2° curso / 2° cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

# Arquitectura de Computadores (AC)

Cuaderno de prácticas. Bloque Práctico 4. Optimización de código

Estudiante (nombre y apellidos): Néstor Rodríguez Vico

Grupo de prácticas: A1

Fecha de entrega:

Fecha evaluación en clase:

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz

Sistema operativo utilizado: Ubuntu 15.10

**Versión de gcc utilizada:** *gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010* 

Adjunte el contenido del fichero /proc/cpuinfo de la máquina en la que ha tomado las medidas

```
processor
vendor_id
                : GenuineIntel
cpu family
                : 6
                : 60
model
model name
                : Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
stepping
microcode
                : 0x1c
                : 2610.449
cpu MHz
cache size
                : 3072 KB
                : 0
physical id
siblings
                : 0
core id
cpu cores
                : 2
apicid
initial apicid: 0
fpu
                : yes
fpu_exception
                 yes
cpuid level
                : yes
wp
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 pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf
eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma cx16 xtpr pdcm pcid
sse4_1 sse4_2 movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm ida
arat epb pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1
avx2 smep bmi2 erms invpcid xsaveopt
bugs
bogomips
               : 4988.62
clflush size
              : 64
cache_alignment
                        : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
processor
vendor_id
               : GenuineIntel
cpu family
                : 6
model
               : 60
model name
                : Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
stepping
microcode
                : 0x1c
                : 2692.773
cpu MHz
cache size
               : 3072 KB
physical id
                : 0
siblings
```

```
core id
               : 0
cpu cores
               : 2
apicid
               : 1
initial apicid : 1
               : yes
fpu_exception : yes
cpuid level
               : 13
               : yes
wn
               : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
flags
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 aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma cx16 xtpr pdcm pcid
sse4_1 sse4_2 movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm ida
arat epb pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1
avx2 smep bmi2 erms invpcid xsaveopt
buas
bogomips
               : 4988.62
clflush size : 64
cache_alignment
                       : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
processor
               : GenuineIntel
vendor_id
cpu family
               : 6
               : 60
model
model name
               : Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
              : 3
steppina
              : 0x1c
microcode
cpu MHz
               : 2710.351
cache size
              : 3072 KB
physical id
               : 0
               : 4
siblinas
core id
               : 1
cpu cores
               : 2
apicid
initial apicid: 2
fpu
                 yes
fpu_exception
              : yes
cpuid level
              : 13
               : yes
wp
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 pdpe1gb rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf
eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm ida
arat epb pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1
avx2 smep bmi2 erms invpcid xsaveopt
buas
               : 4988.62
bogomips
clflush size : 64
cache_alignment
                       : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
processor
               : 3
               : GenuineIntel
vendor id
cpu family
               : 6
model
               : 60
model name
               : Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
              : 3
stepping
              : 0x1c
microcode
               : 2647.851
cpu MHz
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
               : yes
ш
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 pdpe1gb rdtscp lm
```

```
constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm ida arat epb pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt bugs :
bogomips : 4988.62
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:
```

- 1. Para el núcleo que se muestra en la Figura 1 (ver guion de prácticas), y para un programa que implemente la multiplicación de matrices (use variables globales):
  - 1.1 Modifique el código C para reducir el tiempo de ejecución del mismo. Justifique los tiempos obtenidos (use -O2) a partir de la modificación realizada. Incorpore los códigos modificados en el cuaderno.
  - 1.2 Genere los códigos en ensamblador con -O2 para el original y dos códigos modificados obtenidos en el punto anterior (incluido el que supone menor tiempo de ejecución) e incorpórelos al cuaderno de prácticas. Destaque las diferencias entre ellos en el código ensamblador.
  - 1.3 (Ejercicio EXTRA) Intente mejorar los resultados obtenidos transformando el código ensamblador del programa para el que se han conseguido las mejores prestaciones de tiempo

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

main()
{
   ...
   for (ii=0; ii<40000;ii++) {
        X1=0; X2=0;
        for(i=0; i<5000;i++) X1+=2*s[i].a+ii;
        for(i=0; i<5000;i++) X2+=3*s[i].b-ii;
        if (X1<X2) R[ii]=X1 else R[ii]=X2;
   }
}</pre>
```

El cambio principal que se puede realizar es la unión de los dos bucles. A ello se le puede sumar un desenrrollado del bucle. En mi caso, no se porqué, pero el tiempo de ejecución no mejora.

# A) MULTIPLICACIÓN DE MATRICES: CÓDIGO FUENTE: pmm-secuencial.c

```
// Compilar con -02 y -fopenmp
#include <stdlib.h>
#include <stdio.h>
#include <omp.h>
```

```
int main(int argc,char **argv) {
   int i, j, k;
   double t1, t2, total;
   //Leer argumento de entrada (no de componentes del vector)
   if (argc<2){
      printf("Falta tamaño de matriz\n");
      exit(-1);
   }
   unsigned int N = atoi(argv[1]); // Máximo N = 2^32 - 1 = 4294967295
(sizeof(unsigned int) = 4 B)
   double **M1, **M2, **M3;
   //M1 = M2 * M3;
             M1 = (double**) malloc(N*sizeof(double *));
  M2 = (double**) malloc(N*sizeof(double *));
  M3 = (double**) malloc(N*sizeof(double *));
             if ( (M1==NULL) || (M2==NULL) || (M3==NULL) ){
                           printf("Error en la reserva de espacio para las
matrices\n");
                           exit(-2);
             for (i=0; i<N; i++){
                           M1[i] = (double*) malloc(N*sizeof(double));
      M2[i] = (double*) malloc(N*sizeof(double));
      M3[i] = (double*) malloc(N*sizeof(double));
                           if ( M1[i]==NULL || M2[i]==NULL || M3[i]==NULL ){
                                        printf("Error en la reserva de espacio
para los matrices\n");
                                        exit(-2);
                           }
             }
             //A partir de aqui se pueden acceder las componentes de la
matriz como M[i][j]
             //Inicializar matriz y vectores
   for (j=0; j<N; j++){}
                           for (i=0; i<N; i++){
         M1[i][j] = 0;
                                        M2[i][j] = 2;
                                        M3[i][j] = 2;
                           }
             }
   //Medida de tiempo
  t1 = omp_get_wtime();
  //Calcular producto de matriz por matriz M1 = M2 * M3
   for (i=0; i<N; i++){
                           for(j=0; j<N; j++){
                                        for (k=0; k<N; k++){
                                                      M1[i][j] = M1[i][j] +
(M2[i][k]*M3[k][j]);
                                        }
                           }
```

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

Modificación a) –explicación-: Deserrollado del bucle interno. Se reducen las instrucciones de salto

Modificación b) –explicación-: Inicializar las matrices como su traspuesta. Se mejora el acceso a los datos.

#### 1.1. CÓDIGOS FUENTE MODIFICACIONES

a) pmm-secuencial-modificado\_a.c

```
// Compilar con -O2 y -fopenmp
#include <stdlib.h>
#include <stdio.h>
#include <omp.h>
int main(int argc,char **argv) {
   int h, i, j, k;
   double t1, t2, total;
   int suma = 0; int suma2 = 0;
   //Leer argumento de entrada (no de componentes del vector)
   if (argc<2){
      printf("Falta tamaño de matriz\n");
      exit(-1);
   }
   unsigned int N = atoi(argv[1]); // Máximo N = 2^32 - 1 = 4294967295
(sizeof(unsigned int) = 4 B)
   double **M1, **M2, **M3;
   //M1 = M2 * M3;
             M1 = (double**) malloc(N*sizeof(double *));
   M2 = (double**) malloc(N*sizeof(double *));
  M3 = (double**) malloc(N*sizeof(double *));
             if ( (M1==NULL) || (M2==NULL) || (M3==NULL) ){
                           printf("Error en la reserva de espacio para las
matrices\n");
                           exit(-2);
             }
             for (i=0; i<N; i++){
                           M1[i] = (double*) malloc(N*sizeof(double));
      M2[i] = (double*) malloc(N*sizeof(double));
```

