Solution 6

1.

```
ADD   R2, R4, R1          // R1 = R2 + R4
ADD   R4, R6, R5          // R5 = R4 + R6
ADD   R0, R2, R3          // R3 = R0 + R2
NOP                       // Waiting for R1
MOV   R6, (R1)            // MEMORY[R1] = R6
NOP                       // Waiting for R3
MOV   (R3), R6            // R6 = MEMORY[R3]
MOV   R4, R2              // R2 = R4
ADD   #4, R4, R4          // R4 = R4 + 4
NOP                       // Waiting for R2
NOP                       // Waiting for R2
ADD   R0, R2, R1          // R1 = R0 + R2
MOV   R2, R0              // R0 = R2
```

2. See next page...

3.
Given **P = 8** and **Speedup = 5**, we need to solve **5 = 1/(1 – f + f/8)**, which yields **f = 0.91**, i.e., an application program must be 91% parallelizable.

4.
(a)
+5.25 → +101.01 = +1.0101*2^2 =
$(-1)^{\mathbf{0}}*2^{(\mathbf{129}-127)}*1.0101$ → **1 10000001 01010000000000000000000**.

(b)
0 00000000 10000000000000000000000 (underflow) = $(-1)^{\mathbf{0}}*2^{\mathbf{-126}}*0.1$
→ $0.5*2^{-126}$ = **5.877471754*10^{-39}**.

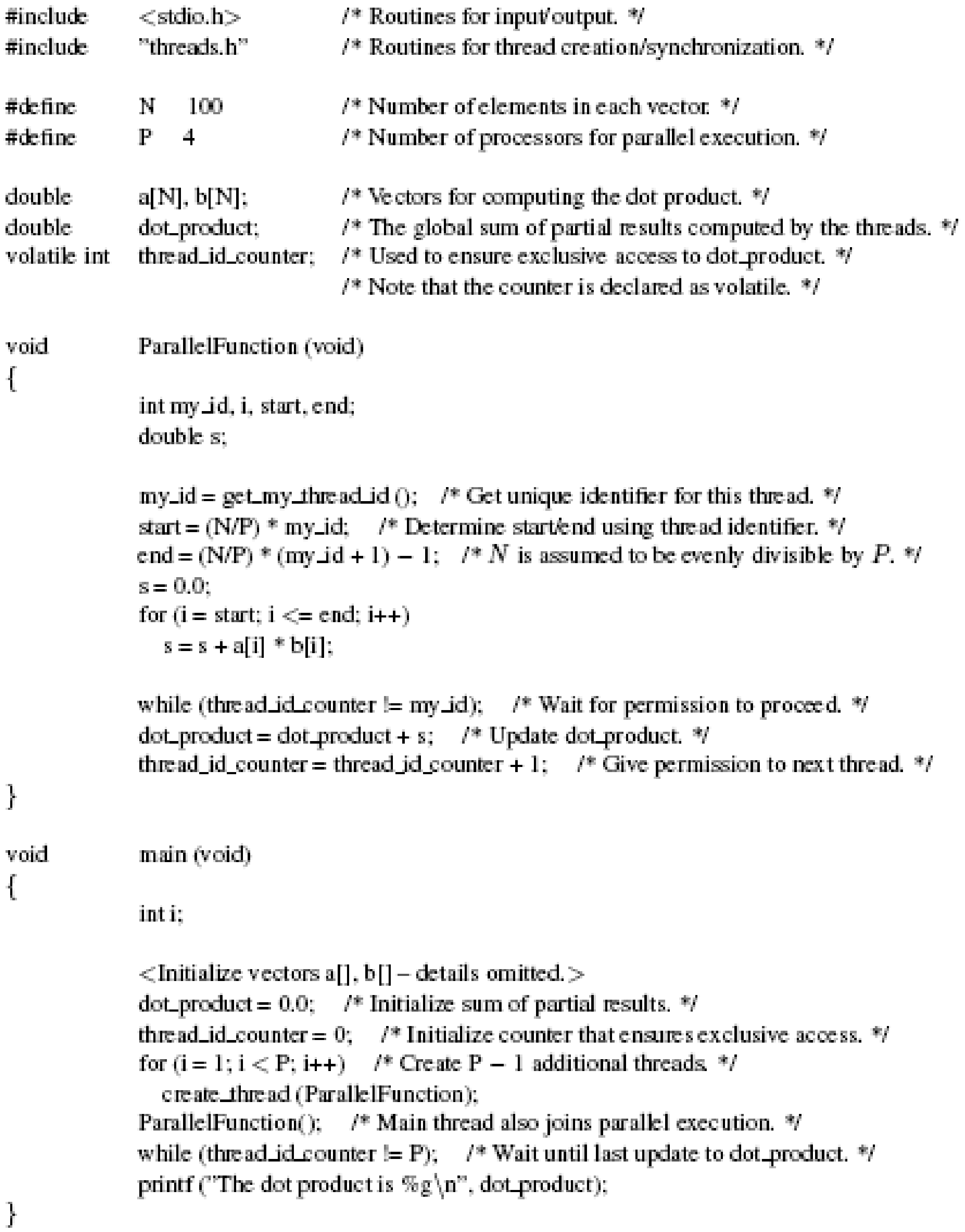
(c)
0 01111111 00000000000000000000000 = $(-1)^{\mathbf{0}}*2^{(\mathbf{127}-127)}*1.0$
→ $1*2^0$ = **1**.

