Cuaderno de prácticas
de Arquitectura de Computadores
Grado en Ingeniería Informática

Memoria
Bloque Práctico 4

Alumno: Manuel Jesús García Manday
DNI: 48893432-D
Grupo: *D3*

Versión de gcc utilizada: *(respuesta)*

Fichero /proc/cpuinfo de la máquina en la que ha tomado las medidas:

```
***Contenido del fichero /proc/cpuinfo ***
/* Tipo de letra Courier New. Tamaño 9.*/
/* INTERLINEADO SENCILLO */

processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model                   : 42
model name  : Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz
stepping    : 7
microcode   : 0x25
cpu MHz                 : 800.000
cache size  : 3072 KB
physical id : 0
siblings    : 4
core id                 : 0
cpu cores   : 2
apicid                  : 0
initial apicid          : 0
fpu                     : yes
fpu_exception           : yes
cpuid level : 13
wp                      : yes
flags                   : 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 rdtscp lm constant_tsc arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer xsave avx lahf_lm arat epb xsaveopt
pln pts dts tpr_shadow vnmi flexpriority ept vpid
bogomips    : 4589.65
clflush size            : 64
cache_alignment         : 64
address sizes           : 36 bits physical, 48 bits virtual
power management:

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model                   : 42
model name  : Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz
stepping    : 7
microcode   : 0x25
cpu MHz                 : 800.000
cache size  : 3072 KB
physical id : 0
siblings    : 4
core id                 : 1
cpu cores   : 2
apicid                  : 2
```

```
initial apicid          : 2
fpu                     : yes
fpu_exception           : yes
cpuid level : 13
wp                      : yes
flags                   : 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 rdtscp lm constant_tsc arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer xsave avx lahf_lm arat epb xsaveopt
pln pts dts tpr_shadow vnmi flexpriority ept vpid
bogomips    : 4589.37
clflush size            : 64
cache_alignment         : 64
address sizes           : 36 bits physical, 48 bits virtual
power management:

processor   : 2
vendor_id   : GenuineIntel
cpu family  : 6
model                   : 42
model name  : Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz
stepping    : 7
microcode   : 0x25
cpu MHz                 : 800.000
cache size  : 3072 KB
physical id : 0
siblings    : 4
core id                 : 0
cpu cores   : 2
apicid                  : 1
initial apicid          : 1
fpu                     : yes
fpu_exception           : yes
cpuid level : 13
wp                      : yes
flags                   : 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 rdtscp lm constant_tsc arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer xsave avx lahf_lm arat epb xsaveopt
pln pts dts tpr_shadow vnmi flexpriority ept vpid
bogomips    : 4589.37
clflush size            : 64
cache_alignment         : 64
address sizes           : 36 bits physical, 48 bits virtual
power management:

processor   : 3
vendor_id   : GenuineIntel
cpu family  : 6
model                   : 42
model name  : Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz
stepping    : 7
```

```
microcode   : 0x25
cpu MHz                 : 800.000
cache size  : 3072 KB
physical id : 0
siblings    : 4
core id                 : 1
cpu cores   : 2
apicid                  : 3
initial apicid          : 3
fpu                     : yes
fpu_exception           : yes
cpuid level : 13
wp                      : yes
flags                   : 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 rdtscp lm constant_tsc arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc aperfmperf pni pclmulqdq dtes64
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer xsave avx lahf_lm arat epb xsaveopt
pln pts dts tpr_shadow vnmi flexpriority ept vpid
bogomips    : 4589.36
clflush size            : 64
cache_alignment         : 64
address sizes           : 36 bits physical, 48 bits virtual
power management:
```

