

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 5. Optimización de código

Estudiante (nombre y apellidos): Juan Carlos Ruiz Fernández

Grupo de prácticas y profesor de prácticas: B3 Mancia Anguita

Fecha de entrega: 9 de junio de 2021

Fecha evaluación en clase:

2º curso / 2º cuatr.
Grado Ing. Inform.

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): **Intel Core i7 7700k 4.2GHz**

Sistema operativo utilizado: **Windows 10 pro x64**

Versión de gcc utilizada: **gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)**

Volcado de pantalla que muestre lo que devuelve `lscpu` en la máquina en la que ha tomado las medidas:

```
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
Address sizes: 36 bits physical, 48 bits virtual
CPU(s): 8
On-line CPU(s) list: 0-7
Thread(s) per core: 2
Core(s) per socket: 4
Socket(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 158
Model name: Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
Stepping: 9
CPU MHz: 4201.000
CPU max MHz: 4201.0000
BogoMIPS: 8402.00
Hypervisor vendor: Windows Subsystem for Linux
```

1. (a) Implementar un código secuencial que calcule la multiplicación de dos matrices cuadradas. Utilizar como base el código de suma de vectores de BP0. Los datos se deben generar de forma aleatoria para un número de filas mayor que 8, como en el ejemplo de BP0, se puede usar `drand48()`.

CAPTURA CÓDIGO FUENTE: `pmm-secuencial.c`

(b) Modificar el código (solo el trozo que calcula la multiplicación) para reducir el tiempo de ejecución. Justificar los tiempos obtenidos (usando siempre -O2) a partir de la modificación realizada. Incorporar los códigos modificados en el cuaderno.

MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación A) –explicación–: He intercambiado las variables del bucle j y k para hacer menor numero de acceso a memoria

Modificación B) –explicación–: He desenrollado los bucles en 4 y comprobado que N sea múltiplo de 4

CÓDIGOS FUENTE MODIFICACIONES

A) Captura de pmm-secuencial-modificado_A.c

```

// pmm-secuencial-modificado_A.c
// Inicializar
if( N < 9 ){
    for( i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            m1[i][j] = i+j;
            m2[i][j] = N * 0.1 - i * 0.1;
            m3[i][j] = 0;
        }
    }
} else{
    srand(time(0));
    for( i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            m1[i][j] = rand() / ((double)rand());
            m2[i][j] = rand() / ((double)rand());
            m3[i][j] = 0;
        }
    }
}

// Calcular suma de vectores
clock_gettime(CLOCK_REALTIME, &cgt1);
for( i = 0; i < N; i++){
    for( int j = 0; j < N; j++){
        for( int k = 0; k < N; k++){
            m3[i][k] += m1[i][j] * m2[j][k];
        }
    }
}
clock_gettime(CLOCK_REALTIME, &cgt2);

ncgt = (double)(cgt2.tv_sec - cgt1.tv_sec) +
        (double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));

// Imprimir resultado de la suma y el tiempo de ejecución
if( N < 10 )
{
    printf("Tiempo:%11.9f\t / Tamaño Vectores:%u\n", ncgt, N);
    for( i = 0; i < N; i++){
        printf("#");
    }
}
  
```

Capturas de pantalla (que muestren la compilación y que el resultado es correcto):

```

juaqua@DESKTOP-RCIHQK2: /r
[JuanCarlosRuizFernandez juaqua@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ºIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer1] 2021-06-03 Thursday
$gcc -O2 pmm-secuencial-modificado_A.c -o pmm-secuencial-modificado_A && ./pmm-secuencial-modificado_A 4
Tiempo:0.000001000 / Tamaño Vectores:4
#0.00 1.00 2.00 3.00 # #0.40 0.40 0.40 0.40 # #1.00 1.00 1.00 1.00 #
#1.00 2.00 3.00 4.00 # #0.30 0.30 0.30 0.30 # #2.00 2.00 2.00 2.00 #
#2.00 3.00 4.00 5.00 # #0.20 0.20 0.20 0.20 # #3.00 3.00 3.00 3.00 #
#3.00 4.00 5.00 6.00 # #0.10 0.10 0.10 0.10 # #4.00 4.00 4.00 4.00 #
[JuanCarlosRuizFernandez juaqua@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ºIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer1] 2021-06-03 Thursday
  