```
M3[i] = (double*) malloc(N*sizeof(double));
                           if ( M1[i]==NULL || M2[i]==NULL || M3[i]==NULL ){
                                         printf("Error en la reserva de espacio
para los matrices\n");
                                         exit(-2);
                           }
             }
              //A partir de aqui se pueden acceder las componentes de la
matriz como M[i][j]
             //Inicializar matriz y vectores
   //Cambiamos el acceso a las matrices
   for (i=0; i<N; i++){
                           for (j=0; j<N; j++){}
         M1[j][i] = 0;
                                         M2[j][i] = 2;
                                         M3[j][i] = 2;
                           }
              }
   //Medida de tiempo
   t1 = omp_get_wtime();
   //Calcular producto de matriz por matriz M1 = M2 * M3
   int bucle_interno = N/8;
   for (i=0; i<N; i++){
                           for(j=0; j<N; j++){
                                         suma = 0; suma2 = 0;
                                         for (h=0, k=0; h < bucle_interno; ++h,
k+=8){}
                                                       suma += (M2[i][k]*M3[k]
[j]);
                                                       suma += (M2[i]
[k+1]*M3[k+1][j]);
                                                       suma += (M2[i]
[k+2]*M3[k+2][j]);
                                                       suma += (M2[i]
[k+3]*M3[k+3][j]);
                                                       suma2 += (M2[i]
[k+4]*M3[k+4][j]);
                                                       suma2 += (M2[i]
[k+5]*M3[k+5][j]);
                                                       suma2 += (M2[i]
[k+6]*M3[k+6][j]);
                                                       suma2 += (M2[i]
[k+7]*M3[k+7][j]);
                                         M1[i][j] = suma + suma2;
                                         for(k=bucle_interno*8; k<N; ++k){</pre>
                                                       suma += (M2[i][k]*M3[k]
[j]);
                                         M1[i][j] = suma + suma2;
                           }
             }
   //Medida de tiempo
              t2 = omp_get_wtime();
              total = t2 - t1;
```

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

#### **1.1. TIEMPOS:**

Modificación	-O2
Sin modificar	3.051889507
Modificación a+b	4.473921622

- 1.1. COMENTARIOS SOBRE LOS RESULTADOS: Una vez más, podemos ver que el resultado no mejora.
- 1.2. CÓDIGO EN ENSAMBLADOR DEL ORIGINAL Y DE DOS MODIFICACIONES (ADJUNTAR AL .ZIP):

  (PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR EVALUADA USE

(PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR EVALUADA, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

pmm-secuencial.s	pmm-secuencial- modificado_b.s	pmm-secuencial- modificado_c.s
/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 */ /* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/ /* INTERLINEADO SENCILLO */	/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 */ /* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/ /* INTERLINEADO SENCILLO */	/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 */ /* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/ /* INTERLINEADO SENCILLO */

# **B) CÓDIGO FIGURA 1:**

CÓDIGO FUENTE: figura1-original.c (ADJUNTAR CÓDIGO FUENTE AL.ZIP)

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

```
struct {
             int a;
             int b;
} s[5000];
int main(int argc, char** argv) {
             int i, ii, X1, X2;
             int R[40000];
             struct timespec cgt1, cgt2;
    double tiempo;
    for(i=0; i<5000;i++)
                           s[i].a = i;
                           s[i].b = i;
   }
             clock_gettime(CLOCK_REALTIME, &cgt1);
             for (ii=0; ii<40000;ii++) {
                           X1=0; X2=0;
                           for(i=0; i<5000;i++){
                                        X1+=2*s[i].a+ii;
                           for(i=0; i<5000;i++){
                                         X2+=3*s[i].b-ii;
                           }
                           if (X1<X2)
                                         R[ii]=X1;
                           else
                                         R[ii]=X2;
             }
             clock_gettime(CLOCK_REALTIME, &cgt2);
             tiempo = (double) (cgt2.tv_sec - cgt1.tv_sec)+ (double)
((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
             printf("a: %i - b: %i", R[5000], s[4999].a);
             printf(" Tiempo(seg.): %11.9f\n", tiempo);
```

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

Modificación a) –explicación-: Union de los bucles. Solo se recorre una vez. Modificación b) –explicación-: Deserrollado del bucle interno. Se reducen las instrucciones de salto

#### 1.1. CÓDIGOS FUENTE MODIFICACIONES

a) figura1-modificado\_a.c

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

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

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

```
int i, ii, X1, X2;
        int X1a=0, X2a=0;
        int X1b=0, X2b=0;
        int X1c=0, X2c=0;
        int X1d=0, X2d=0;
        int X1e=0, X2e=0;
        int R[40000];
        struct timespec cgt1, cgt2;
double tiempo;
for(i=0; i<5000;i++) {
                    s[i].a = i;
                    s[i].b = i;
}
        clock_gettime(CLOCK_REALTIME, &cgt1);
        for (ii=0; ii<40000;ii++) {
                    /*X1=0; X2=0;
                    for(i=0; i<5000;i++){
                                 X1+=2*s[i].a+ii;
                                 X2+=3*s[i].b-ii;
                    }*/
                    X1a=0; X2a=0;
                    X1b=0; X2b=0;
                    X1c=0; X2c=0;
                    X1d=0; X2d=0;
                    X1e=0; X2e=0;
                    for(i=0; i<5000;i+=5) {
                                 X1a+=2*s[i].a+ii;
                                 X1b+=2*s[i+1].a+ii;
                                 X1c+=2*s[i+2].a+ii;
                                 X1d+=2*s[i+3].a+ii;
                                 X1e+=2*s[i+4].a+ii;
                                 X2a+=3*s[i].b-ii;
                                 X2b+=3*s[i+1].b-ii;
                                 X2c+=3*s[i+2].b-ii;
                                 X2d+=3*s[i+3].b-ii;
                                 X2e+=3*s[i+4].b-ii;
                    X1 = X1a+X1b+X1c+X1d+X1e;
                    X2 = X2a+X2b+X2c+X2d+X2e;
                    if (X1<X2)
                                 R[ii]=X1;
                    else
                                 R[ii]=X2;
        }
        clock_gettime(CLOCK_REALTIME, &cgt2);
        tiempo = (double) (cgt2.tv_sec - cgt1.tv_sec)+ (double)
```

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

