```
# File:  /usr/local/spim//exceptions.s
# For use with the new version of spim
#############################################################
# SPIM S20 MIPS simulator.
# The default exception handler for spim.
#
# Copyright (C) 1990-2004 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 " occurred and ignored\n"
__e0_:  .asciiz "  [Interrupt] "
__e1_:  .asciiz "  [TLB]"
__e2_:  .asciiz "  [TLB]"
__e3_:  .asciiz "  [TLB]"
__e4_:  .asciiz "  [Address error in inst/data fetch] "
__e5_:  .asciiz "  [Address error in store] "
__e6_:  .asciiz "  [Bad instruction address] "
__e7_:  .asciiz "  [Bad data address] "
__e8_:  .asciiz "  [Error in syscall] "
__e9_:  .asciiz "  [Breakpoint] "
__e10_: .asciiz "  [Reserved instruction] "
__e11_: .asciiz ""
__e12_: .asciiz "  [Arithmetic overflow] "
__e13_: .asciiz "  [Trap] "
__e14_: .asciiz ""
__e15_: .asciiz "  [Floating point] "
__e16_: .asciiz ""
__e17_: .asciiz ""
__e18_: .asciiz "  [Coproc 2]"
__e19_: .asciiz ""
__e20_: .asciiz ""
__e21_: .asciiz ""
__e22_: .asciiz "  [MDMX]"
__e23_: .asciiz "  [Watch]"
__e24_: .asciiz "  [Machine check]"
__e25_: .asciiz ""
__e26_: .asciiz ""
__e27_: .asciiz ""
__e28_: .asciiz ""
__e29_: .asciiz ""
__e30_: .asciiz "  [Cache]"
__e31_: .asciiz ""
__excp: .word __e0_, __e1_, __e2_, __e3_, __e4_, __e5_, __e6_, __e7_,
            __e8_, __e9_
        .word __e10_, __e11_, __e12_, __e13_, __e14_, __e15_, __e16_,
            __e17_, __e18_,
        .word __e19_, __e20_, __e21_, __e22_, __e23_, __e24_, __e25_,
```



```
            __e26_, __e27_,
        .word __e28_, __e29_, __e30_, __e31_
s1:     .word 0
s2:     .word 0

# This is the exception handler code that the processor runs when
# an exception occurs. It only prints some information about the
# exception, but can server as a model of how to write a handler.
#
# Because we are running in the kernel, we can use $k0/$k1 without
# saving their old values.

# This is the exception vector address for MIPS-1 (R2000):
#       .ktext 0x80000080
# This is the exception vector address for MIPS32:
        .ktext 0x80000180
# Select the appropriate one for the mode in which SPIM is compiled.
        .set noat
        move $k1 $at            # Save $at
        .set at
        sw $v0 s1               # Not re-entrent and we can't trust $sp
        sw $a0 s2               # But, we need to use these registers

        mfc0 $k0 $13            # Cause register
        srl $a0 $k0 2           # Extract ExcCode Field
        andi $a0 $a0 0xf

        # Print information about exception.
        #
        li $v0 4                # syscall 4 (print_str)
        la $a0 __m1_
        syscall

        li $v0 1                # syscall 1 (print_int)
        srl $a0 $k0 2           # Extract ExcCode Field
        andi $a0 $a0 0xf
        syscall

        li $v0 4                # syscall 4 (print_str)
        andi $a0 $k0 0x3c
        lw $a0 __excp($a0)
        nop
        syscall

        bne $k0 0x18 ok_pc      # Bad PC exception requires special checks
        nop

        mfc0 $a0 $14            # EPC
        andi $a0 $a0 0x3        # Is EPC word-aligned?
        beq $a0 0 ok_pc
        nop

        li $v0 10               # Exit on really bad PC
        syscall

ok_pc:
        li $v0 4                # syscall 4 (print_str)
        la $a0 __m2_
        syscall

        srl $a0 $k0 2           # Extract ExcCode Field
        andi $a0 $a0 0xf
        bne $a0 0 ret           # 0 means exception was an interrupt
        nop

# Interrupt-specific code goes here!
# Don't skip instruction at EPC since it has not executed.
```

```
ret:
# Return from (non-interrupt) exception. Skip offending instruction
# at EPC to avoid infinite loop.
#
        mfc0 $k0 $14            # Bump EPC register
        addiu $k0 $k0 4         # Skip faulting instruction
                                # (Need to handle delayed branch case here)
        mtc0 $k0 $14


# Restore registers and reset procesor state
#
        .set noat
        move $at $k1            # Restore $at
        .set at
        lw $v0 s1               # Restore other registers
        lw $a0 s2

        mtc0 $0 $13             # Clear Cause register

        mfc0 $k0 $12            # Set Status register
        ori  $k0 0x1            # Interrupts enabled
        mtc0 $k0 $12

# Return from exception on MIPS32:
        eret

# Return sequence for MIPS-I (R2000):
#       rfe                     # Return from exception handler
                                # Should be in jr's delay slot
#       jr $k0
#        nop



# Standard startup code.  Invoke the routine "main" with arguments:
#       main(argc, argv, envp)
#
        .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
        nop

        li $v0 10
        syscall                 # syscall 10 (exit)

        .globl __eoth
__eoth:
```