```

B) Captura de pmm-secuencial-modificado_B.c

```

// pmm-secuencial-modificado_B.c
// Inicializar
if( N < 9 ){
    for( i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            m1[i][j] = i+j;
            m2[i][j] = N * 0.1 - i * 0.1;
            m3[i][j] = 0;
        }
    }
} else{
    srand(time(0));
    for( i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            m1[i][j] = rand() / ((double)rand());
            m2[i][j] = rand() / ((double)rand());
            m3[i][j] = 0;
        }
    }
}

// Calcular suma de vectores
if( N%4==0 ){
    printf( "N SI ES MULTIPLO DE 4\n" );
    clock_gettime(CLOCK_REALTIME, &cgt1);
    for( i = 0; i < N; i++){
        for( int j = 0; j < N; j++){
            for( int k = 0; k < N; k+=4 ){
                m3[i][k] += m1[i][j] * m2[j][k];
                m3[i][k+1] += m1[i][j] * m2[j][k+1];
                m3[i][k+2] += m1[i][j] * m2[j][k+2];
                m3[i][k+3] += m1[i][j] * m2[j][k+3];
            }
        }
    }
    clock_gettime(CLOCK_REALTIME, &cgt2);
} else{
    clock_gettime(CLOCK_REALTIME, &cgt1);
    for( i = 0; i < N; i++){
        for( int j = 0; j < N; j++){
            for( int k = 0; k < N; k++){
                m3[i][k] += m1[i][j] * m2[j][k];
            }
        }
    }
    clock_gettime(CLOCK_REALTIME, &cgt2);
}
  
```

```

juaqua@DESKTOP-RCIHQK2: /r
/mnt/c/Users/juanc/OneDrive/2ºIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer1] 2021-06-03 Thursday
$gcc -O2 pmm-secuencial-modificado_B.c -o pmm-secuencial-modificado_B && ./pmm-secuencial-modificado_B 4
SI ES MULTIPLO DE 4
Tiempo:0.000001200 / Tamaño Vectores:4
#0.00 1.00 2.00 3.00 # #0.40 0.40 0.40 0.40 # #1.00 1.00 1.00 1.00 #
#1.00 2.00 3.00 4.00 # #0.30 0.30 0.30 0.30 # #2.00 2.00 2.00 2.00 #
#2.00 3.00 4.00 5.00 # #0.20 0.20 0.20 0.20 # #3.00 3.00 3.00 3.00 #
#3.00 4.00 5.00 6.00 # #0.10 0.10 0.10 0.10 # #4.00 4.00 4.00 4.00 #
[JuanCarlosRuizFernandez juaqua@DESKTOP-RCIHQK2:
  
```

TIEMPOS: He recalculado las ejecuciones para un tamaño N==400

Modificación	Breve descripción de las modificaciones	-O2
Sin modificar	-	0.060923700
Modificación A)	Minimizar saltos en memoria, posiciones no consecutivas, intercambiando el uso de las variables en el ultimo bucle: k por j y viceversa	0.032524800
Modificación B)	Desenrollado de bucle para hacer 4 iteraciones. Comprobación de que sea múltiplo de 4 para la multiplicación y si no lo es aplicar los bucles sin desenrollar	0.029016500

COMENTARIOS SOBRE LOS RESULTADOS Y JUSTIFICACIÓN DE LAS MEJORAS EN TIEMPO:

Donde se aprecia unas diferencias mas significativas es entre el secuencial y la modificación A. En la cual con ese intercambio de variables accedo a posiciones de memoria consecutivas (columnas de una misma fila) y no como en el secuencial que accedo de fila en fila.

Con la modificación b no es tan significativa la diferencia pero si se aprecia cierta mejora. Teniendo en cuenta que hay más código a ejecutar.