```
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1mod
a: 12492500 - b: 4999 Tiempo(seg.): 0.189175596
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1
a: 12492500 - b: 4999 Tiempo(seg.): 0.279480772
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1
a: 12492500 - b: 4999 Tiempo(seg.): 0.292558591
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1
a: 12492500 - b: 4999 Tiempo(seg.): 0.284827429
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1mod
a: 12492500 - b: 4999 Tiempo(seg.): 0.171258206
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1mod
a: 12492500 - b: 4999 Tiempo(seg.): 0.177811670
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1mod
a: 12492500 - b: 4999 Tiempo(seg.): 0.191227000
nestor@nestor:~/Dropbox/2/2C/AC/Practicas/5. Bloques/B4/2B$ ./f1mod
a: 12492500 - b: 4999 Tiempo(seg.): 0.183780309
```

#### 1.1. TIEMPOS:

Modificación	-O2
Sin modificar	0.292558591
Modificación a+b	0.171258206

- 1.1. COMENTARIOS SOBRE LOS RESULTADOS: Se produce una mejora apreciable.
- 1.2. CÓDIGO EN ENSAMBLADOR DEL ORIGINAL Y DE DOS MODIFICACIONES (ADJUNTAR AL .ZIP):

(PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR EVALUADA, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

pmm-secuencial.s		pmm-secuencial-modificado_b.s			
.file	"f1.c"	.file	"f1mod.c"		
.section	.rodata.str1	.section	.rodata.str1.		
.1, "aMS", @progbits, 1		1, "aMS", @progbits, 1	1,"aMS",@progbits,1		
.LC1:		.LC1:			
.string	"a: %i - b:	.string	"a: %i - b:		
%i"		%i"			
.LC2:		.LC2:			
.string	II .	.string	"		
Tiempo(seg.): %11.9f\n"		Tiempo(seg.): %11.9f\n"			
.section	.text.unlike	section	.text.unlikel		
ly, "ax", @progbits		y, "ax", @progbits			
.LCOLDB3:		.LCOLDB3:			
.section	.text.startu	.section	.text.startup		
p, "ax", @progbits		,"ax",@progbits	·		

### Proof	LUOTRO			LUCTRO		
	.LHOTB3:	n2olian 4 1E		.LHOTB3:	n2olian 4 1E	
### April			main			
### defunction main: .1F838: .		•			•	
Name	Afunction	. суре	IIIalii,	@function	. cype	IIIalii,
LFB38:	•			-		
.cfi_startproc subq \$160072, krsp .cfi_def_cfa_offset 160080 movq %fs:40, %rax, movq %fax, .cfi_def_cfa_offset 24 .cfi_offset 14, -24 pushq %r15 .cfi_def_cfa_offset 24 .cfi_offset 14, -24 pushq %r13 .cfi_def_cfa_offset 24 .cfi_offset 13, -32 pushq %r12 .cfi_def_cfa_offset 24 .cfi_offset 13, -32 pushq %r13 .cfi_def_cfa_offset 32 .cfi_offset 32 .cfi_offset 33, -32 pushq %r12 .cfi_offset 12, -40 pushq %r13 .cfi_offset 32 .cfi_offset 40 .cfi_offset 62 .cfi_offset 40 .cfi_offset 62 .cfi_offset 63 .cfi_offset 64 .cfi_offset 64 .cfi_offset 64 .cfi_offset 64 .cfi_offset 65 .cfi_offset 66 .cfi_offset 67 .cfi						
Subq	.LFB38:	ofi startarea		.LFB38;	ofi startarea	
		•	¢160072		•	0/r1E
.cfi_def_cfa_offset 160808   .cfi_offset 15, -16   movq	%rcn	Subq	\$100072,			
movq   movq   wfs:40, wrax   movq   wfrax   cfi_def_cfa_offset 24   cfi_offset 14, -24   pushq   wr13   cfi_def_cfa_offset 24   cfi_offset 14, -24   pushq   wr13   cfi_def_cfa_offset 32   cfi_offset 13, -32   pushq   wr12   cfi_def_cfa_offset 32   cfi_offset 13, -32   pushq   wr12   cfi_def_cfa_offset 32   cfi_offset 13, -32   pushq   wr12   cfi_def_cfa_offset 32   cfi_offset 12, -40   pushq   wr12   cfi_def_cfa_offset 48   cfi_offset 12, -40   pushq   wrbp   cfi_def_cfa_offset 48   cfi_offset 12, -40   pushq   wrbp   cfi_def_cfa_offset 48   cfi_offset 12, -40   pushq   wrbp   cfi_def_cfa_offset 48   cfi_offset 56   cfi_offset 56   cfi_offset 5, -48   pushq   wrbp   cfi_def_cfa_offset 48   cfi_offset 5, -48   pushq   wrbp   cfi_def_cfa_offset 48   cfi_offset 56   cfi_offset 5, -48   pushq   wrbp   cfi_def_cfa_offset 48   cfi_offset 56   cfi_offset 5, -48   pushq   wrbp   cfi_def_cfa_offset 56   cfi_offset 56   cfi_def_cfa_offset	701 SP	ofi dof ofo of	Ffcot 160000			
Move						
160056(%rsp)		•	•			
XOrl	160056(%rsn)	movq	/01 αχ,			
.D2align 4,10 .p2align 3  .L2:  movl	100030(701 3p)	vorl	%eax %eax			
.12:     mov1			ποταχ, ποταχ			
Move						
## Wrax,8)  ## wrax,8)  ## mov1	.12:	.p=a==g 0				
		movl	%eax. s(.			
Marax, 8   Marax, 8   Marax   Marax   Marax, 8	%rax.8)		700dri, 0()			
Addq   \$1, %rax   \$1	, • /	movl	%eax, s+4(.			
addq \$1, %rax cpnq \$5000, %rax jne	%rax,8)	<del></del>	,			•
Cmpq	, • /	addg	\$1, %rax			
Jine		•	•			
Teaq			•			
Subq   \$160072, %rsp   call   call   clock_gettime   worl   %r9d, %r9d   movl   \$s+40000,   p2align 4,,10   p2align 3   .12:   movl   %eax, \$s+4(, wax   vorl   well   worl   well   well   well   worl   well   w		-				
Xorl	%rsi	•				
Clock_gettime		xorl	%edi, %edi		.cfi_def_cfa_o	ffset 160128
Xorl		call			movq	%fs:40, %rax
Mov1		clock_gettime			movq	%rax,
### ### ##############################		xorl	%r9d, %r9d	160056(%rsp)		
.p2align 4,,10 .p2align 3 .L2:  movl		movl	\$s+40000,		xorl	%eax, %eax
.L3:    mov1	%r8d					
Mov1				_	.p2align 3	
movl movl ss, weax movl ss, weax sy movl weax, s+4(, xorl wesi, wesi npalign 3 lead standard		.p2align 3		.L2:	-	
movl	.L3:	-			movl	%eax, s(,
Xorl			,	%rax,8)	1	0/
.p2align 4,,10 .p2align 3 .L4:  movl (%rax), %edx addq \$8, %rax leal (%rdi, %rdx,2), %edx  addl %edx, %esi cmpq %rax, %r8 jne .L4 movl \$s+4, %eax xorl %ecx, %ecx .p2align 4,,10 .p2align 3 .L5:  movl (%rax), %edx xorl %r15d, %r15d movl \$s+4, %eax xorl .p2align 4,,10 .p2align 3 .L5:  movl (%rax), %edx addq \$8, %rax leal (%rdx, xorl %r15d, %edx xorl .p2align 3 .L3:  movl %r15d, %edx movl \$s, %eax xorl %r13d, %r13d xorl %r13d, %r13d xorl %r13d, %r13d xorl %r12d, %r12d xorl %ebp, %ebp xorl %ebp, %ebp xorl %ebp, %ebp xorl %ebp, %ebp xorl %r1d, %r11d xorl %esi, %ecx xorl %r1d, %r11d xorl %ebp, %ebp xorl %r1dd, %r1dd xorl %red, %red xorl %red, %r				0/201/ 0)	IIIONT	%eax, s+4(,
.L4:  movl (%rax), %edx addq \$8, %rax leal (%rdi, worl %r15d, %r15d movl call clock_gettime movl ss+4, %eax xorl yealign 3 .L3:  movl (%rax), %edx addl %edx, %esi cmpq %rax, %r8 jne .L4 call clock_gettime movl \$s+4, %eax xorl %ecx, %ecx .p2align 4,,10 .p2align 3 .L3:  .L5:  movl (%rax), %edx addq \$8, %rax leal (%rdx, worl %r13d, %r13d			%esi, %esi	%rax,8)		φ4 0/··
Mov1					•	•
