2º curso / 2º cuatr. Grados Ing. Inform.	**Arquitectura de Computadores (AC)** **Cuaderno de prácticas.** **Bloque Práctico 4. Optimización de código** Estudiante (nombre y apellidos): Jesus Losada Arauzo Grupo de prácticas y profesor de prácticas: B2 Javi

---

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo y se lista con lscpu): *(respuesta)*

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ lscpu
Arquitectura:                         x86_64
  modo(s) de operación de las CPUs:   32-bit, 64-bit
  Address sizes:                      39 bits physical, 48 bits virtual
  Orden de los bytes:                 Little Endian
CPU(s):                               8
  Lista de la(s) CPU(s) en línea:     0-7
ID de fabricante:                     GenuineIntel
  Nombre del modelo:                  Intel(R) Core(TM) i5-10210U CPU @ 1.60G
                                      Hz
    Familia de CPU:                   6
    Modelo:                           142
    Hilo(s) de procesamiento por núcleo: 2
    Núcleo(s) por «socket»:           4
    «Socket(s)»                       1
```

Sistema operativo utilizado: *(respuesta) ubuntu linux debian*

Versión de gcc utilizada: *(respuesta)*

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc
--version
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE
.
```

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

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

MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación A) –explicación-: Esta modificacion es simplemente cambiar la j por la k en

el bucle para que este más junto en memoria

Modificación B) –explicación-: Aqui desenrolamos el bucle en 8 cuentas el bucle del final

...

CÓDIGOS FUENTE MODIFICACIONES

A) Captura de pmm-secuencial-modificado_A.c

```
//En la primera modificacion vamos a cambiar k por j ya que m2 esta accediendo
//por columnas es decir carga toda la fila
// Calcular m3 = m1 * m2
clock_gettime(CLOCK_REALTIME,&cgt1);
for(i = 0; i < N; i++){
    for (k = 0; k < N; k++)
        for (j = 0; j < N; j++)
            m3[i][j] += m1[i][k] * m2[k][j];
}
clock_gettime(CLOCK_REALTIME,&cgt2);
ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
    (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
```

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

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc  -O2  pmm-secuencial-modificado_A.c -o pmm-secuencial-modificado_A -lrt
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./pmm-secuencial-modificado_A 8
Tiempo: 0.000001801      Tamaño: 8

 Matriz 1 (m1):
       0.000000        1.000000        2.000000        3.000000        4.000000        5.000000        6.000000        7.000000
       8.000000        9.000000        10.000000       11.000000       12.000000       13.000000       14.000000       15.000000
       16.000000       17.000000       18.000000       19.000000       20.000000       21.000000       22.000000       23.000000
```

B)

```
//Desenrolamos el bucle el ultimo
// Calcular m3 = m1 * m2
int nueva_iter = N/8;

int aux0,aux1,aux2,aux3,aux4,aux5,aux7,aux6,aux8, l;

clock_gettime(CLOCK_REALTIME,&cgt1);
for(i = 0; i < N; i++){
    for (j = 0; j < N; j++){
        aux0 = aux1 = aux2 = aux3 = aux4 = aux5 = aux6 = aux7 = 0;
        for (k = 0, l = 0; k < nueva_iter; k++, l+=8 ){
            //m3[i][j] += m1[i][k] * m2[k][j];
            aux0 += m1[i][l] * m2[j][l];
            aux1 += m1[i][l+1] * m2[j][l+1];
            aux2 += m1[i][l+2] * m2[j][l+2];
            aux3 += m1[i][l+3] * m2[j][l+3];
            aux4 += m1[i][l+4] * m2[j][l+4];
            aux5 += m1[i][l+5] * m2[j][l+5];
            aux6 += m1[i][l+6] * m2[j][l+6];
            aux7 += m1[i][l+7] * m2[j][l+7];
            //El desenrole
        }
        aux8 = aux0 + aux1 + aux2 + aux3 + aux4 + aux5 + aux6 + aux7;
        m3[i][j] = aux8;

        for (l = nueva_iter*8; l < N; l++)
        {
            aux8 += m1[i][l] * m2[j][l];
        }
        m3[i][j] = aux8;
    }