- (a) Usando como base el código de BP0, generar un programa para evaluar un código de la Figura 1. M y N deben ser parámetros de entrada al programa. Los datos se deben generar de forma aleatoria para valores de M y N mayores que 8, como en el ejemplo de BP0.

CÓDIGO FIGURA 1:**CAPTURA CÓDIGO FUENTE: figura1-original.c**

```

1  struct Vectores
2  {
3      int a;
4      int b;
5  };
6
7  int main(int argc, char **argv)
8  {
9      struct timespec cgt1, cgt2;
10     int N, M;
11     if (argc < 3)
12     { //Leer argumento de entrada (n de componentes de la matriz)
13         printf("Faltan tamaño N y M \n");
14         exit(-1);
15     }
16     N = atoi(argv[1]);
17     M = atoi(argv[2]);
18     struct Vectores s[N];
19     int R[M];
20
21     //Inicializacion
22     if (N > 8 || M > 8)
23     {
24         for (int i = 0; i < N; i++)
25         {
26             s[i].a = i;
27             s[i].b = i;
28         }
29     }
30     else
31     {
32         for (int i = 0; i < N; i++)
33         {
34             s[i].a = rand() % N;
35             s[i].b = rand() % N;
36         }
37     }
38
39     int X1;
40     int X2;
41
42     //Calculo
43     clock_gettime(CLOCK_REALTIME, &cgt1);
44     for (int ii = 0; ii < M; ii++)
45     {
46         X1 = 0;
47         X2 = 0;
48         for (int i = 0; i < N; i++)
49         {
50             X1 += 2 * s[i].a + ii;
51             X2 += 3 * s[i].b - ii;
52         }
53         if (X1 < X2)
54             R[ii] = X1;
55         else
56             R[ii] = X2;
57     }
58
59     clock_gettime(CLOCK_REALTIME, &cgt2);
60     double ncgt = (double)(cgt2.tv_sec - cgt1.tv_sec) +
61         (double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
62     printf("ORIG Tiempo ejecucion %11.9f\n", ncgt);
63 }

```

Figura 1 . Código C++ que suma dos vectores. M y N deben ser parámetros de entrada al programa, usar valores mayores que 1000 en la evaluación.

```

struct {
    int a;
    int b;
} s[N];

```

```

main()
{
    ...
    for (ii=0; ii<M;ii++) {
        X1=0; X2=0;
        for(i=0; i<N;i++) X1+=2*s[i].a+ii;
        for(i=0; i<N;i++) X2+=3*s[i].b-ii;

        if (X1<X2) R[ii]=X1 else R[ii]=X2;
    }
    ...
}

```

(b) Modificar el código C (solo el trozo a evaluar) para reducir el tiempo de ejecución. Justificar los tiempos obtenidos (usando siempre -O2) a partir de la modificación realizada. En las ejecuciones de evaluación usar valores de N y M mayores que 1000. Incorporar los códigos modificados en el cuaderno.

MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación A) –explicación–: El struct tal como esta es mas optimo recorrer ambos elementos en un solo bucle para minimizar los saltos.

Modificación B) –explicación–: Cambiado las multiplicaciones por desplazamientos y sumas

