2º curso / 2º 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): Antonio Miguel Morillo Chica

Grupo de prácticas: D1 Fecha de entrega: 08/06/2016

Fecha evaluación en clase: 02/06/2016

Denominación de marca del chip de procesamiento o procesador (se encuentra en /proc/cpuinfo): Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz

Sistema operativo utilizado: Arch Linux

Versión de gcc utilizada: gcc (GCC) 6.1.1 20160501

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

- 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

A) MULTIPLICACIÓN DE MATRICES: CÓDIGO FUENTE: pmm-secuencial.c (ADJUNTAR CÓDIGO FUENTE AL.ZIP)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif
int main(int argc, char **argv){
  int num_filas=atoi(argv[1]);
  int num_columnas=atoi(argv[1]);
  int chunk=atoi(argv[2]);
  int **matriz1, **matriz2, **resultado;
 matriz1 = (int **)malloc(num_filas*sizeof(int*));
  matriz2 = (int **)malloc(num_filas*sizeof(int*));
  resultado=(int **)malloc(num_filas*sizeof(int*));
  int fila, columna, j;
  struct timespec cgt1,cgt2; double ncgt;
  for (fila=0;fila<num_filas;fila++)</pre>
  matriz1[fila] = (int*)malloc(num_filas*sizeof(int));
  for (fila=0;fila<num_filas;fila++)</pre>
    matriz2[fila] = (int*)malloc(num_filas*sizeof(int));
  for (fila=0;fila<num_filas;fila++)</pre>
   resultado[fila] = (int*)malloc(num_filas*sizeof(int));
    for(fila=0; fila<num filas; fila++){</pre>
    for(columna=0; columna<num_columnas; columna++){</pre>
       matriz1[fila][columna]=2;
    }
  }
  printf("Mostramos la matriz1\n");
  for(fila=0; fila<num_filas; fila++){</pre>
    for(columna=0; columna<num_columnas; columna++){</pre>
       printf(" %d ", matriz1[fila][columna] );
    printf("\n");
  for(fila=0; fila<num_filas; fila++){</pre>
    for(columna=0; columna<num_columnas; columna++){</pre>
       matriz2[fila][columna]=2;
    }
  printf("Mostramos la matriz2\n");
  for(fila=0; fila<num_filas; fila++){</pre>
    for(columna=0; columna<num_columnas; columna++){</pre>
       printf(" %d ", matriz2[fila][columna] );
    printf("\n");
  printf("Multiplicamos la matrices\n");
pmm-secuencial
  clock_gettime(CLOCK_REALTIME, &cgt1);
    for (fila = 0 ; fila < num_filas ; fila++ ){</pre>
```

```
for (columna = 0 ; columna < num_columnas; columna++ ){</pre>
        int producto = 0 ;
         for (j = 0 ; j < num\_columnas ; j++){
            producto += matriz1[fila][j] * matriz2[j][columna];
            resultado[fila][columna] = producto ;
         }
      }
  }
  clock_gettime(CLOCK_REALTIME, &cgt2);
  ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+ (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/
  printf("Tiempo(seg.):%11.9f\t",ncgt);
  printf("Mostramos el resultado\n");
  for(fila=0; fila<num_filas; fila++){</pre>
    for(columna=0; columna<num_columnas; columna++){</pre>
       printf(" %d ", resultado[fila][columna] );
    printf("\n");
  for (fila=0;fila<num filas;fila++)</pre>
   free(matriz1[fila]);
  free(matriz1);
  for (fila=0;fila<num_filas;fila++)</pre>
   free(matriz2[fila]);
  free(matriz2);
  for (fila=0;fila<num_filas;fila++)</pre>
   free(resultado[fila]);
  free(resultado);
}
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: La primera modificación ha sido muy grande, literalmente he cambiado todo el pmm-secuaencial.c de la práctica anterior. En primer lugar he cambiado la declaración de las matrices, ya no uso ningún for para reservar memoria ni las matrices son punteros a punteros, sino solo punteros, es decir, lo organizo en un vector, directame. La segunda modificación consiste en rellenar las matrices, antes las rellenaaba con un solo valor y siempre el mismo, ahora se rellenan con <u>valores</u> dependientes a las iteracción del bubhle que le corresponda. Por último he modificado el algoritmo de la multiplicar matrices, antes guardaba el resultado de la operación en una variariable y a continuación la guardaba, ahora lo hago todo en una misma sentencial.

Modificación b) –**explicación-:** Lo primero que he modificado ha sido modificar el if que comprueba si se ha podido reservar memoria. Como cada && supone un salto lo que he hecho ha sido modificarlo de estaa forma (A \parallel B \parallel C) == 0, que realiza menos saltos. Tambien he modificado el algoritmo para multiplicar matrices cuadradas, ahora el último for hace iteraciones que aumentan en cuatro, 4, 8, 12... Así en cada iteracción realizo 4 operaciones.

1.1. CÓDIGOS FUENTE MODIFICACIONES

a) pmm-secuencial-modificado_a.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char ** argv){
  int N, i, j, k, suma=0,c; int *A, *B, *C;
  struct timespec cgt1,cgt2;
  if(argc < 2){
    printf("Introduzca El numero de filas/columnas:\n");
    exit(-1);
  N = atoi(argv[1]);
  // reservamos memoria
 A = malloc(sizeof(int) * N * N);
B = malloc(sizeof(int) * N * N);
C = malloc(sizeof(int) * N * N);
  if(A == NULL || B == NULL || C == NULL){
    printf("No se ha podido reservar memoria\n");
    exit(-1);
  // inicializamos la matriz A y B
  for(i=0; i<N; ++i){
    for(j=0; j<N; ++j){
    A[i*N+j] = i+j+1;
       B[i*N+j] = i+j*2;
       C[i*N+j] = 0;
    }
  }
  // algoritmo de multiplicacion de matrices cuadradas
  clock_gettime(CLOCK_REALTIME, &cgt1);
  for(i=0; i<N; i++)
    for(j=0; j<N; j++)
for(k=0; k<N; k+=4){
         C[i*N+j] += A[i*N+k] * B[k*N+j];

C[i*N+j] += A[i*N+(k+1)] * B[(k+1)*N+j];

C[i*N+j] += A[i*N+(k+2)] * B[(k+2)*N+j];

C[i*N+j] += A[i*N+(k+3)] * B[(k+3)*N+j];
  clock_gettime(CLOCK_REALTIME, &cgt2);
       double ncgt=(double)(cgt2.tv_sec-cgt1.tv_sec) + (double) ((cgt2.tv_nsec-
cgt1.tv_nsec)/(1.e+9));
  printf("Resultado:\nC[0][0]=%d, C[%d][%d]=%d\n", C[0], N-1, N-1, C[(N-1)*N+(N-1)]);
  printf("Tiempo transcurrido en segundos: %0.6f\n", ncgt);
  // liberamos memoria
  free(A); free(B); free(C);
```

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

```
Codigo:bash—Konsole

Archivo Editar Ver Marcadores Preferencias Ayuda

[mike@looper Codigo]$ ./pmm-secuencial-modificado_a 200
Resultado:
C[0][0]=2666600, C[199][199]=30466900
Tiempo transcurrido en segundos: 0.008574
[mike@looper Codigo]$ ./pmm-secuencial-modificado_b 200
Resultado:
C[0][0]=2666600, C[199][199]=30466900
Tiempo transcurrido en segundos: 0.019194
[mike@looper Codigo]$ 

Codigo:bash
```

b) pmm-secuencial-modificado_b.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char ** argv){
  int N, i, j, k, suma=0, c; int *A, *B, *C;
  struct timespec cgt1,cgt2;
  if(argc < 2){
   printf("Introduzca El numero de filas/columnas:\n");</pre>
     exit(-1);
  N = atoi(argv[1]);
  // reservamos memoria
  A = malloc(sizeof(int) * N * N);
B = malloc(sizeof(int) * N * N);
C = malloc(sizeof(int) * N * N);
   if((A || B || C) == 0){
    printf("No se ha podido reservar memoria\n");
     exit(-1);
  // inicializamos la matriz A y B
  for(i=0; i<N; ++i){
    for(j=0; j<N; ++j){
    A[i*N+j] = i+j+1;
    B[i*N+j] = i+j*2;
       C[i*N+j] = 0;
    }
  }
  // algoritmo de multiplicacion de matrices cuadradas
  clock_gettime(CLOCK_REALTIME, &cgt1);
  for(i=0; i<N; ++i)
     for(j=0; j<N; ++j)
```

1.1. TIEMPOS:

Modificación	-O2
Sin modificar	De 2 a 10 seg. aquí
Modificación a)	0.008574
Modificación b)	0.019194

1.1. COMENTARIOS SOBRE LOS RESULTADOS: Al principio modb era más rapido pero a medid que he lo he ejjecutado unas 10 veces mostraba casi siempre tiempo superiores. Creo que es debido a que en el último bucle hay demasiadas operaciones y es por ello que le cuesta más tiempo.

