2º curso / 2º cuatr.

Grado Ing. Inform.

# **Arquitectura de Computadores (AC)**

Cuaderno de prácticas. Bloque Práctico 4. Optimización de código

Estudiante (nombre y apellidos): Noelia Escalera Mejías

Grupo de prácticas y profesor de prácticas: A2 (Christian Morillas)

Fecha de entrega: 29/05/2019

Fecha evaluación en clase: 29/05/2019

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo):  $Intel @ Core^{TM} i7-4700MQ CPU @ 2.40GHz$ 

**Sistema operativo utilizado:** *KDE neon User Edition* 5.15 x86\_64

Versión de gcc utilizada: 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04)

Volcado de pantalla que muestre lo que devuelve 1scpu en la máquina en la que ha tomado las medidas

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~] 2019-05-15 miércoles
$lscpu
Arquitectura:
                                            x86 64
modo(s) de operación de las CPUs:
                                            32-bit, 64-bit
Orden de los bytes:
                                            Little Endian
CPU(s):
ista de la(s) CPU(s) en línea:
Hilo(s) de procesamiento por núcleo:
Núcleo(s) por «socket»:
«Socket(s)»
Modo(s) NUMA:
ID de fabricante:
Familia de CPU:
                                            GenuineIntel
Modelo:
Nombre del modelo:
                                            Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz
Revisión:
CPU MHz:
                                            2394.742
CPU MHz máx.:
                                            3400,0000
CPU MHz mín.:
                                            800,0000
BogoMIPS:
                                            4788.99
Virtualización:
                                            VT-x
Caché L1d:
                                            32K
Caché L1i:
                                            32K
Caché L3:
                                            6144K
CPU(s) del nodo NUMA 0:
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 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc c
puid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 movbe popcnt tsc
_deadline_timer aes xsave avx f16c rdrand lahf_lm abm cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi fle
priority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt dthe<u>rm ida arat pln pts md_clea</u>r
```

- 1. Para el núcleo que se muestra en el Figura 1, y para un programa que implemente la multiplicación de matrices con datos flotantes en doble precisión (use variables globales):
  - 1.1 Modifique el código C para reducir el tiempo de ejecución (evalúe el tiempo y modifique sólo el trozo que hace la multiplicación y el trozo que se muestra en la Figura 1). Justifique los tiempos obtenidos a partir de la modificación realizada. Incorpore los códigos modificados en el cuaderno.
  - 1.2 Genere los códigos en ensamblador 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

<u>A) MULTIPLICACIÓN DE MATRICES:</u>
CAPTURA CÓDIGO FUENTE: pmm-secuencial.c

```
int main(int argc, char ** argv){
   if (argc != 2){
      printf("Faltan las dimensiones de la matriz y vector\n");
      exit(1);
     int i, j, k, contador, N = atoi(argv[1]);
double **m1, **m2, **resultado, t_inicio, t_fin;
     m1 = (double **)malloc(N*sizeof(double *));
     for(i=0; i<N; i++){
    m1[i] = (double*)malloc(N*sizeof(double));</pre>
     resultado = (double **)malloc(N*sizeof(double));
     for(i=0; i<N; i++){
    resultado[i] = (double*)malloc(N*sizeof(double));</pre>
     m2 = (double **)malloc(N*sizeof(double *));
     for(i=0; i<N; i++){
    m2[i] = (double*)malloc(N*sizeof(double));</pre>
     for(i=0; i<N; i++){
   for(j=0; j<N; j++){
      m2[i][j] = contador;
      contador++;</pre>
     for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        resultado[i][j] = 0;
}</pre>
     t_inicio = omp_get_wtime();
     multiplicacion(m1.m2.resultado.N):
     t_fin = omp_get_wtime();
     double tiempo = t_fin - t_inicio;
     printf("resultado[0][0]=%f",resultado[0][0]);
printf("\nresultado[N-1][N-1]=%f\n",resultado[N-1][N-1]);
     printf("Tiempo: %f\n", tiempo);
     for(i=0; i<N; i++){
    free(m1[i]);</pre>
     for(i=0; i<N; i++){
    free(m2[i]);</pre>
      free(m2);
```

## 1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

**Modificación a)** —**explicación-:** Desenrrollado del bucle más interior de la multiplicación para romper secuencias de instrucciones dependientes.

Modificación b) –explicación-: Trasponemos la segunda matriz para mejorar los accesos a memoria

#### 1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura de pmm-secuencial-modificado\_a.c

```
void multiplicacion (double **m1,double **m2,double **resultado,int N){
    for (int i=0;i<N;i++){
        for (int j=0;j<N;j++){
            double res0=0,res1=0,res2=0,res3=0;
        for (int k=0;k<N;k+=4){
            res0+=resultado[i][j]+m1[i][k]*m2[k][j];
            res1+=resultado[i][j]+m1[i][k+1]*m2[k+1][j];
            res2+=resultado[i][j][k+1]*m2[k+2][j];
            res3+=resultado[i][j]+m1[i][k+2]*m2[k+2][j];
            res3+=resultado[i][j]+m1[i][k+3]*m2[k+3][j];
        }
}</pre>
                       resultado[i][j]=res0+res1+res2+res3;
int main(int argc, char ** argv){
   if (argc != 2){
      printf("Faltan las dimensiones de la matriz y vector\n");
      exit(1);
        int i, j, k, contador, N = atoi(argv[1]);
double **m1, **m2, **resultado, t_inicio, t_fin;
        m1 = (double **)malloc(N*sizeof(double *));
        for(i=0; i<N; i++){
    m1[i] = (double*)malloc(N*sizeof(double));</pre>
        resultado = (double **)malloc(N*sizeof(double));
        for(i=0; i<N; i++){
    resultado[i] = (double*)malloc(N*sizeof(double));</pre>
        m2 = (double **)malloc(N*sizeof(double *));
        for(i=0; i<N; i++){
    m2[i] = (double*)malloc(N*sizeof(double));</pre>
        contador = 1;
        for(i=0; i<N; i++){
   for(j=0; j<N; j++){
     m1[i][j] = contador;
     contador++;</pre>
        contador = 1;
        for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        m2[i][j] = contador;
        contador++;
}</pre>
        for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        resultado[i][j] = 0;
}</pre>
        t_inicio = omp_get_wtime();
        multiplicacion(m1,m2,resultado,N);
        t_fin = omp_get_wtime();
        double tiempo = t_fin - t_inicio;
        printf("Tiempo: %f\n", tiempo);
        for(i=0; i<N; i++){
    free(m1[i]);
}</pre>
        for(i=0; i<N; i++){
   free(m2[i]);</pre>
        free(m2);
        for(i=0; i<N; i++){
   free(resultado[i]);</pre>
        }
        free(resultado):
        return (0);
```

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

