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): Grupo de prácticas:

Fecha de entrega:

Fecha evaluación en clase:

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz

Sistema operativo utilizado: Ubuntu 18.04 x64

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

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

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

Figura 1 . Código C++ que suma dos vectores

A) MULTIPLICACIÓN DE MATRICES:

```
CAPTURA CÓDIGO FUENTE: pmm-secuencial.c
 #include <stdio.h>
#include <stdlib.h>
 #include <string.h>
 #include <time.h>
 void printMatriz(int n, int **m) {
  int i, j;
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
       printf("%d ", m[i][j]);
    printf("\n");
}</pre>
 }
 int main(int argc, char const *argv[]) {
  if (argc < 2) {
    fprintf(stderr, "ERROR: falta numero de filas y columnas\n");</pre>
          exit(1);
     unsigned n, i, j, k;
n = strtol(argv[1], NULL, 10);
     int **a, **b, **c;
a = (int **)malloc(n * sizeof(int *));
b = (int **)malloc(n * sizeof(int *));
c = (int **)malloc(n * sizeof(int *));
for (i = 0; i < n; i++) {
    a[i] = (int *)malloc(n * sizeof(int))
    b[i] = (int *)malloc(n * sizeof(int))
    c[i] = (int *)malloc(n * sizeof(int))
}</pre>
     // Inicialization
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++) {
    a[i][j] = 0;
    b[i][j] = 1;
    c[i][j] = 2;
     struct timespec cgt1, cgt2;
double ncgt;
      // Multiplicacion
     clock_gettime(CLOCK_REALTIME, &cgt1);
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++) {
    for (k = 0; k < n; k++) {
      a[i][j] += b[i][k] * c[k][j];
    }
}</pre>
     }
clock_gettime(CLOCK_REALTIME, &cgt2);
     ncgt = (double)(cgt2.tv_sec - cgt1.tv_sec) +
    (double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
     printf( M2:\n, b
printf("M2:\n");
          printMatriz(n, a);
} else
          printf("Tiempo = %11.9f\t Primera = %d\t Ultima=%d\n", ncgt, a[0]
 [0],
                         a[n - 1][n - 1]);
     return 0;
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: Convertir la matriz en un vector para hacer alineamiento de memoria

Modificación b) –**explicación-:** Cambiar los bucles j y k para optimizar el acceso a memoria de la multiplicación

1.1. CÓDIGOS FUENTE MODIFICACIONES

```
a) Captura de pmm-secuencial-modificado_a.c
| #include <stdio.h>
 #include <stdlib.h>
#include <string.h>
 #include <time.h>
 void printMatriz(int n, int *m) {
     int i, j;
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++)
     printf("%d ", m[i + j]);
  printf("\n");</pre>
 }
 unsigned i, j, k;
     clock_gettime(CLOCK_REALTIME, cgt1);
// Multiplicacion
for (i = 0; i < n; i++) {
   for (k = 0; k < n; k++) {
      for (j = 0; j < n; j++) {
        a[(i * n) + j] += b[(i * n) + k] * c[(k * n) + j];
      }
}</pre>
    }
     clock_gettime(CLOCK_REALTIME, cgt2);
 int main(int argc, char const *argv[]) {
  if (argc < 2) {
    fprintf(stderr, "ERROR: falta numero de filas y columnas\n");</pre>
         exit(1);
     unsigned n = strtol(argv[1], NULL, 10);
unsigned i, j;
int *a, *b, *c, *new_a, *new_b, *new_c;
unsigned n_cuadrado = n * n;
a = (int *)malloc(sizeof(int) * n_cuadrado + 63);
new_a = (int *)((long int)a + 63) & ~(63));
b = (int *)malloc(sizeof(int) * n_cuadrado + 63);
new_b = (int *)((long int)b + 63) & ~(63));
c = (int *)malloc(sizeof(int) * n_cuadrado + 63);
new_c = (int *)(((long int)c + 63) & ~(63));
      unsigned n = strtol(argv[1], NULL, 10);
     // Inicialization

for (i = 0; i < n; i++) {

  for (j = 0; j < n; j++) {

    new_a[(i * n) + j] = 0;

    new_b[(i * n) + j] = 2;

    new_c[(i * n) + j] = 1;
         }
      struct timespec cgt1, cgt2;
     double ncgt;
      pmm(new_a, new_b, new_c, n, &cgt1, &cgt2);
     if (n < 15) {
  printf("M1:\n");
  printMatriz(n, new_b);
  printf("M2:\n");
  printMatriz(n, new_c);
  printf("Sol:\n");
  printMatriz(n, new_a);
}</pre>
     } else
```

```
free(a);
free(b);
free(c);

return 0;
}
```

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

b) Captura de pmm-secuencial-modificado b.c

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

void printMatriz(int n, int **m) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d", m[i][j]);
        }
        printf("\n");
    }
}
int main(int argc, char const *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "ERROR: falta numero de filas y columnas\n");
        exit(1);
}
unsigned n, i, j, k;
    n = strtol(argv[1], NULL, 10);
int **a, **b, **c;
    a = (int **)malloc(n * sizeof(int *));
    b = (int **)malloc(n * sizeof(int *));
    c = (int **)malloc(n * sizeof(int *));
    for (i = 0; i < n; i++) {
        a[i] = (int *)malloc(n * sizeof(int));
        b[i] = (int *)malloc(n * sizeof(int));
    }

// Inicializcion

for (i = 0; i < n; i++) {
        a[i] = 0;
        b(i] = 0;
        b(i) = 1;
        c[i] = 0;
        b(i) = 1;
        c[i] = 0;
        c[i] = 0;
        b(i) = 1;
        c[i] = 0;
        c[i] = 0;
        c[i] = 0;
        c[i] = 1;
        c[i] = 2;
}</pre>
```

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

1.1. TIEMPOS:

Tamaño: 1000

Modificación	-02
Sin modificar	9,084460365
Modificación a)	1,053379518
Modificación b)	8,653854064

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Se ve una gran mejora en la modificacion a ya que al combertir la matriz a un vector se requieren muchos menos accesos a memoria que son lo que realmente produce que el tiempo de ejecución aumente.

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

pmm-secuencial.s		pmm-secuencial- modificado_b.s		pmm-secuencial- modificado_c.s	
secuencial.c" "aMS",@progbits,:	.file "pmm- .text .section .rodata.str1.1,	secuencial-modif	.text .section .rodata.str1.1.	secuencial-modifi "aMS",@progbits,1	.text .section .rodata.str1.1,
<pre>@function printMatriz: .LFB52:</pre>	.string "%d" .text .p2align 4,,15 .globl printMatriz .type printMatriz, .cfi startproc testl %edi, %edi jle .l7 leal -1(%rdi), %eax pushq %r14	<pre>@function printMatriz: .LFB52:</pre>	.string "%d " .text .p2align 4,,15 .globl printMatriz .type printMatriz, .cfi startproc testl %edi, %edi jle .l7 leal -1(%rdi), %eax pushq %r15	@function printMatriz: .LFB52:	.string "%d" .text .p2align 4,,15 .globl printMatriz .type printMatriz, .cfi startproc testl %edi, %edi jle .l7 leal -1(%rdi), %eax pushq %r14
fset 16 -16	.cfi_def_cfa_of .cfi_offset 14, pushq %r13	fset 16 -16	.cfi_def_cfa_of .cfi_offset 15, pushq %r14	fset 16 -16	.cfi_def_cfa_of .cfi_offset 14, pushq %r13
fset 24 -24	.cfi_def_cfa_of .cfi_offset 13, pushq %r12	fset 24 -24	.cfi_def_cfa_of .cfi_offset 14, pushq %r13	fset 24 -24	.cfi_def_cfa_of .cfi_offset 13, pushq %r12
fset 32 -32	.cfi_def_cfa_of .cfi_offset 12, pushq %rbp	fset 32 -32	.cfi_def_cfa_of .cfi_offset 13, pushq %r12	fset 32 -32	.cfi_def_cfa_of .cfi_offset 12, pushq %rbp
fset 40 -40	<pre>.cfi_def_cfa_of .cfi_offset 6, leaq .LC0(%rip),</pre>	fset 40 -40	<pre>.cfi_def_cfa_of .cfi_offset 12, leaq .LC0(%rip),</pre>	fset 40 -40	<pre>.cfi_def_cfa_of .cfi_offset 6, leaq .LC0(%rip),</pre>
%rbp fset 48	pushq %rbx .cfi_def_cfa_of	%r13 fset 48	pushq %rbp .cfi_def_cfa_of	%rbp fset 48	pushq %rbx .cfi_def_cfa_of
-48 %r13	<pre>.cfi_offset 3, leaq 8(%rsi,%rax,8),</pre>	-48	.cfi_offset 6, salq \$2, %rax pushq	-48 %r13	<pre>.cfi_offset 3, leaq 8(%rsi,%rax,8),</pre>
%r12	leaq 4(,%rax,4), movq %rsi, %r14 .p2align 4,,10	fset 56 -56	%rbx .cfi_def_cfa_of .cfi_offset 3, leag	%r12	leaq 4(,%rax,4), movq %rsi, %r14 .p2align 4,,10
.L3:	.þ2aliğn 3′′ xorl	%r15	4(%rsi,%rax), leaq	.L3:	.þ2aliğn 3′′ xorl

.L4:	%ebx, %ebx .p2align 4,,10 .p2align 3		4(%rax), %r14 movq %rsi, %r12	.L4:	%ebx, %ebx .p2align 4,,10 .p2align 3
	movq (%r14), %rax movq	fset 64	subq ['] \$8, %rsp .cfi_def_cfa_of		movq (%r14), %rax movq
	%rbp, %rsi movl \$1,_%edi	.L3:	.p2align 4,,10 .p2align 3		%rbp, %rsi movl \$1,_%edi
	movl (%rax,%rbx),		leaq (%r14,%r12),		movl (%rax,%rbx),
%edx	xorl %eax, %eax	%rbp	movq %r12, %rbx	%edx	xorl %eax, %eax
	addq \$4, %rbx call		.p2align 4,,10 .p2align 3		addq \$4, %rbx call
printf_chk@PLT	call	.L4:	movl (%rbx), %edx	printf_chk@PLT	call
printr_cnk@FET	cmpq %rbx, %r12		xorl %eax, %eax	printr_cnk@rer	cmpq %rbx, %r12
	jne .L4 movl		movq %r13, %rsi movl		jne .L4 movl
	\$10, %edi addq \$8, %r14		\$1, %edi addq		\$10, %edi addq \$8, %r14
	call putchar@PLT		\$4, %rbx call		call putchar@PLT
	cmpq %r13, %r14	printf_chk@PLT	cmpq		cmpq %r13, %r14
	jne .L3 popq		%rbp, %rbx jne .L4		jne .L3 popa
fset 40	%rbx .cfi_def_cfa_of		MONT	fset 40	popq %rbx .cfi_def_cfa_of
1361 40	popq %rbp		\$10, %edi addq \$4, %r12 call	1361 40	popq %rbp
fset 32	.cfi_def_cfa_of popq		putchar@PLT cmpq %r15, %r12	fset 32	.cfi_def_cfa_of popq
foot 24	%r12 .cfi_def_cfa_of		jne .L3	foot 24	%r12 .cfi_def_cfa_of
fset 24	popq %r13		addq \$8, %rsp .cfi_def_cfa_of	fset 24	popq %r13
fset 16	.cfi_def_cfa_of	fset 56	popq %rbx	fset 16	.cfi_def_cfa_of
	popq %r14 .cfi_def_cfa_of	fset 48	.cfi_def_cfa_of		popq %r14 .cfi_def_cfa_of
fset 8 .L7:	ret		<pre>popq %rbp .cfi_def_cfa_of</pre>	fset 8 .L7:	ret
	.cfi_restore 3	fset 40	popq %r12		.cfi_restore 3
	.cfi_restore 12 .cfi_restore 13 .cfi_restore 14	fset 32	.cfi_def_cfa_of		.cfi_restore 12 .cfi_restore 13 .cfi_restore 14
.LFE52:	rep ret .cfi_endproc		popq %r13 .cfi_def_cfa_of	.LFE52:	rep ret .cfi_endproc
	.size printMatriz,	fset 24	popq		.size printMatriz,
printMatriz	.section .rodata.str1.8,	fset 16	%r14 .cfi_def_cfa_of	printMatriz 	.section .rodata.str1.8,
"aMS",@progbits,1	.align 8		popq %r15	"aMS",@progbits,1	.align 8
.LC1:	.string "ERROR: falta	fset 8	.cfi_def_cfa_of ret	.LC1:	.string "ERROR: falta
numero de filas y	/ columnas\n" .section	.L7:	.cfi_restore 3	numero de filas y	columnas\n" .section .rodata.str1.1
.LC2:	.rodata.str1.1 .string		.cfi_restore 6 .cfi_restore 12 .cfi_restore 13	.LC2:	.strina
.LC3:	"M1:" string		.cfi_restore 14 .cfi_restore 15 rep_ret	.LC3:	"M1:""
.LC4:	.string "M2:"	.LFE52:	.cfi_endproc	.LC4:	.string "M2:"
	.string "Sol:" .section	printMatriz	.size printMatriz,		.string "Sol:" .section
I Ce.	.rodata.str1.8 .align 8	· -	.p2align 4,,15 .globl	1.06.	.rodata.str1.8 .align 8
.LC6:	.string "Tiempo =		pmm .type pmm, @function	.LC6:	.string "Tiempo =
%11.9f\t Primera %d\n"	= %d\t Ultima= .section	pmm: .LFB53:	.cfi_startproc	%11.9f\t Primera %d\n"	= %d\t Ultima= .section
ax",@progbits	.text.startup,"		pushq %r15	ax",@progbits	.text.startup,"
	.p2align 4,,15 .globl main	fset 16	<pre>.cfi_def_cfa_of .cfi_offset 15,</pre>		.p2align 4,,15 .globl main
main:	.type main, @function	-16	pushq %r14	main:	.type main, @function
.LFB53:	.cfi_startproc	fset 24	.cfi_def_cfa_of	.LFB53:	.cfi_startproc
	pushq %r15 .cfi_def_cfa_of	-24	.cfi_offset 14, movl		pushq %r15 .cfi_def_cfa_of
fset 16	.cfi_offset 15,		%ecx, %r14d pushq	fset 16	.cfi_offset 15,
-16	pushq %r14	fset 32	%r13 .cfi_def_cfa_of	-16	pushq %r14
Q				itoctura v Tocnología	