1. Para el núcleo que se muestra en la Figura 1, Y para un programa que implemente la multiplicación de matrices:
    a. Modifique el código C para reducir el tiempo de ejecución del mismo. Justifique los tiempos obtenidos a partir de la modificación realizada.
    b. Genere los programas en ensamblador para los programas modificados obtenidos en el punto anterior considerando las distintas opciones de optimización del compilador (-O1, -O2,...). Compare los tiempos de ejecución de las versiones de código ejecutable obtenidas con las distintas opciones de optimización y explique las diferencias en tiempo a partir de las características de dichos códigos.
    c. (Ejercicio EXTRA) Intente mejorar los resultados obtenidos transformando el código ensamblador del programa para el que se han conseguido las mejores prestaciones de tiempo

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

main()
{
   …
   for (ii=1; ii<=40000;ii++) {
      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;
   }
   …
}
```

Figura 1: Núcleo de programa en C para el ejercicio 1.

A) MULTIPLICACIÓN DE MATRICES:

CÓDIGO FUENTE: pmm-secuencial-modificado.c
(ADJUNTAR CÓDIGO FUENTE AL .ZIP)

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int main(int argc, char **argv) {

      int n, i, j, k, cont, aux1, aux2, aux3, aux4;
      struct timespec cgt1,cgt2;
     double ncgt;
     n = atoi(argv[1]); /* tomamos el tamaño de filas y columnas de la matriz */
    int **m1, **m2, **m3, **m2t; /* declaramos dinamicas las matrices */

    /* Hacemos la reserva de memoria dinamica */
   m1 = (int **) malloc(n * sizeof(int*));
   m2 = (int **) malloc(n * sizeof(int*));
   m3 = (int **) malloc(n * sizeof(int*));
   m2t = (int **) malloc(n * sizeof(int*));

  for(i = 0; i < n; i++){
       m1[i] = (int *) malloc(n * sizeof(int));
       m2[i] = (int *) malloc(n * sizeof(int));
       m3[i] = (int *) malloc(n * sizeof(int));
       m2t[i] = (int *) malloc(n * sizeof(int));
  }


  if ((m1 == NULL) || (m2 == NULL) || (m3 == NULL)){
      printf("\nError en la reserva de memoria.");
     exit(-1);
 }

  srand(time(NULL));
  /* inicializamos las matrices */
  for(i = 0; i < n; i++){
      for(j = i; j < n; j++){
          m1[i][j] = ((rand() % 4)+1);
          m2[i][j] = ((rand() % 4)+1);
          m3[i][j] = 0;
      }
  }
```

```
 /* realizamos la multiplicacion */
 clock_gettime(CLOCK_REALTIME,&cgt1);
 /* calculamos la traspuesta de m2 */
 for(i = 0; i < n; i++)
     for(j = 0; j < n; j++)
         m2t[i][j] = m2[j][i];

 for(i = 0; i < n; i++){
    for(j = 0; j < n; j++){
        for(k = 0; k < n; k+=4){
            aux1 = m1[i][k] * m2t[j][k];
            aux2 = m1[i][k+1] * m2t[j][k+1];
            aux3 = m1[i][k+2] * m2t[j][k+2];
            aux4 = m1[i][k+3] * m2t[j][k+3];
        }
       m3[i][j] = aux1 + aux2 + aux3 + aux4;
        aux1 = aux2 = aux3 = aux4 = 0;
    }
 }
 clock_gettime(CLOCK_REALTIME,&cgt2);
  ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+ (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

  printf("Tiempo(seg.):%11.9f\t",ncgt);
  printf("\n\n");
  printf("Valor (0,0): %d \n", m3[0][0]);
  printf("Valor (n-1,n-1): %d \n", m3[n-1][n-1]);
  printf("\n");
}
```

MODIFICACIONES REALIZADAS:

Modificación a) –explicación-:

desenrollado de bucle.

Modificación b) –explicación-:

cálculo de la traspuesta de m2.

Modificación c) -explicación:

desenrollado de bucle y cálculo de la traspuesta de m2.