```
$gcc -fopenmp pmm-secuencial-modificado_a.c -o pmm-secuencial-modificado_a
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes
$./pmm-secuencial-modificado_a 4
resultado[0][0]=90.000000
resultado[N-1][N-1]=600.000000
Tiempo: 0.000003
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes
$./pmm-secuencial-modificado_a 20
resultado[0][0]=53410.000000
resultado[N-1][N-1]=1653400.000000
Tiempo: 0.000076
```

# b) Captura de pmm-secuencial-modificado\_b.c

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
void multiplicacion (double **m1.double **m2.double **resultado.int N){
     for (int i=0;i<N;i++){
  for (int j=0;j<N;j++){
    slocal = 0.0;
    for (int k=0;k<N;k++){
        slocal+=m1[i][k]*m2[j][k];
    }</pre>
                  resultado[i][j] = slocal;
      }
int main(int argc, char ** argv){
   if (argc != 2){
      printf("Faltan las dimensiones de la matriz y vector\n");
      exit(1);
      int i, j, k, fil, col, contador, N = atoi(argv[1]); double **m1, **m2, **resultado, t_inicio, t_fin, swap;
      m1 = (double **)malloc(N*sizeof(double *));
      for(i=0; i<N; i++){
    m1[i] = (double*)malloc(N*sizeof(double));</pre>
      resultado = (double **)malloc(N*sizeof(double));
      for(i=0; i<N; i++){
    resultado[i] = (double*)malloc(N*sizeof(double));</pre>
      m2 = (double **)malloc(N*sizeof(double *));
      for(i=0; i<N; i++){
    m2[i] = (double*)malloc(N*sizeof(double));</pre>
      contador = 1;
      for(i=0; i<N; i++){
   for(j=0; j<N; j++){
      m1[i][j] = contador;
      contador++;
}</pre>
      contador = 1;
     for(i=0; i<N; i++){
   for(j=0; j<N; j++){
       m2[i][j] = contador;
       contador++;
   }
}</pre>
      for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        resultado[i][j] = 0;
}</pre>
            }
      for (i=0;i<N;i++){
            for(j=i+1;j<N;j++){
    swap = m2[i][j];
    m2[i][j] = m2[j][i];
    m2[j][i] = swap;</pre>
      }
      t_inicio = omp_get_wtime();
      multiplicacion(m1,m2,resultado,N);
      t_fin = omp_get_wtime();
      double tiempo = t_fin - t_inicio;
      printf("Tiempo: %f\n", tiempo);
      for(i=0; i<N; i++){
    free(m1[i]);</pre>
      free(m1);
      for(i=0; i<N; i++){
    free(m2[i]);</pre>
      free(m2);
      for(i=0; i<N; i++){
   free(resultado[i]);</pre>
      free(resultado);
      return (0);
```

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

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes $gcc -fopenmp pmm-secuencial-modificado_b [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes $./pmm-secuencial-modificado_b 4 resultado[0][0]=09.000000 resultado[N-1][N-1]=600.000000 Tiempo: 0.000002 [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes $./pmm-secuencial-modificado_b 20 resultado[0][0]=53410.000000 resultado[N-1][N-1]=1653400.000000 Tiempo: 0.000053
```

#### 1.1. TIEMPOS:

Modificación	Breve descripción de las modificaciones	Tiempo	
		(Tamaño 1000)	
Sin modificar	Nada modificado	13.196566	
Modificación a)	Desenrrollado del bucle más interior de la	9.267315	
	multiplicación para romper secuencias de instrucciones		
	dependientes		
Modificación b)	Trasponemos la segunda matriz para mejorar los accesos a	2.863973	
	memoria		

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

Como vemos, hemos conseguido mejorar mucho los tiempos, sobre todo se mejora con la segunda modificación. Esto es debido a que tenemos que hacer muchos accesos a memoria y trasponiendo una matriz los hemos facilidado mucho.

# CÓDIGOS ENSAMBLADOR (parte de la multiplicación)

Original:

```
cltq
multiplicacion:
                                       leag>
                                               0(,%rax,8), %rdx
.LFB5:
                                       movq>
                                              -32(%rbp), %rax
   .cfi_startproc
                                       addq»
                                               %rdx, %rax
   pushq» %rbp
                                               (%rax), %rax
                                       movq»
   .cfi_def_cfa_offset 16
   .cfi_offset 6, -16
                                       movl»
                                              -8(%rbp), %edx
                                       movslq» %edx, %rdx
   movq> %rsp, %rbp
                                       salg»
                                               $3, %rdx
   .cfi_def_cfa_register 6
                                       addg>
                                               %rdx, %rax
   movq» %rdi, -24(%rbp)
                                       movsd> (%rax), %xmm0
          %rsi, -32(%rbp)
   movq»
                                       mulsd> %xmm2, %xmm0
   movq»
          %rdx, -40(%rbp)
                                       movl»
                                              -12(%rbp), %eax
   movl» %ecx, -44(%rbp)
                                       cltq
   movl» $0, -12(%rbp)
                                       leag» 0(,%rax,8), %rdx
   jmp».L2
                                       movq» -40(%rbp), %rax
.L7:
                                       addq»
                                               %rdx, %rax
   movl» $0, -8(%rbp)
                                       movq>
                                              (%rax), %rax
   jmp».L3
                                       movl»
                                              -8(%rbp), %edx
.L6:
                                       movslq» %edx, %rdx
   movl»
           $0, -4(%rbp)
                                               $3, %rdx
                                       salq»
   jmp».L4
                                   » addq»
                                               %rdx, %rax
.L5:
                                   » addsd» %xmm1, %xmm0
   movl»
           -12(%rbp), %eax
                                       movsd> %xmm0, (%rax)
   cltq
                               » addl»
.L4:
» movl»
» cmpl»
                                       addl> $1, -4(%rbp)
   leaq»
          0(,%rax,8), %rdx
   movq»
          -40(%rbp), %rax
                                               -4(%rbp), %eax
          %rdx, %rax
   addq»
                             »
.L3:
»
»
                                       cmpl>
   movq» (%rax), %rax
                                              -44(%rbp), %eax
                                       jl» .L5
          -8(%rbp), %edx
   movl»
                                               $1, -8(%rbp)
   movslq» %edx, %rdx
                                       addl≫
                                   .L3:
   salq» $3, %rdx
                                       movl»
                                               -8(%rbp), %eax
   addq»
          %rdx, %rax
                                       cmpl>
                                               -44(%rbp), %eax
   movsd» (%rax), %xmm1
                                       jl» .L6
   movl» -12(%rbp), %eax
                                       addl»
                                               $1, -12(%rbp)
   cltq
                                   .L2:
   leag»
          0(,%rax,8), %rdx
                                               -12(%rbp), %eax
                                       movl»
   movq»
          -24(%rbp), %rax
                                       cmpl>
                                              -44(%rbp), %eax
          %rdx, %rax
   addq»
                                       jl» .L7
   movq» (%rax), %rax
                                       nop
          -4(%rbp), %edx
   movl»
   movslq» %edx, %rdx
                                       popq» %rbp
                                       .cfi_def_cfa 7, 8
   salq» $3, %rdx
                                       ret
           %rdx, %rax
   addq»
                                       .cfi_endproc
   movsd> (%rax), %xmm2
                                   .LFE5:
          -4(%rbp), %eax
   movl≫
                                       .size» multiplicacion, .-multiplicacion
                                        .section» .rodata
                                       .align 8
```

#### Modificación 1:

```
$3, %rdx
multiplicacion:
                                    salq
                                                                   addsd>
                                                                            %xmm1, %xmm0
.LFB5:
                                    addq
                                            %rdx, %rax
                                                                   movsd>
                                                                            %xmm0, -16(%rbp)
                                            (%rax), %xmm2
    .cfi_startproc
                                   movsd>
                                                                   movl»
                                                                            -44(%rbp), %eax
    pushq> %rbp
                                   movl»
                                            -36(%rbp), %eax
                                                                   cltq
    .cfi_def_cfa_offset 16
                                    cltq
                                                                   leag>
                                                                            0(,%rax,8), %rdx
                                            0(,%rax,8), %rdx
    .cfi_offset 6, -16
                                    leag>
                                                                   movq>
                                                                            -72(%rbp), %rax
           %rsp, %rbp
                                   movq>
                                            -64(%rbp), %rax
                                                                   addq>
                                                                            %rdx, %rax
    .cfi_def_cfa_register 6
                                   addq»
                                            %rdx, %rax
                                                                            (%rax), %rax
                                                                   movq>
            %rdi, -56(%rbp)
                                   movq
                                            (%rax), %rax
                                                                   movl»
                                                                            -40(%rbp), %edx
    movq>
            %rsi, -64(%rbp)
                                   movl»
                                            -40(%rbp), %edx
                                                                   movslq» %edx, %rdx
            %rdx, -72(%rbp)
    movq>
                                   movslq» %edx, %rdx
                                                                            $3, %rdx
                                                                   salq»
    movl»
            %ecx, -76(%rbp)
                                    salq»
                                            $3, %rdx
                                                                            %rdx, %rax
                                                                   addq»
    movl»
            $0, -44(%rbp)
                                    addq>
                                            %rdx, %rax
                                                                            (%rax), %xmm1
                                                                   movsd>
    jmp».L2
                                   movsd>
                                            (%rax), %xmm0
                                                                   movl»
                                                                            -44(%rbp), %eax
                                            %xmm2, %xmm0
%xmm1, %xmm0
.L7:
                                   mulsd>
                                                                   cltq
    movl»
            $0, -40(%rbp)
                                   addsd>
