```
# University of Puerto Rico Mayaguez
# INEL 4206 - Microprocessors
# Problem Set 5 - Interrupt-based I/O
#
# SPIM S20 MIPS simulator.
# The default trap handler for spim.
#
# Copyright (C) 1990-2000 James Larus, larus@cs.wisc.edu.
# ALL RIGHTS RESERVED.
#
# SPIM is distributed under the following conditions:
#
# You may make copies of SPIM for your own use and modify those copies.
#
# All copies of SPIM must retain my name and copyright notice.
#
# You may not sell SPIM or distributed SPIM in conjunction with a commerical
# product or service without the expressed written consent of James Larus.
#
# THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
# IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
# PURPOSE.
#

# $Header: $


# Define the exception handling code.  This must go first!

        .kdata
__m1_:  .asciiz "  Exception "
__m2_:  .asciiz "  Exception occurred and ignored\n"
__e0_:  .asciiz "  [Interrupt] \n"
__e1_:  .asciiz "\n"
__e2_:  .asciiz "\n"
__e3_:  .asciiz "\n"
__e4_:  .asciiz "  [Unaligned address in inst/data fetch] \n"
__e5_:  .asciiz "  [Unaligned address in store] \n"
__e6_:  .asciiz "  [Bad address in text read] \n"
__e7_:  .asciiz "  [Bad address in data/stack read] \n"
__e8_:  .asciiz "  [Error in syscall] \n"
__e9_:  .asciiz "  [Breakpoint] \n"
__e10_: .asciiz "  [Reserved instruction] \n"
__e11_: .asciiz "\n"
__e12_: .asciiz "  [Arithmetic overflow] \n"
__e13_: .asciiz "  [Inexact floating point result] \n"
__e14_: .asciiz "  [Invalid floating point result] \n"
__e15_: .asciiz "  [Divide by 0] \n"
__e16_: .asciiz "  [Floating point overflow] \n"
__e17_: .asciiz "  [Floating point underflow] \n"
__excp: .word __e0_,__e1_,__e2_,__e3_,__e4_,__e5_,__e6_,__e7_,__e8_,__e9_
        .word __e10_,__e11_,__e12_,__e13_,__e14_,__e15_,__e16_,__e17_
handler:.asciiz "  Inside interrupt handler\n"
v0:     .word 0
a0:     .word 0

        .ktext 0x80000080
        .set noat
        # Because we are running in the kernel, we can use $k0/$k1 without
        # saving their old values.
        move $k1 $at    # Save $at
        .set at
        sw $v0 v0       # Not re-entrent and we can't trust $sp
        sw $a0 a0
        mfc0 $k0 $13    # Cause
        addu $0 $0 0
        li $v0 4        # syscall 4 (print_str)
        la $a0 __m1_
        syscall
        li $v0 1        # syscall 1 (print_int)
        srl $a0 $k0 2   # shift Cause reg
        andi $a0 $a0 0x0f
        syscall
        li $v0 4        # syscall 4 (print_str)
        lw $a0 __excp($a0)
        syscall
        sgt $v0 $k0 0x44 # ignore interrupt exceptions
        bgtz $v0 ioint  # Handle I/O Interrupts differently
        bne $k0 0x18 ok_pc # Bad PC requires special checks
        mfc0 $a0, $14   # EPC
        and $a0, $a0, 0x3 # Is EPC word-aligned?
        beq $a0, 0, ok_pc
        li $v0 10       # Exit on really bad PC (out of text)
        syscall

ioint:                  # Handle I/O interrupts
                        # Check which device caused the interrupt
                        # Handle display interrupt
                        # Handle keyboard interrupt
                        # Proceed as for any other interrupt
        la $a0, handler
        li $v0, 4
        syscall

ok_pc:
        li $v0 4        # syscall 4 (print_str)
        la $a0 __m2_
        syscall
        mtc0 $0, $13    # Clear Cause register
ret:    lw $v0 v0
        lw $a0 a0
        mfc0 $k0 $14    # EPC
        .set noat
        move $at $k1    # Restore $at
        .set at
        rfe             # Return from exception handler
        addiu $k0 $k0 4 # Return to next instruction
        jr $k0


# Standard startup code.  Invoke the routine main with no arguments.

        .text
        .globl __start
__start:
        lw $a0, 0($sp)  # argc
        addiu $a1, $sp, 4 # argv
        addiu $a2, $a1, 4 # envp
        sll $v0, $a0, 2
        addu $a2, $a2, $v0
        jal main
        li $v0 10
        syscall         # syscall 10 (exit)

        .globl __eoth
__eoth:
```