}
```

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc  -O2  pmm-secuencial-modificado_B.c -o pmm-secuencial-modificado_B -lrt
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./pmm-secuencial
pmm-secuencial              pmm-secuencial-modificado_A  pmm-secuencial-modificado_B
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./pmm-secuencial-modificado_B 8
Tiempo: 0.000001395      Tamaño: 8

 Matriz 1 (m1):
       0.000000        1.000000        2.000000        3.000000        4.000000        5.000000        6.000000        7.000000
       8.000000        9.000000        10.000000       11.000000       12.000000       13.000000       14.000000       15.000000
       16.000000       17.000000       18.000000       19.000000       20.000000       21.000000       22.000000       23.000000
       24.000000       25.000000       26.000000       27.000000       28.000000       29.000000       30.000000       31.000000
       32.000000       33.000000       34.000000       35.000000       36.000000       37.000000       38.000000       39.000000
       40.000000       41.000000       42.000000       43.000000       44.000000       45.000000       46.000000       47.000000
       48.000000       49.000000       50.000000       51.000000       52.000000       53.000000       54.000000       55.000000
       56.000000       57.000000       58.000000       59.000000       60.000000       61.000000       62.000000       63.000000
```

TIEMPOS:

Modificación	Breve descripción de las modificaciones	-O2
Sin modificar	*Ninguna modificacion*	*0.000002103*
Modificación A)	Cambio en el bucle j por k y lo mismo en el tercero	0.000001801
Modificación B)	Se ha desenrolado el ultimo bucle y ahora se hacen 8 cuentas separadas y luego se suma	0.000001395
...		

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

Con los tiempos que nos ha dado cada uno podemos ver que no hay una gran

reduccion en el tiempo, si hay una diferencia notable cuando le ponemos casos más eleveados siendo el B el que menos tarda y el que más no modificarlo

2. Usar en este ejercicio el programa secuencial disponible en SWAD que utiliza como base el código de la Figura 1. Modificar en el programa el código mostrado en la Figura 1 para reducir el tiempo de ejecución. Justificar los tiempos obtenidos (usando siempre -O2) a partir de la modificación realizada. En las ejecuciones de evaluación usar valores de N y M mayores que 1000. Incorporar los códigos modificados en el cuaderno.

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

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

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

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

MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación A) –explicación-: El doble for que hace distintas operaciones los he juntado, es decir ahora se hacen las dos operaciones en el mismo bucle for

Modificación B) –explicación-: Ahora quitamos el doble bucle for y además desenrolamos el bucle.

...

CÓDIGOS FUENTE MODIFICACIONES

A) Captura figura1-modificado_A.c

```
        s[i].a=-rand()%200;
        s[i].b=rand()%100;
      }
    }

    clock_gettime(CLOCK_REALTIME,&cgt1);
        for (ii=0; ii<M;ii++){
                X1=0; X2=0;
                for(i=0; i<N;i++) X1 += 2*s[i].a + ii; X2 += 3*s[i].b - ii;
                //Unimos los dos for para que sea más optimo
                //for(i=0; i<N;i++) X2 += 3*s[i].b - ii;

                if (X1<X2) {R[ii]=X1;} else {R[ii]=X2;}
        }
    clock_gettime(CLOCK_REALTIME,&cgt2);

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

    // Imprimir resultados
    if(N<10 && M<10){
      for(i=0;i<N;i++){
```

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

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc
-O2 -o figura1-Modif_A figura1-modificado_A.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ls
BP4_Apellido1Apellido2Nombre_Y.odt  figura1-original.c
bp4.zip                             pmm-secuencial
daxpy.c                             pmm-secuencial.c
figura1-Modif_A                     pmm-secuencial-modificado_A
figura1-modificado_A.c              pmm-secuencial-modificado_A.c
figura1-modificado_B.c              pmm-secuencial-modificado_B
figura1-original                    pmm-secuencial-modificado_B.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./fi
gura1-
figura1-Modif_A   figura1-original
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./fi
gura1-Modif_A 1500 1500
Tiempo: 0.004293036
Elemento 0 y 1499 de R: -289754, 23074
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$
```

B)