                                                                            0(,%rax,8), %rdx
                                                                   leag
    jmp».L3
                                   movsd>
                                            -32(%rbp), %xmm1
                                                                            -56(%rbp), %rax
                                                                   movq≫
.L6:
                                   addsd>
                                            %xmm1, %xmm0
                                                                            %rdx, %rax
                                                                   addg>
    pxor>
            %xmm0, %xmm0
                                   movsd>
                                            %xmm0, -32(%rbp)
                                                                            (%rax), %rax
                                                                   movq»
    movsd>
            %xmm0, -32(%rbp)
                                   movl»
                                            -44(%rbp), %eax
                                                                   movl»
                                                                            -36(%rbp), %edx
            %xmm0, %xmm0
    pxor>
                                    cltq
                                                                   movslq» %edx, %rdx
            %xmm0, -24(%rbp)
%xmm0, %xmm0
    movsd>
                                    leag
                                            0(,%rax,8), %rdx
                                                                            $3, %rdx
                                                                   addg>
    pxor>
                                   movq>
                                            -72(%rbp), %rax
                                                                            $3, %rdx
                                                                   salq»
    movsd>
            %xmm0, -16(%rbp)
                                   addg>
                                            %rdx, %rax
                                                                            %rdx, %rax
                                                                   addq>
            %xmm0, %xmm0
    pxor>
                                   movq>
                                            (%rax), %rax
                                                                            (%rax), %xmm2
                                                                   movsd>
    movsd>
            %xmm0, -8(%rbp)
                                   movl>
                                            -40(%rbp), %edx
                                                                   movl»
                                                                            -36(%rbp), %eax
    movl»
            $0, -36(%rbp)
                                   movslq» %edx, %rdx
                                                                   cltq
    jmp».L4
                                   salq»
                                            $3, %rdx
                                                                            $3, %rax
                                                                   addq»
.L5:
                                   addg
                                            %rdx, %rax
                                                                            0(,%rax,8), %rdx
                                                                   leaq»
            -44(%rbp), %eax
                                            (%rax), %xmm1
    movl»
                                   movsd>
                                                                            -64(%rbp), %rax
                                                                   movq»
    cltq
                                   movl»
                                            -44(%rbp), %eax
                                                                            %rdx, %rax
                                                                   addq»
    leag
            0(,%rax,8), %rdx
                                    cltq
                                                                            (%rax), %rax
                                                                   movq»
            -72(%rbp), %rax
    movq>
                                    leag>
                                            0(,%rax,8), %rdx
                                                                            -40(%rbp), %edx
                                                                   movl»
                                            -56(%rbp), %rax
    addg>
            %rdx, %rax
                                   movq»
                                                                   movslq» %edx, %rdx
            (%rax), %rax
    movq>
                                   addg>
                                            %rdx, %rax
                                                                            $3, %rdx
                                                                   salq»
            -40(%rbp), %edx
                                            (%rax), %rax
    movl
                                   movq>
                                                                            %rdx, %rax
                                                                   addq>
    movslq» %edx, %rdx
                                   movl»
                                            -36(%rbp), %edx
                                                                            (%rax), %xmm0
                                                                   movsd>
    salq»
            $3, %rdx
                                   movslq» %edx, %rdx
                                                                            %xmm2, %xmm0
%xmm1, %xmm0
                                                                   mulsd>
    addg>
            %rdx, %rax
                                   addg>
                                            $1, %rdx
                                                                   addsd>
                                            $3, %rdx
    movsd≫
            (%rax), %xmm1
                                   salg>
                                                                            -8(%rbp), %xmm1
                                                                   movsd>
    movl>
            -44(%rbp), %eax
                                   addq»
                                            %rdx, %rax
                                                                            %xmm1, %xmm0
                                                                   addsd>
    cltq
                                   movsd»
                                            (%rax), %xmm2
                                                                            %xmm0, -8(%rbp)
                                                                   movsd≫
    leaq»
            0(,%rax,8), %rdx
                                   movl»
                                            -36(%rbp), %eax
                                                                   addl>
                                                                            $4, -36(%rbp)
    movq>
            -56(%rbp), %rax
                                   cltq
                                                               .L4:
    addg>
            %rdx, %rax
                                   addg>
                                            $1, %rax
                                                                            -36(%rbp), %eax
                                                                   movl»
    movq»
            (%rax), %rax
                                   leaq»
                                            0(,%rax,8), %rdx
                                                                   cmpl>
                                                                            -76(%rbp), %eax
    movl»
            -36(%rbp), %edx
                                   movq>
                                            -64(%rbp), %rax
                                                                   jl» .L5
    movslq» %edx, %rdx
                                   addq»
                                            %rdx, %rax
```