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-	pmm-secuencial-
	modificado_a.s	modificado_b.s
.file "pmm-	.file "	.file "p
secuencial-modificado_a.c"	pmm-secuencial-modificado_a.c"	mm-secuencial-modificado-b.c"
.section .roda	.section .	.section .r
ta.str1.8,"aMS",@progbits,1	rodata	odata.str1.8,"aMS",@progbits,1
.align 8	.align 8	align 8
.LC0:	.LCO:	.LC0:
.string "Intr	string "	string "I.
oduzca El numero de	Introduzca El numero de	ntroduzca El numero de
filas/columnas:"	filas/columnas:"	filas/columnas:"
.align 8	.align 8	.align 8
.LC1:	.LC1:	.LC2:
.string "No	string "	.string "R
se ha podido reservar memoria"	No se ha podido reservar	esultado:\nC[0][0]=%d, C[%d]
.align 8	memoria"	[%d]=%d\n"
.LC3:	.align 8	.align 8
.string "Resu	.LC3:	.LC3:
ltado:\nC[0][0]=%d, C[%d][%d]=%d\n"	string "	string "T
.align 8	Resultado:\nC[0][0]=%d, C[%d]	iempo transcurrido en segundos:
.LC4:	[%d]=%d\n"	%0.6f\n"
.string "Tiem	align 8	.align 8
po transcurrido en segundos:	.LC4:	.LC4:
%0.6f\n"	string "	.string "N
.section .text	Tiempo transcurrido en	o se ha podido reservar memoria"
.startup,"ax",@progbits	segundos: %0.6f\n"	.section .t
.p2align 4,,15	.text	ext.startup,"ax",@progbits
.globl main	.globl	.p2align 4,,15
.type main,	main	.globl
@function	.type	main

			T		T	
main:				main, @function		.type
.LFB21:			main:			main, @function
	.cfi_startproc		.LFB2:		main:	, 5 : :
	pushq	%r15		.cfi_startproc	. LFB21:	
	•					ofi startarea
16	.cfi_def_cfa_o	iiset		pushq		.cfi_startproc
16	c: cc :			%rbp		pushq
	.cfi_offset 15			.cfi_def_cfa_off		%r15
	pushq	%r14	set 16			.cfi_def_cfa_offs
	.cfi_def_cfa_o	ffset		.cfi_offset 6,	et 16	
24			-16			.cfi_offset 15,
	.cfi_offset 14	, -24		movq	-16	
	pushq	, %r13		%rsp, %rbp		pushq
	.cfi def cfa o			.cfi_def_cfa_reg		%r14
32	.cri_ucr_cru_o	11300	ister 6	.cri_ucr_cra_rcg		.cfi_def_cfa_offs
32	ofi offoot 10	22	13161 0	odda	at 24	.cri_der_cra_orrs
	.cfi_offset 13			addq	et 24	-6: -66+ 44
	pushq	%r12		\$-128, %rsp		.cfi_offset 14,
	.cfi_def_cfa_o	ffset		movl	-24	
40				%edi, -100(%rbp)		pushq
	.cfi_offset 12	, -40		movq		%r13
	pushq	%rbp		%rsi, -112(%rbp)		.cfi_def_cfa_offs
	.cfi_def_cfa_o	ffset		movl	et 32	
48		-		\$0, -16(%rbp)		.cfi offset 13,
. •	.cfi_offset 6,	-48		cmpl	-32	
	pusha				52	nucha
	P 1	%rbx		\$1, -100(%rbp)		pushq
	.cfi_def_cfa_o	rrset		jg		%r12
56				.L2		.cfi_def_cfa_offs
	.cfi_offset 3,	-56		movl	et 40	
	subq	\$104,		\$.LCO, %edi		.cfi_offset 12,
%rsp				call	-40	
·	.cfi_def_cfa_o	ffset		puts		pushq
160				movl		%rbp
100	cmpl	\$1,		\$-1, %edi		.cfi_def_cfa_offs
%odi	cmb±	Ψ±,			et 48	u
%edi	ilo	1.00		call	CL 40	ofi offoot o
	jle	. L28		exit	10	.cfi_offset 6,
	movq		.L2:		-48	
	8(%rsi), %rdi			movq -		pushq
	movl	\$10,	112(%rbp), %ra	X		%rbx
%edx				addq		.cfi_def_cfa_offs
	xorl	%esi,		\$8, %rax	et 56	
%esi		,		movq		.cfi_offset 3,
	call			(%rax), %rax	-56	
	strtol			movq		subq
		%eax,		%rax, %rdi		•
0/510	movslq	∕₀cαx,				\$56, %rsp
%r13				call	, ,,,,	.cfi_def_cfa_offs
	movq	%rax,		atoi	et 112	_
%r14				movl		cmpl
	movq	%rax,		%eax, -20(%rbp)		\$1, %edi
40(%rsp)				movl -		jle
' '	movq	%r13,	20(%rbp), %eax			.L30
%r12	•	- /		movslq		movq
· · · · · · · · · · · · · · · · · · ·	movl	%eax,		%eax, %rdx		8(%rsi), %rdi
56(%rcn)	IIIO A T	<i>π</i> υαλ,		movl -		movl
56(%rsp)	imula	0/154.0	20(0/55-) 0/-			
04:-40	imulq	%r13,	20(%rbp), %eax			\$10, %edx
%r12				cltq		xorl
	salq	\$2,		imulq		%esi, %esi
%r12				%rdx, %rax		call
	movq	%r12,		salq		strtol
%rdi	•	,		\$2, %rax		movslq
-	call			movq		%eax, %rbp
	malloc			%rax, %rdi		movq
		0/r10		•		•
0/	movq	%r12,		call		%rax, (%rsp)
%rdi				malloc		movl
	movq	%rax,		movq		%eax, %r15d
%rbx				%rax, -32(%rbp)		movq
701 2071				movl -		%rbp, %rbx
70. 27.	call				I	• •
75. 27.			20(%rbp), %eax			ımu⊥q
	malloc	%r12	20(%rbp), %eax			imulq %rbp. %rbx
		%r12,	20(%rbp), %eax	movslq		%rbp, %rbx
%rdi	malloc	%r12, %rax,	20(%rbp), %eax			•

0/rhn			20 (0/mbm) 0/-			may/a
%rbp			20(%rbp), %eax			movq
	call			cltq		%rbx, %rdi
	malloc			imulq		call
	testq	%rbx,		%rdx, %rax		malloc
%rbx				salq		movq
	movq	%rax,		\$2, %rax		%rbx, %rdi
%rdi		,		movq		movq
/01 UI	movq	%rax,		%rax, %rdi		%rax, %r14
40/%rcn)	illovq	701 ax,				·
48(%rsp)		07-17		call		call
	sete	%dl		malloc		malloc
	testq	%rbp,		movq		movq
%rbp				%rax, -40(%rbp)		%rbx, %rdi
	sete	%al		movl -		movq
	orb	%al,	20(%rbp), %eax			%rax, %r12
%dl				movslq		call
	jne	.L3		%eax, %rdx		malloc
	testq	%rdi,		movl -		testq
%rdi	ccscq	701 UI,	20(%rbp), %eax			%r14, %r14
/oi U1	io		20(%ibp), %eax			·
	je	.L3		cltq		movq
	testl			imulq		%rax, %r13
	%r14d, %r14d			%rdx, %rax		setne
	jle	.L29		salq		%dl
	movl			\$2, %rax		testq
	40(%rsp), %edi			movq		%r12, %r12
	leaq	0(,		%rax, %rdi		setne
%r13,4), %rax	•	` '		call		%al
	movq			malloc		orb
	48(%rsp), %r9			movq		%al, %dl
				%rax, -48(%rbp)		
	movl	J		, , , ,		jne
	56(%rsp), %r100			cmpq		.L15
	xorl	%edx,		\$0, -32(%rbp)		testq
%edx				je		%r13, %r13
	movq	%rax,		.L3		je
24(%rsp)				cmpq		.L3
	movq	%rax,		\$0, -40(%rbp)	.L15:	
%r11	·	•		je		movl
	leal			.L3		(%rsp), %eax
	(%rdi,%rdi), %ı	r8d		cmpq		testl
	xorl	%edi,		\$0, -48(%rbp)		%eax, %eax
%edi	XULT	∞eu⊥,				jle
				jne		•
.L7:				. L4		.L31
	leal		.L3:	_		movl
	1(%rdx), %r12d			movl		(%rsp), %eax
	leal			\$.LC1, %edi		salq
	(%rdx,%r8), %es	si		call		\$2, %rbp
	movq	%rdi,		puts		xorl
%rax	•	,		movl		%edi, %edi
	movl			\$-1, %edi		xorl
	%r12d, %ecx			call		%edx, %edx
	.p2align 4,,10			exit		leal
	.p2align 3		.L4:		1.0	(%rax,%rax), %r8d
.L6:	-			movl	.L8:	
	movl	%edx,		\$0, -4(%rbp)		leal
0(%rbp,%rax)				jmp		1(%rdx), %ebx
	addl	\$2,		.L5		leal
%edx			.L8:			(%rdx,%r8), %esi
	movl	%ecx,		movl		movq
(%rbx,%rax)	-	,		\$0, -8(%rbp)		%rdi, %rax
(,,	movl	\$0,		jmp		mov1
(%r9,%rax)	111O A T	ΨΟ,		۱۱۱۱۲ L6		%ebx, %ecx
(101 5, 101 ax)	add1	Φ1	17.	. 20		
0/	addl	\$1,	.L7:			.p2align 4,,10
%ecx				movl -		.p2align 3
	addq	\$4,	4(%rbp), %eax		.L7:	
%rax				imull -		movl
	cmpl	%edx,	20(%rbp), %eax			%edx, (%r12,%rax)
%esi		,		movl		addl
	jne	.L6		%eax, %edx		\$2, %edx
	addq			movl -		movl
%rdi	auuq	%r11,	0(%/rhn) %000	1110 A T		
%rdi			8(%rbp), %eax			%ecx, (%r14,%rax)
0						