fset 24	.cfi_def_cfa_of	-32	.cfi_offset 13,	fset 24	.cfi_def_cfa_of
	.cfi_offset 14,	- 32	pushq		.cfi_offset 14,
-24	pushq		%r12 .cfi_def_cfa_of	-24	pushq
	%r13	fset 40			%r13 '
fset 32	.cfi_def_cfa_of	-40	.cfi_offset 12,	fset 32	.cfi_def_cfa_of
-32	.cfi_offset 13,		movq %rsi, %r12	-32	.cfi_offset 13,
32	pushq		pushq	32	pushq
	%r12 .cfi_def_cfa_of		%rbp .cfi_def_cfa_of		%r12 .cfi_def_cfa_of
fset 40		fset 48		fset 40	.cfi offset 12,
-40	.cfi_offset 12,	-48	.cfi_offset 6,	-40	.CII_UIISEL 12,
	pushq %rbp		pushq %rbx		pushq %rbp
foot 40	.cfi_def_cfa_of	foot CC	.cfi_def_cfa_of	foot 10	.cfi_def_cfa_of
fset 48	.cfi_offset 6,	fset 56	.cfi_offset 3,	fset 48	.cfi_offset 6,
-48	pushq	-56	movq	-48	pushq
	%rbx ·		%r8, %rsi		%rbx
fset 56	.cfi_def_cfa_of		movq %rdi, %rbx	fset 56	.cfi_def_cfa_of
- 56	.cfi_offset 3,		xorl	-56	.cfi_offset 3,
-50	subq		%edi, %edi movq	-50	subq
	\$88, %rsp .cfi_def_cfa_of		%rdx, %r13 subq		\$88, %rsp .cfi_def_cfa_of
fset 144			\$8, %rsp .cfi_def_cfa_of	fset 144	
	movq %fs:40, %rax	fset 64	.cri_der_cra_or		movq %fs:40, %rax
	movq		movq %r9, %rbp		movq
	%rax, 72(%rsp) xorl		call		%rax, 72(%rsp) xorl
	%eax, %eax cmpl	clock_gettime@PL			%eax, %eax cmpl
	\$1, %edi	- 5250K_90CCIMC@FL	testl		\$1, %edi
	jle .L34		%r14d, %r14d je		jle .L38
	movq		.L13		movq
	8(%rsi), %rdi movl		movl %r14d, %r10d		8(%rsi), %rdi movl
	\$10, %edx xorl		xorl %r15d, %r15d		\$10, %edx xorl
	%esi, %esi		xorl		%esi, %esi
	call strtol@PLT		%r11d, %r11d .p2align 4,,10		call' strtol@PLT
	movl	.L12:	.p2aliğn 3′′		movl
	%eax, %r14d movq	. L12.	movl		%eax, %r14d movq
	%rax, %r15 movq		%r10d, %r9d movl		%rax, %r15 movq
	%rax, 8(%rsp)		%r15d, %r8d		%rax, (%rsp)
	leaq 0(,%r14,8),		subl %r14d, %r9d		leaq 0(,%r14,8),
%r12	movq		movl ′ %r9d, %ecx	%r12	movq
	%r12, %rdi		.p2align 4,,10		%r12, %rdi
	call ['] malloc@PLT	.L16:	.p2align 3		call malloc@PLT
	movq %r12, %rdi		movl		movq %r12, %rdi
	movq		%ecx, %eax leaq		movq
	%rax, %rbx call	%rdx	(%r12,%rax,4),		%rax, %rbx call
	malloc@PLT	, and an	movl		malloc@PLT
	movq %r12, %rdi		%r9d, %eax .p2align 4,,10		movq %r12, %rdi
	movq %rax, %rbp	.L14:	.p2align 3		movq %rax, %rbp
	call		leal		call
	malloc@PLT testl	%esi	(%rax,%r8),		malloc@PLT testl
	%r15d, %r15d		movl		%r15d, %r15d
	movq %rax, %r12		%eax, %edi addl		movq %rax, %r12
	je .L13		\$1, %eax movl		je .L13
	movq	%oci	0(%r13,%rsi,4),		movl
	%r15, %rax salq	%esi	imull		%r15d, %eax salq
	salq \$2, %r14 xorl		(%rdx), %esi addl		salq \$2, %r14 xorl
	%r15d, %r15d		%esi, (%rbx,		%r15d, %r15d
	subl \$1, %eax	%rdi,4)	cmpl		subl \$1, %eax
	leaq		%eax, %r10d		Leaq
	1(%rax), %r13 movq		jne . <u>L14</u>		1(%rax), %r13 movl
	%rax, 16(%rsp)		addl \$1, %ecx		%eax, 20(%rsp) movq
9/5024	0(,%r13,8),		addl		%rax, 8(%rsp)
%rax	movq		%r14d, %r8d cmpl		leaq 0(,%r13,8),
	%r13, 24(%rsp)		%r10d, %ecx	%rax	movq
	movq %rax, %r13		jne .L16		%r13, 24(%rsp)
	.p2align 4,,10 .p2align 3		addl \$1, %r11d		mova
.L14:	-		addl		%rax, %r13 .p2align 4,,10
	movq %r <u>1</u> 4, %rdi		%r14d, %r10d subl	.L14:	.p2align 3
	call		%r14d, %r15d		movq %r14 %rdi
	malloc@PLT movq		cmpl %r11d, %r14d		%r14, %rdi call
	movq %r14, %rdi movq		jne .L12		malloc@PLT
04.45	%rax, (%rbx,	.L13:			movq %r14, %rdi
%r15)	call		addq \$8, %rsp		movq %rax, (%rbx,

		I			
	malloc@PLT movq %r14, %rdi	fset 56	.cfi_def_cfa_of	%r15)	call
	movq		movq %rbp, %rsi xorl		malloc@PLT movq %r14, %rdi
%r15)	%rax, 0(%rbp,		%edi, %edi		movq
	call malloc@PLT		popq %rbx cfi def cfa of	%r15)	%rax, 0(%rbp, call
(%r12 %r15)	movq %rax,	fset 48	.cfi_def_cfa_of		malloc@PLT
(%r12,%r15)	addq \$8, %r15		popq %rbp	(%r12,%r15)	movq %rax,
	cmpg %r15, %r13	fset 40	.cfi_def_cfa_of	(%112,%113)	addq \$8, %r15
	jne L14		%r12 .cfi_def_cfa_of		cmpq %r15, %r13
	movq 24(%rsp), %r13	fset 32	popq		jne .L14
	xorl %r14d, %r14d		%r13 .cfi_def_cfa_of		movq 24(%rsp), %r13
	leaq 0(,%r13,4), %r8	fset 24	popq		xorl %r14d, %r14d
	.p2align 4,,10 .p2align 3		%r14 .cfi_def_cfa_of		leaq 0(,%r13,4), %r8 .p2align 4,,10
.L16:	movq	fset 16	popq %r15		.p2align 4,,10 .p2align 3
%rsi	(%rbx,%r14),	5	%r15 .cfi_def_cfa_of	.L16:	movq
0/501/	movq 0(%rbp,%r14),	fset 8	jmp	%rsi	(%rbx,%r14),
%rcx	xorl	clock_gettime@PLT		%rcx	movq 0(%rbp,%r14),
	%eax, %eax movq (%r12,%r14),	.LFE53:	.cfi_endproc .size	%i CX	xorl %eax, %eax
%rdx	(%12,%114), .p2align 4,,10		pmm,pmm .section		movq (%r12,%r14),
.L15:	.p2align 3	 "aMS",@progbits,1	.rodata.str1.8,	%rdx	.p2align 4,,10
	movl \$0, (%rsi,%rax)	.LC1:	align 8	.L15:	.p2align 3
	<pre>\$0, (%rsi,%rax) mov1 \$1, (%rcx,%rax)</pre>		.string "ERROR: falta		movl \$0,_(%rsi,%rax)
	\$1, (%rcx,%rax) mov1 \$2, (%rdx,%rax)	numero de filas y	/ columnas\n" .section		mov1 \$1, (%rcx,%rax)
	addq \$4, %rax	.LC2:	.rodata.str1.1		mov⊥ \$2, (%rdx,%rax)
	cmpq %rax, %r8		.string "M1:"		addq \$4, %rax
	jne .L15	.LC3:	.string "M2:"		cmpq %rax, %r8
	addq \$8, %r14	.LC4:			jne .L15
	cmpq %r15, %r14 jne		.string "Sol:" .section		addq \$8, %r14
	.L16 leag		.rodata.str1.8 .align 8		cmpq %r14, %r15 jne
	32(%rsp), %rsi xorl	.LC6:	.string		.L16 leaq
	%edi, %edi call	%11.9f\t Primera	"Tiempo =		32(%rsp), %rsi xorl
clock_gettime@PL ⁻	Т	%d\n"	.section		%edi, %edi call
	movq 16(%rsp), %rax	ax",@progbits	.text.startup,"	clock_gettime@PL	Γ
	xorl %r11d, %r11d		.p2align 4,,15 .globl		movq 8(%rsp), %rax
%r15	leaq 4(,%rax,4),		main .type main, @function		%r11d, %r11d
7%1 15	.p2align 4,,10	main:	main, wrunction	%r15	1eaq 4(,%rax,4),
.L25:	.p2align 3 movq	.LFB54:	.cfi_startproc pushq	%r15	.p2align 4,,10 .p2align 3
%r9	(%rbx,%r11),		%r15 .cfi_def_cfa_of	.L27:	movq
	movq 0(%rbp,%r11),	fset 16	.cfi_offset 15,	 %r9	(%rbx,%r11),
%r8	leaq (%r15,%r9),	-16	pushq %r14		movq 0(%rbp,%r11),
%r10		foot 04	%r14 .cfi_def_cfa_of	%r8 	leaq (%r15,%r9),
	movq %r9, %rsi n2align 4 10	fset 24	.cfi_offset 14,	%r10	
.L22:	.p2align 4,,10 .p2align 3	-24	pushq %r13		movq %r9, %rsi .p2align 4,,10
	movl (%rsi), %ecx	fset 32	.cfi_def_cfa_of	.L22:	.p2align 3
	movq %rsi, %rdi	-32	.cfi_offset 13,		movl (%rsi), %ecx
	xorl´ %eax, %eax	_	pushq %r12		movq %rsi, %rdi
	subq %r9. %rdi	fset 40	.cfi_def_cfa_of		xorl %eax, %eax
110.	.p2align 4,,10 .p2align 3	-40	.cfi_offset 12,		subq %r9, %rdi
.L19:	movq		pushq %rbp	110.	.p2align 4,,10 .p2align 3
%rdx	(%r12,%rax,8),	fset 48	.cfi_def_cfa_of	.L19:	movq (%r12 %ray 8)
%edx	movl (%rdx,%rdi),	-48	.cfi_offset 6, pushq	%rdx	(%r12,%rax,8), movl
/ocux	imull (%r8,%rax,4),		%rbx .cfi_def_cfa_of	%edx	(%rdx,%rdi),
%edx		fset 56	.cfi_offset 3,		imull (%r8,%rax,4),
	addq \$1, %rax addl	-56		%edx	
10	%edx, %ecx		subq \$72, %rsp		addq \$1, %rax

