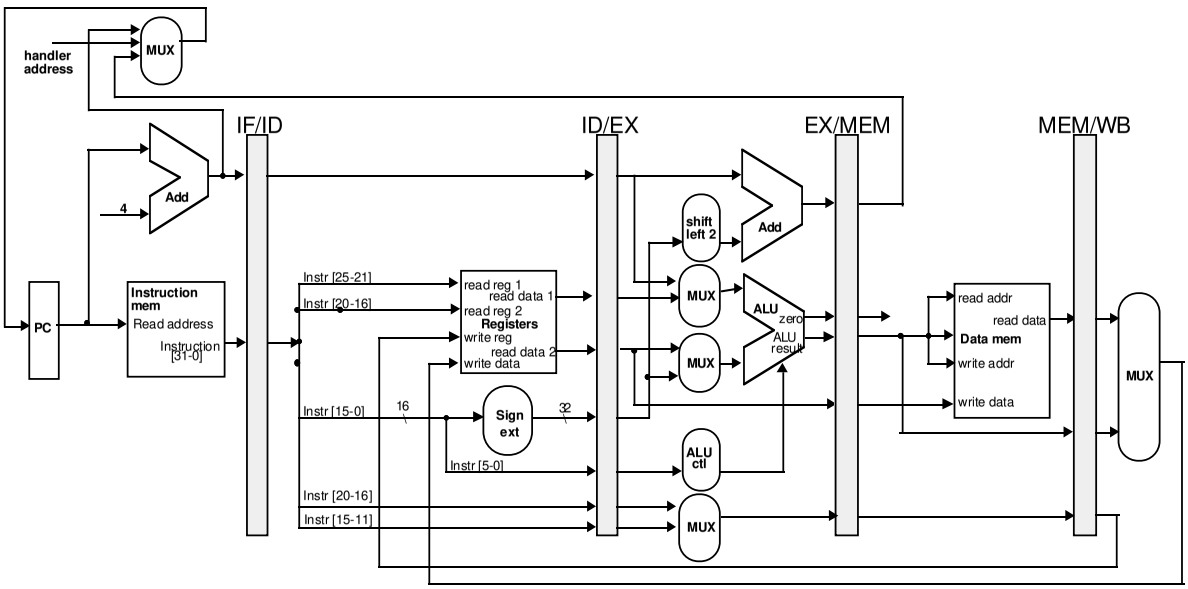
Resume execution

- **using special RFE (return from exception) instruction**
- **similar to jr instruction, but restores processor to user mode as a side effect.**

Where to resume execution?

- **usually re-execute the instruction causing instruction**
- **unless instruction was in branch delay slot, in which case re-execute the branch instruction immediately preceeding the delay slot**

Example: Integer overflow (EX)



Datapath enhanced for exceptions