```
clock_gettime(CLOCK_REALTIME,&cgt1);
    for (ii=0; ii<M;ii++){
            X1=0; X2=0;
            //Desenrolamos el bucle
            for(i=0; i<N;i+=8){
                    X1 += 2*s[i].a + ii;
                    X2 += 3*s[i].b - ii;
                    X1 += 2*s[i+1].a + ii;
                    X2 += 3*s[i+1].b - ii;
                    X1 += 2*s[i+2].a + ii;
                    X2 += 3*s[i+2].b - ii;
                    X1 += 2*s[i+3].a + ii;
                    X2 += 3*s[i+3].b - ii;
                    X1 += 2*s[i+4].a + ii;
                    X2 += 3*s[i+4].b - ii;
                    X1 += 2*s[i+5].a + ii;
                    X2 += 3*s[i+5].b - ii;
                    X1 += 2*s[i+6].a + ii;
                    X2 += 3*s[i+6].b - ii;
                    X1 += 2*s[i+7].a + ii;
                    X2 += 3*s[i+7].b - ii;

            }
            //for(i=0; i<N;i++) X2 += 3*s[i].b - ii;

            if (X1<X2) {R[ii]=X1;} else {R[ii]=X2;}
    }
clock_gettime(CLOCK_REALTIME,&cgt2);
```

```
Elemento 0 y 1499 de R: 339857, -1914839
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc
-O2 -o figura1-Modif_B figura1-modificado_B.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./fi
gura1-Modif_B 1500 1500
Tiempo: 0.005230732
Elemento 0 y 1499 de R: -1240864728, -1238610232
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$
```

TIEMPOS:

Modificación	Breve descripción de las modificaciones	-O2
Sin modificar	*Ninguna*	*0.011110691*
Modificación A)	Juntar dos for en 1	0.002106373
Modificación B)	Desenrolamiento del bucle que habia dos	0.005230732
...		

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

En la modificación a lo unico que se ha hecho es unir los dos bucles for y se nota el aumento de velocidad a la hora de ejecutar el programa, en la modificacion b he usado

la union de for y ademas el desenrole del bucle que acabo de juntar, deberia ir más rapido que el anterior pero no va eso puede deberse a que 1500 no sea multiplo de 8, creo que si lo es.

3. 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 (***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=0;i<N;i++) y[i]= a*x[i] + y[i];

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

CAPTURA CÓDIGO FUENTE: daxpy.c

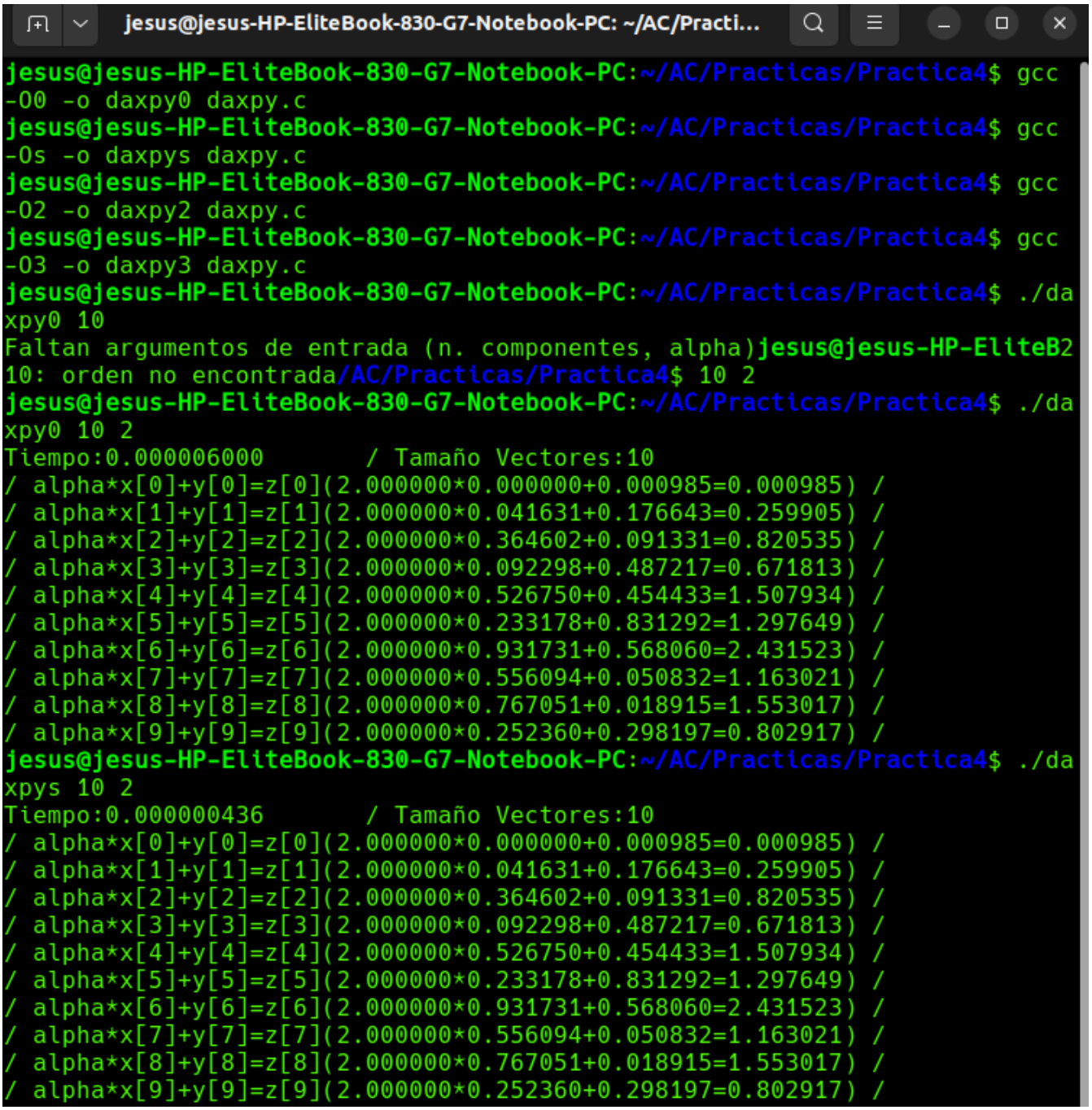
```c
/* daxpy.c
 Double precision-real Alpha x Plus y: z = alpha * x + y