	cmpq %r13, %rax	foot 120	.cfi_def_cfa_of		addl
	movl	fset 128	movq		%edx, %ecx cmpq
	%ecx, (%rsi)		%fs:40, %rax		%r13, %rax
	jne .L19		movq %rax, 56(%rsp)		movl´ %ecx, (%rsi)
	addq \$4, %rsi		xorl		jne
	\$4, %rsi cmpq		%eax, %eax cmpl		.L19
	%rsi, %r10		\$1, %edi		addq \$4, %rsi
	jne .L22		jle .L33		cmpq %rsi, %r10
	adda		movq		jne
	\$8, %r11 cmpq		8(%rsi), %rdi movl		.L22
	%r11, %r14		\$10, %edx xorl		addq \$8, %r11
	jne .L25		xorl		cmpq %r14, %r11
	ieaq		%esi, %esi call		ງne
	48(%rsp), %rsi xorl		strtol@PLT		L27 leaq
	%edi, %edi		movq %rax, %rbp		48(%rsp), %rsi
	call'		movl		xorl
clock_gettime@PL	Т		%eax, (%rsp) imull		%edi, %edi call
	cmpl \$14 8(%ren)		%eax, %eax	 clock_gettime@PL1	г
	\$14, 8(%rsp) jbe		leaq 63(,%rax,4),	CIOCK_gettime@FL	cmpl
	.L18 movq	%r14	movq		\$14, (%rsp)
	16(%rsp), %rdi		%r14, %rdi		jbe .L18
	pxor %xmm0, %xmm0		call malloc@PLT		movq 8(%rsp), %rdi
	pxor		mova		pxor
	%xmm1, %xmm1		%r14, %rdi		%xmm0, %xmm0
	movq (%rbx), %rdx		leaq 63(%rax), %rbx		pxor %xmm1, %xmm1
	leaq '' .LC6(%rip),		movq		movq
%rsi			%rax, %r12 call		(%rbx), %rdx leag
	movq (%rbx,%rdi,8),		malloc@PLT movq	%rsi	.LC6(%rip),
%rax	_		%r14, %rdi	/01 31	movq
	movl (%rdx), %edx		leaq 63(%rax), %r15	%rax	(%rbx,%rdi,8),
	mov1		movq	701 UX	movl
%ecx	(%rax,%rdi,4),		%rax, %r13 call		(%rdx), %edx movl
/ACCX	movq		malloc@PLT		(%rax,%rdi,4),
	56(%rsp), %rax movl		leaq 63(%rax), %r10	%ecx	movq
	\$1, %edi		andq		56(%rsp), %rax
	subq 40(%rsn) %ray		\$-64, %rbx		movl \$1 %edi
	40(%rsp), %rax cvtsi2sdq		andg \$-64, %r15		\$1, %edi subq
	%rax, %xmm0 movq		movq %rax, %r14		40(%rsp), %rax
	48(%rsp), %rax		andq		cvtsi2sdq %rax, %xmm0
	subq 32(%ṛṣp), %rax		\$-64, %r10 test1		movq 48 <u>(</u> %rsp), %rax
	cvts12sdq		%ebp, %ebp		subq
	%rax, %xmm1 movl		je .L22		32(%rsp), %rax cvtsi2sdq
	\$1, %eax		mov1		%rax, %xmm1
	divsd .LC5(%rip),		(%rsp), %esi xorl		movl \$1, %eax
%xmm0			%eax, %eax		divsd
	addsd %xmm1, %xmm0		xorl %r11d, %r11d	%×mm0	.LC5(%rip),
	call		.p2align 4,,10		addsd
printf_chk@PLT		.L24:	.þ2aliğn 3′′		%xmm1, %xmm0 call
.L24:	_		leal	nrintf oblean T	
	xor⊥ %eax, %eax	%ecx	(%rsi,%rax),	printf_chk@PLT .L24:	_
	movq		.p2align 4,,10		movl 20(%rsn) %eav
	72(%rsp), %rbx xorq	.L23:	.p2align 3		20(%rsp), %eax xorl
	xorq %fs:40, %rbx		movl %eax, %edx		%r13d, %r13d leaq
	jne .L35		add⊥	.	8(,%rax,8),
	addq \$88, %rsp		\$1, %eax cmpl	%r14	.p2align 4,,10
	.cfi_remember_s		%ecx, %eax	1.00	.p2align 3
tate	.cfi def cfa of		movl \$0, (%rbx,	.L26:	movq
fset 56		%rdx,4)	_	0/rdi	(%rbx,%r13),
	popq %rbx		movl \$1,	%rdi	call
foot 40	.cfi_def_cfa_of	(%r15,%rdx,4)	· _		free@PLT
fset 48	popq		movl \$2,		movq 0(%rbp,%r13),
	popq %rbp	(%r10,%rdx,4)		%rdi	
fset 40	.cfi_def_cfa_of		jne .L23		call free@PLT
	popq %r12		leal		movq
	%r12 .cfi_def_cfa_of		1(%r11), %edx cmpl	%rdi	(%r12,%r13),
fset 32			%edx, %esi		addq \$8, %r13 call
	popq %r13		je .L34		čali 13
fset 24	.cfi_def_cfa_of		movl %edx, %r11d		free@PLT
1300 24	popq %r14		jmp .L24		cmpq %r14, %r13
	%r14 .cfi_def_cfa_of		.L24 .p2align 4,,10		jne . L26
fset 16		104	.p2align 3	.L25:	
	popq %r15	.L34:	leaq		movq %rbx, %rdi
f 0	.cfi_def_cfa_of		32(%rsp), %r9		call
fset 8	ret		leaq 16(%rsp), %r8		free@PLT movq
.L13:			movq		%rbp, %rdi
			Donto Argi	uitectura v Tecnologí	a do Computadoros

	.cfi_restore_st		%r10, %rdx		call
ate	leaq		movl %ebp, %ecx		free@PLT movq
	32(½rsp), %rsi		mova		%r12, %rdi call
	xorl %edi, %edi		%r15, %rsi movq		call free@PLT
	call		%rbx, %rdi		xorl
clock_gettime@PL	т		movl / %r11d, 12(%rsp)		%eax, %eax movq
CIOCK_gcccimc@rL	leag		movq		72(%rsp), %rbx
	48(%rsp), %rsi xorl		%r10, (%rsp) call		xorq %fs:40, %rbx
	%edi, %edi		pmm		jne
	call		cmpl \$14, %ebp		.L39 addq
clock_gettime@PL	Т		movq		\$88, %rsp
.L18:	lean		(%rsp), %r10 movl	tate	.cfi_remember_s
	leaq .LC2(%rip),		12(%rsp), %r11d		.cfi_def_cfa_of
%rdi	call		jbe .L27	fset 56	popq
	puts@PLT		leal		%rbx
	movq 8(%rsp), %r15	%eax	(%r11,%r11),	fset 48	.cfi_def_cfa_of
	movq	, weak	pxor	1300 40	popq
	%rbp, %rsi movl		%xmm0, %xmm0 pxor		%rbp .cfi_def_cfa_of
	%r15d, %edi		%xmm1, %xmm1	fset 40	
	call printMatriz		movl (%rbx), %edx		popq %r12
	leaq		movl		.cfi_def_cfa_of
%rdi	.LC3(%rip),	%ecx	(%rbx,%rax,4),	fset 32	popq
701 41	call	, , , , , , , , , , , , , , , , , , ,	movq		%r13
	puts@PLT mova		40(%rsp), %rax leaq	fset 24	.cfi_def_cfa_of
	movq %r12, %rsi movl	0/	.LC6(%rip),		popq %r14
	movl %r15d, %edi	%rsi	suba		%r14 .cfi_def_cfa_of
	call		24(½rsp), %rax	fset 16	
	printMatriz leaq		movl \$1, %edi		popq %r15
04 - 4.	.LC4(%rip),		cvtsi2sdq	S	.cfi_def_cfa_of
%rdi	call		%rax, %xmm0 movq	fset 8	ret
	puts@PLT		32(%rsp), %rax	.L13:	
	movq %rbx, %rsi		subq 16(%ṛṣp), %rax	ate	.cfi_restore_st
	movT		cvts12sdq	400	leaq
	%r15d, %edi call		%rax, %xmm1 movl		32(%rsp), %rsi xorl
	printMatriz		\$1, %eax		%edi, %edi
	jmp .L24		divsd .LC5(%rip),		call
.L35:		%xmm0		clock_gettime@PL	Γ
	call		addsd %xmm1, %xmm0		leaq 48(%rsp), %rsi
stack_chk_fail	@PLT		call		xori
.L34:	movq	printf_chk@PLT			%edi, %edi call
0/501	stderr(%rip),	.L26:	ma. / a	alaak gattima@DL	
%rcx	leaq		movq %r12, %rdi	clock_gettime@PL ⁻ .L18:	l
0/rdi	.LC1(%rip),		call		leaq .LC2(%rip),
%rdi	movl		free@PLT movq	%rdi	.LC2(%r1p),
	\$40, %edx		%r <u>1</u> 3, %rdi		call puts@PLT
	movl \$1, %esi		call free@PLT		movq
	call		movg %r14 %rdi		(%rsp), %r15
	fwrite@PLT movl		%r14, %rdi call		movq %rbp, %rsi
	\$1, %edi call		free@PLT		IIIOAT
	exit@PLT		xorl %eax, %eax		%r15d, %edi call
LFE53:	.cfi_endproc		movq		printMatriz leag
	.size		56(%rsp), %rdi xorq	0.1.	leaq .LC3(%rip),
	main,main .section		%fs:40, %rdi jne	%rdi	call
	.rodata.cst8,"a		. L35		puts@PLT
M",@progbits,8	.align 8		addq \$72, %rsp		movq %r12, %rsi
.LC5:	_		.cfi_remember_s		MOAT
	.long 0	tate	.cfi_def_cfa_of		%r15d, %edi call
	.long	fset 56			printMatriz
	1104006501 .ident		popq %rbx		leaq .LC4(%rip),
7.3.0-16ubuntu3)	"GCC: (Ubuntu	foot 40	.cfi_def_cfa_of	%rdi	
7.3.0-16ubuntu3)	7.3.0" .section	fset 48	nona		call puts@PLT
atack "" e · · · · ·	.note.GNU-		popq %rbp		movT
stack,"",@progbi	LS	fset 40	.cfi_def_cfa_of		%r15d, %edi movq
			popq %r12		%rbx, %rsi
			%r12 .cfi_def_cfa_of		call printMatriz
		fset 32			leal
			popq %r13		-1(%r15), %edi testl
		foot 24	.cfi_def_cfa_of		%r15d, %r15d
		fset 24	popq		movl´ %edi, 20(%rsp)
			%r14		ine '` ''
		fset 16	.cfi_def_cfa_of		.L24 jmp_
			popg %r15	120:	. L25
			%r15 .cfi_def_cfa_of	.L39:	call
		fset 8		stack obl. foil	
		.L22:	ret	stack_chk_fail@ .L38:	ÿr∟I
		I.		I .	

```
movq
stderr(%rip),
                                                .cfi_restore_st
                                              leaq
32(%rsp), %r9
leaq
16(%rsp), %r8
movq
%r10, %rdx
xor1
%ecx, %ecx
movq
%r15, %rsi
movq
%rbx, %rdi
movq
%r10, (%rsp)
call
pmm
movq
(%rsp), %r10
ate
                                                                                            %rcx
                                                                                                                                           leaq
.LC1(%rip),
                                                                                            %rdi
                                                                                                                                          movl
$40, %edx
movl
$1, %esi
call
fwrite@PLT
movl
$1, %edi
call
exit@PLT
.cfi_endproc
                                                                                             .LFE53:
                                                                                                                                           .size
main, .-main
.section
.rodata.cst8,"a
.L27:
                                               leag
.LC2(%rip),
                                                                                             M",@progbits,8
%rdi
                                                                                                                                            .align 8
                                              movg
%r10, (%rsp)
call
puts@PLT
movg
%r15, %rsi
movl
%ebp, %edi
call
printMatriz
leag
.LC3(%rip),
                                                                                           .LC5:
.long
0
.long
1104006501
.ident
"GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
section
.note.GNU-
stack,"",@progbits
                                                                                             .LC5:
%rdi
                                               call
puts@PLT
                                               movq
(%rsp), %r10
movl
%ebp, %edi
                                              movq
%r10, %rsi
call
printMatriz
leaq
.LC4(%rip),
%rdi
                                              call
puts@PLT
movq
%rbx, %rsi
movl
%ebp, %edi
call
printMatriz
jmp
.L26
.L35:
                                               call
__stack_chk_fail@PLT .L33:
                                               movq
stderr(%rip),
%rcx
                                               leaq
.LC1(%rip),
%rdi
                                              movl
$40, %edx
movl
$1, %esi
call
fwrite@PLT
movl
$1, %edi
call
exit@PLT
.cfi_endproc
.LFE54:
                                               .size
main, .-main
.section
                                               .rodata.cst8,"a
M",@progbits,8
                                               .align 8
.LC5:
                                               .long
.long
.long
.long
.lid4006501
.ident
"GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
.section
.note.GNU-
stack,"",@progbits
```