movl (%rax), %edx addq \$8, %rax leal (%rdi, xorl %edi, %edi %rot), %edx addl %edx, %esi cmpq %rax, %r8 jne .L4 call clock_gettime .p2align 4,,10 .p2align 3 .L5:  movl (%rax), %edx addq \$8, %rax leal (%rdx, xorl %r13d, %r13d %r13d %r13d %rdx,2), %edx subl addl %edx, %ecx cmpq \$s+40004, xorl %r12d, %r12d leal (%rdx, xorl %r13d, %r13d %r13d %rdx,2), %edx subl %edi, %edx xorl %r13d, %r13d addl %edx, %ecx cmpq \$s+40004, xorl %r10d, %r10d %r10d xorl %r8d, %r8d xor	14:	.μ∠α⊥⊥ΥΠ 3				•
addq \$8, %rax leal (%rdi, xorl %edi, %edi xorl %r15d, %r15d movl \$s+40000, cmpq %rax, %r8 jne .L4 soll sp2align 3 .L3:  .L5:  movl (%rax), %edx movl \$s, %eax sorl p2align 3 .L3:  .L5:  movl (%rax), %edx addq \$8, %rax sorl wr13d, %r13d addq \$8, %rax sorl wr13d, %r13d addq \$8, %rax sorl wr13d, %r13d soll wedx, %ecx sorl wr13d, %r13d soll webp, %ebp worl wr1d, %r11d soll webx, %ebx sorl wr1d, %r11d soll webx, %ebx sorl wr1d, %r11d soll webx, %ebx sorl wr1d, %r10d, %r10d soll wr1d, %r1dd soll webx, %esi sorl wr1d, %r1d wr1d, %r1d, %r1d	. L4:	mov1	(%ray) %ad:		-	
leal				%rci	<b>Tead</b>	±σ(%iSp),
%rdx,2), %edx       addl       %edx, %esi       movl       \$s+40000,         cmpq       %rax, %r8       %r14d       call       clock_gettime         jne       .L4       call       clock_gettime         movl       \$s+4, %eax       .p2align 4,,10       .p2align 3         .bp2align 4,,10       .p2align 3       .L3:         .L5:       movl       %r15d, %r15d         movl       .p2align 3       .L3:         .L5:       movl       %r15d, %r15d         movl       .p2align 4,,10       .p2align 3         .L5:       movl       %r15d, %r15d         movl       .p2align 4,,10       .p2align 3         .L5:       movl       %r15d, %r15d         movl       .p2align 3       .p2align 3         .L5:       movl       %r15d, %edx         xorl       %r13d, %r13d       xorl       %r12d, %r12d         xorl       %r2d, %r12d       %ebx, %ebx       xorl       %ebx, %ebx         %rdx, %edx       xorl       %r11d, %r11d       %r12d, %r12d         %rdx,2), %edx       xorl       %ebx, %ebx       xorl       %r12d, %r12d         %rdx, %edx       xorl       %r12d, %r12d       %r12d, %r12d		•		\/01 2T	vorl	%adi %adi
addl %edx, %esi cmpq %rax, %r8 jne .L4 call clock_gettime .p2align 4,,10 .p2align 3 .L3:  .p2align 4,,10 .p2align 3 .L3:  .b2align 3 .L3:  .c5:  movl (%rax), %edx addq \$8, %rax addq \$8, %rax leal (%rdx, %edx xorl %r13d, %r13d addq \$8, %rax xorl %r12d, %r12d leal (%rdx, xorl %ebp, %ebp xorl %r14d, %r11d addl %edx, %ecx xorl %ebx, %ebx xorl %r10d, %r11d addl %edx, %ecx xorl %r10d, %r10d cmpq \$s+40004, xorl %r10d, %r10d cmpq \$s+40004, xorl %r2d, %r8d, %r8d yne .L5 cmpl %ecx, %esi cmpl %ecx, %esi cmovl %esi, %ecx movl .p2align 4,,10 movl %ecx, 48(%rsp,%r9,4)  addq \$1, %r9 movl (%rax), %ecx worl .p2align 3 .L4:	%rdy 2\ %adv	TCUT	(/oi ull,			
cmpq	/oιux,∠j, %eux	Ihhe	%edy %esi			
jne			•	%r14d	III O A T	Ψ3: 40000,
movl			•	/VI 174	call	clock mettime
xorl		-				JIJOK_GCTTIIIC
.p2align 4,,10 .p2align 3 .b5:  movl (%rax), %edx addq \$8, %rax leal (%rdx, xorl %r13d, %r13d) %rdx,2), %edx  subl %edi, %edx subl %edx, %ecx cmpq \$s+40004, xorl %r10d, %r10d cmpq \$s+40004, xorl %r9d, %r9d %rax  jne .L5 cmpl %ecx, %esi cmpl %ecx, %esi cmpl %ecx, %esi cmpl %ecx, %esi cmovl movl %esi, %ecx movl %ecx, movl %ecx,  48(%rsp,%r9,4) addq \$1, %r9  .L3:  movl %r15d, %edx xorl %r13d, %r13d xorl %ebp, %ebp %roxpl %rebx xorl %r11d, %r11d xorl %r9d, %r9d xorl %r8d, %r8d xorl %esi, %esi xorl %edi, %edi .p2align 4,,10 .p2align 3  .L4: movl (%rax), %ecx						
.p2align 3 .b2align 3 .b5:  movl (%rax), %edx xorl %r13d, %r13d xorl %r12d, %r12d feal (%rdx, xorl %ebp, %ebp %ebp %ebx xorl %ebx, %ebx xorl %r1d, %r11d addl %edx, %ecx xorl %r10d, %r10d cmpq \$s+40004, xorl %r9d, %r9d %rax  jne .L5 xorl %r8d, %r8d yr8d yrax  jne .L5 xorl %r8d, %r8d yr8d yr8d yr9d %r9d xorl %r8d, %r8d yr8d yr9d %rex xorl %esi, %esi xorl %esi, %esi xorl %edi, %edi cmovl %esi, %ecx yr2align 3 48(%rsp,%r9,4) addq \$1, %r9  movl (%rax), %ecx				.L3:		
.L5:       movl       \$\$, %eax         movl       (%rax), %edx       xorl       %r13d, %r13d         addq       \$\$, %rax       xorl       %r12d, %r12d         beal       (%rdx,       xorl       %ebp, %ebp         %rdx,2), %edx       xorl       %ebx, %ebx         subl       %edi, %edx       xorl       %r1dd, %r11d         addl       %edx, %ecx       xorl       %r9d, %r9d         %rax       xorl       %r8d, %r8d         jne       .L5       xorl       %esi, %esi         cmpl       %ecx, %esi       xorl       %edi, %edi         cmovl       %esi, %ecx       .p2align 4,,10         movl       %exx)       .L4:         addq       \$1, %r9       movl       (%rax), %ecx					movl	%r15d, %edx
movl         (%rax), %edx         xorl         %r13d, %r13d           addq         \$8, %rax         xorl         %r12d, %r12d           leal         (%rdx,         xorl         %ebp, %ebp           %rdx,2), %edx         xorl         %ebx, %ebx           subl         %edi, %edx         xorl         %r11d, %r11d           addl         %edx, %ecx         xorl         %r10d, %r10d           cmpq         \$s+40004,         xorl         %r9d, %r9d           %rax         xorl         %r8d, %r8d           jne         .L5         xorl         %esi, %esi           cmpl         %ecx, %esi         xorl         %edi, %edi           cmovl         %ecx,         .p2align 4,,10         .p2align 3           48(%rsp,%r9,4)         .L4:         movl         (%rax), %ecx	.L5:	i			_	•
addq \$8, %rax   xorl %r12d, %r12d		movl	(%rax), %edx		_	
leal (%rdx, xorl %ebp, %ebp %rdx,2), %edx subl %edi, %edx xorl %r11d, %r11d addl %edx, %ecx xorl %r10d, %r10d cmpq \$s+40004, xorl %r9d, %r9d %rax jne .L5 xorl %r8d, %r8d yr8d yr8d yr9d %ecx, %esi xorl %esi, %esi xorl %edi, %edi cmovl %esi, %ecx movl %ecx,  .p2align 4,,10 p2align 3 48(%rsp,%r9,4)						•
%rdx,2), %edx       xorl       %ebx, %ebx         subl       %edi, %edx       xorl       %r11d, %r11d         addl       %edx, %ecx       xorl       %r10d, %r10d         cmpq       \$s+40004,       xorl       %r9d, %r9d         %rax       xorl       %r8d, %r8d         jne       .L5       xorl       %esi, %esi         cmpl       %ecx, %esi       xorl       %edi, %edi         cmovl       %esi, %ecx       .p2align 4,,10       .p2align 3         48(%rsp,%r9,4)       .L4:       movl       (%rax), %ecx		•				•
