2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 4. Optimización de código

Estudiante (nombre y apellidos):
Grupo de prácticas:
Fecha de entrega:
Fecha evaluación en clase:

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): *Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz*

Sistema operativo utilizado: *Ubuntu 18.04 LTS*

Versión de gcc utilizada: *gcc (Ubuntu 7.3.0-16ubuntu3) 7.3.0*

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

```
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~] 2018-05-18 viernes
$lscpu
Arquitectura:                        x86_64
modo(s) de operación de las CPUs:    32-bit, 64-bit
Orden de los bytes:                  Little Endian
CPU(s):                              4
Lista de la(s) CPU(s) en línea:      0-3
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»:              2
«Socket(s)»                          1
Modo(s) NUMA:                        1
ID de fabricante:                    GenuineIntel
Familia de CPU:                      6
Modelo:                              142
Nombre del modelo:                   Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
Revisión:                            9
CPU MHz:                             602.632
CPU MHz máx.:                        2500,0000
CPU MHz mín.:                        400,0000
BogoMIPS:                            5424.00
Virtualización:                      VT-x
Caché L1d:                           32K
Caché L1i:                           32K
Caché L2:                            256K
Caché L3:                            3072K
CPU(s) del nodo NUMA 0:              0-3
Indicadores:                         fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 s
s ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_fr
eq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xs
ave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault invpcid_single pti tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2
smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves ibpb ibrs stibp dtherm arat pln pts hwp hwp_notify
hwp_act_window hwp_epp
```

1. Para el núcleo que se muestra en el Figura 1, y para un programa que implemente la multiplicación de matrices (use variables globales):

   1.1 Modifique el código C para reducir el tiempo de ejecución del mismo. Justifique los tiempos obtenidos (use –O2) a partir de la modificación realizada. Incorpore los códigos modificados en el cuaderno.

   1.2 Genere los códigos en ensamblador con -O2 para el original y dos códigos modificados obtenidos en el punto anterior (incluido el que supone menor tiempo de ejecución) e incorpórelos al cuaderno de prácticas. Destaque las diferencias entre ellos en el código ensamblador.

1.3 (Ejercicio EXTRA) Intente mejorar los resultados obtenidos transformando el código ensamblador del programa para el que se han conseguido las mejores prestaciones de tiempo

Figura 1 . Código C++ que suma dos vectores

```
struct {
        int a;
        int b;
}  s[5000];

main()
{
   …
   for (ii=0; ii<40000;ii++) {
      X1=0; X2=0;
      for(i=0; i<5000;i++)  X1+=2*s[i].a+ii;
      for(i=0; i<5000;i++)  X2+=3*s[i].b-ii;

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

A) MULTIPLICACIÓN DE MATRICES:

CAPTURA CÓDIGO FUENTE: `pmm-secuencial.c`

```
double tmp;
clock_gettime(CLOCK_REALTIME,&cgt1);
#pragma omp parallel for
for(int i=0; i<N; ++i)
    for(int j=0; j<N; ++j)
        for(int z=0; z<N; ++z)
            m_res[i][j] += m1[i][z] * m2[z][j];

clock_gettime(CLOCK_REALTIME,&cgt2);
ncgt =  (double) (cgt2.tv_sec-cgt1.tv_sec)+
        (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: desenrrollado del primer bucle

Modificación b) –explicación-: desenrrollado del primer y segundo bucle

Modificación c) –explicación-: desenrrollado de todos los bucles

…

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de `pmm-secuencial-modificado_a.c`

```
if(N%5==0)
        {
                for(int i=0; i<N; i+=5)
             for(int j=0; j<N; j++)
                for(int z=0; z<N; z++)
                {
                   m_res[i][j] += m1[i][z] * m2[z][j];
                   m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                   m_res[i+2][j] += m1[i+2][z] * m2[z][j];
                   m_res[i+3][j] += m1[i+3][z] * m2[z][j];
                   m_res[i+4][j] += m1[i+4][z] * m2[z][j];
                }
        }
```

```
        else
                if(N%4==0)
                {
                        for(int i=0; i<N; i+=4)
                    for(int j=0; j<N; j++)
                       for(int z=0; z<N; z++)
                       {
                          m_res[i][j] += m1[i][z] * m2[z][j];
                          m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                          m_res[i+2][j] += m1[i+2][z] * m2[z][j];
                          m_res[i+3][j] += m1[i+3][z] * m2[z][j];
                       }
                }
                else
                        if(N%3==0)
                        {
                                for(int i=0; i<N; i+=3)
                            for(int j=0; j<N; j++)
                               for(int z=0; z<N; z++)
                               {
                                  m_res[i][j] += m1[i][z] * m2[z][j];
                                  m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                                  m_res[i+2][j] += m1[i+2][z] * m2[z][j];
                               }
                        }
                        else
                                for(int i=0; i<N; ++i)
                            for(int j=0; j<N; ++j)
                               for(int z=0; z<N; ++z)
                               {
                                  m_res[i][j] += m1[i][z] * m2[z][j];
                               }
```

//

b) ...

```
if(N%5==0)
        {
                for(int i=0; i<N; i+=5)
            for(int j=0; j<N; j+=5)
               for(int z=0; z<N; z++)
               {
                  m_res[i][j] += m1[i][z] * m2[z][j];
                  m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                  m_res[i+2][j] += m1[i+2][z] * m2[z][j];
                  m_res[i+3][j] += m1[i+3][z] * m2[z][j];
                  m_res[i+4][j] += m1[i+4][z] * m2[z][j];

                  m_res[i][j+1] += m1[i][z] * m2[z][j+1];
                  m_res[i+1][j+1] += m1[i+1][z] * m2[z][j+1];
                  m_res[i+2][j+1] += m1[i+2][z] * m2[z][j+1];
                  m_res[i+3][j+1] += m1[i+3][z] * m2[z][j+1];
                  m_res[i+4][j+1] += m1[i+4][z] * m2[z][j+1];

                  m_res[i][j+2] += m1[i][z] * m2[z][j+2];
                  m_res[i+1][j+2] += m1[i+1][z] * m2[z][j+2];
                  m_res[i+2][j+2] += m1[i+2][z] * m2[z][j+2];
                  m_res[i+3][j+2] += m1[i+3][z] * m2[z][j+2];
                  m_res[i+4][j+2] += m1[i+4][z] * m2[z][j+2];
```

```
                m_res[i][j+3] += m1[i][z] * m2[z][j+3];
                m_res[i+1][j+3] += m1[i+1][z] * m2[z][j+3];
                m_res[i+2][j+3] += m1[i+2][z] * m2[z][j+3];
                m_res[i+3][j+3] += m1[i+3][z] * m2[z][j+3];
                m_res[i+4][j+3] += m1[i+4][z] * m2[z][j+3];

                m_res[i][j+4] += m1[i][z] * m2[z][j+4];
                m_res[i+1][j+4] += m1[i+1][z] * m2[z][j+4];
                m_res[i+2][j+4] += m1[i+2][z] * m2[z][j+4];
                m_res[i+3][j+4] += m1[i+3][z] * m2[z][j+4];
                m_res[i+4][j+4] += m1[i+4][z] * m2[z][j+4];
            }
        }
        else
            if(N%4==0)
            {
                 for(int i=0; i<N; i+=4)
                for(int j=0; j<N; j+=4)
                   for(int z=0; z<N; z++)
                   {
                      m_res[i][j] += m1[i][z] * m2[z][j];
                      m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                      m_res[i+2][j] += m1[i+2][z] * m2[z][j];
                      m_res[i+3][j] += m1[i+3][z] * m2[z][j];

                      m_res[i][j+1] += m1[i][z] * m2[z][j+1];
                      m_res[i+1][j+1] += m1[i+1][z] * m2[z][j+1];
                      m_res[i+2][j+1] += m1[i+2][z] * m2[z][j+1];
                      m_res[i+3][j+1] += m1[i+3][z] * m2[z][j+1];

                      m_res[i][j+2] += m1[i][z] * m2[z][j+2];
                      m_res[i+1][j+2] += m1[i+1][z] * m2[z][j+2];
                      m_res[i+2][j+2] += m1[i+2][z] * m2[z][j+2];
                      m_res[i+3][j+2] += m1[i+3][z] * m2[z][j+2];

                      m_res[i][j+3] += m1[i][z] * m2[z][j+3];
                      m_res[i+1][j+3] += m1[i+1][z] * m2[z][j+3];
                      m_res[i+2][j+3] += m1[i+2][z] * m2[z][j+3];
                      m_res[i+3][j+3] += m1[i+3][z] * m2[z][j+3];
                   }
            }
            else
                 if(N%3==0)
                 {
                       for(int i=0; i<N; i+=3)
                      for(int j=0; j<N; j+=3)
                         for(int z=0; z<N; z++)
                         {
                            m_res[i][j] += m1[i][z] * m2[z][j];
                            m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                            m_res[i+2][j] += m1[i+2][z] * m2[z][j];

                            m_res[i][j+1] += m1[i][z] * m2[z][j+1];
                            m_res[i+1][j+1] += m1[i+1][z] * m2[z][j+1];
                            m_res[i+2][j+1] += m1[i+2][z] * m2[z][j+1];

                            m_res[i][j+2] += m1[i][z] * m2[z][j+2];
                            m_res[i+1][j+2] += m1[i+1][z] * m2[z][j+2];
                            m_res[i+2][j+2] += m1[i+2][z] * m2[z][j+2];
```

```
                    }
                }
                else
                    for(int i=0; i<N; ++i)
                  for(int j=0; j<N; ++j)
                    for(int z=0; z<N; ++z)
                    {
                      m_res[i][j] += m1[i][z] * m2[z][j];
                    }
```