B) CÓDIGO FIGURA 1:

CAPTURA CÓDIGO FUENTE: figura1-original.c

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

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

int main() {
    int ii, i, X1, X2;
    int R[40000];

    struct timespec cgt1, cgt2;
    double ncgt;

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

    printf("R[0] = %i, R[39999] = %i\n", R[0], R[39999]);

    return 0;
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: Se unen los bucles internos para realizar la mitad de iteraciones a pesar de hacer más calculos.

Modificación b) –explicación-: Eliminar uno de los bucles for, utilizando uno solo.

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) Captura figura1-modificado_a.c

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

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

void Funcion(int *R) {
    int ii, i, X1, X2;

for (ii = 1; ii <= 40000; ii++) {
        X1 = 0;
        X2 = 0;

    for (i = 0; i < 5000; i += 4) {
        X1 += 2 * s[i] .a + ii;
        X2 += 3 * s[i] .b - ii;
        X1 += 2 * s[i] .i + 1] .a + ii;
        X2 += 3 * s[i] .b - ii;
        X1 += 2 * s[i] .a + ii;
        X2 += 3 * s[i] .b - ii;
        X1 += 2 * s[i] .a + ii;
        X2 += 3 * s[i] .b - ii;
        X1 += 2 * s[i] .a + ii;
        X2 += 3 * s[i] .a + ii;
        X3 += 3 .a + ii;
        X4 += 3 * s[i] .a + ii;
        X5 += 3 .a + ii;
        X5 += 3 .a + ii;
        X6 += 3 .a + ii;
        X7 += 3 .a + ii;
        X8 += 3 .a + ii;
        X9 += 3 .a + ii;
        X1 += 2 .a + ii;
        X1 += 2 .a + ii;
        X2 += 3 .a + ii;
        X3 += 3 .a + ii;
        X4 += 3 .a + ii;
        X9 += 3 .a + ii;
        X1 += 2 .a + ii;
        X1 += 2 .a + ii;
        X1 += 2 .a + ii;
        X1 += 3 .a + ii;
        X2 += 3 .a + ii;
        X3 += 3 .a + ii;
        X4 += 3 .a + ii;
        X4 += 3 .a + ii;
        X3 += 3 .a + ii;
        X4 += 3 .a + ii;
        X5 += 3 .a + ii;
        X4 += 3 .a + ii;
        X4 +=
```

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

```
mié 30 may - 20:36 ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david gcc -02 -lrt figural-original.c -o figural-original

mié 30 may - 20:36 ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david gcc -02 -lrt figural-modificado_a.c -o figural-modificado_a

mié 30 may - 20:36 ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david gcc -02 -lrt figural-modificado_b.c -o figural-modificado_b

mié 30 may - 20:36 ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david //figural-original
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.285508453

mié 30 may - 20:36 ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david //figural-modificado_a
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.156228778

mié 30 may - 20:36 ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david //figural-modificado_b
R[0] = 0, R[39999] = -199995000

Tiempo (seg.) = 0.174465523
```

b) Captura figura1-modificado_b.c

```
#include <stdio.h>
#include <time.h>
struct {
   int a;
int b;
} s[5000];
void Funcion(int *R) {
  int ii, i, X1, X2;
   for (ii = 1; ii <= 40000; ii++) {
  X1 = 0;
  X2 = 0;</pre>
      for (i = 0; i < 5000; i++) {
    X1 += 2 * s[i].a + ii;
    X2 += 2 * s[i].b - ii;
      if (X1 < X2) {
  R[ii] = X1;
} else {
  R[ii] = X2;</pre>
   }
}
int main() {
  int R[40000];
   struct timespec cgt1, cgt2;
double ncgt;
   clock_gettime(CLOCK_REALTIME, &cgt1);
   Funcion(R);
  printf("R[0] = \%i, R[39999] = \%i \n", R[0], R[39999]); printf("\nTiempo (seg.) = %11.9f\n", ncgt);
   return 0;
```

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

```
mié 30 may - 20:36 > ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
        gcc -02 -lrt figural-original.c -o figural-original
mié 30 may - 20:36 > ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica 4
mié 30 may - 20:36     ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/A<u>C/Practicas/Practica 4</u>
mié 30 may - 20:36
                      ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
R[0] = 0, R[39999] = -199995000
mié 30 may - 20:36 > ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
@david > ./figural-modificado a
R[0] = 0, R[39999] = -199995000
                      ~/Escritorio/ETSIIT/2º Curso/2º Cuatrimestre/AC/Practicas/Practica_4
         ./figural-modificado b
     0, R[39999] = -199995000
```

1.1. TIEMPOS:

Modificación	-O2
Sin modificar	0.285508453
Modificación a)	0.156228778
Modificación b)	0.174465523

1.1. COMENTARIOS SOBRE LOS RESULTADOS:

Se puede observar que la modificación a es de la que mejor resultado se obtiene debido a que al unificar todos los cálculos en un solo bucle se reduce el número de iteraciones que se deben de realizar.

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

pmm-secuencial.s		pmm-secuencial- modificado_b.s		pmm-secuencial- modificado_c.s	
original.c"	.file "figura1-	modificado_a.c"	.file "figura1-	modificado_b.c"	.file "figura1-
"aMS",@progbits,1	<pre>.text .section .rodata.str1.1, l .string "R[0] = %i,</pre>	@function	.text .p2align 4,,15 .globl Funcion .type Funcion,	@function	.text .p2align 4,,15 .globl Funcion .type Funcion,
R[39999] = %i\n" .LC2:	.string "\nTiempo	Funcion: .LFB23:	.cfi_startproc leaq	Funcion: .LFB23:	.cfi_startproc
(seg.) = %11.9f\r	ין section .text.startup,"	%r8	40000+s(%rip),	%r9	40000+s(%rip),
ax",@progbits	.p2align 4,,15 .globl	.L2:	\$1, %r9d .p2align 4,,10 .p2align 3	.L2:	\$1, %r10d .p2align 4,,10 .p2align 3
main: .LFB23:	.type main, @function		leaq s(%rip), %rax movl %r9d, %edx		leaq s(%rip), %rax movl %r10d, %esi
5	.cfi_startproc subq \$160072, %rsp .cfi_def_cfa_of		xorl' %ecx, %ecx xorl %r10d, %r10d .p2align 4,,10		xorl %edx, %edx xorl %ecx, %ecx
fset 160080	xorl %edi, %edi leaq	.L3:	mov1	.L3:	%ecx, %ecx .p2align 4,,10 .p2align 3
	16(%rsp), %rsi movq %fs:40, %rax movq %rax,		(%rax), %esi addq \$32, %rax leal (%rdx,%rsi,2),		(%rax), %r8d addq \$8, %rax leal (%rsi,%r8,2),
160056(%rsp)	xorl %eax, %eax call	%r11d	movl -28(%rax), %esi addl	%r8d	addl %r8d, %ecx movl
clock_gettime@PL1	T leaq 40004+s(%rip),	%esi	%r10d, %r11d leal (%rsi,%rsi,2),	%edx	-4(%rax), %r8d leal (%rdx,%r8,2),
%r9	leaq 48(%rsp), %r11 mov1 \$1, %r10d leaq -4(%r9), %r8 .p2align 4,,10 .p2align 3	%r10d	subl %edx, %esi addl %esi, %ecx movl -24(%rax), %esi leal (%rdx,%rsi,2),		subl Mesi, Wedx cmpq Wrax, Wr9 jne L3 cmpl Wedx, Wecx cmovl
.L2:	leaq s(%rip), %rax movl %r10d, %edi xorl		movl -20(%rax), %esi addl %r10d, %r11d leal (%rsi,%rsi,2),	%r10,4)	%ecx, %edx movl %edx, (%rdi, addq \$1, %r10
.L3:	%esi, %esi .p2align 4,,10 .p2align 3	%esi	subl %edx, %esi addl		\$40001, %r10 jne .L2
	movl (%rax), %edx addq \$8, %rax leal (%rdi,%rdx,2),	%r10d	%ecx, %esi movl -16(%rax), %ecx leal (%rdx,%rcx,2),	.LFE23:	rep ret .cfi_endproc .size Funcion,
%edx	addl %edx, %esi cmpq %r8, %rax		movl -12(%rax), %ecx addl %r11d, %r10d leal	"aMS",@progbits,1	string
	jne .L3 leaq	%ecx	(%rcx,%rcx,2), subl	R[39999] = %i\n" .LC2:	*K[0] = %1,

	4+s(%rip), %rax		%edx, %ecx		.string "\nTiempo
	xorl %ecx, %ecx		addl %ecx, %esi	(seg.) = %11.9f\r	"\niiempo ""
	.p2align 4,,10 .p2align 3		movl´ -8(%rax), %ecx	()	.section .text.startup,"
.L4:	_		leal	ax",@progbits	
	movl (%rax), %edx	%ecx	(%rdx,%rcx,2),		.p2align 4,,15 .globl
	adda	, accar	addl		măin
	\$8, %rax leal		%ecx, %r10d movl		.type main, @function
%edx	(%rdx,%rdx,2),		-4(%rax), %ecx leal	main: .LFB24:	, -
/aeux	subl		(%rcx, %rcx, 2),	. LI 024.	.cfi_startproc
	%edi, %edx addl	%ecx	subl		subq \$160072, %rsp
	%edx, %ecx cmpq		%edx, %ecx addl	fset 160080	\$160072, %rsp .cfi_def_cfa_of
	%rax, %r9		%esi, %ecx	1300 100000	xorl
	jne . L4		cmpq %rax, %r8		%edi, %edi leaq
	cmpl		jne '		16(%rsp), %rsi
	%esi, %ecx cmovg		.L3 cmpl		movq %fs:40, %rax
	%esi, %ecx movl		%ecx, %r10d cmov1		movq %rax,
(%r11 %r10 4)	%ecx,		%r10d, %ecx	160056(%rsp)	
(%r11,%r10,4)	addq \$1, %r10		movl %ecx, (%rdi,		xorl %eax, %eax
	\$1, %r10 cmpq	%r9,4)	addq		call
	\$40001, %r10		\$1, %r9	clock_gettime@PL1	T
	jne .L2		cmpq \$40001, %r9		leaq 48(%rsp), %rdi
	leaq 32(%rsp), %rsi		jne .L2		call Funcion
	xorl		rep_ret _		lead
	%edi, %edi call	.LFE23:	.cfi_endproc		32(%rsp), %rsi xorl
clock_gettime@PL	т		.size Funcion,		%edi, %edi call
CIOCK_gettime@rL	movq	Funcion	,		
	40(%rsp), %rax subg		.section .rodata.str1.1,	clock_gettime@PL1	movq
	24(%rsp), %rax	"aMS",@progbits,1 .LC1:	1		40(%rsp), %rax subg
	leaq .LC1(%rip),		strina		24(%rsp), %rax
%rsi	pxor	R[39999] = %i\n"	"R[0]"= %i,		leaq .LC1(%rip),
	%xmm0, %xmm0 mov1	.LC2:	etrina	%rsi	
	160044(%rsp),	, , , , , , , , , , , ,	.string "\nTiempo		pxor %xmm0, %xmm0
%ecx	pxor	(seg.) = %11.9f\r	n" .section		movl 48(%rsp), %edx
	%xmm1, %xmm1 movl	ax",@progbits	.text.startup,"		pxòr %xmm1, %xmm1
	48(%rsp), %edx	ax , wprognics	.p2align 4,,15		movl
	movl \$1, %edi		.globl main	%ecx	160044(%rsp),
	CVts12saq		.type main, @function		movl \$1 %odi
	%rax, %xmm0 movq	main:	main, wranction		\$1, %edi cvtsi2sdq
	32(%rsp), %rax subq	.LFB24:	.cfi_startproc		%rax, %xmm0 movq
	16(%ṛṣp), %rax		subq \$160072, %rsp		32(%rsp), %rax
	cvtsi2sdq %rax, %xmm1		.cfi_def_cfa_of		subq 16(%ṛṣp), %rax
	xorl %eax, %eax	fset 160080	xorl		cvtsi2sdq %rax, %xmm1
	divsd		%edi, %edi		xorl
%×mm0	.LCO(%rip),		leaq 16(%rsp), %rsi		%eax, %eax divsd
	addsd %xmm1, %xmm0		movq %fs:40, %rax	%×mm0	.LCO(%rip),
	movsd		movq		addsd
	%xmm0, 8(%rsp) call	160056(%rsp)	%rax,		%xmm1, %xmm0 movsd
printf chk@PLT			xorl %eax, %eax		%xmm0, 8(%rsp) call
	leaq .LC2(%rip),		call	printf chk@PLT	*
%rsi		clock_gettime@PL	Γ,		movsd
	movsd 8(%rsp), %xmm0		leaq 48(%rsp), %rdi		8(%rsp), %xmm0 leaq
	movl \$1, %edi		call Funcion	%rsi	.LC2(%rip),
	movl		lead	/01 JI	movl
	\$1, %eax call		32(%rsp), %rsi xorl		\$1, %edi movl
printf_chk@PLT			%edi, %edi call		\$1, %eax call
bi Tilci _ciik@bEi	xorl	alaak++:0-:-		neiste -1105:-	OULL
	%eax, %eax movq	clock_gettime@PL ⁻	mova	printf_chk@PLT	movq
%rsi	160056(%rsp),		40(%rsp), %rax subq	%rdx	160056(%rsp),
	xorq		24(%rsp), %rax		xorq %fs:40, %rdx
	%fs:40, %rsi jne		leàq .LC1(%rip),		jne
	.L13 addg	%rsi	pxor		.L12 xorl
	\$160072, %rsp		%xmm0, %xmm0		%eax, %eax
tate	.cfi_remember_s		movl 48(%rsp), %edx		addq \$160072, %rsp
fset 8	.cfi_def_cfa_of		pxor %xmm1, %xmm1	tate	.cfi_remember_s
	ret		mov1 ´	_	.cfi_def_cfa_of
.L13:	.cfi_restore_st	%ecx	160044(%rsp),	fset 8	ret
ate	call		movl \$1, %edi	.L12:	.cfi_restore_st
otosli ski sila			cvtsi2sdq	ate	
stack_chk_fail(WAL I		%rax, %xmm0		call
18			- · ·	itoctura y Tocnología	

