Save state			
Save GPR	addi	$k1,$sp, -XCPSIZE	# save space on stack for state
	sw	$sp, XCT_SP($k1)	# save $sp on stack
	sw	$v0, XCT_V0($k1)	# save $v0 on stack
	...		# save $v1, $ai, $si, $ti,... on stack
	sw	$ra, XCT_RA($k1)	# save $ra on stack
Save hi, lo	mfhi	$v0	# copy Hi
	mflo	$v1	# copy Lo
	sw	$v0, XCT_HI($k1)	# save Hi value on stack
	sw	$v1, XCT_LO($k1)	# save Lo value on stack
Save exception registers	mfc0	$a0, $cr	# copy cause register
	sw	$a0, XCT_CR($k1)	# save $cr value on stack
	...		# save $v1,....
	mfc0	$a3, $sr	# copy status register
	sw	$a3, XCT_SR($k1)	# save $sr on stack
Set sp	move	$sp, $k1	# sp = sp - XCPSIZE
Enable nested exceptions			
	andi	$v0, $a3, MASK1	# $v0 = $sr & MASK1, enable exceptions
	mtc0	$v0, $sr	# $sr = value that enables exceptions
Call C exception handler			
Set `$gp`	move	$gp, GPINIT	# set $gp to point to heap area
Call C code	move	$a0, $sp	# arg1 = pointer to exception stack
	jal	xcpt_deliver	# call C code to handle exception
Restoring state			
Restore most GPR, `hi, lo`	move	$at, $sp	# temporary value of $sp
	lw	$ra, XCT_RA($at)	# restore $ra from stack
	...		# restore $t0,...., $a1
	lw	$a0, XCT_A0($k1)	# restore $a0 from stack
Restore status register	lw	$v0, XCT_SR($at)	# load old $sr from stack
	li	$v1, MASK2	# mask to disable exceptions
	and	$v0, $v0, $v1	# $v0 = $sr & MASK2, disable exceptions
	mtc0	$v0, $sr	# set status register
Exception return			
Restore `$sp` and rest of GPR used as temporary registers	lw	$sp, XCT_SP($at)	# restore $sp from stack
	lw	$v0, XCT_V0($at)	# restore $v0 from stack
	lw	$v1, XCT_V1($at)	# restore $v1 from stack
	lw	$k1, XCT_EPC($at)	# copy old $epc from stack
	lw	$at, XCT_AT($at)	# restore $at from stack
Restore `ERC` and return	mtc0	$k1, $epc	# restore $epc
	eret	$ra	# return to interrupted instruction

FIGURE 5.33 MIPS code to save and restore state on an exception.