	cmpl %r10d, %r12d			addl		movl
	•					
				%edx, %eax		\$0, 0(%r13,%rax)
	movl			cltq		addl
	%r12d, %edx			leaq		\$1, %ecx
i e	jne	. L7		0(,%rax,4), %rdx		addq
	leaq			movq -		\$4, %rax
	64(%rsp), %rsi		32(%rbp), %rax			cmpl
	xorl	%edi,		addq		%edx, %esi
%edi				%rdx, %rax		jne
	movq	%rbx,		movl -		. L7
%r13	_		4(%rbp), %ecx	_		addq
	xorl			movl -		%rbp, %rdi
	%r14d, %r14d		8(%rbp), %edx			cmpl
	call			addl		%r15d, %ebx
	clock_gettime			%ecx, %edx		movl
	movl			addl		%ebx, %edx
	40(%rsp), %eax			\$1, %edx		jne
	leal			movl		.L8
	(%r12,%r12), %			%edx, (%rax)		leaq
	leal	0(,		movl -		16(%rsp), %rsi
%r12,4), %r10d			4(%rbp), %eax			xorl
	movq		20(0/565) 0/-	imull -		%edi, %edi
	48(%rsp), %r15		20(%rbp), %eax	mo.v1		call
	movslq			movl		clock_gettime
	%r12d, %rdi	ው የ		%eax, %edx		movq %r12 %r10
	movl	\$0,	0(0(100) 0(100)	movl -		%r13, %r10
(%rsp)	maya]a		8(%rbp), %eax	add1		movq
	movslq			addl		%r14, %rdi
	%r10d, %r10	% odv		%edx, %eax cltq		xorl
 %r11	movslq	%edx,			112.	%r11d, %r11d
	addl			leaq 0(,%rax,4), %rdx	.L13:	mova
				** **		movq %r12 %r0
	%r12d, %edx subl	\$1,	40(%rbp), %rax	movq -		%r12, %r9 xorl
%eax	Subi	Ψ1,	40(%1 bp), %1 ax	addq		%r8d, %r8d
	salq	\$2,		%rdx, %rax		.p2align 4,,10
%r10	3414	ΨΖ,		movl -		.p2align 3
	movslq	%edx,	8(%rbp), %edx	MOVI	.L11:	.pzarryn 5
%r9	mov3±q	λισαχ,	0(701 bp), 70cux	leal		movl
	movl	%eax,		(%rdx,%rdx),		(%r10,%r8,4),
60(%rsp)		πουπή	%ecx	(MI anymi any)	%esi	(701 10 / 701 0 / 1 / /
, , ,	shrl	\$2,	/0COX	movl -	7,0001	movq
%eax	J	Ψ=/	4(%rbp), %edx			%r9, %rcx
	movq	%rbx,	1(701 bp)) 70cux	addl		xorl
8(%rsp)		701 BX 7		%ecx, %edx		%eax, %eax
	leaq	4(,		movl		.p2align 4,,10
%rax,4), %rax	2009	.(/		%edx, (%rax)		.p2align 3
	movq	%rbp,		movl -	.L10:	. 1, 9 0
16(%rsp)		,	4(%rbp), %eax			movl
, , ,	movq	%rax,	, ,	imull -		(%rdi,%rax,4),
32(%rsp)	•	,	20(%rbp), %eax		%edx	, , , , , , ,
.L12:			, ,	movl		addq
	movq			%eax, %edx		\$1, %rax
	32(%rsp), %rax			movl -		imull
	movq		8(%rbp), %eax			(%rcx), %edx
	8(%rsp), %rbx			addl		àddq
	movq			%edx, %eax		%rbp, %rcx
	16(%rsp), %rbp			cltq		addl
	addq	%r14,		leaq		%edx, %esi
%rax				0(,%rax,4), %rdx		cmpl
	leaq			movq -		%eax, %ebx
	(%rbx,%rax,4),	%r8	48(%rbp), %rax			jg
	xorl	%ebx,		addq		.L10
%ebx				%rdx, %rax		movl
	.p2align 4,,10			movl		%esi,
	.p2align 3			\$0, (%rax)	(%r10,%r8,4)	
.L10:				addl		addq
	movl			\$1, -8(%rbp)		\$1, %r8
i e	(%r15,%rbx,4),	%eax	.L6:			addq
			i .		I	

	movq	%rbp,		movl -		\$4, %r9
%rcx			8(%rbp), %eax			cmpl
	movq	%r13,		cmpl -		%r8d, %ebx
%rdx	0.11		20(%rbp), %eax	• •		jg
	.p2align 4,,10			jl		. L11
10.	.p2align 3			.L7		addl
.L9:	movl			addl \$1, -4(%rbp)		\$1, %r11d addg
	(%rdx), %esi		.L5:	Φ1, -4(%IDP)		%rbp, %r10
	addq	\$16,		movl -		addq
%rdx	aaaq	Ψ±0/	4(%rbp), %eax	110 4 1		%rbp, %rdi
	imull		(cmpl -		cmpl
	(%rcx), %esi		20(%rbp), %eax	·		%r15d, %r11d
	addl	%esi,		jl		jne
%eax				.L8		.L13
	movl	-		leaq -	.L14:	
12(%rdx), %esi			80(%rbp), %rax			leaq
	imull			movq		32(%rsp), %rsi
	(%rcx,%rdi,4),			%rax, %rsi		xorl
0/ooi	addl	%eax,		movl \$0, %edi		%edi, %edi
%esi	movl			call		call clock_gettime
8(%rdx), %eax	III O A T	_		clock_gettime		movq
o(/orux), /oedx	imull			movl		40(%rsp), %rax
	(%rcx,%r11,4),	%eax		\$0, -4(%rbp)		subq
	addl	%esi,		jmp		24(%rsp), %rax
%eax		,		.L9		movl
	movl	-	.L14:			\$.LC2, %edi
4(%rdx), %esi				movl		pxor
	imull			\$0, -8(%rbp)		%xmm0, %xmm0
	(%rcx,%r9,4),	%esi		jmp		movl
	addq	%r10,		.L10		0(%r13), %esi
%rcx			.L13:	-		pxor
0/	addl	%esi,		movl		%xmm1, %xmm1
%eax	omn a	0/redy		\$0, -12(%rbp)		cvtsi2sdq
0/50	cmpq	%rdx,		jmp		%rax, %xmm0
%r8	jne	.L9	.L12:	.L11		movq 32(%rsp), %rax
	movl	%eax,		movl -		subq
(%r15,%rbx,4)		ποαπη	4(%rbp), %eax	110 4 1		16(%rsp), %rax
	addq	\$1,	(imull -		cvtsi2sdq
%rbx	·	•	20(%rbp), %eax			%rax, %xmm1
	addq	\$4,		movl		movq
%rbp				%eax, %edx		(%rsp), %rax
	cmpl	%ebx,		movl -		leal -
%r12d			8(%rbp), %eax		1(%rax), %edx	
	jg	.L10		addl		addl
	movq			%edx, %eax		\$1, %eax
	24(%rsp), %rbx	Φ 1		cltq		imull
(%ren)	addl	\$1,		leaq		%edx, %eax
(%rsp)	addq	%rdi,		0(,%rax,4), %rdx movq -		movl %edx, %ecx
%r14	addy	/or u⊥,	48(%rbp), %rax			divsd
, VI ± 1	movl		(addg		.LC1(%rip), %xmm0
	(%rsp), %eax			%rax, %rdx		cltq
	addq	%rbx,		movl -		movl
%r15	•	,	4(%rbp), %eax			0(%r13,%rax,4),
	addq	%rbx,		imull -	%r8d	. ,.
%r13			20(%rbp), %eax			xorl
	cmpl			movl		%eax, %eax
	56(%rsp), %eax			%eax, %ecx		addsd
	jne	.L12	0.00 1)	movl -		%xmm1, %xmm0
	movq		8(%rbp), %eax	- d d 1		movsd
	8(%rsp), %rbx			addl		%xmm0, 8(%rsp)
	movq			%ecx, %eax		call
.L13:	16(%rsp), %rbp			cltq leaq		printf movsd
	leaq			0(,%rax,4), %rcx		8(%rsp), %xmm0
	80(%rsp), %rsi			movq -		movl
	(- · -
10			<u> </u>		L	

		0/- !!	40 (0/-1) ::			# 100 ° 1
%edi	xorl	%edi,	48(%rbp), %rax	addq		<pre>\$.LC3, %edi mov1</pre>
%eui	call			%rcx, %rax		\$1, %eax
	clock_gettime			movl		call
	movq			(%rax), %ecx		printf
	88(%rsp), %rax			movl -		movq
	subq		4(%rbp), %eax			%r14, %rdi
	72(%rsp), %rax		20(%rhn) %00v	imull -		call free
	pxor %xmm0, %xmm0		20(%rbp), %eax	movl		movq
	movl			%eax, %esi		%r12, %rdi
	60(%rsp), %edi			movl -		call
	movq		12(%rbp), %eax			free
	48(%rsp), %r14			addl		movq
	pxor %xmm1, %xmm1			%esi, %eax cltq		%r13, %rdi call
	cvtsi2sdq	%rax,		leag		free
%×mm0		,		0(,%rax,4), %rsi		addq
	movq			movq -		\$56, %rsp
	80(%rsp), %rax		32(%rbp), %rax			.cfi_remember_sta
	subq			addq	te	ofi dof ofo offo
	64(%rsp), %rax movl			%rsi, %rax movl	et 56	.cfi_def_cfa_offs
	(%r14), %esi			(%rax), %esi		xorl
	movl	%edi,		movl -		%eax, %eax
%ecx	_		12(%rbp), %eax			popq
0/ = d	movl	%edi,	20(0/262) 0/224	imull -		%rbx
%edx	cvtsi2sdq	%rax,	20(%rbp), %eax	movl	et 48	.cfi_def_cfa_offs
%xmm1	CVCSIZSUQ	701 U.X.,		%eax, %edi	CC 40	popq
	movl			movl -		%rbp
	40(%rsp), %eax		8(%rbp), %eax			.cfi_def_cfa_offs
0/	addl	\$1,		addl	et 40	
%eax	imull	%edi,		%edi, %eax cltq		popq %r12
%eax	IMUII	70CUI,		leag		.cfi_def_cfa_offs
	movl			0(,%rax,4), %rdi	et 32	
	\$.LC3, %edi			movq -		popq
	divsd	nmO	40(%rbp), %rax	adda		%r13
	.LC2(%rip), %xı cltq	IIIIO		addq %rdi, %rax	et 24	.cfi_def_cfa_offs
	movl			movl	00 21	popq
	(%r14,%rax,4),	%r8d		(%rax), %eax		%r14
	xorl	%eax,		imull		.cfi_def_cfa_offs
%eax	م ما ما م ما			%esi, %eax	et 16	
	addsd %xmm1, %xmm0			addl %ecx, %eax		popq %r15
	movsd			movl		.cfi_def_cfa_offs
	%xmm0, (%rsp)			%eax, (%rdx)	et 8	
	call			movl -		ret
	printf movsd		4(%rbp), %eax	imull -	.L3:	.cfi_restore_stat
	(%rsp), %xmm0		20(%rbp), %eax		e	. 51 1_1 53 101 5_5141
	mov1		- (Sp), woux	movl	_	movl
	\$.LC4, %edi			%eax, %edx		\$.LC4, %edi
0/	movl	\$1,	0 (0/12) -:-	movl -		call
%eax	call		8(%rbp), %eax	addl		puts orl \$-
	printf			%edx, %eax	1, %edi	υ. <u>-</u> φ-
	movq	%rbx,		cltq		call
%rdi		_		leaq	_	exit
	call	free %rbp		0(,%rax,4), %rdx	.L31:	loog
%rdi	movq	%rbp,	48(%rbp), %rax	movq -		leaq 16(%rsp), %rsi
707 G.I	call	free	(addq		xorl
	movq	%r14,		%rax, %rdx		%edi, %edi
%rdi		_		movl -		call
	call	free	4(%rbp), %eax	imull		clock_gettime
	addq	\$104,		imull -		jmp
			<u> </u>			

