| Method body | | | | |
|---|---|---|---|---|
| Move parameters | | move | $s2, $a0 | # copy parameter $a0 into $s2 (save $a0) |
| Test ptr null | | beq | $a0,$zero,NullPointer | # if $a0==0, goto Error |
| Get array length | | lw | $s3,4($a0) | # $s3 = length of array v |
| Outer loop | for1tst: | move<br>slt<br>beq | $s0, $zero<br>$t0, $s0, $s3<br>$t0, $zero, exit1 | # i = 0<br># reg $t0 = 0 if $s0 Š $s3 (i Š n)<br># go to exit1 if $s0 Š $s3 (i Š n) |
| Inner loop start | for2tst: | addi<br>slti<br>bne | $s1, $s0, -1<br>$t0, $s1, 0<br>$t0, $zero, exit2 | # j = i - 1<br># reg $t0 = 1 if $s1 < 0 (j < 0)<br># go to exit2 if $s1 < 0 (j < 0) |
| Test if j too big | | slt<br>beq | $t0,$s1,$s3<br>$t0,$zero,IndexOutOfBounds | # Temp reg $t0 = 0 if j >= length<br># if j >= length, goto Error |
| Get v[j] | | sll<br>add<br>lw | $t1, $s1, 2<br>$t2, $s2, $t1<br>$t3, 0($t2) | # reg $t1 = j * 4<br># reg $t2 = v + (j * 4)<br># reg $t3 = v[j] |
| Test if j+1 < 0<br>or if j+1 too big | | addi<br>slt<br>bne<br>slt<br>beq | $t1,$s1,1<br>$t0,$t1,$zero<br>$t0,$zero,IndexOutOfBounds<br>$t0,$t1,$s3<br>$t0,$zero,IndexOutOfBounds | # Temp reg $t1 = j+1<br># Temp reg $t0 = 1 if j+1 < 0<br># if j+1 < 0, goto Error<br># Temp reg $t0 = 0 if j+1 >= length<br># if j+1 >= length, goto Error |
| Get v[j+1] | | lw | $t4, 4($t2) | # reg $t4 = v[j + 1] |
| Load method table | | lw | $t5,0($a0) | # $t5 = address of method table |
| Get method addr | | lw | $t5,8($t5) | # $t5 = address of first method |
| Pass parameters | | move<br>move | $a0, $t3<br>$a1, $t4 | # 1st parameter of compareTo is v[j]<br># 2nd param. of compareTo is v[j+1] |
| Set return addr | | la | $ra,L1 | # load return address |
| Call indirectly | | jr | $t5 | # call code for compareTo |
| Test if should skip swap | L1: | slt<br>beq | $t0, $zero, $v0<br>$t0, $zero, exit2 | # reg $t0 = 0 if 0 Š $v0<br># go to exit2 if $t4 Š $t3 |
| Pass parameters and call swap | | move<br>move<br>jal | $a0, $s2<br>$a1, $s1<br>swap | # 1st parameter of swap is v<br># 2nd parameter of swap is j<br># swap code shown in Figure 2.34 |
| Inner loop end | | addi<br>j | $s1, $s1, -1<br>for2tst | # j -= 1<br># jump to test of inner loop |
| Outer loop | exit2: | addi<br>j | $s0, $s0, 1<br>for1tst | # i += 1<br># jump to test of outer loop |

**FIGURE e2.15.12 MIPS assembly version of the method body of the Java version of** sort. The new code is highlighted in this figure. We must still add the code to save and restore registers and the return from the MIPS code found in Figure 2.27. To keep the code similar to that figure, we load v.length into $s3 instead of into a temporary register. To reduce the number of lines of code, we make the simplifying assumption that compareTo is a leaf procedure and we do not need to push registers to be saved on the stack.