```
addsd> -24(%rbp), %xmm0
   addsd> -16(%rbp), %xmm0
           -44(%rbp), %eax
   movl»
   cltq
   leaq 0(,%rax,8), %rdx
   movq» -72(%rbp), %rax
   addq» %rdx, %rax
   movq<sup>></sup> (%rax<sup>'</sup>), %rax
movl<sup>></sup> -40(%rbp), %edx
   movslq» %edx, %rdx
   salq» $3, %rdx
   addq» %rdx, %rax
   addsd> -8(%rbp), %xmm0
   movsd> %xmm0, (%rax)
   addl» $1, -40(%rbp)
.L3:
   movl» -40(%rbp), %eax
   cmpl> -76(%rbp), %eax
   jl<sup>.</sup> .L6
   addl» $1, -44(%rbp)
.L2:
   movl» -44(%rbp), %eax
   cmpl> -76(%rbp), %eax
   jl» .L7
   nop
   popq» %rbp
    .cfi_def_cfa 7, 8
   ret
    .cfi_endproc
.LFE5:
    .size» multiplicacion, .-multiplicacion
    .section» .rodata
    .align 8
```

#### Modificación 2:

```
multiplicacion:
.LFB5:
                                                         (%rax), %xmm0
                                                 movsd>
                                                         %xmm2, %xmm0
                                                 mulsd>
    .cfi_startproc
                                                 addsd» %xmm1, %xmm0
   pushq> %rbp
                                                 cvttsd2si» %xmm0, %eax
   .cfi_def_cfa_offset 16
                                                 movl> %eax, -16(%rbp)
   .cfi_offset 6, -16
                                                         $1, -4(%rbp)
                                                 addl»
   movq> %rsp, %rbp
   .cfi_def_cfa_register 6
                                             .L4:
                                                         -4(%rbp), %eax
                                                 movl»
   movq> %rdi, -24(%rbp)
           %rsi, -32(%rbp)
                                                 cmpl>
                                                         -44(%rbp), %eax
   movq>
           %rdx, -40(%rbp)
                                                 jl» .L5
   movq»
           %ecx, -44(%rbp)
                                                 movl»
                                                         -12(%rbp), %eax
   movl»
            $0, -12(%rbp)
                                                 cltq
   movl»
                                                 leag>
                                                         0(,%rax,8), %rdx
    jmp».L2
                                                 movq>
                                                         -40(%rbp), %rax
.L7:
                                                 addq>
                                                         %rdx, %rax
            $0, -8(%rbp)
   movl»
                                                 movq>
                                                         (%rax), %rax
    jmp».L3
                                                         -8(%rbp), %edx
                                                 movl»
.L6:
                                                 movslq» %edx, %rdx
            $0, -16(%rbp)
   movl»
            $0, -4(%rbp)
                                                 salq»
                                                         $3, %rdx
   movl>
                                                         %rdx, %rax
                                                 addq>
    jmp».L4
                                                 cvtsi2sd» -16(%rbp), %xmm0
.L5:
                                                 movsd> %xmm0, (%rax)
               -16(%rbp), %xmm1
   cvtsi2sd>
                                                         $1, -8(%rbp)
          -12(%rbp), %eax
                                                 addl≫
   movl≫
                                             .L3:
   cltq
                                                 movl»
                                                         -8(%rbp), %eax
           0(,%rax,8), %rdx
   leag
                                                         -44(%rbp), %eax
           -24(%rbp), %rax
                                                 cmpl>
   movq>
                                                 jl» .L6
            %rdx, %rax
   addg>
                                                         $1, -12(%rbp)
                                                 addl>
           (%rax), %rax
   movq>
                                             .L2:
            -4(%rbp), %edx
   movl»
                                                         -12(%rbp), %eax
                                                 movl»
   movslq» %edx, %rdx
            $3, %rdx
                                                         -44(%rbp), %eax
                                                 cmpl>
   salg
                                                 jl» .L7
            %rdx, %rax
   addg>
                                                 nop
   movsd> (%rax), %xmm2
           -8(%rbp), %eax
                                                 popq>
                                                         %rbp
   movl≫
                                                 .cfi_def_cfa 7, 8
   cltq
           0(,%rax,8), %rdx
                                                 ret