//

c)

```
if(N%5==0)
    {
        for(int i=0; i<N; i+=5)
      for(int j=0; j<N; j+=5)
        for(int z=0; z<N; z+=5)
        {
          m_res[i][j] += m1[i][z] * m2[z][j];
          m_res[i+1][j] += m1[i+1][z] * m2[z][j];
          m_res[i+2][j] += m1[i+2][z] * m2[z][j];
          m_res[i+3][j] += m1[i+3][z] * m2[z][j];
          m_res[i+4][j] += m1[i+4][z] * m2[z][j];

          m_res[i][j+1] += m1[i][z] * m2[z][j+1];
          m_res[i+1][j+1] += m1[i+1][z] * m2[z][j+1];
          m_res[i+2][j+1] += m1[i+2][z] * m2[z][j+1];
          m_res[i+3][j+1] += m1[i+3][z] * m2[z][j+1];
          m_res[i+4][j+1] += m1[i+4][z] * m2[z][j+1];

          m_res[i][j+2] += m1[i][z] * m2[z][j+2];
          m_res[i+1][j+2] += m1[i+1][z] * m2[z][j+2];
          m_res[i+2][j+2] += m1[i+2][z] * m2[z][j+2];
          m_res[i+3][j+2] += m1[i+3][z] * m2[z][j+2];
          m_res[i+4][j+2] += m1[i+4][z] * m2[z][j+2];

          m_res[i][j+3] += m1[i][z] * m2[z][j+3];
          m_res[i+1][j+3] += m1[i+1][z] * m2[z][j+3];
          m_res[i+2][j+3] += m1[i+2][z] * m2[z][j+3];
          m_res[i+3][j+3] += m1[i+3][z] * m2[z][j+3];
          m_res[i+4][j+3] += m1[i+4][z] * m2[z][j+3];

          m_res[i][j+4] += m1[i][z] * m2[z][j+4];
          m_res[i+1][j+4] += m1[i+1][z] * m2[z][j+4];
          m_res[i+2][j+4] += m1[i+2][z] * m2[z][j+4];
          m_res[i+3][j+4] += m1[i+3][z] * m2[z][j+4];
          m_res[i+4][j+4] += m1[i+4][z] * m2[z][j+4];

          //

          m_res[i][j] += m1[i][z+1] * m2[z+1][j];
          m_res[i+1][j] += m1[i+1][z+1] * m2[z+1][j];
          m_res[i+2][j] += m1[i+2][z+1] * m2[z+1][j];
          m_res[i+3][j] += m1[i+3][z+1] * m2[z+1][j];
          m_res[i+4][j] += m1[i+4][z+1] * m2[z+1][j];

          m_res[i][j+1] += m1[i][z+1] * m2[z+1][j+1];
          m_res[i+1][j+1] += m1[i+1][z+1] * m2[z+1][j+1];
```

```
m_res[i+2][j+1] += m1[i+2][z+1] * m2[z+1][j+1];
m_res[i+3][j+1] += m1[i+3][z+1] * m2[z+1][j+1];
m_res[i+4][j+1] += m1[i+4][z+1] * m2[z+1][j+1];

m_res[i][j+2] += m1[i][z+1] * m2[z+1][j+2];
m_res[i+1][j+2] += m1[i+1][z+1] * m2[z+1][j+2];
m_res[i+2][j+2] += m1[i+2][z+1] * m2[z+1][j+2];
m_res[i+3][j+2] += m1[i+3][z+1] * m2[z+1][j+2];
m_res[i+4][j+2] += m1[i+4][z+1] * m2[z+1][j+2];

m_res[i][j+3] += m1[i][z+1] * m2[z+1][j+3];
m_res[i+1][j+3] += m1[i+1][z+1] * m2[z+1][j+3];
m_res[i+2][j+3] += m1[i+2][z+1] * m2[z+1][j+3];
m_res[i+3][j+3] += m1[i+3][z+1] * m2[z+1][j+3];
m_res[i+4][j+3] += m1[i+4][z+1] * m2[z+1][j+3];

m_res[i][j+4] += m1[i][z+1] * m2[z+1][j+4];
m_res[i+1][j+4] += m1[i+1][z+1] * m2[z+1][j+4];
m_res[i+2][j+4] += m1[i+2][z+1] * m2[z+1][j+4];
m_res[i+3][j+4] += m1[i+3][z+1] * m2[z+1][j+4];
m_res[i+4][j+4] += m1[i+4][z+1] * m2[z+1][j+4];

//

m_res[i][j] += m1[i][z+2] * m2[z+2][j];
m_res[i+1][j] += m1[i+1][z+2] * m2[z+2][j];
m_res[i+2][j] += m1[i+2][z+2] * m2[z+2][j];
m_res[i+3][j] += m1[i+3][z+2] * m2[z+2][j];
m_res[i+4][j] += m1[i+4][z+2] * m2[z+2][j];

m_res[i][j+1] += m1[i][z+2] * m2[z+2][j+1];
m_res[i+1][j+1] += m1[i+1][z+2] * m2[z+2][j+1];
m_res[i+2][j+1] += m1[i+2][z+2] * m2[z+2][j+1];
m_res[i+3][j+1] += m1[i+3][z+2] * m2[z+2][j+1];
m_res[i+4][j+1] += m1[i+4][z+2] * m2[z+2][j+1];

m_res[i][j+2] += m1[i][z+2] * m2[z+2][j+2];
m_res[i+1][j+2] += m1[i+1][z+2] * m2[z+2][j+2];
m_res[i+2][j+2] += m1[i+2][z+2] * m2[z+2][j+2];
m_res[i+3][j+2] += m1[i+3][z+2] * m2[z+2][j+2];
m_res[i+4][j+2] += m1[i+4][z+2] * m2[z+2][j+2];

m_res[i][j+3] += m1[i][z+2] * m2[z+2][j+3];
m_res[i+1][j+3] += m1[i+1][z+2] * m2[z+2][j+3];
m_res[i+2][j+3] += m1[i+2][z+2] * m2[z+2][j+3];
m_res[i+3][j+3] += m1[i+3][z+2] * m2[z+2][j+3];
m_res[i+4][j+3] += m1[i+4][z+2] * m2[z+2][j+3];

m_res[i][j+4] += m1[i][z+2] * m2[z+2][j+4];
m_res[i+1][j+4] += m1[i+1][z+2] * m2[z+2][j+4];
m_res[i+2][j+4] += m1[i+2][z+2] * m2[z+2][j+4];
m_res[i+3][j+4] += m1[i+3][z+2] * m2[z+2][j+4];
m_res[i+4][j+4] += m1[i+4][z+2] * m2[z+2][j+4];

//

m_res[i][j] += m1[i][z+3] * m2[z+3][j];
m_res[i+1][j] += m1[i+1][z+3] * m2[z+3][j];
m_res[i+2][j] += m1[i+2][z+3] * m2[z+3][j];
m_res[i+3][j] += m1[i+3][z+3] * m2[z+3][j];
```

```
                m_res[i+4][j] += m1[i+4][z+3] * m2[z+3][j];

                m_res[i][j+1] += m1[i][z+3] * m2[z+3][j+1];
                m_res[i+1][j+1] += m1[i+1][z+3] * m2[z+3][j+1];
                m_res[i+2][j+1] += m1[i+2][z+3] * m2[z+3][j+1];
                m_res[i+3][j+1] += m1[i+3][z+3] * m2[z+3][j+1];
                m_res[i+4][j+1] += m1[i+4][z+3] * m2[z+3][j+1];

                m_res[i][j+2] += m1[i][z+3] * m2[z+3][j+2];
                m_res[i+1][j+2] += m1[i+1][z+3] * m2[z+3][j+2];
                m_res[i+2][j+2] += m1[i+2][z+3] * m2[z+3][j+2];
                m_res[i+3][j+2] += m1[i+3][z+3] * m2[z+3][j+2];
                m_res[i+4][j+2] += m1[i+4][z+3] * m2[z+3][j+2];

                m_res[i][j+3] += m1[i][z+3] * m2[z+3][j+3];
                m_res[i+1][j+3] += m1[i+1][z+3] * m2[z+3][j+3];
                m_res[i+2][j+3] += m1[i+2][z+3] * m2[z+3][j+3];
                m_res[i+3][j+3] += m1[i+3][z+3] * m2[z+3][j+3];
                m_res[i+4][j+3] += m1[i+4][z+3] * m2[z+3][j+3];

                m_res[i][j+4] += m1[i][z+3] * m2[z+3][j+4];
                m_res[i+1][j+4] += m1[i+1][z+3] * m2[z+3][j+4];
                m_res[i+2][j+4] += m1[i+2][z+3] * m2[z+3][j+4];
                m_res[i+3][j+4] += m1[i+3][z+3] * m2[z+3][j+4];
                m_res[i+4][j+4] += m1[i+4][z+3] * m2[z+3][j+4];

                //

                m_res[i][j] += m1[i][z+4] * m2[z+4][j];
                m_res[i+1][j] += m1[i+1][z+4] * m2[z+4][j];
                m_res[i+2][j] += m1[i+2][z+4] * m2[z+4][j];
                m_res[i+3][j] += m1[i+3][z+4] * m2[z+4][j];
                m_res[i+4][j] += m1[i+4][z+4] * m2[z+4][j];

                m_res[i][j+1] += m1[i][z+4] * m2[z+4][j+1];
                m_res[i+1][j+1] += m1[i+1][z+4] * m2[z+4][j+1];
                m_res[i+2][j+1] += m1[i+2][z+4] * m2[z+4][j+1];
                m_res[i+3][j+1] += m1[i+3][z+4] * m2[z+4][j+1];
                m_res[i+4][j+1] += m1[i+4][z+4] * m2[z+4][j+1];

                m_res[i][j+2] += m1[i][z+4] * m2[z+4][j+2];
                m_res[i+1][j+2] += m1[i+1][z+4] * m2[z+4][j+2];
                m_res[i+2][j+2] += m1[i+2][z+4] * m2[z+4][j+2];
                m_res[i+3][j+2] += m1[i+3][z+4] * m2[z+4][j+2];
                m_res[i+4][j+2] += m1[i+4][z+4] * m2[z+4][j+2];

                m_res[i][j+3] += m1[i][z+4] * m2[z+4][j+3];
                m_res[i+1][j+3] += m1[i+1][z+4] * m2[z+4][j+3];
                m_res[i+2][j+3] += m1[i+2][z+4] * m2[z+4][j+3];
                m_res[i+3][j+3] += m1[i+3][z+4] * m2[z+4][j+3];
                m_res[i+4][j+3] += m1[i+4][z+4] * m2[z+4][j+3];