…

Modificación	**-O0**	**-O1**	**-O2**	**-O3**	**-Os**
Sin modificar	12.538289798	11.862549792	12.117431448	11.967688910	11.408791704
Modificación a)	6.001910627	3.012976959	2.976773625	2.969447707	2.925181120
Modificación b)	7.554243515	1.234797765	1.289489280	0.805577075	3.094176859
Modificación c)	3.969484593	0.823269841	0.793828851	0.796080590	1.063743255

COMENTARIOS SOBRE LOS RESULTADOS:

Las pruebas se ha realizado con una matriz de 1000 filas y 1000 columnas. Vemos como sin realizar ninguna modificación los tiempos se disparan en relación con las modificaciones. En ambos casos tomando el modo de compilación -O0 los tiempos de ejecución son muy superiores al resto, entre los que existe cierta similutud excepto la opcion -Os en algunos casos, que dependiendo del tipo de modificación que realicemos será mejor o no.

Para el desenrollado de bucle hemos probado con saltos de 2, 4, 6 y 8 iteraciones, siendo la de 4 la que mejor resultado ha dado, de ahí que tengamos 4 variables auxiliares que sean las que almacen el valor para luego sumarse entre si. Con esta modificación vemos como con la opción de compilación -O0 el tiempo baja a la mitad con respecto al resultado sin modificar, pero donde notamos mas diferencia en ahorro de tiempo de ejecución es en el resto de opciones de compilado, siendo la opción -Os la que menor tiempo necesita.

En cuanto al cálculo de la traspuesta de m2, lo hemos realizado para que sea menos costoso el acceso a memoria, de esta manera puede encontrar varios datos de una misma columna en una misma línea de cache, de no haber realizado esta modificación se sufriría un acceso a memoria mas costoso ya que habría que acceder a muchas líneas de caché distintas. En este caso salvo con la opción de compilado -O0 donde el tiempo es superior a los obtenidos anteriormente y -Os que son algo parecidos, con el resto de opciones -O1, -O2 y -O3 existe una notable bajada de tiempo de ejecución.

La tercera modificación realizada es la unión de las dos anteriores y vemos que sin duda es la mejor de todas, ya que los resultados obtenidos sufren una importante reducción en el tiempo de ejecución con respecto a las misma por separado, teniendo un tiempo por debajo de 1 segundo con las opciones -O1, -O2 y -O3, en comparación con los mas de doce segundos que obteniamos con estas opciones de compilado en el código sin modificar.

CAPTURAS DE PANTALLA:

Sin modificar

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 pmm-secuencial.c -o pmm-secuencial -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial 1000
Tiempo(seg.):12.538289798
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 pmm-secuencial.c -o pmm-secuencial -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial 1000
Tiempo(seg.):11.862549792
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 pmm-secuencial.c -o pmm-secuencial -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial 1000
Tiempo(seg.):12.117431448
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 pmm-secuencial.c -o pmm-secuencial -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial 1000
Tiempo(seg.):11.967688910
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os pmm-secuencial.c -o pmm-secuencial -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial 1000
Tiempo(seg.):11.408791704
```

Modificación a)

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):6.001910627
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):3.012976959
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):2.976773625
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):2.969447707
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):2.925181120
```

Modificación b)

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):7.554243515
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):1.234797765
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):1.289489280
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):0.805577075
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):3.094176859
```

Modificación c)

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):3.969484593
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):0.823269841
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):0.793828851
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):0.796080590
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os pmm-secuencial-modificado.c -o pmm-secuencial-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./pmm-secuencial-modificado 1000
Tiempo(seg.):1.063743255
```

B) CÓDIGO FIGURA 1:

CÓDIGO FUENTE: figura1-modificado.c

(ADJUNTAR CÓDIGO FUENTE AL .ZIP)