CÓDIGOS FUENTE MODIFICACIONES

A) Captura figura1-modificado_A.c

```

// figura1-modificado_B.c
1 struct Vectores {
2     int a;
3     int b;
4 };
5
6 int main(int argc, char **argv)
7 {
8     struct timespec cgt1, cgt2;
9     int N,M;
10    if (argc < 3){//Leer argumento de entrada (n de componentes de la
11        printf("Faltan tamaño N y M \n");
12        exit(-1);
13    }
14    N = atoi(argv[1]);
15    M = atoi(argv[2]);
16    struct Vectores s[N];
17    int R[M];
18
19    //Inicializacion
20    if( N>8 || M>8 ){
21        for( int i = 0; i<N; i++){
22            s[i].a = i;
23            s[i].b = i;
24        }
25    }else{
26        for( int i = 0; i<N; i++){
27            s[i].a = rand()%N;
28            s[i].b = rand()%N;
29        }
30    }
31 }
32
33 // figura1-modificado_A.c
34 main(int, char **)
35 {
36     s[i].a = i;
37     s[i].b = i;
38 }
39
40 }else{
41     for( int i = 0; i<N; i++){
42         s[i].a = rand()%N;
43         s[i].b = rand()%N;
44     }
45 }
46
47 int X1;
48 int X2;
49 //Calculo
50 clock_gettime(CLOCK_REALTIME, &cgt1);
51 for (int ii = 0; ii < M; ii++)
52 {
53     X1 = 0;
54     X2 = 0;
55     for (int i = 0; i < N; i++){
56         X1 += 2 * s[i].a + ii;
57         X2 += 3 * s[i].b - ii;
58     }
59     R[ii] = (X1 < X2)? X1 : X2;
60 }
61 clock_gettime(CLOCK_REALTIME, &cgt2);
62 double ncgt = (double)(cgt2.tv_sec - cgt1.tv_sec) +
63     (double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
64 printf("ModA Tiempo ejecucion %11.9f\n", ncgt);
65 return 0;
66 }

```

B) Captura figura1-modificado_B.c

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <time.h>
4
5 struct Vectores {
6     int a;
7     int b;
8 };
9
10 int main(int argc, char **argv)
11 {
12     struct timespec cgt1, cgt2;
13     int N,M;
14     if (argc < 3){//Leer argumento de entrada (n de componentes de la
15         printf("Faltan tamaño N y M \n");
16         exit(-1);
17     }
18     N = atoi(argv[1]);
19     M = atoi(argv[2]);
20     struct Vectores s[N];
21     int R[M];
22
23     //Inicializacion
24     if( (N>8 || M>8) ){
25         for( int i = 0; i<N; i++){
26             s[i].a = i;
27             s[i].b = i;
28         }
29     }else{
30         for( int i = 0; i<N; i++){
31             s[i].a = rand()%N;
32             s[i].b = rand()%N;
33         }
34     }
35
36     int X1;
37     int X2;
38     //Calculo
39     clock_gettime(CLOCK_REALTIME, &cgt1);
40     for (int ii = 0; ii < M; ii++)
41     {
42         X1 = 0;
43         X2 = 0;
44         for (int i = 0; i < N; i+=2){
45             X1 += (s[i].a << 1) + ii;
46             X2 += (s[i].b << 1) + s[i].b - ii;
47         }
48         R[ii] = (X1 < X2)? X1 : X2;
49     }
50     clock_gettime(CLOCK_REALTIME, &cgt2);
51     double ncgt = (double)(cgt2.tv_sec - cgt1.tv_sec) +
52         (double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
53     printf("ModB Tiempo ejecucion %11.9f\n", ncgt);
54     return 0;
55 }

```

```

[juancarlosruizfernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ºIngInfo/2ºCuatri/AC/Prácticas/BP4/ejer2] 2021-06-03 Thursday
$gcc -O2 figura1-original.c -o figura1-original && gcc -O2 figura1-modificado_A.c -o figura1-modificado_A && gcc -O2 figura1-modificado_B.c -o figura1-modificado_B && ./figura1-original 1000 1000 && ./figura1-modificado_A 1000 1000 && ./figura1-modificado_B 1000 1000
ORIG Tiempo ejecucion 0.001080100
ModA Tiempo ejecucion 0.000855600
ModB Tiempo ejecucion 0.000275700
[juancarlosruizfernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ºIngInfo/2ºCuatri/AC/Prácticas/BP4/ejer2] 2021-06-03 Thursday
$

```

TIEMPOS:

Modificación	Breve descripción de las modificaciones	-O2
Sin modificar	-	0.001080100
Modificación A)	El struct tal como esta es mas optimo recorrer ambos elementos en un solo bucle para minimizar los saltos.	0.000855600
Modificación B)	Cambiado las multiplicaciones por desplazamientos y sumas	0.000275700

COMENTARIOS SOBRE LOS RESULTADOS Y JUSTIFICACIÓN DE LAS MEJORAS EN TIEMPO:

Las diferencias son bastante considerables al cambiar la forma de recorrer el array del struct. Tal como estaba era mejor recorrer ambos valores en un mismo bucle, posiciones de memoria consecutivas. Mas considerable es la diferencia sustituyendo la multipliacion por el desplazamiento y la suma.

- El benchmark Linpack ha sido uno de los programas más ampliamente utilizados para evaluar las prestaciones de los computadores. De hecho, se utiliza como base en la lista de los 500 computadores más rápidos del mundo (el Top500 Report). El núcleo de este programa es una rutina que opera con flotantes de doble precisión denominada DAXPY (*Double precision- real Alpha X Plus Y*) que multiplica un vector por una constante y los suma a otro vector (Lección 3/Tema 1):