                m_res[i][j+4] += m1[i][z+4] * m2[z+4][j+4];
                m_res[i+1][j+4] += m1[i+1][z+4] * m2[z+4][j+4];
                m_res[i+2][j+4] += m1[i+2][z+4] * m2[z+4][j+4];
                m_res[i+3][j+4] += m1[i+3][z+4] * m2[z+4][j+4];
                m_res[i+4][j+4] += m1[i+4][z+4] * m2[z+4][j+4];
            }
        }
        else
```

```
if(N%4==0)
{
        for(int i=0; i<N; i+=4)
      for(int j=0; j<N; j+=4)
        for(int z=0; z<N; z+=4)
        {
          m_res[i][j] += m1[i][z] * m2[z][j];
          m_res[i+1][j] += m1[i+1][z] * m2[z][j];
          m_res[i+2][j] += m1[i+2][z] * m2[z][j];
          m_res[i+3][j] += m1[i+3][z] * m2[z][j];

          m_res[i][j+1] += m1[i][z] * m2[z][j+1];
          m_res[i+1][j+1] += m1[i+1][z] * m2[z][j+1];
          m_res[i+2][j+1] += m1[i+2][z] * m2[z][j+1];
          m_res[i+3][j+1] += m1[i+3][z] * m2[z][j+1];

          m_res[i][j+2] += m1[i][z] * m2[z][j+2];
          m_res[i+1][j+2] += m1[i+1][z] * m2[z][j+2];
          m_res[i+2][j+2] += m1[i+2][z] * m2[z][j+2];
          m_res[i+3][j+2] += m1[i+3][z] * m2[z][j+2];

          m_res[i][j+3] += m1[i][z] * m2[z][j+3];
          m_res[i+1][j+3] += m1[i+1][z] * m2[z][j+3];
          m_res[i+2][j+3] += m1[i+2][z] * m2[z][j+3];
          m_res[i+3][j+3] += m1[i+3][z] * m2[z][j+3];

          //

          m_res[i][j] += m1[i][z+1] * m2[z+1][j];
          m_res[i+1][j] += m1[i+1][z+1] * m2[z+1][j];
          m_res[i+2][j] += m1[i+2][z+1] * m2[z+1][j];
          m_res[i+3][j] += m1[i+3][z+1] * m2[z+1][j];

          m_res[i][j+1] += m1[i][z+1] * m2[z+1][j+1];
          m_res[i+1][j+1] += m1[i+1][z+1] * m2[z+1][j+1];
          m_res[i+2][j+1] += m1[i+2][z+1] * m2[z+1][j+1];
          m_res[i+3][j+1] += m1[i+3][z+1] * m2[z+1][j+1];

          m_res[i][j+2] += m1[i][z+1] * m2[z+1][j+2];
          m_res[i+1][j+2] += m1[i+1][z+1] * m2[z+1][j+2];
          m_res[i+2][j+2] += m1[i+2][z+1] * m2[z+1][j+2];
          m_res[i+3][j+2] += m1[i+3][z+1] * m2[z+1][j+2];

          m_res[i][j+3] += m1[i][z+1] * m2[z+1][j+3];
          m_res[i+1][j+3] += m1[i+1][z+1] * m2[z+1][j+3];
          m_res[i+2][j+3] += m1[i+2][z+1] * m2[z+1][j+3];
          m_res[i+3][j+3] += m1[i+3][z+1] * m2[z+1][j+3];

          //

          m_res[i][j] += m1[i][z+2] * m2[z+2][j];
          m_res[i+1][j] += m1[i+1][z+2] * m2[z+2][j];
          m_res[i+2][j] += m1[i+2][z+2] * m2[z+2][j];
          m_res[i+3][j] += m1[i+3][z+2] * m2[z+2][j];

          m_res[i][j+1] += m1[i][z+2] * m2[z+2][j+1];
          m_res[i+1][j+1] += m1[i+1][z+2] * m2[z+2][j+1];
          m_res[i+2][j+1] += m1[i+2][z+2] * m2[z+2][j+1];
          m_res[i+3][j+1] += m1[i+3][z+2] * m2[z+2][j+1];
```

```
                m_res[i][j+2] += m1[i][z+2] * m2[z+2][j+2];
                m_res[i+1][j+2] += m1[i+1][z+2] * m2[z+2][j+2];
                m_res[i+2][j+2] += m1[i+2][z+2] * m2[z+2][j+2];
                m_res[i+3][j+2] += m1[i+3][z+2] * m2[z+2][j+2];

                m_res[i][j+3] += m1[i][z+2] * m2[z+2][j+3];
                m_res[i+1][j+3] += m1[i+1][z+2] * m2[z+2][j+3];
                m_res[i+2][j+3] += m1[i+2][z+2] * m2[z+2][j+3];
                m_res[i+3][j+3] += m1[i+3][z+2] * m2[z+2][j+3];

                //

                m_res[i][j] += m1[i][z+3] * m2[z+3][j];
                m_res[i+1][j] += m1[i+1][z+3] * m2[z+3][j];
                m_res[i+2][j] += m1[i+2][z+3] * m2[z+3][j];
                m_res[i+3][j] += m1[i+3][z+3] * m2[z+3][j];

                m_res[i][j+1] += m1[i][z+3] * m2[z+3][j+1];
                m_res[i+1][j+1] += m1[i+1][z+3] * m2[z+3][j+1];
                m_res[i+2][j+1] += m1[i+2][z+3] * m2[z+3][j+1];
                m_res[i+3][j+1] += m1[i+3][z+3] * m2[z+3][j+1];

                m_res[i][j+2] += m1[i][z+3] * m2[z+3][j+2];
                m_res[i+1][j+2] += m1[i+1][z+3] * m2[z+3][j+2];
                m_res[i+2][j+2] += m1[i+2][z+3] * m2[z+3][j+2];
                m_res[i+3][j+2] += m1[i+3][z+3] * m2[z+3][j+2];

                m_res[i][j+3] += m1[i][z+3] * m2[z+3][j+3];
                m_res[i+1][j+3] += m1[i+1][z+3] * m2[z+3][j+3];
                m_res[i+2][j+3] += m1[i+2][z+3] * m2[z+3][j+3];
                m_res[i+3][j+3] += m1[i+3][z+3] * m2[z+3][j+3];
            }
    }
    else
            if(N%3==0)
            {
                    for(int i=0; i<N; i+=3)
                for(int j=0; j<N; j+=3)
                    for(int z=0; z<N; z+=3)
                    {
                        m_res[i][j] += m1[i][z] * m2[z][j];
                        m_res[i+1][j] += m1[i+1][z] * m2[z][j];
                        m_res[i+2][j] += m1[i+2][z] * m2[z][j];

                        m_res[i][j+1] += m1[i][z] * m2[z][j+1];
                        m_res[i+1][j+1] += m1[i+1][z] * m2[z][j+1];
                        m_res[i+2][j+1] += m1[i+2][z] * m2[z][j+1];

                        m_res[i][j+2] += m1[i][z] * m2[z][j+2];
                        m_res[i+1][j+2] += m1[i+1][z] * m2[z][j+2];
                        m_res[i+2][j+2] += m1[i+2][z] * m2[z][j+2];

                        //

                        m_res[i][j] += m1[i][z+1] * m2[z+1][j];
                        m_res[i+1][j] += m1[i+1][z+1] * m2[z+1][j];
                        m_res[i+2][j] += m1[i+2][z+1] * m2[z+1][j];

                        m_res[i][j+1] += m1[i][z+1] * m2[z+1][j+1];
                        m_res[i+1][j+1] += m1[i+1][z+1] * m2[z+1][j+1];
```

```
                        m_res[i+2][j+1] += m1[i+2][z+1] * m2[z+1][j+1];

                        m_res[i][j+2] += m1[i][z+1] * m2[z+1][j+2];
                        m_res[i+1][j+2] += m1[i+1][z+1] * m2[z+1][j+2];
                        m_res[i+2][j+2] += m1[i+2][z+1] * m2[z+1][j+2];

                        //

                        m_res[i][j] += m1[i][z+2] * m2[z+2][j];
                        m_res[i+1][j] += m1[i+1][z+2] * m2[z+2][j];
                        m_res[i+2][j] += m1[i+2][z+2] * m2[z+2][j];

                        m_res[i][j+1] += m1[i][z+2] * m2[z+2][j+1];
                        m_res[i+1][j+1] += m1[i+1][z+2] * m2[z+2][j+1];
                        m_res[i+2][j+1] += m1[i+2][z+2] * m2[z+2][j+1];

                        m_res[i][j+2] += m1[i][z+2] * m2[z+2][j+2];
                        m_res[i+1][j+2] += m1[i+1][z+2] * m2[z+2][j+2];
                        m_res[i+2][j+2] += m1[i+2][z+2] * m2[z+2][j+2];
                    }
            }
            else
                    for(int i=0; i<N; ++i)
                for(int j=0; j<N; ++j)
                    for(int z=0; z<N; ++z)
                    {
                        m_res[i][j] += m1[i][z] * m2[z][j];
                    }
```