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
```

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


main(){

    srand(time(NULL));
    /* Inicializamos las variables */
    int i, ii, X1, X2, R[40000];
    struct timespec cgt1,cgt2;
    double ncgt;

    for(ii = 0; ii < 5000; ii++){
          s.a[ii] = ((rand() % 10)-5);
          s.b[ii] = ((rand() % 10)-5);
    }

    clock_gettime(CLOCK_REALTIME,&cgt1);
    for (ii = 0; ii < 40000; ii++){
        for(i = 0; i < 5000; i+=8){
                X1 = 2 * s.a[i] + ii;
                X1 = 2 * s.a[i+1] + ii;
                X1 = 2 * s.a[i+2] + ii;
                X1 = 2 * s.a[i+3] + ii;
                 X1 = 2 * s.a[i+4] + ii;
                 X1 = 2 * s.a[i+5] + ii;
                 X1 = 2 * s.a[i+6] + ii;
                 X1 = 2 * s.a[i+7] + ii;
          }

          for(i = 0; i < 5000; i+=8){
                X2 = 3 * s.b[i] - ii;
                X2 = 3 * s.b[i+1] - ii;
                X2 = 3 * s.b[i+2] - ii;
                X2 = 3 * s.b[i+3] - ii;
                X2 = 3 * s.b[i+4] - ii;
                X2 = 3 * s.b[i+5] - ii;
                X2 = 3 * s.b[i+6] - ii;
                X2 = 3 * s.b[i+7] - 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("Tiempo(seg.):%11.9f\t",ncgt);
          printf("\n\n");
          printf("Valor de R[%d]: %d",ii-100,R[ii-100]);
          printf("\n");
}
```

MODIFICACIONES REALIZADAS:

Modificación a) –explicación-:

desenrollado de bucle.

Modificación b) –explicación-:

localidad de los accesos.

Modificación c) -explicació:

desenrollado de bucle y localidad de los accesos.

Modificación	**-O0**	**-O1**	**-O2**	**-O3**	**-Os**
Sin modificar	1.381735651	0.340675681	0.360151659	0.366794546	0.361392877
Modificación a)	0.752011642	0.174392288	0.170342421	0.183979152	0.151513552
Modificación b)	1.373019156	0.278887515	0.208105110	0.182928119	0.167810185
Modificación c)	1.024215151	0.078657497	0.079418976	0.079992978	0.087831123

Una vez obtenidos los resultados en la tabla podemos ver como si no realizamos ningún tipo de modificación el mejor tiempo de ejecución lo obtenemos con la opción de modificación -O1, habiendo muy poca diferencia con el resto salvo con la opción-O0 que es sin duda la más costosa.

En el desenrollado de bucle hemos realizado como en el anterior ejercicio varias pruebas con distintos tipos de iteraciones, siendo en este caso el salto de 8 el que mejor resultado da. Vemos como con este tipo de modificación los tiempo disminuyen a la mitad en comparación a no realizar ningún tipo de modificación, siendo la opción de compilación -O0 la que menor tiempo obtiene.

Como segunda modificación hemos optado por la localidad de los accesos, ya que al existir una estructura con dos campos y según el desarrollo del programa en un primer bucle se accede solo al campo a, mientras que en un segundo bucle solo al campo b, para evitar continuos saltos hemos modificado la definición de la estructura de forma que cada campo es un vector de tantas posiciones como en principio se declaraba el vector (5000). Los tiempos de respuesta obtenidos son mejores que los obtenidos sin realizar ninguna modificación, pero algo superiores en cuanto a la modificación a) exceptuando las opciones de compilado -O3 y -Os donde los tiempos se van acercando.

Como última modificación hemos optado por una híbrida entre las dos anteriores, desenrollado de bucle y localidad de los accesos, siendo esta la que mejor resultados con diferencia da con respecto a las modificaciones anteriores, es decir, las mismas pero por separado. Exceptuando la opción de compilado -O0 donde apenas hay mejora, en el resto es muy notable la bajada del tiempo de ejecución del programa.