subl addl       %edi, %edx wedx, %ecx       xorl       %r11d, %r11d         cmpq       \$s+40004,       xorl       %r9d, %r9d         %rax       yrad, %r8d       xorl       %r8d, %r8d         jne       .L5       xorl       %esi, %esi         cmpl       %ecx, %esi       xorl       %edi, %edi         cmovl       %esi, %ecx       .p2align 4,,10         movl       %ecx,       .b2align 3         48(%rsp,%r9,4)       .L4:       movl       (%rax), %ecx	%rdx,2), %edx		•			
addl cmpq       %edx, %ecx standard       xorl cmpd, %r10d, %r10d         %rax       xorl cmpd, %r9d, %r9d         %rax       xorl cmpd, %r8d, %r8d         jne cmpl cmpl cmpl cmpl cmpl cmovl cmovl cmovl cmovl cmovl cmovl movl cmovl cmpl       xorl cmpl cmpl cmpl cmpl cmpl cmpl cmpl cmp		subl	%edi, %edx			· · · · · · · · · · · · · · · · · · ·
%rax       xorl       %r8d, %r8d         jne       .L5       xorl       %esi, %esi         cmpl       %ecx, %esi       xorl       %edi, %edi         cmovl       %ecx,       .p2align 4,,10         movl       %ecx,       .p2align 3         48(%rsp,%r9,4)       .L4:       movl       (%rax), %ecx			•		_	
%rax       xorl       %r8d, %r8d         jne       .L5       xorl       %esi, %esi         cmpl       %ecx, %esi       xorl       %edi, %edi         cmovl       %esi, %ecx       .p2align 4,,10         movl       %ecx,       .p2align 3         48(%rsp,%r9,4)       .L4:       movl       (%rax), %ecx		cmpq	\$s+40004,		xorl	%r9d, %r9d
cmpl %ecx, %esi xorl %edi, %edi cmovl %esi, %ecx p2align 4,,10 p2align 3 48(%rsp,%r9,4) .L4: addq \$1, %r9 movl (%rax), %ecx	%rax				xorl	%r8d, %r8d
cmovl %esi, %ecx .p2align 4,,10 .p2align 3 48(%rsp,%r9,4) .L4: addq \$1, %r9 movl (%rax), %ecx		jne	.L5		xorl	%esi, %esi
movl %ecx, .p2align 3 48(%rsp,%r9,4) .L4: addq \$1, %r9 movl (%rax), %ecx		cmpl	%ecx, %esi			%edi, %edi
48(%rsp,%r9,4) .L4: movl (%rax), %ecx		_	%esi, %ecx			
addq \$1, %r9 movl (%rax), %ecx	- 4-	movl	%ecx,		.p2align 3	
	48(%rsp,%r9,4)		<b>.</b>	.L4:	-	<b></b>
cmpq \$40000, %r9 addq \$40, %rax		•				
		ciiibd	\$40000, %r9		auaq	ֆ4⊍, %rax

	jne	.L3			leal	(%rdx,
	leaq	32(%rsp),	%rcx,2),	%ecx		
%rsi	•		. ,,		addl	%ecx, %edi
=	xorl	%edi, %edi			movl	-32(%rax),
	call	700017 70001	%ecx			0=(10. 01.7)
			70ECX		1001	/0/rdv
	clock_gettime	10 (0)	۵,		leal	(%rdx,
	movq	40(%rsp),	%rcx,2),	%ecx		
%rax					addl	%ecx, %r8d
	subq	24(%rsp),			movl	-24(%rax),
%rax			%ecx			
	movl	\$.LC1, %esi			leal	(%rdx,
	pxor	%xmm0, %xmm0	%rcx,2),	%ecx		
	movl	,	' ''		addl	%ecx, %r10d
	s+39992(%rip),	%ecx			movl	-16(%rax),
	movl	20048(%rsp),	%ecx		MOVI	10 (701 ax),
0/ody	IIIOAT	20040(%iSp),	//ecx		1001	/0/rdv
%edx		<b>d</b>	0/	0/	leal	(%rdx,
	movl	\$1, %edi	%rcx,2),	%ecx		
	cvtsi2sdq	%rax, %xmm0			addl	%ecx, %ebx
	movq	32(%rsp),			movl	-8(%rax),
%rax			%ecx			
	subq	16(%rsp),			leal	(%rdx,
%rax	•	, ,	%rcx,2),	%ecx		•
	movapd	%xmm0, %xmm1			addl	%ecx, %r12d
	pxor	%xmm0, %xmm0			movl	-36(%rax),
		.LCO(%rip),	%00%		III O V I	ου ( /01 αλ ) ,
0/14	divsd	.LC0(%rip),	%ecx		11	(0/
%xmm1		0/	04:	07-	leal	(%rcx,
	cvtsi2sdq	%rax, %xmm0	%rcx,2),	%ecx		
	xorl	%eax, %eax			subl	%edx, %ecx
	addsd	%xmm1, %xmm0			addl	%ecx, %esi
	movsd	%xmm0,			movl	-28(%rax),
8(%rsp)		,	%ecx			` ''
(   )	call	printf_chk			leal	(%rcx,
	movsd	8(%rsp),	%rcx,2),	%ACV	1001	(701 077)
0/2/mm0	iiiovsu	σ(701 Sp),	701 CX, Z),	/0CCX	ouh1	0/adv 0/aav
%×mm0	1	Φ LCO 0/			subl	%edx, %ecx
	movl	\$.LC2, %esi			addl	%ecx, %r9d
	movl	\$1, %edi			movl	-20(%rax),
	movl	<b>\$1,</b> %eax	%ecx			
	call	printf_chk			leal	(%rcx,
	xorl	%eax, %eax	%rcx,2),	%ecx		
	movq				subl	%edx, %ecx
		%rsi			addl	%ecx, %r11d
	xorq	%fs:40, %rsi			movl	-12(%rax),
	jne	.L15	%ecx			12 (701 ax))
		\$160072,	70ECX		leal	(9/rov
04	addq	\$100072,	0/	0/	Teal	(%rcx,
%rsp			%rcx,2),	%ecx		o. 1
	.cfi_remember_				subl	%edx, %ecx
	.cfi_def_cfa_o	ffset 8			addl	%ecx, %ebp
	ret				movl	-4(%rax),
.L15:			%ecx			
	.cfi_restore_s	tate			leal	(%rcx,
	call		%rcx,2),	%ecx		, ,
	stack_chk_fa	il			subl	%edx, %ecx
	.cfi endproc				addl	%ecx, %r13d
1.5520.	.ci_enuproc					,
.LFE38:					cmpq	%rax, %r14
	size	main,main			jne	. L4
	.section	.text.unlike			addl	%r8d, %edi
ly					addl	%r9d, %esi
.LCOLDE3:					addl	%edi, %r10d
	.section	.text.startu			addl	%esi, %r11d
р	<del>-</del>				addl	%r10d, %ebx
.LHOTE3:					addl	%r11d, %ebp
	COMM	c 40000 22			addl	%ebx, %r12d
	.comm	s, 40000, 32				•
	.section	.rodata.cst8			addl	%ebp, %r13d
,"aM",@progbit	•				cmpl	%r13d, %r12d
	.align 8				cmovl	%r12d, %r13d
.LC0:					movl	%r13d,
	.long	0	48(%rsp,%	6r15,4	)	
	.long	1104006501		•	addq	\$1, %r15
	.ident	"GCC:			cmpq	\$40000, %r15
					- 17 7	

(Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010"		jne	.L3
.section .note.GNU-		leaq	32(%rsp),
stack,"",@progbits	%rsi		
		xorl	%edi, %edi
		call	clock_gettime
		movq	40(%rsp),
	%rax		.5( 56))
	701 47	subq	24(%rsp),
	%ray	Jund	27(/01 3P),
	%rax	mov1	Ф I C1 0/c = ÷
		movl	\$.LC1, %esi
		pxor	%xmm0, %xmm0
		movl	
		s+39992(%rip),	
		movl	20048(%rsp),
	%edx		
		movl	\$1, %edi
		cvtsi2sdq	%rax, %xmm0
		movq	32(%rsp),
	%rax	•	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
	1	subq	16(%rsp),
	%rax		- (
	701 47	movapd	%xmm0, %xmm1
		pxor	%xmm0, %xmm0
		•	· ·
	0/	divsd	.LCO(%rip),
	%xmm1		0/
		cvtsi2sdq	%rax, %xmm0
		xorl	%eax, %eax
		addsd	%xmm1, %xmm0
		movsd	%xmm0,
	8(%rsp)		
		call	printf_chk
		movsd	8(%rsp),
	%×mm0		` '''
	1	movl	\$.LC2, %esi
		movl	\$1, %edi
		movl	\$1, %eax
		call	
			printf_chk
		xorl	%eax, %eax
		movq	160056(%rsp),
	%rbx		
		xorq	%fs:40, %rbx
		jne	.L13
		addq	\$160072, %rsp
		.cfi_remember_	state
		.cfi_def_cfa_o	ffset 56
		popq	%rbx
		.cfi_def_cfa_o	
		popq	%rbp
		.cfi_def_cfa_o	
		popq	%r12
		.cfi_def_cfa_o	
		popq	%r13
		.cfi_def_cfa_o	
		popq	%r14
		.cfi_def_cfa_o	
		popq	%r15
		.cfi_def_cfa_o	ffset 8
		ret	
	.L13:		
		.cfi_restore_s	tate
		call	
		stack_chk_fa	il
		.cfi_endproc	
	.LFE38:	op. 00	
	1.2.255.	.size	main,main
		.size .section	.text.unlikel
	,	. SECTION	.revr.nuitiket
	У		
	.LCOLDE3:	0004400	+ov+ -+
		.section	.text.startup
•	•		