1.1. TIEMPOS: tamaño 1500

Modificación	**-O2**
Sin modificar	*13,088*
Modificación a)	7
Modificación b)	4,84
Modificación c)	4,53

```
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1] 2018-05-21 lunes
$./pmm-secuencial 1500
m[0][0]: 6000.000000
m[1499][1499]: 6000.000000
tiempo:  13.088705880
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1] 2018-05-21 lunes
$./pmm-secuencial-mod 1500
m[0][0]: 6000.000000
m[1499][1499]: 6000.000000
tiempo:  7.006655758
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1] 2018-05-21 lunes
$./pmm-secuencial-mod-1 1500
m[0][0]: 6000.000000
m[1499][1499]: 6000.000000
tiempo:  4.843274749
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1] 2018-05-21 lunes
$./pmm-secuencial-mod-2 1500
m[0][0]: 6000.000000
m[1499][1499]: 6000.000000
tiempo:  4.531795338
```

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

1.2. CÓDIGO EN ENSAMBLADOR DEL ORIGINAL Y DE DOS MODIFICACIONES : (PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR EVALUADA, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

pmm-secuencial.s	pmm-secuencial-modificado_b.s	pmm-secuencial-modificado_c.s

call clock_gettime@PLT
xorl %r9d, %r9d
.p2align 4,,10
.p2align 3
.L15:
movq (%rsp), %rax
movq (%r12,%r9), %rdi
xorl %ecx, %ecx
movq (%rax, %r9), %rsi
.p2align 4,,10
.p2align 3
.L10:
movsd (%rdi, %rcx), %xmm1
xorl %eax, %eax
.p2align 4,,10
.p2align 3
.L7:
movq (%r15,%rax), %rdx
movsd (%rdx, %rcx), %xmm0
mulsd (%rsi, %rax), %xmm0
addq $8, %rax
cmpq %rax, %r13
addsd
	call clock_gettime@PLT
	testl %ebp, %ebp
	je .L74
	movq 96(%rsp), %rax
	testb $3, %al
	movl %eax, %ecx
	jne .L75
	movq 64(%rsp), %rax
	movq 56(%rsp), %rdi
	movq %rax, 32(%rsp)
	movl %ebx, %eax
	movq %rdi, 24(%rsp)
	shrl $2, %eax
	movq %rax, %rdx
	addq $1, %rax
	salq $5, %rax
	salq $5, %rdx
	leaq 32(%rdi, %rdx), %rdi
	movq %rax, 16(%rsp)
	leaq
		call clock_gettime@PLT
		testl %ebp, %ebp
		jne .L28
		movq 104(%rsp), %rax
		movl $0, 100(%rsp)
		movq %rax, 120(%rsp)
		movq 88(%rsp), %rax
		movq %rax, 112(%rsp)
		.p2align 4,,10
		.p2align 3
		.L29:
		movq 112(%rsp), %rax
		movl $0, 32(%rsp)
		movl $16, %r15d
		movq (%rax), %rdi
		movq %rdi, 48(%rsp)
		movq 8(%rax), %rdi
		movq %rdi, 56(%rsp)
		movq

```
            %xmm0,                  8(,                     16(%rax),
%xmm1                   %rbx,8), %rax           %rdi
            jne                     movq                    movq
            .L7                     %rdi,                   %rdi,
            movsd       40(%rsp)                64(%rsp)
            %xmm1,                  movq                    movq
(%rdi,%rcx)                         %rax,                   24(%rax),
            addq        8(%rsp)                 %rdi
            $8, %rcx                .p2align                movq
            cmpq        4,,10                   32(%rax),
            %rcx, %r13              .p2align 3  %rax
            jne         .L14:                   movq
            .L10                    movq                    %rdi,
            addq        32(%rsp),   72(%rsp)
            $8, %r9     %rax                    movq
            cmpq                    xorl                    %rax,
            %r9, %r13               %r9d, %r9d  80(%rsp)
            jne                     movq                    movq
            .L15                    (%rax),                 120(%rsp),
            leaq        %r14                    %rax
            64(%rsp),               movq                    movq
%rsi                    8(%rax),                (%rax),
            xorl        %r13                    %r12
            %edi, %edi              movq                    movq
            call                    16(%rax),               8(%rax),
                        %r12                    %rcx
clock_gettime@PLT                   movq                    movq
                                    24(%rax),               16(%rax),
                        %rbp                    %rbp
                                    movq                    movq
                                    24(%rsp),               24(%rax),
                        %rax                    %rdx
                                    movq                    movq
                                    (%rax),                 32(%rax),
                        %rsi                    %rax
                                    movq                    .p2align
                                    8(%rax),    4,,10
                        %rcx                    .p2align 3
                                    movq        .L12:
                                    16(%rax),               leaq
                        %rdx                    8(%r15),
                                    movq        %rdi
                                    24(%rax),               leaq
                        %rax                    -8(%r15),
                                    .p2align    %r13
                        4,,10                   movq
                                    .p2align 3              48(%rsp),
                        .L17:                   %r11
                                    leaq                    movq
                                    8(%r9),                 40(%rsp),
                        %rbx                    %r10
                                    leaq                    movq
                                    16(%r9),                56(%rsp),
                        %r11                    %r9
                                    leaq                    leaq
```

```
            24(%r9),
%r10
            xorl
            %edi, %edi
            .p2align
4,,10
            .p2align 3
.L15:
            movq
(%r15,%rdi), %r8
            movsd
(%r14,%rdi), %xmm4
            movsd
0(%r13,%rdi), %xmm3
            movapd
            %xmm4,
%xmm1
            movsd
(%r12,%rdi), %xmm2
            movsd
            (%r8,%r9),
%xmm0
            movapd
            %xmm4,
%xmm5
            mulsd
            %xmm0,
%xmm1
            addsd
            (%rsi),
%xmm1
            movsd
            %xmm1,
(%rsi)
            movapd
            %xmm0,
%xmm1
            mulsd
            %xmm3,
%xmm1
            addsd
            (%rcx),
%xmm1
            movsd
            %xmm1,
(%rcx)
            movapd
            %xmm0,
%xmm1
            mulsd
            %xmm2,
```

```
            -16(%r15),
%r14
            movq
            %rdi,
(%rsp)
            leaq
            16(%r15),
%rdi
            movq
            64(%rsp),
%r8
            movq
            80(%rsp),
%rsi
            movl
            $0,
24(%rsp)
            movq
            %rdi,
8(%rsp)
            movq
            72(%rsp),
%rdi
            movq
            %r13,
16(%rsp)
            .p2align
4,,10
            .p2align 3
.L9:
            movq
            (%r10),
%rbx
            movsd
            (%r11),
%xmm5
            movsd
            (%r9),
%xmm4
            movq
            16(%rsp),
%r13
            movapd
            %xmm5,
%xmm1
            addq
            $40, %r11
            movsd
            (%rbx,
%r14), %xmm0
            addq
            $40, %r10
            movsd
            (%r8),
%xmm3
```

```
%xmm1                                   addq
            addsd                       $40, %r9
            (%rdx),                     mulsd
%xmm1                                   %xmm0,
            movsd                   %xmm1
            %xmm1,                      movapd
(%rdx)                                  %xmm5,
            movsd                   %xmm6
            0(%rbp,                     movsd
%rdi), %xmm1                            (%rdi),
            addq                    %xmm2
            $8, %rdi                    addq
            mulsd                       $40, %r8
            %xmm1,                      addq
%xmm0                                   $40, %rdi
            addsd                       addq
            (%rax),                     $40, %rsi
%xmm0                                   addsd
            movsd                       (%r12),
            %xmm0,                  %xmm1
(%rax)                                  movsd
            movsd                       %xmm1,
                                    (%r12)
(%r8,%rbx), %xmm0                       movapd
            mulsd                       %xmm0,
            %xmm0,                  %xmm1
%xmm5                                   mulsd
            addsd                       %xmm4,
            8(%rsi),                %xmm1
%xmm5                                   addsd
            movsd                       (%rcx),
            %xmm5,                  %xmm1
8(%rsi)                                 movsd
            movapd                      %xmm1,
            %xmm3,                  (%rcx)
%xmm5                                   movapd
            mulsd                       %xmm0,
            %xmm0,                  %xmm1
%xmm5                                   mulsd
            addsd                       %xmm3,
            8(%rcx),                %xmm1
%xmm5                                   addsd
            movsd                       0(%rbp),
            %xmm5,                  %xmm1
8(%rcx)                                 movsd
            movapd                      %xmm1,
            %xmm2,                  0(%rbp)
%xmm5                                   movapd
            mulsd                       %xmm0,
            %xmm0,                  %xmm1
%xmm5                                   mulsd
            mulsd                       %xmm2,
            %xmm1,                  %xmm1
%xmm0                                   addsd
            addsd                       (%rdx),
```

	8(%rdx), %xmm5
	movsd %xmm5, 8(%rdx)
	movapd %xmm4, %xmm5
	addsd 8(%rax), %xmm0
	movsd %xmm0, 8(%rax)
	movsd (%r8,%r11), %xmm0
	mulsd %xmm0, %xmm5
	addsd 16(%rsi), %xmm5
	movsd %xmm5, 16(%rsi)
	movapd %xmm3, %xmm5
	mulsd %xmm0, %xmm5
	addsd 16(%rcx), %xmm5
	movsd %xmm5, 16(%rcx)
	movapd %xmm2, %xmm5
	mulsd %xmm0, %xmm5
	mulsd %xmm1, %xmm0
	addsd 16(%rdx), %xmm5
	movsd %xmm5, 16(%rdx)
	addsd 16(%rax),
		%xmm1
		movsd %xmm1, (%rdx)
		movsd -40(%rsi), %xmm1
		mulsd %xmm1, %xmm0
		addsd (%rax), %xmm0
		movsd %xmm0, (%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm6
		addsd 8(%r12), %xmm6
		movsd %xmm6, 8(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rcx), %xmm6
		movsd %xmm6, 8(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rbp), %xmm6
		movsd %xmm6, 8(%rbp)
		movapd %xmm2, %xmm6