CAPTURAS DE PANTALLA:

Sin modificar

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):1.381735651
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.340675681
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.360151659
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.366794546
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.361392877
```

Modificación a)

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.752011642
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.174392288
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.170342421
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.183979152
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.151513552
```

Modificación b)

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):1.373019156
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.278887515
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.208105110
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.182928119
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os figura1.c -o figura1 -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1
Tiempo(seg.):0.167810185
```

Modificación c)

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 figura1-modificado.c -o figura1-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1-modificado
Tiempo(seg.):1.024215151
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 figura1-modificado.c -o figura1-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1-modificado
Tiempo(seg.):0.078657497
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 figura1-modificado.c -o figura1-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1-modificado
Tiempo(seg.):0.079418976
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 figura1-modificado.c -o figura1-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1-modificado
Tiempo(seg.):0.079992978
```

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os figura1-modificado.c -o figura1-modificado -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./figura1-modificado
Tiempo(seg.):0.087831123
```

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];
```

a. Genere los programas en ensamblador para cada una de las opciones de optimización del compilador (-O1, -O2,..) y explique las diferencias que se observan en el código justificando las mejoras en velocidad que acarrean.

b. (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) y compárela con el valor obtenido para Rmax.

CÓDIGO FUENTE: daxpy.c
(ADJUNTAR CÓDIGO FUENTE AL .ZIP)

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

main(){

   int i, n, k, *v1, *v2;
   struct timespec cgt1,cgt2;
   double ncgt;
   n = atoi(argv[1]); // tamaño del vector
   k = atoi(argv[2]); // constante de multiplicación

   /* reservamos memoria para los vectores */
   v1 = (int*) malloc(n * sizeof(int));
   v2 = (int*) malloc(n * sizeof(int))

   srand(time(NULL));
   /* inicializamos v1 */
   for(i = 0; i < n; i++)
        v1[i] = ((rand() % 5)+1);

   /* realizamos la multiplicación */
   clock_gettime(CLOCK_REALTIME,&cgt1);
   for(i = 0; i < n; i++)
       v2[i] = v1[i] * k + v2[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("Tiempo(seg.):%11.9f\t",ncgt);
   printf("\n\n");
}
```

Tiempos ejec.	-O0	-O1	-O2	-O3	-Os
	0.620773272	0.270453305	0.241879576	0.270882138	0.242634673

CAPTURAS DE PANTALLA:

```
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O0 daxpy.c -o daxpy -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./daxpy 90000000 7
Tiempo(seg.):0.620773272

jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O1 daxpy.c -o daxpy -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./daxpy 90000000 7
Tiempo(seg.):0.270453305

jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O2 daxpy.c -o daxpy -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./daxpy 90000000 7
Tiempo(seg.):0.241879576

jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -O3 daxpy.c -o daxpy -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./daxpy 90000000 7
Tiempo(seg.):0.270882138

jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ gcc -Os daxpy.c -o daxpy -lrt
jesus@jesus-SVE14A1M6EB ~/Escritorio/AC/Practicas/Seminario4 $ ./daxpy 90000000 7
Tiempo(seg.):0.242634673
```

COMENTARIOS SOBRE LAS DIFERENCIAS EN ENSAMBLADOR:

CÓDIGO EN ENSAMBLADOR: (ADJUNTAR AL .ZIP) (LIMITAR AQUÍ EL CÓDIGO INCLUÍDO A LA ZONA DEL CÓDIGO ENSAMBLADOR DONDE SE REALIZA LA OPERACIÓN CON VECTORES)

daxpyO0.s

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO AQUÍ*/
/* INTERLINEADO SENCILLO */

            call        clock_gettime
            movl        $0, -28(%rbp)
            jmp         .L4
.L5:
            movl        -28(%rbp), %eax
            cltq
            salq        $2, %rax
            addq        -48(%rbp), %rax
            movl        -28(%rbp), %edx
            movslq      %edx, %rdx
            salq        $2, %rdx
            addq        -56(%rbp), %rdx
            movl        (%rdx), %edx
            movl        %edx, %ecx
            imull       -20(%rbp), %ecx
            movl        -28(%rbp), %edx
            movslq      %edx, %rdx
            salq        $2, %rdx
            addq        -48(%rbp), %rdx
            movl        (%rdx), %edx
            addl        %ecx, %edx
```

```
            movl        %edx, (%rax)
            addl        $1, -28(%rbp)
.L4:
            movl        -28(%rbp), %eax
            cmpl        -24(%rbp), %eax
            jl          .L5
            leaq        -80(%rbp), %rax
            movq        %rax, %rsi
            movl        $0, %edi
            call        clock_gettime
```