   leaq»
           -32(%rbp), %rax
                                                 .cfi_endproc
   movq>
                                             .LFE5:
   addq»
            %rdx, %rax
                                                 .size> multiplicacion, .-multiplicacion
            (%rax), %rax
   movq»
                                                             .rodata
   movl»
            -4(%rbp), %edx
                                                 .section>
                                                 .align 8
   movslq» %edx, %rdx
    salq»
            $3, %rdx
    addq
            %rdx, %rax
```

La principal diferencia que vemos es que en la primera modificación tenemos muchas más instrucciones, lo cuál es lógico debido a el desenrollamiento de bucle. Con la segunda modificación la mayor diferencia que apreciamos es el cambio de índices.

# **B) CÓDIGO FIGURA 1:**

CAPTURA CÓDIGO FUENTE: figura1-original.c

## 1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

**Modificación a) –explicación-:** Desenrrollamos los dos bucles interiores para romper secuencias de instrucciones dependientes.

**Modificación b) –explicación-:** Unificamos los bucles internos para reducir el número de iteraciones.

••

#### 1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figura1-modificado\_a.c

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
void multiplicacion (double *R){
    double X1, X2;
       for (int ii=0;ii<40000;ii++) {
    X1=0;    X2=0;
    double tmp0=0, tmp1=0, tmp2=0, tmp3=0;
    for(int i=0; i<55000;i+=4){
        tmp0+=2*s[i].a+ii;
        tmp1+=2*s[i+1].a+ii;
        tmp2+=2*s[i+2].a+ii;
        tmp3+=2*s[i+3].a+ii;
}</pre>
                 }
X1 = tmp0+tmp1+tmp2+tmp3;
tmp0=0, tmp1=0, tmp2=0, tmp3=0;
for(int i=0; i<5000;i+=4){
    tmp0+=3*s[i].b-ii;
    tmp1+=3*s[i].b-ii;
    tmp2+=3*s[i+3].b-ii;
    tmp3+=3*s[i+3].b-ii;
}</pre>
                 X2=tmp0+tmp1+tmp2+tmp3;
                 if (X1<X2) R[ii]=X1; else R[ii]=X2;</pre>
int main()
{
        int N=40000;
double *R, t_inicio, t_fin;
        R = (double*)malloc(N*sizeof(double));
       for(int i=0; i<5000; i++){
    s[i].a = i;
    s[i].b = i;</pre>
        t_inicio = omp_get_wtime();
       multiplicacion(R);
       t_fin = omp_get_wtime();
       double tiempo = t_fin - t_inicio;
       printf("R[0]=%f",R[0]);
printf("\nR[N-1]=%f\n",R[N-1]);
        printf("Tiempo: %f\n", tiempo);
        return (0);
```

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

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes $scc -fopenmp figural-modificado_a.c -o figural-modificado_a [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes $./figural-modificado_a [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes [NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes [Noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/
    R[N-1]=-162502500.000000
       Tiempo: 0.756480
```

#### b) Captura figura1-modificado\_b.c

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>
struct {
        int a;
int b;
}s[5000];
void multiplicacion (double *R){
   double X1, X2;
       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;
}</pre>
               if (X1<X2) R[ii]=X1; else R[ii]=X2;</pre>
       }
int main()
{
       int N=40000;
double *R, t_inicio, t_fin;
       R = (double*)malloc(N*sizeof(double));
       for(int i=0; i<5000; i++){
    s[i].a = i;
    s[i].b = i;</pre>
       t_inicio = omp_get_wtime();
       multiplicacion(R);
       t_fin = omp_get_wtime();
for(i=0; i<N; i++){
    m2[i] = (double*)malloc(N*sizeof(double));</pre>
       contador = 1;
       for(i=0; i<N; i++){
   for(j=0; j<N; j++){
      m1[i][j] = contador;
      contador++;</pre>
       contador = 1;
       for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        m2[i][j] = contador;
        contador++;
}</pre>
       for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        resultado[i][j] = 0;
}</pre>
       for (i=0;i<N;i++){
   for(j=i+1;j<N;j++){
      swap = m2[i][j];
      m2[i][j] = m2[j][i];
      m2[j][i] = swap;
}</pre>
       t_inicio = omp_get_wtime();
       multiplicacion(m1,m2,resultado,N);
        t_fin = omp_get_wtime();
```

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

[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes

$gcc -fopenmp figura1-modificado_b.c -o figura1-modificado_b
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer1] 2019-05-28 martes

$./figura1-modificado_b

R[0]=24995000.0000000
  R[N-1]=-162502500.000000
 Tiempo: 0.754052
```

#### 1.1. TIEMPOS:

Modificación	Breve descripción de las modificaciones			
Sin modificar	Sin modificar	1.104818		
Modificación a)	Desenrrollado de los dos bucles interiores para romper secuencias de instrucciones dependientes.	0.746580		
Modificación b)	Unificar los bucles internos para reducir iteraciones	0.754052		
•••				

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

Como vemos aquí, la diferencia de tiempos no es muy grande y las dos modificaciones restan más o menos lo mismo.

### CÓDIGO ENSAMBLADOR (sólo multiplicación)

Código original:

```
multiplicacion:
.LFB5:
   .cfi_startproc
   pushq> %rbp
                                        -8(%rbp), %xmm1
                                movsd»
  .cfi_def_cfa_offset 16
                                addsd> %xmm1, %xmm0
   .cfi_offset 6, -16
                                movsd> %xmm0, -8(%rbp)
  movq> %rsp, %rbp
                                addl»
                                        $1, -20(%rbp)
   .cfi_def_cfa_register 6
                            .L5:
   movq> %rdi, -40(%rbp)
                                        $4999, -20(%rbp)
                             » cmpl»
   movl»
           $0, -28(%rbp)
                                jle».L6
   jmp».L2
                                movsd> -8(%rbp), %xmm0
.L10:
                                ucomisd>-16(%rbp), %xmm0
           %xmm0, %xmm0
   pxor»
                                jbe».L12
   movsd> %xmm0, -16(%rbp)
                                movl»
                                        -28(%rbp), %eax
  pxor»
           %xmm0, %xmm0
                                cltq
   movsd> %xmm0, -8(%rbp)
                                leaq» 0(,%rax,8), %rdx
   movl»
           $0, -24(%rbp)
                                        -40(%rbp), %rax
                                movq≫
   jmp≫.L3
                                        %rdx, %rax
                                addq>
.L4:
                                movsd> -16(%rbp), %xmm0
          -24(%rbp), %eax
   movl»
                                movsd> %xmm0, (%rax)
   cltq
                                 jmp>.L9
          0(,%rax,8), %rdx
   leag»
                            .L12:
   leaq»
           s(%rip), %rax
                                        -28(%rbp), %eax
                                movl≫
          (%rdx,%rax), %eax »
   movl≫
                                cltq
   leal» (%rax,%rax), %edx |»
                                leaq» 0(,%rax,8), %rdx
 movl» -28(%rbp), %eax
                                movq> -40(%rbp), %rax
 addl» %edx, %eax
                                addq> %rdx, %rax
  cvtsi2sd> %eax, %xmm0
                                movsd> -8(%rbp), %xmm0
» movsd» -16(%rbp), %xmm1
                                movsd> %xmm0, (%rax)
   addsd> %xmm1, %xmm0
                             .L9:
   movsd> %xmm0, -16(%rbp)
                                        $1, -28(%rbp)
                                addl≫
   addl» $1, -24(%rbp)
                             .L2:
.L3:
                                        $39999, -28(%rbp)
                                cmpl>
   cmpl>
           $4999, -24(%rbp)
                                 jle».L10
   jle».L4
                                nop
           $0, -20(%rbp)
   movl»
                                popq»
                                        %rbp
   jmp≫.L5
                                .cfi_def_cfa 7, 8
.L6:
                                ret
   movl»
           -20(%rbp), %eax
                                 .cfi_endproc
   cltq
                             .LFE5:
           0(,%rax,8), %rdx
                                        multiplicacion, .-multiplicacion
   leag>
                            » .size»
           4+s(%rip), %rax
   leaq»
                                 .section>
                                            .rodata
           (%rdx,%rax), %edx
   movl≫
   movl»
           %edx, %eax
           %eax, %eax
   addl»
           %edx, %eax
   addl»
           -28(%rbp), %eax
   subl»
   cvtsi2sd> %eax, %xmm0
```

# Modificación 1:

	tiplicaci		>>	cvtsi2s	d» %eax, %xmm0	>>	movl»	\$0, -52(%rbp)
.LF				movsd»	-40(%rbp), %xmm1		jmp».L5	40, 52( s. 5p)
»	.cfi_sta	artnroc		addsd»	%xmm1, %xmm0	.L6:		
»	pushq»			movsd>	%xmm0, -40(%rbp)	»	movl»	-52(%rbp), %eax
				movl»			cltq	-32(-61 bp), -seax
>>		f_cfa_offset 16			-56(%rbp), %eax			0/ 9-may 0) 9-mdy
>>		fset 6, -16		addl»	\$2, %eax		leaq»	0(,%rax,8), %rdx
>>		%rsp, %rbp		cltq	0(0 0) 0 1		leaq»	4+s(%rip), %rax
>>				leaq»	0(,%rax,8), %rdx		movl»	(%rdx,%rax), %edx
>>		%rdi, -72(%rbp)		leaq»	s(%rip), %rax		movl»	%edx, %eax
>>		\$0, - <mark>60</mark> (%rbp)		movl»	, , ,		addl»	%eax, %eax
>>	jmp≫.L2			leal»	(%rax,%rax), %edx		addl»	%edx, %eax
.L1	0:			movl»	-60(%rbp), %eax		subl»	-60(%rbp), %eax
>>	pxor»	%xmm0, %xmm0		addl»	%edx, %eax		cvtsi2so	l⊳ %eax,%xmm0
>>	movsd»	%xmm0, -16(%rbp)		cvtsi2s	d⊳ %eax,%xmm0		movsd>	-48(%rbp), %xmm1
>>	pxor»	%xmm0, %xmm0		movsd>	-32(%rbp), %xmm1		addsd>	%xmm1, %xmm0
>>	movsd»	%xmm0, -8(%rbp)		addsd>	%xmm1, %xmm0		movsd>	%xmm0, -48(%rbp)
>>	pxor»			movsd>	%xmm0, -32(%rbp)		movl»	-52(%rbp), %eax
>>	movsd>	%xmm0, -48(%rbp)		movl»	-56(%rbp), %eax		addl»	\$1, %eax
>>	pxor»			addl»	\$3, %eax		cltq	-
>>	movsd»	%xmm0, -40(%rbp)		cltq	. ,		lead»	0(,%rax,8), %rdx
>>	pxor»			lead»	0(,%rax,8), %rdx		leag	4+s(%rip), %rax
>>	movsd»	%xmm0, -32(%rbp)		leag	s(%rip), %rax		movĺ»	(%rdx,%rax), %edx
>>	pxor»			movĺ≫	/o 1 o 1		movl»	%edx, %eax
>>	movsd>	%xmm0, -24(%rbp)		leal»			addl»	%eax, %eax
>>	movl»	A		movl»	-60(%rbp), %eax		addl»	%edx, %eax
>>	jmp≫.L3	7-,,,		addl≫	%edx, %eax		subl»	-60(%rbp), %eax
.L4				cvtsi2s			cvtsi2so	
>>	movl»	-56(%rbp), %eax		movsd>	-24(%rbp), %xmm1		movsd»	-40(%rbp), %xmm1
>>	cltq			addsd>	%xmm1, %xmm0		addsd»	%xmm1, %xmm0
>>	leag»	0(,%rax,8), %rdx		movsd>	%xmm0, -24(%rbp)		movsd>	%xmm0, -40(%rbp)
>>	leaq»	s(%rip), %rax		addl»	\$4, -56(%rbp)		movl»	-52(%rbp), %eax
»	movl»	(%rdx,%rax), %ea			T., 50(010p)		addl»	\$2, %eax
//  >>	leal»	(%rax,%rax), %ed		cmpl»	\$4999, - <mark>56</mark> (%rbp)		cltq	42, ocun
%	movl»	-60(%rbp), %eax		jle».L4	4-222, -20(-010p)		leag»	0(,%rax,8), %rdx
//  >>	addl»	- 1 '		movsd»	-48(%rbp), %xmm0		leag»	4+s(%rip), %rax
>>	cvtsi2s			addsd>	-40(%rbp), %xmm0		movl»	(%rdx,%rax), %edx
	movsd>			addsd»			movl»	
»	addsd>	-48(%rbp), %xmm1			-32(%rbp), %xmm0		addl»	%edx, %eax
>>				movsd»	-24(%rbp), %xmm1			%eax, %eax
>>	movsd»	%xmm0, -48(%rbp)		addsd>	%xmm1, %xmm0		addl»	%edx, %eax
>>		-56(%rbp), %eax		movsd»	%xmm0, -16(%rbp)			-60(%rbp), %eax
>>	addl»	\$1, %eax		pxor»	%xmm0, %xmm0		cvtsi2so	
>>	cltq	0/ 0 0) 0 1		movsd»	%xmm0, -48(%rbp)		movsd»	-32(%rbp), %xmm1
>>	leaq»	0(,%rax,8), %rdx		pxor»	%xmm0, %xmm0		addsd>	%xmm1, %xmm0
>>	leaq»	-,p / ,		movsd»	%xmm0, -40(%rbp)		movsd»	%xmm0, -32(%rbp)
>>	movl»	(%rdx,%rax), %ea		pxor»	%xmm0, %xmm0		movl»	-52(%rbp), %eax
>>	leal»	(%rax,%rax), %ed		movsd>	%xmm0, -32(%rbp)		addl»	\$3, %eax
>>	movl»	-60(%rbp), %eax		pxor»	%xmm0, %xmm0		cltq	
>>	addl»	%edx, %eax	>>	movsd>	%xmm0, -24(%rbp)	>>	leaq»	0(,%rax,8), %rdx