```
.LHOTE3:
                               s,40000,32
                .comm
                                .rodata.cst8,
                .section
"aM",@progbits,8
                .align 8
.LC0:
                .long
                .long
                               1104006501
               .ident
                               "GCC: (Ubuntu
5.2.1-22ubuntu2) 5.2.1 20151010"
                                .note.GNU-
                .section
stack,"",@progbits
```

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 (*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 opciones de optimización del compilador (-O0, -O2, -O3) y explique las diferencias que se observan en el código justificando las mejoras en velocidad que acarrean. Incorpore los códigos al cuaderno de prácticas y destaque las diferencias entre ellos.
- 2.2. (Ejercicio EXTRA) Para la mejor de las opciones, obtenga los tiempos de ejecución con distintos valores de N y determine para su sistema los valores de Rmax (valor máximo del número de operaciones en coma flotante por unidad de tiempo), Nmax (valor de N para el que se consigue Rmax), y N1/2 (valor de N para el que se obtiene Rmax/2). Estime el valor de la velocidad pico (Rpico) del procesador (consulte en [4] el número de ciclos por instrucción punto flotante para la familia y modelo de procesador que está utilizando) y compárela con el valor obtenido para Rmax. -Consulte la Lección 3 del Tema 1.

# CÓDIGO FUENTE: daxpy.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
int main(int argc,char **argv) {
   unsigned int N,i,j,k, 1;
   k=999:
  struct timespec cqt1,cqt2;
  double ncgt; //para tiempo de ejecución
   //Leer argumento de entrada (no de componentes de la matriz)
   if (argc<2) {
      printf("Faltan no componentes del vector\n");
      exit(-1);
  N=atoi(argv[1]);
  int *A, *B, *C;
  A=(int *) malloc(N*sizeof(int));
  B=(int *) malloc(N*sizeof(int));
   C=(int *) malloc(N*sizeof(int));
```

```
//reservamos memoria dinamica.
   for(i=0; i<N; i++) {
      B[i]=20;
      C[i]=30;
   //inicializamosla matriz y el vector
   clock_gettime(CLOCK_REALTIME, &cgt1);
   for (1=0; 1<N; 1++)
      A[1] = k*B[1] + C[1];
   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
   printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
  printf("Primer componente del vector resultado [%d] y el ultimo componente del
vector resultado [%d]\n",A[0],A[N-1]);
   free(B);
   free(C);
   free(A):
   //eliminamos la memoria dinamica.
```