(d)

```
     X   = 1 10000011 100101001111000000000000
    –Y   = 1 01111100 110000000000000000000000
         = 1 10000011 000000011000000000000000
     X +(–Y)  = 1 10000011 100101100111000000000000
              = –1.00101100111*2^4 → –18.8046875
```

This study source was downloaded by 100000843757072 from CourseHero.com on 10-11-2024 20:55:56 GMT -05:00

https://www.coursehero.com/file/16036929/CEng-355-F2013-S6/



```
#include      <stdio.h>             /* Routines for input/output. */
#include      "threads.h"           /* Routines for thread creation/synchronization. */

#define       N    100              /* Number of elements in each vector. */
#define       P    4                /* Number of processors for parallel execution. */

double        a[N], b[N];           /* Vectors for computing the dot product. */
double        dot_product;          /* The global sum of partial results computed by the threads. */
volatile int  thread_id_counter;    /* Used to ensure exclusive access to dot_product. */
                                    /* Note that the counter is declared as volatile. */

void          ParallelFunction (void)
{
              int my_id, i, start, end;
              double s;

              my_id = get_my_thread_id ();    /* Get unique identifier for this thread. */
              start = (N/P) * my_id;     /* Determine start/end using thread identifier. */
              end = (N/P) * (my_id + 1) − 1;    /* N is assumed to be evenly divisible by P. */
              s = 0.0;
              for (i = start; i <= end; i++)
                 s = s + a[i] * b[i];

              while (thread_id_counter != my_id);     /* Wait for permission to proceed. */
              dot_product = dot_product + s;     /* Update dot_product. */
              thread_id_counter = thread_id_counter + 1;     /* Give permission to next thread. */
}

void          main (void)
{
              int i;

              <Initialize vectors a[], b[] – details omitted.>
              dot_product = 0.0;     /* Initialize sum of partial results. */
              thread_id_counter = 0;     /* Initialize counter that ensures exclusive access. */
              for (i = 1; i < P; i++)     /* Create P − 1 additional threads. */
                 create_thread (ParallelFunction);
              ParallelFunction();     /* Main thread also joins parallel execution. */
              while (thread_id_counter != P);     /* Wait until last update to dot_product. */
              printf ("The dot product is %g\n", dot_product);
}
```

This study source was downloaded by 100000843757072 from CourseHero.com on 10-11-2024 20:55:56 GMT -05:00

https://www.coursehero.com/file/16036929/CEng-355-F2013-S6/