```
movq
32(%rsp), %rax
subq
16(%rsp), %rax
cvtsi2sdq
%rax, %xmm1
xorl
%eax, %eax
divsd
.LCO(%rip),
                                     .cfi_endproc
.LFE23:
                                                                                                                                                  _stack_chk_fail@PLT
.cfi_endproc
                                   .size
main, .-main
.comm
s,40000,32
.section
.rodata.cst8,"a
                                                                                                                                               .LFE24:
                                                                                                                                                                                   .size
main, .-main
.comm
s,40000,32
.section
.rodata.cst8,"a
M",@progbits,8
                                     .align 8
.LC0:
                                                                        %xmm0
                                                                                                                                               M",@progbits,8
                                                                                                           addsd
%xmm1, %xmm0
movsd
%xmm0, 8(%rsp)
call
                                    .long
                                                                                                                                                                                    .align 8
                                                                                                                                               .LC0:
                                                                                                                                              .LCU: .long 0 .long 1104006501 .ident "GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0" .section .note.GNU-stack,"",@progbits
                                   0
.long
1104006501
.ident
"GCC: (Ubuntu
7.3.0"
.section
.note.GNU-
7.3.0-16ubuntu3)
                                                                           _printf_chk@PLT
                                                                                                            movsd
8(%rsp), %xmm0
leaq
.LC2(%rip),
stack,"",@progbits
                                                                        %rsi
                                                                                                           movl
$1, %edi
movl
$1, %eax
call
                                                                         __printf_chk@PLT
                                                                                                            movq
160056(%rsp),
                                                                        %rdx
                                                                                                           xorq
%fs:40, %rdx
jne
.l.12
xorl
%eax, %eax
addq
$160072, %rsp
.cfi_remember_s
                                                                        tate
                                                                                                            .cfi_def_cfa_of
                                                                        fset 8
                                                                                                            ret
                                                                        .L12:
                                                                                                            .cfi_restore_st
                                                                        ate
                                                                                                            call
                                                                        __stack_chk_fail@PLT
.cfi_endproc
                                                                        .LFE24:
                                                                                                           .size
main, .-main
.comm
s,40000,32
.section
.rodata.cst8,"a
                                                                        M",@progbits,8
                                                                                                            .align 8
                                                                        .LC0:
                                                                     ..ung
0
.long
1104006501
.ident
"GCC: (Ubuntu
7.3.0-16ubuntu3) 7.3.0"
.section
.note.GNU-
stack,"",@progbits
                                                                                                            .long
```

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 \le N, i++) y[i] = a*x[i] + y[i];
```