%rsp			20(%rbp), %eax			.L14	
701 30	.cfi_remember_s	state	20(1010)) 10001	movl	.L30:		
	.cfi_def_cfa_o			%eax, %ecx		movl	
56				movl -		\$.LC0, %edi	
	xorl	%eax,	8(%rbp), %eax			call	
%eax				addl		puts	
	popq	%rbx		%ecx, %eax	الم الم	orl	\$-
40	.cfi_def_cfa_o	rrset		cltq	1, %edi	0011	
48	popq	%rbp		leaq 0(,%rax,4), %rcx		call exit	
	.cfi_def_cfa_o			movq -		.cfi_endproc	
40			48(%rbp), %rax		.LFE21:	<u>_</u>	
	popq	%r12	, ,	addq		.size	
	.cfi_def_cfa_o	ffset		%rcx, %rax		main,main	
32				movl		.section	.r
	popq	%r13		(%rax), %ecx	odata.cst8,"aM'		
	.cfi_def_cfa_o	ffset	4 (0(-1)	movl -	104	.align 8	
24	nong	0/511	4(%rbp), %eax	imull -	.LC1:	long	0
	popq .cfi_def_cfa_o	%r14 ffcat	20(%rbp), %eax	TIIIUIII -		.long .long	0
16	.cri_uer_cra_o	11361	20(%1 bp), %eax	movl -		1104006501	
10	popq	%r15	12(%rbp), %esi	IIIO V I		.ident	"G
	.cfi_def_cfa_o		=(56), 70001	addl	CC: (GNU) 6.1.1		-
8				\$1, %esi		.section	.n
	ret			addl	ote.GNU-stack,'	'",@progbits	
.L29:				%esi, %eax			
	.cfi_restore_s	tate		cltq			
	leaq			leaq			
	64(%rsp), %rsi	%odi		0(,%rax,4), %rsi			
%edi	xorl	%edi,	32(%rbp), %rax	movq -			
70EUI	call		32(%1 bp), %1 ax	addq			
	clock_gettime			%rsi, %rax			
	movq			movl			
	40(%rsp), %rax			(%rax), %esi			
	subl	\$1,		movl -			
%eax	_		12(%rbp), %eax				
CO (0/222)	movl	%eax,		addl			
60(%rsp)	jmp	.L13		\$1, %eax imull -			
.L3:	Jilip	. L13	20(%rbp), %eax	IIIUII -			
	movl		20(201 50), 20042	movl			
	\$.LC1, %edi			%eax, %edi			
	call	puts		movl -			
	orl	\$-1,	8(%rbp), %eax				
%edi				addl			
1.00	call	exit		%edi, %eax			
.L28:	movil			cltq			
	mo∨l \$.LCO, %edi			leaq 0(,%rax,4), %rdi			
	call	puts		movq -			
	orl	\$-1,	40(%rbp), %rax	٧ ٩			
%edi		,	(= === // === \	addq			
	call	exit		%rdi, %rax			
	.cfi_endproc			movl			
.LFE21:				(%rax), %eax			
	.size	main,		imull			
main	cootica	rodo		%esi, %eax			
ta.cst8,"aM",@	.section nroahits 8	.roda		addl %ecx, %eax			
ca.csco, an ,@	.align 8			movl			
.LC2:				%eax, (%rdx)			
	.long	Θ		movl -			
	.long		4(%rbp), %eax				
	1104006501			imull -			
	.ident	"GCC:	20(%rbp), %eax	_			
(GNU) 6.1.1 20				movl			
.GNU-stack,"",	.section	.note		%eax, %edx			
. UNU-SLACK, "",	⊕hι ∩Anτr2			movl -			
			<u> </u>				

```
8(%rbp), %eax
               addl
               %edx, %eax
               cltq
               leaq
               0(,%rax,4), %rdx
               movq
48(%rbp), %rax
               addq
               %rax, %rdx
               movl
4(%rbp), %eax
               imull
20(%rbp), %eax
               movl
               %eax, %ecx
               movl
8(%rbp), %eax
               addl
               %ecx, %eax
               cltq
               leaq
               0(,%rax,4), %rcx
               movq
48(%rbp), %rax
               addq
               %rcx, %rax
               movl
               (%rax), %ecx
               movl
4(%rbp), %eax
               imull
20(%rbp), %eax
               movl
12(%rbp), %esi
               addl
               $2, %esi
               addl
               %esi, %eax
               cltq
               leaq
               0(,%rax,4), %rsi
               movq
32(%rbp), %rax
               addq
               %rsi, %rax
               movl
               (%rax), %esi
               movl
12(%rbp), %eax
               addl
               $2, %eax
               imull
20(%rbp), %eax
               movl
               %eax, %edi
               movl
8(%rbp), %eax
               addl
               %edi, %eax
               cltq
               leaq
               0(,%rax,4), %rdi
               movq
40(%rbp), %rax
               addq
               %rdi, %rax
               movl
               (%rax), %eax
```

```
imull
               %esi, %eax
               addl
               %ecx, %eax
               movl
               %eax, (%rdx)
               movl
4(%rbp), %eax
               imull
20(%rbp), %eax
               movl
               %eax, %edx
               movl
8(%rbp), %eax
               addl
               %edx, %eax
               cltq
               leaq
               0(,%rax,4), %rdx
               movq
48(%rbp), %rax
               addq
               %rax, %rdx
               movl
4(%rbp), %eax
               imull
20(%rbp), %eax
               movl
               %eax, %ecx
               movl
8(%rbp), %eax
               addl
               %ecx, %eax
               cltq
               leaq
               0(,%rax,4), %rcx
               movq
48(%rbp), %rax
               addq
               %rcx, %rax
               movl
               (%rax), %ecx
               movl
4(%rbp), %eax
               imull
20(%rbp), %eax
               movl
12(%rbp), %esi
               addl
               $3, %esi
               addl
               %esi, %eax
               cltq
               leaq
               0(,%rax,4), %rsi
               movq
32(%rbp), %rax
               addq
               %rsi, %rax
               movl
               (%rax), %esi
               movl
12(%rbp), %eax
               addl
               $3, %eax
               imull
20(%rbp), %eax
               movl
               %eax, %edi
```

```
movl
8(%rbp), %eax
               addl
               %edi, %eax
               cltq
               leaq
               0(,%rax,4), %rdi
               movq
40(%rbp), %rax
               addq
               %rdi, %rax
               movl
               (%rax), %eax
               imull
               %esi, %eax
               addl
               %ecx, %eax
               movl
               %eax, (%rdx)
               addl
               $4, -12(%rbp)
.L11:
               movl
12(%rbp), %eax
               cmpl
20(%rbp), %eax
               jl
                .L12
               addl
               $1, -8(%rbp)
.L10:
               movl
8(%rbp), %eax
               cmpl
20(%rbp), %eax
               jl
                .L13
               addl
               $1, -4(%rbp)
.L9:
               movl
4(%rbp), %eax
               cmpl
20(%rbp), %eax
               jl
                .L14
               leaq
96(%rbp), %rax
               movq
               %rax, %rsi
               movl
               $0, %edi
               call
               {\tt clock\_gettime}
               movq
96(%rbp), %rdx
               movq
80(%rbp), %rax
               subq
               %rax, %rdx
               movq
               %rdx, %rax
               pxor
               %xmm1, %xmm1
               cvtsi2sdq
               %rax, %xmm1
               movq
88(%rbp), %rdx
               movq
```

```
72(%rbp), %rax
               subq
               %rax, %rdx
               movq
               %rdx, %rax
               pxor
               %xmm0, %xmm0
               cvtsi2sdq
               %rax, %xmm0
               movsd
               .LC2(%rip),
%xmm2
               divsd
               %xmm2, %xmm0
               addsd
               %xmm1, %xmm0
               movsd
               %xmm0, -56(%rbp)
               movl
20(%rbp), %eax
               leal
               1(%rax), %edx
               movl
20(%rbp), %eax
               subl
               $1, %eax
               imull
               %edx, %eax
               cltq
               leaq
               0(,%rax,4), %rdx
               movq
48(%rbp), %rax
               addq
               %rdx, %rax
               movl
               (%rax), %esi
               movl
20(%rbp), %eax
               leal
1(%rax), %ecx
               movl
20(%rbp), %eax
               leal
1(%rax), %edx
               movq
48(%rbp), %rax
               movl
               (%rax), %eax
               movl
               %esi, %r8d
               movl
               %eax, %esi
               movl
               $.LC3, %edi
               movl
               $0, %eax
               call
               printf
               movq
56(%rbp), %rax
               movq
               %rax, -120(%rbp)
               movsd
120(%rbp), %xmm0
               movl
               $.LC4, %edi
               movl
               $1, %eax
```

```
call
               printf
               movq
32(%rbp), %rax
               movq
               %rax, %rdi
               call
               free
               movq
40(%rbp), %rax
               movq
               %rax, %rdi
               call
               free
               movq
48(%rbp), %rax
               movq
               %rax, %rdi
               call
               free
               movl
               $0, %eax
               leave
               .cfi_def_cfa 7,
8
               ret
               .cfi_endproc
.LFE2:
               .size
               main, .-main
               .section
rodata
               .align 8
.LC2:
               .long
                               0
                .long
               1104006501
               .ident
GCC: (GNU) 6.1.1 20160501"
                .section
note.GNU-stack,"",@progbits
```