	%xmm0
	movsd %xmm0, 16(%rax)
	movsd (%r8,%r10), %xmm0
	cmpq %rdi, 8(%rsp)
	mulsd %xmm0, %xmm4
	mulsd %xmm0, %xmm3
	mulsd %xmm0, %xmm2
	mulsd %xmm1, %xmm0
	addsd 24(%rsi), %xmm4
	movsd %xmm4, 24(%rsi)
	addsd 24(%rcx), %xmm3
	movsd %xmm3, 24(%rcx)
	addsd 24(%rdx), %xmm2
	movsd %xmm2, 24(%rdx)
	addsd 24(%rax), %xmm0
	movsd %xmm0, 24(%rax)
	jne .L15
	addq $32, %rsi
	addq $32, %r9
	addq $32, %rcx
	addq
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 8(%rdx), %xmm6
		movsd %xmm6, 8(%rdx)
		movapd %xmm5, %xmm6
		addsd 8(%rax), %xmm0
		movsd %xmm0, 8(%rax)
		movsd (%rbx, %r15), %xmm0
		mulsd %xmm0, %xmm6
		addsd 16(%r12), %xmm6
		movsd %xmm6, 16(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rcx), %xmm6
		movq (%rsp), %r13
		movsd %xmm6, 16(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd

```
                $32, %rdx                   16(%rbp),
%xmm6
                addq                        movsd
                $32, %rax                   %xmm6,
                cmpq                        16(%rbp)
%r9,                                        movapd
16(%rsp)                                    %xmm2,
                jne                         %xmm6
                .L17                        mulsd
                addq                        %xmm0,
                $32,                        %xmm6
24(%rsp)                                    mulsd
                addq                        %xmm1,
                $32,                        %xmm0
32(%rsp)                                    addsd
                movq                        16(%rdx),
                24(%rsp),                   %xmm6
%rax                                        movsd
                cmpq                        %xmm6,
                %rax,                       16(%rdx)
40(%rsp)                                    movapd
                jne                         %xmm5,
                .L14                        %xmm6
                jmp                         addsd
                .L11                        16(%rax),
.L74:                                       %xmm0
                movq                        movsd
                56(%rsp),                   %xmm0,
%rax                                        16(%rax)
                movl                        movsd
                $0,                         (%rbx,
76(%rsp)                                    %r13), %xmm0
                movq                        movq
                %r15,                       8(%rsp),
40(%rsp)                                    %r13
                movq                        mulsd
                %rax,                       %xmm0,
88(%rsp)                                    %xmm6
                movq                        addsd
                64(%rsp),                   24(%r12),
%rax                                        %xmm6
                movq                        movsd
                %rax,                       %xmm6,
80(%rsp)                                    24(%r12)
                .p2align                    movapd
4,,10                                       %xmm4,
                .p2align 3                  %xmm6
.L30:                                       mulsd
                movq                        %xmm0,
                80(%rsp),                   %xmm6
%rax                                        addsd
                movl                        24(%rcx),
                $0,                         %xmm6
52(%rsp)                                    movsd
                movl                        %xmm6,
                $16, %r10d
```

```
                movq    24(%rcx)
        24(%rax),               movapd
%rbx                            %xmm3,
                movq    %xmm6
        (%rax),                 mulsd
%r15                            %xmm0,
                movq    %xmm6
        8(%rax),                addsd
%r14                            24(%rbp),
                movq    %xmm6
        16(%rax),               movsd
%r13                            %xmm6,
                movq    24(%rbp)
        32(%rax),               movapd
%r11                            %xmm2,
                movq    %xmm6
        88(%rsp),               mulsd
%rax                            %xmm0,
                movq    %xmm6
        %rbx,                   mulsd
24(%rsp)                        %xmm1,
                movq    %xmm0
        (%rax),                 addsd
%rdi                            24(%rdx),
                movq    %xmm6
        8(%rax),                movsd
%rsi                            %xmm6,
                movq    24(%rdx)
        16(%rax),               addsd
%rcx                            24(%rax),
                movq    %xmm0
        24(%rax),               movsd
%rdx                            %xmm0,
                movq    24(%rax)
        32(%rax),               movsd
%rax                            (%rbx,
                .p2align %r13), %xmm0
4,,10                           mulsd
                .p2align 3      %xmm0,
.L12:                   %xmm5
                leaq            mulsd
        8(%r10),                %xmm0,
%rbx                    %xmm4
                leaq            mulsd
        16(%r10),               %xmm0,
%r9                     %xmm3
                leaq            mulsd
        -16(%r10),              %xmm0,
%r12                    %xmm2
                leaq            addsd
        -8(%r10),               32(%r12),
%rbp                    %xmm5
                xorl            mulsd
        %r8d, %r8d              %xmm1,
                movq    %xmm0
```

```
                %r9,
8(%rsp)
                movq
                %rbx,
16(%rsp)
                .p2align
4,,10
                .p2align 3
.L9:
                movq
                40(%rsp),
%r9
                movsd
                (%r15,%r8), %xmm5
                movsd
                (%r14,%r8), %xmm4
                movq
                24(%rsp),
%rbx
                movapd
                %xmm5,
%xmm1
                movq
                (%r9,%r8),
%r9
                movsd
                0(%r13,%r8), %xmm3
                movsd
                (%rbx,
%r8), %xmm2
                movapd
                %xmm5,
%xmm6
                movsd
                (%r9,%r12), %xmm0
                mulsd
                %xmm0,
%xmm1
                addsd
                (%rdi),
%xmm1
                movsd
                %xmm1,
(%rdi)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm4,
%xmm1
                addsd
                movsd
                %xmm5,
32(%r12)
                addsd
                32(%rcx),
%xmm4
                movsd
                %xmm4,
32(%rcx)
                addsd
                32(%rbp),
%xmm3
                movsd
                %xmm3,
32(%rbp)
                addsd
                32(%rdx),
%xmm2
                movsd
                %xmm2,
32(%rdx)
                addsd
                32(%rax),
%xmm0
                movsd
                %xmm0,
32(%rax)
                movsd
                -32(%r11),
%xmm5
                movq
                -32(%r10),
%rbx
                movsd
                -32(%r9),
%xmm4
                movapd
                %xmm5,
%xmm1
                movsd
                -32(%r8),
%xmm3
                movsd
                (%rbx,
%r14), %xmm0
                movq
                16(%rsp),
%r13
                movsd
                -32(%rdi),
%xmm2
                mulsd
                %xmm0,
%xmm1
                movapd
```

```
                (%rsi),
%xmm1
                movsd
                %xmm1,
(%rsi)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm3,
%xmm1
                addsd
                (%rcx),
%xmm1
                movsd
                %xmm1,
(%rcx)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm2,
%xmm1
                addsd
                (%rdx),
%xmm1
                movsd
                %xmm1,
(%rdx)
                movsd
(%r11,%r8), %xmm1
                addq
                $8, %r8
                mulsd
                %xmm1,
%xmm0
                addsd
                (%rax),
%xmm0
                movsd
                %xmm0,
(%rax)
                movsd
(%r9,%rbp), %xmm0
                mulsd
                %xmm0,
%xmm6
                addsd
                8(%rdi),
%xmm6
                movsd
                %xmm6,
8(%rdi)
                %xmm5,
%xmm6
                addsd
                (%r12),
%xmm1
                movsd
                %xmm1,
(%r12)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm4,
%xmm1
                addsd
                (%rcx),
%xmm1
                movsd
                %xmm1,
(%rcx)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm3,
%xmm1
                addsd
                0(%rbp),
%xmm1
                movsd
                %xmm1,
0(%rbp)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm2,
%xmm1
                addsd
                (%rdx),
%xmm1
                movsd
                %xmm1,
(%rdx)
                movsd
                -32(%rsi),
%xmm1
                mulsd
                %xmm1,
%xmm0
                addsd
                (%rax),
%xmm0
                movsd
                %xmm0,
```

	movapd %xmm4, %xmm6
	mulsd %xmm0, %xmm6
	addsd 8(%rsi), %xmm6
	movsd %xmm6, 8(%rsi)
	movapd %xmm3, %xmm6
	mulsd %xmm0, %xmm6
	addsd 8(%rcx), %xmm6
	movsd %xmm6, 8(%rcx)
	movapd %xmm2, %xmm6
	mulsd %xmm0, %xmm6
	mulsd %xmm1, %xmm0
	addsd 8(%rdx), %xmm6
	movsd %xmm6, 8(%rdx)
	movapd %xmm5, %xmm6
	addsd 8(%rax), %xmm0
	movsd %xmm0, 8(%rax)
	movsd (%r9,%r10), %xmm0
	mulsd %xmm0, %xmm6
	addsd
		(%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm6
		addsd 8(%r12), %xmm6
		movsd %xmm6, 8(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rcx), %xmm6
		movsd %xmm6, 8(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rbp), %xmm6
		movsd %xmm6, 8(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 8(%rdx), %xmm6
		movsd %xmm6, 8(%rdx)
		movapd %xmm5, %xmm6

```
                16(%rdi),
%xmm6                                   addsd
                movsd                   8(%rax),
%xmm6,                  %xmm0
16(%rdi)                                movsd
                movapd                  %xmm0,
%xmm4,                  8(%rax)
%xmm6                                   movsd
                movq                    (%rbx,
16(%rsp),               %r15), %xmm0
%rbx                                    mulsd
                mulsd                   %xmm0,
%xmm0,                  %xmm6
%xmm6                                   addsd
                addsd                   16(%r12),
16(%rsi),               %xmm6
%xmm6                                   movsd
                movsd                   %xmm6,
%xmm6,                  16(%r12)
16(%rsi)                                movapd
                movapd                  %xmm4,
%xmm3,                  %xmm6
%xmm6                                   mulsd
                mulsd                   %xmm0,
%xmm0,                  %xmm6
%xmm6                                   addsd
                addsd                   16(%rcx),
16(%rcx),               %xmm6
%xmm6                                   movq
                movsd                   (%rsp),
%xmm6,                  %r13
16(%rcx)                                movsd
                movapd                  %xmm6,
%xmm2,                  16(%rcx)
%xmm6                                   movapd
                mulsd                   %xmm3,
%xmm0,                  %xmm6
%xmm6                                   mulsd
                mulsd                   %xmm0,
%xmm1,                  %xmm6
%xmm0                                   addsd
                addsd                   16(%rbp),
16(%rdx),               %xmm6
%xmm6                                   movsd
                movsd                   %xmm6,
%xmm6,                  16(%rbp)
16(%rdx)                                movapd
                movapd                  %xmm2,
%xmm5,                  %xmm6
%xmm6                                   mulsd
                addsd                   %xmm0,
16(%rax),               %xmm6
%xmm0                                   mulsd
                movsd                   %xmm1,
%xmm0,                  %xmm0
                                        addsd
```

```
16(%rax)                    16(%rdx),
            movsd           %xmm6
(%r9,%rbx), %xmm0           movsd
            movq            %xmm6,
            8(%rsp),        16(%rdx)
%rbx                        movapd
            mulsd           %xmm5,
            %xmm0,          %xmm6
%xmm6                       addsd
            addsd           16(%rax),
            24(%rdi),       %xmm0
%xmm6                       movsd
            movsd           %xmm0,
            %xmm6,          16(%rax)
24(%rdi)                    movsd
            movapd          (%rbx,
            %xmm4,          %r13), %xmm0
%xmm6                       movq
            mulsd           8(%rsp),
            %xmm0,          %r13
%xmm6                       mulsd
            addsd           %xmm0,
            24(%rsi),       %xmm6
%xmm6                       addsd
            movsd           24(%r12),
            %xmm6,          %xmm6
24(%rsi)                    movsd
            movapd          %xmm6,
            %xmm3,          24(%r12)
%xmm6                       movapd
            mulsd           %xmm4,
            %xmm0,          %xmm6
%xmm6                       mulsd
            addsd           %xmm0,
            24(%rcx),       %xmm6
%xmm6                       addsd
            movsd           24(%rcx),
            %xmm6,          %xmm6
24(%rcx)                    movsd
            movapd          %xmm6,
            %xmm2,          24(%rcx)
%xmm6                       movapd
            mulsd           %xmm3,
            %xmm0,          %xmm6
%xmm6                       mulsd
            mulsd           %xmm0,
            %xmm1,          %xmm6
%xmm0                       addsd
            addsd           24(%rbp),
            24(%rdx),       %xmm6
%xmm6                       movsd
            movsd           %xmm6,
            %xmm6,          24(%rbp)
24(%rdx)                    movapd
                            %xmm2,
```

```
                addsd   24(%rax), %xmm0
                movsd   %xmm0, 24(%rax)
                movsd   (%r9,%rbx), %xmm0
                mulsd   %xmm0, %xmm5
                mulsd   %xmm0, %xmm4
                mulsd   %xmm0, %xmm3
                mulsd   %xmm0, %xmm2
                addsd   32(%rdi), %xmm5
                mulsd   %xmm1, %xmm0
                movsd   %xmm5, 32(%rdi)
                addsd   32(%rsi), %xmm4
                movsd   %xmm4, 32(%rsi)
                addsd   32(%rcx), %xmm3
                movsd   %xmm3, 32(%rcx)
                addsd   32(%rdx), %xmm2
                movsd   %xmm2, 32(%rdx)
                addsd   32(%rax), %xmm0
                movsd   %xmm0, 32(%rax)
                cmpq    %xmm6
                mulsd   %xmm0, %xmm6
                mulsd   %xmm1, %xmm0
                addsd   24(%rdx), %xmm6
                movsd   %xmm6, 24(%rdx)
                addsd   24(%rax), %xmm0
                movsd   %xmm0, 24(%rax)
                movsd   (%rbx, %r13), %xmm0
                mulsd   %xmm0, %xmm5
                mulsd   %xmm0, %xmm4
                mulsd   %xmm0, %xmm3
                mulsd   %xmm0, %xmm2
                addsd   32(%r12), %xmm5
                mulsd   %xmm1, %xmm0
                movsd   %xmm5, 32(%r12)
                addsd   32(%rcx), %xmm4
                movsd   %xmm4, 32(%rcx)
                addsd   32(%rbp), %xmm3
                movsd   %xmm3, 32(%rbp)
```

```
                %r8,
32(%rsp)
                jne
                .L9
                addl
                $5,
52(%rsp)
                addq
                $40, %rdi
                addq
                $40, %rsi
                movl
                52(%rsp),
%ebx
                addq
                $40, %rcx
                addq
                $40, %rdx
                addq
                $40, %rax
                addq
                $40, %r10
                cmpl
                72(%rsp),
%ebx
                jl
                .L12
                addl
                $5,
76(%rsp)
                addq
                $40,
88(%rsp)
                movl
                76(%rsp),
%eax
                addq
                $40,
80(%rsp)
                cmpl
                72(%rsp),
%eax
                jl
                .L30
                movq
                40(%rsp),
%r15
.L11:
                leaq
                128(%rsp),
%rsi
                xorl
                %edi, %edi
                xorl
                %r12d,
                addsd
                32(%rdx),
%xmm2
                movsd
                %xmm2,
32(%rdx)
                addsd
                32(%rax),
%xmm0
                movsd
                %xmm0,
32(%rax)
                movsd
                -24(%r11),
%xmm5
                movq
                -24(%r10),
%rbx
                movsd
                -24(%r9),
%xmm4
                movapd
                %xmm5,
%xmm1
                movsd
                -24(%r8),
%xmm3
                movsd
                (%rbx,
%r14), %xmm0
                movq
                16(%rsp),
%r13
                movsd
                -24(%rdi),
%xmm2
                mulsd
                %xmm0,
%xmm1
                movapd
                %xmm5,
%xmm6
                addsd
                (%r12),
%xmm1
                movsd
                %xmm1,
(%r12)
                movapd
                %xmm0,
%xmm1
                mulsd
                %xmm4,
%xmm1
                addsd
```