daxpyO1.s

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO AQUÍ*/
/* INTERLINEADO SENCILLO */
            call        clock_gettime
            movq        16(%rsp), %rax
            subq        (%rsp), %rax
            cvtsi2sdq   %rax, %xmm1
            movq        24(%rsp), %rax
            subq        8(%rsp), %rax
            cvtsi2sdq   %rax, %xmm0
            divsd       .LC0(%rip), %xmm0
            addsd       %xmm1, %xmm0
            movl        $.LC1, %esi
            movl        $1, %edi
            movl        $1, %eax
            call        __printf_chk
            movl        $.LC2, %edi
            call        puts
            addq        $40, %rsp
            .cfi_remember_state
            .cfi_def_cfa_offset 56
            popq        %rbx
            .cfi_def_cfa_offset 48
            popq        %rbp
            .cfi_def_cfa_offset 40
            popq        %r12
            .cfi_def_cfa_offset 32
            popq        %r13
            .cfi_def_cfa_offset 24
            popq        %r14
            .cfi_def_cfa_offset 16
            popq        %r15
            .cfi_def_cfa_offset 8
            ret
.L2:
            .cfi_restore_state
            movq        %rsp, %rsi
            movl        $0, %edi
            call        clock_gettime
```

daxpyO2.s

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO AQUÍ*/
/* INTERLINEADO SENCILLO */
             call          clock_gettime
             movq          16(%rsp), %rax
             subq          (%rsp), %rax
             movl          $.LC1, %esi
             movl          $1, %edi
             cvtsi2sdq     %rax, %xmm0
             movq          24(%rsp), %rax
             subq          8(%rsp), %rax
             cvtsi2sdq     %rax, %xmm1
             movl          $1, %eax
             divsd         .LC0(%rip), %xmm1
             addsd         %xmm1, %xmm0
             call          __printf_chk
             movl          $.LC2, %edi
             call          puts
             addq          $40, %rsp
             .cfi_remember_state
             .cfi_def_cfa_offset 56
             popq          %rbx
             .cfi_def_cfa_offset 48
             popq          %rbp
             .cfi_def_cfa_offset 40
             popq          %r12
             .cfi_def_cfa_offset 32
             popq          %r13
             .cfi_def_cfa_offset 24
             popq          %r14
             .cfi_def_cfa_offset 16
             popq          %r15
             .cfi_def_cfa_offset 8
             ret
.L2:
             .cfi_restore_state
             movq          %rsp, %rsi
             xorl          %edi, %edi
             call          clock_gettime
```

daxpyO3.s

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO AQUÍ*/
/* INTERLINEADO SENCILLO */

             call          clock_gettime
             movq          %rbx, %rdi
             andl          $15, %edi
             shrq          $2, %rdi
             negq          %rdi
             andl          $3, %edi
             cmpl          %r14d, %edi
             cmova         %r14d, %edi
             testl         %edi, %edi
             movl          %edi, %esi
             je            .L12
             xorl          %eax, %eax
             .p2align 4,,10
```