B) CÓDIGO FIGURA 1:

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

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#ifdef _OPENMP
        #include <omp.h>
#else
        #define omp_get_thread_num() 0
#endif
struct {
        int a;
        int b;
} s[5000];
main(){
        int i, ii;
        int X1, X2, *R;
double t1, t2, tiempo;
R = (int*)malloc(40000*sizeof(int));
        // Inicialización R,S
        for (i=0; i<40000; i++)
                R[i] = 0;
        for (i=0; i<5000; i++){
```

1.1. MODIFICACIONES REALIZADAS (al menos dos modificaciones):

Modificación a) –explicación-: El fichero original utiliza dos bucles para calcular independiente X1 y X2, en la primera modificación reduzco uno de los for pero dejo el mismo número de iteraciones.

Modificación b) –explicación-: La segunda modificación es dividir los calculos en dos for para hacer menos operaciones. Por un lado a y por otro b.

Modificación c) –explicación-: La última es reorganizar, he generado dos for nuevos pese a haberlo eliminado en el apartado a. De esta manera me ahorro iteraciones y realizo más calculos por iteración.

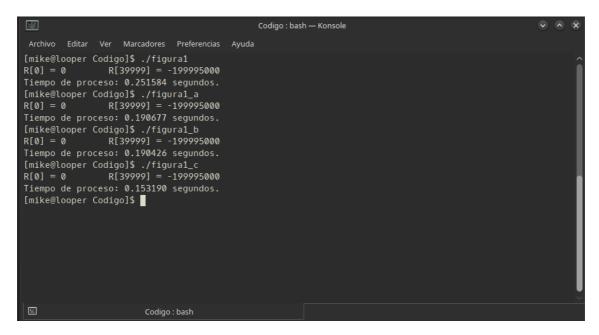
1.1. CÓDIGOS FUENTE MODIFICACIONES

a) figura1-modificado_a.c

(ADJUNTAR CÓDIGO FUENTE AL .ZIP)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#ifdef _OPENMP
        #include <omp.h>
#else
        #define omp_get_thread_num() 0
#endif
struct {
        int a;
       int b;
} s[5000];
main(){
        int i, ii;
       int X1, X2, *R;
double t1, t2, tiempo;
       R = (int*)malloc(40000*sizeof(int));
        // Inicialización R,S
        for (i=0; i<40000; i++)
               R[i] = 0;
        for (i=0; i<5000; i++){
               s[i].a = 0;
               s[i].b = 0;
        t1 = omp_get_wtime();
        // Cálculo del algoritmo
        for (ii=0 ; ii<40000 ; ii++){
               X1=0; X2=0;
```

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



b) figura1-modificado_b.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#ifdef _OPENMP
         #include <omp.h>
#else
         #define omp_get_thread_num() 0
#endif
struct {
         int a;
         int b;
} s[5000];
main(){
         int i, ii;
         int X1, X2, *R;
double t1, t2, tiempo;
R = (int*)malloc(40000*sizeof(int));
         // Inicialización R,S
         for (i=0 ; i<40000 ; i++)
         R[i] = 0;
for (i=0; i<5000; i++){
    s[i].a = 0;
    s[i].b = 0;
```

c) figura1-modificado_c.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#ifdef _OPENMP
        #include <omp.h>
#else
       #define omp_get_thread_num() 0
#endif
struct {
        int a;
       int b;
} s[5000];
main(){
        int i, ii;
       int X1, X2, *R;
double t1, t2, tiempo;
        R = (int*)malloc(40000*sizeof(int));
        // Inicialización R,S
        for (i=0 ; i<40000 ; i++)
               R[i] = 0;
        for (i=0; i<5000; i++){
               s[i].a = 0;
               s[i].b = 0;
        t1 = omp_get_wtime();
        // Cálculo del algoritmo
        for (ii=0 ; ii<40000 ; ii++){
               X1=0; X2=0;
               X1 += 2*s[i+1].a + ii;
                       X2 += 3*s[i].b - ii;
                       X2 += 3*s[i+1].b - ii;
                if (X1<X2)
                       R[ii] = X1;
               else R[ii] = X2;
       t2 = omp_get_wtime();
tiempo = t2-t1;
printf("R[0] = %d\t", R[0]);
        printf("R[39999] = %d", R[39999]);
```

```
printf("\nTiempo de proceso: %f segundos.\n", tiempo);
    free(R);
}
```

1.1. TIEMPOS:

Modificación	-O2
Sin modificar	0,251584
Modificación a)	0.190677
Modificación b)	0.190426
Modificación c)	0.153190

1.1. COMENTARIOS SOBRE LOS RESULTADOS: Como vemos en los resultados las optimizaciones han resultado efectivas.

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

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 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.

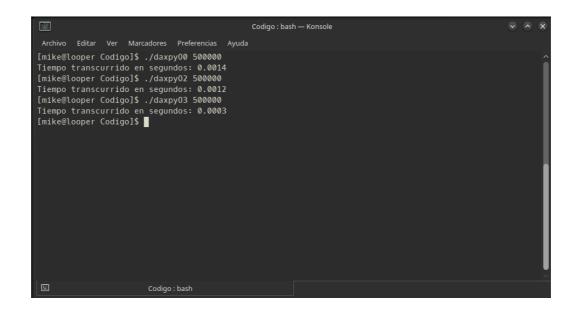
CÓDIGO FUENTE: daxpy.c

(ADJUNTAR CÓDIGO FUENTE AL .ZIP)