```
for (i=0;i<N;i++) y[i]= a*x[i] + y[i];
```

Generar los programas en ensamblador para cada una de las siguientes opciones de optimización del compilador: **-O0, -Os, -O2, -O3**. Explique las diferencias que se observan en el código justificando al mismo tiempo las mejoras en velocidad que acarrearán. Incorporar los códigos al cuaderno de prácticas y destacar las diferencias entre ellos. Sólo se debe evaluar el tiempo del núcleo DAXPY. N deben ser parámetro de entrada al programa.

CAPTURA CÓDIGO FUENTE: daxpy.c

```

1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <time.h>
4
5  int main( int argc, char*argv[] ){
6      int N;
7      double constante;
8      double *x, *y;
9      struct timespec cgt1, cgt2;
10
11      if(argc < 3){
12          printf("Falta tamaño y constante");
13          return 0;
14      }
15
16      N = atoi(argv[1]);
17      constante = atof(argv[2]);
18
19      x = (double *)malloc(N*sizeof(double));
20      y = (double *)malloc(N*sizeof(double));
21
22      constante = atof(argv[2]);
23
24      for(int i = 0; i < N; i++){
25          x[i] = rand()%N;
26          y[i] = rand()%N;
27      }
28
29      clock_gettime(CLOCK_REALTIME, &cgt1);
30      for(int i = 0; i < N; i++) y[i] = constante*x[i] + y[i];
31      clock_gettime(CLOCK_REALTIME, &cgt2);
32
33      double ncgt = (double)(cgt2.tv_sec - cgt1.tv_sec) +
34                  (double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
35      printf("DAXPY Tiempo ejecucion %11.9f\n", ncgt);
36  }

```

Tiempos ejec. Longitud vectores=9000000	-O0	-Os	-O2	-O3
	0.021555900	0.019251700	0.019052200	0.018430800

CAPTURAS DE PANTALLA (que muestren la compilación y que el resultado es correcto):

```

[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$gcc -O0 daxpy.c -o daxpy0
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$gcc -Os daxpy.c -o daxpy1
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$gcc -O2 daxpy.c -o daxpy2
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$gcc -O3 daxpy.c -o daxpy3