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

CAPTURA CÓDIGO FUENTE: daxpy.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
clock_gettime(CLOCK_REALTIME, cgt2);
int_main(int argc, char *argv[]) {
   if (argc < 3) {
    fprintf(stderr, "ERROR: falta tam del vector y constante\n");</pre>
      exit(1);
  int *y, *x;
int a = strtol(argv[2], NULL, 10);
unsigned n = strtol(argv[1], NULL,
y = (int *)malloc(n * sizeof(int));
x = (int *)malloc(n * sizeof(int));
                                                      10);
  unsigned i;
for (i = 0; i < n; i++) {
    y[i] = i + 2;
    x[i] = i * 2;
   struct timespec cgt1, cgt2;
double ncgt;
   DAXPY(y, x, a, n, \&cgt1, \&cgt2);
             (double)(cgt2.tv_sec - cgt1.tv_sec) +
(double)((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));
   printf("y[0] = %i, y[%i] = %i\n", y[0], n - 1, y[n - 1]);
printf("\nTiempo (seg.) = %11.9f\n", ncgt);
   free(y);
free(x);
   return 0:
```

Tiempos ejec.	-O0	-Os	-O2	-O3	
	0.355463460	0.125826016	0.103762091	0.082723644	

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

COMENTARIOS QUE EXPLIQUEN LAS DIFERENCIAS EN ENSAMBLADOR:

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

daxpy0	00.s	daxp	y0s.s	daxpy	02.s	daxp	y03.s
<pre>@function DAXPY:</pre>	.file "daxpy.c" .text .globl DAXPY .type DAXPY,	" @function	.file "daxpy.c .text .globl DAXPY .type DAXPY,	4,,15	.file "daxpy.c" .text .p2align .globl DAXPY .type DAXPY,	4,,15	.file "daxpy.c" .text .p2align .globl DAXPY .type DAXPY,
tproc	.cfi_star	DAXPY: .LFB23: rtproc	.cfi_sta	@function DAXPY: .LFB41:	.cfi_start	<pre>@function DAXPY: .LFB41:</pre>	.cfi_star
cfa_offset 16	%rbp .cfi_def_ .cfi_offs	_cfa_offset	pushq %r13 .cfi_def 16 .cfi off	fa_offset 16	pushq %r14 .cfi_def_c	tproc cfa_offset :	pushq %r14 .cfi_def_
,	<pre>movq %rsp, .cfi_def_</pre>	set 13, -16		t 14, -16	.cfi_offse pushq %r13	et 14, -16	.cfi_offs pushq %r13

	•		0.4	I		I	. 6 . 1 . 6
cfa_register	suba	_cfa_offset	.cfi_off	fa_offset 24	.cfi_def_c	cfa_offset 24	.cfi_def_ !
	\$64, %rsp	set 12, -24	movq	t 13, -24	.cfi_offse	et 13, -24	.cfi_offs
	%rdi, -		%rsi,	13, -24	movl	et 13, -24	movl
24(%rbp)	movq	%r12	pushq	%r14d	%ecx,	%r14d	%ecx,
	%rsi, -		%rbp '	701 144	pushq	701 144	pushq
32(%rbp)	movl	_cfa_offset	.cfi_def		%r12 .cfi_def_c		%r12 .cfi_def_
00(0(1)	%edx, -		.cfi_off	fa_offset 32		cfa_offset 32	
36(%rbp)	movl	set 6, -32	pushq	t 12, -32	.cfi_offse	et 12, -32	.cfi_offs
40 (0(mbm)	%ecx, -		%rbx	,	pushq	,	pushq
40(%rbp)	movq	cfa offset	.cfi_def 40		%rbp .cfi def c		%rbp .cfi def
40(%rbp)	%r8, -	10	.cfi_off	fa_offset 40	ofi offco	cfa_offset 40	cfi_offs
48(%rbp)	movq	set 3, -40	movq	t 6, -40	.cfi_offse	et 6, -40	.011_0118
56(%rbp)	%r9, -	%rsi	%r8,		movq %rsi, %rbp		movq %rsi,
30(701 5P)	movq	701 31	mova		pushq	%r12	
48(%rbp), %ra	- 1X	%rbx	%rdi,		%rbx .cfi_def_c		pushq %rbx
10(701 55)) 701 0	movq	701 27	xorl	fa_offset 48		-555+ 40	.cfi def
%rsi	%rax,	%edi	%edi,	t 3, -48	.cfi_offse	cfa_offset 48	.cfi_offs
	movl \$0 %edi		movl %edy	,	movq %r8, %rsi	et 3, -48	_
	\$0, %edi call	%r13d	%edx,		movq		movq %r8, %rsi
clock gettime			subq \$24,		%rdí, %rbx xorl		movq %rdi,
GIOCK_GCCCIMC	movl	%rsp			%edi, %edi	%rbx	
4(%rbp)	\$0, -	_cfa_offset	.cfi_def		movl %edx,		xorl %edi,
701 56)	jmp		movl	%r12d	,	%edi	,
.L3:	. L2	%ebp	%ecx,		movq %r9, %r13		movl %edx,
	movl	, , , , , , , , , , , , , , , , , , ,	movq		call	%r13d	,
%eax	-4(%rbp),	8(%rsp)	%r9,	clock_gettime	@PLT		movq %r9, %rbp
	leaq	` ' ' '	call		testl %r14d,		subq \$16, %rsp
%rax,4), %rdx		clock_gettir		%r14d	,		.cfi_def_
	movq		movq 8(%rsp),		je .L2	cfa_offset 64	call
24(%rbp), %ra	ıx	%r9	_		leal		
	addq %rdx,		xorl %eax,	%eax	-1(%r14),	clock_gettime	e@PLT testl
%rax		%eax	,,,		leaq	0/514d	%r14d,
	movl (%rax),	.L2:	cmpl	%rax,4), %rdx	4(,	%r14d	je .L2
%ecx	movl	%ebp	%eax,		xorl %eax, %eax		L2 leag
	-4(%rbp),	/левр	jbe		.p2align		16(%r12),
%eax	leaq		.L6 movl	4,,10	.p2align 3	%rax	cmpq
0(0(,	(0/:=4.0, 0/:==::		.L3:	-	0/	%rax,
%rax,4), %rdx	(mo∨q	(%r12,%rax,	imull		movl 0(%rbp,	%rbx	leag
32(%rhn) %ra	- '	%edx	%r13d,	%rax), %ecx	imull	%rax	16(%rbx),
32(%rbp), %ra	addg	/JEUA	addl		%r12d,	701 U.A	setnb
%rax	%rdx,	(%rbx,%rax,4	%edx,	%ecx	addl		%dl cmpq
131 47	movl	(701 577 701 477	incq	(0/mb) - 0/	%ecx,	0/110	%rax,
%eax	(%rax),		%rax imp	(%rbx,%rax)	addq \$4, %rax cmpq	%r12	setnb
	imull	16.	jmp .L2		\$4, '%rax		%al
36(%rbp), %ea		.L6:	ạddq		cmpq %rax, %rdx		orb %al, %dl
	movl %eax,	%rsp	\$24,		jne ' .L3		je ′ .L3_
%edx	-	•	.cfi_def	.L2:			cmpl
	movl -4(%rbp),	_cfa_offset	40 mo∨q		popq %rbx		\$6, %r14d jbe
%eax	_	0/roi	%r9,	fo officet 40	.cfi_def_c		.L3
	leaq 0(,	%rsi	xorl	fa_offset 40	movq		movq %rbx,
%rax,4), %rsi	_	%edi	%edi,		%r13, %rsi xorl	%rsi	xorl
	movq -	/JEUT	popq %rbx		%edi, %edi		%edi,
24(%rbp), %ra	ıx addq		%rbx .cfi_def		popq %rbp	%edi	shrq
0/	%rsi,	_cfa_offset	32	EEE	.cfi_def_c		\$2, %rsi
%rax	addl		popq %rbp	fa_offset 32	popa		negq %rsi
	%ecx,		.cfi_def		popq %r12		and1
วา							

0/ a d v		-££+	0.4				ΦΩ 0/00-
%edx	movl	_cfa_offset	popq	fa_offset 24	.cfi_def_c		\$3, %esi je
(%rax)	%edx,		%r12 .cfi def		popq %r13		.L4 movl
	addl \$1, -	_cfa_offset		fa offset 16	.cfi_def_c	%eax	(%r12),
4(%rbp)	Ψ1,		popq %r13	14_011300 10	popq %r14	70CUX	movl
.L2:	movl	_cfa_offset	.cfi_def 8		%r14 .cfi_def_c		\$1, %edi imull
%eax	-4(%rbp),		jmp	fa_offset 8	jmp	%eax	%r13d,
70CUX	cmpl	clock_gettim				70CUX	addl
40(%rbp), %ea	ax	proc	.cfi_end	clock_gettime	@PLT .cfi_endpr	(%rbx)	%eax,
	jb .L3	.LFE23:	.size	oc .LFE41:		,	cmpl \$1, %esi
	movq				.size		je
56(%rbp), %ra	ax	DAXPY,DAX	(PY .section	DAXPY	DAXPY,		.L4 movl
	movq %rax,	str1.1,"aMS"	.rodata.		.section .rodata.st	%eax	4(%r12),
%rsi	_ ′	1,1	, wprogbics	r1.8,"aMS",@p	roabits,1	70CUX	movl
	movl \$0, %edi	. LC0:	.string	.LC0:	.align 8		\$2, %edi imull
	\$0, %edi call	 falta tam de	"ERROR:		.string "ERROR:	%eax	%r13d,
clock_gettime	•	constante\n"		falta tam del	vector y	70CUX	addl
	nop leave	.LC2:	.string	constante\n"	.section	4(%rbx)	%eax,
cfa 7, 8	.cfi_def_	 %iv[%i]	"∨เคา ≚ั	r1.1,"aMS",@p	.rodata.st	, ,	cmpl \$3, %esi
3.4.7, 0	ret	%i, y[%i] = .LC3:	70± (11	.LC2:			jne
roc	.cfi_endp		.string "\		.string "y[0] =		.L4 movl
.LFE5:	.size	nTiempo (seg %11.9f\n"	j.) =	%i, y[%i] = % .LC3:	si\n" -	%eax	8(%r12),
DAVDV	DAXPY,	/022101 (11	.section	12001	.string	70047	movl
DAXPY	.section	artup,"ax",@	.text.st progbits	(seg.) = %11.	"\nTiempo 9f\n"		\$3, %edi imull
	.rodata .align 8		.globl main		.section .text.star	%eax	%r13d,
.LC0:			.type	tup,"ax",@pro	gbits	, and an	addl
	.string "ERROR:	@function	maın,	4,,15	.p2align	8(%rbx)	%eax,
falta tam del constante\n"	L vector y	main: .LFB24:			.globl main	.L4:	movl
.LC2:	.string	rtproc	.cfi_sta		.type main,	12(%rsp)	%r13d,
0/= ./[0/=] = 0	"∨[0] ≚		pushq	@function	main,	12(70139)	movl
%i, y[%i] = % .LC3:	9Τ / II		%r14 .cfi_def	main: .LFB42:		%r8d	%r14d,
	.string "\nTiempo	_cfa_offset	16 .cfi_off	proc	.cfi_start		xorl %eax,
(seg.) = %11.	.9f\n"	set 14, -16		p. 00	pushq %r14	%eax	movd
	.text .globl		pushq %r <u>1</u> 3		.cfi_def_c		12(%rsp),
	măin .type	_cfa_offset	.cfi_def 24	fa_offset 16	.cfi_offse	%×mm4	subl
@function	main,	set 13, -24	.cfi_off	t 14, -16		%r8d	%esi,
main:		300 13, -24	pushq		pushq %r13	701 GG	movl
.LFB6:	.cfi_star		%r12 .cfi_def	fa_offset 24	.cfi_def_c	%esi	%esi,
tproc	pushq	_cfa_offset	32 .cfi_off	t 13, -24	.cfi_offse		salq \$2, %rsi
	%rbp .cfi_def_	set 12, -32	pushq	-, <u>-</u> .	pushq %r12		movl %r8d,
cfa_offset 16	5		%rbp '	fo officet oo	.cfi_def_c	%r9d	_
et 6, -16	.cfi_offs	_cfa_offset	.cfi_def 40	fa_offset 32	.cfi_offse		xorl %edx,
	movq %rsp,	set 6, -40	.cfi_off	t 12, -32	pushq	%edx	pshufd
%rbp	.cfi def		pushq %rbx		%rbp .cfi_def_c	%xmm4, %xmm2	\$0,
cfa_register	6		.cfi_def	fa_offset 40		/0/1111114 / /0/111111Z	leaq
	addq \$-128,	_cfa_offset	48 .cfi_off	t 6, -40	.cfi_offse	%rsi), %rcx	(%rbx,
%rsp	movl	set 3, -48	subq		pushq %rbx		shrl \$2, %r9d
100(%rbp)	%edi, -	%ren	\$64,	fa offcot 40	.cfi_def_c		addq %r12,
100(%rbp)	movq	%rsp	.cfi_def	fa_offset 48	.cfi_offse	%rsi	