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(int argc, char ** argv)
      struct timespec cgt1, cgt2;
      int N, a=3, i;
      int *y, *x;
      if(argc < 2){
             printf("Introduzca El tamaño N:\n");
             exit(-1);
      }
      N=atoi(argv[1]);
      y = malloc(sizeof(int) * N+1);
      x = malloc(sizeof(int) * N+1);
      if(y == NULL || x == NULL){
    printf("No se ha podido reservar memoria\n");
             exit(-1);
      }
      // inicializacion
      for(i=1; i<=N; ++i){
             x[i]=i+1;
             y[i]=i+2;
      }
      // daxpy
      clock_gettime(CLOCK_REALTIME, &cgt1);
      for (i=1; i<=N; i++)
             y[i] += a*x[i];
      clock_gettime(CLOCK_REALTIME,&cgt2);
      double
               ncgt=(double)(cgt2.tv_sec-cgt1.tv_sec) + (double)
((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
      printf("Tiempo transcurrido en segundos: %0.4f\n", ncgt);
      return 0;
```

Tiempes eies	-O0	-O2	-O3
Tiempos ejec.	0,0014	0,0012	0,0003

CAPTURAS DE PANTALLA:



COMENTARIOS SOBRE LAS DIFERENCIAS EN ENSAMBLADOR: Ninguno.

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		daxı	oy02.s		daxpy	03.s		
	.file	"d		.file	"daxpy		.file	"d
axpy.c"			.c"			axpy.c"		
	.section	.r		.section	.rodat		.section	.r
odata			a.str1.1,"aM	IS",@progbits,1		odata.str1.1,"	aMS",@progbit	s,1
.LC0:			.LC0:			.LC0:		
	.string	"I		.string	"Intro		.string	"I
ntroduzca El	tama\303\2610	N:"	duzca El tam	na\303\261o N:"		ntroduzca El t	ama\303\261o	N:"
	.align 8			.section	.rodat		.section	.r
.LC1:			a.str1.8,"aM	IS",@progbits,1		odata.str1.8,"	aMS",@progbit	s,1
	.string	"N		.align 8			.align 8	
o se ha podi	do reservar me	moria"	.LC1:			.LC1:		
	.align 8			.string	"No se		.string	"N
.LC3:			ha podido re	eservar memoria"		o se ha podido	reservar mem	oria"
	.string	"T		.align 8			.align 8	
iempo transc	urrido en segu	ındos:	.LC3:			.LC7:		
%0.4f\n"				.string	"Tiemp		.string	"Т
	.text		o transcurri	.do en segundos:	%0.4f\n"		rido en segun	dos:
	.globl			.section	.text.	%0.4f\n"		
	main		startup,"ax"	,@progbits			.section	.t
	.type			.p2align 4,,1	15	ext.startup,"a	x",@progbits	
	main, @func	tion		.globl	main		.p2align 4,,	15
main:				.type	main,		.globl	
.LFB2:			@function				main	
	.cfi_startp	roc	main:				.type	
	pushq		.LFB21:				main, @funct	ion
	%rbp			.cfi_startpro	С	main:		
	.cfi_def_cf	a_offs		pushq	%r13	.LFB21:		
et 16				.cfi_def_cfa_	_offset		.cfi_startpr	ос
	.cfi_offset	6,	16				pushq	
-16				.cfi_offset 1	•		%r12	
	movq			pushq	%r12		.cfi_def_cfa	_offs
	%rsp, %rbp			.cfi_def_cfa_	_offset	et 16		

	.cfi_def_cfa_regi	24				.cfi_offset 12,
ster 6	0		.cfi_offset 12	24	-16	_ ,
500. 0	subg		pusha	%rbp	10	pushq
	•		.cfi_def_cfa_o			
	\$112, %rsp	00	.cri_uer_cra_o	11361		%rbp
	movl	32				.cfi_def_cfa_offs
	%edi, -84(%rbp)		.cfi_offset 6,	-32	et 24	
	movq		pushq	%rbx		.cfi_offset 6,
	%rsi, -96(%rbp)		.cfi_def_cfa_o	ffset	-24	
	movl	40				pushq
	\$3, -8(%rbp)		.cfi_offset 3,	-40		%rbx
	cmpl		subq	\$40,		.cfi_def_cfa_offs
	•	0/10010	Subq	Φ40,	-+ 00	.cri_uer_cra_orrs
	\$1, -84(%rbp)	%rsp			et 32	
	jg		.cfi_def_cfa_o	ffset		.cfi_offset 3,
	.L2	80			-32	
	movl		cmpl	\$1,		subq
	\$.LCO, %edi	%edi				\$48, %rsp
	call		jle	.L20		.cfi_def_cfa_offs
	puts		movq	1220	et 80	1011_001_010_0110
			•		er 00	1
	movl \$-		8(%rsi), %rdi			cmpl
1, %edi			movl	\$10,		\$1, %edi
	call	%edx				jle
	exit		xorl	%esi,		.L46
.L2:		%esi		,		movq
	movq -		call	strtol		8(%rsi), %rdi
06(%rhn) %rax						
96(%rbp), %ra>		0/110	movq	%rax,		mov1
	addq	%r13	_			\$10, %edx
	\$8, %rax		movl	%eax,		xorl
	movq	%r12d				%esi, %esi
	(%rax), %rax		cltq			call
	movq		leaq	1(,		strtol
	%rax, %rdi	%rax,4), %rbp		\ /		movq
	call	, , un, + /, , , , oi bp	mova	%rbp,		%rax, %rbx
		0/rdi	movq	∕₀ı nh,		
	atoi	%rdi				cltq
	movl		call	malloc		leaq
	%eax, -12(%rbp)		movq	%rbp,		1(,%rax,4), %r12
	movl -	%rdi				movq
12(%rbp), %ea>	<		movq	%rax,		%r12, %rdi
(cltq	%rbx		,		call
	•	701 67	0011	malloc		malloc
	salq		call			
	\$2, %rax		testq	%rbx,		movq
	addq	%rbx				%r12, %rdi
	\$1, %rax		movq	%rax,		movq
	movq	%rbp				%rax, %rbp
	%rax, %rdi		je	.L3		call
	call		testq	%rax,		malloc
	malloc	%rav	20024	701 U.A.		
		%rax	io			testq %rbn %rbn
	movq		je	.L3		%rbp, %rbp
	%rax, -24(%rbp)		testl	%r13d,		movq
	movl -	%r13d				%rax, %r12
12(%rbp), %ea>	(leal			je
	cltq		3(%r13), %ecx			.L3
	salq		mov1	\$3,		testq
		%ody	v T	ΨΟ,		•
	\$2, %rax	%edx		Φ0		%rax, %rax
	addq		movl	\$2,		je
	\$1, %rax	%eax				.L3
	movq		jle	.L21		testl
	%rax, %rdi		.p2align 4,,10			%ebx, %ebx
	call		.p2align 3			jle
	malloc	.L10:	, <u>-</u> g •			.L47
			mov1	%edx,		_
	movq	4 (0/12/2000	movl	∕₀cux,		leaq
	%rax, -32(%rbp)	-4(%rbx,%rax,4	,			4(%rax), %rax
	cmpq		addl	\$1,		shrq
	\$0, -24(%rbp)	%edx				\$2, %rax
	je		movl	%eax,		negq
	.L3	-4(%rbp,%rax,4		•		%rax
	cmpq	Σρ, οι αλ, τ	addq	\$1,		andl
		%ray	uuuq	Ψ±,		
	\$0, -32(%rbp)	%rax	a	٠ - الم		\$3, %eax
	jne		cmpl	%edx,		cmpl
	. L4	%ecx				%ebx, %eax

.L3:			jne	.L10		cmova
	movl		movq	%rsp,		%ebx, %eax
	\$.LC1, %edi	%rsi	•			cmpl
	call		xorl	%edi,		\$6, %ebx
	puts	%edi	X01 1	/0CU1/		cmovbe
	movl \$-	70CU1	call			%ebx, %eax
d 0/adå	1110.47 20-					•
1, %edi			clock_gettime			testl
	call		movl	\$1,		%eax, %eax
	exit	%eax				je
.L4:			.p2align 4,,10			. L24
	movl		.p2align 3			cmpl
	\$1, -4(%rbp)	.L8:				\$1, %eax
	jmp		movl			movl
	.L5		0(%rbp,%rax,4),	. %edx		\$2, 4(%r12)
.L6:			leal	(%rdx,		movl
1201	movl -	%rdx,2), %edx	1041	(701 47.7)		\$3, 4(%rbp)
1/0/rhn) 0/00v	111071	// // // // // // // // // // // // //	odd]	%odv		
4(%rbp), %eax	-14	(0/	addl	%edx,		je
	cltq	(%rbx,%rax,4)				.L25
	leaq		addq	\$1,		cmpl
	0(,%rax,4), %rdx	%rax				\$2, %eax
	movq -		cmpl	%eax,		movl
32(%rbp), %rax		%r12d				\$3, 8(%r12)
, .,,	addq		jge	.L8		movl
	%rdx, %rax	.L9:				\$4, 8(%rbp)
	movl -		leag			je
4(%rbp), %edx			16(%rsp), %rsi			.L26
¬(∞ιυμ), %eux	add1			0/04:		
	addl	0/odi	xorl	%edi,		cmpl
	\$1,	%edi				\$3, %eax
	movl		call			movl
	%edx, (%rax)		clock_gettime			\$4, 12(%r12)
	movl -		movq			movl
4(%rbp), %eax			24(%rsp), %rax			\$5, 12(%rbp)
	cltq		subq			je
	leag		8(%rsp), %rax			.L27
	0(,%rax,4), %rdx		movl	\$.LC3,		cmpl
	movq -	%edi		Ψ. Εσσ,		\$4, %eax
24/0/rhn\ 0/rc:	•	/UCU1	nvor	0/ymm0		
24(%rbp), %rax		0/ vmmC	pxor	%xmm0,		movl
	addq	%×mm0		.		\$5, 16(%r12)
	%rdx, %rax		pxor	%xmm1,		movl
	movl -	%xmm1				\$6, 16(%rbp)
4(%rbp), %edx			cvtsi2sdq	%rax,		je
	addl	%xmm0				.L28
	\$2, %edx		movq			cmpl
	movl		16(%rsp), %rax			\$6, %eax
	%edx, (%rax)		subq			movl
	addl		(%rsp), %rax			\$6, 20(%r12)
	\$1, -4(%rbp)			%rov		\$0, 20(%112) movl
15.	Ψ±, -4(%ιυμ)	0/2/mm1	cvtsi2sdq	%rax,		
.L5:	1	%xmm1	1			\$7, 20(%rbp)
	movl -		movl	\$1,		jne
4(%rbp), %eax		%eax				.L29
	cmpl -		divsd			movl
12(%rbp), %eax			.LC2(%rip), %xr	nm0		\$7, 24(%r12)
	jle		addsd	%xmm1,		movl
	.L6	%×mm0		,		\$8, 24(%rbp)
	leaq -		call	printf		movl
64/0/rhn\ 0/rc:	-					
64(%rbp), %rax		0/12.2	addq	\$40,	10.	\$7, %ecx
	movq	%rsp			.L8:	1
	%rax, %rsi		.cfi_remember_s			cmpl
	movl		.cfi_def_cfa_o1	rfset		%eax, %ebx
	\$0, %edi	40				je
	call		xorl	%eax,		.L48
	clock_gettime	%eax			.L7:	
	movl		popq	%rbx		movl
	\$1, -4(%rbp)		.cfi_def_cfa_o1			%ebx, %r8d
	jmp	32				leal -
		52	nong	0/rhn	1/0/rhy\ 0/00=	
1.0	.L7		popq	%rbp	1(%rbx), %esi	
.L8:	_		.cfi_def_cfa_of	rrset		movl
	movl -	24				%eax, %edi
4(%rbp), %eax			popq	%r12		subl
		1				

	cltq	16	.cfi_def_cfa_o	offset		%eax, %r8d leal -
	leaq	10		0/10	4 (0(=0) 0(=d)	Tear -
	0(,%rax,4), %rdx		popq		4(%r8), %edx	l- 1
/	movq -		.cfi_def_cfa_o	orrset 8		subl
24(%rbp), %rax			ret			%eax, %esi
	addq	.L21:				shrl
	%rax, %rdx		.cfi_restore_s	state		\$2, %edx
	movl -		movq	%rsp,		addl
4(%rbp), %eax		%rsi				\$1, %edx
	cltq		xorl	%edi,		cmpl
	leaq	%edi				\$2, %esi