Tiempes eies	-O0	-O2	-O3	
Tiempos ejec.	0.004218060	0.002621203	0.002164585	

#### **CAPTURAS DE PANTALLA:**

COMENTARIOS SOBRE LAS DIFERENCIAS EN ENSAMBLADOR: El código que se saca de la opción -O3 es el más eficiente, pero también es más difícil de entender.

CÓDIGO EN ENSAMBLADOR (ADJUNTAR AL .ZIP): (PONER AQUÍ SÓLO LA ZONA DEL CÓDIGO ENSAMBLADOR DONDE ESTÁ EL CÓDIGO EVALUADO, USE COLORES PARA DESTACAR LAS DIFERENCIAS)

```
daxpy00.s
       .file
              "daxpy.c"
                      .rodata
        .section
       .align 8
.LC0:
        .string "Faltan no componentes del vector"
       .align 8
.LC2:
       .string "Tiempo(seg.):%11.9f\t / Tama\303\261o Vectores:%u\n"
.LC3:
       .string "A[0] = %d // A[N-1] = %d.\n"
       .text
       .globl main
       .type main, @function
main:
.LFB2:
        .cfi_startproc
       pushq %rbp
```

```
.cfi_def_cfa_offset 16
        .cfi_offset 6, -16
movq %rsp, %rbp
        mova
        .cfi_def_cfa_register 6
        addq
                 $-128, %rsp
                %edi, -100(%rbp)
%rsi, -112(%rbp)
        movl
        movq
                %fs:40, %rax
        movq
                %rax, -8(%rbp)
%eax, %eax
$999, -88(%rbp)
$1, -100(%rbp)
        movq
        xorl
        movl
        cmpl
        jg
                 . L2
        movl
                 $.LCO, %edi
        call
                 puts
                 $-1, %edi
        movl
        call
                 exit
.L2:
        movq
                 -112(%rbp), %rax
        addq
                 $8, %rax
                 (%rax), %rax
        movq
        movq
                 %rax, %rdi
        call
                 atoi
        mov1
                 %eax, -84(%rbp)
                 -84(%rbp), %eax
        movl
        salq
                 $2, %rax
        movq
                 %rax, %rdi
                 malloc
        call
                 %rax, -80(%rbp)
        movq
        movl
                 -84(%rbp), %eax
                 $2, %rax
        salq
        movq
                 %rax, %rdi
        call
                 malloc
        movq
                 %rax, -72(%rbp)
        movĺ
                 -84(%rbp), %eax
        salq
                 $2, %rax
                 %rax, %rdi
        movq
        call
                 malloc
                 %rax, -64(%rbp)
        movq
                 $0, -96(%rbp)
        movi
                 .L3
        jmp
.L4:
        movl
                 -96(%rbp), %eax
        leaq
                 0(,%rax,4), %rdx
                 -72(%rbp), %rax
        movq
                 %rdx, %rax
$20, (%rax)
        addq
        movĺ
                 -96(%rbp), %eax
        movl
                 0(,%rax,4), %rdx
-64(%rbp), %rax
        leaq
        movq
        addq
                 %rdx, %rax
                $30, (%rax)
$1, -96(%rbp)
        movl
        addl
.L3:
        movl
                 -96(%rbp), %eax
                 -84(%rbp), %eax
        cmpl
        ib
                 .L4
                 -48(%rbp), %rax
        leaq
        movq
                 %rax, %rsi
        movl
                 $0, %edi
                 clock_gettime
        call
                 $0, -92(%rbp)
.L5
        movl
        jmp
.L6:
        movl
                 -92(%rbp), %eax
        leaq
                 0(,%rax,4), %rdx
        movq
                 -80(%rbp), %rax
        addq
                 %rdx, %rax
                 -92(%rbp), %edx
        movi
                 0(,%rdx,4), %rcx
        leaq
        movq
                 -72(%rbp), %rdx
        addq
                 %rcx, %rdx
                 (%rdx), %edx
        movl
                 -88(%rbp), %edx
-92(%rbp), %ecx
        imull
        movl
        leaq
                 0(,%rcx,4), %rsi
```

```
movq
                -64(%rbp), %rcx
       addq
               %rsi, %rcx
       movi
               (%rcx), %ecx
       addl
               %ecx, %edx
       movl
               %edx, (%rax)
       addl
               $1, -92(%rbp)
.L5:
       movl
                -92(%rbp), %eax
       cmpl
                -84(%rbp), %eax
       jb
               .L6
       leaq
                -32(%rbp), %rax
               %rax, %rsi
       movq
       movl
               $0, %edi
       call
               clock_gettime
               -32(%rbp), %rdx
       movq
               -48(%rbp), %rax
       movq
       subq
               %rax, %rdx
       movq
               %rdx, %rax
               %xmm1, %xmm1
       pxor
                       %rax, %xmm1
       cvtsi2sdq
               -24(%rbp), %rdx
-40(%rbp), %rax
       movq
       movq
       subq
               %rax, %rdx
       movq
               %rdx, %rax
               %xmm0, %xmm0
       pxor
       cvtsi2sdq
                       %rax, %xmm0
       movsd
               .LC1(%rip), %xmm2
               %xmm2, %xmm0
       divsd
       addsd
               %xmm1, %xmm0
       movsd
               %xmm0, -56(%rbp)
       movl
               -84(%rbp), %edx
               -56(%rbp), %rax
       movq
       mov1
               %edx, %esi
       movq
               %rax, -120(%rbp)
               -120(%rbp), %xmm0
$.LC2, %edi
       movsd
       movl
               $1, %eax
       movl
       call
               printf
       movl
                -84(%rbp), %eax
               $1, %eax
       subl
       movl
               %eax, %eax
       leaq
               0(,%rax,4), %rdx
       movq
                -80(%rbp), %rax
       addq
               %rdx, %rax
       mov1
                (%rax), %edx
       movq
                -80(%rbp), %rax
       movĺ
                (%rax), %eax
               %eax, %esi
$.LC3, %edi
       movl
       movl
               $0, %eax
       movl
       call
               printf
                .
-72(%rbp), %rax
       movq
       movq
               %rax, %rdi
       call
               free
       movq
                -64(%rbp), %rax
       movq
               %rax, %rdi
       call
               free
                -80(%rbp), %rax
       movq
       movq
               %rax, %rdi
       call
               free
               $0, %eax
-8(%rbp), %rsi
       mov1
       movq
       xorq
               %fs:40, %rsi
       jе
                .L8
       call
                __stack_chk_fail
.L8:
       leave
       .cfi_def_cfa 7, 8
       ret
        .cfi_endproc
.LFE2:
        .size main, .-main
        .section
                        .rodata
        .align 8
.LC1:
        .long
```

```
.long 1104006501
.ident "GCC: (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010"
.section .note.GNU-stack,"",@progbits
```

```
daxpy02.s
       .file
              "daxpy.c"
       .section
                      .rodata.str1.8, "aMS", @progbits, 1
       .align 8
.LC0:
       .string "Faltan no componentes del vector"
       .align 8
.LC2:
       .LC3:
       .string "A[0] = %d // A[N-1] = %d.\n"
                      .text.unlikely, "ax", @progbits
       .section
.LCOLDB4:
       .section
                      .text.startup, "ax", @progbits
.LHOTB4:
       .p2align 4,,15
       .globl main
       .type main, @function
main:
.LFB38:
       .cfi_startproc
       pushq %r14
       .cfi_def_cfa_offset 16
       .cfi_offset 14, -16
       pushq %r13
       .cfi_def_cfa_offset 24
       .cfi_offset 13, -24
       pushq %r12
       .cfi_def_cfa_offset 32
       .cfi_offset 12, -32
       pushq %rbp
       .cfi_def_cfa_offset 40
       .cfi_offset 6, -40
       pushq %rbx
       .cfi_def_cfa_offset 48
       .cfi_offset 3, -48
       subq
              $48, %rsp
       .cfi_def_cfa_offset 96
              %fs:40, %rax
       movq
       movq
              %rax, 40(%rsp)
%eax, %eax
       xorl
       cmpl
               $1, %edi
               .L17
       jle
       movq
              8(%rsi), %rdi
       movl
              $10, %edx
       xorl
              %esi, %esi
       call
              strtol
              %eax, %r12d
%rax, %rbp
       mov1
       mova
       movl
              %eax, %r13d
              $2, %r12
%r12, %rdi
       salq
       movq
       call
              malloc
       movq
              %r12, %rdi
              %rax, %r14
       movq
       call
              malloc
              %r12, %rdi
       movq
       movq
              %rax, %rbx
       call
               malloc
              %ebp, %ebp
       testl
              %rax, %r12
       movq
       jе
               .L3
       xorl
              %eax, %eax
       p2align 4,,10
       .p2align 3
.L4:
               $20, (%rbx, %rax, 4)
       movl
              $30, (%r12, %rax, 4)
       movl
              $1, %rax
       addq
```

```
cmpl
                %eax, %r13d
                 .L4
        ja
                %rsp, %rsi
%edi, %edi
        movq
        xorl
        call
                clock_gettime
        xorl %eax, %eax .p2align 4,,10
        .p2align 3
.L6:
        imull
                $999, (%rbx,%rax,4), %edx
        addl
                 (%r12, %rax, 4), %edx
                %edx, (%r14,%rax,4)
        mov1
        addq
                $1, %rax
        cmp1
                %eax, %r13d
                 . L6
        ja
.L7:
                16(%rsp), %rsi
        leaq
        xorl
                %edi, %edi
                clock_gettime
        call
                24(%rsp), %rax
8(%rsp), %rax
        movq
        subq
        movl
                %ebp, %edx
                %xmm0, %xmm0
        pxor
                $.LC2, %esi
%xmm1, %xmm1
        movl
        pxor
        movl
                $1, %edi
        cvtsi2sdq
                        %rax, %xmm0
                16(%rsp), %rax
        movq
        subq
                 (%rsp), %rax
        cvtsi2sdq
                         %rax, %xmm1
        movl
                $1, %eax
                .LC1(%rip), %xmm0
%xmm1, %xmm0
        divsd
        addsd
        call
                  _printf_chk
                -1(%rbp), %eax
(%r14), %edx
$.LC3, %esi
        leal
        movl
        movl
        movl
                $1, %edi
                (%r14,%rax,4), %ecx
%eax, %eax
        movl
        xorl
        call
                  _printf_chk
        movq
                %rbx, %rdi
        call
                free
                %r12, %rdi
        movq
        call
                free
        movq
                %r14, %rdi
        call
                free
        xorl
                %eax, %eax
                40(%rsp), %rcx
        mova
        xorq
                %fs:40, %rcx
        jne
                 .L18
        addq
                $48, %rsp
        .cfi_remember_state
        .cfi_def_cfa_offset 48
        popq
                %rbx
        .cfi_def_cfa_offset 40
                %rbp
        popq
        .cfi_def_cfa_offset 32
        popq
                %r12
        .cfi_def_cfa_offset 24
        popq
                %r13
        .cfi_def_cfa_offset 16
        popq
                %r14
        .cfi_def_cfa_offset 8
        ret
.L3:
        .cfi_restore_state
        movq
                %rsp, %rsi
        xorl
                %edi, %edi
        call
                {\tt clock\_gettime}
        jmp
                 .L7
.L18:
        call
                 __stack_chk_fail
.L17:
        movl
                $.LCO, %edi
        call
                puts
```

```
orl
               $-1, %edi
       call
               exit
       .cfi endproc
.LFE38:
       .size main, .-main
       .section .text.unlikely
.LCOLDE4:
       .section
                     .text.startup
.LHOTE4:
       .section
                      .rodata.cst8, "aM", @progbits, 8
       .align 8
.LC1:
       .long 0
       .long 1104006501
.ident "GCC: (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010"
                       .note.GNU-stack,"",@progbits
       .section
```

```
daxpy03.s
               .file
                               "daxpy.c"
               .section
                               .rodata.str1.8, "aMS", @progbits, 1
               .align 8
.LC0:
               .string
                               "Faltan no componentes del vector"
               .align 8
.LC4:
                               "Tiempo(seg.):%11.9f\t / Tama\303\2610 Vectores:%u\n"
               .string
                               .rodata.str1.1, "aMS", @progbits, 1
               .section
.LC5:
               .string
                               "A[0] = %d // A[N-1] = %d.\n"
               .section
                               .text.unlikely, "ax", @progbits
.LCOLDB6:
               .section
                              .text.startup, "ax", @progbits
.LHOTB6:
               .p2align 4,,15
                        main
               .globl
               .type
                              main, @function
main:
.LFB38:
               .cfi_startproc
                             %r15
               pushq
               .cfi_def_cfa_offset 16
               .cfi_offset 15, -16
               pushq
                              %r14
               .cfi_def_cfa_offset 24
               .cfi_offset 14, -24
                              %r13
               .cfi_def_cfa_offset 32
               .cfi_offset 13, -32
               pushq
                              %r12
               .cfi_def_cfa_offset 40
               .cfi_offset 12, -40
               pushq
                              %rbp
               .cfi_def_cfa_offset 48
               .cfi_offset 6, -48
                              %rbx
               .cfi_def_cfa_offset 56
               .cfi_offset 3, -56
               subq
                              $56, %rsp
               .cfi_def_cfa_offset 112
               movq
                              %fs:40, %rax
                              %rax, 40(%rsp)
%eax, %eax
               movq
               xorl
               cmpl
                              $1, %edi
                              .L48
               jle
                              8(%rsi), %rdi
               movq
                              $10, %edx
               movl
                              %esi, %esi
               xorl
               call
                              strtol
```

```
movl
                              %eax, %ebp
               movq
                              %rax, %rbx
               salq
                              $2, %rbp
                              %rbp, %rdi
               movq
               call
                              malloc
                              %rbp, %rdi
               movq
                              %rax, %r14
               movq
               call
                              malloc
               movq
                              %rbp, %rdi
%rax, %r12
               movq
                              malloc
               call
               testl
                              %ebx, %ebx
                              %rax, %r13
               movq
               jе
                              .L3
                              %r12, %rbp
               movq
               movl
                              %ebx, %edx
                              $15, %ebp
               andl
               shrq
