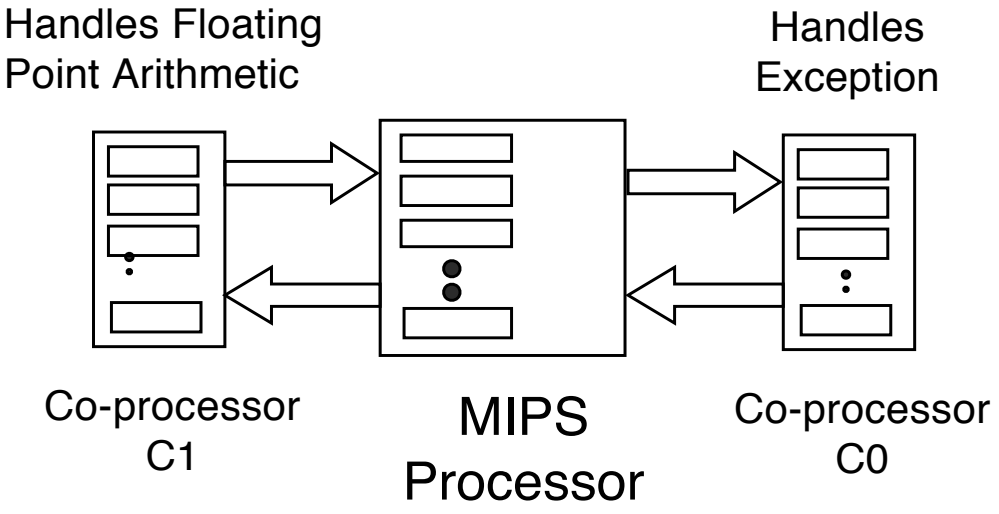
The Basics of Exception Handling



Interrupts

Initiated outside the instruction stream

Arrive asynchronously (at no specific time),

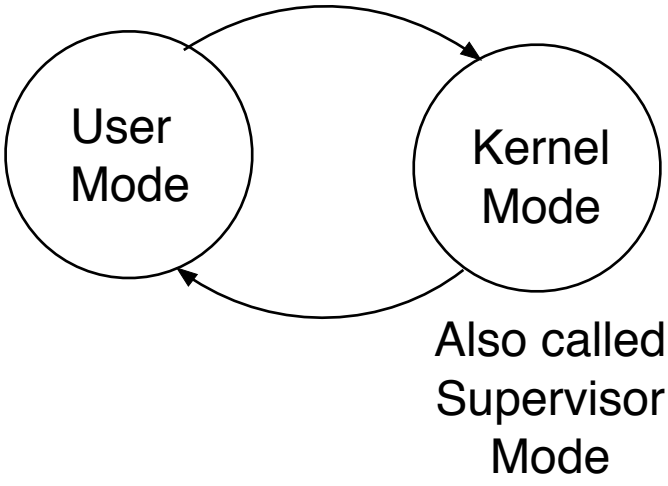
Examples:

- I/O device status change
- I/O device error condition

Traps occur due to something in instruction stream.

Examples:

- Unaligned address error
- Arithmetic overflow
- System call



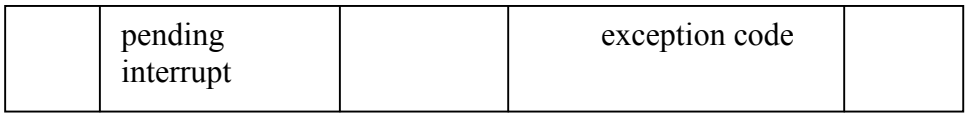
An exception takes away control from the user and transfers it to the supervisor (i.e. the operating system)

Think of the various reasons an exception can occur.

An exception triggers an **unscheduled procedure call**.

Coprocessor C0 has a cause register (Register 13) that contains a 4-bit code to identify the cause of exception