	0(,%rax,4), %rcx		call			leal
	movq -		clock_gettime			0(,%rdx,4), %r9d
24(%rbp), %rax	1		jmp	.L9		jbe
(addq	.L20:	3 P			.L10
	%rcx, %rax		movl	\$.LC0,		movl
	movl	%edi		4.200,		%ecx, 12(%rsp)
	(%rax), %ecx	70001	call	puts		leag
	movl -		orl	\$-1,		4(,%rdi,4), %rdi
4 (0/, , , , , ,) 0/, , , , ,	IIIOVI -	0/ = 4 4	011	Φ-1,		
4(%rbp), %eax	-14	%edi	11			xorl
	cltq		call	exit		%eax, %eax
	leaq	.L3:	-			movd
	0(,%rax,4), %rsi		movl	\$.LC1,		12(%rsp), %xmm6
	movq -	%edi				xorl
32(%rbp), %rax			call	puts		%esi, %esi
	addq		orl	\$-1,		movdqa
	%rsi, %rax	%edi				.LC3(%rip), %xmm4
	movl		call	exit		leaq
	(%rax), %eax		.cfi_endproc			(%r12,%rdi), %r10
	imull -	.LFE21:				addq
8(%rbp), %eax			.size			%rbp, %rdi
ο(/oι υμ), /οσαλ	addl		main,main			pshufd
				.rodat		•
	%ecx, %eax	a aata llamii en	.section	. rouat		\$0, %xmm6, %xmm0
	movl	a.cst8,"aM",@p				movdqa
	%eax, (%rdx)		.align 8			.LC4(%rip), %xmm3
	addl	.LC2:	_			movdqa
	\$1, -4(%rbp)		.long	0		.LC5(%rip), %xmm2
.L7:			.long			paddd
	movl -		1104006501			.LC2(%rip), %xmm0
4(%rbp), %eax			.ident	"GCC: .	.L11:	
	cmpl -	(GNU) 6.1.1 20:	160501"			movdqa
12(%rbp), %eax			.section	.note.		%xmm0, %xmm1
	jle	GNU-stack,"",@	progbits			movdqa
	.L8	, , , ,	· ·			%xmm0, %xmm5
	leaq -					paddd
80(%rbp), %rax						%xmm2, %xmm0
(.o. op) / .o. u/	movq					addl
	%rax, %rsi					\$1, %esi
	movl					paddd
						•
	\$0, %edi					%xmm3, %xmm1
	call					movups
	clock_gettime					%xmm0, (%rdi,
	movq -			%	%rax)	
80(%rbp), %rdx						paddd
	movq -					%xmm4, %xmm5
64(%rbp), %rax						movaps
	subq					%xmm1,
	%rax, %rdx			((%r10,%rax)	
	movq			`	,	addq
	%rdx, %rax					\$16, %rax
	pxor					cmpl
	%xmm1, %xmm1					%esi, %edx
	cvtsi2sdq					jbe
	%rax, %xmm1					. L49
	•					
70 (0/mbm) 0/	movq -					movdqa
72(%rbp), %rdx						%xmm5, %xmm0
	movq -					jmp
						144
56(%rbp), %rax						.L11
56(%rbp), %rax	subq			.	.L49:	·LII

%rax, %	Srdx			addl
	, an			
movq				%r9d, %ecx
%rdx, %	rax			cmpl
· ·				
pxor				%r9d, %r8d
%×mmΘ,	%xmm0			je
cvtsi2s	nh:			.L13
	•			. 210
%rax, %	SXMM⊍		.L10:	
movsd				leal
	in) %vmm2			
	ip), %xmm2			1(%rcx), %edx
divsd				leal
%xmm2,	%xmm0			2(%rcx), %eax
	7071111110			
addsd				movslq
%xmm1,	%xmm0			%ecx, %rsi
movsd				cmpl
				•
%×mm0,	-40(%rbp)			%edx, %ebx
movq	· · · · · · ·			movl
40(%rbp), %rax				%edx,
movq			(%r12,%rsi,4)	
·	101(0/rhn)		, , , , , , , , , , , , , , , , , , , ,	mov1
	104(%rbp)			movl
movsd	-			%eax, 0(%rbp,
104(%rbp), %xmm0			%rsi,4)	• • •
			··· · · · · · · · · · · · · · · · · ·	2.1
movl				jl
\$.LC3,	%edi			.L13
movl				leal
\$1, %ea	ιX			3(%rcx), %esi
call				movslq
printf				%edx, %rdx
movl				cmpl
\$0, %ea	ιx			%eax, %ebx
leave				movl
	f cfo 7 0			
	ef_cfa 7, 8		,	%eax,
ret			(%r12,%rdx,4)	
.cfi_er	dnroc			movl
	p. 00			
.LFE2:				%esi, 0(%rbp,
.size			%rdx,4)	
main, .	-main		, ,	jl
				_
.sectio	n .r			.L13
odata				cltq
.align	8			addl
.LC2:				\$4, %ecx
	0			
.long	0			movl
.long				%esi,
1104006	561		(%r12,%rax,4)	÷
			(101 ±2, 101 UA, 4)	
.ident	"G			movl
CC: (GNU) 6.1.1 201605	6 01 "			%ecx, 0(%rbp,
			0/rov 4\	, o(
.sectio			%rax,4)	
ote.GNU-stack,"",@prog	bits		.L13:	
, , , , , , ,				leaq
				16(%rsp), %rsi
				xorl
				%edi, %edi
				call
				clock_gettime
				leaq
				4(%rbp), %rdx
				movq
				%rdx, %rax
				shrq
				•
				\$2, %rax
				negq
				%rax
				andl
				\$3, %eax
				cmpl
				%ebx, %eax
				cmova
				%ebx, %eax
				cmpl
				\$4, %ebx
1		1		

		cmovbe
		%ebx, %eax
	147.	νιστική νιστικ
	.L17:	_
		testl
		%eax, %eax
		je
		.L50
	.L23:	
		imull
		\$3, 4(%r12), %ecx
		addl
		%ecx, (%rdx)
		cmpl
		\$1, %eax
		je
		. L32
		imull
		\$3, 8(%r12), %edx
		addl
		%edx, 8(%rbp)
		cmpl
		\$2, %eax
		je
		.L33
		imull
		\$3, 12(%r12),
	%edx	
		addl
		%edx, 12(%rbp)
		cmpl
		\$4, %eax
		jne
		.L34
		imull
		imull
		imull \$3, 16(%r12),
	%edx	
	%edx	\$3, 16(%r12),
	%edx	\$3, 16(%r12), movl
	%edx	\$3, 16(%r12), movl \$5, %ecx
	%edx	\$3, 16(%r12), movl \$5, %ecx addl
	%edx	\$3, 16(%r12), movl \$5, %ecx addl
		\$3, 16(%r12), movl \$5, %ecx
	%edx .L19:	\$3, 16(%r12), movl \$5, %ecx addl %edx, 16(%rbp)
		\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1
		\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1
		\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx
		\$3, 16(%r12), movl \$5, %ecx addl %edx, 16(%rbp) cmpl %eax, %ebx je
	.L19:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx
		\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22
	.L19:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22
	.L19:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1
	.L19:	\$3, 16(%r12), movl \$5, %ecx addl %edx, 16(%rbp) cmpl %eax, %ebx je .L22 movl %ebx, %r8d
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1
	.L19:	\$3, 16(%r12), movl \$5, %ecx addl %edx, 16(%rbp) cmpl %eax, %ebx je .L22 movl %ebx, %r8d
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 -
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1
	.L19: .L18:	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 mov1 %eax, %edi sub1 %eax, %r8d lea1 -
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 mov1 %eax, %edi sub1 %eax, %r8d lea1 -
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 -
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 -
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 -
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - \$2, %edx
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - \$2, %edx add1
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - \$2, %edx add1 \$1, %edx
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - \$2, %edx add1 \$1, %edx
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1 0(,%rdx,4), %r10d
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1 0(,%rdx,4), %r10d
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1 0(,%rdx,4), %r10d jbe
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1 0(,%rdx,4), %r10d jbe .L21
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1 0(,%rdx,4), %r10d jbe
	.L19: .L18: 1(%rbx), %esi	\$3, 16(%r12), mov1 \$5, %ecx add1 %edx, 16(%rbp) cmp1 %eax, %ebx je .L22 mov1 %ebx, %r8d lea1 - mov1 %eax, %edi sub1 %eax, %r8d lea1 - sub1 %eax, %r8d lea1 - sub1 %eax, %esi shr1 \$2, %edx add1 \$1, %edx cmp1 \$2, %esi lea1 0(,%rdx,4), %r10d jbe .L21

		4(,%rdi,4), %rdi
		xorl
		%eax, %eax
		xorl
		%esi, %esi
		leaq
		0(%rbp,%rdi), %r9
		addq
		%r12, %rdi
	.L15:	
		movdqu
		(%rdi,%rax),
	%xmm1	
	70XIIIII 1	- 443
		addl
		\$1, %esi
		movdqa
		%xmm1, %xmm0
		pslld
		psitu
		\$1, %xmm0
		paddd
		%xmm1, %xmm0
		paddd
		(%r9,%rax), %xmm0
		movaps
		%xmm0, (%r9,%rax)
		addq
		\$16, %rax
		cmpl
		%edx, %esi
		jb
		.L15
		addl
		%r10d, %ecx
		cmpl
		%r10d, %r8d
		je
		.L22
	.L21:	
		movela
		movslq
		%ecx, %rax
		addl
		\$1, %ecx
		imull
		\$3,
	(%r12,%rax,4),	%edx
		addl
		0/
		%edx, 0(%rbp,
	%rax,4)	
	, . ,	cmp.l
		cmpl
		%ecx, %ebx
		jl
) ±
		.L22
		movslq
		%ecx, %rcx
		imull
		\$3,
	(%r12,%rcx,4),	%eax
	•	addl
		%eax, 0(%rbp,
	%rcx,4)	
	.L22:	_
		leaq
		32(%rsp), %rsi
		xorl
		%edi, %edi
		call
		clock_gettime
		movq
		movq 40(%rsp), %rax
		movq 40(%rsp), %rax

		pxor
		%xmm0, %xmm0
		subq
		24(%rsp), %rax
		pxor
		%xmm1, %xmm1
		movl
		\$.LC7, %edi
		cvtsi2sdq
		%rax, %xmm0
		movq
		32(%rsp), %rax
		subq
		16(%rsp), %rax
		10(%1Sp), %1ax
		cvtsi2sdq
		%rax, %xmm1
		movl
		\$1, %eax
		divsd
		.LC6(%rip), %xmm0
		addsd
		%xmm1, %xmm0
		call
		printf
		addq
		\$48, %rsp
		.cfi_remember_sta
	te	
	Le	
		.cfi_def_cfa_offs
	et 32	
	1	
		xorl
		%eax, %eax
		popq
		%rbx
		.cfi_def_cfa_offs
	et 24	
	EL 24	
		popq
		%rbp
		.cfi_def_cfa_offs
	et 16	
		popq
		%r12
		.cfi_def_cfa_offs
	et 8	
	er o	
		ret
	.L50:	
	.L50:	
		.cfi_restore_stat
	.L50: e	.cfi_restore_stat
		.cfi_restore_stat
