Bounds check

```
swap:  beq   $a0,$zero,NullPointer        #if $a0==0,goto Error
       lw    $t2,-4($a0)                  # Temp reg $t2 = length of array v
       slt   $t0,$a1,$zero                # Temp reg $t0 = 1 if k < 0
       bne   $t0,$zero,IndexOutOfBounds   # if k < 0, goto Error
       slt   $t0,$a1,$t2                  # Temp reg $t0 = 0 if k >= length
       beq   $t0,$zero,IndexOutOfBounds   # if k >= length, goto Error
       addi  $t1,$a1,1                    # Temp reg $t1 = k+1
       slt   $t0,$t1,$zero                # Temp reg $t0 = 1 if k+1 < 0
       bne   $t0,$zero,IndexOutOfBounds   # if k+1 < 0, goto Error
       slt   $t0,$t1,$t2                  # Temp reg $t0 = 0 if k+1 >= length
       beq   $t0,$zero,IndexOutOfBounds   # if k+1 >= length, goto Error
```

Method body

```
       sll   $t1, $a1, 2                  # reg $t1 = k * 4
       add   $t1, $a0, $t1                # reg $t1 = v + (k * 4)
                                          # reg $t1 has the address of v[k]
       lw    $t0, 8($t1)                  # reg $t0 (temp) = v[k]
       lw    $t2, 12($t1)                 # reg $t2 = v[k + 1]
                                          # refers to next element of v
       sw    $t2, 8($t1)                  # v[k] = reg $t2
       sw    $t0, 12($t1)                 # v[k+1] = reg $t0 (temp)
```

Procedure return

```
       jr    $ra                          # return to calling routine
```

FIGURE e2.15.11 MIPS assembly code of the procedure swap in **Figure 2.24.**