Cause register

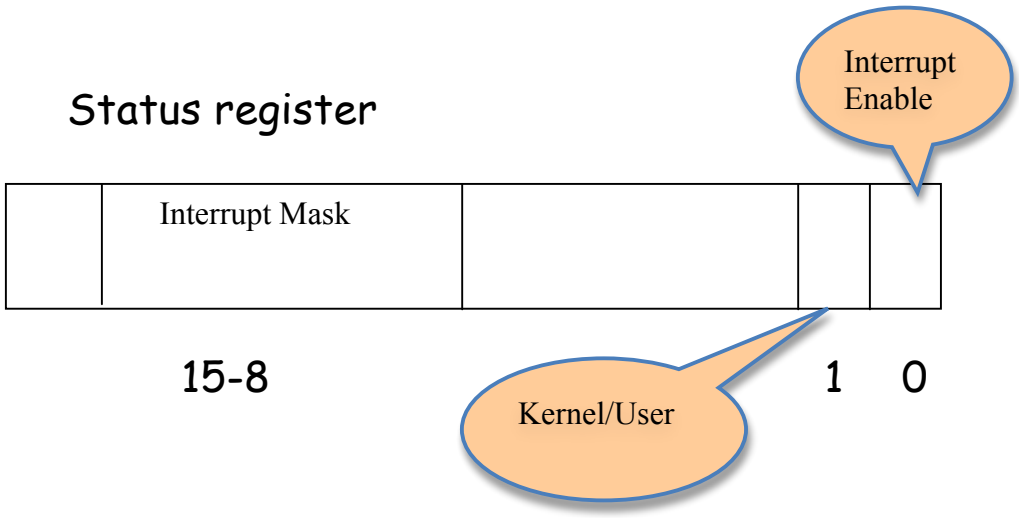


[Exception Code = 0 means I/O interrupt

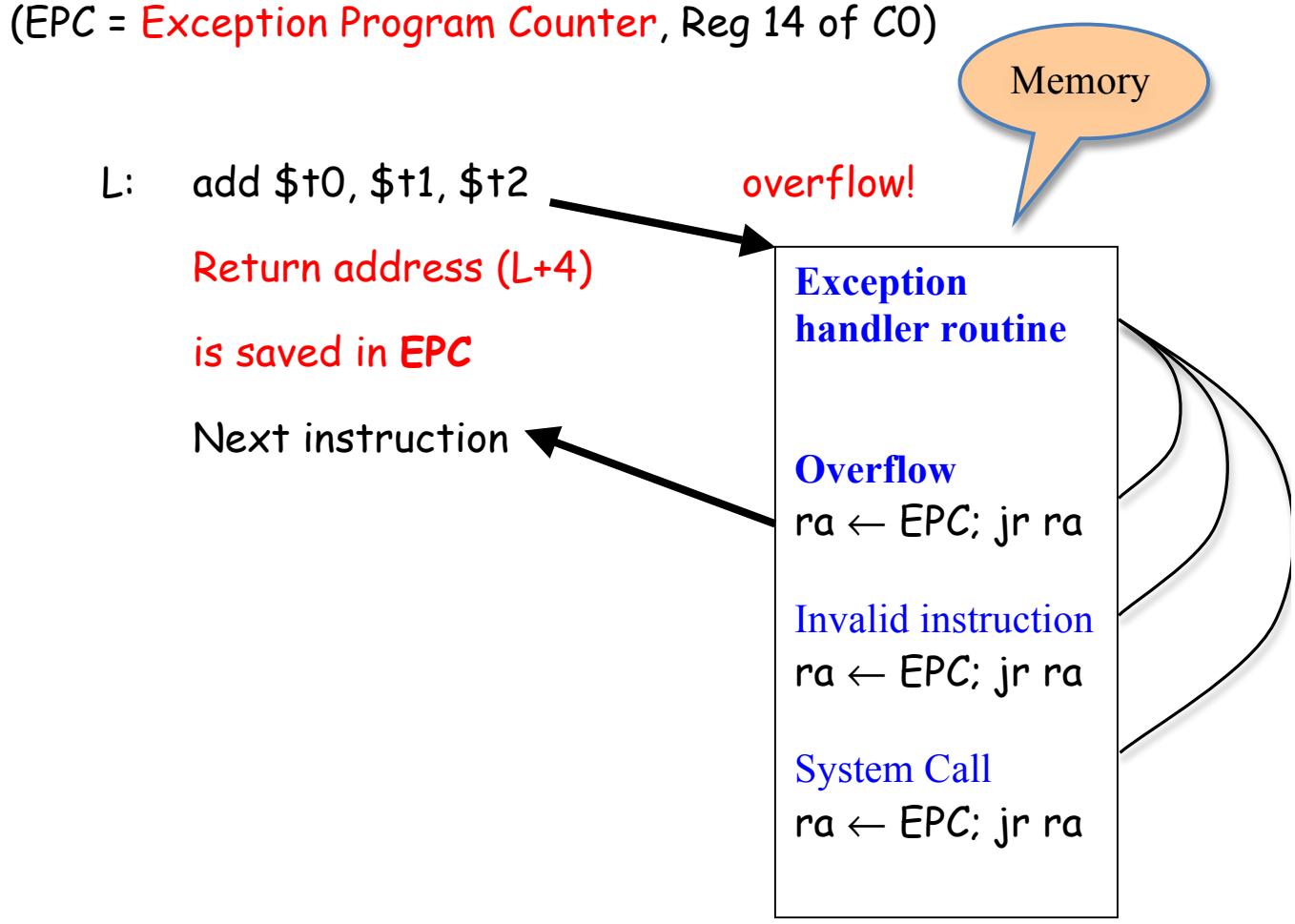
= 12 means arithmetic overflow etc]

MIPS instructions that cause overflow (or some other violation) lead to an ***exception***, which sets the exception code. It then switches to the kernel mode (designated by a bit in the status register of C0, register 12) and transfers control to a predefined address to invoke a routine (exception handler) for handling the exception.