112(%rbp)	%rsi, -	_cfa_offset	mova	t 3, -48	subq		movdqa %xmm2,
` ' '	movq %fs:40,	%rax	%fs:40,		\$64, %rsp .cfi_def_c	%xmm3	psrlq
%rax	,	, un	movq %ray	fa_offset 112		%×mm3	\$32,
	movq		%rax,	Donto Aven	itaatura v Taanale	aía de Computadore	20

0 (0(1)	%rax, -	56(%rsp)			movq	4 40	.p2align
8(%rbp)	vorl		xorl	0/50	%fs:40,	4,,10	naalian
	xorl %eax,	%eax	%eax,	%rax	movq	3	.p2align
%eax	<i>π</i> c αλ,	/0Cax	cmpl		%rax,	.L6:	
/00 ax	cmpl		\$2, %edi	56(%rsp)	701 W/Y	1201	movdqu
	\$2, -			(-	xorl		(%rsi,
100(%rbp)			jg .L8		%eax, %eax	%rax), %xmm0	`
	jg . L5		movq		cmpl'		addl
	movq	stderr(%rip),	%rci		\$2, %edi jle		\$1, %edx movdqa
	шочч	Stueri (%i ip),	leag		.L18		%xmm0,
stderr(%rip)	, %rax		.LCO(%ri		movq	%xmm1	70741111107
	′ movq	p), %rdi	-		16(%rsi),		psrlq
0/	%rax,		call	%rdi		0/0	\$32,
%rcx	movl	fputs@PLT			movq %rsi, %rbx	%×mm0	pmuluda
	\$40, %edx	I puts@FLI	movl		mov1		%xmm3,
	mov1		\$1. %edi		\$10, %edx	%×mm0	70XIIIIIO ,
	\$1, %esi		\$1, %edi call		xorl		pshufd
	leag		exit@PLT		%esį, %esi	0, 0	\$8,
) %rdi	.LCÖ(%rip	.L8:	mova		movl´	%xmm0, %xmm0	nmuluda
), %rdi	call		movq		\$-1, %r13d call		pmuludq %xmm2,
	Cull	16(%rsi), %rd	li		strtol@PLT	%xmm1	70XIIIII2 ,
fwrite@PLT		() /	movq		movq		pshufd
	movl	04.1	%rsi,	04 - 12	8(%rˈbx),	0,	\$8,
	\$1, %edi call	%rbx	mov/1	%rdi	vorl	%xmm1, %xmm1	nunnaki da
	call exit@PLT		movl \$10,		xorl %esi, %esi		punpckldq %xmm0,
.L5:	CVTCALFI	%edx	Ψ±0,		movl	%xmm1	/J/IIIIIO,
	movq		xorl		\$10, %edx		movdqa
	- '		%esi,		movq		(%rcx,
112(%rbp), %	rax	%esi	11		%rax, %r12	%rax), %xmm0	
	addq \$16, %rax		call		call strtol@PLT		paddd %ymm1
	movq	strtol@PLT			movl	%×mm0	%xmm1,
	(%rax),	301 001@121	movq		%eax, %ebp	707(111110	movaps
%rax			8(%rbx),		movq		%×mm0,
	movl	%rdi			%rax, %r14	(%rcx,%rax)	
	\$10, %edx		movl		salq′ \$2, %rbp		addq \$16, %rax
	\$0, %esi	%edx	\$10,		ΦΖ, %ΓυΡ		\$16, %rax cmpl
	movq	70CUX	xorl		%rbp, %rdi		%edx,
	%rax,		%esi,		call	%r9d	,
%rdi	11	%esi			malloc@PLT		ja
	call		movd		movq %rbp %rdi		. L6
strtol@PLT		%r13	%rax,		%rbp, %rdi movq		movl %r8d,
301001@101	movl	701 13	call		%rax, %rbx	%edx	701 OU,
	%eax, -		0411		call [']	700 071	andl
80(%rbp)		strtol@PLT	_		malloc@PLT		\$-4, %edx
	movq		mov1		testl		cmpl
112(%rbp), %	rav	%r12d	%eax,	%r14d	%r14d,	%r8d	%edx,
112(%) bp), %	addq	701 12U	movq	701 140	movq	701 Ou	leal
	\$8, %rax		%rax,		%rax, %rbp		(%rdi,
	movq	%rbx			je	%rdx), %eax	
0/ray	(%rax),		movl		.L12		je .L2 ₋
%rax	movl	%r14d	%eax,		leal		.L2 movl
	\$10, %edx	/VI 17U	salo	(%r14,%r13),	%esi		%eax,
	movl		salq \$2, %r12	1,111 = 0,7	xorl	%edx	,
	\$0, %esi		movq %r12,		%eax, %eax		movl
	movq %ray	%rdi	%r12,		movq′ %rci %r12	(0/r10 0/rdy 4)	%00%
%rdi	%rax,	%rdi	call		%rsi, %r13 addq \$1, %rsi .p2align	(%r12,%rdx,4)	, %ecx imull
, vi u ±	call		Juli		\$1, %rsi		%r13d,
		malloc@PLT			.p2align	%ecx	
strtol@PLT	may : 1		movq	4,,10	m0s1		addl
	movl %oay	%rdi	%r12,	1121	.p2align 3	(%rhy %rdy 4)	%ecx,
76(%rbp)	%eax, -	%rdi	movq	.L13:	leal	(%rbx,%rdx,4)	leal
. 5(701 50)	movl		%rax,		2(%rax),		1(%rax),
	-	%rbp	•	%edx		%edx	
76(%rbp), %e	ax		call		mov1		cmpl
	salq \$2 %ray	mallea@DLT		(0/rhy 0/roy 4)	%edx,	0/r14d	%edx,
	\$2, %rax movq	malloc@PLT	movq	(%rbx,%rax,4)	leal	%r14d	jbe
	%rax,		%rax,		(%rax,		.L2
%rdi	,	%r12	•	%rax), %edx	,		movl
	call		xorl	''	movl	(0) 15 0	
molloo@D: T		%00V	%eax,	0/9/rhn 9/22:	,%edx,	(%r12,%rdx,4)	, %ecx
malloc@PLT	movq	%eax .L9:		0(%rbp,%rax,4	t) adda		imull %r13d,
	%rax, -	, LJ.	cmpl		1) addq \$1, %rax cmpq	%ecx	701 13 0,
72(%rbp)	,		%eax,		cmpq		addl
` ' '	movl	%r14d	,		%rsi, %rax		%ecx,
7.4		I		I	*	I .	-

	_		jbe		jne	(%rbx,%rdx,4)	
76(%rbp), %ea	ax salu		.L13 leal	.L12:	.L13	(701 57, 701 47, 4)	leal 2(%rax),
	salq \$2, %rax movq	%edx	2(%rax),		leaq 32(%rsp),	%edx	cmpl
%rdi	%rax,		movl %edx,	%r9	leaq	%r14d	%edx,
malloc@PLT	call	0(%rbp,%rax,4	leal (%rax,	%r8	16(%rsp), movl		jbe .L2 movl
marrocer E1	movq	%rax), %edx	_	0/	%r14d,	(0/:-40 0/: A)	
64(%rbp)	%rax, - movl	(%r12,%rax,4)	movl %edx,	%ecx	movl %r12d,	(%r12,%rdx,4)	, %ecx imull %r13d,
84(%rbp)	\$0, -	, , , , ,	incq %rax	%edx	movq	%ecx	addl
04(%i bp)	jmp .L6		jmp .L9		%rbp, %rsi mova	(%rbx,%rdx,4)	%ecx,
.L7:	movl	.L13:	leaq		%rbx, %rdi call		leal 3(%rax),
84(%rbp), %ea		40(%rsp), %r9) _		DAXPY movq	%edx	cmpl
	leal 2(%rax),		leaq	%rax	40(%rsp),	%r14d	%edx,
%edx	movl	24(%rsp), %r8	movl		subq 24(%rsp),		jbe .L2_
84(%rbp), %ea	-	%ecx	%ebx,	%rax	leag		movl
	îleaq 0(,		movl %r13d,	, %rsi	.LC2(%rip)	(%r12,%rdx,4)	imull
%rax,4), %rc>		%edx	movq	, 5_	pxor %xmm0,	%ecx	%r13d,
72(%rbp), %ra		 %rsi	%r12,	%×mm0	movl	7.00071	addl %ecx,
	addq %rcx,	701 31	movq %rbp,	%edx	(%rbx),	(%rbx,%rdx,4)	leal
%rax	movl	%rdi	call		pxor %xmm1,	%edx	4(%rax),
(%rax)	%edx,		DAXPY movq	%×mm1	movl		cmpl %edx,
,	movl -	48(%rsp), %ra	•	%ecx	%r13d,	%r14d	jbe
84(%rbp), %ea	ax leal	(subq		movl \$1, %edi		.L2 movl
%rax), %edx	(%rax,	32(%rsp), %ra	ax leaq		cvtsi2sdq %rax,	(%r12,%rdx,4)	
70. 00.77	movl	p), %rsi	.LC2(%ri	%xmm0	movq	(10. ==) 10. (2.1)	addl \$5, %eax
84(%rbp), %ea	ax leag	p), %131	movl 0(%rbp),	%rax	32(%rsp),		imull %r13d,
%rax,4), %rc>	0(, '	%edx	movl	70. 67.	subq 16(%rsp),	%ecx	addl
	movq -		\$1, %edi	%rax	cvtsi2sdq	(%rbx,%rdx,4)	%ecx,
64(%rbp), %ra	ax addg	cvtsi2sdq %xmm0	%rax,	%×mm1	%rax,		cmpl %eax,
%rax	%rcx,		movq		movl %r13d,	%r14d	ibe
	movl %edx,	40(%rsp), %ra	ax suba	%eax	movl		.L2 imull
(%rax)	addl	24(%rsp), %ra	•	%rax,4), %r8d	(%rbx,	(%r12,%rax,4)	
84(%rbp)	\$1, -	cvtsi2sdq	%rax,		xorl %eax, %eax	(10. == / /01 (4./ + /	addl %r13d,
.L6:	movl	%xmm1	leal		divsd .LC1(%rip)	(%rbx,%rax,4)	701 ±00,
84(%rbp), %ea	-	1(%rbx), %ea>	-	, %×mm0	addsd		addq \$16, %rsp
5-(M bp), %ee	cmpl		movl 0(%rbp,	%xmm0	%xmm1,	mber_state	.cfi_reme
76(%rbp), %ea	ax ib	%rax,4), %r80	movq	7.57.111110	movsd %xmm0,	cfa_offset 48	.cfi_def_
	.L7 leag	%rcx	%rax,	8(%rsp)	call	014_01136t 46	movq %rbp,
32(%rbp), %r8	<u>-</u> '		xorl %eax,	printf_chk@		%rsi	xorl
52(% bp), %	leaq	%eax	divsd	printr_cire	movsd	%edi	%edi,
48(%rbp), %rd		n) %vmm0	.LC1(%ri	%×mm0	8(%rsp),	%edi	popq %rbx
76 (0/shs\ 0/-	movl -	p), %xmm0	addsd	%roi	leaq .LC3(%rip)	ofo officet 10	.cfi_def_
76(%rbp), %ed	movl	%×mm0	%xmm1,	, %rsi	movl	cfa_offset 40	popq
80(%rbp), %ed		0(0/505)	movsd %xmm0,		\$1, %edi movl	ofo offi-t 00	%rbp .cfi_def_
64(%rbn) %	movq -	8(%rsp)	call		\$1, %eax call	cfa_offset 32	popq %r12
64(%rbp), %rs	2.1			Donto Argu	iitootura y Toonak	 ogía de Computadore	

	movq -	printf_chk@PLT movsd	printf_chk@PLT movq	.cfi_def_ cfa_offset 24
72(%rbp),	%rax	8(%rsp),	%rbx, %rdi	popq
	movq %r8, %r9	%xmm0 leaq	call free@PLT	%r13 .cfi_def_
	movq	LC3(%ri	mova	cfa_offset 16
	%rdi, %r8	p), %rsi	%rbp, %rdi	popq
	movq %rax,	movl \$1, %edi	call free@PLT	%r14 .cfi_def_
%rdi	•	movb	xorl	cfa_offset 8
	call	\$1, %al	%eax, %eax	jmp
	DAXPY movq	call	movq 56(%rsp),	clock_gettime@PLT_
22/2/ 1)	- '	printf_chk@PLT	%rcx	p2align.
32(%rbp),	%rax movq	movq %rbp,	xorq %fs:40,	4,,10 .p2align
	- '	%rdi	%rcx	3
48(%rbp),		call	jne .L19	.L3:
	subq %rax,	free@PLT movq	adda	.cfi_rest ore_state
%rdx	,	%r12,	\$64, %rsp	leal
	movq %rdx,	%rdi call	.cfi_remem ber_state	-1(%r14), %eax
%rax	,	free@PLT	cfi def c	leaq
	cvtsi2sdq %rax,	xorl %eav	fa_offset 48	%ray 4) %rcy 4(,
%xmm1	/01 αλ ₁	%eax, %eax	popq %rbx	%rax,4), %rcx xorl
	movq	movq	.cfi_def_c	%eax,
24(%rbp),	%rdx	56(%rsp), %rcx	fa_offset 40	%eax .p2align
_ ((() () () ()	movq	xora	popq %rbp	4,,10
40(%rbp),	- %rax	%fs:40, %rcx	.cfi_def_c fa_offset 32	.p2align
το (701 υρ) ,	subq	je	popq %r12	.L8:
%rdx	%rax,	L11 call	%r12 .cfi_def_c	movl
761 U.X	movq	Call	fa_offset 24	(%r12,%rax), %edx_
0/15.0.1	%rdx,	stack_chk_fail@PLT	popq %r13	imull
%rax	cvtsi2sdq	.L11: addg	.cfi_def_c	%r13d, %edx
	%rax,	\$64,	fa_offset 16	addl
%×mm0	movsd	%rsp .cfi_def	popq %r14	%edx, (%rbx,%rax)
	.LC1(%rip	_cfa_offset 48	.cfi_def_c	addq \$4, %rax
), %xmm2	divsd	popq %rbx	fa_offset 8 ret	\$4, %rax cmpq
	%xmm2,	.cfi def	.L18:	%rcx,
%xmm0	addsd	_cfa_offset 40	.cfi_resto	%rax
	%xmm1,	popq %rbp	re_state movq	jne .L8
%×mm0		cfi def.	'	jmp .L2
