Exceptions

An exception is an event that alters the normal sequential instruction flow.

- it is not like a branch or jump.
- the PC is automatically loaded with an fixed address in the operating system (OS).
- can change priveledge from a user level to the OS kernel level.

Three main categories of exceptions

1. Processor notices that a process did a bad thing:
   - tried to execute an unknown instruction
   - integer overflow when executing and add, sub, or addi instruction
   - tried to access memory that isn't allowed
   - tried to access an invalid memory address (such as using a null pointer or access a word with an odd address)
2. A program that intentionally generates an exception.
   - system calls (the 'syscall' instruction)
   - implementing breakpoints to help in debugging a program
3. Interrupts
   - External hardware demanding the processor's attention:
     → e.g., I/O devices such as mouse movement, keyboard activity, network controller, hard drive controller, etc.

What happens when there is an exception

The processor will begin executing instructions in an exception handler.

There a couple of extra registers involved:

- EPC (exception program counter)
- Cause register

1. Record the address of the instruction that was executing when the exception occurred. This is stored in the EPC. This is important for:

   (a) continuing the program after the exception handler is finish.
   (b) help in debugging if the exception handler terminates the program.

   The type of the exception is recorded in the Cause register.

2. Immediately begin executing the exception handler code at the fixed address 0x8000_0180 in the OS kernel.

Example:

Suppose we have code running on a real MIPS processor:

```
andi   $t2, $s4, 0xff
sll    $t3, $t2, 8
or     $s2, $s2, $t3
lw     $t1, ($t0)        <-- text-segment address 0x0040_0090
addiu  $t0, $t0, 4
sw     $t1, ($s0)
addiu  $s0, $s0, 4
slt    $t4, $t0, $s7
```

1. Say $t0 = 0x1001_0003 (odd address, so lw causes an exception)

   - previous instructions (andi, sll, or) should be allowed to complete, and then:

     - EPC gets 0x0040_0090
     - Cause register gets code identifying "unaligned word address"
     - later instructions flushed.

2. Say that an external device has caused an interrupt.

- whatever instruction was being executed (called the victim) is flushed along with all later instructions
- The EPC will get the address of the victim, and the Cause register will get code identifying "interrupt".

Challenges in pipelined processor

- completing instructions in the pipeline before the victim.
- flushing the victim and all following instructions from the pipeline
- identifying victim's address, since the current value of PC is pointing to a later instruction.