**Writing and Using MIPS exception handlers in MARS**

**Introduction**

*Exception handlers*, also known as *trap handlers* or *interrupt handlers*, can easily be incorporated into a MIPS program. This guide is not intended to be comprehensive but provides the essential information for writing and using exception handlers.

Although the same mechanism services all three, *exceptions*, *traps* and *interrupts* are all distinct from each other. Exceptions are caused by exceptional conditions that occur at runtime such as invalid memory address references. Traps are caused by instructions constructed especially for this purpose, listed below. Interrupts are caused by external devices.

MARS partially but not completely implements the exception and interrupt mechanism of SPIM.

**Essential Facts**

Some essential facts about writing and using exception handlers include:

* MARS simulates basic elements of the MIPS32 exception mechanism.
* The MIPS instruction set includes a number of instructions that conditionally trigger a trap exception based on the relative values of two registers or of a constant and a register: teq, teqi (trap if equal), tne, tnei (trap if not equal), tge, tgeu, tgei, tgeiu (trap if greater than or equal), tlt, tltu, tlti, tltiu (trap if less than)
* When an exception occurs,
  1. Coprocessor 0 register $12 (status) bit 1 is set
  2. Coprocessor 0 register $13 (cause) bits 2-6 are set to the exception type (codes below)
  3. Coprocessor 0 register $14 (epc) is set to the address of the instruction that triggered the exception
  4. If the exception was caused by an invalid memory address, Coprocessor 0 register $8 (vaddr) is set to the invalid address.
  5. Execution flow jumps to the MIPS instruction at memory location 0x800000180. This address in the kernel text segment (.ktext directive) is the standard MIPS32 exception handler location. The only way to change it in MARS is to change the MIPS memory configuration through the Settings menu item Memory Configuration.
* There are three ways to include an exception handler in a MIPS program
  1. Write the exception handler in the same file as the regular program. An example of this is presented below.
  2. Write the exception handler in a separate file, store that file in the same directory as the regular program, and select the Settings menu item "Assemble all files in directory"
  3. Write the exception handler in a separate file, store that file in any directory, then open the "Exception Handler..." dialog in the Settings menu, check the check box and browse to that file.
* If there is no instruction at location 0x800000180, MARS will terminate the MIPS program with an appropriate error message.
* The exception handler can return control to the program using the eret instruction. This will place the EPC register $14 value into the Program Counter, so be sure to increment $14 by 4 before returning to skip over the instruction that caused the exception. The mfc0 and mtc0 instructions are used to read from and write to Coprocessor 0 registers.
* Bits 8-15 of the Cause register $13 can also be used to indicate pending interrupts. Currently this is used only by the Keyboard and Display Simulator Tool, where bit 8 represents a keyboard interrupt and bit 9 represents a display interrupt. For more details, see the Help panel for that Tool.
* Exception types declared in mars.simulator.Exceptions, but not necessarily implemented, are ADDRESS\_EXCEPTION\_LOAD (4), ADDRESS\_EXCEPTION\_STORE (5), SYSCALL\_EXCEPTION (8), BREAKPOINT\_EXCEPTION (9), RESERVED\_INSTRUCTION\_EXCEPTION (10), ARITHMETIC\_OVERFLOW\_EXCEPTION (12), TRAP\_EXCEPTION(13), DIVIDE\_BY\_ZERO\_EXCEPTION (15), FLOATING\_POINT\_OVERFLOW (16), and FLOATING\_POINT\_UNDERFLOW (17).
* When writing a non-trivial exception handler, your handler must first save general purpose register contents, then restore them before returning.

**Example of Trap Handler**

The sample MIPS program below will immediately generate a trap exception because the trap condition evaluates true, control jumps to the exception handler, the exception handler returns control to the instruction following the one that triggered the exception, then the program terminates normally.

.text

main:

teqi $t0,0 # immediately trap because $t0 contains 0

li $v0, 10 # After return from exception handler, specify exit service

syscall # terminate normally

# Trap handler in the standard MIPS32 kernel text segment

.ktext 0x80000180

move $k0,$v0 # Save $v0 value

move $k1,$a0 # Save $a0 value

la $a0, msg # address of string to print

li $v0, 4 # Print String service

syscall

move $v0,$k0 # Restore $v0

move $a0,$k1 # Restore $a0

mfc0 $k0,$14 # Coprocessor 0 register $14 has address of trapping instruction

addi $k0,$k0,4 # Add 4 to point to next instruction

mtc0 $k0,$14 # Store new address back into $14

eret # Error return; set PC to value in $14

.kdata

msg:

.asciiz "Trap generated"

**Widely Used Exception Handler**

The exception handler exceptions.s provided with the SPIM simulator will assemble and run under MARS. The MARS assembler will generate warnings because this program contains directives that it does not recognize, but as long as the setting "Assembler warnings are considered errors" is *not* set this will not cause any problems.