	movsd %xmm0, -	_cfa_offset 32	stderr(%rip), %rcx leag	.∟∠ .cfi_endp
56(%rbp)	,	popq %r12	.ICO(%rin)	roc
	movl -	.cfi_def _cfa_offset 24	, %rdi movl	.LFE41: .size
76(%rbp),	%eax	popq %r13	\$40, %edx	DAXPY,
	subl \$1,_%eax		movl	DAXPY .section
	movl	.cfi_def _cfa_offset 16	\$1, %esi call	.rodata.s
%00%	%eax,	popq %r14	fwrite@PLT	tr1.8,"aMS",@progbits,
%eax	leag	.cfi def	movl \$1,_%edi	.align 8
0/50: 43 0	0(,	_cfa_offset 8	call	.LCO:
%rax,4), 9	%rdx movq	ret .cfi_end	exit@PLT	.string "ERROR:
70/2/ 1 3	- ·	proc	call	falta tam del vector y
72(%rbp),	%rax addq	.LFE24: .size	stack chk fail@PLT	constante\n" .section
	%rdx,	main,	cfi_endpr	.rodata.s
%rax	movl	main .section	oc .LFE42:	tr1.1,"aMS",@progbits,
	(%rax),	.rodata.	.size	1 .LC5:
%edx		cst8,"aM",@progbits,8	main,	string
	movl -	align 8.	main .section	"y[0] = %i,_y[%i] = %i\n"
76(%rbp),		.long	.rodata.cs	.LC6:
	leal -1(%rax),	0 .long	t8,"aM",@progbits,8 .align 8	.string "\nTiempo
%esi			.LC1:	(seg.) = %11.9f\n"
	movq	1104006501 .ident	.long 0	.section .text.sta
72(%rbp),		"GCC:	.long	rtup,"ax",@progbits
	movl ((Ubuntu 7.3.0- 16ubuntu3) 7.3.0"	1104006501 .ident	.p2align
%eax	(%rax),	section	"GCC:	4,,15 .globl
	movl %ody	.note.GN U-stack,"",@progbits	(Ubuntu 7.3.0-	māin
26	%edx,	U-Stack, ,@prognits	16ubuntu3) 7.3.0"	.type

%ecx		.section		main,
%edx	movl %esi,	.note.GNU- stack,"",@progbits	<pre>@function main: .LFB42:</pre>	·
70CUX	movl %eax,		tproc	.cfi_star
%esi	leag		ср. 00	pushq %r14
), %rdi	.LC2(%rip		cfa_offset 1	.cfi_def_ 6
	movl \$0, %eax call		et 14, -16	.cfi_offs pushq
printf@PLT				%r13 .cfi def
50 (0(mbm) 0(m	movq -		cfa_offset 2	4 .cfi_offs
56(%rbp), %r	ax movq %rax, -		et 13, -24	pushq %r12
120(%rbp)	movsd		cfa_offset 3	.cfi_def_ 2
120(%rbp), %	- ×mm0		et 12, -32	.cfi_offs
), %rdi	leaq .LC3(%rip			pushq %rbp .cfi_def_
77 701 01	movl \$1, %eax		cfa_offset 4	o .cfi_offs
nrintf@DLT	call		et 6, -40	pushq
printf@PLT	movq		cfa_offset 4	%rbx .cfi_def_ 8
72(%rbp), %r	movq		et 3, -48	.cfi_offs
%rdi	%rax, call			subq \$64, %rsp
	free@PLT movq		cfa_offset 1	.cfi_def_ 12 movq
64(%rbp), %r	ax		%rax	%fs:40,
%rdi	movq %rax,		56(%rsp)	movq %rax,
761 U.I	call free@PLT		30 (%i Sp)	xorl %eax,
	movl \$0, %eax		%eax	cmpl
%rcx	movq -8(%rbp),			\$2, %edi jle .L45
701 CX	xorq %fs:40,			movq 16(%rsi),
%rcx	je .L9 ₋		%rdi	movq
	call		%rbp	%rsi, movl
stack_chk_ .L9:				\$10, %edx xorl
-5- 7 0	leave .cfi_def_		%esi	%esi,
cfa 7, 8	ret .cfi_endp		%r14d	movl \$-1,
roc .LFE6:				call
	.size main,		strtol@PLT	movq
main	.section .rodata		%rdi	8(%rˈbp), xorl
.LC1:	.align 8		%esi	%esi,
	.long			movl \$10, %edx
1104006501	.long		%rbx	movq %rax,
	.ident "GCC:			call
(Ubuntu 7.3.0 16ubuntu3) 7	.3.0"		strtol@PLT	movl
-stack,"",@p	.section .note.GNU roabits		%r12d	%eax, movq
	90	Depto. Arquitectura y Tecnolo	l ogía de Computado	•

		%rax,
	%r13	
		salq \$2, %r12 movq %r12,
	%rdi	call
	11 07 -	Call
	malloc@PLT	movq
	%rdi	movq %r12,
	701 U.I	movq %rax,
	%rbp	
		call
	malloc@PLT	test1
	×.10.1	testl %r13d,
	%r13d	
	%r12	movq %rax,
	701 12	je .L30
		movq %rbp,
	%rdx	
		addl %r13d,
	%r14d	
		movl \$4, %ecx shrq \$2, %rdx negq %rdx andl \$3, %edx leal 3(%rdx),
		shrq \$2. %rdx
		negq
		andl
		\$3, %edx leal
	%eax	3(%rdx),
	/\deax	cmpl
		cmpl \$4, %eax cmovb
	%eax	%ecx,
	, was	cmpl %eax,
	%r14d	
		jb .L37 testl %edx,
		testl %edx
	%edx	
		movl \$0,
	8(%rsp)	
		.L32
		\$1, %edx
	2/2/	je .L32 cmpl \$1, %edx movl \$2,
	0(%rbp)	movl \$0,
	(%r12)	
	(*** == /	je .L39 cmpl \$3, %edx movl \$3,
		cmp1
		⇔3, %eax movl
	4(%rbp)	
	-(\(\text{if } \text{if } \text{if } \)	movl \$2,
	4(%r12)	
		jne .L40 movl \$4,
		movĺ \$4
	8(%rbp)	
		movl \$4,
28	8(%r12)	

			movil
		- / >	movl \$3,
		8(%rsp) .L32:	
		. 2021	movd
		%xmm4	8(%rsp),
			movl %r13d,
		%r8d	
			xorl %eax,
		%eax	subl
		0/r0d	%edx,
		%r8d	movdqa .LC2(%rip
), %xmm3	.LC2(%rip
		,, , , , , , , , , , , , , , , , , , ,	pshufd \$0,
		%xmm4, %xmm0	
			salq \$2, %rdx movl %r8d,
			movl
		%edi	
			movdqa .LC3(%rip
), %xmm2	
			leaq 0(%rbp,
		%rdx), %rsi	
			\$2, %edi
		•	shrl \$2, %edi addq %r12,
		%rdx	xorl
		%ecx	%ecx,
		∞ ECX	paddd .LC1(%rip
), %xmm0	.LC1(%rip
		4,,10	.p2align
			.p2align
		3 .L34:	
			movdqa %xmm0,
		%xmm1	
			\$1, %ecx
			addl \$1, %ecx paddd %xmm2,
		%xmm1	
			movaps %xmm1,
		(%rsi,%rax)	
		%xmm1	movdqa %xmm0,
		/0XIIIII111	paddd %xmm3,
		%×mm0	
		-	pslld \$1, %xmm1 movups %xmm1,
			movups
		(%rdx,%rax)	
		. ,	addq \$16, %rax cmpl %edi,
			cmp1
		%ecx	
			jb .L34 movl %r8d,
			movl
		%edx	
			movl 8(%rsp),
		%eax	
			andl \$-4, %edx addl %edx,
			add⊥ %edx,
	Depto. Arquitectura y Tecnolo	aía do Computador	,

		1
	%eax	
		cmpl %edx,
	%r8d	
	70. 00.	je .L30
	1.04	.L30
	.L31:	1001
		leal 2(%rax),
	%eax	
		leal (%rax,
	(/ray/) (/aai	(%rax,
	%rax), %esi	movl
		%eax,
	%ecx	
		movl
	0(%rbp,%rcx,4	%edx,
	0(%rbp,%rcx,4	mov1
		%esi,
	(%r12.%rcx.4)	
		leal
	0/0.01	1(%rax),
	%ecx	cmn1
		cmpl %ecx,
	%r13d	
		jbe
		jbe .L30 leal 3(%rax),
		3(%ray)
	%esi	
		movl
		%ecx,
	%edi	addl
		%ecx,
	%ecx	
		cmpl %edx,
	0/ .40.1	%edx,
	%r13d	movl
	(%r12,%rdi,4) 0(%rbp,%rdi,4 %ecx	70COX,
	, , ,	movl
	0/0/rbp 0/rdi 4	%esi,
	७(%rbp,%ru±,4) ihe
		.L30
		leal
	0/	4(%rax),
	%ecx	mov1
		movl %edx,
	%Δ Λ 1	
		addl %edx,
	%edx	‰eux,
	/0CUA	cmp1
		cmpl %esi,
	%r13a	
		movl %edv
	(%r12,%rdi,4) 0(%rbp,%rdi,4 %edx	%edx,
	(.o / / or u _ / - T)	movl
		%ecx,
	⊍(%rbp,%rdi,4) ibo
		130 106
		ieal
		5(%rax),
	%edx	may/1
		movi %esi,
	%edi	70C3I,
		addl
		%esi,
	%esi	
		cmpl %ecx,
	%r13d	/UCUX,
		movl
		%esi,
	(%r12,%rdi,4)	movl
		%edx.
1	0(%rbp,%rdi,4	\
	0(% bp, % ux, 4	,

		T	
			jbe .L30 leal 6(%rax),
		%EUT	
			movl %ecx,
			addl %ecx,
		l%ecx	cmpl %edx,
		%r13d	movl
		(%r12,%rsi,4)	%ecx,
		101%rbn_%rs1.4	movl %edi,)
			jbe .L30 movl
		%acv	%eax,
			addl \$7, %eax addl %edx,
		/0 C UX	
		0(%rbp,%rcx,4	movl %eax,
			movl %edx,
		(%r12,%rcx,4) .L30:	lead
		1%r9	leaq 32(%rsp),
		%r8	leaq 16(%rsp),
			movl %ebx,
			movl %r13d,
		%ecx	movq %r12,
		%rsi	movq %rbp,
		l%rai	
			call DAXPY movq 40(%rsp),
		∣%rax	
		∣%rax	subq 24(%rsp),
), %rsi	leaq .LC5(%rip
		%×mm0	pxor %xmm0,
			movl 0(%rbp),
		%eax	pxor %xmm1,
		%xmm1	movl %r14d,
		%ecx	
			movl \$1, %edi cvtsi2sdq %rax,
		%XIIIIIU	
		∣%rax	movq 32(%rsp),
	Dento Arquitectura y Tecnolo		subq 16(%rsp),

		%rax	out of Ooda
			cvtsi2sdq %rax,
		%xmm1	
			movl
		%eax	%r14d,
			movl
			0(%rbp,
		%rax,4), %r8d	xorl
			%eax,
		%eax	
			divsd .LC4(%rip
), %xmm0	
			addsd
		%xmm0	%xmm1,
			movsd
			%xmm0,
		8(%rsp)	call
		printf_chk@	PLT
			movsd 8(%rsp),
		%xmm(·)	
			leaq .LC6(%rip
), %rsi	.LC6(%rip
			movl
			\$1, %edi movl
			MOVT
			\$1, %eax call
		printf_chk@	PLT
			movq %rbp,
		%rdi	
			call free@PLT
			MOVa
			movq %r12,
		%rdi	call
			free@PLT
			free@PLT xorl
		%eax	%eax,
		/0Cax	mova
		07.1	movq 56(%rsp),
		%rbx	vora
			xorq %fs:40,
		%r DX	
			jne .L46
			adda
			\$64, %rsp .cfi_reme
		mber_state	.ci T_i eille
			.cfi_def_
		cfa_offset 48	
			popq %rbx
			.cfi def
		cfa_offset 40	
			popq %rbp
			.cfi def
		cfa_offset 32	nona
			popq %r12 .cfi_def_
		ofo cffort C:	.cfi_def_
		cra_orrset 24	
			popq %r13
			.cfi def
		cfa_offset 16	nong
			popq %r14
			.cfi_def_
		cfa_offset 8	ret
		.L39:	
			.cfi_rest
3.2	 Donto Arquitoctura y Tocnolog	′ 1 0	

<u> </u>		
	ore_state	movl \$1,
	8(%rsp)	jmp .L32
	.L40:	.L32 movl
	8(%rsp)	\$2,
		jmp .L32
	.L37:	xorl %eax,
	%eax	jmp .L31
	.L45:	movq
	stderr(%rip),	%rcx
), %rdi	leaq .LCO(%rip
	,,	movl \$40, %edx movl \$1, %esi call
	fwrite@PLT	call
	1 WI 1 CO @ I E I	movl \$1, %edi call
	.L46:	call exit@PLT
		call
	stack_chk_f	ail@PLT .cfi_endp
	roc .LFE42:	.size
	main	main,
	st16 "aM" @nr	.section .rodata.c
	st16,"aM",@pr .LC1:	
		.long
		.long 1
		.long 2 .long 3
	1.021	3 .align 16
	.LC2:	.long 4
		4 .long
		.long 4 .long
		.align 16
	.LC3:	.long
		2 .long 2
		.long 2
		. Long
	st8,"aM",@pro	.section .rodata.c gbits,8 .align 8
	.LC4:	.lona
		0 .long
Dento, Ar	quitectura y Tecnología de Computadore	26

	1104006501
	.ident "GCC:
	(Ubuntu 7.3.0- 16ubuntu3) 7.3.0"
	.section
	.note.GNU
	-stack,"",@progbits