	%r12d
	leaq .LC3(%rip), %rbp
	call clock_gettime@PLT
		(%rcx), %xmm1
		movsd %xmm1, (%rcx)
		movapd %xmm0, %xmm1
		mulsd %xmm3, %xmm1
		addsd 0(%rbp), %xmm1
		movsd %xmm1, 0(%rbp)
		movapd %xmm0, %xmm1
		mulsd %xmm2, %xmm1
		addsd (%rdx), %xmm1
		movsd %xmm1, (%rdx)
		movsd -24(%rsi), %xmm1
		mulsd %xmm1, %xmm0
		addsd (%rax), %xmm0
		movsd %xmm0, (%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm6
		addsd 8(%r12), %xmm6
		movsd %xmm6, 8(%r12)
		movapd %xmm4,

		%xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rcx), %xmm6
		movsd %xmm6, 8(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rbp), %xmm6
		movsd %xmm6, 8(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 8(%rdx), %xmm6
		movsd %xmm6, 8(%rdx)
		movapd %xmm5, %xmm6
		addsd 8(%rax), %xmm0
		movsd %xmm0, 8(%rax)
		movsd (%rbx, %r15), %xmm0
		mulsd %xmm0, %xmm6
		addsd 16(%r12), %xmm6

		movsd %xmm6, 16(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rcx), %xmm6
		movq (%rsp), %r13
		movsd %xmm6, 16(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rbp), %xmm6
		movsd %xmm6, 16(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 16(%rdx), %xmm6
		movsd %xmm6, 16(%rdx)
		movapd %xmm5, %xmm6
		addsd 16(%rax), %xmm0
		movsd %xmm0, 16(%rax)
		movsd

		(%rbx, %r13), %xmm0
		movq 8(%rsp), %r13
		mulsd %xmm0, %xmm6
		addsd 24(%r12), %xmm6
		movsd %xmm6, 24(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 24(%rcx), %xmm6
		movsd %xmm6, 24(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 24(%rbp), %xmm6
		movsd %xmm6, 24(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 24(%rdx), %xmm6
		movsd %xmm6, 24(%rdx)
		addsd 24(%rax),

		%xmm0
		movsd %xmm0, 24(%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm5
		mulsd %xmm0, %xmm4
		mulsd %xmm0, %xmm3
		mulsd %xmm0, %xmm2
		addsd 32(%r12), %xmm5
		mulsd %xmm1, %xmm0
		movsd %xmm5, 32(%r12)
		addsd 32(%rcx), %xmm4
		movsd %xmm4, 32(%rcx)
		addsd 32(%rbp), %xmm3
		movsd %xmm3, 32(%rbp)
		addsd 32(%rdx), %xmm2
		movsd %xmm2, 32(%rdx)
		addsd 32(%rax), %xmm0
		movsd %xmm0, 32(%rax)
		movsd -16(%r11), %xmm5

		movq -16(%r10), %rbx
		movsd -16(%r9), %xmm4
		movapd %xmm5, %xmm1
		movsd -16(%r8), %xmm3
		movsd (%rbx, %r14), %xmm0
		movq 16(%rsp), %r13
		movsd -16(%rdi), %xmm2
		mulsd %xmm0, %xmm1
		movapd %xmm5, %xmm6
		addsd (%r12), %xmm1
		movsd %xmm1, (%r12)
		movapd %xmm0, %xmm1
		mulsd %xmm4, %xmm1
		addsd (%rcx), %xmm1
		movsd %xmm1, (%rcx)
		movapd %xmm0, %xmm1
		mulsd %xmm3, %xmm1
		addsd 0(%rbp), %xmm1
		movsd

		%xmm1, 0(%rbp)
		movapd %xmm0, %xmm1
		mulsd %xmm2, %xmm1
		addsd (%rdx), %xmm1
		movsd %xmm1, (%rdx)
		movsd -16(%rsi), %xmm1
		mulsd %xmm1, %xmm0
		addsd (%rax), %xmm0
		movsd %xmm0, (%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm6
		addsd 8(%r12), %xmm6
		movsd %xmm6, 8(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rcx), %xmm6
		movsd %xmm6, 8(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0,

		%xmm6
		addsd 8(%rbp), %xmm6
		movsd %xmm6, 8(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 8(%rdx), %xmm6
		movsd %xmm6, 8(%rdx)
		movapd %xmm5, %xmm6
		addsd 8(%rax), %xmm0
		movsd %xmm0, 8(%rax)
		movsd (%rbx, %r15), %xmm0
		mulsd %xmm0, %xmm6
		addsd 16(%r12), %xmm6
		movsd %xmm6, 16(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rcx), %xmm6
		movq (%rsp), %r13

		movsd %xmm6, 16(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rbp), %xmm6
		movsd %xmm6, 16(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 16(%rdx), %xmm6
		movsd %xmm6, 16(%rdx)
		movapd %xmm5, %xmm6
		addsd 16(%rax), %xmm0
		movsd %xmm0, 16(%rax)
		movsd (%rbx, %r13), %xmm0
		movq 8(%rsp), %r13
		mulsd %xmm0, %xmm6
		addsd 24(%r12), %xmm6
		movsd %xmm6, 24(%r12)
		movapd

		%xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 24(%rcx), %xmm6
		movsd %xmm6, 24(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 24(%rbp), %xmm6
		movsd %xmm6, 24(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 24(%rdx), %xmm6
		movsd %xmm6, 24(%rdx)
		addsd 24(%rax), %xmm0
		movsd %xmm0, 24(%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm5
		mulsd %xmm0, %xmm4
		mulsd %xmm0,

		%xmm3
		mulsd %xmm0, %xmm2
		addsd 32(%r12), %xmm5
		mulsd %xmm1, %xmm0
		movsd %xmm5, 32(%r12)
		addsd 32(%rcx), %xmm4
		movsd %xmm4, 32(%rcx)
		addsd 32(%rbp), %xmm3
		movsd %xmm3, 32(%rbp)
		addsd 32(%rdx), %xmm2
		movsd %xmm2, 32(%rdx)
		addsd 32(%rax), %xmm0
		movsd %xmm0, 32(%rax)
		movsd -8(%r11), %xmm5
		movq -8(%r10), %rbx
		movsd -8(%r9), %xmm4
		movapd %xmm5, %xmm1
		movsd -8(%r8), %xmm3
		movsd (%rbx, %r14), %xmm0

		movq 16(%rsp), %r13
		movsd -8(%rdi), %xmm2
		mulsd %xmm0, %xmm1
		movapd %xmm5, %xmm6
		addsd (%r12), %xmm1
		movsd %xmm1, (%r12)
		movapd %xmm0, %xmm1
		mulsd %xmm4, %xmm1
		addsd (%rcx), %xmm1
		movsd %xmm1, (%rcx)
		movapd %xmm0, %xmm1
		mulsd %xmm3, %xmm1
		addsd 0(%rbp), %xmm1
		movsd %xmm1, 0(%rbp)
		movapd %xmm0, %xmm1
		mulsd %xmm2, %xmm1
		addsd (%rdx), %xmm1
		movsd %xmm1, (%rdx)
		movsd

