```
1     vmovapd       (%r11),%zmm4          # Load 8 elements of C into %zmm4
2     mov           %rbx,%rcx             # register %rcx = %rbx
3     xor           %eax,%eax             # register %eax = 0
4     vmovapd       0x20(%r11),%zmm3      # Load 8 elements of C into %zmm3
5     vmovapd       0x40(%r11),%zmm2      # Load 8 elements of C into %zmm2
6     vmovapd       0x60(%r11),%zmm1      # Load 8 elements of C into %zmm1
7     vbroadcastsd (%rax,%r8,8),%zmm0     # Make 8 copies of B element in %zmm0
8     add           $0x8,%rax             # register %rax = %rax + 8
9     vfmadd231pd   (%rcx),%zmm0,%zmm4    # Parallel mul & add %zmm0, %zmm4
10    vfmadd231pd   0x20(%rcx),%zmm0,%zmm3 # Parallel mul & add %zmm0, %zmm3
11    vfmadd231pd   0x40(%rcx),%zmm0,%zmm2 # Parallel mul & add %zmm0, %zmm2
12    vfmadd231pd   0x60(%rcx),%zmm0,%zmm1 # Parallel mul & add %zmm0, %zmm1
13    add           %r9,%rcx              # register %rcx = %rcx
14    cmp           %r10,%rax             # compare %r10 to %rax
15    jne           50 <dgemm+0x50>       # jump if not %r10 != %rax
16    add           $0x1, %esi            # register % esi = % esi + 1
17    vmovapd       %zmm4, (%r11)         # Store %zmm4 into 8 C elements
18    vmovapd       %zmm3, 0x20(%r11)     # Store %zmm3 into 8 C elements
19    vmovapd       %zmm2, 0x40(%r11)     # Store %zmm2 into 8 C elements
20    vmovapd       %zmm1, 0x60(%r11)     # Store %zmm1 into 8 C elements
```

FIGURE 4.82 The x86 assembly language for the body of the nested loops generated by compiling the unrolled C code in Figure 4.81.