                              $2, %rbp
               negq
                              %rbp
                              $3, %ebp
               andl
               cmpl
                              %ebx, %ebp
               cmova
                              %ebx, %ebp
                              $5, %ebx
               cmpl
                              .L49
               ja
.L4:
               cmpl
                              $1, %edx
                              $20, (%r12)
               mov1
               movl
                              $30, 0(%r13)
               jе
                              .L26
               cmpl
                              $2, %edx
               movl
                              $20, 4(%r12)
                              $30, 4(%r13)
               movl
               jе
                              .L27
               cmpl
                              $3, %edx
               movl
                              $20, 8(%r12)
               movl
                              $30, 8(%r13)
               jе
                              .L28
                              $5, %edx
               cmpl
               movl
                              $20, 12(%r12)
               movl
                              $30, 12(%r13)
                              .L29
               ine
                              $20, 16(%r12)
               movl
               movl
                              $30, 16(%r13)
               movl
                              $5, %eax
.L6:
               cmpl
                              %edx, %ebx
                              .L50
               jе
.L5:
               movl
                              %ebx, %edi
                              -1(%rbx), %r15d
               leal
               movl
                              %edx, %ecx
                              %edx, %edi
               subl
                              -4(%rdi), %esi
               leal
                              %r15d, %r11d
               movl
                              %edx, %r11d
               subl
                              $2, %esi
               shrl
                              $1, %esi
               addl
               cmpl
                              $2, %r11d
               leal
                              0(,%rsi,4), %r8d
               jbe
                              .L8
                              $2, %rcx
               salq
               movdqa
                              .LC1(%rip), %xmm1
                              (%r12,%rcx), %r10
               leaq
                              %edx, %edx
%r13, %rcx
               xorl
               addq
                              .LC2(%rip), %xmm0
               movdqa
               xorl
                              %r9d, %r9d
.L9:
               addl
                              $1, %r9d
```

```
%xmm1, (%r10,%rdx)
               movaps
                              %xmm0, (%rcx,%rdx)
               movups
               addq
                              $16, %rdx
                              %esi, %r9d
               cmpl
               jb
                              .L9
                              %r8d, %eax
               addl
                              %r8d, %edi
               cmpl
               jе
                              .L7
.L8:
               movl
                              %eax, %edx
                              $20, (%r12,%rdx,4)
               movl
               movl
                              $30, 0(%r13,%rdx,4)
               leal
                              1(%rax), %edx
               cmpl
                              %edx, %ebx
               jbe
                              .L7
                              $2, %eax
               addl
               movl
                              $20, (%r12,%rdx,4)
               movl
                              $30, 0(%r13,%rdx,4)
               cmpl
                              %eax, %ebx
               jbe
                              .L7
               movl
                              $20, (%r12, %rax, 4)
               movl
                              $30, 0(%r13,%rax,4)
.L7:
                              %edi, %edi
               xorl
               movq
                              %rsp, %rsi
               call
                              clock_gettime
                              $4, %ebx
               cmpl
               ja
                              .L51
                              %ebx, %ebp
               movl
.L17:
              imull
                              $999, (%r12), %eax
               addl
                              0(%r13), %eax
               cmpl
                              $1, %ebp
               movl
                              %eax, (%r14)
               jе
                              .L32
               imull
                              $999, 4(%r12), %eax
               addl
                              4(%r13), %eax
                              $2, %ebp
               cmpl
                              %eax, 4(%r14)
               movl
               jе
                              .L33
               imull
                              $999, 8(%r12), %eax
               addl
                              8(%r13), %eax
                              $4, %ebp
               cmpl
               movl
                              %eax, 8(%r14)
               jne
                              .L34
               imull
                              $999, 12(%r12), %eax
                              12(%r13), %eax
               addl
               movl
                              %eax, 12(%r14)
               movl
                              $4, %eax
.L19:
               cmpl
                              %ebp, %ebx
                              .L22
               jе
.L18:
                              %ebx, %r8d
               movl
                              %r15d, %esi
               movl
                              %ebp, %edx
               movl
               subl
                              %ebp, %r8d
               subl
                              %ebp, %esi
               leal
                              -4(%r8), %ecx
               shrl
                              $2, %ecx
                              $1, %ecx
               addl
                              $2, %esi
               cmpl
                              0(,%rcx,4), %r9d
               leal
               ibe
                              .L21
               salq
                              $2, %rdx
               xorl
                              %edi, %edi
               leaq
                              (%r12,%rdx), %r11
              leaq
                              0(%r13,%rdx), %r10
               leaq
                              (%r14,%rdx), %rsi
```

```
xorl
                              %edx, %edx
.L13:
              movdqa
                              (%r11,%rdx), %xmm1
              addl
                             $1, %edi
              movdqa
                             %xmm1, %xmm2
                             $4, %xmm2
              pslld
                             %xmm2, %xmm0
              movdqa
                             $3, %xmm0
              pslld
              psubd
                             %xmm2, %xmm0
              psubd
                             %xmm1, %xmm0
                             %xmm0, %xmm1
              movdqa
              pslld
                             $3, %xmm1
              paddd
                             %xmm1, %xmm0
                              (%r10,%rdx), %xmm1
              movdqu
              paddd
                             %xmm1, %xmm0
              movups
                             %xmm0, (%rsi,%rdx)
                             $16, %rdx
              addq
              cmpl
                             %edi, %ecx
               ja
                              .L13
                             %r9d, %eax
              addl
              cmpl
                             %r9d, %r8d
              jе
                              .L22
.L21:
                             %eax, %edx
              movl
              imull
                             $999, (%r12,%rdx,4), %ecx
              addl
                             0(%r13,%rdx,4), %ecx
              movl
                             %ecx, (%r14,%rdx,4)
              leal
                             1(%rax), %edx
                             %edx, %ebx
              cmpl
              jbe
                             .L22
              imull
                             $999, (%r12,%rdx,4), %ecx
              addl
                             $2, %eax
              addl
                             0(%r13,%rdx,4), %ecx
              cmpl
                             %eax, %ebx
              movl
                             %ecx, (%r14,%rdx,4)
              jbe
                              .L22
               imull
                              $999, (%r12,%rax,4), %edx
                             0(%r13,%rax,4), %edx
              addl
                             %edx, (%r14,%rax,4)
              movl
.L22:
              leaq
                             16(%rsp), %rsi
                             %edi, %edi
              xorl
                             clock_gettime
              call
              movq
                             24(%rsp), %rax
              subq
                             8(%rsp), %rax
              movl
                             %ebx, %edx
                             %xmm0, %xmm0
              pxor
                             $.LC4, %esi
              movl
              pxor
                             %xmm1, %xmm1
                             $1, %edi
              movl
              cvtsi2sdq
                             %rax, %xmm0
              movq
                             16(%rsp), %rax
                             (%rsp), %rax
              subq
              cvtsi2sdq
                             %rax, %xmm1
              movl
                             $1, %eax
              divsd
                              .LC3(%rip), %xmm0
              addsd
                             %xmm1, %xmm0
              call
                              __printf_chk
                              (%r14,%r15,4), %ecx
              movl
              movl
                              (%r14), %edx
                             $.LC5, %esi
              movl
                             $1, %edi
              movl
                             %eax, %eax
              xorl
              call
                              __printf_chk
              movq
                              %r12, %rdi
                             free
              call
              movq
                              %r13, %rdi
                              free
              call
                             %r14, %rdi
              movq
```

```
call
                               free
               xorl
                               %eax, %eax
               movq
                               40(%rsp), %rbx
                               %fs:40, %rbx
               xorq
               jne
                               .L52
                               $56, %rsp
               addq
               .cfi_remember_state
               .cfi_def_cfa_offset 56
                               %rbx
               popq
               .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
               .cfi_def_cfa_offset 8
               ret
.L50:
               .cfi_restore_state
                               -1(%rbx), %r15d
               leal
               jmp
                               .L7
.L51:
               xorl
                               %eax, %eax
               testl
                               %ebp, %ebp
                               .L18
               jе
                               .L17
               jmp
.L49:
                               %edx, %edx
               xorl
               xorl
                               %eax, %eax
                               %ebp, %ebp
               testl
               jе
                               .L5
               movl
                               %ebp, %edx
               jmp
                               .L4
.L34:
               movl
                               $3, %eax
               jmp
                               .L19
.L32:
                               $1, %eax
               movl
                               .L19
               jmp
.L33:
               movl
                               $2, %eax
               jmp
                               .L19
.L28:
                               $3, %eax
               movl
               jmp
                               .L6
.L29:
               movl
                               $4, %eax
                               .L6
               jmp
.L26:
               movl
                               $1, %eax
                               .L6
               jmp
.L27:
               movl
                               $2, %eax
               jmp
                               .L6
.L3:
               movq
                               %rsp, %rsi
                               %edi, %edi
               xorl
               movl
                               $4294967295, %r15d
                               clock_gettime
               call
               jmp
                               .L22
.L48:
               movl
                               $.LC0, %edi
               call
                               puts
                               $-1, %edi
               orl
               call
                               exit
```

```
.L52:
                call
                                __stack_chk_fail
                .cfi_endproc
.LFE38:
                                main, .-main
                .size
                .section
                                .text.unlikely
.LCOLDE6:
                .section
                                .text.startup
.LHOTE6:
                .section
                                .rodata.cst16, "aM", @progbits, 16
                .align 16
.LC1:
                                20
                .long
                .long
                                20
                .long
                                20
                .long
                                20
                .align 16
.LC2:
                                30
                .long
                .long
                                30
                .long
                                30
                .long
                                30
                .section
                                .rodata.cst8,"aM",@progbits,8
                .align 8
.LC3:
                .long
                .long
                                1104006501
                                "GCC: (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010" .note.GNU-stack,"",@progbits
                .ident
                .section
```