[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$./daxpy0 9000000 5
DAXPY Tiempo ejecucion 0.021555900
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$./daxpy1 9000000 5
DAXPY Tiempo ejecucion 0.019251700
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$./daxpy2 9000000 5
DAXPY Tiempo ejecucion 0.019052200
[JuanCarlosRuizFernandez juanca@DESKTOP-RCIHQK2:
/mnt/c/Users/juanc/OneDrive/2ªIngInfo 2ºCuatri/AC/Prácticas/BP4/ejer3] 2021-06-09 Wednesday
$./daxpy3 9000000 5
DAXPY Tiempo ejecucion 0.018430800

```


COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

A parte de los tamaños, siendo el -O3 el más largo porque desenrolla bucles, tenemos las comprobaciones:

O0 hace la comprobación en L6 para volver a L7

O2 hace la comprobación al principio para saltar a L12 y al final de L6 siempre salta a si mismo (L6)

O3 la hace solamente al final de su etiqueta L7

CÓDIGO EN ENSAMBLADOR (no es necesario introducir aquí el código como captura de pantalla, ajustar el tamaño de la letra para que una instrucción no ocupe más de un renglón):

(PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR DONDE ESTÁ EL CÓDIGO EVALUADO, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

daxpy00.s	daxpy0s.s	daxpy02.s	daxpy03.s
<pre> call clock_gettime@PLT movl \$0, -88(%rbp) jmp .L6 .L7: movl -88(%rbp), %eax cltq leaq 0(,%rax,8), %rdx movq -72(%rbp), %rax addq %rdx, %rax movsd (%rax), %xmm0 movapd %xmm0, %xmm1 mulsd -80(%rbp), %xmm1 movl -88(%rbp), %eax cltq leaq 0(,%rax,8), %rdx movq -64(%rbp), %rax addq %rdx, %rax movsd (%rax), %xmm0 movl -88(%rbp), %eax cltq leaq 0(,%rax,8), %rdx movq -64(%rbp), %rax addq %rdx, %rax addsd %xmm1, %xmm0 movsd %xmm0, (%rax) addl \$1, -88(%rbp) .L6: movl -88(%rbp), %eax cmpl -84(%rbp), %eax jl .L7 leaq -32(%rbp), %rax movq %rax, %rsi movl \$0, %edi call clock_gettime@PLT </pre>	<pre> call clock_gettime@PLT xorl %eax, %eax .L6: cmpl %eax, %ebx jle .L12 movsd 8(%rsp), %xmm0 mulsd 0(%r13,%rax,8), %xmm0 addsd 0(%rbp,%rax,8), %xmm0 movsd %xmm0, 0(%rbp,%rax,8) incq %rax jmp .L6 .L12: leaq 40(%rsp), %rsi xorl %edi, %edi call clock_gettime@PLT </pre>	<pre> call clock_gettime@PLT xorl %eax, %eax .p2align 4,,10 .p2align 3 .L7: movsd 8(%rsp), %xmm0 mulsd (%r12,%rax,8), %xmm0 movq %rax, %rdx addsd (%r14,%rax,8), %xmm0 movsd %xmm0, (%r14,%rax,8) addq \$1, %rax cmpq %rdx, %r13 jne .L7 .L8: leaq 32(%rsp), %rsi xorl %edi, %edi call clock_gettime@PLT </pre>	<pre> call clock_gettime@PLT cmpl \$1, %r12d je .L9 movsd 8(%rsp), %xmm1 shrl %r12d xorl %edx, %edx salq \$4, %r12 unpcklpd %xmm1, %xmm1 .p2align 4,,10 .p2align 3 .L7: movupd 0(%r13,%rdx), %xmm0 movupd (%r15,%rdx), %xmm2 mulpd %xmm1, %xmm0 addpd %xmm2, %xmm0 movups %xmm0, (%r15,%rdx) addq \$16, %rdx cmpq %rdx, %r12 jne .L7 .L9: leaq 32(%rsp), %rsi xorl %edi, %edi call clock_gettime@PLT movq 40(%rsp), %rax pxor %xmm0, %xmm0 subq 24(%rsp), %rax cvtsi2sdq %rax, %xmm0 pxor %xmm1, %xmm1 movq 32(%rsp), %rax subq 16(%rsp), %rax cvtsi2sdq %rax, %xmm1 movl \$1, %edi movl \$1, %eax divsd .LC1(%rip), %xmm0 leaq .LC2(%rip), %rsi addsd %xmm1, %xmm0 call __printf_chk@PLT .L3: movq 56(%rsp), %rax xorq %fs:40, %rax jne .L18 addq \$72, %rsp .cfi_remember_state .cfi_def_cfa_offset 56 xorl %eax, %eax popq %rbx .cfi_def_cfa_offset 48 popq %rbp .cfi_def_cfa_offset 40 popq %r12 .cfi_def_cfa_offset 32 popq %r13 .cfi_def_cfa_offset 24 popq %r14 .cfi_def_cfa_offset 16 popq %r15 .cfi_def_cfa_offset 8 ret .L17: .cfi_restore_state leaq .LC0(%rip), %rsi movl \$1, %edi call __printf_chk@PLT jmp .L3 .L4: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime@PLT </pre>