```
1   #include <x86intrin.h>
2   #define UNROLL (4)
3
4   void dgemm (int n, double* A, double* B, double* C)
5   {
6     for (int i = 0; i < n; i+=UNROLL*8)
7       for (int j = 0; j < n; ++j){
8         __m512d c[UNROLL];
9         for (int r=0;r<UNROLL;r++)
10          c[r] =  _mm512_load_pd(C+i+r*8+j*n); //[ UNROLL];
11
12        for( int k = 0; k < n; k++ )
13        {
14          __m512d bb = _mm512_broadcastsd_pd(_mm_load_sd(B+j*n+k));
15          for (int r=0;r<UNROLL;r++)
16            c[r] = _mm512_fmadd_pd(_mm512_load_pd(A+n*k+r*8+i), bb, c[r]);
17        }
18
19        for (int r=0;r<UNROLL;r++)
20          _mm512_store_pd(C+i+r*8+j*n, c[r]);
21        }
22    }
```

FIGURE 4.81 Optimized C version of DGEMM using C intrinsics to generate the AVX subword-parallel instructions for the x86 (Figure 3.21) and loop unrolling to create more opportunities for instruction-level parallelism. Figure 4.82 shows the assembly language produced by the compiler for the inner loop, which unrolls the three for-loop bodies to expose instruction level parallelism.