```
                .p2align 3
.L5:
                movl            (%r12,%rax,4), %edx
                leal            1(%rax), %ecx
                imull           %r13d, %edx
                addl            %edx, (%rbx,%rax,4)
                addq            $1, %rax
                cmpl            %eax, %edi
                ja              .L5
                cmpl            %edi, %r14d
                je              .L11
.L4:
                subl            %edi, %r14d
                movl            %r14d, %edi
                shrl            $2, %edi
                leal            0(,%rdi,4), %r9d
                testl           %r9d, %r9d
                je              .L7
                movl            %r13d, 12(%rsp)
                salq            $2, %rsi
                xorl            %eax, %eax
                movd            12(%rsp), %xmm0
                leaq            (%r12,%rsi), %r8
                xorl            %edx, %edx
                addq            %rbx, %rsi
                pshufd          $0, %xmm0, %xmm2
                .p2align 4,,10
                .p2align 3
.L8:
                movdqu          (%r8,%rax), %xmm1
                addl            $1, %edx
                movdqa          %xmm2, %xmm3
                movdqa          %xmm1, %xmm0
                psrldq          $4, %xmm1
                psrldq          $4, %xmm3
                pmuludq         %xmm3, %xmm1
                pshufd          $8, %xmm1, %xmm1
                pmuludq         %xmm2, %xmm0
                pshufd          $8, %xmm0, %xmm0
                punpckldq       %xmm1, %xmm0
                paddd           (%rsi,%rax), %xmm0
                movdqa          %xmm0, (%rsi,%rax)
                addq            $16, %rax
                cmpl            %edi, %edx
                jb              .L8
                addl            %r9d, %ecx
                cmpl            %r9d, %r14d
                je              .L11
.L7:
                movslq          %ecx, %rdx
                xorl            %eax, %eax
                salq            $2, %rdx
                addq            %rdx, %rbx
                addq            %rdx, %r12
                .p2align 4,,10
                .p2align 3
.L10:
                movl            (%r12,%rax,4), %edx
                imull           %r13d, %edx
                addl            %edx, (%rbx,%rax,4)
                addq            $1, %rax
                leal            (%rcx,%rax), %edx
```

```
            cmpl        %ebp, %edx
            jl          .L10
.L11:
            leaq        32(%rsp), %rsi
            xorl        %edi, %edi
            call        clock_gettime
```

daxpyOs.s

```
/* Tipo de letra Courier New. Tamaño 9.*/
/* COPIAR Y PEGAR CÓDIGO AQUÍ*/
/* INTERLINEADO SENCILLO */

            call        clock_gettime
            xorl        %eax, %eax
            jmp         .L4
.L5:
            movl        0(%rbp,%rax,4), %edx
            imull       %r13d, %edx
            addl        %edx, (%r12,%rax,4)
            incq        %rax
.L4:
            cmpl        %eax, %ebx
            jg          .L5
            leaq        16(%rsp), %rsi
            xorl        %edi, %edi
            call        clock_gettime
```

Una vez obtenidos los ficheros con el código en ensamblador de cada una de las opciones de compilado pasamos a analizarlos.

-O0

ocupa 29 instrucciones máquina, comenzando la primera llamada en la instrucción 83 y terminando con la última llamada en la instrucción 113.

-O1

ocupa 34 instrucciones máquina, comenzando la primera llamada en la instrucción 89 y terminando con la última llamada en la instrucción 124.

-O2

ocupa 34 instrucciones máquina, comenzando la primera llamada en la instrucción 94 y terminando con la última llamada en la instrucción 129.

-O3

ocupa 79 instrucciones máquina, comenzando la primera llamada en la instrucción 81 y terminando con la última llamada en la instrucción 161.

-Os

ocupa 12 instrucciones máquina, comenzando la primera llamada en la instrucción 67 y terminando con la última llamada en la instrucción 80.

Vemos como la opción de compilado -Os es la que menos instrucciones máquina ejecuta con diferencia sobre las demás, siendo la opción -O3 la que mas intrucciones máquina ejecuta, algo más de seis veces que -Os. Podemos comprobar como la opción -O3 usa muchas intrucciones que -Os no las utiliza, como por ejemplo la instrucción *.p2align,* que rellena el contador de posición a un límite de almacenamiento en particular. También podemos encontrar entre otras la instrucción *punpckldq*, que realiza desempaquetados y entrelazados de elementos.