		-8(%rsi), %xmm1
		mulsd %xmm1, %xmm0
		addsd (%rax), %xmm0
		movsd %xmm0, (%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm6
		addsd 8(%r12), %xmm6
		movsd %xmm6, 8(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rcx), %xmm6
		movsd %xmm6, 8(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 8(%rbp), %xmm6
		movsd %xmm6, 8(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1,

		%xmm0
		addsd 8(%rdx), %xmm6
		movsd %xmm6, 8(%rdx)
		movapd %xmm5, %xmm6
		addsd 8(%rax), %xmm0
		movsd %xmm0, 8(%rax)
		movsd (%rbx, %r15), %xmm0
		mulsd %xmm0, %xmm6
		addsd 16(%r12), %xmm6
		movsd %xmm6, 16(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rcx), %xmm6
		movq (%rsp), %r13
		movsd %xmm6, 16(%rcx)
		movapd %xmm3, %xmm6
		mulsd %xmm0, %xmm6
		addsd 16(%rbp), %xmm6
		movsd %xmm6, 16(%rbp)

		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 16(%rdx), %xmm6
		movsd %xmm6, 16(%rdx)
		movapd %xmm5, %xmm6
		addsd 16(%rax), %xmm0
		movsd %xmm0, 16(%rax)
		movsd (%rbx, %r13), %xmm0
		movq 8(%rsp), %r13
		mulsd %xmm0, %xmm6
		addsd 24(%r12), %xmm6
		movsd %xmm6, 24(%r12)
		movapd %xmm4, %xmm6
		mulsd %xmm0, %xmm6
		addsd 24(%rcx), %xmm6
		movsd %xmm6, 24(%rcx)
		movapd %xmm3, %xmm6
		mulsd

		%xmm0, %xmm6
		addsd 24(%rbp), %xmm6
		movsd %xmm6, 24(%rbp)
		movapd %xmm2, %xmm6
		mulsd %xmm0, %xmm6
		mulsd %xmm1, %xmm0
		addsd 24(%rdx), %xmm6
		movsd %xmm6, 24(%rdx)
		addsd 24(%rax), %xmm0
		movsd %xmm0, 24(%rax)
		movsd (%rbx, %r13), %xmm0
		mulsd %xmm0, %xmm5
		mulsd %xmm0, %xmm4
		mulsd %xmm0, %xmm3
		mulsd %xmm0, %xmm2
		addsd 32(%r12), %xmm5
		mulsd %xmm1, %xmm0
		movsd %xmm5, 32(%r12)
		addsd 32(%rcx),

		%xmm4
		movsd %xmm4, 32(%rcx)
		addsd 32(%rbp), %xmm3
		movsd %xmm3, 32(%rbp)
		addsd 32(%rdx), %xmm2
		movsd %xmm2, 32(%rdx)
		addsd 32(%rax), %xmm0
		movsd %xmm0, 32(%rax)
		addl $5, 24(%rsp)
		movl 24(%rsp), %ebx
		cmpl 96(%rsp), %ebx
		jl .L9
		addl $5, 32(%rsp)
		addq $40, %r12
		addq $40, %rcx
		movl 32(%rsp), %edi
		addq $40, %rbp
		addq $40, %rdx
		addq $40, %rax
		addq $40, %r15
		cmpl 96(%rsp), %edi
		jl

		.L12 addl $5, 100(%rsp) addq $40, 120(%rsp) movl 100(%rsp), %eax addq $40, 112(%rsp) cmpl 96(%rsp), %eax jl .L29 .L11: leaq 160(%rsp), %rsi xorl %edi, %edi xorl %r12d, %r12d leaq .LC3(%rip) , %rbp call clock_gettime@PLT

B) CÓDIGO FIGURA 1:
CAPTURA CÓDIGO FUENTE: `figura1-original.c`

```c
int main()
{
    int X1, X2;
    int R[40000];
    struct timespec cgt1,
                    cgt2;
    double          ncgt; //para tiempo de ejecución

    for (int i = 0; i < 5000; ++i)
    {
        s[i].a = i % 6;
        s[i].b = i % 12;
    }

    clock_gettime(CLOCK_REALTIME,&cgt1);

    for (int ii=0; ii<40000; ++ii){
        X1=0; X2=0;

        for(int i=0; i<5000;i++)
            X1+=2*s[i].a+ii;

        for(int i=0; i<5000;i++)
            X2+=3*s[i].b-ii;

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

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

    printf("R[0]: %d\n", R[0]);
    printf("m[39999]: %d\n", R[39999]);
    printf("tiempo:  %11.9f\n", ncgt);
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: unir los dos bucles internos

Modificación b) –explicación-: desenrollar el bucle interno de la modificación a

...

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figura1`-modificado_a.c`

```
clock_gettime(CLOCK_REALTIME,&cgt1);

for (int ii=0; ii<40000; ++ii){
    X1=0; X2=0;

    for(int i=0; i<5000;i++)
    {
        X1+=2*s[i].a+ii;
        X2+=3*s[i].b-ii;
    }

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

clock_gettime(CLOCK_REALTIME,&cgt2);
```

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

b)

```
clock_gettime(CLOCK_REALTIME,&cgt1);

for (int ii=0; ii<40000; ++ii){
    X1=0; X2=0;

    for(int i=0; i<5000;i+=5)
    {
        X1+=2*s[i].a+ii;
        X1+=2*s[i+1].a+ii;
        X1+=2*s[i+2].a+ii;
        X1+=2*s[i+3].a+ii;
        X1+=2*s[i+4].a+ii;

        X2+=3*s[i].b-ii;
        X2+=3*s[i+1].b-ii;
        X2+=3*s[i+2].b-ii;
        X2+=3*s[i+3].b-ii;
        X2+=3*s[i+4].b-ii;
        //_mm_prefetch(s[i+32],_MM_HINT_T0);
    }

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

clock_gettime(CLOCK_REALTIME,&cgt2);
```

```
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1/B] 2018-05-28 lunes
$./f./figura1-original
R[0]: 24992
m[39999]: -199912548
tiempo:  0.307736419
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1/B] 2018-05-28 lunes
$./figura1-mod-1
R[0]: 24992
m[39999]: -199912548
tiempo:  0.225723487
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer1/B] 2018-05-28 lunes
$./figura1-mod-2
R[0]: 24992
m[39999]: -199912548
tiempo:  0.177971926
```

1.1. TIEMPOS:

Modificación	**-O2**
Sin modificar	*0,307736419*
Modificación a)	0,225723487
Modificación b)	0,177971926
...	

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

1.2. CÓDIGO EN ENSAMBLADOR DEL ORIGINAL Y DE DOS MODIFICACIONES: (PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR EVALUADA, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

pmm-secuencial.s	pmm-secuencial-modificado_b.s	pmm-secuencial-modificado_c.s
call clock_gettime@PLT leaq 40004+s(%rip), %r9 leaq 48(%rsp), %r11 xorl %r10d, %r10d leaq -4(%r9), %r8 .p2align 4,,10 .p2align 3 .L3: leaq s(%rip),	call clock_gettime@PLT leaq 48(%rsp), %r10 leaq 40000+s(%rip), %r8 xorl %r9d, %r9d .p2align 4,,10 .p2align 3 .L3: leaq s(%rip), %rax movl %r9d, %edi xorl	call clock_gettime@PLT leaq 48(%rsp), %r10 leaq 40000+s(%rip), %r8 xorl %r9d, %r9d .p2align 4,,10 .p2align 3 .L3: leaq s(%rip), %rax movl %r9d, %edx xorl

```
%rax
            movl
            %r10d,
%edi
            xorl
            %esi, %esi
            .p2align
4,,10
            .p2align 3
.L4:
            movl
            (%rax),
%edx
            addq
            $8, %rax
            leal
            (%rdi,
%rdx,2), %edx
            addl
            %edx, %esi
            cmpq
            %r8, %rax
            jne
            .L4
            leaq
            4+s(%rip),
%rax
            xorl
            %ecx, %ecx
            .p2align
4,,10
            .p2align 3
.L5:
            movl
            (%rax),
%edx
            addq
            $8, %rax
            leal
            (%rdx,
%rdx,2), %edx
            subl
            %edi, %edx
            addl
            %edx, %ecx
            cmpq
            %r9, %rax
            jne
            .L5
            cmpl
            %esi, %ecx
            cmovg
            %esi, %ecx
            movl
            %ecx,
            %ecx, %ecx
            xorl
            %esi, %esi
            .p2align
4,,10
            .p2align 3
.L4:
            movl
            (%rax),
%edx
            addq
            $8, %rax
            leal
            (%rdi,
%rdx,2), %edx
            addl
            %edx, %esi
            movl
            -4(%rax),
%edx
            leal
            (%rdx,
%rdx,2), %edx
            subl
            %edi, %edx
            addl
            %edx, %ecx
            cmpq
            %rax, %r8
            jne
            .L4
            cmpl
            %ecx, %esi
            cmovl
            %esi, %ecx
            movl
            %ecx,
(%r10,%r9,4)
            addq
            $1, %r9
            cmpq
            $40000,
%r9
            jne
            .L3
            leaq
            32(%rsp),
%rsi
            xorl
            %edi, %edi
            call