 Para compilar usar (-lrt: real time library):
        gcc  -O2  daxpy.c -o daxpy -lrt

 Para ejecutar use: daxpy longitud alpha

*/

#include <stdlib.h>     // biblioteca con funciones atoi(),rand(), srand(), malloc() y free()
#include <stdio.h>      // biblioteca donde se encuentra la función printf()
#include <time.h>       // biblioteca donde se encuentra la función clock_gettime()
//#define VECTOR_LOCAL
#define VECTOR_GLOBAL
//#define VECTOR_DYNAMIC
#define VECTOR_GLOBAL

#ifdef VECTOR_GLOBAL
#define MAX 33554432    //=2^25

double x[MAX], y[MAX], z[MAX];
#endif

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

  int i;

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

  //Leer argumento de entrada (nº de componentes del vector)
  if (argc<3){
    printf("Faltan argumentos de entrada (n. componentes, alpha)");
    exit(-1);
  }
```

```c
  int N = atoi(argv[1]);        // Máximo N =2^32-1=4294967295 (sizeof(int) = 4 B)
  double alpha = atof(argv[2]);
  #ifdef VECTOR_LOCAL
  double x[N], y[N], z[N];   // Tamaño variable local en tiempo de ejecución ...
                             // disponible en C a partir de C99
  #endif
  #ifdef VECTOR_GLOBAL
  if (N>MAX) N=MAX;
  #endif
  #ifdef VECTOR_DYNAMIC
  float *x, *y, *z;
  x = (float*) malloc(N*sizeof(float));// malloc necesita el tamaño en bytes
  y = (float*) malloc(N*sizeof(float));
  z = (float*) malloc(N*sizeof(float));
  #endif

  //Inicializar vectores
  if (N < 9)
    for (i = 0; i < N; i++)
    {
      x[i] = N * 0.1 + i * 0.1; y[i] = N * 0.1 - i * 0.1;
    }
  else
  {
    //srand(time(0));
    for (i = 0; i < N; i++)
    {
      x[i] = drand48();
      y[i] = drand48();
    }
  }