```
addsd» -24(%rbp), %xmm0
                                » .cfi_endproc
   addsd> -16(%rbp), %xmm0
                                .LFE5:
   movl» -44(%rbp), %eax
                                 » .size» multiplicacion, .-multiplicacion
                                » .section» .rodata
   cltq
» leaq» 0(,%rax,8), %rdx
» movq» -72(%rbp), %rax
» addq» %rdx, %rax
» movq
» (%rax), %rax
» movl
» -40(%rbp), %edx
» movslq» %edx, %rdx
  salq» $3, %rdx
addq» %rdx, %rax
  addsd> -8(%rbp), %xmm0
» movsd» %xmm0, (%rax)
   addl» $1, -40(%rbp)
.L3:
   movl> -40(%rbp), %eax cmpl> -76(%rbp), %eax
   jl» .L6
   addl» $1, -44(%rbp)
.L2:
  movl» -44(%rbp), %eax
   cmpl> -76(%rbp), %eax
   jl» .L7
   nop
  popq» %rbp
   .cfi_def_cfa 7, 8
  ret
   .cfi_endproc
.LFE5:
  .size» multiplicacion, .-multiplicacion
   .section> .rodata
   .align 8
```

#### Modificación 2:

```
multiplicacion:
.LFB5:
   .cfi_startproc
   pushq> %rbp
                                         movsd» (%rax), %xmm0
   .cfi_def_cfa_offset 16
                                         mulsd» %xmm2, %xmm0
   .cfi_offset 6, -16
                                         addsd> %xmm1, %xmm0
   movq> %rsp, %rbp
                                         cvttsd2si» %xmm0, %eax
   .cfi_def_cfa_register 6
                                         movl> %eax, -16(%rbp)
   movq> %rdi, -40(%rbp)
                                         addl> $1, -4(%rbp)
   movl»
           $0, -24(%rbp)
                                      .L4:
   jmp».L2
                                         movl»
                                                 -4(%rbp), %eax
.L8:
                                                 -44(%rbp), %eax
                                         cmpl>
   pxor»
           %xmm0, %xmm0
                                         jl» .L5
           %xmm0, -16(%rbp)
   movsd>
                                                 -12(%rbp), %eax
                                         movl»
           %xmm0, %xmm0
   pxor»
                                         cltq
   movsd> %xmm0, -8(%rbp)
                                                 0(,%rax,8), %rdx
                                         leag»
   movl»
           $0, -20(%rbp)
                                         movq≫
                                                 -40(%rbp), %rax
   jmp».L3
                                         addg>
                                                 %rdx, %rax
.L4:
                                         movq≫
                                                 (%rax), %rax
   movl»
           -20(%rbp), %eax
                                         movl»
                                                 -8(%rbp), %edx
   cltq
                                         movslq» %edx, %rdx
   leag
           0(,%rax,8), %rdx
                                         salq» $3, %rdx
   leag
           s(%rip), %rax
                                         addq» %rdx, %rax
   movl»
           (%rdx,%rax), %eax
           (%rax,%rax), %edx
-24(%rbp). %eax
                                         cvtsi2sd> -16(%rbp), %xmm0
   leal»
                                         movsd> %xmm0, (%rax)
   movl»
           -24(%rbp), %eax
          %edx, %eax
                                         addl≫
                                                 $1, -8(%rbp)
   addl»
                                     .L3:
   cvtsi2sd> %eax, %xmm0
                                         movl»
                                                 -8(%rbp), %eax
   movsd» -16(%rbp), %xmm1
                                         cmpl>
                                                 -44(%rbp), %eax
   addsd> %xmm1, %xmm0
                                         jl» .L6
   movsd> %xmm0, -16(%rbp)
                                         addl»
                                                 $1, -12(%rbp)
   movl» -20(%rbp), %eax
                                     .L2:
   cltq
                                         movl»
                                                 -12(%rbp), %eax
   leaq»
           0(,%rax,8), %rdx
                                         cmpl>
                                                 -44(%rbp), %eax
   leaq»
           4+s(%rip), %rax
                                         il» .L7
           (%rdx,%rax), %edx
   movl»
                                         nop
           %edx, %eax
   movl»
                                                 %rbp
                                         popq»
   addl»
           %eax, %eax
                                         .cfi_def_cfa 7, 8
   addl» %edx, %eax
                                         ret
           -<mark>24</mark>(%rbp), %eax
   subl»
                                          .cfi_endproc
   cvtsi2sd» %eax, %xmm0
   movsd> -8(%rbp), %xmm1
addsd> %xmm1, %xmm0
                                     » .size» multiplicacion, .-multiplicacion
                                          .section>
                                                      .rodata
   movsd> %xmm0, -8(%rbp)
                                     » .align 8
   addl≫
           $1, -20(%rbp)
.L3:
           $4999, -20(%rbp)
   cmpl>
    jle».L4
```

Las principales diferencias que encontramos son que en la primera modificación volvemos a tener más instrucciones y en la segunda tenemos menos saltos.

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 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 
$$(i=1;i\leq N,i++)$$
  $y[i]=a*x[i] + y[i];$ 

- 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. Sólo se debe evaluar el tiempo del núcleo DAXPY
- 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 y compárela con el valor obtenido para Rmax. -Consulte la Lección 3 del Tema 1.

CAPTURA CÓDIGO FUENTE: daxpy.c

```
#include estd(ib,h>
#include comp.h>
#include comp.h>
#include ctine,h>

void daxpy(double a, double *x){
    for (int i=1;k=100000000;i++) y[i]= a*x[i] + y[i];
}

int main(){
    double a = 100, *y, *x, t_inicio, t_fin;
    y = (double*)malloc(100000000*sizeof(double));
    x = (double*)malloc(100000000*sizeof(double));

    for (int i=0; k<100000000; i++){
        y[i] = i;
        x[i] = i;
    }

    t_inicio = omp_get_wtime();

    daxpy(a,y,x);

    t_fin = omp_get_wtime();

    double tiempo = t_fin - t_inicio;
    printf(*y[o]=xf*,y[o]);
    printf(*y[o]=xf*,y[o]);
    printf(*y[o]=xf*,y[o]);
    printf(*y[o]=xf*,y[o]);
    printf(*Tiempo: %f\n^*, tiempo);
    return (0);
}</pre>
```

m· ·	-O0	-Os	-O2	-O3
Tiempos ejec.	0.351345	0.232318	0.241435	0.234977

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

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:∼/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$qcc -00 -fopenmp daxpy.c -o daxpy
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
 ./daxpy
y[0]=0.000000
y[N-1]=10099999899.000000
Tiempo: 0.351345
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$gcc -Os -fopenmp daxpy.c -o daxpy
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$./daxpy
v[0]=0.000000
y[N-1]=10099999899.000000
 iempo: 0.232318
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$gcc -O2 -fopenmp daxpy.c -o daxpy
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$./daxpy
y[0]=0.000000
y[N-1]=10099999899.000000
 iempo: 0.241435
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$gcc -O3 -fopenmp daxpy.c -o daxpy
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp4/ejer2] 2019-05-28 martes
$./daxpy
y[0]=0.000000
y[N-1]=10099999899.000000
Tiempo: 0.234977
```

# 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)

daxpy00.s	daxpy0s.s	daxpy02.s	daxpy03.s
daxpy: .IFB5: . cfi_startproc .pushq %rbp . cfi_def_cfa_offset 16 . cfi_offset 6, -16 . movq %rsp, %rbp . cfi_def_cfa_register 6 . movd %xmm0, -24(%rbp) . movq %rdi, -32(%rbp) . movq %rsi, -40(%rbp) . movl \$1, -4(%rbp) . jmp .L2 .L3: . movl -4(%rbp), %eax . cltq . leaq 0(,%rax,8), %rdx . movd -40(%rbp), %rax . addq %rdx, %rax . movsd (%rax), %xmm0 . mulsd -24(%rbp), %xmm0 . mulsd -24(%rbp), %eax . cltq . leaq 0(,%rax,8), %rdx . movd (%rax), %xmm0 . movl -4(%rbp), %eax . cltq . leaq 0(,%rax,8), %rdx . movd -32(%rbp), %rax . addq %rdx, %rax . movd (%rax), %xmm1 . movl -4(%rbp), %eax . cltq . leaq 0(,%rax,8), %rdx . movd %rdx, %rax . movd %rdx, %rax . addq %rdx, %rax . addq %rdx, %rax . addq %rdx, %rax . addd %xmm1, %xmm0 . movd %xmm0, (%rax) . addl \$1, -4(%rbp) . L12: . cmpl \$100000000, -4(%rbp) . jle .L3 . nop . popq %rbp . cfi_def_cfa 7, 8 . ret . cfi_endproc .LFE5: . stze daxpy,daxpy . section . rodata	dexpy:   .ff_Startproc	daxpy: .LFB41:	daxpy: LFB41: LEB41: LE

Las diferencias entre Os y O2 son mínimas, se cambia básicamente el nombre de la última etiqueta y O2 usa dos instrucciones p2align. De O0 a Os hay muchísimos cambios, se eliminan muchos desplazamientos e incluso una etiqueta, queda un código mucho más compacto. En O3 podemos apreciar que se aplica desenrollar el bucle.