Status register

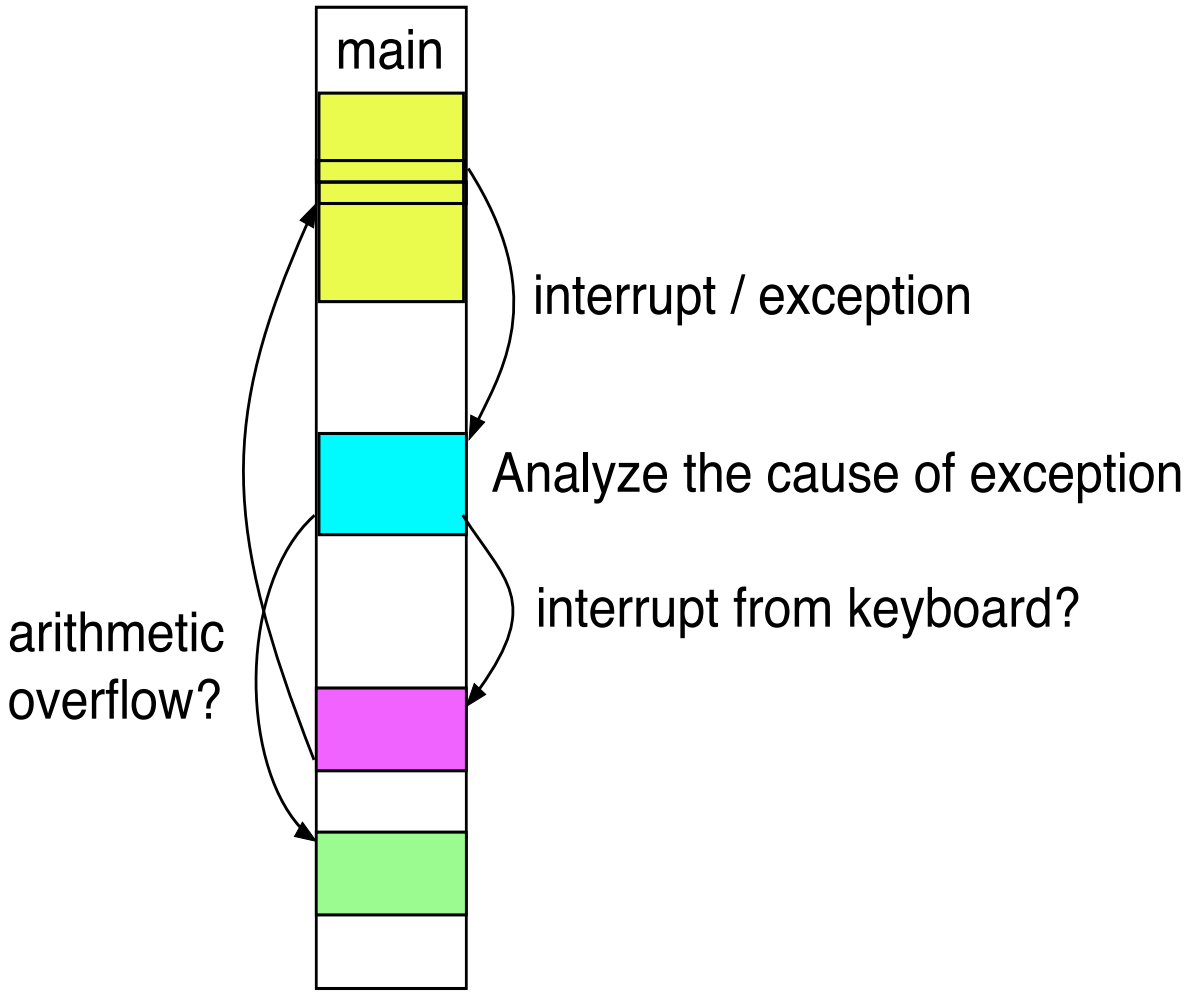


(EPC = Exception Program Counter, Reg 14 of C0)



The Exception Handler determines the cause of the exception by looking at the exception code bits. Then it jumps to the appropriate exception handling routine. Finally, it returns to the main program.

Visualizing Exception Handling



Exceptions cause mostly unscheduled procedure calls.

Example: Read one input from a Keyboard

Consider reading a value from the keyboard. Assume that the interrupt enable bit is set to 1. The first line, ".text 0x80000080" places the code explicitly at the memory location where the *interrupt service routine* is called.

```
.text       0x80000080
            mfc0 $k0, $13          # $k0 = $Cause;
            mfc0 $k1, $14          # $k1 = $EPC;
            andi $k0, $k0, 0x003c # $k0 &= 0x003c (hex);
                                   # Filter the Exception Code;
            bne $k0, $zero, NotIO # if ($k0 ≠ 0) go to NotIO
                                   # Exception Code 0 => I/O instr.
            sw $ra, save0($0)      # save0 = $ra;
            jal ReadByte           # ReadByte(); (Get the byte).
            lw $ra, save0($0)      # $ra = save0;
            jr $k1                 # return;
NotIO:      Other routines here
```

Note that procedure ReadByte must save all registers that it plans to use, and restore them later.