  clock_gettime(CLOCK_REALTIME,&cgt1);
  //Cálculos daxpyz
  for(i=0; i<N; i++)
    z[i] = alpha*x[i] + y[i];
```

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

  //Imprimir resultado de la suma y el tiempo de ejecución
  if (N<11) {
  printf("Tiempo:%11.9f\t / Tamaño Vectores:%d\n",ncgt,N);
  for(i=0; i<N; i++)
    printf("/ alpha*x[%d]+y[%d]=z[%d](%8.6f*%8.6f+%8.6f=%8.6f) /\n",
           i,i,i,alpha,x[i],y[i],z[i]);

  } else {
    printf("Tiempo:%11.9f\t / Tamaño Vectores:%d\t/ alpha*x[0]+y[0]=z[0](%8.6f*%8.6f+%8.6f=%8.6f) / / alpha*x[%d]
+y[%d]=z[%d](%8.6f*%8.6f+%8.6f=%8.6f) /\n",
           ncgt,N,alpha,x[0],y[0],z[0],N-1,N-1,N-1,alpha,x[N-1],y[N-1],z[N-1]);
  }

  #ifdef VECTOR_DYNAMIC
  free(x); // libera el espacio reservado para v1
  free(y); // libera el espacio reservado para v2
  free(z); // libera el espacio reservado para v3
  #endif
  return 0;
}
```

Tiempos ejec. Longitud vectores=10	-O0	-Os	-O2	-O3
	0.000006000	0.000000436	0.000000399	0.000000199

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


jesus@jesus-HP-EliteBook-830-G7-Notebook-PC: ~/AC/Practi...
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O0 -o daxpy0 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -Os -o daxpys daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O2 -o daxpy2 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O3 -o daxpy3 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./daxpy0 10
Faltan argumentos de entrada (n. componentes, alpha)jesus@jesus-HP-EliteB210: orden no encontrada/AC/Practicas/Practica4$ 10 2
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./daxpy0 10 2
Tiempo:0.000006000 / Tamaño Vectores:10
/ alpha*x[0]+y[0]=z[0](2.000000*0.000000+0.000985=0.000985) /
/ alpha*x[1]+y[1]=z[1](2.000000*0.041631+0.176643=0.259905) /
/ alpha*x[2]+y[2]=z[2](2.000000*0.364602+0.091331=0.820535) /
/ alpha*x[3]+y[3]=z[3](2.000000*0.092298+0.487217=0.671813) /
/ alpha*x[4]+y[4]=z[4](2.000000*0.526750+0.454433=1.507934) /
/ alpha*x[5]+y[5]=z[5](2.000000*0.233178+0.831292=1.297649) /
/ alpha*x[6]+y[6]=z[6](2.000000*0.931731+0.568060=2.431523) /
/ alpha*x[7]+y[7]=z[7](2.000000*0.556094+0.050832=1.163021) /
/ alpha*x[8]+y[8]=z[8](2.000000*0.767051+0.018915=1.553017) /
/ alpha*x[9]+y[9]=z[9](2.000000*0.252360+0.298197=0.802917) /
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./daxpys 10 2
Tiempo:0.000000436 / Tamaño Vectores:10
/ alpha*x[0]+y[0]=z[0](2.000000*0.000000+0.000985=0.000985) /
/ alpha*x[1]+y[1]=z[1](2.000000*0.041631+0.176643=0.259905) /
/ alpha*x[2]+y[2]=z[2](2.000000*0.364602+0.091331=0.820535) /
/ alpha*x[3]+y[3]=z[3](2.000000*0.092298+0.487217=0.671813) /
/ alpha*x[4]+y[4]=z[4](2.000000*0.526750+0.454433=1.507934) /
/ alpha*x[5]+y[5]=z[5](2.000000*0.233178+0.831292=1.297649) /
/ alpha*x[6]+y[6]=z[6](2.000000*0.931731+0.568060=2.431523) /
/ alpha*x[7]+y[7]=z[7](2.000000*0.556094+0.050832=1.163021) /
/ alpha*x[8]+y[8]=z[8](2.000000*0.767051+0.018915=1.553017) /
/ alpha*x[9]+y[9]=z[9](2.000000*0.252360+0.298197=0.802917) /