clock_gettime@PLT
            %ecx, %ecx
            xorl
            %edi, %edi
            .p2align
4,,10
            .p2align 3
.L4:
            movl
            (%rax),
%esi
            addq
            $40, %rax
            leal
            (%rdx,
%rsi,2), %esi
            addl
            %esi, %edi
            movl
            -32(%rax),
%esi
            leal
            (%rdx,
%rsi,2), %esi
            addl
            %esi, %edi
            movl
            -24(%rax),
%esi
            leal
            (%rdx,
%rsi,2), %esi
            addl
            %edi, %esi
            movl
            -16(%rax),
%edi
            leal
            (%rdx,
%rdi,2), %edi
            addl
            %edi, %esi
            movl
            -8(%rax),
%edi
            leal
            (%rdx,
%rdi,2), %edi
            addl
            %esi, %edi
            movl
            -36(%rax),
%esi
            leal
            (%rsi,
%rsi,2), %esi
```

(%r11,%r10,4) addq $1, %r10 cmpq $40000, %r10 jne .L3 leaq 32(%rsp), %rsi xorl %edi, %edi call clock_gettime@PLT		subl %edx, %esi addl %esi, %ecx movl -28(%rax), %esi leal (%rsi, %rsi,2), %esi subl %edx, %esi addl %esi, %ecx movl -20(%rax), %esi leal (%rsi, %rsi,2), %esi subl %edx, %esi addl %ecx, %esi movl -12(%rax), %ecx leal (%rcx, %rcx,2), %ecx subl %edx, %ecx addl %ecx, %esi movl -4(%rax), %ecx leal (%rcx, %rcx,2), %ecx subl %edx, %ecx addl %esi, %ecx cmpq %rax, %r8 jne .L4 cmpl %ecx, %edi cmovl %edi, %ecx movl %ecx, (%r10,%r9,4)

		addq $1, %r9
		cmpq $40000, %r9
		jne .L3
		leaq 32(%rsp), %rsi
		xorl %edi, %edi
		call clock_gettime@PLT

2. 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 denominada `DAXPY` (***D**ouble precision- real **A**lpha **X** **P**lus **Y***) que multiplica un vector por una constante y los suma a otro vector (Lección 3/Tema 1):

```
for (i=1;i<=N,i++) y[i]= a*x[i] + y[i];
```

2.1. Genere 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 acarrean. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.

2.2. (Ejercicio EXTRA) Para la mejor de las opciones, obtenga los tiempos de ejecución con distintos valores de N y determine para su sistema los valores de Rmax (valor máximo del número de operaciones en coma flotante por unidad de tiempo), Nmax (valor de N para el que se consigue Rmax), y N1/2 (valor de N para el que se obtiene Rmax/2). Estime el valor de la velocidad pico (Rpico) del procesador (consulte en [4] el número de ciclos por instrucción punto flotante para la familia y modelo de procesador que está utilizando) y compárela con el valor obtenido para Rmax. -Consulte la Lección 3 del Tema 1.

CAPTURA CÓDIGO FUENTE: `daxpy.c`

```c
int main(int argc, char const *argv[])
{

    struct timespec cgt1,
                    cgt2;
    double          ncgt; //para tiempo de ejecución


    const int tamanioVector = 80000;
    const int cte = 3548;

    int vector1[tamanioVector];
    int vector2[tamanioVector];

    clock_gettime(CLOCK_REALTIME,&cgt1);

    for (int i = 0; i < tamanioVector; ++i)
    {
        vector1[i] = i;
        vector2[i] = i;

        vector2[i] += cte*vector1[i];
    }


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


    printf("V2[0]: %d\n", vector2[0]);
    printf("V2[79999]: %d\n", vector2[79999]);
    printf("tiempo:  %11.9f\n", ncgt);

    return 0;
}
```

	-O0	-Os	-O2	-O3
Tiempos ejec.	*0.001962 002*	0.00123 7994	0.000716 498	0.001530 562

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

```
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$./dgcc -O0 daxpy.c -o daxpy-0
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$gccgcc -Os daxpy.c -o daxpy-s
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$gccgcc -O2 daxpy.c -o daxpy-2
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$gccgcc -O3 daxpy.c -o daxpy-3
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$gcc./daxpy-0
V2[0]: 0
V2[79999]: 283916451
tiempo:  0.001962002
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$./d./daxpy-s
V2[0]: 0
V2[79999]: 283916451
tiempo:  0.001237994
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$./d./daxpy-2
V2[0]: 0
V2[79999]: 283916451
tiempo:  0.000716498
[JoseJavierAlonsoRamos jjavier98@jjavier98-Lenovo-ideapad-310-15IKB:~/G
II/2º/2º Semestre/AC/SEMINARIOS/Seminario_4/ejer2] 2018-05-28 lunes
$./d./daxpy-3
V2[0]: 0
V2[79999]: 283916451
tiempo:  0.001530562
```

COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

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)

daxpyO0.s	daxpyOs.s	daxpyO2.s	daxpyO3.s

call clock_gettime@PLT
movl $0, -124(%rbp)
jmp .L2
.L3:
movq -112(%rbp), %rax
movl -124(%rbp), %edx
	call clock_gettime@PLT
	shrq $2, %rbx
	xorl %eax, %eax
	leaq 0(, %rbx,4), %r12
	.L2:
	imull $3549, %eax, %edx
		call clock_gettime@PLT
		shrq $2, %r13
		xorl %edx, %edx
		xorl %eax, %eax
		leaq 0(, %r13,4), %rbx
		.p2align 4,,10
		.p2align 3
			call clock_gettime@PLT
			shrq $2, %r14
			leaq 3(%rsp), %rbx
			leaq 0(, %r14,4), %r12
			shrq $2, %rbx
			movq %r12,

```
            movslq
            %edx,
%rdx
            movl
            -
124(%rbp), %ecx
            movl
            %ecx,
(%rax,%rdx,4)
            movq
            -
96(%rbp), %rax
            movl
            -
124(%rbp), %edx
            movslq
            %edx,
%rdx
            movl
            -
124(%rbp), %ecx
            movl
            %ecx,
(%rax,%rdx,4)
            movq
            -
96(%rbp), %rax
            movl
            -
124(%rbp), %edx
            movslq
            %edx,
%rdx
            movl
            (%rax,
%rdx,4), %ecx
            movq
            -
112(%rbp), %rax
            movl
            -
124(%rbp), %edx
            movslq
            %edx,
%rdx
            movl
            (%rax,
%rdx,4), %eax
            imull
            -
132(%rbp), %eax
            addl
            %eax,
%ecx
            movq
            -
96(%rbp), %rax
            movl
            -
124(%rbp), %edx
            movslq
            %edx,
%rdx
            movl
            %ecx,
(%rax,%rdx,4)
            addl
            $1, -
124(%rbp)
.L2:
            movl
            -
124(%rbp), %eax
            cmpl
            -
128(%rbp), %eax
            jl
            .L3
            leaq
            -
64(%rbp), %rax
            movq
            %rax,
%rsi
            movl
            $0, %edi
            call
```

```
            movl
            %eax,
0(%r13,%rax,4)
            movl
            %edx,
(%r12,%rax,4)
            incq
            %rax
            cmpq
            $80000,
%rax
            jne
            .L2
            leaq
            -
56(%rbp), %rsi
            xorl
            %edi,
%edi
            call

clock_gettime@PLT
```

```
.L2:
            movl
            %edx,
(%rbx,%rax,4)
            movl
            %eax,
(%r12,%rax,4)
            addq
            $1, %rax
            addl
            $3549,
%edx
            cmpq
            $80000,
%rax
            jne
            .L2
            leaq
            -64(%rbp),
%rsi
            xorl
            %edi, %edi
            call

clock_gettime@PLT
```

```
%rax
            leaq
            0(,
%rbx,4), %r13
            shrq
            $2, %rax
            negq
            %rax
            andl
            $3, %eax
            je
            .L7
            cmpl
            $1, %eax
            movl
            $0, 0(,
%r14,4)
            movl
            $0, 0(,
%rbx,4)
            je
            .L8
            cmpl
            $3, %eax
            movl
            $1, 4(,
%r14,4)
            movl
            $3549,
4(,%rbx,4)
            jne
            .L9
            movl
            $2, 8(,
%r14,4)
            movl
            $7098,
8(,%rbx,4)
            movl
            $79997,
%r9d
            movl
            $3, -
88(%rbp)
.L2:
            movd
            -
88(%rbp), %xmm3
            movl
            $80000,
%r8d
            movl
            %eax,
%ecx
            subl
            %eax,
%r8d
            salq
            $2, %rcx
            xorl
            %eax,
%eax
            pshufd
            $0,
%xmm3, %xmm1
            movl
            %r8d,
%edi
            movdqa
            .LC1(%rip
), %xmm2
            leaq

(%r12,%rcx), %rsi
            shrl
            $2, %edi
            addq
            %r13,
%rcx
            xorl
            %edx,
%edx
            paddd
            .LC0(%rip
), %xmm1
            .p2align
4,,10
            .p2align
```

clock_gettime@PLT			3
			.L4:
			movdqa %xmm1, %xmm0
			addl $1, %edx
			movaps %xmm1, (%rsi,%rax)
			pslld $3, %xmm0
			psubd %xmm1, %xmm0
			pslld $4, %xmm0
			psubd %xmm1, %xmm0
			pslld $3, %xmm0
			psubd %xmm1, %xmm0
			pslld $2, %xmm0
			paddd %xmm1, %xmm0
			movups %xmm0, (%rcx,%rax)
			addq $16, %rax
			cmpl %edx, %edi
			paddd %xmm2, %xmm1
			ja .L4
			movl -88(%rbp), %eax
			movl %r8d, %ecx
			movl %r9d, %edx
			andl $-4, %ecx
			subl %ecx, %edx
			addl %ecx, %eax
			cmpl %ecx, %r8d
			je .L5
			imull $3548, %eax, %ecx
			movslq %eax, %rsi
			cmpl $1, %edx
			movl %eax, (%r12,%rsi,4)
			leal (%rax, %rcx), %edi
			movl %edi, 0(%r13,%rsi,4)
			leal 1(%rax), %esi
			je .L5
			movslq

			%esi, %rdi
			addl $2, %eax
			cmpl $2, %edx
			movl %esi, (%r12,%rdi,4)
			leal 3548(%rcx,%rsi), %esi
			movl %esi, 0(%r13,%rdi,4)
			je .L5
			movslq %eax, %rdx
			leal 7096(%rcx,%rax), %eax
			movl %eax, 0(%r13,%rdx,4)
			.L5:
			leaq -64(%rbp), %rsi
			xorl %edi, %edi
			call clock_gettime@PLT