		.cfi_restore_stat
		.cfi_restore_stat movl \$1, %ecx
		.cfi_restore_stat movl \$1, %ecx jmp
		.cfi_restore_stat movl \$1, %ecx jmp
	е	.cfi_restore_stat movl \$1, %ecx
		.cfi_restore_stat mov1 \$1, %ecx jmp .L18
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax
	е	.cfi_restore_stat mov1 \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax negq
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax negq %rax
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax negq %rax andl
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax negq %rax andl
	е	.cfi_restore_stat movl \$1, %ecx jmp .L18 leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime leaq 4(%rbp), %rdx movq %rdx, %rax shrq \$2, %rax negq %rax

Cmp1			
Mebx, Weax Mebx, Weax Mebx, Weax Mebx, Weax Mebx			cmpl
Cmova Nebx, Neax Cmp1 34, Nebx 34, Nebx 34 34 34 34 34 34 34 3			
Metal Meta			
cmp1 \$4, %ebx ja 1.17 mov1 %ebx, %eax jmp 1.17 mov1 \$1, %eex jmp 1.19 1.28: mov1 \$5, %eex jmp 1.19 1.29: mov1 \$6, %eex jmp 1.29			
S.4, %ebx ja			
1			
1.17			
mov1 %ebx, %eax jmp .123 .124: mov1 .51, %ecx jmp .17 .133: mov1 .53, %ecx jmp .119 .134: mov1 .52, %ecx jmp .119 .128: mov1 .55, %ecx jmp .148 .129: mov1 .56, %ecx jmp .129: .18 .125: mov1 .52, %ecx jmp .125: mov1 .52, %ecx jmp .126: .18 .126: mov1 .127: mov1 .128 .127: mov1 .128 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .127: .18 .128: .128: .129: .146: mov1 .129: .146:			Ja
Sept., %eax jmp			
Imp 1.23			MONT
.L24: mov1 \$1, %ecx jmp .17 .L33: mov1 \$3, %ecx jmp .L19 .L34: mov1 \$4, %ecx jmp .L19 .L32: mov1 \$2, %ecx jmp .L19 .L28: mov1 \$5, %ecx jmp .L19 .L28: mov1 \$5, %ecx jmp .L19 .L29: mov1 \$5, %ecx jmp .L19 .L29: mov1 \$5, %ecx jmp .L19 .L29: mov1 \$6, %ecx jmp .L18 .L27: mov1 \$6, %ecx jmp .L8 .L27: mov1 \$6, %ecx jmp .L8 .L27: mov1 \$6, %ecx jmp .L8 .L26: mov1 \$1, %ecx jmp .L8 .L27: mov1 \$2, %ecx jmp .L8 .L27: mov1 \$3, %ecx jmp .L8 .L26: mov1 \$4, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L26: mov1 \$4, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq leaq lea(s%rsp), %rsi xor1 %eddi, %edi call call clock, gettime jmp .L22			
1.24:			jmp
mov1 \$1, %ecx jmp .17 .133: mov1 \$3, %ecx jmp .119 .134: mov1 \$4, %ecx jmp .119 .132: mov1 \$2, %ecx jmp .119 .128: mov1 \$5, %ecx jmp .119 .128: mov1 \$5, %ecx jmp .18 .129: mov1 \$5, %ecx jmp .18 .129: mov1 \$5, %ecx jmp .18 .129: mov1 \$6, %ecx jmp .18 .147: mov1 \$6, %ecx jmp .18 .18 .18 .18 .18 .18 .18 .18 .18 .18			.L23
\$1, %ecx jmp		.L24:	
Jmp .17 .133:			movl
Jmp .17 .133:			\$1, %ecx
.133: mov1			
.1.33: mov1 S3, %ecx jmp .1.19 .1.34: mov1 S4, %ecx jmp .1.19 .1.28: mov1 S2, %ecx jmp .1.19 .1.28: mov1 S6, %ecx jmp .1.8 .1.29: mov1 S6, %ecx jmp .1.8 .1.25: mov1 S7, %ecx jmp .1.8 .1.26: mov1 S2, %ecx jmp .1.8 .1.27: mov1 S3, %ecx jmp .1.8 .1.26: mov1 S4, %ecx jmp .1.8 .1.27: mov1 S4, %ecx jmp .1.8 .1.27: .1.8 .1.47:			
S3, %ecx jmp .119 .119 .134:		133.	
S3, %ecx jmp		. 2001	mov1
Jmp .134:			
L34: mov %4, %ecx jmp			
1.134:			Juih
mov1 S4, %ecx jmp .i.19 .i.19 .i.19 .i.19 .i.28: mov1 S2, %ecx jmp .i.19 .i.28: mov1 S5, %ecx jmp .i.8 .i.			.L19
\$4, %ecx jmp		.L34:	_
jmp .132:			
jmp .132:			
.L32: mov1 \$2, %ecx jmp .L19 .L28: mov1 \$5, %ecx jmp .L8 .L29: mov1 \$6, %ecx jmp .L8 .L25: mov1 \$2, %ecx jmp .L8 .L26: .L27: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%fsp), %rsi xor1 %edi, %edi call call clock_gettime jmp .L22 .L46:			
L32:			
mov1 \$2, %ecx jmp		.L32:	
\$2, %ecx jmp L19 .L28: mov1 \$5, %ecx jmp L8 .L29: mov1 \$6, %ecx jmp L8 .L25: mov1 \$2, %ecx jmp L8 .L26: mov1 \$3, %ecx jmp L8 .L26: mov1 \$3, %ecx jmp L8 .L27: mov1 \$4, %ecx jmp L8 .L47: lead 16(%rsp), %rsi xor1 %edi, %edi call clock_gettime jmp .L22 .L46: mov1			mov1
Jimp			
.L28: mov1 \$5, %ecx jmp .L8 .L29: mov1 \$6, %ecx jmp .L8 .L25: mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1			
L28: mov1			
mov1 \$5, %ecx jmp .18 .L29: mov1 \$6, %ecx jmp .18 .L25: mov1 \$2, %ecx jmp .18 .L26: mov1 \$3, %ecx jmp .18 .L27: mov1 \$4, %ecx jmp .18 .L27: mov1 \$4, %ecx jmp .18 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi call clock_gettime jmp .L22 .L46: mov1		1.00	. LIA
\$5, %ecx jmp		.L28:	
Jmp			
.L29: mov1 \$6, %ecx jmp .L8 .L25: mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1 %edi wedi cal1 clock_gettime jmp .L22 .L46:			
.L29: mov1 \$6, %ecx jmp .L8 .L25: mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1 %edi wedi cal1 clock_gettime jmp .L22 .L46:			jmp
.L29: mov1			
mov1 \$6, %ecx jmp .L8 .L25: mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi call clock_gettime jmp .L22 .L46: mov1		.L29:	
\$6, %ecx jmp .18 .L25: mov1 \$2, %ecx jmp .18 .L26: mov1 \$3, %ecx jmp .18 .L27: mov1 \$4, %ecx jmp .18 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1			movl
jmp			
.L25: mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi call call clock_gettime jmp .L22 .L46:			
.L25: mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L27: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1			
mov1 \$2, %ecx jmp .L8 .L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi call clock_gettime jmp .L22 .L46: mov1		1.05	. L8
\$2, %ecx jmp .L8 .L26: movl \$3, %ecx jmp .L8 .L27: movl \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl		.L25:	_
jmp			mov1
.L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1 mov1 ### 1.122 ### 1.122 ### 1.122 ### 1.121 ### 1.122 ### 1.121 ### 1.122 ### 1.121 ### 1.122 ### 1.121 ### 1.122 ### 1.121 ### 1.122 ### 1.121 ### 1.122 ### 1.121 ### 1.122 ### 1.122 ### 1.122 ### 1.122 ### 1.122 ### 1.122 ### 1.122			\$2, %ecx
.L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi call clock_gettime jmp .L22 .L46: mov1			jmp
.L26: mov1 \$3, %ecx jmp .L8 .L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi call clock_gettime jmp .L22 .L46: mov1			.L8
movl \$3, %ecx jmp .L8 .L27: movl \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl		.L26:	
\$3, %ecx jmp .L8 .L27: movl \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46:			movl
jmp			
.L27: movl \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl			
.L27: mov1 \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xor1 %edi, %edi cal1 clock_gettime jmp .L22 .L46: mov1			
movl \$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl		127.	. 20
\$4, %ecx jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl		. L ∠ / .	mov1
jmp .L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl			
.L8 .L47: leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46: movl			
leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46:			
leaq 16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46:			.L8
16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46:		.L47:	
16(%rsp), %rsi xorl %edi, %edi call clock_gettime jmp .L22 .L46:			leaq
xorl %edi, %edi call clock_gettime jmp .L22 .L46:			16(%rsp), %rsi
%edi, %edi call clock_gettime jmp .L22 .L46:			xorl
call clock_gettime jmp .L22 .L46: movl			%edi, %edi
clock_gettime jmp .L22 .L46: movl			
jmp .L22 .L46:			clock mettime
.L22 .L46:			
.L46:			Jiiih
movl		1.40.	.L22
movl \$.LC0, %edi		.L4b:	,
\$.LC0, %edi			movT
			\$.LCO, %edi

	call
	puts
	orl \$
	1, %edi
	call
	Call
	exit
	.L3:
	movl
	\$.LC1, %edi
	call
	puts
	orl \$
	1, %edi
	call
	exit
	.cfi_endproc
	.LFE21:
	.size
	main,main
	atii, atii
	.section .
	odata.cst16,"aM",@progbits,16
	.align 16
	.LC2:
	.long 0
	.long 1
	long
	.long 2
	.long 3
	.align 16
	.LC3:
	.long 4
	.long 4
	.long 4
	.align 16
	.LC4:
	.long 1
	.long 1
	.long 1
	.long 1
	.align 16
	.LC5:
	.long 2
	section .
	odata.cst8,"aM",@progbits,8
	.align 8
	.LC6:
	.long 0
	.long
	1104006501
	.ident "0
	CC: (GNU) 6.1.1 20160501"
	.section
	ote.GNU-stack,"",@progbits
I	Carollo Caroll, Epi Ogbico