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./da
xpy2 10 2
Tiempo:0.000000399       / Tamaño Vectores:10
/ alpha*x[0]+y[0]=z[0](2.000000*0.000000+0.000985=0.000985) /
/ alpha*x[1]+y[1]=z[1](2.000000*0.041631+0.176643=0.259905) /
/ alpha*x[2]+y[2]=z[2](2.000000*0.364602+0.091331=0.820535) /
/ alpha*x[3]+y[3]=z[3](2.000000*0.092298+0.487217=0.671813) /
/ alpha*x[4]+y[4]=z[4](2.000000*0.526750+0.454433=1.507934) /
/ alpha*x[5]+y[5]=z[5](2.000000*0.233178+0.831292=1.297649) /
/ alpha*x[6]+y[6]=z[6](2.000000*0.931731+0.568060=2.431523) /
/ alpha*x[7]+y[7]=z[7](2.000000*0.556094+0.050832=1.163021) /
/ alpha*x[8]+y[8]=z[8](2.000000*0.767051+0.018915=1.553017) /
/ alpha*x[9]+y[9]=z[9](2.000000*0.252360+0.298197=0.802917) /
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ ./da
xpy3 10 2
Tiempo:0.000000199       / Tamaño Vectores:10
/ alpha*x[0]+y[0]=z[0](2.000000*0.000000+0.000985=0.000985) /
/ alpha*x[1]+y[1]=z[1](2.000000*0.041631+0.176643=0.259905) /
/ alpha*x[2]+y[2]=z[2](2.000000*0.364602+0.091331=0.820535) /
/ alpha*x[3]+y[3]=z[3](2.000000*0.092298+0.487217=0.671813) /
/ alpha*x[4]+y[4]=z[4](2.000000*0.526750+0.454433=1.507934) /
/ alpha*x[5]+y[5]=z[5](2.000000*0.233178+0.831292=1.297649) /
/ alpha*x[6]+y[6]=z[6](2.000000*0.931731+0.568060=2.431523) /
/ alpha*x[7]+y[7]=z[7](2.000000*0.556094+0.050832=1.163021) /
/ alpha*x[8]+y[8]=z[8](2.000000*0.767051+0.018915=1.553017) /
/ alpha*x[9]+y[9]=z[9](2.000000*0.252360+0.298197=0.802917) /
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$
```

COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

Con la opcion -O0 no hay ninguna optimización , el codigo es bastante más largo que el resto además hay más sobrecarga de la pila en esta optimización que en las otras , con la opción -O2 se reduce bastante el código y hace que vaya mucho más rápido y con la opción -O3 hace demasiadas optimizaciones e incluso puede llegar a saltarse restricciones que si trabajamos con memoria dinámica puede hacer que falle nuestro código.

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O0 -S -o daxpyO0 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -Os -S -o daxpyOs daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O1 -S -o daxpyO1 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O3 -S -o daxpyO3 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$ gcc -O2 -S -o daxpyO2 daxpy.c
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica4$
```

CÓDIGO EN ENSAMBLADOR (no es necesario introducir aquí el código como captura de pantalla, ajustar el tamaño de la letra para que una instrucción no ocupe más de un renglón)**:**
(PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR DONDE ESTÁ EL CÓDIGO EVALUADO, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

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

.L11:
movl -72(%rbp), %eax
cltq
leaq 0(,%rax,8), %rdx
leaq x(%rip), %rax
movsd (%rdx,%rax), %xmm0
movapd %xmm0, %xmm1
mulsd -64(%rbp), %xmm1
movl -72(%rbp), %eax
cltq
leaq 0(,%rax,8), %rdx
leaq y(%rip), %rax
movsd (%rdx,%rax), %xmm0
addsd %xmm1, %xmm0
movl -72(%rbp), %eax
cltq
leaq 0(,%rax,8), %rdx
leaq z(%rip), %rax
movsd %xmm0, (%rdx,%rax)
addl $1, -72(%rbp)
.L10:
movl -72(%rbp), %eax
cmpl -68(%rbp), %eax
jl .L11
	.L8:
	cmpl %eax, %r12d
	jle .L20
	movq %r14, %xmm0
	mulsd (%rcx,%rax,8), %xmm0
	addsd (%rsi,%rax,8), %xmm0
	movsd %xmm0, (%rdx,%rax,8)
	incq %rax
	jmp .L8
		.L8:
		cmpl %eax, %r12d
		jle .L20
		movq %r14, %xmm0
		mulsd (%rcx,%rax,8), %xmm0
		addsd (%rsi,%rax,8), %xmm0
		movsd %xmm0, (%rdx,%rax,8)
		incq %rax
		jmp .L8
			.L12:
			movapd 0(%rbp,%rax), %xmm0
			mulpd %xmm1, %xmm0
			addpd (%r12,%rax), %xmm0
			movaps %xmm0, (%r15,%rax)
			addq $16, %rax
			cmpq %rax, %rdx
			jne .L12
			movl %ecx, %eax
			andl $-2, %eax
			andl $1, %ecx
			je .L11