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

FIGURE 3.22 The x86 assembly language for the body of the nested loops generated by compiling the optimized C code in Figure 3.21. Note the similarities to Figure 2.44 on page 167, of Chapter 2, with the primary difference being that the original floating-point operations are now using ZMM registers and using the pd versions of the instructions for parallel double precision instead of the sd version for scalar double precision, and it is performing a single multiply–add instruction instead of a separate multiply instruction and a separate add instruction.